การวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้

LAGANGAIN RELAXATION COMBINED WITH SELF-ADAPTIVE DIFFERENTIAL EVOLUTION ALGORITHM FOR UNIT COMMITMENT

นางสาวกฤษณา ผนึกดี

นายชนกานต์ บัวบาน

นางสาวชุติภา เรืองจิตร

โครงงานวิศวกรรมนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

วิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า แขนงวิชาวิศวกรรมไฟฟ้ากำลัง

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ

ปีการศึกษา 2557

การวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้

LAGANGAIN RELAXATION COMBINED WITH SELF-ADAPTIVE DIFFERENTIAL EVOLUTION ALGORITHM FOR UNIT COMMITMENT

นางสาวกฤษณา ผนึกดี

นายชนกานต์ บัวบาน

นางสาวชุติภา เรืองจิตร

โครงงานวิศวกรรมนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

วิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า แขนงวิชาวิศวกรรมไฟฟ้ากำลัง

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ

ปีการศึกษา 2557

หัวข้อโครงงานวิศวกรรม สาขาวิศวกรรมไฟฟ้า

เรื่อง การวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้

โดย นางสาวกฤษณา ผนึกดี

นายชนกานต์ บัวบาน

นางสาวชุติภา เรืองจิตร

ภาควิชา วิศวกรรมไฟฟ้า

อาจารย์ที่ปรึกษา อาจารย์ ดร. ธนาธิป สุ่มอิ่ม

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ อนุมัติให้นับโครงงานวิศวกรรมไฟฟ้า เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์

(รองศาสตราจารย์ ดร.เวคิน ปิยรัตน์)

คณะกรรมการสอบโครงงานวิศวกรรม

..... ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ปฐมทัศน์ จิระเดชะ)

..... กรรมการ

(อาจารย์ ดร.ธนาธิป สุ่มอิ่ม)

..... กรรมการ

(อาจารย์ ดร.คมกฤษ ประเสริฐวงษ์)

การวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้
ปีการศึกษา 2557

โดย **อาจารย์ที่ปรึกษา**

นางสาวกฤษณา ผนึกดี ดร. ธนาธิป สุ่มอิ่ม
นายชนกานต์ บัวบาน
นางสาวชุติภา เรืองจิตร

บทคัดย่อ

โครงงานวิศวกรรมนี้มีวัตถุประสงค์เพื่อแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้ โดยสร้างเป็นหน้าต่างโปรแกรมเพื่อให้สะดวกต่อการใช้งาน และเพื่อให้ได้ประสิทธิภาพสูงสุด สำหรับปัญหาการวางแผนเดินเครื่องโรงไฟฟ้านี้ต้องพิจารณาเงื่อนไขและข้อจำกัดต่างๆ ของเครื่องกำเนิดไฟฟ้า ดังต่อไปนี้ คือ กำลังไฟฟ้าสมดุล กำลังการผลิตสำรอง เงื่อนไขของเครื่องกำเนิดไฟฟ้า เวลาเดินเครื่องกำเนิดไฟฟ้าอย่างน้อยที่สุด และเวลาหยุดเดินเครื่องกำเนิดไฟฟ้าอย่างน้อยที่สุด ส่วนเงื่อนไขการสูญเสียในสายส่งจะไม่ถูกนำมาคิดในโครงงานวิศวกรรมนี้ สำหรับโครงงานวิศวกรรมนี้ระบบไฟฟ้าซึ่งมีเครื่องกำเนิดไฟฟ้าจำนวน 10 ยูนิต ระยะเวลาการวางแผนการผลิตไฟฟ้าจำนวน 24 ชั่วโมง จะถูกนำมาทดสอบโดยใช้กับโปรแกรมสำหรับการแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้าที่สร้างขึ้น และนำผลลัพธ์ที่ได้รับไปเปรียบเทียบกับผลลัพธ์ที่ได้จากการแก้ไขปัญหาด้วยวิธี Lagrangian Relaxation, Genetic Algorithm, Lagrangian Relaxation and Genetic Algorithm, Evolutionary Programming, Ant Colony Search Algorithm และLagrangian Relaxation with a Differential Evolution Algorithm จากผลการทดสอบสามารถสรุปได้ว่าวิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้ สามารถแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้าได้อย่างมีประสิทธิภาพ เป็นที่ยอมรับได้

Lagrangian Relaxation with Self-Adaptive Differential Evolution Algorithm for Unit Commitment

Academic Year 2014

By **Advisor**

Ms. Kritsana Panuekdee — Dr. Thanathip Sum-Im

Mr. Chanakan Buaban

Ms. Chutipa Reungjit

ABSTRACT

In this project, the Lagrangian Relaxation (LR) combined with Self-Adaptive Differential Evolution Algorithm (SaDEA) method (LR-SaDEA) is proposed for solving unit commitment (UC) problem. A graphical user interface (GUI) is applied to this program for easy to use. The unit commitment problem is formulated as the minimization of a performance index, and consists of several equality and inequality constraints that are power balance, spinning reserve, generator limits, minimum up time and minimum down time. However, transmission line losses are not considered in this research. In this project, a Lagrangian Relaxation with self-adaptive differential evolution algorithm for unit commitment problem is demonstrated via the analysis of 10 units test system with 24 working hours. A detailed comparative study among the Lagrangian Relaxation (LR), genetic algorithm (GA), Lagrangian Relaxation and genetic algorithm (LRGA), evolutionary programming (EP), ant colony search algorithm (ACSA), Lagrangian Relaxation with a differential evolution algorithm (LR-DEA) and Lagrangian Relaxation with self-adaptive differential evolution algorithm method (LR-SaDEA) is presented. From the experimental results, the Lagrangian Relaxation with self-adaptive differential evolution algorithm method has high accuracy of solution achievement.

กิตติกรรมประกาศ

คณะผู้จัดทำขอกราบขอบพระคุณ ดร.ธนาธิป สุ่มอิ่ม อาจารย์ที่ปรึกษาโครงงานวิศวกรรม และกรรมการสอบโครงงานวิศวกรรม ผู้ให้คำปรึกษาในการค้นคว้าในงานวิจัยและในด้านต่างๆ ตลอดจนตรวจแก้ไขปริญญานิพนธ์จนกระทั่งเสร็จสมบูรณ์ และขอกราบขอบพระคุณ ผศ.ดร.ปฐมทัศน์ จิระเดชะ ประธานกรรมการสอบโครงงานวิศวกรรม ดร.คมกฤษ ประเสริฐวงษ์ กรรมการสอบโครงงานวิศวกรรมที่ให้คำแนะนำในการแก้ไขปริญญานิพนธ์ให้มีความถูกต้องสมบูรณ์

ขอกราบขอบพระคุณอาจารย์ภาควิชาวิศวกรรมไฟฟ้าทุกท่านที่ได้อบรมสั่งสอนและมอบความรู้อันมีคุณค่าเพื่อนำมาประยุกต์ใช้ในงานวิจัยนี้

ขอกราบขอบพระคุณพ่อ คุณแม่ที่ได้อบรมสั่งสอน เป็นกำลังใจและสนับสนุนในทุกๆด้าน และขอขอบคุณทุกท่านที่มีส่วนเกี่ยวข้องกับงานวิจัยในครั้งนี้

สารบัญ

สารบัญ(ต่อ)

สารบัญตาราง

ตารางที่ **หน้าที่**

สารบัญภาพ

ประมวลคำศัพท์และคำย่อ

คำย่อ	คำเต็ม
EAs	Evolutionary Algorithms
DEA	Differential Evolution Algorithm
UC	Unit Commitment
GENCOs	Generation Companies
ED	Economic Dispatch
IP	Integer Programming
DP	Dynamic Programming
LR	Lagrangian Relaxation
HSC	Hot-Start Cost
CSC	Cold-Start Cost
MUT	Minimum Up Time
MDT	Minimum Down Time
GAs	Genetic Algorithms
EP	Evolutionary Programming
ESs	Evolution Strategies
CGA	Conventional Genetic Algorithm
SD	Standard Deviation
ACSA	Ant Colony Search Algorithm
LRGA	Lagrangian Relaxation and Genetic Algorithm
LR-DEA	Lagrangian Relaxation with a Differential Evolution Algorithm
SaDEA	Self-Adaptive Differential Evolution Algorithm

รายการสัญลักษณ์

สัญลักษณ์	ความหมาย	หน่วย
T	จำนวนชั่วโมงที่พิจารณา	-
N	จำนวนเครื่องกำเนิดไฟฟ้า	-
U_{it}	สถานะของเครื่องกำเนิดไฟฟ้า i ณ เวลาที่ t เมื่อ $U_{it} = 0$ แทนสถานะหยุดเดินเครื่องกำเนิดไฟฟ้า และ $U_{it} = 1$ แทนสถานะเดินเครื่องกำเนิดไฟฟ้า	-
$F_i(P_{it})$	ฟังก์ชั่นต้นทุนเชื้อเพลิงของเครื่องกำเนิดไฟฟ้า i ขณะผลิตกำลังไฟฟ้าขนาด P ที่เวลา t	-
ST_{it}	ต้นทุนการเริ่มเดินเครื่องกำเนิดไฟฟ้า i	$
P_{it}	กำลังไฟฟ้าที่ผลิตจากเครื่องกำเนิดไฟฟ้า i ณ เวลาที่ t	MW
P_{Dt}	ค่ากำลังไฟฟ้าที่โหลดต้องการ ณ เวลาที่ t	MW
R_t	ค่ากำลังไฟฟ้าสำรอง ณ เวลาที่ t	MW
$P_{i,max}$	ค่ากำลังไฟฟ้าสูงสุดที่เครื่องกำเนิดไฟฟ้า i สามารถผลิตได้	MW
$P_{i,min}$	กำลังผลิตต่ำสุดของเครื่องกำเนิดไฟฟ้า i	MW
P_i	กำลังไฟฟ้าของเครื่องกำเนิดไฟฟ้า i	MW
$GONT_{i(t-1)}$	จำนวนชั่วโมงที่เครื่องกำเนิดไฟฟ้า i เดินเครื่องเป็นเวลาติดต่อกัน $(t-1)$	-
MTU_i	เวลาเดินเครื่องอย่างน้อยที่สุดของเครื่องกำเนิดไฟฟ้า i	-
$GOFFT_{i(t-1)}$	ระยะเวลาที่เครื่องกำเนิดไฟฟ้า i หยุดเดินเครื่องเป็นเวลาติดต่อกัน $(t-1)$	-
MDU_i	เวลาหยุดเดินเครื่องอย่างน้อยที่สุดของเครื่องกำเนิดไฟฟ้า i	-
P_{Total}	ต้นทุนรวมในการผลิตกำลังไฟฟ้า	$/h
P_{load}	กำลังไฟฟ้าที่โหลดต้องการ	MW
P_{loss}	กำลังสูญเสียในสายส่ง	MW

บทที่ 1

บทนำ

1.1 ความเป็นมาของโครงงาน

ในปัจจุบันพลังงานไฟฟ้าเป็นปัจจัยสำคัญที่เข้ามามีบทบาทในการดำรงชีวิตของมนุษย์ พลังงานไฟฟ้าเป็นตัวแปรสำคัญในการนำมาใช้งาน ด้านโทรคมนาคม ด้านการศึกษา รวมทั้งด้านธุรกิจ และอุตสาหกรรม เช่น ระบบแสงสว่าง ระบบจราจร เครื่องจักรกลในโรงงานอุตสาหกรรม เป็นต้น เนื่องจากในปัจจุบันการขยายตัวของประชากร และระบบเศรษฐกิจมีการขยายตัวอยู่ตลอดเวลา ส่งผลให้อัตราความต้อการในการใช้พลังงานไฟฟ้าเพิ่มมากขึ้น ซึ่งในแต่ละช่วงเวลามีปริมาณความต้องการใช้พลังงานไฟฟ้าไม่เท่ากัน ขึ้นอยู่กับการใช้งานในแต่ละหน่วยงาน หรือบุคคล ดังนั้นในการผลิตพลังงานไฟฟ้าจึงจำเป็นต้องมีการจัดการวางแผนในการผลิตกำลังไฟฟ้าให้มีความเพียงพอต่อความต้องการของผู้บริโภค โดยคำนึงถึงเงื่อนไขข้อจำกัดด้านการผลิต การส่งจ่าย และด้านความคุ้มค่าในทางเศรษฐศาสตร์[1] ซึ่งเรียกกระบวนการนี้ว่า การวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้า (Unit Commitment)

การวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้า (Unit Commitment) มีจุดมุ่งหมายในการวางแผนเดินเครื่องโรงไฟฟ้าล่วงหน้าเพื่อให้เพียงพอต่อความต้องการ และเกิดประสิทธิภาพสูงที่สุด โดยตรงตามเงื่อนไขข้อจำกัดของเครื่องกำเนิดไฟฟ้าและการส่งผ่านกำลังไฟฟ้า ซึ่งโดยทั่วไปแล้ว การวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้า จะสามารถวางแผนได้ตั้งแต่ 1 วันจนถึง 1 สัปดาห์ และในแต่ละชั่วโมงสามารถกำหนดการทำงานของเครื่องกำเนิดไฟฟ้าได้ว่าจะกำหนดให้ตัวไหนทำงานหรือไม่ทำงาน การวางแผนนี้ต้องคำนึงถึงตัวแปรต่างๆของเครื่องกำเนิดไฟฟ้าแต่ละเครื่อง (เช่น เวลาในการทำงานของเครื่องกำเนิดไฟฟ้าที่น้อยที่สุด เวลาในการหยุดการทำงานของเครื่องกำเนิดไฟฟ้าที่น้อยที่สุด เป็นต้น) แต่ปริมาณในการผลิตไฟฟ้าจะสามารถกำหนดล่วงหน้าได้ 5 นาทีก่อนส่งจ่าย การกำหนดปริมาณในการผลิตไฟฟ้านั้นเป็นการส่งจ่ายไฟฟ้าอย่างประหยัด มันคือการวางแผนผลิตกำลังไฟฟ้าในแต่ละช่วงเวลาเพื่อให้เพียงพอต่อความต้องการโดยมีต้นทุนในการผลิตที่ต่ำที่สุด(A. Lelic, 2012)

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาทฤษฎีการวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้า (Unit Commitment)

1.2.2 เพื่อศึกษาวิธีการทำงานของดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับค่าได้ (Self-Adaptive Differential Evolution Algorithm)

1.2.3 สามารถนำวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้มาประยุกต์ใช้ในการแก้ไขปัญหา และคำนวณ การวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้า

1.2.4 เพื่อศึกษา และ ออกแบบการเขียนโปรแกรม MATLAB โดยการนำวิธีการดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับค่าได้ (Self-Adaptive Differential Evolution Algorithm) มาประยุกต์ใช้ในการแก้ไขปัญหา และคำนวณ การวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้าให้มีค่าเหมาะสมในเงื่อนไขด้านความคุ้มค่าทางด้านเศรษฐศาสตร์ [1] และการผลิตกำลังไฟฟ้าเพียงพอต่อความต้องการของผู้บริโภค

1.2.5 ศึกษาและออกแบบการสร้าง Graphical User Interface (GUI) ใน MATLAB จนกระทั่งสามารถนำมาประยุกต์ใช้งาน เป็นหน้าต่างในการคำนวณเพื่อแก้ไขปัญหา และคำนวณ การวางแผนเดินเครื่องกำเนิดไฟฟ้าของระบบผลิตไฟฟ้า

1.3 ขอบเขตของโครงงาน

ออกแบบ และสร้างโปรแกรมสำหรับแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้าโดยนำวิธีการดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมมาประยุกต์ใช้โดยมีคุณสมบัติดังต่อไปนี้

1.3.1 ออกแบบ และทดสอบโดยใช้โปรแกรม MATLAB

1.3.2 สร้างหน้าต่างโปรแกรมที่สามารถแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้าที่เป็นแบบดั้งเดิม (Traditional Unit Commitment) โดยพิจารณาถึงเงื่อนไข และข้อจำกัดต่างๆ ดังนี้

1.3.2.1 กำลังไฟฟ้าสมดุล (Power Balance)

1.3.2.2 ข้อจำกัดของเครื่องกำเนิดไฟฟ้า (Generation Limit)

1.3.2.3 เวลาเดินเครื่องกำเนิดไฟฟ้าอย่างน้อยที่สุด (Minimum Up Time)

1.3.2.4 เวลาหยุดเดินเครื่องกำเนิดไฟฟ้าอย่างน้องที่สุด (Minimum Down Time)

1.3.2.5 กำลังไฟฟ้าสำรอง (Spinning Reserve) สำหรับเงื่อนไขการสูญเสียในสายส่งจะไม่ถูกพิจารณาในโครงงานนี้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 ได้รับความรู้เกี่ยวกับทฤษฎีการวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้า (Unit Commitment)

1.4.2 สามารถนำวิธีการของ Self-Adaptive Differential Evolution Algorithm มาประยุกต์ใช้ในการแก้ไขปัญหา และคำนวณ การวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้า ให้มีค่าเหมาะสมในเงื่อนไขด้านความคุ้มค่าทางด้านเศรษฐศาสตร์ และการผลิตกำลังไฟฟ้าเพียงพอต่อความต้องการของผู้บริโภค

1.4.3 สามารถนำความรู้ และ วิธีการทำงานของดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับค่าได้ (Self-Adaptive Differential Evolution Algorithm) มาประยุกต์ใช้ในการแก้ไขปัญหาทางวิศวกรรมต่างๆได้

1.4.4 สามารถนำความรู้จากการศึกษาและออกแบบการเขียน Graphical User Interface (GUI) ใน MATLAB จนกระทั่งสามารถนำมาประยุกต์ใช้งาน เป็นหน้าต่างในการทำงานของโปรแกรมสำหรับใช้แก้ไขปัญหา เพื่อความสะดวกต่อผู้ใช้งานได้มากขึ้น

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

ปัจจุบันในแต่ละภูมิภาคมีการส่งจ่ายกำลังไฟฟ้าหลายร้อยกิกะวัตต์ให้กับผู้บริโภค ซึ่งในแต่ละช่วงเวลาของวัน ความต้องการในการใช้ไฟฟ้านั้นไม่เท่ากัน จึงต้องมีการวางแผนการส่งจ่ายกำลังไฟฟ้า เพื่อให้มั่นใจว่าระบบจะยังคงทำงานได้อย่างมีประสิทธิภาพ ในแต่ละวันจะมีการกำหนดสถานะการเดินเครื่องกำเนิดไฟฟ้าแต่ละเครื่องว่ามีเครื่องใดติดเครื่องใดดับบ้าง เรียกว่า การวางแผนเดินเครื่องโรงไฟฟ้า (Unit Commitment: UC) ปัญหาส่วนใหญ่ คือ การกำหนดสถานะการทำงานของเครื่องกำเนิดไฟฟ้าจำนวนมาก โดยมีข้อจำกัดที่ซับซ้อน ดังนั้น การศึกษาการวางแผนเดินเครื่องโรงไฟฟ้า (Unit Commitment: UC) จึงกลายเป็นสิ่งที่สำคัญ ในบทความนี้มีวัตถุประสงค์เพื่อให้เข้าใจหลักการของการวางแผนเดินเครื่องโรงไฟฟ้า จากอดีตจนถึงปัจจุบัน

ในส่วนที่ 2 จะเป็นการอธิบายหลักการของการวางแผนเดินเครื่องโรงไฟฟ้า และในส่วนที่ 3 จะมุ่งเน้นไปที่ปัญหาการกำหนดสถานการณ์ทำงานของเครื่องกำเนิดไฟฟ้า และเปรียบเทียบการแก้ปัญหา 2 วิธี นอกจากนี้ยังมีรายละเอียดอื่นๆ ที่เกี่ยวข้องอีกด้วย

2.1 ปัญหาการวางแผนเดินเครื่องโรงไฟฟ้า (Unit Commitment: UC)

การวางแผนเดินเครื่องโรงไฟฟ้ามีวัตถุประสงค์เพื่อ จัดตารางสถานการณ์ติด – ดับของเครื่องกำเนิดไฟฟ้าทั้งหมดให้ได้ประสิทธิภาพที่สูงที่สุด เพื่อให้เพียงพอต่อความต้องการของโหลด และความต้องการสำรอง โดยมีข้อจำกัดของเครื่องกำเนิดไฟฟ้า และการส่งจ่ายด้วย โดยทั่วไปแล้วการวางแผนเดินเครื่องโรงไฟฟ้าจะวางแผนสำหรับระยะเวลา 1 วัน จนถึง 1 สัปดาห์ และในแต่ละชั่วโมงมีเครื่องกำเนิดไฟฟ้าใดติด – ดับบ้าง โดยการกำหนดสถานะของเครื่องกำเนิดไฟฟ้านั้นจะต้องพิจารณาคุณสมบัติของเครื่องกำเนิดไฟฟ้าด้วย เช่น เวลาเดินเครื่องกำเนิดไฟฟ้าอย่างน้อยที่สุด (Minimum Up Time) เวลาหยุดเดินเครื่องกำเนิดไฟฟ้าอย่างน้อยที่สุด (Minimum Down Time) กำลังไฟฟ้าสมดุล (Power Balance) ข้อจำกัดของเครื่องกำเนิดไฟฟ้า (Generator Limit) กำลังไฟฟ้าสำรอง (Spinning Reserve) กำลังไฟฟ้าสูญเสียเนื่องจากสายส่ง (Transmission Loss) เป็นต้น การแก้ปัญหาเพื่อให้ต้นทุนการผลิตไฟฟ้ารวม ณ ช่วงเวลาใดเวลาหนึ่ง มีราคาที่ต่ำที่สุด และโหลด ณ ช่วงเวลานั้นได้รับกำลังไฟฟ้าที่เพียงพอกับความต้องการ การแก้ปัญหาแบบนี้เราเรียกว่า การแก้ปัญหาการจ่ายโหลดอย่างประหยัด (Economic Dispatch: ED)

2.2 วิธีการแก้ไขปัญหา

การแก้ปัญหาการจ่ายโหลดอย่างประหยัด (Economic Dispatch: ED) มีวัตถุประสงค์เพื่อลดค่าใช้จ่ายของระบบทั้งหมดให้มีค่าต่ำที่สุด โดยมีเครื่องกำเนิดไฟฟ้าจำนวน (N) เครื่อง ในจำนวนชั่วโมงที่พิจารณา (T) ต้นทุนทั้งหมดของเครื่องกำเนิดไฟฟ้าแต่ละเครื่อง คือ ค่าเชื้อเพลิงที่ใช้ในช่วงเวลาที่พิจารณา (F_i) รวมกับค่าใช้จ่ายในการเริ่มเดินเครื่องกำเนิดไฟฟ้า (Start-Up Cost: ST_{it})

$$F_{Total\ Cost} = \sum_{t=1}^{T}\sum_{i=1}^{N}\left[F_i(P_{it}) + \left(1 - U_{i,t-1}\right)ST_{it}\right]U_{it} \qquad (2.1)$$

เมื่อ T คือ จำนวนชั่วโมงที่พิจารณา

N คือ จำนวนเครื่องกำเนิดไฟฟ้า

U_{it} คือ สถานะของเครื่องกำเนิดไฟฟ้า i ณ เวลาที่ t

เมื่อ $U_{it} = 0$ แทนสถานะหยุดเดินเครื่องกำเนิดไฟฟ้า และ

$U_{it} = 1$ แทนสถานะเดินเครื่องกำเนิดไฟฟ้า

$F_i(P_{it})$ คือ ฟังก์ชันต้นทุนเชื้อเพลิงของเครื่องกำเนิดไฟฟ้า i ขณะผลิตกำลังไฟฟ้าขนาด P ที่เวลา t

ST_{it} คือ ต้นทุนการเริ่มเดินเครื่องกำเนิดไฟฟ้า i

P_{it} คือ กำลังไฟฟ้าที่ผลิตจากเครื่องกำเนิดไฟฟ้า i ณ เวลาที่ t

2.2.1 เงื่อนไขของระบบ

การแก้ปัญหาการวางแผนเดินเครื่องโรงไฟฟ้ามีวัตถุประสงค์เพื่อลดต้นทุนในการผลิตไฟฟ้าให้ต่ำที่สุด ซึ่งแผนการเดินเครื่องกำเนิดไฟฟ้าต้องสอดคล้องกับเงื่อนไขของระบบไฟฟ้าและเครื่องกำเนิดไฟฟ้าด้วย โดยพิจารณาเงื่อนไขดังต่อไปนี้

2.2.1.1 กำลังไฟฟ้าสมดุล (Power Balance) เงื่อนไข คือ กำลังผลิตไฟฟ้ารวมของเครื่องกำเนิดไฟฟ้าเข้ากับระบบไฟฟ้า ณ เวลาที่พิจารณา ต้องมีค่าเท่ากับค่ากำลังไฟฟ้าที่โหลดต้องการ ณ เวลานั้น

$$\sum_{i=1}^{N} U_{it} P_{it} = P_{Dt} \qquad (2.2)$$

เมื่อ P_{Dt} คือ ค่ากำลังไฟฟ้าที่โหลดต้องการ ณ เวลาที่ t

2.2.1.2 กำลังการผลิตสำรอง (Spinning Reserve)

กำลังการผลิตไฟฟ้าสำรองมีจุดมุ่งหมายเพื่อเป็นหลักประกันว่า เมื่อเกิดการเปลี่ยนแปลงของระบบไฟฟ้า ระบบจะยังคงจ่ายไฟฟ้าต่อไปได้ ซึ่งการเปลี่ยนแปลงมี 2 กรณีคือ

2.2.1.2.1 การปลดเครื่องกำเนิดไฟฟ้าเครื่องใดเครื่องหนึ่ง เนื่องจากสาเหตุใดก็ตาม โหลดในระบบไฟฟ้าจะยังคงได้รับกำลังไฟฟ้าที่เพียงพอกับความต้องการเหมือนเดิม

2.2.1.2.1 ความต้องการของโหลดมากขึ้น เครื่องกำเนิดไฟฟ้าที่ต่ออยู่กับระบบไฟฟ้าขณะนั้น ต้องสามารถเพิ่มกำลังการผลิตไฟฟ้าได้เพียงพอกับความต้องการของโหลดที่เพิ่มขึ้น

$$\sum_{i=1}^{N} U_{it} P_{i,max} \geq R_t + P_{Dt} \qquad (2.3)$$

เมื่อ R_t คือ ค่ากำลังไฟฟ้าสำรอง ณ เวลาที่ t

$P_{i,max}$ คือ ค่ากำลังไฟฟ้าสูงสุดที่เครื่องกำเนิดไฟฟ้า i สามารถผลิตได้

2.2.2 เงื่อนไขของเครื่องกำเนิดไฟฟ้า (Generator Constrains)

ตั้งแต่อดีตจนถึงปัจจุบัน เครื่องกำเนิดไฟฟ้าถูกใช้ในการผลิตกำลังไฟฟ้ากันอย่างแพร่หลาย โดยเครื่องกำเนิดไฟฟ้านั้นมีหลายชนิด เช่น โรงไฟฟ้าพลังร่วมกังหันก๊าซ (Combined cycle gas turbine), ถ่านหิน (coal-fired), นิวเคลียร์ (nuclear) เป็นต้น ซึ่งในแต่ละชนิดมีการทำงานที่แตกต่างกัน มีลักษณะทางกายภาพที่แตกต่างกัน

เครื่องกำเนิดไฟฟ้าแต่ละเครื่องจะมีขีดจำกัดในการผลิตกำลังไฟฟ้า ว่าเครื่องกำเนิดไฟฟ้านั้นสามารถผลิตกำลังไฟฟ้าได้สูงที่สุดและต่ำที่สุดเท่าใด ซึ่งการผลิตกำลังไฟฟ้าในแต่ละช่วงเวลานั้น ต้องคำนึงถึงขีดจำกัดของการผลิตกำลังไฟฟ้าของเครื่องกำเนิดไฟฟ้าแต่ละเครื่องด้วย

$$P_{i,min} \leq P_i \leq P_{i,max} \qquad (2.4)$$

เมื่อ $P_{i,min}$ คือ กำลังผลิตต่ำสุดของเครื่องกำเนิดไฟฟ้า i (MW)

P_i คือ กำลังไฟฟ้าของเครื่องกำเนิดไฟฟ้า i (MW)

$P_{i,max}$ คือ ค่ากำลังไฟฟ้าสูงสุดที่เครื่องกำเนิดไฟฟ้า i สามารถผลิตได้ (MW)

โดยปกติแล้วเครื่องกำเนิดไฟฟ้าไม่สามารถสั่งเดินเครื่อง หรือดับเครื่องได้ทันที จะต้องมีช่วงระยะเวลาหนึ่ง ซึ่งปกติอาจใช้เวลาหลายชั่วโมง และระยะเวลาดังกล่าวของแต่ละเครื่องมีค่าไม่เท่ากัน ซึ่งก็คือ เวลาเดินเครื่องกำเนิดไฟฟ้าอย่างน้อยที่สุด (Minimum Up Time: T_i^{up}) และเวลาหยุดเดินเครื่องกำเนิดไฟฟ้าอย่างน้อยที่สุด (Minimum Down Time: T_i^{down})

Minimum Up Time: T_i^{up}

$$U_{it} = 1 \quad \text{for} \quad \sum_{h=t-T_i^{up}}^{t-1} U_{it} < T_i^{up} \tag{2.5}$$

Minimum Down Time: T_i^{down}

$$U_{it} = 0 \quad \text{for} \quad \sum_{h=t-T_i^{down}}^{t-1} (1 - U_{it}) < T_i^{down} \tag{2.6}$$

2.2.3 วิธีลากรางจ์รีแล็กเซชัน (Lagrangian Relaxation: LR)

จากเงื่อนไขต่างๆ เราสามารถเขียนสมการ Lagrangian ได้ดังนี้

$$L(P, U, \lambda) = F_{Total\ Cost} + \sum_{t=1}^{T} \lambda_t \left(P_{Dt} - \sum_{i=1}^{N} U_{it} P_{it} \right)$$

หรือ

$$L(P, U, \lambda) = \sum_{t=1}^{T} \sum_{i=1}^{N} [F_i(P_{it}) + (1 - U_{i,t-1}) ST_{it}] \cdot U_{it} + \sum_{t=1}^{T} \lambda_t \left(P_{Dt} - \sum_{i=1}^{N} U_{it} P_{it} \right) \tag{2.7}$$

วิธีลากรางจ์รีแล็กเซชัน (Lagrangian Relaxation, LR) ที่ใช้ในการแก้ปัญหาการวางแผนเดินเครื่องโรงไฟฟ้านั้นเป็นวิธีที่อาศัยหลักการ Dual Optimization ในการหาคำตอบของปัญหา หลักการนี้จะแก้ปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าโดยการไม่คำนึงถึงเงื่อนไขของปัญหาเป็นเวลาชั่วคราว (Relaxing) และคำนวณปัญหาเสมือนว่าไม่มีเงื่อนไขอยู่ในปัญหา การคำนวณหาค่าที่เหมาะสมที่สุด (Optimization) ของปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าจากหลักการ Dual Optimization นั้นทำได้โดย การหาค่าสูงสุด (Maximize) ของฟังก์ชันลากรางจ์โดยพิจารณาตัวคูณลากรางจ์ (Lagrange Multiplier, λ) พร้อมกันกับหาค่าต่ำสุด (Minimize) ของฟังก์ชันลากรางจ์โดยพิจารณาตัวแปรที่ต้องการหาคำตอบของปัญหาในขณะที่กำหนดให้ตัวคูณลากรางจ์มีค่าคงที่ นั่นคือ

$$q^*(\lambda) = \max q(\lambda) \tag{2.8}$$

เมื่อ $$q(\lambda) = \min L(P, U, \lambda) \quad (2.9)$$

ขั้นตอนหลักๆของวิธีการนี้ประกอบด้วย 2 ขั้นตอน คือ

1. ทำการหาค่าตัวคูณลากรางจ์ที่สามารถทำให้ $q(\lambda)$ มีค่ามากขึ้น

2. สมมติว่าตัวคูณลากรางจ์ที่ได้จากขั้นตอนที่ 1 มีค่าคงที่ จากนั้นทำการหาค่าต่ำสุดของฟังก์ชันลากรางจ์ (L) โดยพิจารณาตัวแปร P และ U

เมื่อสิ้นสุดตามขั้นตอนที่กล่าวมาข้างต้น จะได้ผลลัพธ์ที่เรียกว่า Dual Solution (q^*) ซึ่งเป็นผลของ ที่ได้ในแต่ละรอบการคำนวณ

$$q^*(\lambda) = \sum_{t=1}^{T}\sum_{i=1}^{N}[F_i(P_{it}) - \lambda_t P_{it}] \cdot U_{it} + ST_{it}U_{it} + \lambda_t P_{Dt} \quad (2.10)$$

ผลลัพธ์ของ Dual solution (q^*) นี้ไม่ใช่คำตอบที่แท้จริงของปัญหาการวางแผนเดินเครื่องโรงไฟฟ้า และขั้นตอนต่อไปหลังจากได้ Dual Solution แล้ว จะทำการคำนวณต่อไปโดยพิจารณาเฉพาะเครื่องกำเนิดไฟฟ้าที่มีสถานะเปิด $(U_i = 1)$ ในผลของ Dual Solution โดยการนำเฉพาะเครื่องกำเนิดไฟฟ้าที่มีสถานะเปิดเหล่านี้มาคำนวณการแก้ปัญหาการจ่ายโหลดอย่างประหยัด (Economic Dispatch) สำหรับแต่ละชั่วโมงในช่วงเวลาทั้งหมดที่พิจารณา และผลลัพธ์ที่ได้แต่ละชั่วโมงในขั้นตอนนี้เมื่อนำมารวมกันแล้วจะถูกเรียกว่า Primal Solution (J^*)

$$J^* = \sum_{t=1}^{T}\sum_{i=1}^{N}\left[F_i(P_{it}) + \left(1 - U_{i,t-1}\right)ST_{it}\right]U_{it} \quad (2.11)$$

เมื่อได้ Primal Solution แล้ว ขั้นตอนต่อไปคือการค่าตัวคูณลากรางจ์ตัวใหม่ที่สามารถทำให้ $q(\lambda)$ ซึ่งก็คือ q^* ของแต่ละรอบการคำนวณมีค่ามากขึ้นกว่าเดิม การหาค่าตัวคูณลากรางจ์ตัวใหม่โดยทั่วไปจะอาศัยหลักการกราเดียนท์ในการปรับเปลี่ยนค่าของตัวคูณลากรางจ์ ซึ่งความสัมพันธ์ที่ใช้ในการหาค่าตัวคูณลากรางจ์ตัวใหม่เป็นดังสมการที่ 2.12

$$\lambda_t^{iter+1} = \lambda_i^{iter} + \alpha\frac{dq}{d\lambda} \quad (2.12)$$

เมื่อ $iter$ คือจำนวนรอบของการคำนวณ และ α คือค่าที่ใช้กำหนดระยะในการปรับเปลี่ยนค่าตัวคูณลากรางจ์ตัวใหม่ ซึ่งการกำหนดค่า α นี้จะอาศัยค่าของ $\frac{dq}{d\lambda}$ มากำหนด โดยมีหลักการที่ใช้โดยทั่วไปเป็นดังนี้

$$\alpha = \alpha_1 \quad \text{เมื่อ} \quad \text{มีค่าเป็นบวก}$$

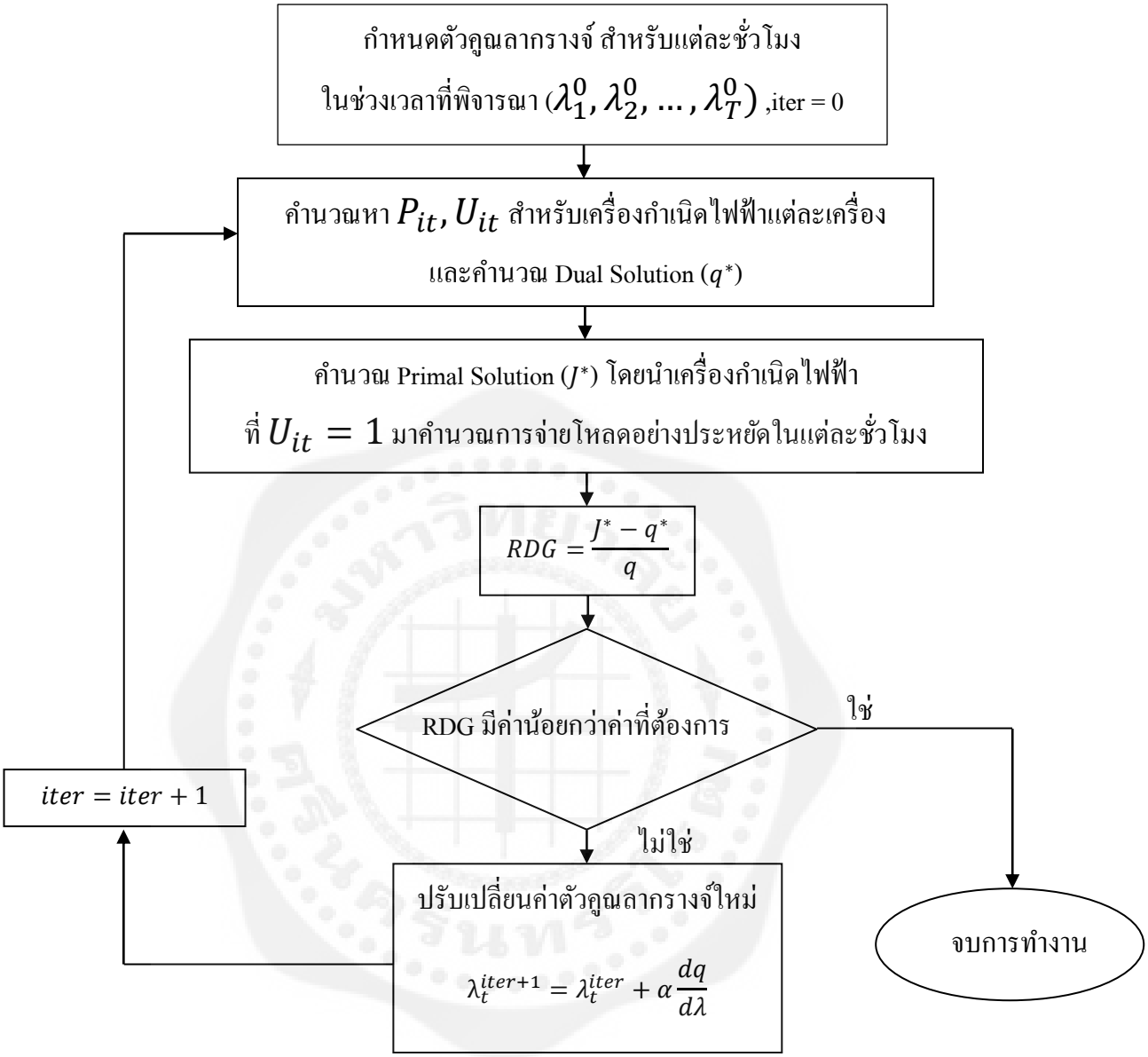
$$\alpha = \alpha_2 \quad \text{เมื่อ} \quad \text{มีค่าเป็นลบ}$$

โดยที่ $\alpha_1 \gg \alpha_2$ เช่น $\alpha_1 = 0.2, \alpha_2 = 0.005$

ตัวคูณลากรางจ์ตัวใหม่ที่ได้นี้จะถูกนำกลับไปหา Dual Solution และ Primal Solution ในรอบการคำนวณใหม่อีกครั้งตามขั้นตอนเดิมข้างต้น ซึ่งการคำนวณนี้จะเป็นการคำนวณแบบวนรอบ โดยจะอาศัยค่า Relative Duality Gap (RDG) เป็นเกณฑ์ในการหยุดคำนวณ โดยจะหยุดคำนวณเมื่อ RDG มีค่าน้อยกว่าค่าที่ต้องการ เช่นมีค่าน้อยกว่า 0.1 หรือ 0.05 เป็นต้น และความสัมพันธ์ที่ใช้ในการคำนวณหาค่า RDG เป็นดังสมการที่ 2.20

$$RDG = \frac{J^* - q^*}{q^*} \tag{2.13}$$

เมื่อสิ้นสุดการคำนวณ ผลลัพธ์ของปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าที่ได้คือสถานะ (เปิด/ปิด) และกำลังไฟฟ้าของเครื่องกำเนิดไฟฟ้าแต่ละเครื่องที่ได้จากการคำนวณในส่วนของ Primal Solution ในรอบการคำนวณสุดท้าย



รูปที่ 2.1 ขั้นตอนการแก้ไขปัญหาการวางแผนเดินเครื่องกำเนิดไฟฟ้าโดยวิธีลากรางจ์รีแล็กเซชั่น

2.2.3.1 ตัวอย่างการแก้ปัญหายูนิตคอมมิทเมนต์โดยใช้วิธีลากรางจ์รีแล็กเซชัน

ขั้นตอนการแก้ไขปัญหายูนิตคอมมิทเมนต์จะถูกแสดงไว้ในโจทย์ตัวอย่าง โดยโจทย์ตัวอย่างนี้จะไม่นำค่า startup cost, minimum up time และ minimum down time มาใช้ในการแก้ไขปัญหา

$$F_1 = 400 + 8P_1 + 0.001P_1^2 \quad \$/hr \;\; 100 \le P_1 \le 500\ MW$$

$$F_2 = 80 + 5P_2 + 0.004P_2^2 \quad \$/hr \;\; 50 \le P_2 \le 200\ MW$$

$$F_3 = 200 + 6P_3 + 0.002P_3^2 \quad \$/hr \;\; 100 \le P_3 \le 350\ MW$$

รอบที่ 1

ในรอบแรกจะเสมือนว่า $\lambda = 0$, เริ่มด้วยการปรับค่า λ

จาก $q(\lambda) \;=\; \sum_{t=1}^{T}\sum_{i=1}^{N}[F_i(P_{it}) - \lambda_t P_{it}]\cdot U_{it} + ST_{it}U_{it} + \lambda_t P_{Dt}$

$$\frac{\partial\, q(\lambda)}{\partial\lambda} = P_{Dt} - \sum_{i=1}^{N} U_{it}P_{it} = 0$$

ถ้า $\frac{\partial\, q(\lambda)}{\partial\lambda}$ เป็นบวก ใช้ $\alpha = 0.01$

ถ้า $\frac{\partial\, q(\lambda)}{\partial\lambda}$ เป็นลบ ใช้ $\alpha = 0.002$

ชั่วโมงที่ 1 $\lambda_1 = \lambda_1 + (P_{1,load})(0.01) = 300 \times 0.01 = 3$

ชั่วโมงที่ 2 $\lambda_2 = \lambda_2 + (P_{2,load})(0.01) = 800 \times 0.01 = 8$

รอบที่ 2 $\lambda_1 = 3$ และ $\lambda_2 = 8$

Unit 1

$$\frac{\partial\, q(\lambda)}{\partial P_1} = \frac{\partial(400 + 8P_1 + 0.001P_1^2 - \lambda_t P_1)}{\partial P_1} = 8 + 0.002 - \lambda_t$$

ดังนั้น $P_1 = \frac{\lambda_t - 8}{0.002}$

Unit1	$P_1(cal)$	P_1	$F_1(P_1)$	$F_1(P_1) - \lambda P_1$	U_{1t}
λ_1	-2500	100	1210	910	0
λ_2	0	100	1210	410	0

ขั้นตอนที่ 1 จะทำการหาค่ากำลังไฟฟ้า P_1 ในแต่ละชั่วโมง

ขั้นตอนที่ 2 เมื่อได้ค่ากำลังไฟฟ้า P_1 ในแต่ละชั่วโมงแล้ว หากค่า P_1 ในแต่ละชั่วโมงไม่เป็นไปตามเงื่อนไขของเครื่องกำเนิดไฟฟ้า จะต้องบังคับค่าของ P_1 ให้เป็นไปตามเงื่อนไขของเครื่องกำเนิดไฟฟ้าคือกำหนดให้ค่า P_1 เป็นค่าพิกัดต่ำสุดที่กำหนดในเงื่อนไข ถ้าค่าของ P_1 ที่ได้จากข้างต้นน้อยกว่าค่าพิกัดต่ำสุดในเงื่อนไขหรือกำหนดให้ค่า P_1 เป็นค่าพิกัดสูงสุดในเงื่อนไข ถ้าค่าของ P_1 ที่ได้จากข้างต้นมากกว่าค่าพิกัดสูงสุดในเงื่อนไข

ขั้นตอนที่ 3 จะทำการหาค่าของ U_{it} โดยแทนค่า λ_1, P_1 ที่ได้จากข้างต้นลงใน สมการ $F_1(P_1) - \lambda P_1$ ถ้าค่าที่ได้ออกมาเป็นบวก ค่าต่ำสุดของเทอมนั้นที่เป็นไปได้คือ 0 ซึ่งทำได้โดยกำหนดให้ $U_{it} = 0$ หรือหากค่าที่ได้มีค่าไม่เป็นบวก(เป็นค่าลบหรือศูนย์) การที่จะทำให้ค่าของเทอมนั้นต่ำสุดได้โดยกำหนด $U_{it} = 1$, หาค่าใน unit 2,3 ต่อ

Unit 2

$$\frac{\partial\, q(\lambda)}{\partial P_2} = \frac{\partial(80 + 5P_2 + 0.004P_2^2 - \lambda_t P_2)}{\partial P_2} = 5 + 0.008 - \lambda_t$$

ดังนั้น $P_2 = \frac{\lambda_t - 5}{0.008}$

Unit2	$P_2(cal)$	P_2	$F_2(P_2)$	$F_2(P_2) - \lambda P_2$	U_{2t}
λ_1	-250	50	340	190	0
λ_2	375	200	1240	-360	1

Unit 3

$$\frac{\partial\, q(\lambda)}{\partial\lambda} = \frac{\partial(200 + 6P_3 + 0.002P_3^2 - \lambda_t P_3)}{\partial\lambda} = 6 + 0.004 - \lambda_t$$

ดังนั้น $P_3 = \frac{\lambda_t - 6}{0.004}$

Unit3	$P_3(cal)$	P_3	$F_3(P_3)$	$F_3(P_3) - \lambda P_3$	U_{3t}
λ_1	-750	100	820	520	0
λ_2	500	350	2545	-255	1

เมื่อสิ้นสุดขั้นตอนในข้างต้น จะสามารถหาค่าของ dual solution $(q^*(\lambda))$ ได้โดยการแทนค่า P, λ, U_{it} ลงในฟังก์ชั่นลากรางจ์ (L) จะได้ค่า $q(\lambda)$ ซึ่งเป็นค่า dual solution $(q^*(\lambda))$ ที่ได้ของแต่ละรอบการคำนวณ

$$q(\lambda) = \sum_{t=1}^{2}\sum_{i=1}^{3}[F_i(P_{it}) - \lambda_t P_{it}] \cdot U_{it} + \sum_{t=1}^{2}\lambda_t P_{Dt}$$

$$q(\lambda) = [F_2(P_{22}) - \lambda_2 P_{22}] + [F_3(P_{32}) - \lambda_2 P_{32}] + 3(300) + 8(800)$$

$$q(\lambda) = -360 - 255 + 900 + 6400 = 6685$$

ขั้นตอนต่อไปจะหา primal solution (J^*) โดยการพิจารณาเฉพาะเครื่องที่มีสถานะเปิด $U_{it} = 1$ ในผลของ dual solution $(q^*(\lambda))$ มาคำนวณการจ่ายโหลดอย่างประหยัดโดยจะใช้วิธีแบบดั้งเดิม (ในการทดลองจะใช้ วิธีการ DEA ในการคำนวณการจ่ายโหลดอย่างประหยัด) แต่อย่างไรก็ตามค่าของ U_{it} ที่ได้มาจากผลของ dual solution $(q^*(\lambda))$ ในบางกรณีไม่สามารถนำมาคำนวณการจ่ายโหลดอย่างประหยัดได้หรือไม่ให้คำตอบที่เป็นไปได้(Feasible Solution) เช่น ในกรณีของโจทย์ตัวอย่างนี้ กรณีที่อาจจะเกิดขึ้นได้ คือ

(1) กรณีที่ค่าของ U_{it} ที่ได้จาก dual solution มีค่าเป็นศูนย์ทั้งหมด

(2) กรณีเครื่องกำเนิดไฟฟ้าที่ได้จาก dual solution มีขนาดพิกัดรวมไม่เพียงพอที่จะจ่ายให้โหลดของระบบ

(3) กรณีที่ความต้องการใช้กำลังไฟฟ้าของระบบมีค่าน้อยกว่าผลรวมของค่าพิกัดต่ำสุดของเครื่องกำเนิดไฟฟ้าทุกเครื่องที่มีสถานะเปิดจาก dual solution

กรณีเหล่านี้จะทำให้ไม่สามารถหาคำตอบของการจ่ายโหลดอย่างประหยัดได้ ดังนั้นเมื่อมีกรณีเหล่านี้เกิดขึ้นในขั้นตอนนี้ จะทำให้ไม่สามารถหาค่าของ J^* ได้ และจากการใช้ค่าของ Relative Duality Gap (RDG) เป็นตัวกำหนดสำหรับการหยุดวนรอบการคำนวณ การกำหนดให้ J^* เป็นค่ามากๆค่าหนึ่งก็เป็นทางออกหนึ่งเมื่อเกิดกกรณีเหล่านี้ เพราะ J^* ที่มีค่ามากๆ จะทำให้ Relative Duality Gap (RDG) มีค่ามาก ในขณะที่การวนรอบของการคำนวณก็จะหยุดต่อเมื่อ RDG มีค่าน้อยกว่าที่ต้องการ ซึ่งในตัวอย่างนี้จะกำหนดให้ J^* มีค่า 10,000 ต่อหนึ่งโรง

$$J^* = 10{,}000 \times 2 = 20{,}000$$

$$RDG = \frac{J^* - q^*}{q^*} = \frac{20000 - 6685}{6685} = 1.99$$

จะเห็นได้ว่าค่า RDG ที่คำนวณได้ยังมีค่ามาก จึงต้องทำการคำนวณรอบต่อไป โดยทำการปรับเปลี่ยนตัวคูณลากรางจ์ใหม่ (ในการทดลองเราจะเปลี่ยนตัวคูณลากรางจ์ด้วยวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม)

$$\frac{\partial\, q(\lambda)}{\partial \lambda} = P_{Dt} - \sum_{i=1}^{N} U_{it} P_{it}$$

$t = 1$ $\quad \frac{dq}{dt} = 300 > 0$ ดังนั้น $\alpha = 0.01$

$t = 2$ $\quad \frac{dq}{dt} = -200 - 350 + 800 > 0$ ดังนั้น $\alpha = 0.01$

เราก็จะได้ตัวคูณลากรางจ์ใหม่

$t = 1$ $\quad \lambda_1 = 3 + 300(0.01) = 6$

$t = 2$ $\quad \lambda_2 = 8 + 250(0.01) = 10.5$

รอบที่ 3

$t = 1$ $\quad \lambda_1 = 6$

$t = 2$ $\quad \lambda_2 = 10.5$

Unit 1

$$\frac{\partial\, q(\lambda)}{\partial P_1} = \frac{\partial(400 + 8P_1 + 0.001P_1^2 - \lambda_t P_1)}{\partial P_1} = 8 + 0.002 - \lambda_t$$

ดังนั้น $P_1 = \frac{\lambda_t - 8}{0.002}$

Unit1	$P_1(cal)$	P_1	$F_1(P_1) - \lambda P_1$	U_{1t}
λ_1	-1000	100	610	0
λ_2	1250	500	-600	1

Unit 2

$$\frac{\partial\, q(\lambda)}{\partial P_2} = \frac{\partial(80 + 5P_2 + 0.004P_2^2 - \lambda_t P_2)}{\partial P_2} = 5 + 0.008 - \lambda_t$$

ดังนั้น $P_2 = \frac{\lambda_t - 5}{0.008}$

Unit2	$P_2(cal)$	P_2	$F_2(P_2) - \lambda P_2$	U_{2t}
λ_1	125	125	17.5	0
λ_2	687.5	200	-860	1

Unit 3

$$\frac{\partial\, q(\lambda)}{\partial P_3} = \frac{\partial(200 + 6P_3 + 0.002P_3^2 - \lambda_t P_3)}{\partial P_3} = 6 + 0.004 - \lambda_t$$

ดังนั้น $P_3 = \frac{\lambda_t - 6}{0.004}$

Unit3	$P_3(cal)$	P_3	$F_3(P_3) - \lambda P_3$	U_{3t}
λ_1	0	100	220	0
λ_2	1125	350	-1130	1

$$q(\lambda) = \sum_{t=1}^{2}\sum_{i=1}^{3}[F_i(P_{it}) - \lambda_t P_{it}] \cdot U_{it} + \sum_{t=1}^{2}\lambda_t P_{Dt}$$

$$q(\lambda) = -600 - 860 - 1130 + 6(300) + 10.5(800) = 7610$$

ทำการคำนวณการจ่ายโหลดอย่างประหยัดโดยวิธีแบบดั้งเดิม ซึ่งในโจทย์ตัวอย่างเราจะไม่พูดถึงวิธีคำนวณดังกล่าวเพราะในการทดลองจะใช้วิธีการดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม ในการคำนวณการจ่ายโหลดอย่างประหยัด จึงนำค่าที่ได้มาแสดง

$$P_1^{ed} = 250, P_2^{ed} = 200, P_3^{ed} = 350$$

จากสมการ

$$q(\lambda) = \sum_{t=1}^{2}\sum_{i=1}^{3}[F_i(P_{it}) - \lambda_t P_{it}] \cdot U_{it} + \sum_{t=1}^{2}\lambda_t P_{Dt}$$

$$J^* = F_1(P_{12}) + F_2(P_{22}) + F_3(P_{32}) + 10000$$

$$J^* = 2462.5 + 1240 + 2545 + 10000 = 16247.5$$

$$RDG = \frac{J^* - q^*}{q^*} = \frac{16247.5 - 7610}{7610} = 1.14$$

จะเห็นได้ว่าค่า RDG ที่ได้ยังมีค่ามากอยู่จึงทำการปรับเปลี่ยนตัวคูณลากรางจ์ใหม่

$$\frac{\partial\, q(\lambda)}{\partial \lambda} = P_{Dt} - \sum_{i=1}^{N} U_{it} P_{it}$$

$t = 1$ $\lambda_1 = 6 + 300(0.01) = 9$

$t = 2$ $\lambda_2 = 10.5 + 250(0.01) = 10$

จากนั้นก็ทำการเพิ่มค่าจำนวนรอบขึ้นไปจนผลลัพธ์ได้ตามน้อยกว่าค่าที่ต้องการ ซึ่งเราอาจจะกำหนดให้ RDG มีค่าน้อยกว่า 0.001 เป็นต้น

2.2.4 ปัญหาการจ่ายโหลดอย่างประหยัด (Economic Dispatch Problem)

ปัญหาการจ่ายโหลดอย่างประหยัด เป็นปัญหาสำคัญของระบบการส่งจ่ายกำลังไฟฟ้าในการดำเนินการผลิตกำลังไฟฟ้า เพื่อให้สามารถส่งจ่ายกำลังไฟฟ้าให้เพียงพอกับความต้องการของ

ผู้ใช้ และมีต้นทุนการผลิตต่ำที่สุด วัตถุประสงค์ของการจ่ายโหลดอย่างประหยัด คือการทำให้เครื่องกำเนิดไฟฟ้า ผลิตกำลังไฟฟ้าโดยใช้เชื้อเพลิงในการผลิตน้อยที่สุด จึงถือได้ว่าเป็นปัญหาการหาค่าเหมาะสม (Optimization Problem) ปัญหาการหาค่าที่เหมาะสมที่สุดของเครื่องกำเนิดไฟฟ้ากำหนดให้ฟังก์ชันของเชื้อเพลิง (Fuel Cost Function) เป็นฟังก์ชันเป้าหมาย (Objective Function) เพื่อต้องการให้มีค่าน้อยที่สุด

$$F_T = \sum_{i=1}^{N} F_i(P_i) \tag{2.14}$$

เมื่อ F_T คือ ต้นทุนรวมในการผลิตกำลังไฟฟ้า

F_i คือ ราคาเชื้อเพลิงของเครื่องกำเนิดเครื่องที่ i

เป้าหมายหลักของการจ่ายโหลดอย่างประหยัดของเครื่องกำเนิดไฟฟ้า คือ การหาค่ากำลังการผลิตไฟฟ้าที่เหมาะสมที่สุดของเครื่องกำเนิดแต่ละเครื่องให้ตรงตามความต้องการของโหลด โดยใช้เชื้อเพลิงน้อยที่สุด ซึ่งจะขึ้นอยู่กับข้อจำกัดการทำงานของเครื่องกำเนิดไฟฟ้าแต่ละเครื่อง เราสามารถเขียนสมการของการผลิตกำลังไฟฟ้าได้ดังนี้ คือ

$$\sum_{i=1}^{N} (P_i) = P_{load} + P_{loss} \tag{2.15}$$

เมื่อ P_i คือ กำลังไฟฟ้าที่ผลิตจากเครื่องกำเนิดไฟฟ้าเครื่องที่ i

P_{load} คือ กำลังไฟฟ้าที่โหลดต้องการ

P_{loss} คือ กำลังไฟฟ้าสูญเสียในระบบ [30No3-2]

โดยปกติแล้วฟังก์ชันของราคาเชื้อเพลิงที่ใช้ในปัญหาการจ่ายโหลดอย่างประหยัดจะอยู่ในรูปของสมการกำลังสอง (Quadratic Function) หรือสมการโพลิโนเมียล (Polynomial Equation) ซึ่งเป็นฟังก์ชันราคาที่ราบเรียบ (Smooth Cost Function) และสมการต่อไปนี้

$$F_i(P_i) = a_i + b_i P_i + c_i P_i^2$$

เมื่อ a_i, b_i, c_i คือ สัมประสิทธิ์ของฟังก์ชันอัตราความร้อนของเครื่องกำเนิดไฟฟ้าเครื่องที่ i (นางสาววิลาสินี ศึกษาการ, 2551)

2.3 พื้นฐานดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม (Basis of Differential Evolution Algorithm)

DEA เป็นวิวัฒนาการของอัลกอริธึมแบบใหม่ที่มีการใช้ค่าตัวแปรจริงในการทำงานและอาศัยการมิวเตชั่น (Mutation) เป็นวิธีการค้นหาคำตอบ วิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม(DEA) มีหลักการทำงานคล้ายกับ Conventional Genetic Algorithm (CGA) ซึ่งวิธีการทั้งสองจะรักษาประชากรของการแก้ปัญหาที่เกิดขึ้นและใช้กลไกการคัดเลือกในการเลือกบุคคล (individuals) ที่ดีที่สุดจากประชากรแตกต่าง

สำหรับข้อแตกต่างของ DEA กับ CGA มีดังต่อไปนี้

- DEA ทำงานโดยตรงในจุดที่เวคเตอร์ตั้งอยู่ ขณะที่ CGA อาศัยหลักการของสตริงที่เป็นข้อมูลเลขฐานสอง
- CGA อาศัยหลัก recombination ในการสำรวจและค้นหาพื้นที่ ในขณะที่ DEA ใช้วิธีพิเศษจากการ mutation ที่เป็นตัวดำเนินการที่โดดเด่น
- DEA เป็นส่วนหนึ่งของวิวัฒนาการที่ระดับพฤติกรรมแต่ละบุคคล (individuals) มุ่งเน้นการเชื่อมโยงระหว่างบุคคลกับรุ่นลูกหลาน ในขณะที่ CGA ยังคงรักษาการเชื่อมโยงทางพันธุกรรมไว้

DEA จะทำงานแบบขนาน โดยใช้ประชากร P ขนาด N_p ในประชาการ $P^{(G)}$ จะประกอบด้วย สมาชิกที่อยู่ในรูปเวคเตอร์ $X^{(G)}$ ในทุกๆเจนเนอเรชั่น G ระหว่างขั้นตอนการเพิ่มประสิทธิภาพ DEA จะรักษาประชากร $P^{(G)}$ ของเวคเตอร์ N_p ของปัญหา

$$P^{(G)} = \left[X_1^{(G)}, \dots, X_i^{(G)}, \dots, X_{N_P}^{(G)}\right], i = 1, \dots, N_P \tag{2.16}$$

โดยแต่ละ $X_i^{(G)}$ จะมีมิติเท่ากับ D มิติ เป็นพารามิเตอร์เลขจำนวนเต็ม การแก้ไขปัญหาขึ้นอยู่กับพารามิเตอร์ D

$$X_i^{(G)} = \left[x_{1,i}^{(G)}, \dots, x_{j,i}^{(G)}, \dots, x_{D,i}^{(G)}\right], i = 1, \dots, N_P; j = 1, \dots, D \tag{2.17}$$

2.3.1 การสมมติค่าเริ่มต้น (Initialization)

ในขั้นตอนแรกของ DEA เริ่มจากการกำหนดค่าที่เป็นไปได้เริ่มต้น โดนเริ่มจากการกำหนดขอบเขตบนและขอบเขตล่างของตัวแปรในการตัดสินใจแต่ละตัว โดยขอบเขตนี้จะต้องครอบคลุมจุดที่ให้คำตอบที่ดีที่สุดจากนั้นให้สุ่มหาประชากรคำตอบที่เป็นไปได้เริ่มต้น (Initial

Population) โดยกำหนดให้โอกาสที่จะถูกเลือกของคำตอบมีค่าสม่ำเสมอ (Uniform Probability Distribution) โดยคำตอบแต่ละคำตอบซึ่งเรียกว่า Decision Vector จะมีมิติเท่ากับ D และจำนวนคำตอบที่เป็นไปได้เริ่มต้นเท่ากับ N_p จากนั้นคำนวณหา Function Value ของแต่ละคำตอบเริ่มต้นที่เป็นไปได้ ดังสมการที่ 2.18

$$x_{j,i}^{(G=0)} = x_j^{min} + rand_j[0,1].\left(x_j^{max} - x_j^{min}\right) \qquad (2.18)$$

เมื่อ $i = 1, \dots, N_p$

$j = 1, \dots, D$

$x_{j,i}^{(G=0)}$ คือ ค่าเริ่มต้นของพารามิเตอร์ j^{th} ของเวกเตอร์อิสระ i^{th}

x_j^{min}, x_j^{max} คือ ค่าต่ำสุดและสูงสุดของพารามิเตอร์ j^{th} ตามลำดับ

2.3.2 การมิวเตชั่น (Mutation)

กระบวนการ DEA จะดำเนินการตามหลักการพันธุกรรมพื้นฐาน 3 ขั้นตอนดังนี้คือ มิวเตชั่น(Mutation) ครอสโอเวอร์(Crossover) และการคัดเลือก(Selection)เพื่อให้ได้ประชากรในเจนเนอเรชั่นถัดไปคือ $P^{(G+1)}$ โดยใช้ประชากรในปัจจุบัน $P^{(G)}$

การมิวเตชั่นจะเริ่มต้นด้วยการกำหนดเวกเตอร์เป้าหมาย $(X_{i,G})$ โดยที่ $i = 1,2,3, \dots, N_P$ แล้วจึงทำการสุ่มเลือก Random Vector จำนวน 3 เวกเตอร์ $(X_{r1,G}, X_{r2,G}, X_{r3,G})$ จากประชากรตั้งต้นโดยที่จะต้องไม่ซ้ำกับเวกเตอร์เป้าหมาย แล้วจึงจะคำนวณหามิวแตนท์เวกเตอร์ $(V_{i,G+1})$ จากความสัมพันธ์

$$V_{i,G+1} = X_{r1,G} + F(X_{r2,G} - X_{r3,G}); i = 1,2,3, \dots, N_P \qquad (2.19)$$

โดย $X_{i,G}$ คือ เวกเตอร์เป้าหมาย (Target Vector)

$V_{i,G+1}$ คือ มิวแตนท์เวกเตอร์ (Mutant Vector)

F คือ จำนวนจริงที่มีค่าคงที่และมีค่าระหว่าง 0 ถึง 2 (Weighing Factor)

$X_{r1,G}, X_{r2,G}, X_{r3,G}$ คือ Random Vector

โดยที่ $i, r1, r2, r3 \in \{1, \dots, N_P\}$ และ $i \neq r1 \neq r2 \neq r3$

การมิวเตชั่นนั้นจะมีอยู่ด้วยกัน 10 แบบซึ่งแบบที่ 1-5 นั้นจะเป็นแบบดั้งเดิมที่ถูกเสนอขึ้นโดยสตอร์น และแบบที่ 6-10 เป็นแบบใหม่ ดังนี้

DEA1 ในวิธีแรกนี้เวกเตอร์ดัดแปลงจะถูกสร้างขึ้นจากสมการที่ 2.19

$$V_i^{(G)} = X_{best}^{(G)} + F\left(X_{r2}^{(G)} - X_{r3}^{(G)}\right), i = 1, \dots, N_P \qquad (2.20)$$

เมื่อ $X_{best}^{(G)}$ เป็นเวคเตอร์ที่ดีที่สุดในรุ่นปัจจุบัน

DEA2 จะมีวิธีการทำงานคล้ายกับวิธีแรกแต่จะแตกต่างกันโดยที่วิธีนี้จะทำงานอยู่บนฐานของเวกเตอร์สมาชิกที่สุ่มเลือกมา $X_{r1}^{(G)}$ คำนวณได้ ดังสมการที่ 2.21

$$V_i^{(G)} = X_{r1}^{(G)} + F\left(X_{r2}^{(G)} - X_{r3}^{(G)}\right), i = 1, \dots, N_P \qquad (2.21)$$

DEA3 วิธีนี้จะทำการดัดแปลงเวกเตอร์โดยการใช้ความแตกต่างจากวิธีที่ดีที่สุดและความแตกต่างของเวกเตอร์สมาชิกที่สุ่มเลือกมา ดังสมาการที่ 2.22

$$V_i^{(G)} = X_i^{(G)} + \lambda\left(X_{best}^{(G)} - X_i^{(G)}\right) + F\left(X_{r1}^{(G)} - X_{r2}^{(G)}\right), i = 1, \dots, N_P \qquad (2.22)$$

λ ถูกใช้เป็นตัวควบคุมรูปแบบ ซึ่งโดยปกติจะตั้งค่าให้เท่ากับ F เพื่อลดจำนวนตัวแปรควบคุม

DEA4 มีการดัดแปลงเวกเตอร์โดยการใช้ค่าความแตกต่างของสองเวกเตอร์สมาชิกที่สุ่มเลือกมา

ดังสมการที่ 2.23

$$V_i^{(G)} = X_{best}^{(G)} + F\left(X_{r1}^{(G)} + X_{r2}^{(G)} - X_{r3}^{(G)} - X_{r4}^{(G)}\right), i = 1, \dots, N_P \qquad (2.23)$$

DEA5 มีการทำงานคล้ายกับ DEA4 แต่จะแตกต่างกันโดยการแทนค่าที่ดีที่สุด ($X_{best}^{(G)}$) ใน DEA4 ด้วยเวกเตอร์สมาชิกที่สุ่มเลือกมา ดังสมการที่ 2.24

$$V_i^{(G)} = X_{r5}^{(G)} + F\left(X_{r1}^{(G)} + X_{r2}^{(G)} - X_{r3}^{(G)} - X_{r4}^{(G)}\right), i = 1, \dots, N_P \qquad (2.24)$$

DEA6 มีการทำงานคล้ายกับ DEA1 ในการสร้าง Mutant Vector แต่จะแตกต่างกันที่วิธีนี้จะแทนเวกเตอร์ที่สุ่มเลือกมา $(X_{r2}^{(G)}, X_{r3}^{(G)})$ เป็นค่าเวกเตอร์สมาชิกที่ดีที่สุดกับค่าเวกเตอร์ที่จะทำการดัดแปลง $(X_{best}^{(G)}, X_i^{(G)})$ ดังสมการที่ 2.25

$$V_i^{(G)} = X_{best}^{(G)} + F\left(X_{best}^{(G)} - X_i^{(G)}\right), i = 1, \dots, N_P \qquad (2.25)$$

DEA7 มีหลักการคิดเหมือนกับ DEA4 แต่จะใช้เวกเตอร์ 3 เวกเตอร์ที่ต่างกันในการดัดแปลงดังสมการที่ 2.26

$$V_i^{(G)} = X_{best}^{(G)} + F\left(X_{best}^{(G)} - X_i^{(G)} - X_{r1}^{(G)} - X_{r2}^{(G)}\right), i = 1, \dots, N_P \qquad (2.26)$$

DEA8 มีหลักการคิดเหมือนกับ DEA3 แต่จะต่างกันตรงที่ DEA3 จะคิดบนฐานของเวกเตอร์สมาชิกที่ดีที่สุด ส่วน DEA8 จะคิดบนฐานของเวกเตอร์ตัวที่ถูกดัดแปลงดังสมการที่ 2.27

$$V_i^{(G)} = X_{best}^{(G)} + \lambda\left(X_{best}^{(G)} - X_i^{(G)}\right) + F\left(X_{r1}^{(G)} - X_{r2}^{(G)}\right), i = 1, \dots, N_P \qquad (2.27)$$

DEA9 มีการทำงานคล้ายกับ DEA5 แต่จะแตกต่างกันที่วิธีนี้จะใช้การสุ่มเวกเตอร์สองเวกเตอร์เท่านั้นดังสมการที่ 2.28

$$V_i^{(G)} = X_{best}^{(G)} + F\left(X_{best}^{(G)} + X_i^{(G)} - X_{r1}^{(G)} - X_{r2}^{(G)}\right), i = 1, \dots, N_P \qquad (2.28)$$

DEA10 วิธีนี้จะคล้ายกับ DEA6 แต่จะแทน $X_i^{(G)}$ ด้วย $X_{best}^{(G-1)}$ จากเจเนอเรชั่นก่อนหน้า ดังสมการที่ 2.29

$$V_i^{(G)} = X_{best}^{(G)} + F\left(X_{best}^{(G)} - X_{best}^{(G-1)}\right), i = 1, \dots, N_P \qquad (2.29)$$

2.3.3 การครอสโอเวอร์ (Crossover)

ในกระบวนการนี้จะช่วยเพิ่มความหลากหลายให้กับ Mutant Vector ในรุ่นปัจจุบัน โดยจะสร้างเวกเตอร์ทดลอง $u_{j,i}^{(G)}$ (Trial Vector) ด้วยการผสมผสานระหว่าง Mutant Vector (V_i) กับเวกเตอร์เป้าหมาย X_i (Target Vector) โดยสามารถหาค่าเวกเตอร์ทดลองได้จากสมการ

$$U_i^{(G)} = u_{j,i}^{(G)} = \begin{cases} V_{j,i}^{(G)} \text{ ถ้า } rand_j[0,1] \le CR \text{ หรือ } j = s \\ X_{j,i}^{(G)} \text{ ในกรณีอื่นๆ} \end{cases} \qquad (2.30)$$

เมื่อ $u_{j,i}^{(G)}$ = Trial Vector

$V_{j,i}^{(G)}$ = Mutant Vector

$X_{j,i}^{(G)}$ = Target Vector

$rand_j$ = ค่าที่สุ่มขึ้นมา มีค่า (0,1)

CR = Crossover Constant มีค่าเป็นเลขจำนวนจริงระหว่าง 0 ถึง 1

S = ค่า Index จากการสุ่มเลือกมีค่าเป็นเลขจำนวนเต็มระหว่าง 1,2,…,D ซึ่งทำให้มั่นใจได้ว่า อย่างน้อย 1 พารามิเตอร์มาจาก Mutant Vector

2.3.4 การคัดเลือก (Selection)

การคัดเลือกเป็นกระบวนการคัดสรรหาคำตอบเป็นกระบวนการสุดท้าย โดยมีวิธีการคือ เปรียบเทียบค่าของฟังก์ชั่นของเวกเตอร์ทดลอง (Trial Vector)กับเวกเตอร์เป้าหมาย (Target Vector) ถ้าเวกเตอร์ทดลองให้ค่าของฟังก์ชันที่ดีกว่า มันก็จะแทนที่เวกเตอร์เป้าหมายในเจนเนอเรชั่นต่อไป จากนั้นก็จะทำซ้ำขั้นตอนที่ 2 ถึงขั้นตอนที่ 3 จนครบทุกเวกเตอร์ในเจนเนอเรชั่นปัจจุบัน และแทนที่เจนเนอเรชั่นปัจจุบันด้วยเจนเนอเรชั่นต่อไป แล้วทำซ้ำกระบวนการทั้งหมดจนถึง Stopping Criteria

$$X_i^{(G+1)} = \begin{cases} U_i^{(G)} \text{ถ้า } f\left(U_i^{(G)}\right) \leq f\left(X_i^{(G)}\right) \\ X_i^{(G)} \text{ ในกรณีอื่นๆ} \end{cases} \quad (2.31)$$

ดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม แบบปรับตัวเองได้ (Self-Adaptive Differential Evolution Algorithm) ดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม แบบนี้ จะแตกต่างจากดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม แบบดั้งเดิมตรงที่แบบดั้งเดิมนั้นไม่สามารถปรับค่าควบคุมการทำงาน Scaling mutation factor(F)กับCrossover constant(CR) ได้ ทำให้คำตอบที่ได้ไม่หลากหลาย แต่ดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม แบบปรับตัวเองได้ จะสามารถปรับเปลี่ยนค่าพารามิเตอร์ควบคุม F กับ CR ได้ จะทำให้ได้ค่าคำตอบที่หลากหลายมากขึ้น โดยส่วนมากจะทำการกำหนดช่วงค่าที่จะนำไปใช้ เช่น F อยู่ระหว่าง [0.5,1.0] โดยทั่วไปแล้วค่าที่นำมาควบคุมพารามิเตอร์ F และ CR จะอยู่ในช่วง 0 ถึง 1

การทำงานของ DEA

ขั้นตอนการทำงานของ DEA เป็นดังนี้

ขั้นตอนที่ 1 รับค่าพารามิเตอร์ควบคุมต่างๆ $F\ CR\ N_p\ D$

ขั้นตอนที่ 2 สมมติค่าประชากรเริ่มต้นโดยเป็นเวกเตอร์ขนาด D จำนวน N_p เวกเตอร์

ขั้นตอนที่ 3 ประเมินค่าประชากรที่สมมติได้จากขั้นตอนที่ 2

ขั้นตอนที่ 4 สร้างเวกเตอร์มิวเตชั่นจากประชากรที่มีจำนวน N_p เวกเตอร์

ขั้นตอนที่ 5 สร้างเวกเตอร์ทดลองจากการครอสโอเวอร์จำนวน N_p เวกเตอร์

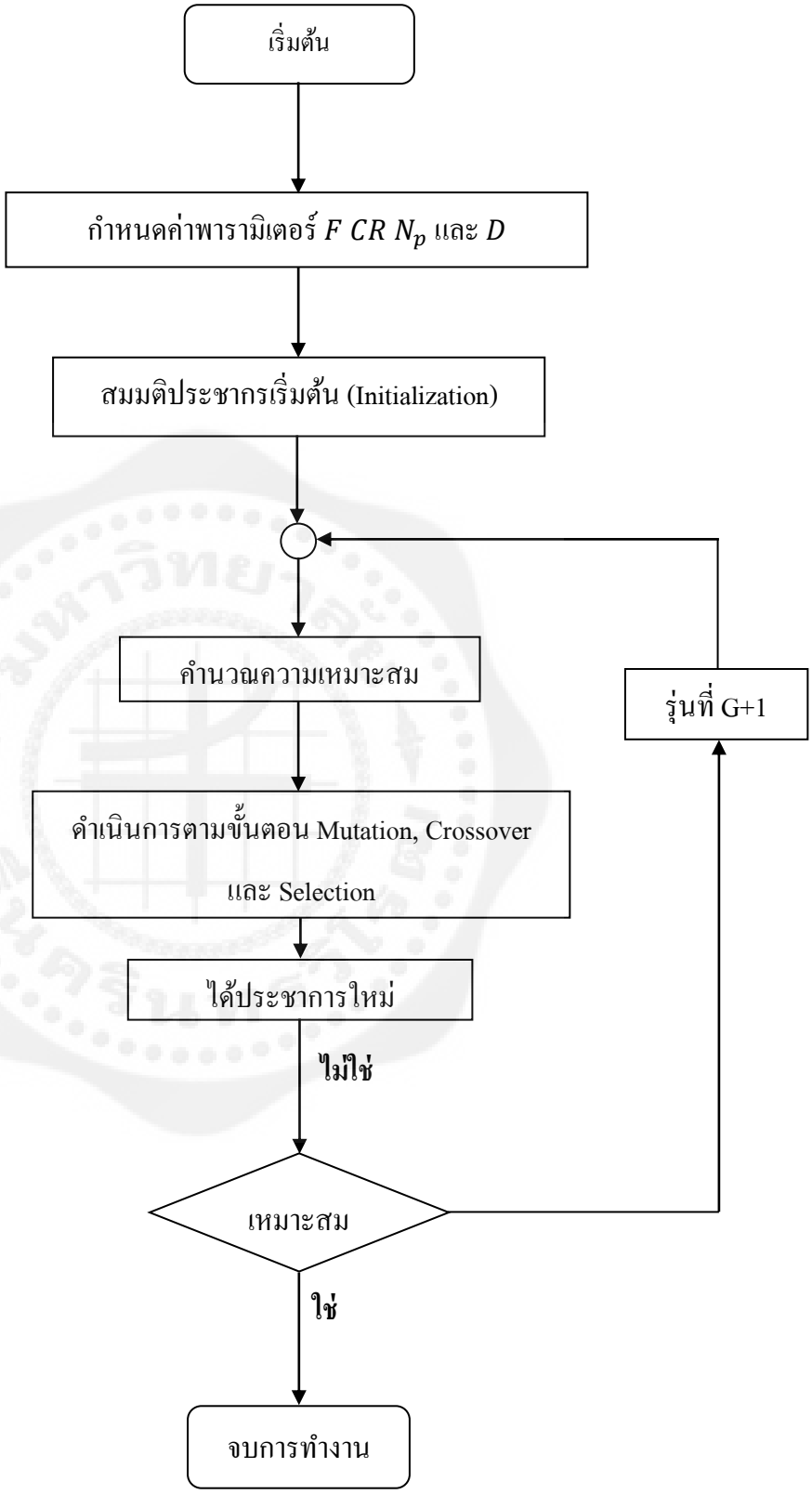
ขั้นตอนที่ 6 ประเมินค่าเวกเตอร์ทดลองที่สร้างขึ้น

ขั้นตอนที่ 7 ทำการคัดเลือกโดยเปรียบเทียบค่าระหว่างประชากรเดิมกับเวกเตอร์ทดลองที่สร้างขึ้น

ขั้นตอนที่ 8 นำประชากรที่ทำการคัดเลือกแล้วใช้เป็นประชากรสำหรับในรอบการคำนวณถัดไป

ขั้นตอนที่ 9 ทำซ้ำขั้นตอนที่ 4 ถึงขั้นตอนที่ 8 จนกว่าค่าที่ดีที่สุดของประชากรจะเป็นค่าที่เหมาะสม หรือจบการคำนวณรอบสูงสุด

สามารถสรุปการทำงานของ DEA เป็นแผนผังได้ตามรูปที่ 2.2



รูปที่ 2.2 แผนผังแสดงการทำงาน DEA

ตัวอย่าง การทำงานของดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม (DEA)

ให้ฟังก์ชั่นวัตถุประสงค์(Objective Function) :

$f(x)=x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8$

ขั้นตอนที่ 1 กำหนดพารามิเตอร์ที่ใช้ในการควบคุม

ตารางที่ 2.1 แสดงค่าพารามิเตอร์ที่ใช้ในการควบคุม

จำนวนตัวแปรทั้งหมด (Decision Parameter,D)	8
จำนวนประชากร (Population size,N_p)	5
Scaling mutation factor (F)	0.7
Crossover constant (CR)	0.6

หมายเหตุ: N_p ควรมีค่ามากว่า D เพื่อเพิ่มความหลากหลายของคำตอบ

ขั้นตอนที่ 2 ทำการ Initialize ประชาการ P ด้วยสมการที่ 18

ตารางที่ 2.2 แสดงค่าที่ได้จากการ Initialize ประชาการ P

Parameters/Individuals	Individual 1	Individual 2	Individual3	Individual 4	Individual 5
Parameter 1 (x_1)	0.95	0.57	0.18	0.92	0.6
Parameter 2 (x_2)	0.43	0.88	0.29	0.87	0.79
Parameter 3 (x_3)	0.38	0.93	0.99	0.65	0.28
Parameter 4 (x_4)	0.78	0.74	0.86	0.47	0.34
Parameter 5 (x_5)	0.64	0.81	0.39	0.38	0.56
Parameter 6 (x_6)	0.55	0.66	0.42	0.82	0.93
Parameter 7 (x_7)	0.71	0.59	0.56	0.21	0.86
Parameter 8 (x_8)	0.82	0.28	0.8	0.33	0.33
Fitness $f(x)$	5.26	5.46	4.49	4.65	4.69

โดย Fitness Function ก็คือ ฟังก์ชั่นวัตถุประสงค์(Objective function)

ขั้นตอนที่ 3 เลือก ฟังก์ชั่นเป้าหมาย(target vector,i) และ random vector($r1, r2$ และ $r3$) จากประชาการที่สมมติขึ้นในขั้นตอนที่ 2 โดยที่ $i, r1, r2$ และ $r3$จะต้องอยู่ใน $\{1, \ldots., N_p\}$ และ $i \neq r1 \neq r2 \neq r3$ ในที่นี้เราจะเลือก

ตารางที่ 2.3 แสดงการเลือกฟังก์ชั่นเป้าหมายและ random vector

i	1
$r1$	5
$r2$	3
$r3$	4

ขั้นตอนที่ 4 ทำการหา mutant vector($V_{i,G+1}$) โดยใช้สมการที่ 4 ได้ผลดังตารางที่ 2.4

ตารางที่ 2.4 แสดงผลการสร้าง mutant vector หรือกระบวนการ mutation

	X_{r1}	X_{r2}	X_{r3}	$X_{r2} - X_{r3}$	$F(X_{r2} - X_{r3})$	$X_{r1} + F(X_{r2} - X_{r3})$
Parameter 1 (x_1)	0.6	0.18	0.92	-0.74	-0.518	0.082
Parameter 2 (x_2)	0.79	0.29	0.29	-0.58	-0.406	0.384
Parameter 3 (x_3)	0.28	0.99	0.99	0.34	0.238	0.518
Parameter 4 (x_4)	0.34	0.86	0.86	0.39	0.273	0.613
Parameter 5 (x_5)	0.56	0.39	0.39	0.01	0.007	0.567
Parameter 6 (x_6)	0.93	0.42	0.42	-0.4	-0.28	0.65
Parameter 7 (x_7)	0.86	0.56	0.56	0.35	0.245	1.105
Parameter 8 (x_8)	0.33	0.8	0.8	0.47	0.329	0.659
Fitness $f(x)$	4.69	4.49	4.65	-	-	4.578

ขั้นตอนที่ 5 ทำการ crossover โดยใช้สมการที่ 2.30 (ถ้าสุ่มค่าแล้วน้อยกว่าหรือเท่ากับค่า CR ให้เลือก mutant vector ($V_{j,i}^{(G)}$) ถ้านอกจากนั้นให้เลือก target vector ($X_{j,i}^{(G)}$)) จะได้ผลดังตารางที่ 2.5

ตารางที่ 2.5 แสดงกระบวนการ crossover

	Target vector	Mutant vector	Trial vector	Random(0,1)
Parameter 1 (x_1)	0.95	0.082	0.082	0.43
Parameter 2 (x_2)	0.43	0.384	0.384	0.15
Parameter 3 (x_3)	0.38	0.518	0.38	0.78
Parameter 4 (x_4)	0.78	0.613	0.613	0.44
Parameter 5 (x_5)	0.64	0.567	0.64	0.91
Parameter 6 (x_6)	0.55	0.65	0.65	0.27
Parameter 7 (x_7)	0.71	1.105	0.71	0.66
Parameter 8 (x_8)	0.82	0.659	0.659	0.35
Fitness $f(x)$	5.26	4.578	4.118	-

ขั้นตอนที่ 6 ทำการคัดเลือกระหว่าง Target vector กับ Trial vector โดยการเลือกค่าที่เหมาะสม เพื่อส่งไปยังประชาการรุ่นต่อไป ในที่นี้จะเลือก Trial vector เพราะมีค่าต่ำที่สุดเมื่อ Fitness Function คือสมการ f(x)

ตารางที่ 2.6 แสดงประชาการในรุ่นที่ 2

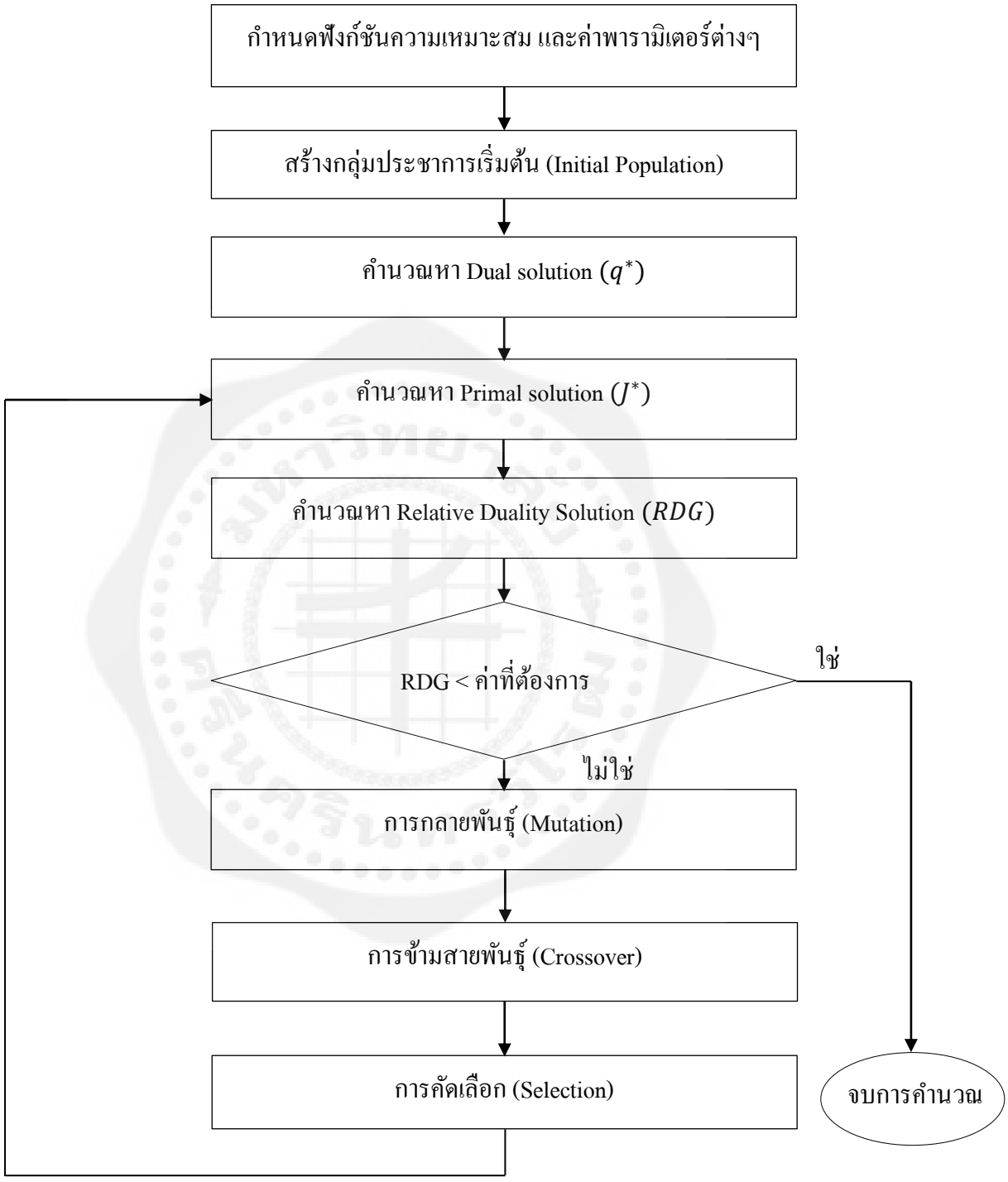
Parameters/Individuals	Individual 1	Individual 2	Individual3	Individual 4	Individual 5
Parameter 1 (x_1)	0.082				
Parameter 2 (x_2)	0.384				
Parameter 3 (x_3)	0.38				
Parameter 4 (x_4)	0.613				
Parameter 5 (x_5)	0.64				
Parameter 6 (x_6)	0.65				
Parameter 7 (x_7)	0.71				
Parameter 8 (x_8)	0.659				
Fitness $f(x)$	4.118				

ขั้นตอนที่ 7 ทำซ้ำในขั้นตอนที่ 3-6 อีกครั้ง จนได้ค่าที่เหมาะสมสำหรับคำตอบ(DEA จะทำการคำนวณแบบขนาน ซึ่งจะทำให้ได้ประชากรในรุ่นที่ 2 มาพร้อมกัน และนำมาคำนวณค่าที่เหมาะสม หากยังไม่พบค่าตามที่ต้องการก็จะทำการสร้างประชาการในรุ่นที่ 3 ต่อไป)

2.4 การประยุกต์ใช้วิธีลากรางจ์รีแล็กเซชันร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่น

วิธีลากรางจ์รีแล็กเซชันร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม (Lagrangian Relaxation combined with Differential Evolution Algorithm, LRDEA) เป็นวิธีที่ประยุกต์รวมวิธีลากรางจ์รีแล็กเซชัน (LR) กับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม (DEA) เข้าด้วยกัน โดยการนำวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม มาใช้ในการปรับเปลี่ยนค่าของตัวคูณลากรางจ์ ให้กับวิธีลากรางจ์รีแล็กเซชันในการแก้ปัญหายูนิตคอมมิทเมนต์ ซึ่งวิธีวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมมาใช้ในการปรับเปลี่ยนค่าตัวคูณลากรางจ์ให้กับวิธีลากรางจ์รีแล็กเซชันนี้ จะทำให้การลู่เข้าหาคำตอบของวิธีลากรางจ์รีแล็กเซชันกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม ดีกว่าการลู่เข้าหาคำตอบของวิธีลากรางจ์รีแล็กเซชัน อีกทั้งช่วยให้คุณภาพของผลลัพธ์ที่ได้ดีขึ้นกว่าการใช้วิธีลากรางจ์รีแล็กเซชันในการแก้ปัญหา

ขั้นตอนโดยสรุปของการแก้ปัญหายูนิตคอมมิทเมนต์โดยใช้วิธีลากรางจ์รีแล็กเซชันร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม แสดงได้ดังรูปที่ 2.3



รูปที่ 2.3 การแก้ปัญหายูนิตคอมมิทเมนต์โดยใช้วิธีลากรางจ์รีแล็กเซชันร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม

2.4.1 ตัวอย่างการคำนวณของวิธีลากรางจ์รีแล็กเซชันร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม

ฟังก์ชั่นราคาเชื้อเพลิงและเงื่อนไขพิกัดของเครื่องกำเนิดไฟฟ้าของแต่ละเครื่อง

$F_1 = 850 + 9P_1 + 0.001P_1^2 \quad \$/hr \;\; 200 \le P_1 \le 1000 \text{ MW} \;\; T_{1,min,up} = 7, \;\; T_{1,min,doen} = 2$

$F_2 = 600 + 8P_2 + 0.002P_2^2 \quad \$/hr \;\; 150 \le P_2 \le 850 \text{ MW} \;\; T_{1,min,up} = 3, \;\; T_{1,min,doen} = 2$

$F_3 = 550 + 9P_3 + 0.002P_3^2 \quad \$/hr \;\; 100 \le P_3 \le 750 \text{ MW} \;\; T_{1,min,up} = 3, \;\; T_{1,min,doen} = 1$

$F_4 = 600 + 8P_4 + 0.004P_4^2 \quad \$/hr \;\; 100 \le P_4 \le 650 \text{ MW} \;\; T_{1,min,up} = 2, \;\; T_{1,min,doen} = 2$

$F_5 = 500 + 7P_5 + 0.008P_5^2 \quad \$/hr \;\; 80 \le P_5 \le 500 \text{ MW} \;\; T_{1,min,up} = 2, \;\; T_{1,min,doen} = 1$

$F_6 = 400 + 6P_6 + 0.005P_6^2 \quad \$/hr \;\; 50 \le P_6 \le 400 \text{ MW} \;\; T_{1,min,up} = 2, \;\; T_{1,min,doen} = 1$

$F_7 = 400 + 9P_7 + 0.006P_7^2 \quad \$/hr \;\; 20 \le P_7 \le 350 \text{ MW} \;\; T_{1,min,up} = 1, \;\; T_{1,min,doen} = 1$

ความต้องการไฟฟ้า 8 ชั่วโมงตามลำดับ 900, 800, 1000, 1300, 1250, 1800, 3000, 2000 MW

สมการราคาต้นทุน

$$q(\lambda) = \sum_{t=1}^{T}\sum_{i=1}^{N}[F_i(P_{it}) - \lambda_t P_{it}] \cdot U_{it} + ST_{it}U_{it} + \lambda_t P_{Dt}$$

$$\frac{\partial\, q(\lambda)}{\partial \lambda} = P_{Dt} - \sum_{i=1}^{N} U_{it}P_{it}$$

กำหนดค่า λ เริ่มต้นเท่ากับ 2 โดย ในรอบที่ 1 เปรียบเสมือนยังไม่มีการจ่ายโหลดโรงไฟฟ้าทั้งหมดจะดับ และทำการปรับค่า λ

ถ้า $\frac{\partial\, q(\lambda)}{\partial \lambda}$ เป็นบวก ใช้ $\alpha = 0.01$

ถ้า $\frac{\partial\, q(\lambda)}{\partial \lambda}$ เป็นลบ ใช้ $\alpha = 0.002$

$t = 1 \qquad \lambda_1 = 2 + 975(0.01) = 11.75$

$t = 2 \qquad \lambda_2 = 2 + 950(0.01) = 11.50$

$t = 3 \qquad \lambda_3 = 2 + 850(0.01) = 10.50$

$t = 4 \qquad \lambda_4 = 2 + 800(0.01) = 10.00$

$t = 5$ $\lambda_5 = 2 + 900(0.01) = 11.00$

$t = 6$ $\lambda_6 = 2 + 1350(0.01) = 15.50$

$t = 7$ $\lambda_7 = 2 + 1850(0.01) = 20.50$

$t = 8$ $\lambda_8 = 2 + 2350(0.01) = 25.50$

การคำนวณกำลังไฟฟ้าของ generator ตัวที่ 1 รอบที่ 1 ทั้งหมด 8 ชั่วโมง

$$\frac{\partial\, q(\lambda)}{\partial P_1} = \frac{\partial(850 + 9P_1 + 0.001P_1^2 - \lambda_t P_1)}{\partial P_1} = 9 + 0.002P_1 - \lambda_t$$

ดังนั้น $P_1 = \frac{\lambda_t - 9}{0.002}$

ตารางที่ 2.7 แสดงข้อมูลกำลังไฟฟ้าและสถานะของ Generator ตัวที่ 1 รอบที่ 1

t	P_1	U_i^t	$F(P_1) - \lambda P_1$
1	1375	1	-900
2	1250	1	-650
3	750	1	-2337.5
4	500	1	-4400
5	1000	1	-150
6	3250	1	-4650
7	5750	1	-9650
8	8250	1	-14650

การคำนวณกำลังไฟฟ้าของ generator ตัวที่ 2 รอบที่ 1 ทั้งหมด 8 ชั่วโมง

$$\frac{\partial\, q(\lambda)}{\partial P_2} = \frac{\partial(600 + 8P_2 + 0.002P_2^2 - \lambda_t P_2)}{\partial P_2} = 8 + 0.004P_2 - \lambda_t$$

ดังนั้น $P_2 = \frac{\lambda_t - 8}{0.004}$

ตารางที่ 2.8 แสดงข้อมูลกำลังไฟฟ้าและสถานะของ Generator ตัวที่ 2 รอบที่ 1

t	P_2	U_i^t	$F(P_2) - \lambda P_2$
1	625	0	-571.875
2	583.333	0	-420.483
3	416.667	0	79.1667
4	333.333	0	266.667
5	500	0	-150
6	1250	1	-3607.5
7	2083.33	1	-7857.5
8	2916.667	1	-12107.5

การคำนวณกำลังไฟฟ้าของ generator ตัวที่ 3 รอบที่ 1 ทั้งหมด 8 ชั่วโมง

$$\frac{\partial\, q(\lambda)}{\partial P_3} = \frac{\partial(550 + 9P_3 + 0.002P_3^2 - \lambda_t P_3)}{\partial P_3} = 9 + 0.004P_3 - \lambda_t$$

ดังนั้น $P_3 = \frac{\lambda_t - 9}{0.004}$

ตารางที่ 2.9 แสดงข้อมูลกำลังไฟฟ้าและสถานะของ Generator ตัวที่ 3 รอบที่ 1

t	P_3	U_i^t	$F(P_3) - \lambda P_3$
1	687.5	0	-395.3125
2	625	0	-231.25
3	375	0	268.75
4	250	0	425
5	500	0	50
6	1625	1	-6200
7	2875	1	-6950
8	4125	1	-10700

การคำนวณกำลังไฟฟ้าของ generator ตัวที่ 4 รอบที่ 1 ทั้งหมด 8 ชั่วโมง

$$\frac{\partial\, q(\lambda)}{\partial P_4} = \frac{\partial(600 + 8P_4 + 0.004P_4^2 - \lambda_t P_4)}{\partial P_4} = 8 + 0.008P_4 - \lambda_t$$

ดังนั้น $P_4 = \frac{\lambda_t - 8}{0.008}$

ตารางที่ 2.10 แสดงข้อมูลกำลังไฟฟ้าและสถานะของ Generator ตัวที่ 4 รอบที่ 1

t	P_4	U_i^t	$F(P_4) - \lambda P_4$
1	468.75	0	-278.90625
2	437.5	0	-165.625
3	312.5	0	209.375
4	250	0	350
5	375	0	37.5
6	937.5	1	-5185
7	1562.5	1	-5835
8	2187.5	1	-9085

การคำนวณกำลังไฟฟ้า generator ตัวที่ 5 รอบที่ 1 ทั้งหมด 8 ชั่วโมง

$$\frac{\partial\, q(\lambda)}{\partial P_5} = \frac{\partial(500 + 7P_5 + 0.008P_5^2 - \lambda_t P_5)}{\partial P_5} = 7 + 0.016P_5 - \lambda_t$$

ดังนั้น $P_5 = \frac{\lambda_t - 7}{0.016}$

ตารางที่ 2.11 แสดงข้อมูลกำลังไฟฟ้าและสถานะของ Generator ตัวที่ 5 รอบที่ 1

t	P_5	U_i^t	$F(P_5) - \lambda P_5$
1	296.875	0	-205.0781
2	281.25	0	-132.8125
3	281.75	0	7.8125
4	187.5	0	218.75
5	250	0	0
6	531.25	1	-3750
7	843.75	1	-4250
8	1156.25	1	-6750

การคำนวณกำลังไฟฟ้าของ generator ตัวที่ 6 รอบที่ 1 ทั้งหมด 8 ชั่วโมง

$$\frac{\partial\, q(\lambda)}{\partial P_6} = \frac{\partial(400 + 6P_6 + 0.005P_6^2 - \lambda_t P_6)}{\partial P_6} = 6 + 0.010P_6 - \lambda_t$$

ดังนั้น $P_6 = \frac{\lambda_t - 6}{0.010}$

ตารางที่ 2.12 แสดงข้อมูลกำลังไฟฟ้าและสถานะของ Generator ตัวที่ 6 รอบที่ 1

t	P_6	U_i^t	$F(P_6) - \lambda P_6$
1	575	1	-1100
2	550	1	-1000
3	450	1	-600
4	400	1	-400
5	500	1	-800
6	950	1	-4200
7	1450	1	-4600
8	1950	1	-6600

การคำนวณกำลังไฟฟ้าของ generator ตัวที่ 7 รอบที่ 1 ทั้งหมด 8 ชั่วโมง

$$\frac{\partial\, q(\lambda)}{\partial P_7} = \frac{\partial(400 + 9P_7 + 0.007P_7^2 - \lambda_t P_7)}{\partial P_7} = 9 + 0.014P_7 - \lambda_t$$

ดังนั้น $P_7 = \frac{\lambda_t - 6}{0.014}$

ตารางที่ 2.13 แสดงข้อมูลกำลังไฟฟ้าและสถานะของ Generator ตัวที่ 7 รอบที่ 1

t	P_7	U_i^t	$F(P_7) - \lambda P_7$
1	229.167	0	84.8958
2	208.33	0	139.5833
3	125	0	306.25
4	83.333	0	358.4833
5	166.667	0	233.3003
6	541.667	1	-2540
7	958.33	1	-2890
8	1375	1	-4640

การคำนวณค่า Start Up Cost

Unit 1 Start Up 0 ครั้ง ดังนั้น Start Up Cost = 1000✕0 = 0 $

Unit 2 Start Up 1 ครั้ง ดังนั้น Start Up Cost = 900✕1 = 900 $

Unit 3 Start Up 1 ครั้ง ดังนั้น Start Up Cost = 800✕1 = 800 $

Unit 4 Start Up 1 ครั้ง ดังนั้น Start Up Cost = 700✕1 = 700 $

Unit 5 Start Up 1 ครั้ง ดังนั้น Start Up Cost = 600✕1 = 600 $

Unit 6 Start Up 0 ครั้ง ดังนั้น Start Up Cost = 500✕0 = 0 $

Unit 7 Start Up 1 ครั้ง ดังนั้น Start Up Cost = 400✕1 = 400 $

จะได้ค่า Start UP Cost ทั้งหมด 3400 $

คำนวณหาค่า $q^*(\lambda)$ จากสมการ

$$q(\lambda) = \sum_{t=1}^{T}\sum_{i=1}^{N}[F_i(P_{it}) - \lambda_t P_{it}] \cdot U_{it} + ST_{it}U_{it} + \lambda_t P_{Dt}$$

$[F_i(P_{it}) - \lambda_t P_{it}]$ คิดเฉพาะโรงที่เปิด

$$q(\lambda) = 40883.75\ \$$$

เมื่อทำการหาค่า $q(\lambda)$ จะได้ผลลัพธ์ dual solution($q^*(\lambda)$) ซึ่งไม่ใช่คำตอบที่แท้จริงของปัญหา โดยจะทำการคำนวณต่อไปโดยพิจารณาเครื่องกำเนิดไฟฟ้าที่มีสถานะเปิด ($U_{it} = 1$) ในผลลัพธ์ของ dual solution โดยนำมาคำนวณการจ่ายโหลดอย่างประหยัดด้วยวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม และนำผลลัพธ์ที่ได้จากขั้นตอนนี้มาคำนวณต้นทุนในการเดินเครื่องกำเนิดไฟฟ้าที่แท้จริง โดยเรียกว่า primal solution (J^*)

การคำนวณหาค่า primal solution (J^*)

ขั้นตอนนี้จะใช้วิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม ในการหาคำตอบ โดยการกำหนดค่าความต้องการไฟฟ้าและขอบเขตของเครื่องกำเนิดไฟฟ้า จากนั้นจะทำการหาค่ากำลังไฟฟ้าโดยการสุ่มตัวเลข ต่อมาจะทำการปรับค่าให้อยู่ในขอบเขตที่กำหนด เมื่อได้กำลังไฟฟ้าที่เหมาะสม จะนำมาแทนค่าในสมการเชื้อเพลิงคือ

$$F(P_i) = \alpha_i + \beta_i P_i + \gamma_i P_i^2$$

ในตัวอย่างนี้ เราจะใช้การหาค่ากำลังไฟฟ้าที่เหมาะสมจากโปรแกรม

ตารางที่ 2.14 สถานะเครื่องกำเนิดไฟฟ้า(U_t) 8 ชั่วโมง

t	U_1	U_2	U_3	U_4	U_5	U_6	U_7
1	1	0	0	0	0	1	0
2	1	0	0	0	0	1	0
3	1	0	0	0	0	1	0
4	1	0	0	0	0	1	0
5	1	0	0	0	0	1	0
6	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1

ตารางที่ 2.15 ค่ากำลังไฟฟ้าเหมาะสมที่ได้จากวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม

t	Unit 1	Unit 2	Unit 3	Unit 4	Unit 5	Unit 6	Unit 7
1	575	0	0	0	0	400	0
2	550	0	0	0	0	400	0
3	450	0	0	0	0	400	0
4	400	0	0	0	0	400	0
5	500	0	0	0	0	400	0
6	200	150	100	100	80	400	320
7	670	150	100	100	80	400	350
8	1000	320	100	100	80	400	350

นำค่ากำลังไฟฟ้าทั้งหมดมาคำนวณ primal solution

$$J^* = \sum_{t=1}^{T}\sum_{i=1}^{N}\left[F_i(P_{it}) + \left(1 - U_{i,t-1}\right)ST_{it}\right]U_{it}$$

โดยแทนค่ากำลังไฟฟ้าลงในสมการเชื้อเพลิงในการเดินเครื่องทั้ง 8 ชั่วโมงรวมกับค่า Start Up Cost จะได้ราคาต้นทุนในการเดินเครื่องกำเนิดไฟฟ้าหรือ primal solution เท่ากับ 112569.725 $ และหลังจากนั้นก็จะคำนวณค่า RDG

$$RDG = \frac{J^* - q^*}{q^*} = \frac{112569.725 - 40883.75}{40883.75} = 1.7534$$

ซึ่งค่า RDG ที่ได้ยังมีค่ามาก แสดงว่าคำตอบยังไม่ใกล้เคียงจึงต้องทำการหาในรอบถัดไป

2.5 เอกสารที่เกี่ยวข้องกับโครงงาน

S.A. Kazarlis, A.G. Bakirtzis และ V. Petridis: Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Greece ได้ทำการศึกษา และทำการวิจัยเรื่อง “A Genetic Algorithm Solution to the Unit Commitment” โดยงานวิจัยนี้มีวัตถุประสงค์เพื่อนำเสนอการการแก้ไขปัญหาการวางแผนเดินเครื่องกำเนิดไฟฟ้า โดยหาค่าที่เหมาะสมจากกระบวนการเจเนติกอัลกอริทึม โดยงานวิจัยชิ้นนี้ได้ทำการทดลอง และคำนวณกับโรงไฟฟ้าจำนวน 100 ยูนิตขึ้นไป และผลการวิจัยพบว่าในการคำนวณหาคำตอบโดยวิธีเจนเนติกอัลกอริทึม ผลลัพธ์ที่ได้แสดงให้เห็นพบว่าค่าที่เหมาะสมที่สุด(S.A. Kazarlis, 1996)

Chuan-Ping Cheng, Chih-Wen Liu, และ Chun-Chang Liu ได้ทำการศึกษาทำวิจัยเรื่อง “Unit Commitment by Lagrangian Relaxation and Genetic Algorithm” โดยงานวิจัยนี้มีวัตถุประสงค์เพื่อนำเสนอการแก้ปัญหาการวางแผนเครื่องเดินไฟฟ้าโดยวิธีเจเนติกอัลกอริทึมในกระบวนการลากรางคีรีแล็กซ์ เพื่อเป็นการปรับปรุงประสิทธิภาพในการหาคำตอบ ที่ใช้ในการแก้ไขปัญหาวางแผนเครื่องไฟฟ้า และผลการวิจัยพบว่าผลลัพธ์ในกระบวนการหาคำตอบของวิธีเจเนติกอัลกอริทึมในกระบวนการลากรางคีรีแล็กซ์เซชั่น ในการแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้า ส่งผลให้ผลลัพธ์ในการแก้ไขปัญหาลู่เข้าสู่คำตอบ โดยคำตอบที่ได้มีประสิทธิภาพเพิ่มมากขึ้น เมื่อนำค่าผลลัพธ์ในการหาคำตอบ โดยวิธีเจนเนติกอัลกอริทึม ลากรางคีรีแล็กซ์เซชั่น และเจเนติกอัลกอริทึมในกระบวนการลากรางคีรีแล็กซ์เซชั่น โดยผลสรุปที่ได้ออกมาคือวิธีการหาคำตอบโดยวิธีเจเนติกอัลกอริทึมในกระบวนการลากรางคีรีแล็กซ์เซชั่น ได้ผลลัพธ์ที่ดีที่สุด(C.P. Cheng, 2000)

T. Sum-Im และ W. Ongsakul ได้ทำการศึกษาทำวิจัยเรื่อง “Ant Colony Search Algorithm for unit Commitment” โดยงานวิจัยนี้มีวัตถุประสงค์เพื่อแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้า โดยแก้ปัญหาด้วยวิธีอาณาจักรของมดเป็นการสังเกตพฤติกรรมของฝูงมด วิธีอาณาจักรของมดเป็นเหมือนการใช้มดในการแก้ไขปัญหาที่ดีที่สุดของการแก้ไขปัญหาการวางแผนเดินเครื่องกำเนิดไฟฟ้าของโรงไฟฟ้าพลังงานความร้อน วิธีอาณาจักรของมดเป็นการหาคำตอบแบบขนาน โดยในการแก้ไขปัญหาจะแบ่ง 2 ปัญหาย่อย ได้แก่ การวางแผนเดินเครื่องกำเนิดไฟฟ้าโดยใช้วิธีอาณาจักรของมด และปัญหาการส่งจ่ายไฟฟ้าอย่างประหยัดโดยใช้วิธีหาแลมด้า ในการแก้ไขปัญหากับโรงไฟฟ้าจำนวนทั้งหมด 10 ยูนิต โดยการวางแผน แก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าเป็นการลดค่าใช้จ่ายของค่าต่างๆ ซึ่งหลังจากนั้นก็ได้นำค่าที่ได้จากการแก้ไขปัญหาโดยวิธีอาณาจักรของมดเปรียบเทียบกับวิธีการอื่นๆ ผลที่ได้ออกมาคือวิธีการแก้ไขปัญหาโดยวิธีอาณาจักรของมดมีค่าใช้จ่ายในการผลิตที่เหมาะสมที่สุด(T. Sum-Im, 2003)

Thanathip Sum-Im ได้ทำการศึกษาทำวิจัยเรื่อง “Lagrangian Relaxation Combined with Differential Evolution Algorithm for Unit Commitment Problem” โดยงานวิจัยนี้มีจุดประสงค์เพื่อแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้า โดยวิธีการแก้ไขปัญหาเป็นการรวมกันของกระบวนการลากรางคีรีแล็กซ์ และดิฟเฟอเรนเชียลอีโวลูชั่นอัลกริทึม สำหรับโรงไฟฟ้าพลังงานความร้อน ซึ่งข้อดีของการใช้วิธีดิฟเฟอเรนเชียลอีโวลูชั่นอัลกอริทึม เป็นวิธีแก้ไขปัญหาแบบขนาน โดยการวางแผนเดินเครื่องโรงไฟฟ้าถูกกำหนดว่า ต้องมีต้นทุนในการผลิตที่ต่ำที่สุด ซึ่งต้นทุนนั้นได้แก่ ค่าในการสตาร์ทโรงไฟฟ้า ราคาเชื้อเพลิง และรวมถึงความแตกต่างของโรงไฟฟ้า เช่น กำลังไฟฟ้าสมดุล กำลังผลิตสำรอง เงื่อนของเครื่องกำเนิดไฟฟ้า ในการแก้ไขปัญหานี้ ในการแก้ไขปัญหานี้จะเป็นแก้ไขปัญหาของระบบที่มีทั้งหมด 10 ยูนิตโดยมีการเปรียบเทียบกับค่าผลลัพธ์กับการหาคำตอบของวิธีการต่างๆด้วย จากการทำวิจัยพบว่าวิธีการแก้ไขปัญหาด้วยวิธีการรวมกันของกระบวนการลากรางคีรีแล็กซ์ และดิฟเฟอเรนเชียลอีโวลูชั่นอัลกอริทึม ในการแก้ไขปัญหาวางแผนเดินเครื่องโรงไฟฟ้า พบว่าค่าใช้จ่ายในการผลิตที่ดีที่สุดในบรรดาการแก้ปัญหาโดยใช้วิธีอื่น (Thanathip Sum-Im, 2014)

บทที่ 3

ขั้นตอนและวิธีการดำเนินงาน

3.1 ขั้นตอนการทำงาน

ในการศึกษาและจัดทำโครงงานนี้ได้แบ่งขั้นตอนในการดำเนินงานดังนี้

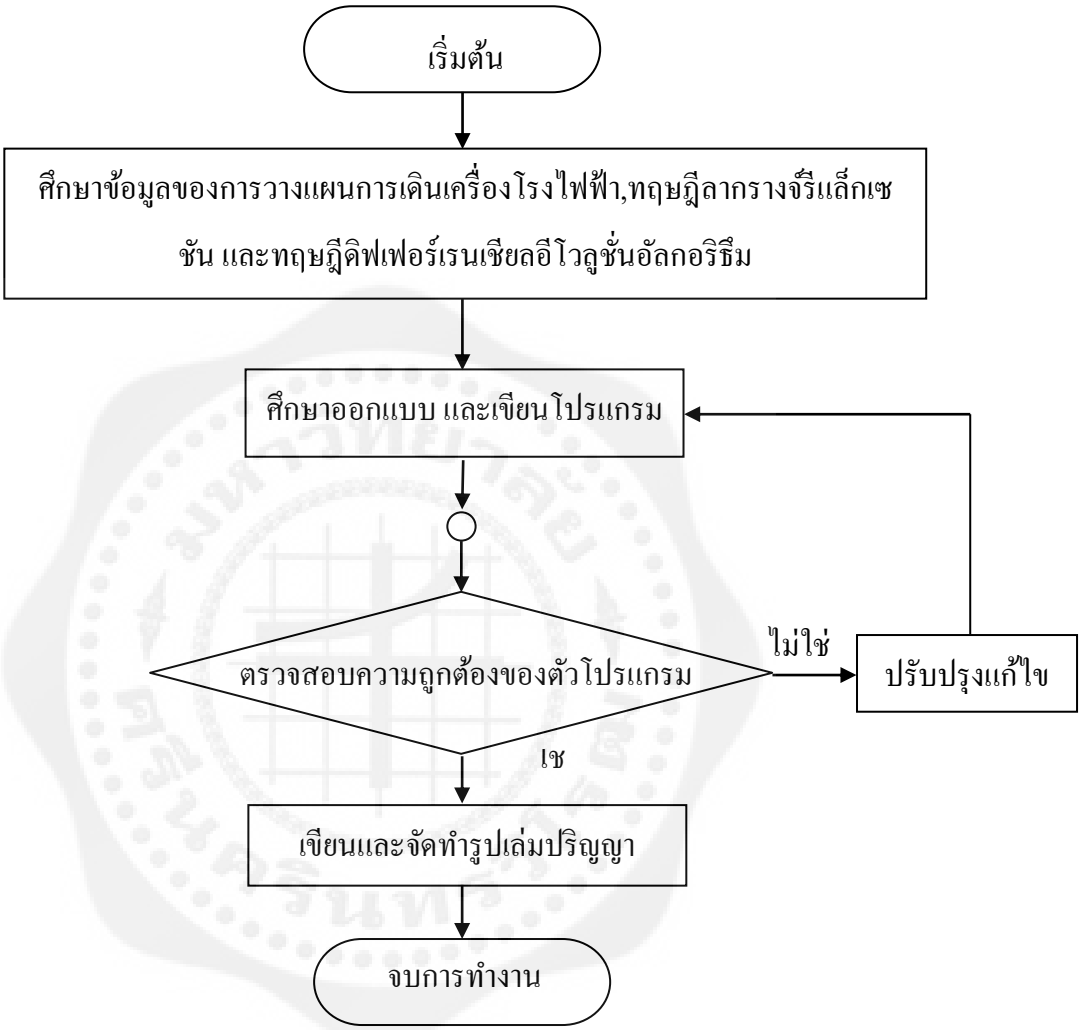
3.1.1 ศึกษาเกี่ยวกับเอกสารที่เกี่ยวข้องกับการคำนวณค่าการวางแผนการเดินเครื่องโรงไฟฟ้าจากหนังสือ และเอกสารต่างๆ

3.1.2 ศึกษาการทำงานของวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับค่าได้

3.1.3 ศึกษา ออกแบบ และเขียนโปรแกรมคอมพิวเตอร์

3.1.4 วิเคราะห์ ตรวจสอบความถูกต้อง และสรุปผลที่ได้จากการศึกษา

3.1.5 เขียนโครงงาน และจัดทำรูปเล่ม



รูปที่ 3.1 แผนผังแสดงขั้นตอนการทำงาน

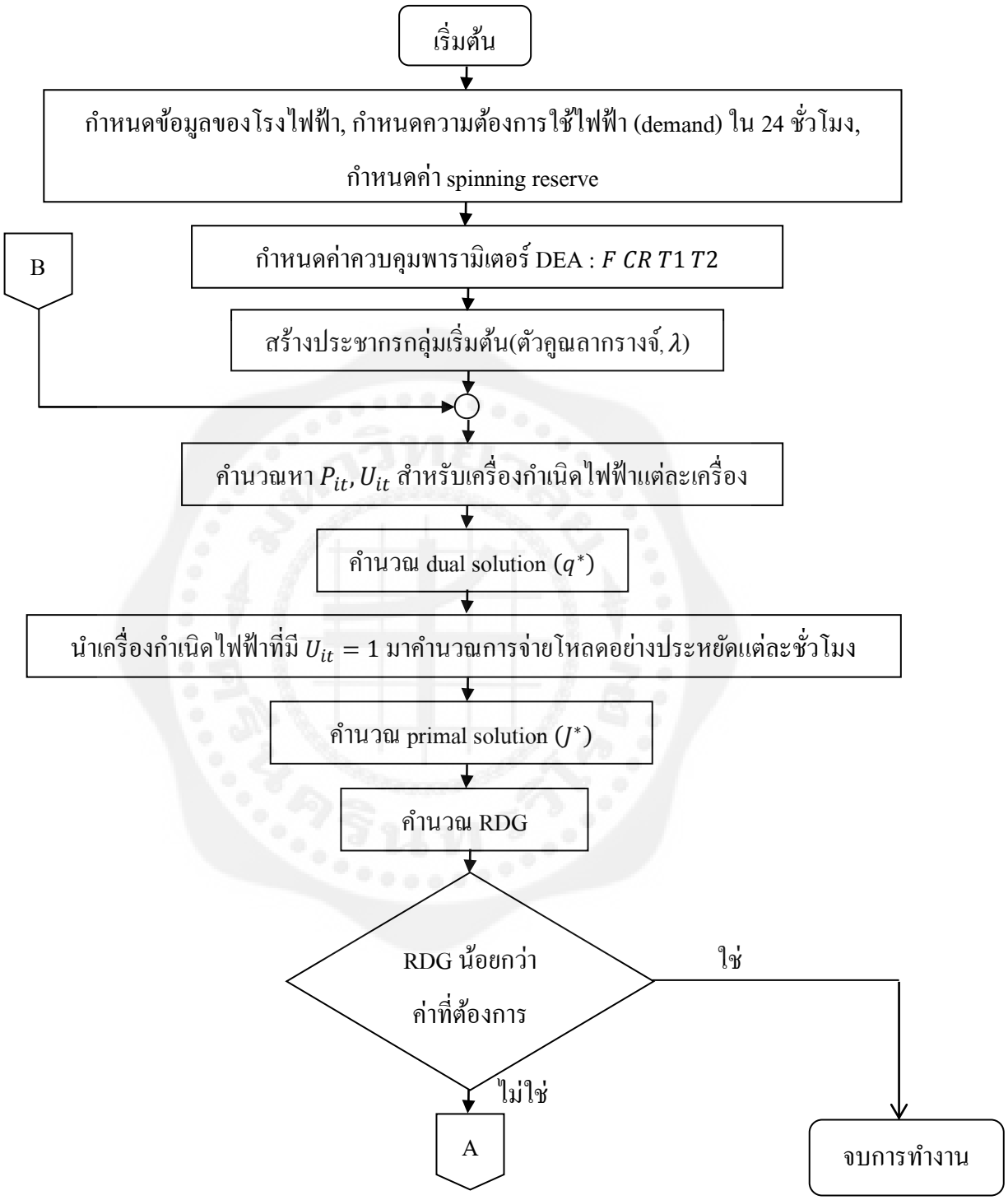
3.2 แผนผังการดำเนินการ

ตารางที่ 3.1 ตารางแสดงระยะเวลาดำเนินงาน

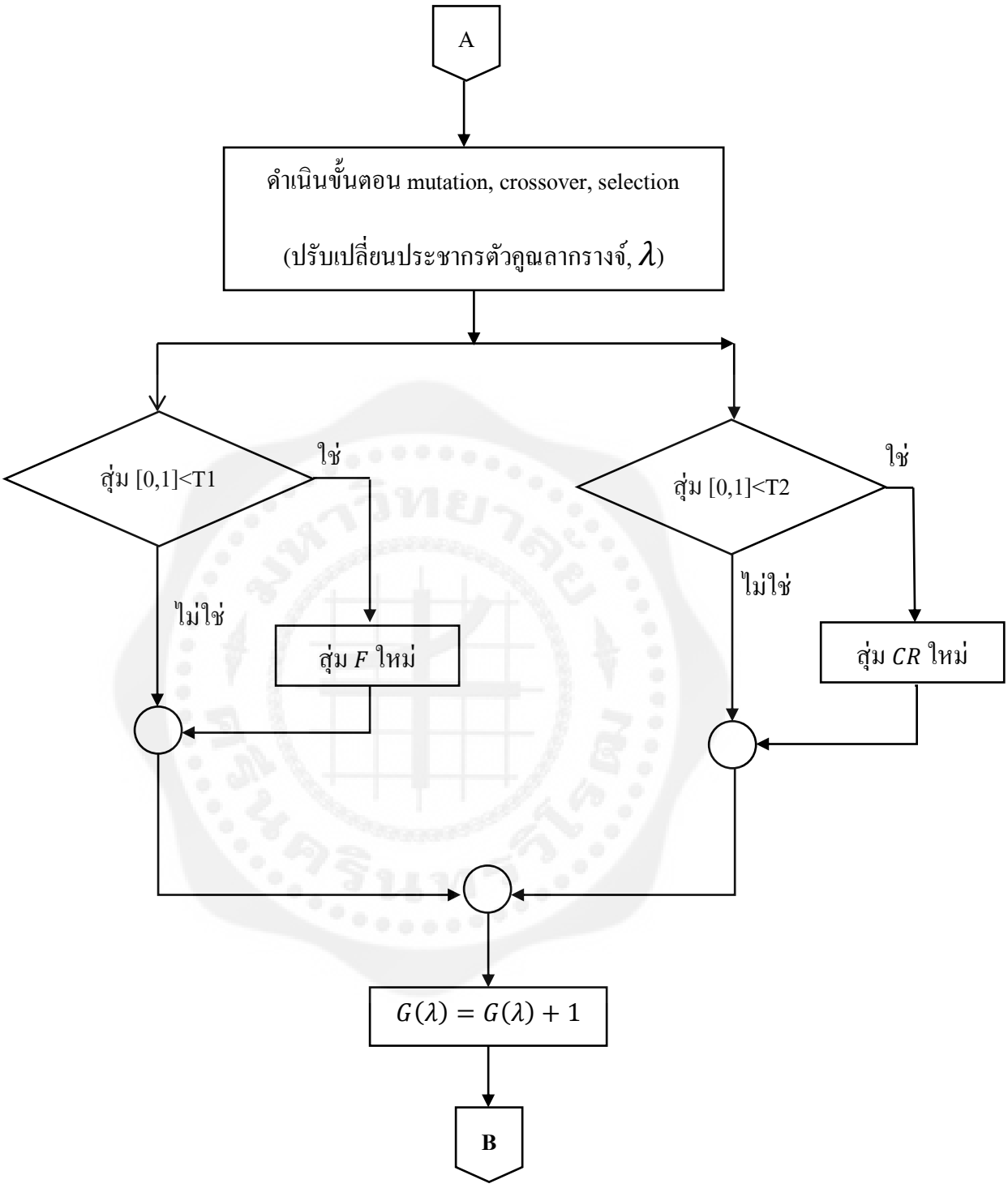
ขั้นตอน	ระยะเวลา (เดือน)								
	ส.ค. 2557	ก.ย. 2557	ต.ค. 2557	พ.ย. 2557	ธ.ค. 2557	ม.ค. 2557	ก.พ. 2557	มี.ค. 2557	เม.ย. 2557
นำเสนอหัวข้อโครงงาน									
ศึกษาข้อมูลทางทฤษฎี									
ออกแบบ วิเคราะห์ และปรับปรุงโปรแกรม									
สรุปผล และจัดทำรูปเล่มโครงงานวิศวกรรม									

3.3 แสดงขั้นตอนและวิธีการดำเนินงาน

โครงงานนี้ทำการศึกษาการแก้ปัญหาการวางแผนการเดินเครื่องกำเนิดไฟฟ้า โดยการเขียนโปรแกรม ด้วยโปรแกรม MATLAB R2014a โดยใช้วิธีลากรางจรีแล็กเซชันร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึม โดยมีหลักการคิด และการทำงานของโปรแกรมดังแสดงในรูปที่ 3.2



รูปที่ 3.2 แผนผังหลักการคิดโปรแกรม



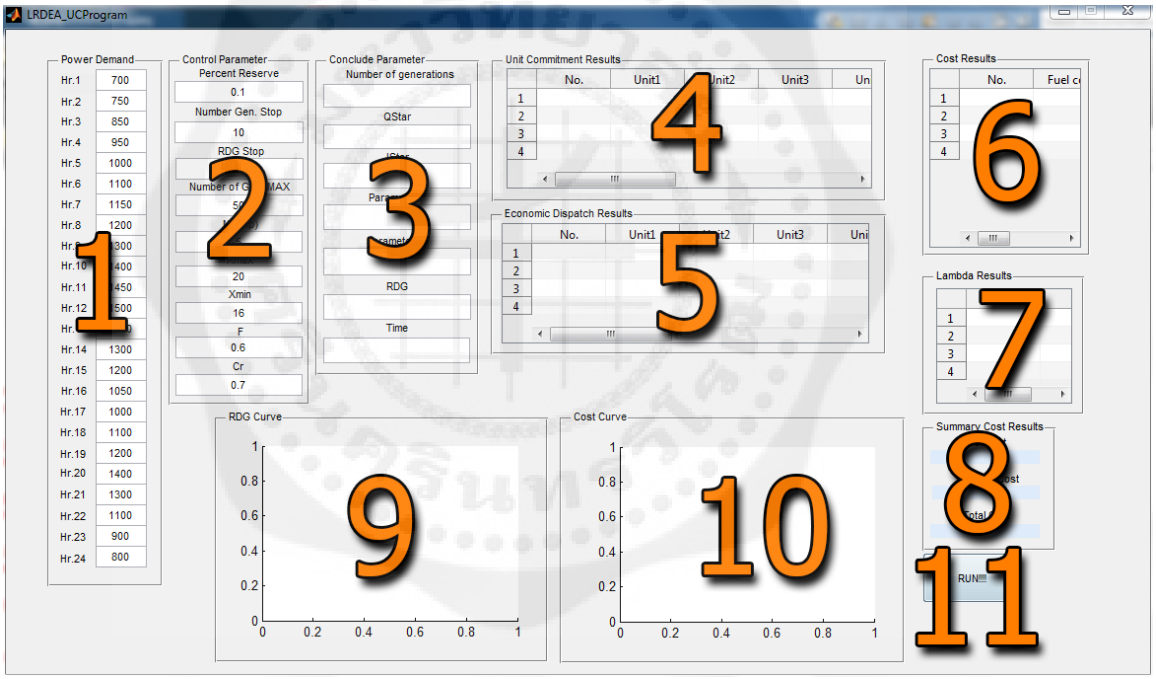
รูปที่ 3.2 แผนผังหลักการคิดโปรแกรม(ต่อ)

3.4 การกำหนดปัญหา

การกำหนดปัญหาจะใช้ วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้ ในการหาตารางการเดินเครื่องกำเนิดไฟฟ้า โดยเอาต์พุตแต่ละยูนิตให้ราคาต้นทุนต่ำที่สุด ขณะที่ค่า Fitness Function พิจารณาจาก RDG และ primal solution

ในส่วนการหากำลังการผลิตจะทำการหาการจ่ายโหลดอย่างประหยัดด้วย DEA เพื่อให้ได้ราคาต้นทุนที่มีค่าต่ำที่สุด โดยที่ค่ากำลังไฟฟ้าเอาต์พุตนั้นจะต้องผ่านเงื่อนไขข้อจำกัดดังนี้ คือ กำลังไฟฟ้าสมดุล กำลังการผลิตสำรอง และขอบเขตการผลิตในแต่ละหน่วย

3.5 วิธีการใช้หน้าต่างของโปรแกรม LR-DEA UC



รูปที่ 3.3 หน้าต่างโปรแกรมสำหรับการแก้ไขปัญหา

ในส่วนของการใช้หน้าต่างโปรแกรม LR-DEA UC มีวิธีการใช้โปรแกรมในแต่ละส่วนดังรายละเอียดต่อไปนี้

หมายเลข 1 ส่วนสำหรับป้อนค่าความต้องการใช้กำลังไฟฟ้าในแต่ละชั่วโมง

หมายเลข 2 ส่วนสำหรับป้อนค่า Parameter ต่างๆ ได้แก่ กำลังการผลิตสำรอง (Percent of Power Reserve), จำนวนรอบที่จะหยุดการคำนวณเมื่อไม่เจอค่าใหม่ (Max. Repeat

Generation), เกณฑ์ในการหยุดการคำนวณ (RDG Criteria), จำนวนรอบสูงสุดในการคำนวณ (Max. Generation), ตัวคูณ Parameter (Mul(Np)), ขอบขาบน (X_{max}), ขอบขาล่าง (X_{min}), Scaling Mutation Factor (F) และ Crossover Constant (CR)

หมายเลข 3 ส่วนแสดงข้อมูลสรุปของผลลัพธ์ในการคำนวณมี 7 ค่า ได้แก่ จำนวนรอบในการคำนวณ (Number of Generations), Dual solution (q^*), Primal Solution (J^*), Scaling Mutation Factor (F), Crossover Constant (CR), เกณฑ์ในการหยุดการคำนวณ (RDG) และเวลาที่ใช้ในการคำนวณ (Time)

หมายเลข 4 ส่วนแสดงตารางผลลัพธ์สถานะของโรงไฟฟ้าที่ได้จากการแก้ไขปัญหา UC โดยการแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้

หมายเลข 5 ส่วนแสดงตารางผลลัพธ์กำลังไฟฟ้าของโรงไฟฟ้าที่ได้จากการแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอลิธึมแบบปรับตัวเองได้

หมายเลข 6 ส่วนแสดงตารางผลลัพธ์ราคาต้นทุนการผลิตรายชั่วโมง ซึ่งประกอบด้วย Fuel Cost, Startup Cost และ Total Cost ที่ได้จากการวางแผนเดินเครื่องโรงไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้

หมายเลข 7 ส่วนแสดงค่า λ ที่ใช้ในแต่ละชั่วโมง ที่ได้จากกระบวนการ DEA ที่ใช้การวางแผนเดินเครื่องโรงไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้ในรอบนั้น

หมายเลข 8 ส่วนแสดงผลลัพธ์ของข้อมูล 3 ค่า คือ Fuel Cost, Startup Cost และ Total Cost

หมายเลข 9 ส่วนแสดงกราฟ Relative Duality Gap (RDG) กับจำนวนรอบในการคำนวณ

หมายเลข 10 ส่วนแสดงกราฟ ต้นทุนในการผลิต กับจำนวนรอบในการคำนวณ

หมายเลข 11 ปุ่มกดสำหรับให้โปรแกรมทำงาน

3.6 รายละเอียดของเครื่องคำนวณที่ใช้

สำหรับการสร้างโปรแกรมการคำนวณการวางแผนเดินเครื่องโรงไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้ และการทดสอบปัญหาผู้จัดทำได้ใช้คอมพิวเตอร์ ซึ่งมีคุณลักษณะดังตารางที่ 3.2

ตารางที่ 3.2 คุณลักษณะคอมพิวเตอร์ที่ใช้ในโครงงาน

หัวข้อ	รายละเอียด
รุ่น	DELL INSPIRON N5110
CPU	Intel® Core™ i7-2630 QM CPU @ 2.00GHz
ระบบประมวลผล	Window 7 Ultimate
หน่วยความจำ (RAM)	6.00 GB
โปรแกรมคำนวณ	MATLAB 2014a

บทที่ 4

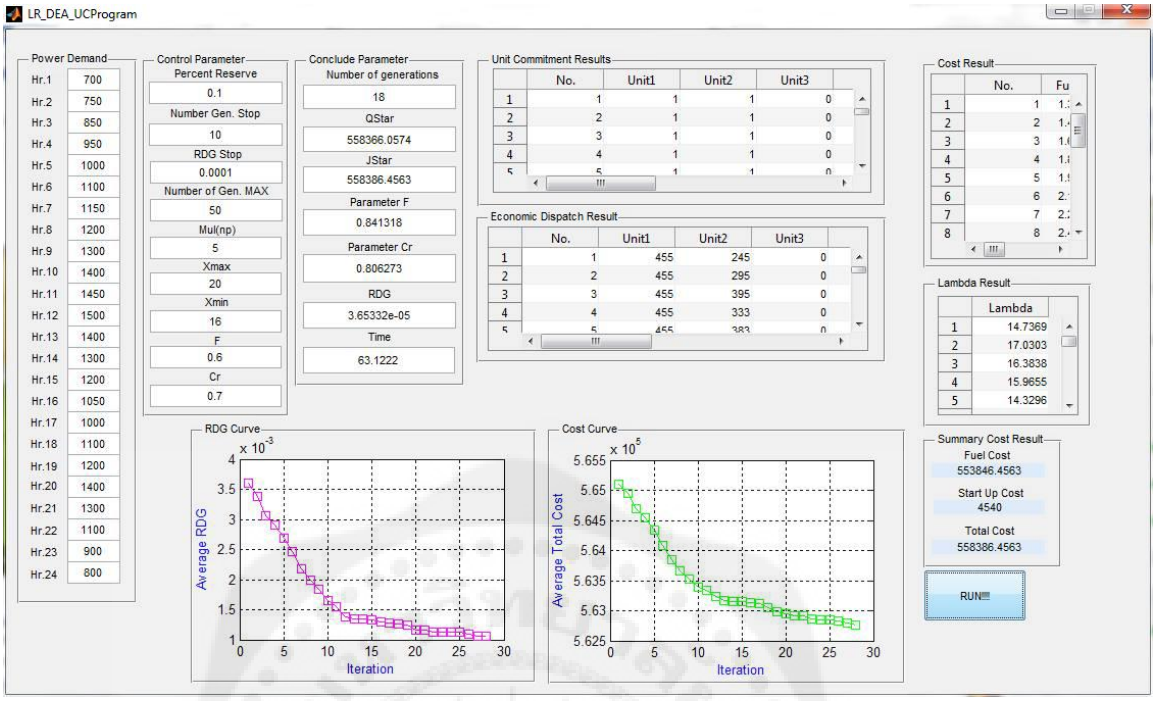
ผลการทดลอง

สำหรับการทำโครงงานวิศวกรรมนี้ โปรแกรมที่เลือกใช้งานคือโปรแกรม MATLAB 2014a เพื่อใช้ประยุกต์ในการแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้

ระบบที่ใช้ทดสอบ คือ เครื่องกำเนิดไฟฟ้าจำนวน 10 เครื่อง แสดงในภาคผนวก ก ดังตารางที่ ก1.1 ระยะเวลาการทำงาน 24 ชั่วโมง ซึ่งมีโหลดรายชั่วโมงดังนี้ ชั่วโมงที่ 1 โหลดเท่ากับ 700 MW ชั่วโมงที่ 2 โหลดเท่ากับ 750 MW ชั่วโมงที่ 3 โหลดเท่ากับ 850 MW ชั่วโมงที่ 4 โหลดเท่ากับ 950 MW ชั่วโมงที่ 5 โหลดเท่ากับ 1000 MW ชั่วโมงที่ 6 โหลดเท่ากับ 1100 MW ชั่วโมงที่ 7 โหลดเท่ากับ 1150 MW ชั่วโมงที่ 8 โหลดเท่ากับ 1200 MW และตั้งแต่ชั่วโมงที่ 9 ถึงชั่วโมงที่ 24 มีโหลดรายชั่วโมงตามภาคผนวก ก ในตารางที่ ก1.2 ในหัวข้อต่อไป จะแสดงขั้นตอนการทำงานของโปรแกรมการคำนวณเพื่อแก้ปัญหาการวางแผนการเดินเครื่องกำเนิดไฟฟ้า โดยจะแสดงผลการทดสอบโปรแกรมที่ค่าความต้องการกำลังไฟฟ้าของโหลดตั้งแต่ชั่วโมงที่ 1 ถึง 24

4.1 ผลของการทดลองในการแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้า

จากการทดลองโดยโปรแกรมที่ทำการออกแบบ ในการแก้ไขปัญหาวางแผนเดินเครื่องโรงไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้ โดยใช้เครื่องกำเนิดไฟฟ้าจำนวน 10 ยูนิต ทำงานเป็นเวลาเวลา 24 ชั่วโมง



รูปที่ 4.1 แสดงหน้าต่างการทำงานของ GUI ของการแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้า

ตารางที่ 4.1 แสดงผลของค่าใช้จ่ายในแต่ละรอบการคำนวณสำหรับการเดินเครื่องโรงไฟฟ้าจำนวน 100 รอบ

Iterations	Total Cost($)	RDG (10^{-5})	Time (sec)
1	560647.788	2.29	45.7609
2	561830.7551	3.69929	42.5477
3	561240.4941	8.32109	32.6312
4	560994.4016	3.25	38.5855
5	561992.0296	3.68958	37.9628
6	561240.4941	2.77	45.9136
7	560187.4905	2.090	49.4355
8	559163.0084	5.76	48.0043
9	561240.4941	2.45998	57.4123

ตารางที่ 4.1(ต่อ) แสดงผลของค่าใช้จ่ายในการผลิตในแต่ละรอบการคำนวณสำหรับการเดินเครื่องโรงไฟฟ้าฟ้าจำนวน 100 รอบ

Iterations	Total Cost($)	RDG (10^{-5})	Time (sec)
10	561240.4941	3.58819	39.3241
11	561240.4941	9.55	30.1046
12	559915.4819	7.34893	30.3228
13	561420.5628	8.36	60.5192
14	560647.788	5.01207	54.4586
15	561240.4941	5.2	38.7312
16	561992.0296	2.28926	35.8544
17	561420.5628	2.63	19.7624
18	561240.4941	8.72	56.7437
19	560647.788	9.27772	49.125
20	560187.4905	6.14917	34.2899
21	560187.4905	4.36195	40.1146
22	561240.4941	5.97	47.3542
23	560616.6741	9.14	29.7187
24	561420.5628	2.42	29.0449
25	560647.788	8.18	54.1331
26	560616.6741	9.62188	44.7123
27	560187.4905	3.43	39.7077
28	560187.4905	9.85482	44.367
29	561420.5628	2.26	32.6656
30	561240.4941	2.19763	50.6608
31	561240.4941	7.07355	34.0475
32	560647.788	8.8	48.3645
33	560187.4905	7.42	35.3359
34	561240.4941	8.71027	60.8426
35	561240.4941	1.53854	40.3591

ตารางที่ 4.1(ต่อ) แสดงผลของค่าใช้จ่ายในการผลิตในแต่ละรอบการคำนวณสำหรับการเดินเครื่องโรงไฟฟ้าฟ้าจำนวน 100 รอบ

Iterations	Total Cost($)	RDG (10^{-5})	Time (sec)
36	561240.4941	2.31905	39.1629
37	560396.0806	4.3	41.5966
38	561240.4941	9.28	31.0641
39	561240.4941	2.17	37.7807
40	561240.4941	1.5	45.5257
41	561240.4941	7.37	38.5497
42	561420.5628	6.68	37.1344
43	561240.4941	5.4	32.1541
44	560187.4905	6.34	61.3318
45	560187.4905	3.7	83.6354
46	561240.4941	2.43	27.3935
47	561240.4941	9.17	46.6287
48	561240.4941	8.64	40.6055
49	560396.0806	1.42	60.2773
50	561240.4941	6.98	48.84
51	561420.5628	7.33	35.7511
52	561240.4941	3.67	59.1069
53	561240.4941	2.33	23.7867
54	561992.0296	6.69	56.4164
55	561240.4941	6.52	32.0681
56	561420.5628	8.64	34.7548
57	561240.4941	8.46	73.7901
58	561240.4941	8.81	47.467
59	561240.4941	6.21	42.7966
60	561240.4941	5.00	37.9953
61	561240.4941	8.75	56.4419

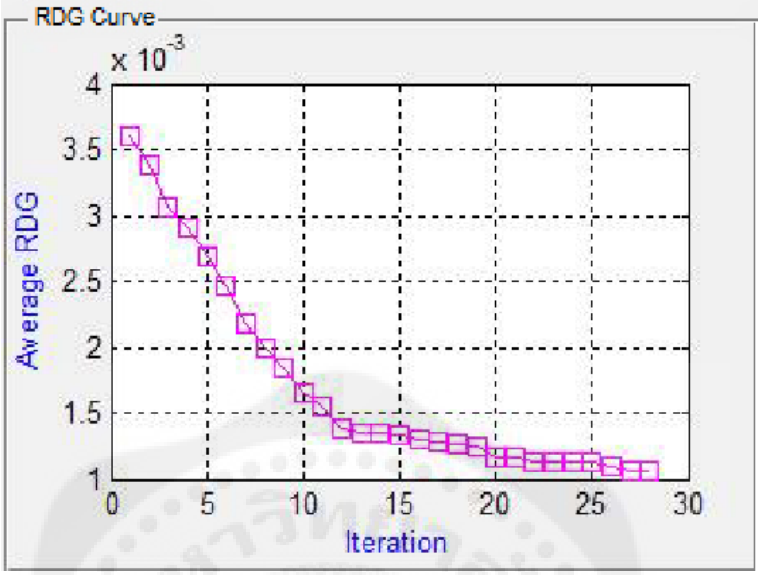
ตารางที่ 4.1(ต่อ) แสดงผลของค่าใช้จ่ายในการผลิตในแต่ละรอบการคำนวณสำหรับการเดินเครื่องโรงไฟฟ้าฟ้าจำนวน 100 รอบ

Iterations	Total Cost($)	RDG (10^{-5})	Time (sec)
62	561240.4941	6.21	40.9972
63	561420.5628	3.59	65.8417
64	561420.5628	6.56	43.8684
65	561240.4941	3.49	56.3527
66	561240.4941	4.87	49.1378
67	561240.4941	8.43	32.9789
68	560187.4905	9.16	52.0568
69	561240.4941	3.68	42.3815
70	561240.4941	2.55	33.6478
71	561420.5628	6.48	81.3371
72	558386.4563	3.65	63.1222
73	561240.4941	6.9	53.1538
74	560187.4905	1.87	61.4193
75	561240.4941	1.35	34.8853
76	560647.788	9.84	54.0138
77	560187.4905	7.16	37.2612
78	561240.4941	2.14	40.1091
79	561240.4941	1.34	50.0841
80	560187.4905	7.28	48.5686
81	561240.4941	9.66	31.6894
82	561240.4941	6.08	42.9201
83	561240.4941	1.03	49.8909
84	560982.8588	7.16	64.9791
85	559753.2694	9.42	43.7767
86	560616.6741	3.70	50.8105
87	560187.4905	1.81	71.9359

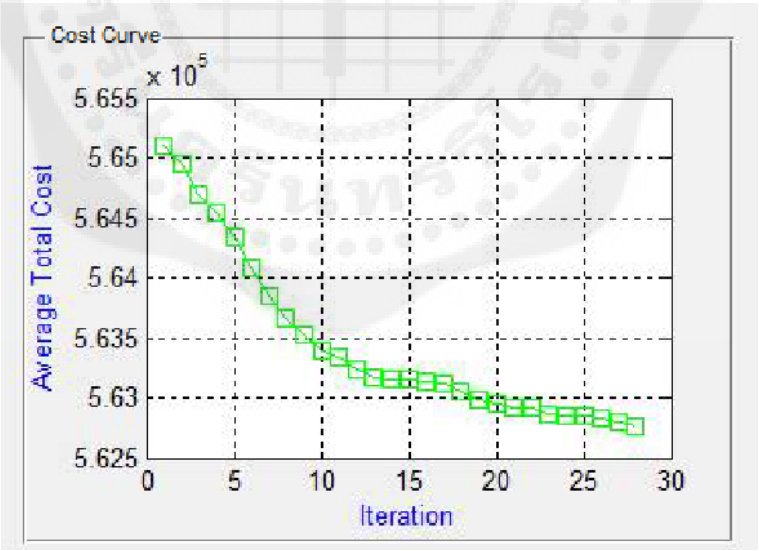
ตารางที่ 4.1(ต่อ) แสดงผลของค่าใช้จ่ายในการผลิตในแต่ละรอบการคำนวณสำหรับการเดินเครื่องโรงไฟฟ้าฟ้าจำนวน 100 รอบ

Iterations	Total Cost($)	RDG (10^{-5})	Time (sec)
88	561240.4941	6.20	47.2609
89	561240.4941	5.65	38.6265
90	561240.4941	1.74	58.2727
91	561420.5628	7.79	45.5158
92	560396.0806	8.38	40.9821
93	561240.4941	6.37	32.8813
94	560647.788	1.94	57.2167
95	561240.4941	5.04	38.1276
96	559996.8841	6.30	53.2076
97	561420.5628	9.26	21.5151
98	559405.4314	9.19	51.5852
99	560187.4905	4.03	27.8283
100	561240.4941	9.46	51.8278

จากการคำนวณทั้งหมด 100 รอบ ค่าที่ได้จากการคำนวณในรอบที่ 72 เป็นค่าที่ดีที่สุด กล่าวคือมีค่าค่าใช้จ่ายในการผลิตสำหรับการเดินเครื่องโรงไฟฟ้าที่มีค่าต่ำที่สุด แล้วเพียงพอต่อความต้องการ สำหรับการใช้ไฟฟ้า โดยสามารถแสดงผลทางการคำนวณได้ดังต่อไปนี้



รูปที่ 4.2 กราฟการลู่เข้าหาคำตอบของค่าช่องว่าง (Relative Duality Gap)ในการแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้า



รูปที่ 4.3 กราฟการลู่เข้าหาคำตอบของราคาค่าใช้จ่ายในการผลิตในการแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้า

ตารางที่ 4.2 แสดงผลของสถานะจองเครื่องกำเนิดไฟฟ้าในเวลา 24 ชั่วโมง

เวลา (ชั่วโมง)	สถานะของเครื่องกำเนิดไฟฟ้า 24 ชั่วโมง									
	Unit1	**Unit2**	**Unit3**	**Unit4**	**Unit5**	**Unit6**	**Unit7**	**Unit8**	**Unit9**	**Unit10**
1	1	1	0	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	0	0
4	1	1	0	0	1	0	0	0	0	0
5	1	1	0	0	1	0	0	0	0	0
6	1	1	0	0	1	0	0	0	0	0
7	1	1	0	0	1	0	0	0	0	0
8	1	1	1	1	1	0	0	0	0	0
9	1	1	1	1	1	0	0	0	0	0
10	1	1	1	1	1	0	0	0	0	0
11	1	1	1	1	1	0	1	0	0	0
12	1	1	1	1	1	0	1	0	0	0
13	1	1	1	1	1	0	1	0	0	0
14	1	1	1	1	1	0	0	0	0	0
15	1	1	1	1	1	0	0	0	0	0
16	1	1	1	1	1	0	0	0	0	0
17	1	1	1	1	1	0	0	0	0	0
18	1	1	1	1	1	0	0	0	0	0
19	1	1	1	1	1	0	0	0	0	0
20	1	1	1	1	1	0	0	0	0	0
21	1	1	1	1	1	0	0	0	0	0
22	1	1	1	1	1	0	0	0	0	0
23	1	1	1	1	1	0	0	0	0	0
24	1	1	1	1	1	0	0	0	0	0

ตารางที่ 4.3 แสดงผลลัพธ์กำลังไฟฟ้าของเครื่องกำเนิดไฟฟ้าแต่ละยูนิต

เวลา (ชั่วโมง)	กำลังไฟฟ้าของเครื่องกำเนิดไฟฟ้า(MW) 10 ยูนิต									
	Unit1	**Unit2**	**Unit3**	**Unit4**	**Unit5**	**Unit6**	**Unit7**	**Unit8**	**Unit9**	**Unit10**
1	455	245	0	0	0	0	0	0	0	0
2	455	295	0	0	0	0	0	0	0	0
3	455	395	0	0	0	0	0	0	0	0
4	455	333	0	0	162	0	0	0	0	0
5	455	383	0	0	162	0	0	0	0	0
6	455	455	0	0	190	0	0	0	0	0
7	455	455	0	0	240	0	0	0	0	0
8	455	455	108	20	162	0	0	0	0	0
9	455	455	130	98	190	0	0	0	0	0
10	455	455	130	130	240	0	0	0	0	0
11	455	455	130	130	162	0	0	0	0	0
12	455	455	130	130	162	0	0	0	0	0
13	455	455	130	130	230	0	0	0	0	0
14	455	455	130	98	250	0	30	0	0	0
15	455	455	108	20	250	0	80	0	0	0
16	455	393	20	20	205	0	25	0	0	0
17	455	343	20	20	162	0	0	0	0	0
18	455	443	20	20	162	0	0	0	0	0
19	455	455	108	20	162	0	0	0	0	0
20	455	455	130	130	230	0	0	0	0	0
21	455	455	130	98	162	0	0	0	0	0
22	455	443	20	20	162	0	0	0	0	0
23	455	243	20	20	162	0	0	0	0	0
24	448	150	20	20	162	0	0	0	0	0

ตารางที่ 4.4 แสดงราคาค่าใช้จ่ายในการผลิต($)สำหรับการเดินเครื่องโรงไฟฟ้าในเวลา 24 ชั่วโมง

เวลา (ชั่วโมง)	Fuel Cost ($)	Start Up Cost ($)	Total Cost ($)
1	13683.1298	0	13683.1298
2	14554.4998	0	14554.4998
3	16301.8898	0	16301.8898
4	18963.6287	1800	20763.6287
5	19837.7267	0	19837.7267
6	21689.9778	0	21689.9778
7	22760.5478	0	22760.5478
8	24626.1229	2220	26846.1229
9	26308.2153	0	26308.2153
10	28297.3008	0	28297.3008
11	30042.4198	520	30562.4198
12	31433.7648	0	31433.7648
13	28935.5120	0	28935.5120
14	26308.2153	0	26308.2153
15	24626.1229	0	24626.1229
16	22056.3763	0	22056.3763
17	21181.3343	0	21181.3343
18	22932.3343	0	22932.3343
19	24626.1229	0	24626.1229
20	28297.3008	0	28297.3008
21	26308.2153	0	26308.2153
22	22932.3343	0	22932.3343
23	19437.8023	0	19437.8023
24	17704.9280	0	17704.9280
Total	553846.4563	4540	558386.4563

ตารางที่ 4.5 แสดงค่าทางสถิติของค่าใช้จ่ายในการผลิตสำหรับการเดินเครื่องโรงไฟฟ้าในเวลา 24 ชั่วโมง

Results	Best	Worst	Average	SD
Total Cost ($)	558,386.4563	561,992.0296	560,944.3325	606.5282
RDG (10^{-5})	3.65	6.69	5.4	2.86
Calculation Time (sec)	63.1222	56.4164	45.1304	12.2262

จากตารางที่ 4.6 พบว่า ค่าข้อมูลที่ได้จากการทดลองมีค่าผลรวมของต้นทุนในการผลิตที่ดีที่สุดอยู่ที่ 558,386.4563 $ ค่าที่แย่ที่สุดคือ 561,992.0296 $ และ ค่าเวลาในการคำนวณ มีค่าที่ดีที่สุด 19.7624 วินาที ค่าที่แย่ที่สุดคือ 83.6354 วินาที ซึ่งจะเห็นได้ว่าค่าเบี่ยงเบนมาตรฐานค่อนข้างสูง เนื่องจากค่าการทดลองของผลรวมของต้นทุนในการผลิตและเวลาที่ใช้ในการคำนวณมีการกระจายตัวของข้อมูลค่อนข้างสูง ซึ่งแตกต่างจาก RDGค่าความแปรปรวนต่ำ เนื่องจากค่า RDG ได้มีการตั้ง

ตารางที่ 4.6 สรุปผลการคำนวณสำหรับการเดินเครื่องโรงไฟฟ้าที่ทำให้ค่าใช้จ่ายในการผลิตต่ำที่สุด

Value	Best total production
Dual Solution	558366.0574
Primal Solution	558386.4563
Total Cost ($)	558386.4563
Parameter F	[0.6,1]
Parameter Cr	[0.7,0.9]
RDG (10^{-5})	3.65
Calculation time (sec)	63.1222

จากตารางที่ 4.6 พบว่าผลการทดลองจากที่ดีที่สุดจากการจากการคำนวณทั้งหมด 100 รอบ ค่าที่ทำให้ค่าใช้จ่ายในการผลิตสำหรับการเดินเครื่องโรงไฟฟ้าที่ดีที่สุดคือ 558,386.4563$ ค่า RDG $3.65 \times \mathbf{10}^{-5}$ เวลาในการคำนวณคือ 63.1222 วินาที ค่าพารามิเตอร์ F ที่เหมาะสมอยู่ในช่วง [0.6,1] และค่าพารามิเตอร์ Cr ที่เหมาสมอยู่ในช่วง [0.7,0.9] ซึ่งถือว่าเป็นค่าที่ดีมากเลยทีเดียว

ตารางที่ 4.7 ตารางเปรียบเทียบของการแก้ไขปัญหาด้วยวิธีต่างๆ

Method	Best total production Cost ($)
LR (S.A. Kazarlis, 1996)	565,825
GA (S.A. Kazarlis, 1996)	565,825
LRGA (C.P. Cheng, 2000)	564,800
EP (K.A. Juste, 1999)	564,551
ACSA (T. Sum-Im, 2003)	564,049
LR-DEA (Thanathip Sum-Im, 2014)	563,946
LR-SaDEA	558,368

จากตารางที่ 4.7 พบว่าในการหาค่าใช้จ่ายในการผลิตสำหรับการเดินเครื่องโรงไฟฟ้าค่าที่มีค่าใช้จ่ายมากที่สุดคือ LR และ GA รองลงมาคือ LRGA EP LR-DEA ตามลำดับ แล้วค่าที่ดีที่สุดคือ LR-SaDEA ที่ทำให้ค่าใช้จ่ายดีที่สุด

4.2 วิเคราะห์ผลการทดลอง

ในการทดลองครั้งนี้เป็นการทดลองการแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้ โดยใช้เครื่องกำเนิดไฟฟ้าจำนวน 10 ยูนิต ทำงานเป็นเวลา 24 ชั่วโมง และกำลังการผลิตสำรอง 10% ซึ่งจากการทดลองพบว่า ค่าใช้จ่ายสำหรับการเดินเครื่องโรงไฟฟ้าโดยวิธี LR-SaDEA มีค่าใช้จ่ายที่ดีกว่าวิธีอื่นๆซึ่งสามารถดูได้จาก ตารางที่ 4.7 ซึ่งเป็นผลมาจากการทำงานที่มีประสิทธิภาพของโปรแกรมการคำนวณ สามารถหาค่าที่เหมาะสมต่อกระบวนการ LR-SaDEA (Lagrangian Self Adaptive Differential Evolution Algorithm) ได้เป็นอย่างดี ส่วนเวลาในการคำนวณเมื่อเปรียบเทียบกับวิธีอื่นๆไม่สามารถเปรียบเทียบได้ เนื่องจากไม่มีข้อมูลที่สามารถนำมาเปรียบเทียบได้ แต่เมื่อเปรียบเทียบกับผลลัพธ์ที่ได้จากการคำนวณแล้วพบว่า โปรแกรมสามารถใช้เวลามากกว่าเกณฑ์ที่เหมาะสม

บทที่ 5

สรุปผลการทดลองและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

โครงงานวิศวกรรมนี้ได้นำเสนอการประยุกต์ใช้วิธี SaDEA เพื่อแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้ ซึ่งผู้จัดทำโครงงานวิศวกรรมได้ดำเนินการศึกษา พัฒนา และออกแบบโปรแกรมการแก้ไขปัญหานี้ โดยใช้โปรแกรม MATLAB 2014a ร่วมกับหน้าต่าง Graphical User Interface (GUI) ซึ่งผู้ออกแบบนำวิธี SaDEA นำมาประยุกต์ใช้ ซึ่งได้กำหนดช่วงพารามิเตอร์ดังนี้ X_{max}=20 , X_{min}=16 ,F=16 ,Cr=0.7 เพื่อความสะดวกต่อการใช้งานกับคอมพิวเตอร์ในการแก้ปัญหา โดยสามารถสรุปผลได้ดังนี้คือ

การแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้ โดยใช้เครื่องกำเนิดไฟฟ้าจำนวน 10 ยูนิต ทำงานเป็นเวลา 24 ชั่วโมง ทำการทดลองโดยมีเงื่อนไขว่า Relative Duality gap มีค่าอยู่ในช่วง 10^{-5} กำลังผลิตสำรอง (Spinning Reserve) เท่ากับ 10% จะเห็นได้ว่าการแก้ไขปัญหานี้ผลลัพธ์ของต้นทุนในการผลิตที่ค่าใช้จ่ายที่ต่ำที่สุดในการทดลองในแต่ละรอบการคำนวณจะมีค่าที่แตกต่างกันออกไป เนื่องจากแต่ละรอบในการคำนวณนั้น ค่า λ ที่ได้ทำการสุ่มค่า และปรับตัวมันเองในแต่ละรอบการคำนวณจะไม่เหมือนกันจึงทำให้ส่งผลต่อค่าเชื้อเพลิง ทำให้แต่ละรอบการคำนวณราคาเชื้อเพลิงจะไม่เท่ากัน และในการทดลองให้โปรแกรมคำนวณจำนวน 100 รอบ พบข้อมูลทางสถิติว่าค่าใช้จ่ายในการเดินเครื่องโรงไฟฟ้านั้นมีค่าเบี่ยงเบนมาตรฐาน (SD) ค่อนข้างสูง

สรุปได้ว่า SaDEA นั้นมีความเหมาะสมในการแก้ไขปัญหาการเดินเครื่องโรงไฟฟ้า โดยโปรแกรมที่มีการออกแบบสามารถหาค่าใช้จ่ายได้ต่ำต่อการใช้งาน และสามารถนำไปใช้กับระบบไฟฟ้ากำลังได้จริงในการแก้ไขปัญหาเดินเครื่องโรงไฟฟ้า

5.2 ข้อเสนอแนะ

ค่าพารามิเตอร์ต่างๆที่มีการป้อนค่าข้อมูล ในการคำนวณการวางแผนเดินเครื่องโรงไฟฟ้าในการทำงานของโปรแกรมจะเร็วจะขึ้นอยู่กับ จำนวนรอบในการคำนวณ การประมวลผลของโปรแกรม ประสิทธิภาพของคอมพิวเตอร์ และสำคัญที่สุดคือรุ่นของโปรแกรมที่ใช้ในการออกแบบ ในการการลองคำตอบแรกอาจจะไม่ใช่ต่ำใช้จ่ายที่ต่ำที่สุด เพราะเนื่องจากการใช้ SaDEA เป็นการสุ่มค่าของ initial ในแต่ละรอบการคำนวณจะมีค่าไม่เท่ากัน จึงทำให้ค่าใช้จ่ายในการเดินเครื่องโรงไฟฟ้าอาจจะมีความคลาดเคลื่อนในแต่ละรอบการคำนวณ แล้วทำให้ค่าเบี่ยงเบนมาตรฐานสูง ดังนั้นคำนวณที่จะมีการทดลองซ้ำเพื่อให้ได้คำตอบที่ดีที่สุด

หากมีงานวิจัยเพื่อพัฒนาในครั้งต่อไปในอนาคต ควรจะมีการพิจารณาการคำนวณการสูญเสียและการไหลของกำลังไฟฟ้าที่สายส่งที่สามารถรองรับการวางแผนเดินเครื่องโรงฟ้านี้ได้ เพื่อเพิ่มประสิทธิภาพในการทำงานของโปรแกรมให้สามารถนำไปใช้ได้จริงในระบบไฟฟ้ากำลังในอนาคต

เอกสารอ้างอิง

1. นายณัฐพจน์ วิราภรณ์. ผลกระทบทางการเงินของมาตรฐานพลังงานหมุนเวียนที่มีต่อต้นทุนดำเนินงานของผู้ผลิตไฟฟ้า. ปริญญานิพนธ์วิศวกรรมศาสตร์มหาบัณฑิต สาขาวิศวกรรมไฟฟ้า มหาวิทยาลัยเชียงใหม่, 2551.

2. A. Lelic, I. "Unit Commitment and Dispatch." Introduction to Wholesale Electricity Markets (WEM 101), Northampton, MA, November 5-9, 2012.

3. S.A. Kazarlis, A.G. Bakirt and V. Petridis, “A genetic algorithm solution to the unit commitment problem” *IEEE Trans. Power System*, vol.11, pp. 83-92, Feb. 1996.

4. C.P. Cheng, C.W. Liu, and C.C. Lui., “Unit commitment by Lagrangian relaxation and genetic algorithm,” IEEE Tran. Power Syst., vol.15, no.2, pp. 707-714, May 2000.

5. T. Sum-Im and W. Ongsakul, “Ant colony search algorithm for unit commitment,” Proc. *The IEEE International Conference on Industrial Technology (ICT2003)*, pp. 72-77, Maribor, Slovenia, 10^{th}-12^{th} Dec. 2003.

6. Thanathip Sum-Im, “Lagrangian relaxation combined with the differential evolution algorithm for unit Commitment problem” Emerging Technology and Factory Automation (ETFA), IEEE, pp. 1-6, Barcelona, 16^{th}-19^{th} Sept. 2014.

7. Thanatip Sum-Im. A Novel Differential Evolution Algorithmic Approach to Transmission Expansion Planning. *Ph.D. thesiss*, Brunel University., March 2009.

8. นางสาววิลาสินี ศึกษาการ., “การจ่ายโหลดอย่างประหยัดที่มีฟังก์ชันราคาที่ไม่ราบเรียบโดยวิธีทำให้เหมาะสมแบบกลุ่มอนุภาคร่วมกับวิธีโปรแกรมกำลังสองแบบลำดับ,” *วิทยานิพนธ์บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์*, หน้า 23-35, 2551.

9. สุวิทย์ คำว่าง, จงรักษ์ บุญเส็ง และ ศราวุธ โพธิยา, “การแก้ไขปัญหาการจ่ายโหลดอย่างประหยัดโดยใช้วิธีการค้นหาแบบตาบู”., *วิศวกรรมสาร ม.ข.* ปีที่ 30 ฉบับที่ 3 (189-200) กรกฎาคม – กันยายน. 2546

10. K.A. Juste, H. Kita, E.Tanaka, and J. Hasegawa, “An evolutionary programming solution to the Unit commitment problem,” *IEEE Trans. Power Syst.*, vol. 14, no.4, pp. 1452-1459, Nov. 1999

ภาคผนวก

ภาคผนวก ก ข้อมูลระบบไฟฟ้าที่ใช้ทดลอง

โดยโปรแกรมที่ทำการออกแบบ ในการแก้ไขปัญหาวางแผนเดินเครื่องโรงไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นอัลกอริธึมแบบปรับตัวเองได้ โดยใช้เครื่องกำเนิดไฟฟ้าจำนวน 10 ยูนิต ทำงานเป็นเวลา 24 ชั่วโมง โดยจะมีการแสดงรายละเอียดเกี่ยวกับโรงไฟฟ้าดังนี้

ตารางที่ ก1.1 แสดงรายละเอียดเกี่ยวกับข้อมูลเครื่องกำเนิดไฟฟ้าจำนวน 10 ยูนิต

Unit	**1**	**2**	**3**	**4**	**5**
P_{max}(MW)	455	455	130	130	162
P_{min}(MW)	150	150	20	20	25
a ($/h)	1000	970	700	680	450
b (MWh)	16.19	17.26	16.60	16.50	19.70
c ($/MW2-h)	0.00048	0.00031	0.002	0.00211	0.00398
T_{up}(hr)	8	8	5	5	6
T_{down}(hr)	8	8	5	5	6
S_h(hot start)($)	4500	5000	550	560	900
S_c(cold start)($)	9000	10000	1100	1120	1800
$T_{cold\ start}$ (hr)	5	5	4	4	4
Initial state (hr)	8	8	-5	-5	-6

ตารางที่ ก1.1 (ต่อ) แสดงรายละเอียดเกี่ยวกับข้อมูลเครื่องกำเนิดไฟฟ้าจำนวน 10 ยูนิต

Unit	6	7	8	9	10
P_{max}(MW)	80	85	55	55	55
P_{min}(MW)	20	25	10	10	10
a ($/h)	370	480	660	665	670
b (MWh)	22.26	27.74	25.92	27.27	27.79
c ($/MW2-h)	0.00712	0.0079	0.00413	0.00222	0.00173
T_{up}(hr)	3	3	1	1	1
T_{down}(hr)	3	3	1	1	1
S_h(hot start)($)	170	260	30	30	30
S_c(cold start)($)	340	520	60	60	60
$T_{cold\ start}$ (hr)	2	2	0	0	0
Initial state (hr)	-3	-3	-1	-1	-1

ตารางที่ ก1.2 แสดงข้อมูลความต้องการกำลังไฟฟ้าในเวลา 24 ชั่วโมง และ กำลังการผลิตสำรอง 10%

Hour	Load (MW)	Reserve(MW)	Hour	Load (MW)	Reserve(MW)
1	700	70	13	1400	140
2	750	75	14	1300	130
3	850	85	15	1200	120
4	950	95	16	1050	105
5	1000	100	17	1000	100
6	1100	110	18	1100	110
7	1150	115	19	1200	120
8	1200	120	20	1400	140
9	1300	130	21	1300	130
10	1400	140	22	1100	110
11	1450	145	23	900	90
12	1500	150	24	800	80

ภาคผนวก ข โปรแกรมที่ใช้ทดลอง

โปรแกรมหน้าต่าง GUI ที่ใช้ในการแก้ไขปัญหาการวางแผนเดินเครื่องโรงไฟฟ้าของระบบผลิตไฟฟ้าโดยใช้วิธีลากรางจ์ร่วมกับวิธีดิฟเฟอร์เรนเชียลอีโวลูชั่นแบบปรับตัวเองได้

```
function varargout = UCGUI(varargin)
%UCGUI M-file for UCGUI.fig
%      UCGUI, by itself, creates a new UCGUI or raises the existing
%      singleton*.
%
%      H = UCGUI returns the handle to a new UCGUI or the handle to
%      the existing singleton*.
%
%      UCGUI('Property','Value',...) creates a new UCGUI using the
%      given property value pairs. Unrecognized properties are passed
via
%      varargin to UCGUI_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      UCGUI('CALLBACK') and UCGUI('CALLBACK',hObject,...) call the
%      local function named CALLBACK in UCGUI.M with the given input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help UCGUI

% Last Modified by GUIDE v2.5 15-Mar-2015 21:13:46

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @UCGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @UCGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before UCGUI is made visible.
function UCGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

% Choose default command line output for UCGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes UCGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = UCGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6
as a double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7
as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit8
as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9
as a double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10
as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11
as a double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12
as a double


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit13 as text
%        str2double(get(hObject,'String')) returns contents of edit13
as a double


% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14
as a double


% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15
as a double


% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit16_Callback(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit16 as text
%        str2double(get(hObject,'String')) returns contents of edit16
as a double


% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit17_Callback(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit17 as text
%        str2double(get(hObject,'String')) returns contents of edit17
as a double


% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit18_Callback(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit18 as text
%        str2double(get(hObject,'String')) returns contents of edit18
as a double


% --- Executes during object creation, after setting all properties.
function edit18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit19_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit19 as text
%        str2double(get(hObject,'String')) returns contents of edit19
as a double


% --- Executes during object creation, after setting all properties.
function edit19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit20_Callback(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit20 as text
%        str2double(get(hObject,'String')) returns contents of edit20
as a double


% --- Executes during object creation, after setting all properties.
function edit20_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end



function edit21_Callback(hObject, eventdata, handles)
% hObject    handle to edit21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit21 as text
%        str2double(get(hObject,'String')) returns contents of edit21
as a double


% --- Executes during object creation, after setting all properties.
function edit21_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit22_Callback(hObject, eventdata, handles)
% hObject    handle to edit22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit22 as text
%        str2double(get(hObject,'String')) returns contents of edit22
as a double


% --- Executes during object creation, after setting all properties.
function edit22_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit23_Callback(hObject, eventdata, handles)
% hObject    handle to edit23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit23 as text
%        str2double(get(hObject,'String')) returns contents of edit23
as a double


% --- Executes during object creation, after setting all properties.
function edit23_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit24_Callback(hObject, eventdata, handles)
% hObject    handle to edit24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit24 as text
%        str2double(get(hObject,'String')) returns contents of edit24
as a double


% --- Executes during object creation, after setting all properties.
function edit24_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

tic
%ข้อมูล Gennerator
%-------------------------------------------------------------------------
-----
%gen1
gamma1=1000;
B1=16.19;
r1=0.00048;
P1min=150;
P1max=455;
%gen2
gamma2=970;
    B2=17.26;
    r2=0.00031;
    P2min=150;
    P2max=455;
%gen3
gamma3=700;
    B3=16.6;
    r3=0.002;
    P3min=20;
    P3max=130;
%gen4
gamma4=680;
    B4=16.5;
    r4=0.00211;
    P4min=20;
    P4max=130;
%gen5
 gamma5=450;
    B5=19.7;
    r5=0.00398;
    P5min=162;
    P5max=250;
%gen6
gamma6=370;
    B6=22.26;
    r6=0.00712;
    P6min=20;
    P6max=80;
%gen7
gamma7=480;
    B7=27.74;
    r7=0.00079;
    P7min=25;
    P7max=85;
%gen8
gamma8=660;
    B8=25.92;
    r8=0.00431;
    P8min=10;
    P8max=55;
%gen9
B9=27.27;
    r9=0.00222;
```

```
    P9min=10;
    P9max=55;
%gen10
gamma10=670;
    B10=27.79;
    r10=0.00173;
    P10min=10;
    P10max=55;
%------------------------------------------------------------------------
------

%percentreserve
percentr = str2double(get(handles.edit25,'string'));
PL=[700,750,850,950,1000,1100,1150,1200,1300,1400,1450,1500,1400,1300
,1200,1050,1000,1100,1200,1400,1300,1100,900,800];
PL(1,1)=str2double(get(handles.edit1,'string'));
PL(1,2)=str2double(get(handles.edit2,'string'));
PL(1,3)=str2double(get(handles.edit3,'string'));
PL(1,4)=str2double(get(handles.edit4,'string'));
PL(1,5)=str2double(get(handles.edit5,'string'));
PL(1,6)=str2double(get(handles.edit6,'string'));
PL(1,7)=str2double(get(handles.edit7,'string'));
PL(1,8)=str2double(get(handles.edit8,'string'));
PL(1,9)=str2double(get(handles.edit9,'string'));
PL(1,10)=str2double(get(handles.edit10,'string'));
PL(1,11)=str2double(get(handles.edit11,'string'));
PL(1,12)=str2double(get(handles.edit12,'string'));
PL(1,13)=str2double(get(handles.edit13,'string'));
PL(1,14)=str2double(get(handles.edit14,'string'));
PL(1,15)=str2double(get(handles.edit15,'string'));
PL(1,16)=str2double(get(handles.edit16,'string'));
PL(1,17)=str2double(get(handles.edit17,'string'));
PL(1,18)=str2double(get(handles.edit18,'string'));
PL(1,19)=str2double(get(handles.edit19,'string'));
PL(1,20)=str2double(get(handles.edit20,'string'));
PL(1,21)=str2double(get(handles.edit21,'string'));
PL(1,22)=str2double(get(handles.edit22,'string'));
PL(1,23)=str2double(get(handles.edit23,'string'));
PL(1,24)=str2double(get(handles.edit24,'string'));

%------------------------------------------------------------------------
------
%Initial value
InitST_Cost=[9000,10000,1100,1120,1800,340,520,60,60,60];
InitStateHr=[8,8,-5,-5,-6,-3,-3,-1,-1,-1];
TOnMin=[8,8,5,5,6,3,3,1,1,1];
TDownMin=[8,8,5,5,6,3,3,1,1,1];

%PL reserve
PLReserve=PL*percentr;
T=[1:24];
disp('-------------Display Results-------------')
disp(['Initial Lagrange Multiplier'])
disp('Power Demand(PL)')
disp(' T    PL')
disp(num2str([T;PL]'))
```

```
keep=1;
check=1;
stop=0;
rdgkeep=1;
RDGgraph=[];
Numjstar=[];

Numgraph1=[];
rdgala=[];
Jstarala=[];


%intitial-----------------------------------------------------------------
-----
D = 24;

rdgstop = str2double(get(handles.edit0026,'string'));
ngs = str2double(get(handles.edit27,'string'));
nogm = str2double(get(handles.edit28,'string'));
Xmax = str2double(get(handles.edit29,'string'));
Xmin = str2double(get(handles.edit30,'string'));
F = str2double(get(handles.edit31,'string'));
Cr1 = str2double(get(handles.edit32,'string'));
np1 = str2double(get(handles.edit33,'string'));
Cr=ones(1,D).*Cr1;
np=D.*np1;
indi1=(ones(np,D).*Xmin)+((rand(np,D)).*((ones(np,D).*Xmax)-
(ones(np,D).*Xmin)));
%รอบgeneration
for aaa3=1:nogm

%-------------------------------------------------------------------
-----
%รอบการคำนวณRDG
for ibest=1:np
lambda=indi1(ibest,:);


    %Generator No.1

    P1=(lambda-B1)/(2*r1);
    U1=(P1>=P1max);
    U1old=U1;
    PD1old=U1old.*P1max;
    %Generator No.2

    P2=(lambda-B2)/(2*r2);
    U2=(P2>=P2max);
    U2old=U2;
    PD2old=U2old.*P2max;

    %Generator No.3

    P3=(lambda-B3)/(2*r3);
    U3=(P3>=P3max);
```

```
    U3old=U3;
    PD3old=U3old.*P3max;

    %Generator No.4

    P4=(lambda-B4)/(2*r4);
    U4=(P4>=P4max);
    U4old=U4;
    PD4old=U4old.*P4max;
    %Generator No.5

    P5=(lambda-B5)/(2*r5);
    U5=(P5>=P5max);
    U5old=U5;
    PD5old=U5old.*P5max;
    %Generator No.6

    P6=(lambda-B6)/(2*r6);
    U6=(P6>=P6max);
    U6old=U6;
    PD6old=U6old.*P6max;

    %Generator No.7

    P7=(lambda-B7)/(2*r7);
    U7=(P7>=P7max);
    U7old=U7;
    PD7old=U7old.*P7max;

    %Generator No.8

    P8=(lambda-B8)/(2*r8);
    U8=(P8>=P8max);
    U8old=U8;
    PD8old=U8old.*P8max;

    %Generator No.9
    gamma9=665;

    P9=(lambda-B9)/(2*r9);
    U9=(P9>=P9max);
    U9old=U9;
    PD9old=U9old.*P9max;

    %Generator No.10

    P10=(lambda-B10)/(2*r10);
    U10=(P10>=P10max);
    U10old=U10;
    PD10old=U10old.*P10max;


Pmax=[P1max,P2max,P3max,P4max,P5max,P6max,P7max,P8max,P9max,P10max];
    U=[U1;U2;U3;U4;U5;U6;U7;U8;U9;U10]';
    Unew=U;
```

```
     %----------------Update U Step 1-----------------
    %Check if sum Pmax >PL and replace with Pmax and Update U


PU=[U1*P1max;U2*P2max;U3*P3max;U4*P4max;U5*P5max;U6*P6max;U7*P7max;U8
*P8max;U9*P9max;U10*P10max]';
    lessPLindex=find(sum(PU')<(PL+PLReserve));
    lessPLvalue=U(lessPLindex,:);

PLsel=[P1max*lessPLvalue(:,1),P2max*lessPLvalue(:,2),P3max*lessPLvalu
e(:,3),P4max*lessPLvalue(:,4),P5max*lessPLvalue(:,5),P6max*lessPLvalu
e(:,6),P7max*lessPLvalue(:,7),P8max*lessPLvalue(:,8),P9max*lessPLvalu
e(:,9),P10max*lessPLvalue(:,10)];
    difvalue=PL(lessPLindex)-sum(PLsel');

    fillIndex=lessPLvalue~=1;

PLfill=[P1max*fillIndex(:,1),P2max*fillIndex(:,2),P3max*fillIndex(:,3
),P4max*fillIndex(:,4),P5max*fillIndex(:,5),P6max*fillIndex(:,6),P7ma
x*fillIndex(:,7),P8max*fillIndex(:,8),P9max*fillIndex(:,9),P10max*fil
lIndex(:,10)];
    for i=1:size(PLfill,1)
        srt=sort(PLfill(i,:),'descend');
        cums=cumsum(srt);
        needfill=srt(1:find(cums==min(min(cums(difvalue(i)<cums)))));
        for j=1:length(needfill)
            if needfill(j)~=0
                Unew(lessPLindex(i),PLfill(i,:)==needfill(j))=1;
             end
        end
    end

    U=Unew;

U1=U(:,1)';U2=U(:,2)';U3=U(:,3)';U4=U(:,4)';U5=U(:,5)';U6=U(:,6)';U7=
U(:,7)';U8=U(:,8)';U9=U(:,9)';U10=U(:,10)';
    %---------------Update U Step 2--------------
    %Check TOnmin and TDownMin value
    numon=zeros(1,10);
    numdown=zeros(1,10);
    numon(find(InitStateHr>0))=InitStateHr(find(InitStateHr>0));
    for i=1:size(U,1)
       for j=1:size(U,2)
            if U(i,j)==1
               if numon(j)<TOnMin(j)
                    numon(j)=numon(j)+1;
                end
            else
                if numon(j)<TOnMin(j)&numon(j)>0
                    Unew(i,j)=1;
                    numon(j)=numon(j)+1;
                else
                    numon(j)=0;
                end
            end
        end
```

```
    end

   Ux=Unew;


numdown(find(InitStateHr<0))=abs(InitStateHr(find(InitStateHr<0)));
   for i=1:24
       for j=1:size(Ux,2)
            if Ux(i,j)==0
                if i==24 & numdown(j)< TDownMin(j)
                    st=i-numdown(j);
                    if i-numdown(j)<=0
                    st=1;
                    end
                    Unew(st:i,j)=1;
                end
                    numdown(j)=numdown(j)+1;
            else
                if numdown(j)<TDownMin(j)
                    st=i-numdown(j);
                    if i-numdown(j)<=0
                    st=1;
                    end
                    Unew(st:i,j)=1;
                end
                numdown(j)=0;
            end
       end
    end



%PUnew=[Unew(:,1)*P1max,Unew(:,2)*P2max,Unew(:,3)*P3max,Unew(:,4)*P4m
ax,Unew(:,5)*P5max,Unew(:,6)*P6max,Unew(:,7)*P7max,Unew(:,8)*P8max,Un
ew(:,9)*P9max,Unew(:,10)*P10max];
    U=Unew;

U1=U(:,1)';U2=U(:,2)';U3=U(:,3)';U4=U(:,4)';U5=U(:,5)';U6=U(:,6)';U7=
U(:,7)';U8=U(:,8)';U9=U(:,9)';U10=U(:,10)';



    P1(find((U1old~=U1)==1))=P1max;
    P2(find((U2old~=U2)==1))=P2max;
    P3(find((U3old~=U3)==1))=P3max;
    P4(find((U4old~=U4)==1))=P4max;
    P5(find((U5old~=U5)==1))=P5max;
    P6(find((U6old~=U6)==1))=P6max;
    P7(find((U7old~=U7)==1))=P7max;
    P8(find((U8old~=U7)==1))=P8max;
    P9(find((U9old~=U9)==1))=P9max;
    P10(find((U10old~=U10)==1))=P10max;

    P1new=P1;
    P1new(P1>=P1max)=P1max;
```

```
P1new(P1<=P1min)=P1min;
CP1=gamma1+B1*P1new+r1*P1new.^2;
GenTable1=[T;P1;U1;CP1-lambda.*P1new]';

% disp(' T               P               U      CP-lambda*P');
% disp(num2str(GenTable1));

P2new=P2;
P2new(P2>=P2max)=P2max;
P2new(P2<=P2min)=P2min;
CP2=gamma2+B2*P2new+r2*P2new.^2;
GenTable2=[T;P2;U2;CP2-lambda.*P2new]';

% disp(' T               P               U      CP-lambda*P');
% disp(num2str(GenTable2));

P3new=P3;
P3new(P3>=P3max)=P3max;
P3new(P3<=P3min)=P3min;
CP3=gamma3+B3*P3new+r3*P3new.^2;
GenTable3=[T;P3;U3;CP3-lambda.*P3new]';

%  disp(' T               P               U      CP-lambda*P');
%  disp(num2str(GenTable3));

P4new=P4;
P4new(P4>=P4max)=P4max;
P4new(P4<=P4min)=P4min;
CP4=gamma4+B4*P4new+r4*P4new.^2;
GenTable4=[T;P4;U4;CP4-lambda.*P4new]';

% disp(' T               P               U      CP-lambda*P');
% disp(num2str(GenTable4));

P5new=P5;
P5new(P5>=P5max)=P5max;
P5new(P5<=P5min)=P5min;
CP5=gamma5+B5*P5new+r5*P5new.^2;
GenTable5=[T;P5;U5;CP5-lambda.*P5new]';

% disp(' T               P               U      CP-lambda*P');
% disp(num2str(GenTable5));

P6new=P6;
P6new(P6>=P6max)=P6max;
P6new(P6<=P6min)=P6min;
CP6=gamma6+B6*P6new+r6*P6new.^2;
GenTable6=[T;P6;U6;CP6-lambda.*P6new]';

% disp(' T               P               U      CP-lambda*P');
% disp(num2str(GenTable6));

P7new=P7;
P7new(P7>=P7max)=P7max;
P7new(P7<=P7min)=P7min;
CP7=gamma7+B7*P7new+r7*P7new.^2;
```

```
 GenTable7=[T;P7;U7;CP7-lambda.*P7new]';

% disp(' T               P               U      CP-lambda*P');
% disp(num2str(GenTable7));

 P8new=P8;
 P8new(P8>=P8max)=P8max;
 P8new(P8<=P8min)=P8min;
 CP8=gamma8+B8*P8new+r8*P8new.^2;
 GenTable8=[T;P8;U8;CP8-lambda.*P8new]';

% disp(' T               P               U      CP-lambda*P');
% disp(num2str(GenTable8));

 P9new=P9;
 P9new(P9>=P9max)=P9max;
 P9new(P9<=P9min)=P9min;
 CP9=gamma9+B9*P9new+r9*P9new.^2;
 GenTable9=[T;P9;U9;CP9-lambda.*P9new]';

% disp(' T               P               U      CP-lambda*P');
% disp(num2str(GenTable9));

P10new=P10;
 P10new(P10>=P10max)=P10max;
 P10new(P10<=P10min)=P10min;
 CP10=gamma10+B10*P10new+r10*P10new.^2;
 GenTable10=[T;P10;U10;CP10-lambda.*P10new]';

% disp(' T               P               U      CP-lambda*P');
% disp(num2str(GenTable10));

 PD1=U1.*P1max;
 PD2=U2.*P2max;
 PD3=U3.*P3max;
 PD4=U4.*P4max;
 PD5=U5.*P5max;
 PD6=U6.*P6max;
 PD7=U7.*P7max;
 PD8=U8.*P8max;
 PD9=U9.*P9max;
 PD10=U10.*P10max;

 P=[PD1;PD2;PD3;PD4;PD5;PD6;PD7;PD8;PD9;PD10]';



 %////////////q*,J*,RDG calculation//////////////////////

 InitState=InitStateHr>0;
 U_ST1=[InitState(1),U1];
 U_ST2=[InitState(2),U2];
 U_ST3=[InitState(3),U3];
 U_ST4=[InitState(4),U4];
 U_ST5=[InitState(5),U5];
```

```
    U_ST6=[InitState(6),U6];
    U_ST7=[InitState(7),U7];
    U_ST8=[InitState(8),U8];
    U_ST9=[InitState(9),U9];
    U_ST10=[InitState(10),U10];
    %find start up position

numStartup1=0;numStartup2=0;numStartup3=0;numStartup4=0;numStartup5=0
;numStartup6=0;numStartup7=0;numStartup8=0;numStartup9=0;numStartup10
=0;
    k=1;
    for i=2:25
        startup1(k)=(U_ST1(i)-U_ST1(i-1))==1;
        startup2(k)=(U_ST2(i)-U_ST2(i-1))==1;
        startup3(k)=(U_ST3(i)-U_ST3(i-1))==1;
        startup4(k)=(U_ST4(i)-U_ST4(i-1))==1;
        startup5(k)=(U_ST5(i)-U_ST5(i-1))==1;
        startup6(k)=(U_ST6(i)-U_ST6(i-1))==1;
        startup7(k)=(U_ST7(i)-U_ST7(i-1))==1;
        startup8(k)=(U_ST8(i)-U_ST8(i-1))==1;
        startup9(k)=(U_ST9(i)-U_ST9(i-1))==1;
        startup10(k)=(U_ST10(i)-U_ST10(i-1))==1;
        k=k+1;
    end

ST_Cost1=zeros(1,24);ST_Cost2=zeros(1,24);ST_Cost3=zeros(1,24);ST_Cos
t4=zeros(1,24);ST_Cost5=zeros(1,24);ST_Cost6=zeros(1,24);ST_Cost7=zer
os(1,24);ST_Cost8=zeros(1,24);ST_Cost9=zeros(1,24);ST_Cost10=zeros(1,
24);

sumofclose1=0;sumofclose2=0;sumofclose3=5;sumofclose4=5;sumofclose5=6
;sumofclose6=3;sumofclose7=3;sumofclose8=1;sumofclose9=1;sumofclose10
=1;
    for i=1:24
        if U1(i)==0
            sumofclose1=sumofclose1+1;
        end
        if U2(i)==0
            sumofclose2=sumofclose2+1;
        end
        if U3(i)==0
            sumofclose3=sumofclose3+1;
        end
        if U4(i)==0
            sumofclose4=sumofclose4+1;
        end
        if U5(i)==0
            sumofclose5=sumofclose5+1;
        end
        if U6(i)==0
            sumofclose6=sumofclose6+1;
        end
        if U7(i)==0
            sumofclose7=sumofclose7+1;
        end
        if U8(i)==0
            sumofclose8=sumofclose8+1;
        end
```

```
if U9(i)==0
    sumofclose9=sumofclose9+1;
end
if U10(i)==0
    sumofclose10=sumofclose10+1;
end
if startup1(i)==1
    if sumofclose1>5
        ST_Cost1(i)=startup1(i)*InitST_Cost(1);
    else
        ST_Cost1(i)=startup1(i)*InitST_Cost(1)/2;
    end
    sumofclose1=0;
else
    ST_Cost1(i)=0;
end
if startup2(i)==1
    if sumofclose2>5
        ST_Cost2(i)=startup2(i)*InitST_Cost(2);
    else
        ST_Cost2(i)=startup2(i)*InitST_Cost(2)/2;
    end
    sumofclose2=0;
else
    ST_Cost2(i)=0;
end
if startup3(i)==1
    if sumofclose3>4
        ST_Cost3(i)=startup3(i)*InitST_Cost(3);
    else
        ST_Cost3(i)=startup3(i)*InitST_Cost(3)/2;
    end
    sumofclose3=0;
else
    ST_Cost3(i)=0;
end
if startup4(i)==1
    if sumofclose4>4
        ST_Cost4(i)=startup4(i)*InitST_Cost(4);
    else
        ST_Cost4(i)=startup4(i)*InitST_Cost(4)/2;
    end
    sumofclose4=0;
else
    ST_Cost4(i)=0;
end
if startup5(i)==1
    if sumofclose5>4
        ST_Cost5(i)=startup5(i)*InitST_Cost(5);
    else
        ST_Cost5(i)=startup5(i)*InitST_Cost(5)/2;
    end
    sumofclose5=0;
else
    ST_Cost5(i)=0;
end
if startup6(i)==1
    if sumofclose6>2
```

```
                ST_Cost6(i)=startup6(i)*InitST_Cost(6);
            else
                ST_Cost6(i)=startup6(i)*InitST_Cost(6)/2;
            end
            sumofclose6=0;
        else
            ST_Cost6(i)=0;
        end
        if startup7(i)==1
            if sumofclose7>2
                ST_Cost7(i)=startup7(i)*InitST_Cost(7);
            else
                ST_Cost7(i)=startup7(i)*InitST_Cost(7)/2;
            end
            sumofclose7=0;
        else
            ST_Cost7(i)=0;
        end
        if startup8(i)==1
            if sumofclose8>0
                ST_Cost8(i)=startup8(i)*InitST_Cost(8);
            else
                ST_Cost8(i)=startup8(i)*InitST_Cost(8);
            end
            sumofclose8=0;
        else
            ST_Cost8(i)=0;
        end
        if startup9(i)==1
            if sumofclose9>0
                ST_Cost9(i)=startup9(i)*InitST_Cost(9);
            else
                ST_Cost9(i)=startup9(i)*InitST_Cost(9);
            end
            sumofclose9=0;
        else
            ST_Cost9(i)=0;
        end
        if startup10(i)==1
            if sumofclose10>0
                ST_Cost10(i)=startup10(i)*InitST_Cost(10);
            else
                ST_Cost10(i)=startup10(i)*InitST_Cost(10);
            end
            sumofclose10=0;
        else
            ST_Cost10(i)=0;
        end
    end

    %Start-Up Cost calculation

ST_CostByTime=ST_Cost1+ST_Cost2+ST_Cost3+ST_Cost4+ST_Cost5+ST_Cost6+S
T_Cost7+ST_Cost8+ST_Cost9+ST_Cost10;

ST_CostByUnit=[sum(ST_Cost1),sum(ST_Cost2),sum(ST_Cost3),sum(ST_Cost4
),sum(ST_Cost5),sum(ST_Cost6),sum(ST_Cost7),sum(ST_Cost8),sum(ST_Cost
9),sum(ST_Cost10)];
```

```
%ST_CostByTime=startup1*InitST_Cost(1)+startup2*InitST_Cost(2)+startu
p3*InitST_Cost(3)+startup4*InitST_Cost(4)...
    %
+startup5*InitST_Cost(5)+startup6*InitST_Cost(6)+startup7*InitST_Cost
(7)+startup8*InitST_Cost(8)+startup9*InitST_Cost(9)+startup10*InitST_
Cost(10);

%ST_CostByUnit=[sum(startup1*InitST_Cost(1)),sum(startup2*InitST_Cost
(2)),sum(startup3*InitST_Cost(3)),...
    %
sum(startup4*InitST_Cost(4)),sum(startup5*InitST_Cost(5)),sum(startup
6*InitST_Cost(6)),sum(startup7*InitST_Cost(7)),sum(startup8*InitST_Co
st(8)),sum(startup9*InitST_Cost(9)),sum(startup10*InitST_Cost(10))];
    %q* calculation
    qstar=sum((CP1-lambda.*P1new).*U1)+sum((CP2-
lambda.*P2new).*U2)+sum((CP3-lambda.*P3new).*U3)+sum((CP4-
lambda.*P4new).*U4)...
        +sum((CP5-lambda.*P5new).*U5)+sum((CP6-
lambda.*P6new).*U6)+sum((CP7-lambda.*P7new).*U7)+sum((CP8-
lambda.*P8new).*U8)+sum((CP9-lambda.*P9new).*U9)+sum((CP10-
lambda.*P10new).*U10)+sum(ST_CostByTime)+sum(lambda.*PL);



    %Pedc (Power economic dispacth calculation)

gamma=[gamma1,gamma2,gamma3,gamma4,gamma5,gamma6,gamma7,gamma8,gamma9
,gamma10];
    B=[B1,B2,B3,B4,B5,B6,B7,B8,B9,B10];
    r=[r1,r2,r3,r4,r5,r6,r7,r8,r9,r10];

Pmin=[P1min,P2min,P3min,P4min,P5min,P6min,P7min,P8min,P9min,P10min];

Pmax=[P1max,P2max,P3max,P4max,P5max,P6max,P7max,P8max,P9max,P10max];
    PmaxSel=zeros(10);
    Pedc=zeros(24,10);
    CPedcByTime=zeros(24,1);



    InitCost=0; %use if all U=0
    for t =1:24
        PmaxSel=zeros(10);
        PminSel=zeros(10);
        PmaxSel=Pmax(U(t,:)==1);
        PminSel=Pmin(U(t,:)==1);
        gammaSel=gamma(U(t,:)==1);
        BSel=B(U(t,:)==1);
        rSel=r(U(t,:)==1);
        index=find(U(t,:)==1);
        if (sum(U(t,:))==1)
            if(sum(PmaxSel)>=PL(t))
                Pedc(t,index)=PL(t);

CPSel=gammaSel+BSel.*Pedc(t,index)+rSel.*Pedc(t,index).^2;
                CPedcByTime(t)=CPSel;
            else
                CPedcByTime(t)=InitCost;
```

```
                Pedc(t,:)=0;
            end
        elseif(sum(U(t,:))==0)||sum(PmaxSel)<PL(t)
            CPedcByTime(t)=InitCost;
            Pedc(t,:)=0;
        else
            ptemp=PL(t);
            sumcp=zeros(length(PmaxSel),1);
            pmax=PmaxSel;
            pmin=PminSel;
            gammasel=gammaSel;
            bsel=BSel;
            rsel=rSel;
            PedcTempAll=zeros(length(pmax),length(pmax));
            for i=1:length(pmax)
                difp=sum(pmax)-ptemp;
                pedcTemp=pmax;
                for j=length(pmax):-1:1
                    if(difp<pmax(j))
                        if(pmax(j)-difp)>pmin(j)
                            pedcTemp(j)=pmax(j)-difp;
                            difp=0;
                        else
                            pedcTemp(j)=pmin(j);
                            difp=difp-pmax(j)+pmin(j);
                        end
                    else
                        pedcTemp(j)=pmin(j);
                        difp=difp-pmax(j)+pmin(j);
                    end
                end
                PedcTempAll(i,:)=pedcTemp;
                cp=0;
                cp=gammasel+bsel.*pedcTemp+rsel.*pedcTemp.^2;
                sumcp(i)=sum(cp);
                pmax=circshift(pmax,[1 1]);
                pmin=circshift(pmin,[1 1]);
                gammasel=circshift(gammasel,[1 1]);
                bsel=circshift(bsel,[1 1]);
                rsel=circshift(rsel,[1 1]);
            end
            lsindex=find(sumcp==min(sumcp));
            pedcTemp=circshift(PedcTempAll(lsindex(1),:),[1,-
(lsindex(1)-1)]);

            Pedc(t,index)=pedcTemp;

CPSel=gammaSel+BSel.*Pedc(t,index)+rSel.*Pedc(t,index).^2;
            CPedcByTime(t)=sum(CPSel);

        end
    end

   %J* calculation
   Jstar=sum(CPedcByTime)+sum(ST_CostByTime);

   %RDG calculation
```

```
   rdg=abs(Jstar-qstar)/qstar;
    %-------------------------------------------------------------------
------
   JStar(ibest)=Jstar;
   QStar(ibest)=qstar;
   RDG(ibest)=rdg;

   PedcTABLE(:,:,ibest)=[T;Pedc']';
   %diplay value


CostTABLE(:,:,ibest)=[T;CPedcByTime';ST_CostByTime;(CPedcByTime'+ST_C
ostByTime)]';
   %TotCostTABLE(iter)=['    Total    ',num2str(sum(CPedcByTime)),'
',num2str(sum(ST_CostByTime)),'
',num2str(sum(CPedcByTime)+sum(ST_CostByTime))];


TBC=num2str([T;CPedcByTime';ST_CostByTime;(CPedcByTime'+ST_CostByTime
)]');

%-------------------------------------------------------------------
-----
   if rdg<rdgstop & keep==1
       Jstarkeep=Jstar;
       Qstarkeep=qstar;
       rdgkeep=rdg;
       keep=keep+1;
       lambdakeep=lambda;
       Numgen=aaa3;
       Numround=ibest;
       keepF=F;
       keepCr=Cr;
       uctablekeep=[T;U1;U2;U3;U4;U5;U6;U7;U8;U9;U10]';

qtablekeep=num2str([T;PD1;PD2;PD3;PD4;PD5;PD6;PD7;PD8;PD9;P10]');
       edtable=([T;Pedc']');
       totaledtable=TBC;
       ccc=num2str(sum(CPedcByTime));
       ccc1=num2str(sum(ST_CostByTime));
       ccc2=num2str(sum(CPedcByTime)+sum(ST_CostByTime));

chikeep=([T;CPedcByTime';ST_CostByTime;(CPedcByTime'+ST_CostByTime)]'
);
   end
   if rdg<rdgstop & keep==2
       if Jstar<Jstarkeep
           Jstarkeep=Jstar;
           Qstarkeep=qstar;
           rdgkeep=rdg;
           lambdakeep=lambda;
           Numgen=aaa3;
           Numround=ibest;
           keepF=F;
           keepCr=Cr;
           uctablekeep=[T;U1;U2;U3;U4;U5;U6;U7;U8;U9;U10]';
```

```
qtablekeep=num2str([T;PD1;PD2;PD3;PD4;PD5;PD6;PD7;PD8;PD9;P10]');
           edtable=([T;Pedc']');
           totaledtable=TBC;
           ccc=num2str(sum(CPedcByTime));
           ccc1=num2str(sum(ST_CostByTime));
           ccc2=num2str(sum(CPedcByTime)+sum(ST_CostByTime));

chikeep=([T;CPedcByTime';ST_CostByTime;(CPedcByTime'+ST_CostByTime)]'
);
       end
   end
%-------------------------------------------------------------------
------

rdgala=[rdgala,rdg];
Jstarala=[Jstarala,Jstar];




%-------------------------------------------------------------------
-----
end
%-------------------------------------------------------------------
------
rdgalamin=mean(rdgala);
Jstaralamin=mean(Jstarala);

Numgraph=aaa3;

Numgraph1=[Numgraph1,Numgraph];
RDGgraph=[RDGgraph,rdgalamin];
Numjstar=[Numjstar,Jstaralamin];

rdgala=[];
Jstarala=[];



%-------------------------------------------------------------------
-----
    if rdgkeep<rdgstop & check==1
        Jstarcheck=Jstarkeep;
        check=check+1;
    end
    if rdgkeep<rdgstop & check==2
        if Jstarcheck==Jstarkeep
            stop=stop+1;
        else
            Jstarcheck=Jstarkeep;
            stop=0;
        end
    end
    if stop==ngs
```

```
        break
    end
%-------------------------------------------------------------------------
-----
for aaa2=1:np


%-------------------------------------------------------------------------
------
%สุ่มค่า r1 r2 r3
%-------------------------------------------------------------------------
------
tgv=indi1(aaa2,:);
fmtv=randperm(np);

xr1=fmtv(1,1);
xr2=fmtv(1,2);
xr3=fmtv(1,3);
while xr1==aaa2 & xr2==aaa2 & xr3==aaa2
fmtv=randperm(np);

xr1=fmtv(1,1);
xr2=fmtv(1,2);
xr3=fmtv(1,3);
end

xr11=indi1(xr1,:);
xr22=indi1(xr2,:);
xr33=indi1(xr3,:);

%ค่าหา mutant vector
%-------------------------------------------------------------------------
------
mtv=abs(xr11+((xr22-xr33).*F));

%หาค่า trial vector
%-------------------------------------------------------------------------
------
Rcro=rand(1,D);


for rcc=1:D

tgv1=tgv(1,rcc);
mtv1=mtv(1,rcc);
Cr1=Cr(1,rcc);
Rcro1=Rcro(1,rcc);

if Rcro1 <= Cr1
    tavv=mtv1;
else
    tavv=tgv1;
end
```

```
if ~isempty(tavv)
trial(rcc) = tavv;
end


end

%-------------------------------------------------------------------
-----
for aaa=1:2
    if aaa==1
        lambda=tgv;
    else
        lambda=trial;
    end
    %-------------------------------------------------------------------
-----
    %qstar
    T=[1:24];


    %Generator No.1

    P1=(lambda-B1)/(2*r1);
    U1=(P1>=P1max);
    U1old=U1;
    PD1old=U1old.*P1max;
    %Generator No.2

    P2=(lambda-B2)/(2*r2);
    U2=(P2>=P2max);
    U2old=U2;
    PD2old=U2old.*P2max;

    %Generator No.3

    P3=(lambda-B3)/(2*r3);
    U3=(P3>=P3max);
    U3old=U3;
    PD3old=U3old.*P3max;

    %Generator No.4

    P4=(lambda-B4)/(2*r4);
    U4=(P4>=P4max);
    U4old=U4;
    PD4old=U4old.*P4max;
    %Generator No.5

    P5=(lambda-B5)/(2*r5);
    U5=(P5>=P5max);
    U5old=U5;
    PD5old=U5old.*P5max;
    %Generator No.6
```

```
    P6=(lambda-B6)/(2*r6);
    U6=(P6>=P6max);
    U6old=U6;
    PD6old=U6old.*P6max;

    %Generator No.7

    P7=(lambda-B7)/(2*r7);
    U7=(P7>=P7max);
    U7old=U7;
    PD7old=U7old.*P7max;

    %Generator No.8

    P8=(lambda-B8)/(2*r8);
    U8=(P8>=P8max);
    U8old=U8;
    PD8old=U8old.*P8max;

    %Generator No.9
    gamma9=665;

    P9=(lambda-B9)/(2*r9);
    U9=(P9>=P9max);
    U9old=U9;
    PD9old=U9old.*P9max;

    %Generator No.10

    P10=(lambda-B10)/(2*r10);
    U10=(P10>=P10max);
    U10old=U10;
    PD10old=U10old.*P10max;


Pmax=[P1max,P2max,P3max,P4max,P5max,P6max,P7max,P8max,P9max,P10max];
    U=[U1;U2;U3;U4;U5;U6;U7;U8;U9;U10]';
    Unew=U;

     %----------------Update U Step 1-----------------
    %Check if sum Pmax >PL and replace with Pmax and Update U


PU=[U1*P1max;U2*P2max;U3*P3max;U4*P4max;U5*P5max;U6*P6max;U7*P7max;U8
*P8max;U9*P9max;U10*P10max]';
    lessPLindex=find(sum(PU')<(PL+PLReserve));
    lessPLvalue=U(lessPLindex,:);

PLsel=[P1max*lessPLvalue(:,1),P2max*lessPLvalue(:,2),P3max*lessPLvalu
e(:,3),P4max*lessPLvalue(:,4),P5max*lessPLvalue(:,5),P6max*lessPLvalu
e(:,6),P7max*lessPLvalue(:,7),P8max*lessPLvalue(:,8),P9max*lessPLvalu
e(:,9),P10max*lessPLvalue(:,10)];
    difvalue=PL(lessPLindex)-sum(PLsel');

    fillIndex=lessPLvalue~=1;
```

```
PLfill=[P1max*fillIndex(:,1),P2max*fillIndex(:,2),P3max*fillIndex(:,3
),P4max*fillIndex(:,4),P5max*fillIndex(:,5),P6max*fillIndex(:,6),P7ma
x*fillIndex(:,7),P8max*fillIndex(:,8),P9max*fillIndex(:,9),P10max*fil
lIndex(:,10)];
    for i=1:size(PLfill,1)
        srt=sort(PLfill(i,:),'descend');
        cums=cumsum(srt);
        needfill=srt(1:find(cums==min(min(cums(difvalue(i)<cums)))));
        for j=1:length(needfill)
            if needfill(j)~=0
                Unew(lessPLindex(i),PLfill(i,:)==needfill(j))=1;
             end
        end
    end

    U=Unew;

U1=U(:,1)';U2=U(:,2)';U3=U(:,3)';U4=U(:,4)';U5=U(:,5)';U6=U(:,6)';U7=
U(:,7)';U8=U(:,8)';U9=U(:,9)';U10=U(:,10)';
    %---------------Update U Step 2--------------
    %Check TOnmin and TDownMin value
    numon=zeros(1,10);
    numdown=zeros(1,10);
    numon(find(InitStateHr>0))=InitStateHr(find(InitStateHr>0));
    for i=1:size(U,1)
       for j=1:size(U,2)
            if U(i,j)==1
              if numon(j)<TOnMin(j)
                  numon(j)=numon(j)+1;
              end
            else
                if numon(j)<TOnMin(j)&numon(j)>0
                   Unew(i,j)=1;
                   numon(j)=numon(j)+1;
                else
                   numon(j)=0;
                end
            end
        end

    end

   Ux=Unew;


numdown(find(InitStateHr<0))=abs(InitStateHr(find(InitStateHr<0)));
   for i=1:24
       for j=1:size(Ux,2)
            if Ux(i,j)==0
                if i==24 & numdown(j)< TDownMin(j)
                    st=i-numdown(j);
                    if i-numdown(j)<=0
                    st=1;
                    end
                    Unew(st:i,j)=1;
                end
```

```
                    numdown(j)=numdown(j)+1;
            else
                if numdown(j)<TDownMin(j)
                    st=i-numdown(j);
                    if i-numdown(j)<=0
                    st=1;
                    end
                    Unew(st:i,j)=1;
                end
                numdown(j)=0;
            end
        end
    end


%PUnew=[Unew(:,1)*P1max,Unew(:,2)*P2max,Unew(:,3)*P3max,Unew(:,4)*P4m
ax,Unew(:,5)*P5max,Unew(:,6)*P6max,Unew(:,7)*P7max,Unew(:,8)*P8max,Un
ew(:,9)*P9max,Unew(:,10)*P10max];
    U=Unew;

U1=U(:,1)';U2=U(:,2)';U3=U(:,3)';U4=U(:,4)';U5=U(:,5)';U6=U(:,6)';U7=
U(:,7)';U8=U(:,8)';U9=U(:,9)';U10=U(:,10)';



    P1(find((U1old~=U1)==1))=P1max;
    P2(find((U2old~=U2)==1))=P2max;
    P3(find((U3old~=U3)==1))=P3max;
    P4(find((U4old~=U4)==1))=P4max;
    P5(find((U5old~=U5)==1))=P5max;
    P6(find((U6old~=U6)==1))=P6max;
    P7(find((U7old~=U7)==1))=P7max;
    P8(find((U8old~=U7)==1))=P8max;
    P9(find((U9old~=U9)==1))=P9max;
    P10(find((U10old~=U10)==1))=P10max;

    P1new=P1;
    P1new(P1>=P1max)=P1max;
    P1new(P1<=P1min)=P1min;
    CP1=gamma1+B1*P1new+r1*P1new.^2;
    GenTable1=[T;P1;U1;CP1-lambda.*P1new]';

   % disp(' T               P               U       CP-lambda*P');
   % disp(num2str(GenTable1));

    P2new=P2;
    P2new(P2>=P2max)=P2max;
    P2new(P2<=P2min)=P2min;
    CP2=gamma2+B2*P2new+r2*P2new.^2;
    GenTable2=[T;P2;U2;CP2-lambda.*P2new]';

   % disp(' T               P               U       CP-lambda*P');
   % disp(num2str(GenTable2));

    P3new=P3;
```

```
  P3new(P3>=P3max)=P3max;
  P3new(P3<=P3min)=P3min;
  CP3=gamma3+B3*P3new+r3*P3new.^2;
  GenTable3=[T;P3;U3;CP3-lambda.*P3new]';

%  disp(' T               P               U      CP-lambda*P');
%  disp(num2str(GenTable3));

  P4new=P4;
  P4new(P4>=P4max)=P4max;
  P4new(P4<=P4min)=P4min;
  CP4=gamma4+B4*P4new+r4*P4new.^2;
  GenTable4=[T;P4;U4;CP4-lambda.*P4new]';

 % disp(' T               P               U      CP-lambda*P');
 % disp(num2str(GenTable4));

  P5new=P5;
  P5new(P5>=P5max)=P5max;
  P5new(P5<=P5min)=P5min;
  CP5=gamma5+B5*P5new+r5*P5new.^2;
  GenTable5=[T;P5;U5;CP5-lambda.*P5new]';

% disp(' T               P               U      CP-lambda*P');
% disp(num2str(GenTable5));

  P6new=P6;
  P6new(P6>=P6max)=P6max;
  P6new(P6<=P6min)=P6min;
  CP6=gamma6+B6*P6new+r6*P6new.^2;
  GenTable6=[T;P6;U6;CP6-lambda.*P6new]';

% disp(' T               P               U      CP-lambda*P');
% disp(num2str(GenTable6));

  P7new=P7;
  P7new(P7>=P7max)=P7max;
  P7new(P7<=P7min)=P7min;
  CP7=gamma7+B7*P7new+r7*P7new.^2;
  GenTable7=[T;P7;U7;CP7-lambda.*P7new]';

 % disp(' T               P               U      CP-lambda*P');
 % disp(num2str(GenTable7));

  P8new=P8;
  P8new(P8>=P8max)=P8max;
  P8new(P8<=P8min)=P8min;
  CP8=gamma8+B8*P8new+r8*P8new.^2;
  GenTable8=[T;P8;U8;CP8-lambda.*P8new]';

 % disp(' T               P               U      CP-lambda*P');
 % disp(num2str(GenTable8));

  P9new=P9;
  P9new(P9>=P9max)=P9max;
  P9new(P9<=P9min)=P9min;
```

```
    CP9=gamma9+B9*P9new+r9*P9new.^2;
    GenTable9=[T;P9;U9;CP9-lambda.*P9new]';

   % disp(' T               P               U      CP-lambda*P');
   % disp(num2str(GenTable9));

   P10new=P10;
    P10new(P10>=P10max)=P10max;
    P10new(P10<=P10min)=P10min;
    CP10=gamma10+B10*P10new+r10*P10new.^2;
    GenTable10=[T;P10;U10;CP10-lambda.*P10new]';

   % disp(' T               P               U      CP-lambda*P');
   % disp(num2str(GenTable10));

    PD1=U1.*P1max;
    PD2=U2.*P2max;
    PD3=U3.*P3max;
    PD4=U4.*P4max;
    PD5=U5.*P5max;
    PD6=U6.*P6max;
    PD7=U7.*P7max;
    PD8=U8.*P8max;
    PD9=U9.*P9max;
    PD10=U10.*P10max;



    %////////////q*,J*,RDG calculation///////////////////////

    InitState=InitStateHr>0;
    U_ST1=[InitState(1),U1];
    U_ST2=[InitState(2),U2];
    U_ST3=[InitState(3),U3];
    U_ST4=[InitState(4),U4];
    U_ST5=[InitState(5),U5];
    U_ST6=[InitState(6),U6];
    U_ST7=[InitState(7),U7];
    U_ST8=[InitState(8),U8];
    U_ST9=[InitState(9),U9];
    U_ST10=[InitState(10),U10];
    %find start up position

numStartup1=0;numStartup2=0;numStartup3=0;numStartup4=0;numStartup5=0
;numStartup6=0;numStartup7=0;numStartup8=0;numStartup9=0;numStartup10
=0;
    k=1;
    for i=2:25
        startup1(k)=(U_ST1(i)-U_ST1(i-1))==1;
        startup2(k)=(U_ST2(i)-U_ST2(i-1))==1;
        startup3(k)=(U_ST3(i)-U_ST3(i-1))==1;
        startup4(k)=(U_ST4(i)-U_ST4(i-1))==1;
        startup5(k)=(U_ST5(i)-U_ST5(i-1))==1;
        startup6(k)=(U_ST6(i)-U_ST6(i-1))==1;
        startup7(k)=(U_ST7(i)-U_ST7(i-1))==1;
        startup8(k)=(U_ST8(i)-U_ST8(i-1))==1;
```

```
        startup9(k)=(U_ST9(i)-U_ST9(i-1))==1;
        startup10(k)=(U_ST10(i)-U_ST10(i-1))==1;
        k=k+1;
    end

ST_Cost1=zeros(1,24);ST_Cost2=zeros(1,24);ST_Cost3=zeros(1,24);ST_Cos
t4=zeros(1,24);ST_Cost5=zeros(1,24);ST_Cost6=zeros(1,24);ST_Cost7=zer
os(1,24);ST_Cost8=zeros(1,24);ST_Cost9=zeros(1,24);ST_Cost10=zeros(1,
24);

sumofclose1=0;sumofclose2=0;sumofclose3=5;sumofclose4=5;sumofclose5=6
;sumofclose6=3;sumofclose7=3;sumofclose8=1;sumofclose9=1;sumofclose10
=1;
    for i=1:24
        if U1(i)==0
            sumofclose1=sumofclose1+1;
        end
        if U2(i)==0
            sumofclose2=sumofclose2+1;
        end
        if U3(i)==0
            sumofclose3=sumofclose3+1;
        end
        if U4(i)==0
            sumofclose4=sumofclose4+1;
        end
        if U5(i)==0
            sumofclose5=sumofclose5+1;
        end
        if U6(i)==0
            sumofclose6=sumofclose6+1;
        end
        if U7(i)==0
            sumofclose7=sumofclose7+1;
        end
        if U8(i)==0
            sumofclose8=sumofclose8+1;
        end
        if U9(i)==0
            sumofclose9=sumofclose9+1;
        end
        if U10(i)==0
            sumofclose10=sumofclose10+1;
        end
        if startup1(i)==1
            if sumofclose1>5
                ST_Cost1(i)=startup1(i)*InitST_Cost(1);
            else
                ST_Cost1(i)=startup1(i)*InitST_Cost(1)/2;
            end
            sumofclose1=0;
        else
            ST_Cost1(i)=0;
        end
        if startup2(i)==1
            if sumofclose2>5
                ST_Cost2(i)=startup2(i)*InitST_Cost(2);
            else
```

```
        ST_Cost2(i)=startup2(i)*InitST_Cost(2)/2;
    end
    sumofclose2=0;
else
    ST_Cost2(i)=0;
end
if startup3(i)==1
    if sumofclose3>4
        ST_Cost3(i)=startup3(i)*InitST_Cost(3);
    else
        ST_Cost3(i)=startup3(i)*InitST_Cost(3)/2;
    end
    sumofclose3=0;
else
    ST_Cost3(i)=0;
end
if startup4(i)==1
    if sumofclose4>4
        ST_Cost4(i)=startup4(i)*InitST_Cost(4);
    else
        ST_Cost4(i)=startup4(i)*InitST_Cost(4)/2;
    end
    sumofclose4=0;
else
    ST_Cost4(i)=0;
end
if startup5(i)==1
    if sumofclose5>4
        ST_Cost5(i)=startup5(i)*InitST_Cost(5);
    else
        ST_Cost5(i)=startup5(i)*InitST_Cost(5)/2;
    end
    sumofclose5=0;
else
    ST_Cost5(i)=0;
end
if startup6(i)==1
    if sumofclose6>2
        ST_Cost6(i)=startup6(i)*InitST_Cost(6);
    else
        ST_Cost6(i)=startup6(i)*InitST_Cost(6)/2;
    end
    sumofclose6=0;
else
    ST_Cost6(i)=0;
end
if startup7(i)==1
    if sumofclose7>2
        ST_Cost7(i)=startup7(i)*InitST_Cost(7);
    else
        ST_Cost7(i)=startup7(i)*InitST_Cost(7)/2;
    end
    sumofclose7=0;
else
    ST_Cost7(i)=0;
end
if startup8(i)==1
    if sumofclose8>0
```

```
                ST_Cost8(i)=startup8(i)*InitST_Cost(8);
            else
                ST_Cost8(i)=startup8(i)*InitST_Cost(8);
            end
            sumofclose8=0;
        else
            ST_Cost8(i)=0;
        end
        if startup9(i)==1
            if sumofclose9>0
                ST_Cost9(i)=startup9(i)*InitST_Cost(9);
            else
                ST_Cost9(i)=startup9(i)*InitST_Cost(9);
            end
            sumofclose9=0;
        else
            ST_Cost9(i)=0;
        end
        if startup10(i)==1
            if sumofclose10>0
                ST_Cost10(i)=startup10(i)*InitST_Cost(10);
            else
                ST_Cost10(i)=startup10(i)*InitST_Cost(10);
            end
            sumofclose10=0;
        else
            ST_Cost10(i)=0;
        end
    end

    %Start-Up Cost calculation

ST_CostByTime=ST_Cost1+ST_Cost2+ST_Cost3+ST_Cost4+ST_Cost5+ST_Cost6+S
T_Cost7+ST_Cost8+ST_Cost9+ST_Cost10;

ST_CostByUnit=[sum(ST_Cost1),sum(ST_Cost2),sum(ST_Cost3),sum(ST_Cost4
),sum(ST_Cost5),sum(ST_Cost6),sum(ST_Cost7),sum(ST_Cost8),sum(ST_Cost
9),sum(ST_Cost10)];

%ST_CostByTime=startup1*InitST_Cost(1)+startup2*InitST_Cost(2)+startu
p3*InitST_Cost(3)+startup4*InitST_Cost(4)...
    %
+startup5*InitST_Cost(5)+startup6*InitST_Cost(6)+startup7*InitST_Cost
(7)+startup8*InitST_Cost(8)+startup9*InitST_Cost(9)+startup10*InitST_
Cost(10);

%ST_CostByUnit=[sum(startup1*InitST_Cost(1)),sum(startup2*InitST_Cost
(2)),sum(startup3*InitST_Cost(3)),...
    %
sum(startup4*InitST_Cost(4)),sum(startup5*InitST_Cost(5)),sum(startup
6*InitST_Cost(6)),sum(startup7*InitST_Cost(7)),sum(startup8*InitST_Co
st(8)),sum(startup9*InitST_Cost(9)),sum(startup10*InitST_Cost(10))];
    %q* calculation
    qstar=sum((CP1-lambda.*P1new).*U1)+sum((CP2-
lambda.*P2new).*U2)+sum((CP3-lambda.*P3new).*U3)+sum((CP4-
lambda.*P4new).*U4)...
        +sum((CP5-lambda.*P5new).*U5)+sum((CP6-
lambda.*P6new).*U6)+sum((CP7-lambda.*P7new).*U7)+sum((CP8-
```

```
lambda.*P8new).*U8)+sum((CP9-lambda.*P9new).*U9)+sum((CP10-
lambda.*P10new).*U10)+sum(ST_CostByTime)+sum(lambda.*PL);

    %Pedc (Power economic dispacth calculation)

gamma=[gamma1,gamma2,gamma3,gamma4,gamma5,gamma6,gamma7,gamma8,gamma9
,gamma10];
    B=[B1,B2,B3,B4,B5,B6,B7,B8,B9,B10];
    r=[r1,r2,r3,r4,r5,r6,r7,r8,r9,r10];

Pmin=[P1min,P2min,P3min,P4min,P5min,P6min,P7min,P8min,P9min,P10min];

Pmax=[P1max,P2max,P3max,P4max,P5max,P6max,P7max,P8max,P9max,P10max];
    PmaxSel=zeros(10);
    Pedc=zeros(24,10);
    CPedcByTime=zeros(24,1);


    InitCost=0; %use if all U=0
    for t =1:24
        PmaxSel=zeros(10);
        PminSel=zeros(10);
        PmaxSel=Pmax(U(t,:)==1);
        PminSel=Pmin(U(t,:)==1);
        gammaSel=gamma(U(t,:)==1);
        BSel=B(U(t,:)==1);
        rSel=r(U(t,:)==1);
        index=find(U(t,:)==1);
        if (sum(U(t,:))==1)
            if(sum(PmaxSel)>=PL(t))
                Pedc(t,index)=PL(t);

CPSel=gammaSel+BSel.*Pedc(t,index)+rSel.*Pedc(t,index).^2;
                CPedcByTime(t)=CPSel;
            else
                CPedcByTime(t)=InitCost;
                Pedc(t,:)=0;
            end
        elseif(sum(U(t,:))==0)||sum(PmaxSel)<PL(t)
            CPedcByTime(t)=InitCost;
            Pedc(t,:)=0;
        else
            ptemp=PL(t);
            sumcp=zeros(length(PmaxSel),1);
            pmax=PmaxSel;
            pmin=PminSel;
            gammasel=gammaSel;
            bsel=BSel;
            rsel=rSel;
            PedcTempAll=zeros(length(pmax),length(pmax));
            for i=1:length(pmax)
                difp=sum(pmax)-ptemp;
                pedcTemp=pmax;
                for j=length(pmax):-1:1
                    if(difp<pmax(j))
                        if(pmax(j)-difp)>pmin(j)
                            pedcTemp(j)=pmax(j)-difp;
```

```
                            difp=0;
                        else
                            pedcTemp(j)=pmin(j);
                            difp=difp-pmax(j)+pmin(j);
                        end
                    else
                        pedcTemp(j)=pmin(j);
                        difp=difp-pmax(j)+pmin(j);
                    end
                end
                PedcTempAll(i,:)=pedcTemp;
                cp=0;
                cp=gammasel+bsel.*pedcTemp+rsel.*pedcTemp.^2;
                sumcp(i)=sum(cp);
                pmax=circshift(pmax,[1 1]);
                pmin=circshift(pmin,[1 1]);
                gammasel=circshift(gammasel,[1 1]);
                bsel=circshift(bsel,[1 1]);
                rsel=circshift(rsel,[1 1]);
            end
            lsindex=find(sumcp==min(sumcp));
            pedcTemp=circshift(PedcTempAll(lsindex(1),:),[1,-
(lsindex(1)-1)]);

            Pedc(t,index)=pedcTemp;

CPSel=gammaSel+BSel.*Pedc(t,index)+rSel.*Pedc(t,index).^2;
            CPedcByTime(t)=sum(CPSel);

        end
    end

   %J* calculation
   Jstar=sum(CPedcByTime)+sum(ST_CostByTime);
   rdg=abs(Jstar-qstar)/qstar;


    if aaa==1
        rdgx1=rdg;
        jstarx1=Jstar;
        qstarx1=qstar;
    else
        rdgx2=rdg;
        jstarx2=Jstar;
        qstarx2=qstar;
    end
end

if rdgx2<rdgx1  & jstarx2<jstarx1
    pop=trial;
    else
    pop=tgv;
end

if aaa2==1
gendea=pop;
else
```

```
gendea=[gendea;pop];
end
end
%-------------------------------------------------------------------
------
T1=rand();
T2=rand();
if T1<0.1
    F=0.6+(rand(1)*(1-0.6));
end
if T2<0.1
    Cr=(0.7+(rand(1)*(0.9-0.7)))*ones(1,D);
end

indi1=gendea;
end

%-------------------------------------------------------------------
------
%แสดงผลลัพธ์
axes(handles.axes1);
plot(Numgraph1,RDGgraph,'-sm')
grid;
xlabel('Iteration','color','b');
ylabel('Average RDG','color','b');

axes(handles.axes2);
plot(Numgraph1,Numjstar,'-sg')
grid;
xlabel('Iteration','color','b');
ylabel('Average Total Cost','color','b');




lambdakeep1=lambdakeep
disp(['Numgen=',num2str(Numgen)]);
disp(['Numround=',num2str(Numround)]);
disp(['QStar=',num2str(Qstarkeep)]);
disp(['JStar=',num2str(Jstarkeep)]);
disp(['RDG=',num2str(rdgkeep)]);
disp(['Parameter F =',num2str(keepF)]);
disp(['Parameter Cr =',num2str(keepCr)]);
keepCr1=keepCr(1,1);
set(handles.text35,'string',Numgen);
set(handles.text39,'string',num2str(Qstarkeep));
set(handles.text41,'string',num2str(Jstarkeep));
set(handles.text49,'string',rdgkeep);
set(handles.text43,'string',keepF);
set(handles.text45,'string',keepCr1);
set(handles.anstime,'string',ccc2);
set(handles.text54,'string',ccc);
set(handles.text57,'string',ccc1);
```

```
set(handles.uitable7,'Data',uctablekeep);
set(handles.uitable8,'Data',edtable);
set(handles.uitable13,'Data',lambdakeep.');




disp(['Unit Commitment Table ']);
    disp('     T     U1    U2    U3   U4    U5    U6    U7    U8
U9    U10');
    disp(uctablekeep);
    disp(['================================================']);
    disp(['Solution of Dual Problem ']);
    disp(' T   P1     P2    P3    P4    P5    P6    P7    P8    P9
P10');
    disp(qtablekeep);
    disp(['================================================']);
    disp(['======The best Economic Dispatch======']);
   disp('     T    P1    P2     P3    P4   P5    P6     P7     P8
P9    P10');
   disp(edtable);
   disp(['================================================']);
   disp(['=====The best Cost Comparation=====']);
   disp(' T          FuelCost    StartUpCost     TotalCost');
   disp(totaledtable);
   disp(['    Total   ',ccc,'     ',ccc1,'   ',ccc2]);
   disp(['================================================']);

   toc;

   set(handles.text53,'string',toc);
   chi=chikeep;
   set(handles.uitable12,'Data',chi)



function edit25_Callback(hObject, eventdata, handles)
% hObject    handle to edit25 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit25 as text
%        str2double(get(hObject,'String')) returns contents of edit25
as a double


% --- Executes during object creation, after setting all properties.
function edit25_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit25 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit0026_Callback(hObject, eventdata, handles)
% hObject    handle to edit0026 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit0026 as text
%        str2double(get(hObject,'String')) returns contents of
edit0026 as a double


% --- Executes during object creation, after setting all properties.
function edit0026_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit0026 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit27_Callback(hObject, eventdata, handles)
% hObject    handle to edit27 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit27 as text
%        str2double(get(hObject,'String')) returns contents of edit27
as a double


% --- Executes during object creation, after setting all properties.
function edit27_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit27 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit28_Callback(hObject, eventdata, handles)
% hObject    handle to edit28 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit28 as text
%        str2double(get(hObject,'String')) returns contents of edit28
as a double


% --- Executes during object creation, after setting all properties.
function edit28_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit28 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit29_Callback(hObject, eventdata, handles)
% hObject    handle to edit29 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit29 as text
%        str2double(get(hObject,'String')) returns contents of edit29
as a double


% --- Executes during object creation, after setting all properties.
function edit29_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit29 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end



function edit30_Callback(hObject, eventdata, handles)
% hObject    handle to edit30 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit30 as text
%        str2double(get(hObject,'String')) returns contents of edit30
as a double


% --- Executes during object creation, after setting all properties.
function edit30_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit30 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit31_Callback(hObject, eventdata, handles)
% hObject    handle to edit31 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit31 as text
%        str2double(get(hObject,'String')) returns contents of edit31
as a double


% --- Executes during object creation, after setting all properties.
function edit31_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit31 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit32_Callback(hObject, eventdata, handles)
% hObject    handle to edit32 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit32 as text
%        str2double(get(hObject,'String')) returns contents of edit32
as a double


% --- Executes during object creation, after setting all properties.
function edit32_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit32 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit33_Callback(hObject, eventdata, handles)
% hObject    handle to edit33 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit33 as text
%        str2double(get(hObject,'String')) returns contents of edit33
as a double


% --- Executes during object creation, after setting all properties.
function edit33_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit33 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function text46_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text46 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

ประวัตินิสิตผู้ทำโครงงาน

ชื่อ-สกุล นางสาว กฤษณา ผนึกดี

วัน เดือน ปีเกิด 16 เมษายน พ.ศ.2536

สถานที่เกิด จังหวัดกรุงเทพ

สถานที่อยู่ปัจจุบัน บ้านเลขที่ 72/24 หมู่ 8
ถนน สุขุมวิท ตำบล บางเมืองใหม่
อำเภอ เมืองฯ จังหวัด สมุทรปราการ 10270

เบอร์โทรศัพท์ 081 – 808 – 4791

E-mail kasanit-fine@hotmail.com

ประวัติการศึกษา

พุทธศักราช 2548 มัธยมศึกษาตอนต้น โรงเรียนสตรีสมุทรปราการ

พุทธศักราช 2551 มัธยมศึกษาตอนปลาย แผนกวิทยาศาสตร์-คณิตศาสตร์
โรงเรียนสตรีสมุทรปราการ

พุทธศักราช 2554 กำลังศึกษาระดับปริญญาตรี สาขาวิศวกรรมศาสตร์
ภาควิชาวิศวกรรมไฟฟ้า มหาวิทยาลัยศรีนครินทรวิโรฒ

ประวัตินิสิตผู้ทำโครงงาน

ชื่อ-สกุล นาย ชนกานต์ บัวบาน

วัน เดือน ปีเกิด 24 มกราคม พ.ศ.2536

สถานที่เกิด จังหวัดกรุงเทพมหานคร

สถานที่อยู่ปัจจุบัน บ้านเลขที่ 13 หมู่ 11 ตำบล อ้อมน้อย
อำเภอ กระทุ่มแบน
จังหวัด สมุทรสาคร 74130

เบอร์โทรศัพท์ 089 – 761 4452

E-mail tan_pea@hotmail.co.th

ประวัติการศึกษา

พุทธศักราช 2548 มัธยมศึกษาตอนต้น โรงเรียนอ้อมน้อยโสภณชนูปถัมภ์

พุทธศักราช 2551 มัธยมศึกษาตอนปลาย แผนกวิทยาศาสตร์-คณิตศาสตร์
โรงเรียนอ้อมน้อยโสภณชนูปถัมภ์

พุทธศักราช 2554 กำลังศึกษาระดับปริญญาตรี สาขาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ

ประวัตินิสิตผู้ทำโครงงาน

ชื่อ-สกุล นางสาว ชุติภา เรืองจิตร

วัน เดือน ปีเกิด 15 พฤศจิกายน พ.ศ.2535

สถานที่เกิด จังหวัดน่าน

สถานที่อยู่ปัจจุบัน บ้านเลขที่ 20/9 หมู่ 3 ตำบล แม่สา

อำเภอ แม่ริม จังหวัด เชียงใหม่ 50180

เบอร์โทรศัพท์ 092 – 765 9871

E-mail a.electrical.a@gmail.com

ประวัติการศึกษา

พุทธศักราช 2548	มัธยมศึกษาตอนต้น โรงเรียนดาราวิทยาลัย
พุทธศักราช 2551	มัธยมศึกษาตอนปลาย แผนกวิทยาศาสตร์-คณิตศาสตร์ โรงเรียนดาราวิทยาลัย
พุทธศักราช 2554	กำลังศึกษาระดับปริญญาตรี สาขาวิศวกรรมศาสตร์ ภาควิชาวิศวกรรมไฟฟ้า มหาวิทยาลัยศรีนครินทรวิโรฒ