การจัดตารางการผลิตสำหรับระบบการผลิตแบบไหลเลื่อนยืดหยุ่น ที่มีเครื่องจักรขนานแบบไม่สัมพันธ์กันและมีประสิทธิภาพไม่เท่ากัน

วิมลสิริ ประภาติกุล

วิทยานิพนธ์เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
บัณฑิตศึกษา สาขาเทคโนโลยีวิศวกรรม
สถาบันเทคโนโลยีไทย-ญี่ปุ่น
ปีการศึกษา 2568

DETERMINATION OF JOB SCHEDULING FOR A FLEXIBLE FLOW SHOP WITH UNRELATED PARALLEL MACHINES AND NON-UNIFORM PERFORMANCE

Wimonsiri Prapatigul

A Thesis Submitted in Partial Fulfillment of Requirements
for The Degree of Master of Engineering Program in Engineering Technology
Graduate Studies
Thai-Nichi Institute of Technology
Academic Year 2025

หัวข้อวิทยานิพนธ์	การจัดตารางการผลิตสำหรับระบบการผลิตแบบไหลเลื่อนยืดหยุ่น ที่มีเครื่องจักรขนานแบบไม่สัมพันธ์กันและมีประสิทธิภาพไม่เท่ากัน
โดย	วิมลสิริ ประภาติกุล
สาขาวิชา	เทคโนโลยีวิศวกรรม
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รองศาสตราจารย์อัญชลี สุพิทักษ์

บัณฑิตศึกษา สถาบันเทคโนโลยีไทย-ญี่ปุ่น อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... รองอธิการบดีฝ่ายวิชาการ
(รองศาสตราจารย์ ดร.วรากร ศรีเชวงทรัพย์)
วันที่.....เดือน.....พ.ศ.....

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ธีรพล ศิลาวรรณ์)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.นพดล ศรีพุทธา)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ดอน แก้วดก)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์
(รองศาสตราจารย์อัญชลี สุพิทักษ์)

วิมลสิริ ประภาติกุล : การจัดตารางการผลิตสำหรับระบบการผลิตแบบไหลเลื่อนยืดหยุ่น ที่มีเครื่องจักรขนานแบบไม่สัมพันธ์กันและมีประสิทธิภาพไม่เท่ากัน. อาจารย์ที่ปรึกษา : รองศาสตราจารย์อัญชลี สุพิทักษ์, 74 หน้า.

งานวิจัยนี้มุ่งศึกษาปัญหาการจัดตารางการผลิตในระบบ Flexible Flow Shop ที่มีลักษณะเครื่องจักรแบบ Unrelated Parallel Machines ซึ่งเครื่องจักรในแต่ละสถานีมีสมรรถนะการทำงานที่แตกต่างกัน โดยมีวัตถุประสงค์เพื่อลดระยะเวลาการผลิตรวม (Makespan) และเวลาไหลเฉลี่ยของงาน (Average Flow Time) ซึ่งเป็นตัวชี้วัดสำคัญของประสิทธิภาพการผลิตในระบบที่มีความซับซ้อนสูง

ในการศึกษานี้ ได้นำ Genetic Algorithm มาใช้เป็นวิธีการค้นหาคำตอบที่เหมาะสม โดยออกแบบโครโมโซมในรูปแบบลำดับงาน และใช้หลักการเลือกเครื่องจักรที่ให้เวลาสิ้นสุดเร็วที่สุดในการจัดตารางการผลิต จากนั้นทำการเปรียบเทียบประสิทธิภาพกับวิธี Brute Force Algorithm โดยกำหนดเงื่อนไขการทดลองให้ใช้ชุดข้อมูลเดียวกัน และจำกัดเวลาในการรันโปรแกรมเท่ากันที่ 2 นาที ผลการทดลองจาก 10 กรณี ของปัญหา 20 งาน พบว่า Genetic Algorithm สามารถให้ผลลัพธ์ที่มีประสิทธิภาพดีกว่า Brute Force Algorithm อย่างสม่ำเสมอในภาพรวม โดยมีค่า Makespan เฉลี่ยเท่ากับ 107.40 ขณะที่ Brute Force Algorithm มีค่าเฉลี่ยเท่ากับ 113.70 คิดเป็นการลดลงประมาณ 5.54% นอกจากนี้ ค่า Average Flow Time ของ Genetic Algorithm มีค่าเฉลี่ยเท่ากับ 68.15 ซึ่งต่ำกว่าวิธี Brute Force ที่มีค่าเฉลี่ย 71.72 หรือปรับปรุงได้ประมาณ 4.97% แม้ในบางกรณีค่า Average Flow Time ของ Brute Force จะต่ำกว่าเล็กน้อย แต่เมื่อพิจารณาความสม่ำเสมอของผลลัพธ์ในทุกกรณีทดลอง พบว่า Genetic Algorithm ให้คำตอบที่มีเสถียรภาพและมีประสิทธิภาพมากกว่า

จากผลการศึกษา สามารถสรุปได้ว่า Genetic Algorithm เป็นวิธีที่เหมาะสมสำหรับการแก้ปัญหาการจัดตารางการผลิตในระบบ Flexible Flow Shop ที่มีความซับซ้อนสูง โดยเฉพาะในกรณีที่มีข้อจำกัดด้านเวลาในการคำนวณ ซึ่งสอดคล้องกับผลการทดลองและการวิเคราะห์ในบทที่ 4 และสามารถนำไปประยุกต์ใช้ในระบบการผลิตจริงเพื่อเพิ่มประสิทธิภาพในการดำเนินงานได้

บัณฑิตศึกษา ลายมือชื่อนักศึกษา .....

สาขาวิชาเทคโนโลยีวิศวกรรม ลายมือชื่ออาจารย์ที่ปรึกษา.....

ปีการศึกษา 2568

WIMONSIRI PRAPATIGUL: DETERMINATION OF JOB SCHEDULING FOR A FLEXIBLE FLOW SHOP WITH UNRELATED PARALLEL MACHINES AND NON-UNIFORM PERFORMANCE. ADVISOR: ASSOC. PROF. ANCHALEE SUPITHAK, 74 PP.

This research investigates a production scheduling problem in a Flexible Flow Shop system with Unrelated Parallel Machines, where machines at each production stage have different processing capabilities. The objective is to minimize the total production time Makespan and Average Flow Time, which are important performance indicators in manufacturing systems.

In this study, a Genetic Algorithm is applied as the optimization method. The chromosome structure is designed based on job sequences, while a machine selection rule using the earliest completion time is employed to generate production schedules. The proposed approach is compared with the Brute Force Algorithm under the same experimental conditions using identical datasets and a fixed computation time of 2 minutes.

Experimental results from 10 problem instances with 20 jobs show that the Genetic Algorithm provides better overall performance than the Brute Force method. The average Makespan obtained by the Genetic Algorithm is 107.40, compared with 113.70 for the Brute Force method, representing an improvement of 5.54%. In addition, the Average Flow Time achieved by the Genetic Algorithm is 68.15, lower than 71.72 from the Brute Force method, corresponding to an improvement of 4.97%.

The results indicate that the Genetic Algorithm is an effective approach for solving complex Flexible Flow Shop scheduling problems under computational time constraints and can be applied to improve efficiency in real-world production systems.

Graduate Studies Student's Signature .....
Field of Study Enginerring Technology Advisor's Signature.....
Academic Year 2025

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยความกรุณาอย่างยิ่งจาก รศ.อัญชลี สุพิทักษ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ผู้ซึ่งได้ให้คำแนะนำ ข้อคิดเห็น และแนวทางในการดำเนินงานวิจัยอย่างใกล้ชิดมาโดยตลอด รวมถึงการตรวจสอบและปรับปรุงเนื้อหาให้มีความถูกต้องและสมบูรณ์ ผู้วิจัยขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาสนี้

ขอขอบพระคุณคณาจารย์ทุกท่านใน สถาบันเทคโนโลยีไทย-ญี่ปุ่น ที่ได้ถ่ายทอดความรู้ทางวิชาการและประสบการณ์อันมีคุณค่า ซึ่งเป็นพื้นฐานสำคัญในการดำเนินงานวิจัยครั้งนี้ให้สำเร็จลุล่วง

ขอขอบคุณเจ้าหน้าที่และบุคลากรของสถาบันฯ ที่ให้ความช่วยเหลือและอำนวยความสะดวกในด้านต่าง ๆ ตลอดระยะเวลาของการศึกษา

สุดท้ายนี้ ผู้วิจัยขอขอบพระคุณครอบครัว และเพื่อน ๆ ที่ให้การสนับสนุน กำลังใจ และความช่วยเหลือในทุกด้าน จนทำให้สามารถผ่านพ้นอุปสรรคต่าง ๆ และดำเนินการวิจัยจนสำเร็จได้ด้วยดี

วิมลสิริ ประภาติกุล

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ในปัจจุบันภาคอุตสาหกรรมการผลิตมีการแข่งขันที่สูงขึ้นอย่างต่อเนื่อง ผู้ประกอบการจำเป็นต้องเพิ่มประสิทธิภาพของกระบวนการผลิตเพื่อลดต้นทุน ลดระยะเวลาในการส่งมอบสินค้า และเพิ่มความพึงพอใจของลูกค้า หนึ่งในปัจจัยสำคัญที่มีผลโดยตรงต่อประสิทธิภาพการผลิตคือ การจัดตารางการผลิต (Production Scheduling) ซึ่งเป็นกระบวนการกำหนดลำดับและเวลาการทำงานของแต่ละงานบนเครื่องจักรในระบบการผลิต

ระบบการผลิตในโรงงานอุตสาหกรรมสมัยใหม่มักมีความซับซ้อนมากขึ้น โดยเฉพาะระบบการผลิตแบบ Flexible Flow Shop (FFS) ซึ่งเป็นระบบที่แต่ละงานต้องผ่านหลายขั้นตอนการผลิต (Stages) และในแต่ละขั้นตอนสามารถเลือกใช้เครื่องจักรได้มากกว่าหนึ่งเครื่อง (Parallel Machines) ทำให้ระบบมีความยืดหยุ่นสูง แต่ในขณะเดียวกันก็เพิ่มความซับซ้อนของการจัดตารางการผลิต

ปัญหาการจัดตารางการผลิตแบบ Flexible Flow Shop จัดเป็นปัญหาประเภท NP-hard ซึ่งหมายความว่า เมื่อขนาดของปัญหาเพิ่มขึ้น จำนวนวิธีการจัดตารางที่เป็นไปได้จะเพิ่มขึ้นแบบทวีคูณ ส่งผลให้การหาคำตอบที่เหมาะสมที่สุด (Optimal Solution) ด้วยวิธีการคำนวณแบบ Brute Force ใช้เวลาสูงมากและไม่เหมาะสมต่อการนำไปใช้ในสถานการณ์จริง

นอกจากนี้ ในระบบการผลิตจริง เครื่องจักรในแต่ละขั้นตอนมักมีประสิทธิภาพที่แตกต่างกัน (Unrelated Parallel Machines หรือ Non-uniform Performance Machines) เนื่องจากความแตกต่างด้านเทคโนโลยี อายุการใช้งาน หรือสภาพการทำงาน ส่งผลให้เวลาในการประมวลผล (Processing Time) ของงานบนเครื่องจักรแต่ละเครื่องไม่เท่ากัน ซึ่งยิ่งเพิ่มความซับซ้อนของปัญหาการจัดตารางการผลิต

ในช่วงหลายปีที่ผ่านมา นักวิจัยได้ให้ความสนใจในการนำวิธีการเชิงเมตาฮิวริสติก (Metaheuristics) เช่น Genetic Algorithm (GA), Particle Swarm Optimization (PSO) และ Ant Colony Optimization (ACO) มาใช้ในการแก้ปัญหาการจัดตารางการผลิต เนื่องจากสามารถค้นหาคำตอบที่ใกล้เคียงคำตอบที่เหมาะสมที่สุดได้ภายในเวลาที่จำกัด โดยเฉพาะ Genetic Algorithm ซึ่งเป็นวิธีการเชิงวิวัฒนาการที่มีความสามารถในการค้นหาคำตอบในพื้นที่ขนาดใหญ่ได้อย่างมีประสิทธิภาพ

อย่างไรก็ตาม จากการศึกษางานวิจัยที่ผ่านมา พบว่า งานส่วนใหญ่ยังมุ่งเน้นไปที่ระบบที่เครื่องจักรมีประสิทธิภาพเท่ากัน (Homogeneous Machines) หรือไม่ได้พิจารณาการเปรียบเทียบ

กับวิธีการแบบ Brute Force ภายใต้ข้อจำกัดด้านเวลาอย่างชัดเจน อีกทั้งยังมีข้อจำกัดในด้านการวิเคราะห์ตัวชี้วัดหลายมิติ เช่น Makespan และ Average Flow Time ร่วมกัน

ดังนั้น งานวิจัยนี้จึงมุ่งเน้นการพัฒนาแบบจำลองและวิธีการจัดตารางการผลิตในระบบ Flexible Flow Shop ที่มีเครื่องจักรขนานแบบประสิทธิภาพไม่เท่ากัน โดยประยุกต์ใช้ Genetic Algorithm เพื่อค้นหาคำตอบที่เหมาะสม พร้อมทั้งทำการเปรียบเทียบประสิทธิภาพกับวิธี Brute Force ภายใต้เวลาการคำนวณที่จำกัด เพื่อให้ได้แนวทางที่เหมาะสมสำหรับการประยุกต์ใช้ในอุตสาหกรรมจริง

1.2 นิยามปัญหาการวิจัย

ปัญหาที่พิจารณาในงานวิจัยนี้คือ ปัญหาการจัดตารางการผลิตในระบบ Flexible Flow Shop ซึ่งประกอบด้วยงานจำนวนหลายงาน ที่ต้องผ่านขั้นตอนการผลิตจำนวนขั้นตอนมากกว่าหนึ่งขั้นตอน โดยในแต่ละขั้นตอนมีเครื่องจักรขนานจำนวนหลายเครื่อง และเครื่องจักรแต่ละเครื่องมีเวลาในการประมวลผลที่แตกต่างกันแบบไม่สม่ำเสมอ

วัตถุประสงค์ของปัญหาคือ การกำหนดลำดับการผลิตและการเลือกเครื่องจักรในแต่ละขั้นตอน เพื่อให้ค่าระยะเวลาการผลิตรวม (Makespan) และค่าเวลาเฉลี่ยของงานในระบบ (Average Flow Time) มีค่าน้อยที่สุด

1.3 วัตถุประสงค์ของงานวิจัย

1.3.1 เพื่อพัฒนาแบบจำลองและอัลกอริทึม Genetic Algorithm สำหรับแก้ปัญหาการจัดตารางการผลิตในระบบ Flexible Flow Shop ที่มีเครื่องจักรขนานแบบประสิทธิภาพไม่เท่ากันและไม่สม่ำเสมอ Non-Unniform

1.3.2 เพื่อเปรียบเทียบประสิทธิภาพของวิธี Genetic Algorithm กับวิธี Brute Force ภายใต้ข้อจำกัดด้านเวลาการคำนวณ

1.3.3 เพื่อประเมินประสิทธิภาพของวิธีที่พัฒนาขึ้นโดยใช้ตัวชี้วัด Makespan และ Average Flow Time

1.4 ขอบเขตของงานวิจัย

1.4.1 ศึกษาปัญหาการจัดตารางการผลิตในระบบ Flexible Flow Shop ที่มีหลายงานหลายขั้นตอน และมีเครื่องจักรขนานในแต่ละขั้นตอน

1.4.2 พิจารณาเครื่องจักรที่มีประสิทธิภาพแตกต่างกัน (Unrelated Parallel Machines)

1.4.3 กำหนดตัวชี้วัดประสิทธิภาพ ได้แก่ Makespan และ Average Flow Time

1.4.4 ใช้ Genetic Algorithm เป็นวิธีหลักในการค้นหาคำตอบ และใช้ Brute Force เป็นวิธีเปรียบเทียบ

1.4.5 ใช้โปรแกรม Python ในการจำลองและประเมินผลการทดลอง

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ได้แนวทางและแบบจำลองในการจัดตารางการผลิตที่สามารถนำไปประยุกต์ใช้ในอุตสาหกรรมจริง

1.5.2 ได้วิธีการประยุกต์ใช้ Genetic Algorithm สำหรับปัญหาที่มีความซับซ้อนสูง

1.5.3 สามารถลดระยะเวลาการผลิตรวม (Makespan) และเพิ่มประสิทธิภาพการใช้ทรัพยากรเครื่องจักร

1.5.4 เป็นแนวทางสำหรับการพัฒนาระบบสนับสนุนการตัดสินใจ (Decision Support System) ในงานวางแผนการผลิต

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1 ปัญหาการจัดตารางการผลิต

ปัญหาการผลิตในอุตสาหกรรมมีความหลากหลายและซับซ้อน ขึ้นอยู่กับประเภทของอุตสาหกรรมและลักษณะของกระบวนการผลิต ปัญหาเหล่านี้ส่งผลกระทบต่อประสิทธิภาพการผลิต ต้นทุน คุณภาพ และความสามารถในการแข่งขันขององค์กร โดยปัญหาที่พบได้บ่อยในระบบการผลิตสามารถแบ่งออกได้ดังนี้

2.1.1 ปัญหาด้านกระบวนการผลิต

ปัญหาด้านกระบวนการผลิตเป็นปัญหาที่ส่งผลโดยตรงต่อประสิทธิภาพของระบบการผลิต โดยปัญหาที่พบได้บ่อย ได้แก่ การเกิดคอขวด (Bottleneck) ซึ่งเป็นขั้นตอนที่จำกัดกำลังการผลิตโดยรวมของระบบ นอกจากนี้ยังพบปัญหาของเสีย (Waste) ที่เกิดจากการสูญเสียวัตถุดิบ เวลา และทรัพยากร รวมถึงความผันผวนของกระบวนการผลิต (Variability) ที่ทำให้การผลิตขาดความต่อเนื่อง อีกทั้งยังมีปัญหาการหยุดทำงานของเครื่องจักร (Downtime) และการวางแผนการผลิตที่ไม่มีประสิทธิภาพ ซึ่งส่งผลให้เกิดความล่าช้าในการผลิตและต้นทุนที่เพิ่มสูงขึ้น

2.1.2 ปัญหาด้านคุณภาพ

ปัญหาด้านคุณภาพเป็นอีกปัจจัยสำคัญที่ส่งผลต่อความน่าเชื่อถือขององค์กร โดยปัญหาที่พบ ได้แก่ ผลิตภัณฑ์ไม่ได้มาตรฐาน การควบคุมคุณภาพไม่ทั่วถึง และความไม่สอดคล้องของผลิตภัณฑ์กับข้อกำหนดของลูกค้า ปัญหาเหล่านี้อาจทำให้เกิดของเสีย การส่งคืนสินค้า และความไม่พึงพอใจของลูกค้า

2.1.3 ปัญหาด้านทรัพยากร

ปัญหาด้านทรัพยากรเกี่ยวข้องกับการใช้ทรัพยากรในการผลิตให้เกิดประสิทธิภาพสูงสุด โดยอาจเกิดจากการขาดแคลนวัตถุดิบ ปัญหาด้านแรงงานและทักษะของพนักงาน รวมถึงการใช้เครื่องจักรที่ล้าสมัยหรือเกิดการขัดข้องบ่อยครั้ง ซึ่งส่งผลกระทบต่อความสามารถในการผลิตและต้นทุนการดำเนินงาน

2.1.4 ปัญหาด้านการจัดการ

ปัญหาด้านการจัดการมักเกิดจากการสื่อสารภายในองค์กรที่ไม่ชัดเจน การตัดสินใจที่ไม่มีข้อมูลสนับสนุนที่เพียงพอ และการขาดการปรับปรุงกระบวนการทำงานอย่างต่อเนื่อง ส่งผลให้การวางแผนและควบคุมการผลิตไม่มีประสิทธิภาพ

2.1.5 ปัญหาด้านเทคโนโลยี

ปัญหาด้านเทคโนโลยีเกิดจากการเลือกใช้เทคโนโลยีที่ไม่เหมาะสมกับระบบการผลิต การขาดการปรับปรุงและอัปเดตเทคโนโลยี รวมถึงการเชื่อมโยงระบบสารสนเทศที่ไม่มีประสิทธิภาพ ซึ่งอาจส่งผลให้ข้อมูลในการวางแผนและตัดสินใจมีความผิดพลาดหรือไม่ทันต่อสถานการณ์ จากปัญหาดังกล่าว จะเห็นได้ว่าการจัดตารางการผลิตมีบทบาทสำคัญต่อการเพิ่มประสิทธิภาพของระบบการผลิต โดยเฉพาะในระบบที่มีหลายขั้นตอนและหลายเครื่องจักร ซึ่งต้องอาศัยวิธีการที่สามารถจัดการกับความซับซ้อนได้อย่างมีประสิทธิภาพ

2.2 ทฤษฎีที่เกี่ยวข้อง

การจัดตารางการผลิต (Scheduling Theory) เป็นศาสตร์ที่เกี่ยวข้องกับการจัดสรรทรัพยากร เช่น เครื่องจักร คนงาน และวัตถุดิบ ให้กับงานต่าง ๆ ในช่วงเวลาที่เหมาะสม เพื่อให้บรรลุเป้าหมายของระบบการผลิต เช่น การลดเวลาการผลิตรวม (Makespan) การลดต้นทุน และการส่งมอบงานตรงเวลา [1]

แนวคิดพื้นฐานของการจัดตารางการผลิตประกอบด้วยทรัพยากร (Resources) งาน (Jobs) ข้อจำกัด (Constraints) และเป้าหมาย (Objectives) โดยการจัดตารางการผลิตที่ดีจะช่วยเพิ่มประสิทธิภาพการใช้ทรัพยากร ลดเวลาว่างของเครื่องจักร และลดต้นทุนในการผลิต [2]

2.2.1 การจัดตารางการผลิตแบบ Flow Shop

เป็นระบบที่งานทุกงานต้องผ่านขั้นตอนการผลิตเดียวกันตามลำดับที่กำหนดไว้ล่วงหน้า เครื่องจักรถูกจัดเรียงตามลำดับของกระบวนการผลิตอย่างชัดเจน [1] เป้าหมายหลักของระบบนี้คือการลด Makespan และเพิ่มประสิทธิภาพการใช้เครื่องจักร

2.2.2 การจัดตารางการผลิตแบบ Job Shop

เป็นระบบที่งานแต่ละงานมีลำดับการผลิตแตกต่างกัน ทำให้มีเส้นทางการไหลของงานหลากหลายและมีความซับซ้อนสูง [3] ปัญหาประเภทนี้มักถูกจัดอยู่ในกลุ่ม NP-hard ซึ่งยากต่อการหาคำตอบที่เหมาะสมที่สุด

2.2.3 การจัดตารางการผลิตแบบ Parallel Machine

เป็นระบบที่มีเครื่องจักรหลายเครื่องสามารถทำงานชนิดเดียวกันได้พร้อมกัน โดยแบ่งออกเป็น Identical Parallel Machines, Uniform Parallel Machines และ Unrelated Parallel Machines [1] ซึ่งในกรณีของ Unrelated Parallel Machines เวลาในการประมวลผลจะขึ้นอยู่กับชนิดของงานและเครื่องจักร ทำให้ปัญหามีความซับซ้อนสูงขึ้น

2.2.4 Flexible Flow Shop Scheduling

เป็นระบบที่ผสมผสานระหว่าง Flow Shop และ Parallel Machine โดยในแต่ละขั้นตอนการผลิตสามารถมีเครื่องจักรมากกว่าหนึ่งเครื่องที่ทำงานเดียวกันได้ [4] ระบบดังกล่าวมีความยืดหยุ่นสูง แต่ทำให้ปัญหาการจัดตารางการผลิตมีความซับซ้อนมากขึ้น

2.2.5 วิธี Exact และ Heuristic

วิธี Exact เช่น Brute Force และ Integer Linear Programming (ILP) สามารถหาคำตอบที่เหมาะสมที่สุดได้ แต่เมื่อจำนวนงานเพิ่มขึ้น เวลาในการคำนวณจะเพิ่มขึ้นอย่างรวดเร็ว [2] ดังนั้น งานวิจัยจำนวนมากจึงใช้วิธี Heuristic เช่น Shortest Processing Time (SPT) และ Longest Processing Time (LPT) เพื่อช่วยลดเวลาในการคำนวณ แม้ว่าวิธีดังกล่าวอาจไม่สามารถให้คำตอบที่ดีที่สุดได้ในทุกกรณี [2]

2.2.6 Metaheuristic และ Genetic Algorithm

Metaheuristic เป็นวิธีการค้นหาคำตอบสำหรับปัญหาที่มีความซับซ้อนสูง โดยสามารถค้นหาคำตอบที่มีคุณภาพดีภายในเวลาที่เหมาะสม ตัวอย่างวิธี Metaheuristic ที่นิยมใช้ ได้แก่ Genetic Algorithm (GA), Simulated Annealing (SA), Tabu Search และ Particle Swarm Optimization (PSO) [5]-[7]

Genetic Algorithm เป็นอัลกอริทึมที่พัฒนาขึ้นโดย Holland [5] ซึ่งเลียนแบบกระบวนการวิวัฒนาการทางธรรมชาติ ประกอบด้วยกระบวนการสำคัญ ได้แก่ Selection, Crossover และ Mutation

Genetic Algorithm มีจุดเด่นคือสามารถค้นหาคำตอบในพื้นที่คำตอบขนาดใหญ่ และสามารถหลีกเลี่ยงการติดอยู่ในคำตอบเฉพาะที่ (Local Optimum) ได้ดี [8]

2.2.7 Makespan และ Average Flow Time

Makespan คือเวลาที่ใช้ในการผลิตงานทั้งหมดให้เสร็จสมบูรณ์ ซึ่งเป็นตัวชี้วัดสำคัญของประสิทธิภาพระบบการผลิต [1]

Average Flow Time คือค่าเฉลี่ยของเวลาที่งานแต่ละงานใช้ภายในระบบ ตั้งแต่เริ่มเข้าสู่กระบวนการจนเสร็จสิ้นการผลิต [2]

การลด Makespan และ Average Flow Time จะช่วยเพิ่มประสิทธิภาพการใช้เครื่องจักร ลดเวลารอคอย และเพิ่มความสามารถในการแข่งขันขององค์กร

2.3 งานวิจัยที่เกี่ยวข้อง

ปัญหาการจัดตารางการผลิต โดยเฉพาะในระบบ Flexible Flow Shop และ Flexible Job Shop เป็นปัญหาที่มีความซับซ้อนสูงและจัดอยู่ในกลุ่ม NP-hard [1] ดังนั้นจึงมีงานวิจัยจำนวนมากมุ่งเน้นการพัฒนาอัลกอริทึมสำหรับหาคำตอบที่มีประสิทธิภาพภายในเวลาที่จำกัด

Ruiz และ Maroto [9] ได้นำเสนอ Genetic Algorithm สำหรับ Hybrid Flow Shop Scheduling ที่มี setup time และ machine eligibility โดยผลการทดลองพบว่า GA สามารถลด Makespan ได้อย่างมีประสิทธิภาพในปัญหาขนาดใหญ่

Brandimarte [10] ศึกษาปัญหา Flexible Job Shop Scheduling โดยใช้ Tabu Search ในการค้นหาคำตอบ และได้พัฒนา benchmark dataset ซึ่งถูกนำมาใช้อย่างแพร่หลายในงานวิจัยด้าน Scheduling

Gao et al. [11] พัฒนา Hybrid Genetic Algorithm ร่วมกับ Local Search และ Variable Neighborhood Descent สำหรับ Flexible Job Shop Scheduling ซึ่งสามารถลด Makespan ได้ดีกว่า Genetic Algorithm แบบดั้งเดิม

Kacem et al. [12] ศึกษาปัญหา Flexible Job Shop Scheduling แบบหลายวัตถุประสงค์ โดยใช้ Evolutionary Algorithm ร่วมกับ Fuzzy Logic เพื่อพิจารณาทั้ง Makespan และภาระงานของเครื่องจักรพร้อมกัน

Pezzella et al. [13] ได้นำเสนอการปรับปรุง population, crossover และ mutation ของ Genetic Algorithm เพื่อเพิ่มประสิทธิภาพในการแก้ปัญหา Flexible Job Shop Scheduling ส่งผลให้ได้คำตอบที่มีคุณภาพสูงขึ้น

Li และ Gao [14] พัฒนา Hybrid Genetic Algorithm ร่วมกับ Tabu Search เพื่อเพิ่มประสิทธิภาพในการค้นหาคำตอบและลดโอกาสการติดอยู่ในคำตอบเฉพาะที่

Zhang et al. [15] พัฒนา Hybrid Particle Swarm Optimization (PSO) สำหรับ Flexible Job Shop Scheduling แบบหลายวัตถุประสงค์ ซึ่งสามารถลด Makespan และเพิ่มประสิทธิภาพการใช้เครื่องจักรได้

Nowicki และ Smutnicki [7] ได้นำเสนอ Tabu Search Algorithm สำหรับ Job Shop Scheduling โดยสามารถค้นหาคำตอบได้อย่างรวดเร็วและมีประสิทธิภาพในการใช้กับปัญหาที่มีความซับซ้อนสูง

Jain และ Meeran [16] ได้ศึกษาทบทวนวรรณกรรมเกี่ยวกับ Job Shop Scheduling โดยสรุปแนวโน้ม วิธีการ และความท้าทายของปัญหาการจัดตารางการผลิตในอดีต ปัจจุบัน และอนาคต

ในประเทศไทย สุภาวดี บุญช่วย และคณะ [17] ศึกษาการจัดตารางงานแบบ Flexible Job Shop โดยใช้ Enhanced Genetic Algorithm พบว่าสามารถลด Makespan ได้อย่างมีประสิทธิภาพ

พิสิฐ พงษ์ประเสริฐ [18] ศึกษาการลด Makespan ในระบบ Flexible Flow Shop โดยใช้ ILP และ Heuristic เพื่อเปรียบเทียบประสิทธิภาพของวิธี Exact และ Approximation

ณัฐวุฒิ ศรีสมบัติ และคณะ [19] ศึกษาปัญหาการจัดตารางบนเครื่องจักรแบบขนานที่ไม่สัมพันธ์กัน (Unrelated Parallel Machines) โดยใช้ Mathematical Model และ Heuristic ในการทำงานภาคอุตสาหกรรมจริง

ธนภัทร ศรีสุข [20] ศึกษาการจัดตารางงานที่มีเวลาตั้งค่าขึ้นกับลำดับงาน โดยใช้ Forward Scheduling และ Backward Scheduling เพื่อลดจำนวนเครื่องจักรและเพิ่มประสิทธิภาพการผลิต

กิตติพงษ์ แสงทอง และคณะ [21] ศึกษาการประยุกต์ใช้ Dispatching Rules ในโรงงานจริง ซึ่งสามารถลด Makespan ได้มากกว่า 12%

จากการศึกษางานวิจัยที่ผ่านมา พบว่า งานวิจัยส่วนใหญ่มุ่งเน้นการลด Makespan และเพิ่มประสิทธิภาพการใช้เครื่องจักร อย่างไรก็ตาม ยังมีงานวิจัยจำนวนน้อยที่ศึกษา Flexible Flow Shop Scheduling ร่วมกับ Unrelated Parallel Machines และเปรียบเทียบประสิทธิภาพของ Genetic Algorithm กับ Brute Force ภายใต้ข้อจำกัดเวลาในการประมวลผล ดังนั้น งานวิจัยนี้จึงมุ่งเน้นการใช้ Genetic Algorithm เพื่อแก้ปัญหาดังกล่าว โดยพิจารณาทั้ง Makespan และ Average Flow Time ภายใต้เวลาการประมวลผลที่กำหนด

ตารางที่ 2.1 ตารางเปรียบเทียบงานวิจัยที่เกี่ยวข้อง

ลำดับ	ผู้วิจัย	ลักษณะงานวิจัย	วิธีที่ใช้	จุดเด่นของงานวิจัย	ความเกี่ยวข้องกับงานวิจัยนี้
1	Pinedo [1]	Scheduling Theory	Scheduling Theory	อธิบายพื้นฐาน Scheduling	ใช้เป็นพื้นฐานทางทฤษฎี
2	Baker and Trietsch [2]	Sequencing and Scheduling	Heuristic Rules	อธิบาย SPT และ LPT	ใช้เปรียบเทียบกับ GA
3	Jain and Meeran [3]	Job Shop Scheduling	Literature Review	สรุปแนวโน้ม Scheduling	ใช้อธิบายปัญหา Scheduling
4	Brandimarte [4]	Flexible Job Shop Scheduling	Tabu Search	Benchmark ของ FJSP	ใช้แนวคิดการเลือกเครื่องจักร
5	Holland [5]	Genetic Algorithm	Genetic Algorithm	พื้นฐานของ GA	ใช้อธิบายหลักการ GA
6	Kirkpatrick et al. [6]	Optimization Problem	Simulated Annealing	แนวคิด Metaheuristic	ใช้เปรียบเทียบกับ GA
7	Nowicki and Smutnicki [7]	Job Shop Scheduling	Tabu Search	ค้นหาคำตอบได้รวดเร็ว	ใช้เปรียบเทียบ Metaheuristic
8	Gen and Cheng [8]	Engineering Optimization	Genetic Algorithm	อธิบาย GA เชิงวิศวกรรม	สนับสนุนการใช้ GA
9	Ruiz and Maroto [9]	Hybrid Flow Shop Scheduling	Genetic Algorithm	ลด Makespan	สนับสนุนการใช้ GA
10	Brandimarte [10]	Flexible Job Shop Scheduling	Tabu Search	Benchmark ของ FJSP	ใช้แนวคิดการเลือกเครื่องจักร

ตารางที่ 2.1 ตารางเปรียบเทียบงานวิจัยที่เกี่ยวข้อง (ต่อ)

ลำดับ	ผู้วิจัย	ลักษณะงานวิจัย	วิธีที่ใช้	จุดเด่นของงานวิจัย	ความเกี่ยวข้องกับงานวิจัยนี้
11	Gao et al. [11]	Flexible Job Shop Scheduling	Hybrid GA	ลด Makespan ได้ดี	สนับสนุน Hybrid GA
12	Kacem et al. [12]	Multi-objective FJSP	Evolutionary Algorithm	พิจารณาหลายวัตถุประสงค์	เกี่ยวข้องกับ Makespan
13	Pezzella et al. [13]	Flexible Job Shop Scheduling	Genetic Algorithm	ปรับปรุง crossover และ mutation	ใกล้เคียงโครงสร้าง GA
14	Li and Gao [14]	Flexible Job Shop Scheduling	Hybrid GA + Tabu Search	ลด Local Optimum	สนับสนุน Hybrid GA
15	Zhang et al. [15]	Flexible Job Shop Scheduling	Hybrid PSO	ลด Makespan	เปรียบเทียบกับ PSO
16	Jain and Meeran [16]	Literature Review	Literature Review	สรุปแนวโน้ม Scheduling	ใช้อ้างความซับซ้อน
17	สุภาวดี บุญช่วย และคณะ [17]	Flexible Job Shop Scheduling	Enhanced GA	ลด Makespan	ใกล้เคียงงานวิจัยนี้
18	พิสิฐ พงษ์ประเสริฐ [18]	Flexible Flow Shop Scheduling	ILP + Heuristic	เปรียบเทียบ exact และ heuristic	เปรียบเทียบกับ Brute Force
19	ณัฐวุฒิ ศรีสมบัติ และคณะ [19]	Unrelated Parallel Machines	Mathematical Model + Heuristic	ใช้กับอุตสาหกรรมจริง	ตรงกับปัญหาใน thesis

ตารางที่ 2.1 ตารางเปรียบเทียบงานวิจัยที่เกี่ยวข้อง (ต่อ)

ลำดับ	ผู้วิจัย	ลักษณะงานวิจัย	วิธีที่ใช้	จุดเด่นของงานวิจัย	ความเกี่ยวข้องกับงานวิจัยนี้
20	ธนภัทร ศรีสุข [20]	Scheduling with Setup Time	Forward / Backward Scheduling	ลดจำนวนเครื่องจักร	อธิบายผลกระทบของลำดับงาน
21	กิตติพงษ์ แสงทอง และคณะ [21]	Job Shop Production	Dispatching Rules	ลด Makespan ได้มากกว่า 12%	ใช้เป็นตัวอย่างโรงงานจริง

จากตารางที่ 2.1 พบว่า งานวิจัยส่วนใหญ่มุ่งเน้นการแก้ปัญหาการจัดตารางการผลิตในระบบ Flexible Flow Shop และ Flexible Job Shop ซึ่งเป็นปัญหาที่มีความซับซ้อนสูงและจัดอยู่ในกลุ่ม NP-hard โดยมีการนำวิธีการต่าง ๆ มาใช้ในการแก้ปัญหา เช่น Heuristic, Tabu Search, Simulated Annealing, Particle Swarm Optimization และ Genetic Algorithm

จากการเปรียบเทียบพบว่า Genetic Algorithm เป็นวิธีที่ได้รับความนิยมอย่างแพร่หลาย เนื่องจากสามารถค้นหาคำตอบในพื้นที่คำตอบขนาดใหญ่ได้อย่างมีประสิทธิภาพ และสามารถลดค่า Makespan ได้ดีในปัญหาขนาดใหญ่ [9], [11], [13] นอกจากนี้ ยังมีการพัฒนา Hybrid Genetic Algorithm ร่วมกับวิธีการอื่น เช่น Tabu Search และ Local Search เพื่อเพิ่มประสิทธิภาพในการค้นหาคำตอบและลดโอกาสการติดอยู่ในคำตอบเฉพาะที่

สำหรับงานวิจัยในประเทศไทย พบว่ามีการประยุกต์ใช้ Genetic Algorithm และ Heuristic กับระบบการผลิตจริง โดยสามารถช่วยลด Makespan และเพิ่มประสิทธิภาพการใช้เครื่องจักรได้ อย่างไรก็ตาม งานวิจัยส่วนใหญ่ยังมุ่งเน้นเฉพาะการลด Makespan และยังมีการศึกษาจำนวนน้อยที่พิจารณาปัญหา Flexible Flow Shop ร่วมกับเครื่องจักรแบบ Unrelated Parallel Machines รวมถึงการวิเคราะห์ Average Flow Time ร่วมกับ Makespan

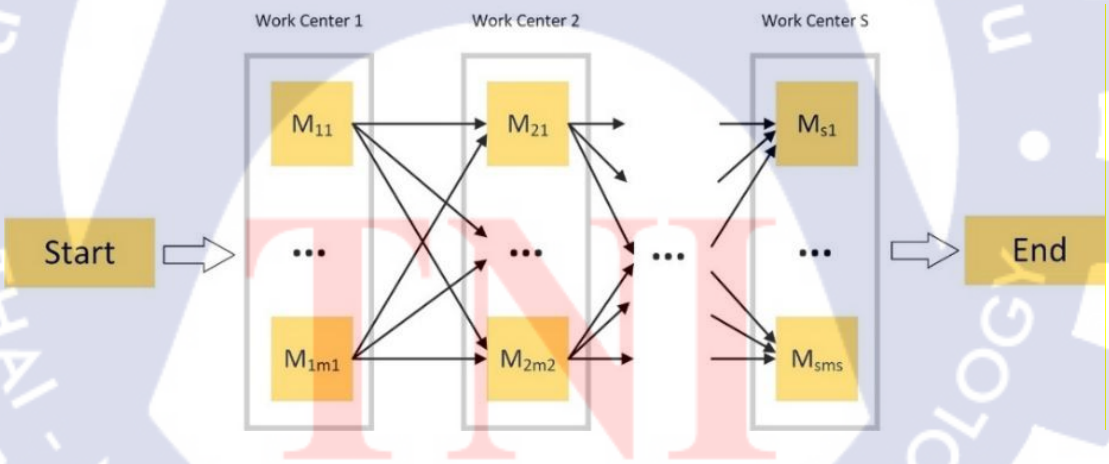
ดังนั้น งานวิจัยนี้จึงมุ่งเน้นการใช้ Genetic Algorithm สำหรับการจัดตารางการผลิตในระบบ Flexible Flow Shop ที่มีเครื่องจักรแบบ Unrelated Parallel Machines โดยเปรียบเทียบประสิทธิภาพกับวิธี Brute Force ภายใต้ข้อจำกัดเวลาในการประมวลผล เพื่อให้ได้แนวทางที่เหมาะสมสำหรับการประยุกต์ใช้ในระบบการผลิตจริง

บทที่ 3
วิธีการวิจัย

3.1 การอธิบายปัญหาและระบบการผลิต

ในอุตสาหกรรมการผลิตสมัยใหม่ การจัดตารางการผลิต (Production Scheduling) เป็นปัจจัยสำคัญที่มีผลต่อประสิทธิภาพของระบบการผลิต โดยเฉพาะในระบบที่มีความยืดหยุ่นสูงและมีเครื่องจักรหลายเครื่องที่สามารถทำงานทดแทนกันได้ ปัญหาการจัดตารางการผลิตในลักษณะดังกล่าวมักถูกจัดอยู่ในกลุ่มของ Flexible flow shop (FFS) ซึ่งเป็นปัญหาที่มีความซับซ้อนสูงและจัดอยู่ในกลุ่มปัญหา NP-hard กล่าวคือ เมื่อจำนวนงานและจำนวนเครื่องจักรเพิ่มขึ้น จำนวนวิธีในการจัดตารางการผลิตจะเพิ่มขึ้นอย่างรวดเร็ว ทำให้ไม่สามารถหาคำตอบที่เหมาะสมที่สุดได้ง่ายด้วยวิธีการคำนวณแบบทั่วไป

งานวิจัยนี้มุ่งศึกษาปัญหาการจัดตารางการผลิตในระบบ Flexible flow shop โดยมีวัตถุประสงค์เพื่อค้นหาลำดับการผลิตของงานที่เหมาะสมที่สุด เพื่อลดเวลาการผลิตรวมของระบบให้ต่ำที่สุด โดยใช้ อัลกอริทึมเชิงพันธุกรรม Genetic Algorithm (GA) เป็นเครื่องมือในการค้นหาคำตอบที่เหมาะสม



รูปที่ 3.1 แสดงตัวอย่างแบบจำลองการผลิต

ระบบการผลิตที่ใช้ในการศึกษาประกอบด้วย งานจำนวน 6 งาน (Jobs) โดยแต่ละงานต้องผ่านกระบวนการผลิตจำนวน 3 ขั้นตอน (Stages) ตามลำดับ ได้แก่ ขั้นตอนที่ 1 ขั้นตอนที่ 2 และขั้นตอนที่ 3 ซึ่งสอดคล้องกับกระบวนการผลิตแบบลำดับขั้น (Sequential Processing) กล่าวคือ งานแต่ละงานต้องดำเนินการในขั้นตอนก่อนหน้าให้เสร็จสิ้นก่อนจึงจะสามารถเข้าสู่ขั้นตอนถัดไปได้

ในแต่ละขั้นตอนการผลิตมีเครื่องจักรจำนวน 2 เครื่องจักร ที่สามารถใช้ทำงานแทนกันได้ โดยเครื่องจักรแต่ละเครื่องมีเวลาการผลิตที่แตกต่างกัน โดยทั่วไปเครื่องจักรกลุ่มที่สองจะมีประสิทธิภาพสูงกว่าและใช้เวลาในการผลิตสั้นกว่าเครื่องจักรกลุ่มแรก ดังนั้นในการจัดตารางการผลิต ระบบจะต้องพิจารณาทั้งลำดับของงานและการเลือกเครื่องจักรที่เหมาะสมในแต่ละขั้นตอน เพื่อให้เวลาการผลิตรวมของระบบมีค่าต่ำที่สุด

ในระบบการผลิตจริง เครื่องจักรที่อยู่ในขั้นตอนเดียวกันมักมีประสิทธิภาพการทำงานที่แตกต่างกัน ไม่ว่าจะเป็นความเร็วในการผลิต เทคโนโลยีของเครื่องจักร หรือสภาพการใช้งาน ส่งผลให้เวลาในการประมวลผลของงานบนเครื่องจักรแต่ละเครื่องไม่เท่ากัน ลักษณะดังกล่าวเรียกว่า Non-uniform Performance Machine ซึ่งหมายถึงเครื่องจักรที่มีความสามารถในการประมวลผลแตกต่างกัน โดยงานเดียวกันเมื่อถูกประมวลผลบนเครื่องจักรต่างเครื่องจะใช้เวลาไม่เท่ากัน ในงานวิจัยนี้ เครื่องจักรถูกแบ่งออกเป็น 2 กลุ่ม ได้แก่

- เครื่องจักรกลุ่มที่ 1 (Slow Machine) มีเวลาในการผลิตอยู่ในช่วง 11–15 หน่วยเวลา
- เครื่องจักรกลุ่มที่ 2 (Fast Machine) มีเวลาในการผลิตอยู่ในช่วง 5–10 หน่วยเวลา

ซึ่งแสดงให้เห็นว่าเครื่องจักรแต่ละเครื่องมีประสิทธิภาพแตกต่างกันอย่างชัดเจน โดยเครื่องจักรกลุ่มที่สองสามารถประมวลผลงานได้เร็วกว่า ดังนั้น ปัญหาการจัดตารางการผลิตในงานวิจัยนี้จึงไม่เพียงแต่ต้องพิจารณาลำดับของงาน (Job Sequence) เท่านั้น แต่ยังต้องพิจารณาการเลือกเครื่องจักร (Machine Assignment) ที่เหมาะสมในแต่ละขั้นตอนร่วมด้วย เพื่อให้ได้ผลลัพธ์ที่ดีที่สุดในด้าน Makespan และ Average Flow Time ลักษณะของระบบดังกล่าวสอดคล้องกับปัญหา Flexible flow shop (FFS) ที่มีเครื่องจักรแบบ heterogeneous หรือ non-uniform ซึ่งเพิ่มความซับซ้อนของปัญหาและทำให้การค้นหาคำตอบที่เหมาะสมทำได้ยากขึ้น

ในการจำลองระบบการผลิต งานวิจัยนี้กำหนดให้เวลาในการผลิตของเครื่องจักรถูกกำหนดในรูปของช่วงค่า (Range) เพื่อสะท้อนลักษณะความไม่แน่นอนของกระบวนการผลิตจริง โดยกำหนดให้เครื่องจักรกลุ่มแรกมีช่วงเวลาในการผลิตระหว่าง 11–15 หน่วยเวลา และเครื่องจักรกลุ่มที่สองมีช่วงเวลาในการผลิตระหว่าง 5–10 หน่วยเวลา ซึ่งค่าดังกล่าวจะถูกสุ่มในแต่ละรอบของการทดลอง

ในการประเมินประสิทธิภาพของตารางการผลิต งานวิจัยนี้ใช้ตัวชี้วัดหลักสองตัว ได้แก่ Makespan และ Average Flow Time โดย Makespan หมายถึง เวลาที่ใช้ในการทำงานจนกระทั่งงานสุดท้ายในระบบเสร็จสิ้นทั้งหมด ส่วน Average Flow Time หมายถึง ค่าเฉลี่ยของเวลาที่งานแต่ละงานใช้ในระบบการผลิตตั้งแต่เริ่มต้นจนเสร็จสิ้น

จากลักษณะของปัญหาที่มีความซับซ้อนสูง งานวิจัยนี้จึงเลือกใช้ Genetic Algorithm ซึ่งเป็นอัลกอริทึมในกลุ่ม Metaheuristic ที่มีประสิทธิภาพในการค้นหาคำตอบของปัญหาการจัดตาราง

การผลิต โดยอัลกอริทึมจะจำลองกระบวนการวิวัฒนาการตามธรรมชาติ ผ่านกระบวนการคัดเลือก (Selection) การผสมพันธุ์ (Crossover) และการกลายพันธุ์ (Mutation) เพื่อค้นหาตารางการผลิตที่ให้ค่า Makespan และ Average Flow Time ต่ำที่สุด

เพื่อดำเนินการทดลอง แบบจำลองของระบบการผลิตและอัลกอริทึม Genetic Algorithm ถูกพัฒนาโดยใช้ภาษา Python และทำการรันบนแพลตฟอร์ม Google Colab โดยโปรแกรมจะทำการจำลองกระบวนการผลิต คำนวณเวลาเริ่มต้นและเวลาสิ้นสุดของงานในแต่ละขั้นตอน รวมทั้งประเมินค่าประสิทธิภาพของตารางการผลิตที่ได้จากอัลกอริทึม

งานวิจัยนี้ยังได้ทำการเปรียบเทียบผลลัพธ์กับวิธีการค้นหาคำตอบแบบ Brute Force เพื่อใช้เป็นค่ามาตรฐาน (Benchmark) สำหรับประเมินประสิทธิภาพของอัลกอริทึม วิธี Brute Force เป็นวิธีการค้นหาคำตอบที่เป็นไปได้ทั้งหมดของปัญหา โดยทำการสร้างลำดับการผลิตของงานทุกกรณีที่เป็นไปได้ และคำนวณค่าประสิทธิภาพของตารางการผลิตแต่ละลำดับ จากนั้นจึงเลือกคำตอบที่ให้ค่าเป้าหมายดีที่สุด วิธีดังกล่าวสามารถให้คำตอบที่เหมาะสมที่สุด (Optimal Solution) ได้อย่างแน่นอน อย่างไรก็ตาม วิธี Brute Force มีข้อจำกัดสำคัญคือจำนวนลำดับที่ต้องคำนวณจะเพิ่มขึ้นอย่างรวดเร็วตามจำนวนงานในระบบ

ตารางที่ 3.1 ตัวอย่างเวลาของเครื่องจักร

Job	Stage	Machine 1 (ช้า)	Machine 2 (เร็ว)
J1	Stage 1	11	6
J1	Stage 2	12	8
J1	Stage 3	12	8
J2	Stage 1	14	9
J2	Stage 2	12	7
J2	Stage 3	15	5
J3	Stage 1	11	7
J3	Stage 2	15	9
J3	Stage 3	13	10
J4	Stage 1	11	7
J4	Stage 2	13	9
J4	Stage 3	15	9

ตารางที่ 3.1 ตัวอย่างเวลาของเครื่องจักร (ต่อ)

Job	Stage	Machine 1 (ช้า)	Machine 2 (เร็ว)
J5	Stage 1	11	7
J5	Stage 2	13	7
J5	Stage 3	12	9

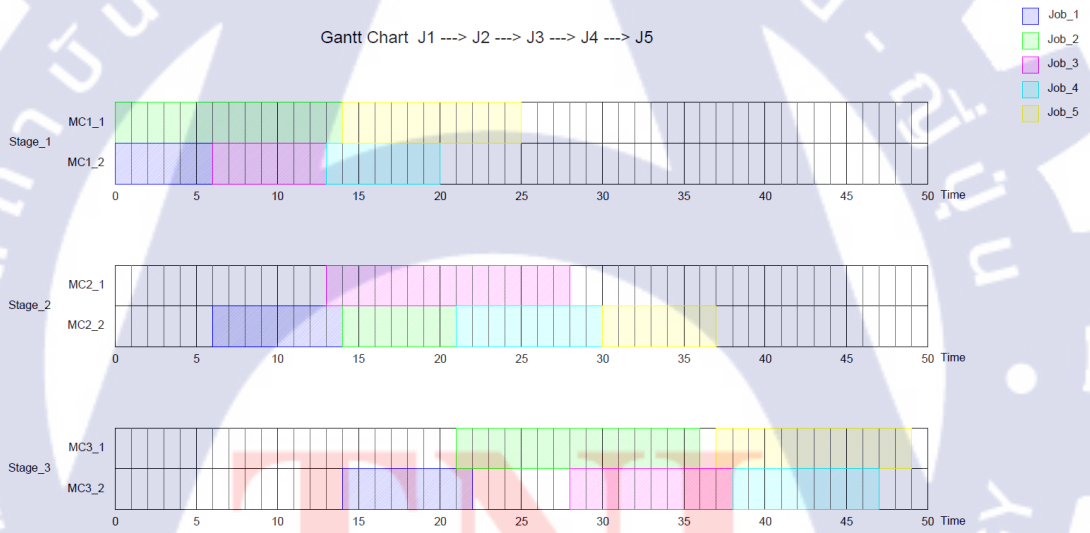
ตารางที่ 3.2 ตัวอย่างเวลาเริ่มและสิ้นสุดงานของลำดับงาน J1 ⟶ J2 ⟶ J3⟶ J4⟶ J5

Sequence	Job	Stage	Start Time	End Time
J1 ⟶ J2 ⟶ J3⟶ J4⟶ J5	J1	Stage 1	0	6
	J1	Stage 2	6	13
	J1	Stage 3	13	22
	J2	Stage 1	0	14
	J2	Stage 2	14	21
	J2	Stage 3	21	36
	J3	Stage 1	6	13
	J3	Stage 2	13	28
	J3	Stage 3	28	30
	J4	Stage 1	13	20
	J4	Stage 2	21	30
	J4	Stage 3	38	47
	J5	Stage 1	14	25
	J5	Stage 2	30	36
	J5	Stage 3	37	49

จากตารางจะเห็นได้ว่า งานแต่ละงานต้องผ่านกระบวนการผลิตจำนวน 3 ขั้นตอน (Stages) ตามลำดับ โดยเวลาเริ่มต้นของแต่ละขั้นตอนจะถูกกำหนดจากเงื่อนไขสำคัญ 2 ประการ ได้แก่ 1.งานต้องเสร็จสิ้นจากขั้นตอนก่อนหน้าแล้ว 2.เครื่องจักรที่ใช้ในขั้นตอนถัดไปต้องพร้อมใช้งาน ดังนั้น เวลาเริ่มต้นของงานในแต่ละขั้นตอนจึงขึ้นอยู่กับทั้งสถานะของงานและสถานะความพร้อมของเครื่องจักร

ตัวอย่างเช่น งาน J1 เริ่มทำงานใน Stage 1 ที่เวลา 0 และเสร็จสิ้นที่เวลา 6 จากนั้นจึงสามารถเข้าสู่ Stage 2 ได้ที่เวลา 6 และเสร็จสิ้นที่เวลา 13 ก่อนเข้าสู่ Stage 3 และเสร็จสิ้นทั้งหมดที่เวลา 22 ในขณะที่งาน J4 แม้ว่าจะเสร็จสิ้น Stage 2 ที่เวลา 30 แต่ไม่สามารถเริ่มทำงานใน Stage 3 ได้ทันที เนื่องจากเครื่องจักรใน Stage 3 ยังไม่พร้อมใช้งาน ทำให้ต้องรอจนถึงเวลา 38 จึงจะสามารถเริ่มทำงานได้ ซึ่งสะท้อนให้เห็นข้อจำกัดของการใช้ทรัพยากรเครื่องจักรร่วมกันในระบบ Flexible Job Shop

จากตารางยังแสดงให้เห็นว่า งานบางงานสามารถเริ่มทำงานพร้อมกันได้ในขั้นตอนเดียวกัน หากมีเครื่องจักรมากกว่าหนึ่งเครื่องที่พร้อมใช้งาน เช่น งาน J1 และ J2 ใน Stage 1 ซึ่งเริ่มต้นที่เวลา 0 พร้อมกันบนเครื่องจักรคนละเครื่อง ผลลัพธ์จากตารางดังกล่าวถูกนำไปใช้ในการคำนวณค่า Makespan และ Average Flow Time รวมถึงใช้สำหรับสร้างแผนภาพ Gantt Chart เพื่อแสดงภาพรวมของตารางการผลิตในขั้นตอนถัดไป



รูปที่ 3.2 แสดงตัวอย่าง Gantt Chart ของลำดับงาน J1 ⟶ J2 ⟶ J3⟶ J4⟶ J5

สำหรับกรณีศึกษาที่ใช้ในงานวิจัยนี้ ซึ่งประกอบด้วยงานจำนวน 5 งาน จำนวนลำดับการผลิตที่เป็นไปได้ทั้งหมดสามารถคำนวณได้จากแฟกทอเรียลของจำนวนงาน ดังนี้ 5! = 120

กล่าวคือ จำเป็นต้องคำนวณตารางการผลิตทั้งหมด 120 ลำดับ เพื่อค้นหาคำตอบที่ดีที่สุด เมื่อจำนวนงานเพิ่มขึ้น จำนวนลำดับที่เป็นไปได้จะเพิ่มขึ้นอย่างรวดเร็ว ตัวอย่างเช่น

ตารางที่ 3.3 จำนวนลำดับงานที่เป็นไปได้

จำนวนงาน	จำนวนลำดับที่เป็นไปได้
5	120
6	720
7	5,040
10	3,628,800

จากตารางที่ 3.3 จะเห็นได้ว่าหากใช้วิธี Brute Force จะต้องคำนวณลำดับงานเพิ่มขึ้นอย่างมาก เมื่อจำนวนงานเพิ่มขึ้น ทำให้ไม่เหมาะสำหรับการใช้งานกับระบบการผลิตขนาดใหญ่ ดังนั้นงานวิจัยนี้จึงใช้ Genetic Algorithm ซึ่งเป็นอัลกอริทึมในกลุ่ม Metaheuristic ที่สามารถค้นหาคำตอบที่ดีได้ภายในเวลาการคำนวณที่เหมาะสม โดยไม่จำเป็นต้องตรวจสอบทุกลำดับที่เป็นไปได้ เหมือนกับวิธี Brute Force แต่งานวิจัยนี้ใช้วิธี Brute Force เพื่อคำนวณคำตอบที่ดีที่สุดของกรณีศึกษาขนาดเล็ก และนำผลลัพธ์ดังกล่าวมาใช้เปรียบเทียบกับผลลัพธ์ที่ได้จาก Genetic Algorithm เพื่อประเมินประสิทธิภาพของอัลกอริทึมในการค้นหาตารางการผลิตที่ให้ค่า Makespan และ Average Flow Time ต่ำที่สุด

3.2 ตัวแปรและสมมติฐานของแบบจำลอง

การสร้างแบบจำลองสำหรับปัญหาการจัดตารางการผลิตในระบบ Flexible Job Shop มีการกำหนดตัวแปรและสัญลักษณ์ที่ใช้ในแบบจำลอง เพื่ออธิบายโครงสร้างของปัญหาและใช้ในการพัฒนาอัลกอริทึมสำหรับการแก้ปัญหาได้อย่างเป็นระบบ โดยในงานวิจัยนี้ได้กำหนดสัญลักษณ์และตัวแปรที่เกี่ยวข้องดังต่อไปนี้

ตารางที่ 3.4 เซตและดัชนี

สัญลักษณ์	ความหมาย
i	ดัชนีของงาน (Job index)
j	ดัชนีของขั้นตอนการผลิต (Stage index)
k	ดัชนีของเครื่องจักร (Machine index)
J	เซตของงานทั้งหมด
S	เซตของขั้นตอนการผลิต
M_j	เซตของเครื่องจักรที่สามารถทำงานในขั้นตอนที่ j

สำหรับกรณีศึกษาที่ใช้ในงานวิจัยนี้ กำหนดให้มีจำนวน 6 งาน (Jobs) โดยแต่ละงานต้องผ่านกระบวนการผลิตจำนวน 3 ขั้นตอน (Stages) ตามลำดับ ได้แก่ ขั้นตอนที่ 1 ขั้นตอนที่ 2 และขั้นตอนที่ 3 ในแต่ละขั้นตอนการผลิตมีเครื่องจักรจำนวน 2 เครื่องจักร (Machine) ที่สามารถใช้ทำงานแทนกันได้

ตารางที่ 3.5 พารามิเตอร์

สัญลักษณ์	ความหมาย
p_{ijk}	เวลาในการผลิตของงาน i ในขั้นตอน j บนเครื่องจักร k
C_i	เวลาเสร็จสิ้นของงาน i
C_{max}	เวลาการผลิตรวมของระบบ (Makespan)
n	จำนวนงานทั้งหมดในระบบ

พารามิเตอร์เป็นค่าคงที่ที่ใช้ในแบบจำลอง ซึ่งกำหนดจากข้อมูลของระบบการผลิต ในงานวิจัยนี้ เวลาในการผลิตของเครื่องจักรถูกกำหนดในรูปแบบช่วงค่า (Range) เพื่อสะท้อนลักษณะความไม่แน่นอนของระบบการผลิต โดยกำหนดให้ เครื่องจักรกลุ่มที่ 1 มีเวลาการผลิตในช่วง 11–15 หน่วยเวลาและเครื่องจักรกลุ่มที่ 2 มีเวลาการผลิตในช่วง 5–10 หน่วยเวลา ค่าดังกล่าวจะถูกสุ่มใหม่ในแต่ละรอบของการทดลอง

ตารางที่ 3.6 ตัวแปรการตัดสินใจ

สัญลักษณ์	ความหมาย
S_{ijk}	เวลาเริ่มต้นของงาน iในขั้นตอน jบนเครื่องจักร k
C_{ijk}	เวลาเสร็จสิ้นของงาน iในขั้นตอน jบนเครื่องจักร k
x_{ijk}	ตัวแปรที่แสดงว่างาน iถูกประมวลผลบนเครื่องจักร kในขั้นตอน j

โดยตัวแปร x_{ijk}เป็นตัวแปรแบบไบนารี ซึ่งมีค่า

$$x_{ijk} = \begin{cases} 1 & \text{ถ้างาน } i \text{ ถูกประมวลผลบนเครื่องจักร } k \text{ ในขั้นตอน } j \\ 0 & \text{กรณีอื่น} \end{cases}$$

เพื่อให้การสร้างแบบจำลองมีความชัดเจนและสามารถวิเคราะห์ปัญหาได้อย่างเป็นระบบ งานวิจัยนี้ได้กำหนดสมมติฐานของระบบการผลิตดังต่อไปนี้

1. งานทุกงานต้องผ่านทุกขั้นตอนการผลิตตามลำดับที่กำหนด
2. ในแต่ละขั้นตอน งานสามารถถูกประมวลผลได้บนเครื่องจักรเพียงหนึ่งเครื่องเท่านั้น
3. เครื่องจักรหนึ่งเครื่องสามารถทำงานได้ทีละงานในช่วงเวลาเดียวกัน
4. งานทุกงานพร้อมเข้าสู่กระบวนการผลิตตั้งแต่เวลาเริ่มต้น
5. เครื่องจักรทุกเครื่องพร้อมใช้งานตั้งแต่เริ่มต้นการผลิต
6. ไม่มีเวลาในการปรับตั้งเครื่องจักร (Setup Time)

สมมติฐานดังกล่าวช่วยให้สามารถสร้างแบบจำลองทางคณิตศาสตร์และพัฒนาอัลกอริทึมสำหรับการแก้ปัญหาได้อย่างเหมาะสม และสอดคล้องกับลักษณะของปัญหาการจัดตารางการผลิตในระบบ Flexible Job Shop

3.3 การสร้างแบบจำลองปัญหา

การสร้างแบบจำลองทางคณิตศาสตร์สำหรับปัญหาการจัดตารางการผลิตในระบบ Flexible Job Shop โดยมีวัตถุประสงค์เพื่อกำหนดลำดับการผลิตของงานและการเลือกเครื่องจักรที่เหมาะสมในแต่ละขั้นตอน เพื่อให้เวลาการผลิตรวมของระบบมีค่าต่ำที่สุด

ในการสร้างแบบจำลองปัญหา งานวิจัยนี้พิจารณาระบบการผลิตที่ประกอบด้วยงานจำนวน n งาน ซึ่งแต่ละงานต้องผ่านขั้นตอนการผลิตตามลำดับ โดยในแต่ละขั้นตอนมีเครื่องจักรหลายเครื่องที่สามารถเลือกใช้งานได้ การจัดตารางการผลิตจึงต้องพิจารณาทั้งลำดับของงานและการเลือกเครื่องจักรให้เหมาะสม

3.3.1 ฟังก์ชันวัตถุประสงค์ (Objective Function)

ในงานวิจัยนี้มีวัตถุประสงค์หลักคือการลดเวลาการผลิตรวมของระบบ หรือ Makespan ซึ่งหมายถึงเวลาที่ใช้จนกระทั่งงานสุดท้ายในระบบเสร็จสิ้น

ฟังก์ชันวัตถุประสงค์สามารถเขียนได้ดังนี้ $\min C_{max}$ โดยที่ $C_{max} = \max(C_i)$ เมื่อ C_i = เวลาเสร็จสิ้นของงานที่ i ดังนั้นเป้าหมายของแบบจำลองคือการหาตารางการผลิตที่ทำให้ค่า C_{max} มีค่าต่ำที่สุด

นอกจากนี้ งานวิจัยยังพิจารณาตัวชี้วัดประสิทธิภาพอีกตัวหนึ่งคือ Average Flow Time ซึ่งเป็นค่าเฉลี่ยของเวลาที่งานแต่ละงานใช้ในระบบการผลิต โดยสามารถคำนวณได้ดังนี้

$$F = \frac{1}{n}\sum_{i=1}^{n} F_i \quad (1)$$

โดยที่

n คือ จำนวนงานทั้งหมดในระบบ

3.3.2 ข้อจำกัดของแบบจำลอง (Model Constraints)

เพื่อให้ตารางการผลิตที่ได้มีความสอดคล้องกับระบบการผลิตจริง จำเป็นต้องกำหนดข้อจำกัดของแบบจำลองดังต่อไปนี้

3.3.2.1 ข้อจำกัดของลำดับขั้นตอนการผลิต

งานแต่ละงานต้องผ่านขั้นตอนการผลิตตามลำดับที่กำหนด โดยขั้นตอนถัดไปจะเริ่มได้ก็ต่อเมื่อขั้นตอนก่อนหน้าเสร็จสิ้นแล้ว

$$S_{i,j+1} \geq C_{ij} \quad (2)$$

โดยที่

$S_{i,j+1}$ คือ เวลาเริ่มต้น (Start Time) ของงาน iในขั้นตอนการผลิตถัดไป $(j+1)$

C_{ij} คือ เวลาเสร็จสิ้น (Completion Time) ของงาน iในขั้นตอนการผลิตที่ j

สมการดังกล่าวใช้กำหนดข้อจำกัดของลำดับขั้นตอนการผลิต ซึ่งหมายความว่า งานแต่ละงานจะสามารถเริ่มทำงานในขั้นตอนถัดไปได้ ก็ต่อเมื่อขั้นตอนก่อนหน้าเสร็จสิ้นแล้วเท่านั้น ดังนั้นค่าเวลาเริ่มต้นของงานในขั้นตอนที่ $j+1$ จะต้องมากกว่าหรือเท่ากับเวลาเสร็จสิ้นของงานในขั้นตอนที่ j

ข้อจำกัดนี้ช่วยให้ตารางการผลิตมีความสอดคล้องกับลักษณะการทำงานจริงของระบบ Flexible Job Shop ซึ่งงานแต่ละงานต้องผ่านกระบวนการผลิตตามลำดับ และไม่สามารถข้ามขั้นตอนการผลิตได้

3.3.2.2 ข้อจำกัดของเครื่องจักร

เครื่องจักรหนึ่งเครื่องสามารถประมวลผลงานได้ทีละงานเท่านั้น ดังนั้นจึงไม่สามารถมีงานสองงานทำงานบนเครื่องจักรเดียวกันในช่วงเวลาเดียวกันได้

3.3.2.3 ข้อจำกัดของการเลือกเครื่องจักร

ในแต่ละขั้นตอน งานสามารถเลือกใช้เครื่องจักรได้เพียงหนึ่งเครื่องเท่านั้น

$$\sum_{k \in M_j} x_{ijk} = 1 \quad (3)$$

3.3.2.4 ความสัมพันธ์ระหว่างเวลาเริ่มต้นและเวลาสิ้นสุด

เวลาเสร็จสิ้นของงานในแต่ละขั้นตอนสามารถคำนวณได้จากเวลาเริ่มต้นบวกกับเวลาในการผลิต

$$C_{ijk} = S_{ijk} + p_{ijk} \quad (4)$$

โดยที่

p_{ijk} คือ เวลาในการผลิตของงาน i ในขั้นตอน j บนเครื่องจักร k

3.4 การออกแบบ Genetic Algorithm

เนื่องจากปัญหาการจัดตารางการผลิตในระบบ Flexible flow shop (FFS) เป็นปัญหาที่มีความซับซ้อนสูงและจัดอยู่ในกลุ่มปัญหา NP-hard การใช้วิธีการค้นหาคำตอบทั้งหมดแบบ Brute Force จะมีความซับซ้อนในการคำนวณเพิ่มขึ้นอย่างรวดเร็วเมื่อจำนวนงานเพิ่มขึ้น ดังนั้นงานวิจัยนี้จึงเลือกใช้ Genetic Algorithm (GA) ซึ่งเป็นอัลกอริทึมในกลุ่ม Metaheuristic ที่สามารถค้นหาคำตอบที่มีคุณภาพดีภายในเวลาการคำนวณที่เหมาะสม

Genetic Algorithm เป็นอัลกอริทึมที่เลียนแบบกระบวนการวิวัฒนาการตามธรรมชาติ โดยอาศัยหลักการคัดเลือกตามความเหมาะสม (Survival of the Fittest) ผ่านกระบวนการหลัก ได้แก่ การคัดเลือก (Selection) การผสมพันธุ์ (Crossover) และการกลายพันธุ์ (Mutation) เพื่อสร้างประชากรรุ่นใหม่ที่มีคุณภาพดีขึ้นอย่างต่อเนื่อง

ในงานวิจัยนี้ Genetic Algorithm ถูกออกแบบเพื่อค้นหาลำดับการผลิตของงานที่ทำให้ค่า Makespan และ Average Flow Time ต่ำที่สุด โดยรายละเอียดขององค์ประกอบของอัลกอริทึมมีดังต่อไปนี้

3.4.1 <u>โครงสร้างโครโมโซม (Chromosome Representation)</u>

ในงานวิจัยนี้ โครโมโซมถูกออกแบบให้อยู่ในรูปของ ลำดับของงาน (Job Sequence) ซึ่งใช้แทนลำดับการเข้าสู่กระบวนการผลิตของงานแต่ละงานในระบบ ตัวอย่างเช่น J3 ,J1 ,J5 ,J2 ,J4 โดยแต่ละตำแหน่งในโครโมโซมเรียกว่า ยีน (Gene) ซึ่งแทนงานหนึ่งงานในระบบการผลิต ลำดับของยีนในโครโมโซมจะแสดงลำดับการจัดตารางการผลิตของงาน ในการกำหนดเครื่องจักรที่ใช้ในแต่ละขั้นตอน งานวิจัยนี้ใช้กฎ Fastest Available Machine Rule กล่าวคือ ระบบจะเลือกเครื่องจักรที่สามารถทำให้งานเสร็จสิ้นเร็วที่สุดจากเครื่องจักรที่พร้อมใช้งานในขณะนั้น

3.4.2 การสร้างประชากรเริ่มต้น (Initial Population)

ประชากรเริ่มต้นของอัลกอริทึมถูกสร้างขึ้นโดยการสุ่มลำดับของงานทั้งหมดในระบบ โดยในแต่ละโครโมโซมจะต้องประกอบด้วยงานครบทุกงานโดยไม่ซ้ำกัน ตัวอย่างเช่น J2 ,J4 ,J1 ,J5 ,J3 การสร้างประชากรเริ่มต้นแบบสุ่มช่วยเพิ่มความหลากหลายของคำตอบเริ่มต้น ซึ่งมีผลต่อประสิทธิภาพในการค้นหาคำตอบของอัลกอริทึม

3.4.3 ฟังก์ชัน Fitness (Fitness Function)

ฟังก์ชัน Fitness ใช้สำหรับประเมินคุณภาพของโครโมโซมแต่ละตัว โดยในงานวิจัยนี้กำหนดให้ค่า Fitness มีความสัมพันธ์กับค่า Makespan ของตารางการผลิต โดยสามารถเขียนได้ดังนี้

$$Fitness = \frac{1}{C_{max}} \quad (5)$$

โดยที่

C_{max} คือ ค่า Makespan ของตารางการผลิต

การกำหนดฟังก์ชัน Fitness ในรูปแบบนี้ทำให้โครโมโซมที่ให้ค่า Makespan ต่ำมีค่า Fitness สูงกว่า และมีโอกาสถูกเลือกเข้าสู่กระบวนการสร้างประชากรรุ่นถัดไปมากกว่า

3.4.4 กระบวนการคัดเลือก (Selection)

กระบวนการคัดเลือกใช้สำหรับเลือกโครโมโซมที่มีคุณภาพดีจากประชากรปัจจุบันเพื่อใช้เป็นพ่อแม่พันธุ์ในการสร้างประชากรรุ่นใหม่ ในงานวิจัยนี้ใช้วิธี **Tournament Selection** โดยสุ่มเลือกโครโมโซมจำนวนหนึ่งจากประชากร แล้วเลือกโครโมโซมที่มีค่า Fitness สูงที่สุดเป็นผู้ชนะของการแข่งขัน และถูกนำไปใช้ในการผสมพันธุ์ต่อไป วิธีนี้ช่วยเพิ่มโอกาสให้โครโมโซมที่มีคุณภาพดีถูกส่งต่อไปยังประชากรรุ่นถัดไป

3.4.5 กระบวนการผสมพันธุ์ (Crossover)

กระบวนการ Crossover ใช้สำหรับสร้างโครโมโซมลูกจากโครโมโซมพ่อแม่ โดยในงานวิจัยนี้ใช้วิธี **Order Crossover (OX)** ซึ่งเป็นวิธีที่เหมาะสมกับปัญหาการจัดลำดับ หลักการของ Order Crossover คือการเลือกช่วงหนึ่งของโครโมโซมจากพ่อแม่ตัวแรก แล้วเติมยีนที่เหลือจากพ่อแม่ตัวที่สองโดยรักษาลำดับของยีนเดิมไว้ เพื่อให้โครโมโซมลูกยังคงมีงานครบทุกงานโดยไม่ซ้ำกัน

3.4.6 กระบวนการกลายพันธุ์ (Mutation)

กระบวนการ Mutation มีวัตถุประสงค์เพื่อเพิ่มความหลากหลายของประชากร และลดโอกาสที่อัลกอริทึมจะติดอยู่กับคำตอบเฉพาะกลุ่ม (Local Optimum) ในงานวิจัยนี้ใช้วิธี **Swap Mutation** ซึ่งเป็นการสุ่มเลือกตำแหน่งของยีนสองตำแหน่งในโครโมโซม แล้วทำการสลับตำแหน่งของยีนทั้งสอง ตัวอย่างเช่น ก่อน Mutation J3 ,J1 ,J5 ,J2 ,J4

จากตัวอย่างโครโมโซมก่อนการเกิด Mutation อัลกอริทึมจะทำการสุ่มเลือกตำแหน่งของยีนจำนวน 2 ตำแหน่ง เพื่อทำการสลับตำแหน่งกัน โดยในตัวอย่างนี้ได้ทำการเลือกยีนตำแหน่งที่ 2 และตำแหน่งที่ 5 ซึ่งได้แก่ J1 และ J4 หลังจากทำการสลับตำแหน่งของยีนทั้งสอง จะได้โครโมโซมใหม่หลัง Mutation เป็น J3 ,J4 ,J5 ,J2 ,J1

กระบวนการดังกล่าวเรียกว่า Swap Mutation ซึ่งช่วยเพิ่มความหลากหลายของประชากร (Population Diversity) และลดโอกาสที่อัลกอริทึมจะติดอยู่กับคำตอบเฉพาะกลุ่ม (Local Optimum)

3.4.7 เงื่อนไขการหยุดการทำงาน (Termination Condition)

อัลกอริทึมจะทำการพัฒนาโครโมโซมรุ่นใหม่อย่างต่อเนื่องจนกว่าจะถึงเงื่อนไขการหยุดการทำงานที่กำหนดไว้ ในงานวิจัยนี้กำหนดเงื่อนไขดังต่อไปนี้

- จำนวนประชากร (Population Size) = 50
- จำนวนรอบการพัฒนา (Generations) = 200
- อัตราการผสมพันธุ์ (Crossover Rate) = 0.9
- อัตราการกลายพันธุ์ (Mutation Rate) = 0.1

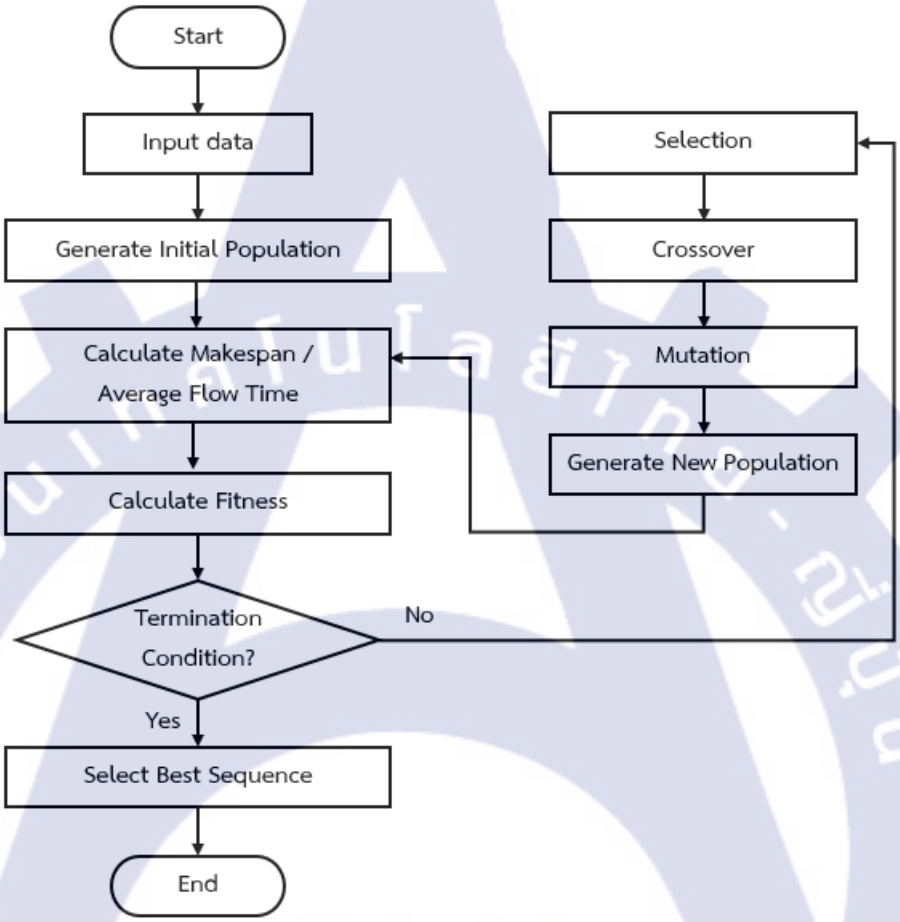
เมื่ออัลกอริทึมทำงานครบจำนวน Generation ที่กำหนด ระบบจะเลือกโครโมโซมที่มีค่า Fitness สูงที่สุดเป็นคำตอบที่ดีที่สุดของปัญหา

3.4.8 ขั้นตอนการทำงานของ Genetic Algorithm

ขั้นตอนการทำงานของอัลกอริทึมสามารถสรุปได้ดังนี้

1. สร้างประชากรเริ่มต้นแบบสุ่ม
2. คำนวณค่า Fitness ของโครโมโซมแต่ละตัว
3. เลือกโครโมโซมที่มีคุณภาพดีด้วยกระบวนการ Selection
4. สร้างโครโมโซมลูกด้วยกระบวนการ Crossover
5. ทำการ Mutation เพื่อเพิ่มความหลากหลายของประชากร
6. สร้างประชากรรุ่นใหม่และทำซ้ำกระบวนการ

7.หยุดการทำงานเมื่อครบจำนวน Generation ที่กำหนดโครโมโซมที่ให้ค่า Fitness สูงที่สุดจะถูกเลือกเป็นคำตอบที่ดีที่สุดของปัญหา



รูปที่ 3.3 Flow chart Genetic Algorithm

แผนภาพแสดงขั้นตอนการทำงานของอัลกอริทึมเชิงพันธุกรรม (Genetic Algorithm) สำหรับแก้ปัญหา Flexible Job Shop Scheduling Problem โดยเริ่มจากการสร้างประชากรเริ่มต้นแบบสุ่ม จากนั้นทำการถอดรหัสโครโมโซมเป็นตารางการผลิต คำนวณค่า Makespan และ Average Flow Time เพื่อประเมินค่า Fitness แล้วจึงดำเนินกระบวนการคัดเลือก การผสมพันธุ์ และการกลายพันธุ์เพื่อสร้างประชากรรุ่นใหม่ ทำซ้ำจนกว่าจะครบจำนวน Generation ที่กำหนด แล้วเลือกคำตอบที่ดีที่สุดเป็นผลลัพธ์สุดท้าย โดยในการถอดรหัสโครโมโซม จะมีการเลือกเครื่องจักร

ตามหลัก Fastest Available Machine Rule ซึ่งสอดคล้องกับลักษณะเครื่องจักรแบบ Non-uniform

3.5 การพัฒนาแบบจำลองด้วยภาษา Python

ในการวิจัยครั้งนี้ แบบจำลองการจัดตารางการผลิตสำหรับปัญหา Flexible Job Shop Scheduling Problem (FJSP) ได้ถูกพัฒนาขึ้นโดยใช้ภาษา Python เนื่องจากเป็นภาษาที่มีความยืดหยุ่นสูง เหมาะสมสำหรับการพัฒนาแบบจำลองเชิงคำนวณ การจำลองกระบวนการผลิต และการประยุกต์ใช้อัลกอริทึมเชิงเมตาฮิวริสติก เช่น Genetic Algorithm ในการพัฒนาแบบจำลองครั้งนี้ ผู้วิจัยใช้แพลตฟอร์ม Google Colab เป็นสภาพแวดล้อมสำหรับการเขียนและทดสอบโปรแกรม เนื่องจากสามารถรันโค้ด Python ได้โดยตรงผ่านเว็บเบราว์เซอร์ และเหมาะสมสำหรับการทดลองเชิงคำนวณโดยไม่จำเป็นต้องติดตั้งโปรแกรมเพิ่มเติมในเครื่องคอมพิวเตอร์ แบบจำลองที่พัฒนาขึ้นในภาษา Python ประกอบด้วยองค์ประกอบหลักดังต่อไปนี้

3.5.1 <u>การกำหนดข้อมูลของระบบการผลิต</u>

ในขั้นตอนแรก ผู้วิจัยได้กำหนดข้อมูลนำเข้าของระบบการผลิต ได้แก่ จำนวนงาน จำนวนขั้นตอนการผลิต จำนวนเครื่องจักรในแต่ละขั้นตอน และเวลาในการผลิตของแต่ละงานบนเครื่องจักรแต่ละเครื่อง โดยข้อมูลดังกล่าวถูกเก็บในรูปแบบโครงสร้างข้อมูลของ Python เช่น list และ dictionary เพื่อให้สามารถเข้าถึงและประมวลผลข้อมูลได้อย่างเป็นระบบ

สำหรับกรณีศึกษานี้ ระบบการผลิตประกอบด้วยงานจำนวน 5 งาน ขั้นตอนการผลิต 3 ขั้นตอน และในแต่ละขั้นตอนมีเครื่องจักรจำนวน 2 เครื่อง โดยเวลาการผลิตของเครื่องจักรถูกกำหนดในรูปแบบช่วงค่า (Range) กล่าวคือ เครื่องจักรกลุ่มที่หนึ่งมีเวลาในการผลิตอยู่ในช่วง 11–15 หน่วยเวลา และเครื่องจักรกลุ่มที่สองมีเวลาในการผลิตอยู่ในช่วง 5–10 หน่วยเวลา ซึ่งค่าดังกล่าวจะถูกสุ่มใหม่ในแต่ละรอบของการทดลองเพื่อสะท้อนลักษณะความไม่แน่นอนของระบบการผลิตจริง

3.5.2 <u>การพัฒนาฟังก์ชันคำนวณตารางการผลิต</u>

หลังจากกำหนดข้อมูลนำเข้าแล้ว ผู้วิจัยได้พัฒนาฟังก์ชันสำหรับคำนวณตารางการผลิต (Schedule) จากลำดับของงานที่กำหนด โดยโปรแกรมจะคำนวณเวลาเริ่มต้นและเวลาเสร็จสิ้นของงานแต่ละงานในแต่ละขั้นตอนการผลิต หลักการสำคัญที่ใช้ในการคำนวณเวลาเริ่มต้นของงาน คือ งานจะสามารถเริ่มทำงานในขั้นตอนใดขั้นตอนหนึ่งได้ ก็ต่อเมื่อ

1. งานนั้นเสร็จสิ้นจากขั้นตอนก่อนหน้าแล้ว
2. เครื่องจักรที่ถูกเลือกพร้อมใช้งานแล้ว

ดังนั้นเวลาเริ่มต้น (Start Time) ของงานจึงถูกกำหนดจากค่าสูงสุดระหว่างเวลาที่งานพร้อมและเวลาที่เครื่องจักรว่าง จากนั้นเวลาเสร็จสิ้นของงานจะคำนวณจาก เวลาเริ่มรวมกับเวลาการทำงานของเครื่องจักร กระบวนการดังกล่าวถูกใช้ซ้ำกับทุกงานและทุกขั้นตอนการผลิต จนกระทั่งได้ตารางการผลิตครบทั้งหมด

3.5.3 การเลือกเครื่องจักรด้วยกฎ Fastest Available Machine Rule

ในการจำลองการผลิต งานวิจัยนี้ใช้กฎ Fastest Available Machine Rule สำหรับการเลือกเครื่องจักรในแต่ละขั้นตอน กล่าวคือ โปรแกรมจะทำการพิจารณาเครื่องจักรทุกเครื่องที่สามารถใช้งานได้ในขั้นตอนนั้น แล้วเลือกเครื่องจักรที่ทำให้งานเสร็จสิ้นเร็วที่สุดจากสถานะความพร้อมใช้งานในขณะนั้น แนวทางดังกล่าวช่วยให้การจัดสรรเครื่องจักรมีประสิทธิภาพมากขึ้น และสะท้อนหลักการตัดสินใจที่เหมาะสมสำหรับระบบ Flexible Job Shop

3.5.4 การพัฒนาอัลกอริทึม Genetic Algorithm

เมื่อสามารถคำนวณตารางการผลิตและค่า Makespan จากลำดับงานที่กำหนดได้แล้ว ผู้วิจัยได้พัฒนาอัลกอริทึม Genetic Algorithm ในภาษา Python เพื่อใช้ค้นหาลำดับการผลิตที่เหมาะสมที่สุด อัลกอริทึมที่พัฒนาขึ้นประกอบด้วยองค์ประกอบหลัก ได้แก่

1. การสร้างประชากรเริ่มต้นแบบสุ่ม
2. การคำนวณค่า Fitness ของแต่ละโครโมโซม
3. การคัดเลือกโครโมโซมที่มีคุณภาพดี
4. การผสมพันธุ์ด้วยวิธี Order Crossover
5. การกลายพันธุ์ด้วยวิธี Swap Mutation
6. การสร้างประชากรรุ่นใหม่และทำซ้ำตามจำนวน Generation ที่กำหนด

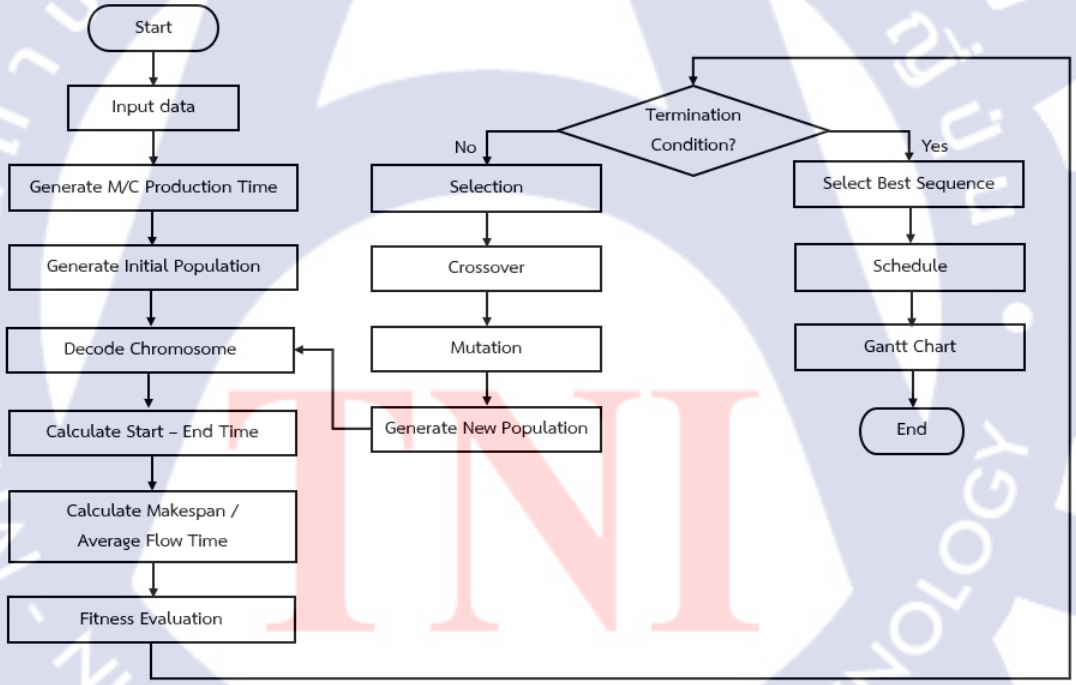
3.5.5 การแสดงผลลัพธ์

โปรแกรม Python ที่พัฒนาขึ้นสามารถแสดงผลลัพธ์ในหลายรูปแบบ ได้แก่ ลำดับการผลิตที่ดีที่สุด (Best Sequence) ค่า Makespan ค่า Average Flow Time ตารางเวลาเริ่มต้นและสิ้นสุดของงานในแต่ละขั้นตอน และแผนภาพ Gantt Chart สำหรับแสดงภาพรวมของตารางการผลิต การแสดงผลในรูปแบบดังกล่าวช่วยให้ผู้วิจัยสามารถวิเคราะห์ผลลัพธ์ของอัลกอริทึมได้อย่างชัดเจน และนำผลไปใช้ในบทวิเคราะห์ผลการทดลองได้อย่างเหมาะสม

3.5.6 การตรวจสอบความถูกต้องของแบบจำลอง

เพื่อให้มั่นใจว่าแบบจำลองที่พัฒนาขึ้นในภาษา Python มีความถูกต้อง ผู้วิจัยได้ตรวจสอบผลลัพธ์ของโปรแกรมโดยเปรียบเทียบตารางการผลิตที่ได้กับหลักการของระบบการผลิตจริง เช่น งานแต่ละงานต้องผ่านทุกขั้นตอนตามลำดับ เครื่องจักรหนึ่งเครื่องต้องไม่ทำงานซ้อนกัน เวลาเริ่มต้นและเวลาเสร็จสิ้นต้องสอดคล้องกับสมการของแบบจำลอง ค่า Makespan และ Average Flow Time ที่ได้จากโปรแกรมต้องคำนวณสอดคล้องกับนิยามเชิงทฤษฎี

นอกจากนี้ ผู้วิจัยยังใช้วิธี Brute Force สำหรับกรณีปัญหาขนาดเล็กเพื่อใช้เป็นค่าอ้างอิงในการตรวจสอบคุณภาพของคำตอบที่ได้จาก Genetic Algorithm การพัฒนาแบบจำลองด้วยภาษา Python ในงานวิจัยนี้ช่วยให้สามารถจำลองระบบการผลิต Flexible Job Shop ได้อย่างมีประสิทธิภาพ ทั้งในด้านการคำนวณตารางการผลิต การประยุกต์ใช้อัลกอริทึม Genetic Algorithm และการแสดงผลลัพธ์ในรูปแบบที่เหมาะสมสำหรับการวิเคราะห์เชิงวิชาการ ซึ่งถือเป็นขั้นตอนสำคัญในการเชื่อมโยงระหว่างแบบจำลองทางคณิตศาสตร์กับการทดลองเชิงคำนวณ



รูปที่ 3.4 Flow chart Python for GA

แผนภาพแสดงขั้นตอนการทำงานของโปรแกรม Python ที่พัฒนาขึ้นสำหรับแก้ปัญหา Flexible Job Shop Scheduling Problem โดยเริ่มจากการกำหนดข้อมูลนำเข้าของระบบการผลิตและพารามิเตอร์ของอัลกอริทึม จากนั้นโปรแกรมจะสุ่มค่าเวลาในการผลิต สร้างประชากรเริ่มต้น และถอดรหัสโครโมโซมเป็นตารางการผลิตเพื่อคำนวณค่า Makespan และ Average Flow Time แล้วนำค่าดังกล่าวไปใช้ในการประเมิน Fitness ของแต่ละโครโมโซม หลังจากนั้นโปรแกรมจะดำเนินกระบวนการ Selection, Crossover และ Mutation เพื่อสร้างประชากรรุ่นใหม่ และทำซ้ำจนกว่าจะครบจำนวน Generation ที่กำหนด แล้วจึงแสดงผลลัพธ์ที่ดีที่สุดในรูปของลำดับการผลิต ตารางเวลา และ Gantt Chart

3.6 การออกแบบการทดลอง

ในการวิจัยครั้งนี้ การออกแบบการทดลองมีวัตถุประสงค์เพื่อประเมินประสิทธิภาพของอัลกอริทึม Genetic Algorithm (GA) สำหรับการแก้ปัญหา Flexible Job Shop Scheduling Problem (FJSP) โดยพิจารณาความสามารถในการค้นหาลำดับการผลิตที่ให้ค่า Makespan และ Average Flow Time ต่ำที่สุด รวมถึงเปรียบเทียบผลลัพธ์กับวิธี Brute Force ในกรณีของปัญหาขนาดเล็ก

การทดลองถูกออกแบบให้สอดคล้องกับลักษณะของระบบการผลิตที่ศึกษา และครอบคลุมทั้งการกำหนดข้อมูลนำเข้า การตั้งค่าพารามิเตอร์ของอัลกอริทึม การสุ่มข้อมูลเวลาในการผลิต และการประเมินผลลัพธ์จากการทดลองหลายรอบ เพื่อให้ได้ผลการทดลองที่มีความน่าเชื่อถือ

3.6.1 <u>ข้อมูลทดลองของระบบการผลิต</u>

กรณีศึกษาที่ใช้ในการทดลองเป็นระบบการผลิตแบบ Flexible Job Shop ซึ่งประกอบด้วยงานจำนวน 6 งาน (Jobs) ขั้นตอนการผลิตจำนวน 3 ขั้นตอน (Stages) ในแต่ละขั้นตอนมีเครื่องจักรจำนวน 2 เครื่องจักร โดยกำหนดให้เครื่องจักรในแต่ละขั้นตอนมีประสิทธิภาพแตกต่างกัน ดังนี้

1. เครื่องจักรกลุ่มที่หนึ่ง (M1_1, M2_1, M3_1) มีเวลาในการผลิตอยู่ในช่วง 11–15 หน่วยเวลา

2.เครื่องจักรกลุ่มที่สอง (M1_2, M2_2, M3_2) มีเวลาในการผลิตอยู่ในช่วง 5–10 หน่วยเวลา

เวลาในการผลิตดังกล่าวถูกกำหนดแบบสุ่มในแต่ละรอบของการทดลอง เพื่อสะท้อนลักษณะความไม่แน่นอนของระบบการผลิตจริง นอกจากกรณีศึกษาขนาดเล็กที่ใช้จำนวนงาน 6 งาน เพื่อเปรียบเทียบกับวิธี Brute Force อย่างครบถ้วนแล้ว งานวิจัยนี้ยังได้ขยายการทดลองไปยังกรณีปัญหาขนาดใหญ่ขึ้น โดยกำหนดให้มีจำนวนงานเท่ากับ 20 งาน (20 Jobs) เพื่อศึกษาประสิทธิภาพของอัลกอริทึมในสภาวะที่มีความซับซ้อนสูงมากขึ้น

ในการทดลองดังกล่าว ได้กำหนดเงื่อนไขเวลาในการประมวลผลเท่ากับ 2 นาทีต่อการรันหนึ่งครั้ง สำหรับทั้งวิธี Genetic Algorithm (GA) และวิธี Brute Force เพื่อให้สามารถเปรียบเทียบประสิทธิภาพของทั้งสองวิธีได้อย่างเป็นธรรมภายใต้ข้อจำกัดของเวลาเดียวกัน

จากลักษณะของปัญหา เมื่อจำนวนงานเพิ่มขึ้นเป็น 20 งาน จำนวนลำดับการผลิตที่เป็นไปได้จะมีค่าเท่ากับ 20! ซึ่งมีขนาดใหญ่มากจนไม่สามารถคำนวณได้ครบถ้วนภายในเวลาที่กำหนดด้วยวิธี Brute Force ดังนั้น วิธี Brute Force ในกรณีนี้จะสามารถค้นหาคำตอบได้เพียงบางส่วนของพื้นที่คำตอบ (Search Space) เท่านั้น ส่งผลให้คำตอบที่ได้อาจไม่ใช่คำตอบที่ดีที่สุด

ในขณะที่ Genetic Algorithm เป็นวิธีการค้นหาแบบ Metaheuristic ที่สามารถสำรวจพื้นที่คำตอบได้อย่างมีประสิทธิภาพภายในเวลาที่จำกัด โดยอาศัยกระบวนการคัดเลือก การผสมพันธุ์ และการกลายพันธุ์ ทำให้สามารถค้นหาคำตอบที่มีคุณภาพดีได้แม้ในกรณีที่ไม่สามารถตรวจสอบทุกลำดับการผลิตได้ครบถ้วน

3.6.2 <u>การกำหนดค่าพารามิเตอร์ของ Genetic Algorithm</u>

ค่าพารามิเตอร์ของอัลกอริทึม Genetic Algorithm ถูกกำหนดขึ้นโดยพิจารณาจากแนวทางที่นิยมใช้ในงานวิจัยด้าน Metaheuristic ร่วมกับการทดลองเบื้องต้น (Preliminary Experiment) เพื่อให้เกิดความสมดุลระหว่างคุณภาพของคำตอบและเวลาในการคำนวณ โดยรายละเอียดของแต่ละพารามิเตอร์มีดังนี้

1. Population Size = 50 ขนาดประชากรถูกกำหนดให้มีค่า 50 เพื่อให้มีความหลากหลายของคำตอบ (Solution Diversity) เพียงพอในการค้นหาพื้นที่คำตอบ (Search Space) โดยหากกำหนดค่าน้อยเกินไป อาจทำให้อัลกอริทึมติดอยู่ที่คำตอบเฉพาะกลุ่ม (Local Optimum) ในขณะที่หากกำหนดค่าสูงเกินไปจะส่งผลให้ใช้เวลาในการคำนวณเพิ่มขึ้นอย่างไม่จำเป็น ดังนั้นค่า 50 จึงเป็นค่าที่เหมาะสมสำหรับปัญหาขนาดกลางถึงขนาดใหญ่ เช่น 20 Jobs

2. Number of Generations = 200 จำนวนรอบการพัฒนาถูกกำหนดที่ 200 รุ่น เนื่องจากจากการทดลองเบื้องต้นพบว่าค่า Makespan มีแนวโน้มคงที่ (Convergence) หลังจากประมาณ 150–200 Generations การเพิ่มจำนวน Generations มากกว่านี้ไม่ส่งผลให้คุณภาพคำตอบดีขึ้นอย่างมีนัยสำคัญ แต่จะเพิ่มเวลาในการคำนวณ ดังนั้นจึงเลือกใช้ค่า 200 เพื่อให้ได้คำตอบที่มีเสถียรภาพภายในเวลาที่เหมาะสม

3. Crossover Rate = 0.9 อัตราการผสมพันธุ์ถูกกำหนดให้มีค่าสูง (0.9) เพื่อเพิ่มโอกาสในการแลกเปลี่ยนข้อมูลระหว่างโครโมโซม ซึ่งช่วยให้เกิดการสร้างคำตอบใหม่ที่มีคุณภาพดี โดยค่าในช่วง 0.8–0.95 เป็นช่วงที่นิยมใช้ในงานวิจัยด้าน Genetic Algorithm

4. Mutation Rate = 0.1 อัตราการกลายพันธุ์ถูกกำหนดที่ 0.1 เพื่อรักษาสมดุลระหว่างการสำรวจพื้นที่คำตอบใหม่ (Exploration) และการใช้ประโยชน์จากคำตอบเดิม (Exploitation) โดยค่าที่ต่ำเกินไปอาจทำให้เกิดการกระจุกตัวของคำตอบ ส่วนค่าที่สูงเกินไปจะทำให้อัลกอริทึมมีลักษณะคล้ายการสุ่มมากเกินไป

5. Selection Method = Tournament Selection เลือกใช้วิธี Tournament Selection เนื่องจากเป็นวิธีที่มีความง่ายในการคำนวณ และสามารถรักษาสมดุลระหว่างการคัดเลือกคำตอบที่ดีและการคงความหลากหลายของประชากรได้ดี อีกทั้งยังเหมาะสมกับปัญหาการจัดลำดับ (Scheduling Problem)

6. Crossover Method = Order Crossover (OX) เลือกใช้ Order Crossover (OX) เนื่องจากเป็นวิธีที่เหมาะสมกับปัญหาการจัดลำดับงาน (Permutation-based Problem) โดยสามารถรักษาลำดับของงานและป้องกันการเกิดค่าซ้ำในโครโมโซม ทำให้ได้คำตอบที่ถูกต้องตามเงื่อนไขของปัญหา

7. Mutation Method = Swap Mutation เลือกใช้ Swap Mutation ซึ่งเป็นการสลับตำแหน่งของงานสองตำแหน่ง เนื่องจากเป็นวิธีที่เรียบง่ายและมีประสิทธิภาพในการเพิ่มความหลากหลายของคำตอบ โดยไม่ทำลายโครงสร้างของโครโมโซม

8. Number of Runs = 10 กำหนดจำนวนรอบการทดลองเท่ากับ 10 ครั้ง เพื่อประเมินความเสถียรของอัลกอริทึม เนื่องจาก Genetic Algorithm เป็นวิธีแบบสุ่ม (Stochastic) ผลลัพธ์ในแต่ละรอบอาจแตกต่างกัน การทดลองหลายรอบช่วยให้สามารถวิเคราะห์ค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานได้อย่างน่าเชื่อถือ

3.6.3 ขั้นตอนการทดลอง

การทดลองในงานวิจัยนี้ดำเนินการตามลำดับขั้นตอนดังต่อไปนี้

1. กำหนดข้อมูลนำเข้าของระบบการผลิต ได้แก่ จำนวนงาน จำนวนขั้นตอนการผลิต จำนวนเครื่องจักร และช่วงเวลาในการผลิตของเครื่องจักรแต่ละเครื่อง

2. สุ่มค่าเวลาในการผลิตของเครื่องจักรในแต่ละรอบของการทดลองตามช่วงค่าที่กำหนด

3. สร้างประชากรเริ่มต้นของอัลกอริทึม Genetic Algorithm แบบสุ่ม

4. ดำเนินกระบวนการของ Genetic Algorithm ประกอบด้วยการประเมินค่า Fitness การคัดเลือก การผสมพันธุ์ และการกลายพันธุ์

5. ทำซ้ำกระบวนการจนกว่าจะครบจำนวน Generation ที่กำหนด

6. บันทึกลำดับการผลิตที่ดีที่สุด ค่า Makespan และค่า Average Flow Time ที่ได้จากแต่ละรอบการทดลอง

7. ทำการทดลองซ้ำจำนวน 10 ครั้ง เพื่อประเมินเสถียรภาพของอัลกอริทึม

8. เปรียบเทียบผลลัพธ์ของ Genetic Algorithm กับผลลัพธ์จากวิธี Brute Force ในกรณีปัญหาขนาดเล็ก

3.6.4 <u>การทดลองซ้ำหลายรอบ</u>

เนื่องจากอัลกอริทึม Genetic Algorithm เป็นอัลกอริทึมที่มีลักษณะการค้นหาแบบสุ่ม (stochastic search) ผลลัพธ์ที่ได้จากการทดลองในแต่ละครั้งอาจแตกต่างกัน ดังนั้นงานวิจัยนี้จึงกำหนดให้มีการทดลองซ้ำจำนวน 10 รอบ โดยในแต่ละรอบจะมีการสุ่มค่า Processing Time ใหม่ เพื่อให้สามารถประเมินเสถียรภาพของอัลกอริทึมได้อย่างเหมาะสม

ผลการทดลองจากทั้ง 10 รอบจะถูกนำมาวิเคราะห์โดยใช้ค่าทางสถิติ ได้แก่

1. ค่าเฉลี่ย (Mean)
2. ส่วนเบี่ยงเบนมาตรฐาน (Standard Deviation)

3.6.5 <u>การเปรียบเทียบกับวิธี Brute Force</u>

เพื่อประเมินประสิทธิภาพของ Genetic Algorithm อย่างเป็นระบบ งานวิจัยนี้ได้ใช้วิธี Brute Force เป็นค่าอ้างอิงสำหรับเปรียบเทียบผลลัพธ์ โดยในกรณีศึกษาที่มีงานจำนวน 5 งาน สามารถสร้างลำดับการผลิตที่เป็นไปได้ทั้งหมดจำนวน 6! = 720 ลำดับ และคำนวณค่าประสิทธิภาพของแต่ละลำดับได้ครบถ้วน ผลลัพธ์จาก Brute Force จะใช้เป็นคำตอบที่ดีที่สุดเชิงทฤษฎี (Optimal Solution) เพื่อใช้เปรียบเทียบกับผลลัพธ์ที่ได้จาก Genetic Algorithm ว่าอัลกอริทึมสามารถค้นหาคำตอบที่ใกล้เคียงหรือเท่ากับคำตอบที่ดีที่สุดได้มากน้อยเพียงใด

3.6.6 <u>ตัวชี้วัดสำหรับการประเมินผล</u>

ตัวชี้วัดที่ใช้ในการประเมินผลการทดลองในงานวิจัยนี้ประกอบด้วย

1. Makespan ใช้วัดเวลาที่งานสุดท้ายในระบบเสร็จสิ้น โดยเป็นตัวชี้วัดหลักของการจัดตารางการผลิต

$$C_{max} = \max(C_i) \quad (6)$$

โดยที่

C_i คือ เวลาเสร็จสิ้นของงานที่ i

2. Average Flow Time ใช้วัดค่าเฉลี่ยของเวลาที่งานแต่ละงานใช้ในระบบการผลิต

$$Average\ Flow\ Time = \frac{1}{n}\sum_{i=1}^{n} C_i \quad (7)$$

โดยที่

n คือ จำนวนงานทั้งหมดในระบบ

C_i คือ เวลาเสร็จสิ้นของงานที่ i

3. ค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐาน

การออกแบบการทดลองในงานวิจัยนี้ถูกกำหนดขึ้นเพื่อให้สามารถประเมินประสิทธิภาพของอัลกอริทึม Genetic Algorithm ได้อย่างชัดเจนและน่าเชื่อถือ โดยการกำหนดข้อมูลนำเข้า การตั้งค่าพารามิเตอร์ของอัลกอริทึม การทดลองซ้ำหลายรอบ และการเปรียบเทียบผลลัพธ์กับวิธี Brute Force ซึ่งช่วยให้ผลการทดลองที่ได้มีความเหมาะสมสำหรับการวิเคราะห์และอภิปรายผล ใช้ประเมินเสถียรภาพของอัลกอริทึมเมื่อทำการทดลองหลายรอบ

3.7 วิธีการวิเคราะห์ผลการทดลอง

หลังจากดำเนินการทดลองด้วยอัลกอริทึม Genetic Algorithm และวิธี Brute Force แล้ว ผู้วิจัยได้ทำการวิเคราะห์ผลลัพธ์ที่ได้จากการทดลอง เพื่อประเมินประสิทธิภาพของอัลกอริทึมในการแก้ปัญหา Flexible Job Shop Scheduling Problem โดยพิจารณาจากตัวชี้วัดด้านประสิทธิภาพของตารางการผลิต และการเปรียบเทียบผลลัพธ์ระหว่างวิธีการต่าง ๆ การวิเคราะห์ผลการทดลองในงานวิจัยนี้ประกอบด้วยขั้นตอนดังต่อไปนี้

3.7.1 <u>การวิเคราะห์ค่า Makespan</u>

ค่า Makespan เป็นตัวชี้วัดหลักที่ใช้ประเมินคุณภาพของตารางการผลิต โดยหมายถึงเวลาที่งานสุดท้ายในระบบการผลิตเสร็จสิ้นทั้งหมด ค่า Makespan ที่ต่ำกว่าจะสะท้อนถึงประสิทธิภาพของตารางการผลิตที่ดีกว่า เนื่องจากสามารถทำให้งานทั้งหมดในระบบเสร็จสิ้นได้ในเวลาที่สั้นลง ดังนั้นในการวิเคราะห์ผลการทดลอง ผู้วิจัยจะเปรียบเทียบค่า Makespan ที่ได้จากอัลกอริทึม Genetic Algorithm กับค่าที่ได้จากวิธี Brute Force

3.7.2 การวิเคราะห์ค่า Average Flow Time

Average Flow Time ใช้สำหรับวัดระยะเวลาที่งานแต่ละงานใช้ในระบบการผลิตโดยเฉลี่ย ซึ่งสามารถคำนวณได้จากสมการ ค่าของ Average Flow Time ที่ต่ำแสดงให้เห็นว่าระบบสามารถประมวลผลงานได้อย่างรวดเร็ว และลดระยะเวลาที่งานต้องรออยู่ในระบบการผลิต

3.7.3 การเปรียบเทียบผลลัพธ์กับวิธี Brute Force

เพื่อประเมินประสิทธิภาพของ Genetic Algorithm อย่างเป็นระบบ ผู้วิจัยได้ใช้วิธี Brute Force เป็นค่ามาตรฐานสำหรับการเปรียบเทียบ โดยวิธี Brute Force จะทำการคำนวณลำดับการผลิตที่เป็นไปได้ทั้งหมด และเลือกคำตอบที่ให้ค่า Makespan ต่ำที่สุด

สำหรับกรณีศึกษาที่มีจำนวนงาน 5 งาน สามารถสร้างลำดับการผลิตทั้งหมดได้จำนวน 120 ลำดับ จากนั้นผู้วิจัยจะเปรียบเทียบค่า Makespan ที่ได้จาก Genetic Algorithm กับค่าที่ได้จาก Brute Force เพื่อประเมินว่าอัลกอริทึมสามารถค้นหาคำตอบที่ใกล้เคียงกับคำตอบที่ดีที่สุดได้มากน้อยเพียงใด

3.7.4 การวิเคราะห์ผลจากการทดลองหลายรอบ

เนื่องจากอัลกอริทึม Genetic Algorithm มีลักษณะการค้นหาแบบสุ่ม ผลลัพธ์ที่ได้ในแต่ละครั้งของการทดลองอาจแตกต่างกัน ดังนั้นในการวิจัยนี้จึงทำการทดลองซ้ำจำนวน 10 รอบ เพื่อใช้ประเมินความสม่ำเสมอของผลลัพธ์ที่ได้จากอัลกอริทึม

3.7.5 การแสดงผลลัพธ์ของตารางการผลิต

ผลลัพธ์ของการทดลองจะถูกนำเสนอในรูปแบบต่าง ๆ เพื่อให้สามารถวิเคราะห์ได้อย่างชัดเจน ได้แก่ ตารางสรุปผลการทดลองจากแต่ละรอบของ Genetic Algorithm ตารางเปรียบเทียบผลลัพธ์ระหว่าง Genetic Algorithm และ Brute Force ตารางแสดงค่า Makespan และ Average Flow Time และแผนภาพ Gantt Chart เพื่อแสดงลำดับการผลิตของงานในแต่ละเครื่องจักร

การนำเสนอผลลัพธ์ในรูปแบบดังกล่าวช่วยให้สามารถเห็นภาพรวมของตารางการผลิต และวิเคราะห์ประสิทธิภาพของอัลกอริทึมได้อย่างชัดเจน

3.7.6 การอภิปรายผล

หลังจากวิเคราะห์ผลลัพธ์จากการทดลองแล้ว ผู้วิจัยจะนำผลลัพธ์ที่ได้ไปอภิปรายในบทถัดไป โดยพิจารณาประเด็นสำคัญ เช่น ความสามารถของ Genetic Algorithm ในการค้นหาคำตอบที่มีค่า Makespan ต่ำ ความใกล้เคียงของคำตอบที่ได้จาก Genetic Algorithm เมื่อเทียบกับคำตอบ

จาก Brute Force ความสม่ำเสมอของผลลัพธ์จากการทดลองหลายรอบ และข้อดีและข้อจำกัดของวิธีการที่ใช้

3.8 การทดลองกับปัญหาขนาดใหญ่

เพื่อประเมินประสิทธิภาพของอัลกอริทึม Genetic Algorithm ในการแก้ปัญหาการจัดตารางการผลิตในกรณีที่มีความซับซ้อนสูง งานวิจัยนี้ได้ทำการทดลองเพิ่มเติมกับปัญหาขนาดใหญ่ โดยกำหนดให้มีจำนวนงานเท่ากับ 20 งาน ภายใต้โครงสร้างระบบการผลิตแบบ Flexible Job Shop เช่นเดียวกับกรณีศึกษาหลัก

เนื่องจากเมื่อจำนวนงานเพิ่มขึ้นเป็น 20 งาน จำนวนลำดับการผลิตที่เป็นไปได้จะมีค่าเท่ากับ 20! ซึ่งมีขนาดใหญ่มาก ทำให้ไม่สามารถใช้วิธี Brute Force ในการค้นหาคำตอบที่เหมาะสมที่สุดได้ภายในเวลาที่เหมาะสม ในการทดลองนี้จึงกำหนดเงื่อนไขเวลาในการประมวลผล เท่ากับ 2 นาที ต่อการรันหนึ่งครั้ง สำหรับทั้งวิธี Genetic Algorithm และวิธี Brute Force เพื่อใช้เป็นเกณฑ์ในการเปรียบเทียบประสิทธิภาพของทั้งสองวิธีภายใต้ข้อจำกัดเดียวกัน

สำหรับวิธี Genetic Algorithm อัลกอริทึมจะทำการค้นหาคำตอบภายในระยะเวลา 2 นาที โดยดำเนินกระบวนการตามขั้นตอนของ GA ได้แก่ การสร้างประชากรเริ่มต้น การคัดเลือก การผสมพันธุ์ และการกลายพันธุ์ จนกว่าจะครบเงื่อนไขเวลาที่กำหนด และเลือกคำตอบที่ดีที่สุดที่พบภายในช่วงเวลาดังกล่าว และวิธี Brute Force จะทำการสร้างและประเมินลำดับการผลิตเท่าที่สามารถทำได้ภายในเวลา 2 นาที ซึ่งเนื่องจากข้อจำกัดด้านเวลา วิธีนี้จะไม่สามารถตรวจสอบทุกลำดับที่เป็นไปได้ ทำให้คำตอบที่ได้อาจยังไม่ใช่คำตอบที่ดีที่สุด

ผลลัพธ์จากการทดลองจะถูกนำมาเปรียบเทียบในด้านค่า Makespan และ Average Flow Time เพื่อประเมินความสามารถของ Genetic Algorithm ในการค้นหาคำตอบที่มีคุณภาพภายใต้ข้อจำกัดของเวลา และแสดงให้เห็นถึงความเหมาะสมของอัลกอริทึมสำหรับการประยุกต์ใช้กับปัญหาการจัดตารางการผลิตขนาดใหญ่

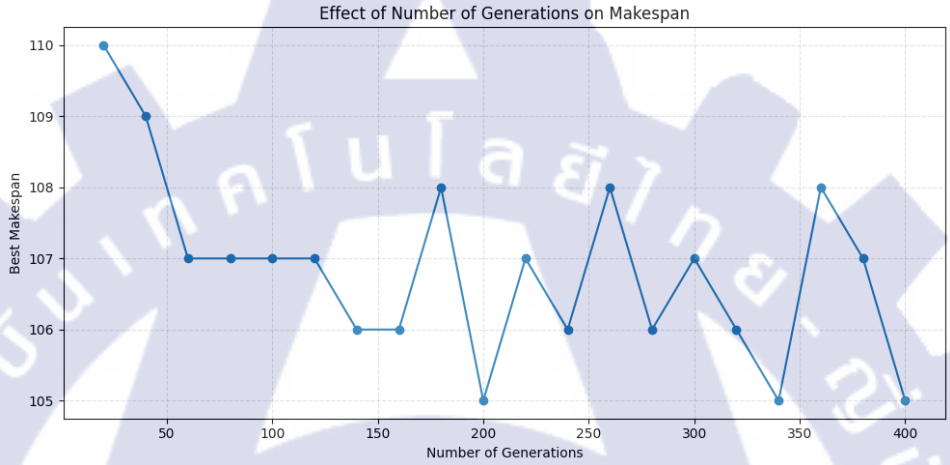
ตารางที่ 3.7 ตัวอย่างข้อมูลเวลาในการผลิตของกรณีศึกษา 20 งาน

Job	S1-M1_1	S1-M1_2	S2-M2_1	S2-M2_2	S3-M3_1	S3-M3_2
J1	11	5	13	6	12	6
J2	11	10	15	5	15	8
J3	11	5	11	6	12	9
J4	15	5	15	6	15	8
J5	12	8	15	7	11	6
J6	14	7	13	6	12	7
J7	11	5	14	5	13	7
J8	15	7	11	10	14	9
J9	11	8	11	9	13	10
J10	15	7	15	6	11	5
J11	12	7	11	6	11	8
J12	13	8	13	6	13	7
J13	12	10	13	10	11	9
J14	12	9	12	6	14	8
J15	13	10	15	6	13	5
J16	12	5	13	8	13	5
J17	12	9	13	6	14	8
J18	14	6	13	6	12	10
J19	15	9	13	10	18	8
J20	15	8	13	6	12	9

3.9 ผลวิเคราะห์การเลือก Generation 200

เพื่อกำหนดจำนวนรอบการพัฒนา (Number of Generations) ที่เหมาะสมสำหรับอัลกอริทึม Genetic Algorithm งานวิจัยนี้ได้ทำการทดลองโดยปรับจำนวน Generations ในช่วงตั้งแต่ 20 ถึง 400 และบันทึกค่าที่ดีที่สุดของ Makespan ในแต่ละช่วง ดังแสดงในรูปที่ 3.9.1 จากกราฟจะเห็นได้ว่า ในช่วงเริ่มต้นของการทำงาน (20–60 Generations) ค่า Makespan มีแนวโน้มลดลงอย่างรวดเร็ว จากประมาณ 110 หน่วยเวลา ลดลงมาอยู่ที่ประมาณ 107 หน่วยเวลา ซึ่งแสดงให้เห็นว่าอัลกอริทึมสามารถค้นหาคำตอบที่ดีขึ้นได้อย่างมีประสิทธิภาพในช่วงต้น หลังจากนั้น ในช่วง

ประมาณ 60–140 Generations ค่า Makespan มีแนวโน้มคงที่อยู่ที่ประมาณ 106–107 หน่วยเวลา แสดงให้เห็นว่าอัลกอริทึมเริ่มเข้าสู่สภาวะที่การปรับปรุงคำตอบทำได้ลดลง เมื่อพิจารณาช่วงตั้งแต่ 140 Generations เป็นต้นไป พบว่าค่า Makespan มีการแกว่งตัวเล็กน้อยในช่วงประมาณ 105–108 หน่วยเวลา โดยไม่มีแนวโน้มลดลงอย่างต่อเนื่อง ซึ่งแสดงว่าอัลกอริทึมได้เข้าสู่ภาวะลู่เข้า (Convergence) แล้ว และการเพิ่มจำนวน Generations ไม่ได้ช่วยให้ได้คำตอบที่ดีขึ้นอย่างมีนัยสำคัญ



รูปที่ 3.5 ความสัมพันธ์ระหว่างจำนวน Generations กับค่า Makespan

แม้ว่าจะมีบางช่วงที่ค่า Makespan ลดลงต่ำสุดที่ประมาณ 105 หน่วยเวลา เช่น ที่ 200 และ 400 Generations แต่ค่าดังกล่าวไม่ได้แตกต่างจากช่วงก่อนหน้าอย่างชัดเจน และเกิดขึ้นในลักษณะของการแกว่งตัว (Fluctuation) มากกว่าการปรับปรุงอย่างต่อเนื่อง ดังนั้น การกำหนดจำนวน Generations เท่ากับ 200 จึงถือเป็นค่าที่เหมาะสม เนื่องจากสามารถให้คำตอบที่มีคุณภาพดีและมีเสถียรภาพ โดยไม่จำเป็นต้องเพิ่มเวลาในการคำนวณเกินความจำเป็น ซึ่งสอดคล้องกับหลักการของการออกแบบอัลกอริทึมเชิงเมตาฮิวริสติกที่มุ่งเน้นความสมดุลระหว่างคุณภาพของคำตอบและประสิทธิภาพในการคำนวณ

บทที่ 4
ผลการทดลองและการวิเคราะห์ผล

4.1 ตารางเวลาของเครื่องจักรที่ใช้ในการคำนวณ

ผลการทดลองของการแก้ปัญหาการจัดตารางการผลิตในระบบ Flexible Flow Shop โดยมีวัตถุประสงค์เพื่อลดค่า Makespan และ Average Flow Time ของระบบการผลิต โดยใช้ Genetic Algorithm (GA) และเปรียบเทียบผลลัพธ์กับวิธี Brute Force ซึ่งทำการค้นหาคำตอบจากทุกความเป็นไปได้ การทดลองดำเนินการโดยใช้ชุดข้อมูล Processing Time จำนวน 10 ชุด เพื่อประเมินประสิทธิภาพของวิธีการทั้งสองในหลายกรณีได้

ตาราง 4.1 ข้อมูลเวลาในการผลิตของเครื่องจักร สุ่มครั้งที่ 1

Job	S1-M1_1	S1-M1_2	S2-M2_1	S2-M2_2	S3-M3_1	S3-M3_2
J1	13	9	13	9	15	7
J2	15	5	11	5	14	9
J3	12	6	13	5	11	9
J4	15	7	15	5	15	10
J5	13	8	11	7	15	7
J6	15	8	15	8	11	6

ตาราง 4.2 ข้อมูลเวลาในการผลิตของเครื่องจักร สุ่มครั้งที่ 2

Job	S1-M1_1	S1-M1_2	S2-M2_1	S2-M2_2	S3-M3_1	S3-M3_2
J1	11	6	12	10	13	6
J2	14	5	11	8	15	6
J3	13	10	12	6	14	9
J4	14	10	14	5	12	5
J5	15	8	12	8	12	8
J6	12	8	13	7	11	10

ตาราง 4.3 ข้อมูลเวลาในการผลิตของเครื่องจักร สุ่มครั้งที่ 3

Job	S1-M1_1	S1-M1_2	S2-M2_1	S2-M2_2	S3-M3_1	S3-M3_2
J1	11	5	14	8	14	10
J2	12	8	11	9	13	9
J3	11	5	12	8	15	7
J4	15	5	15	9	14	6
J5	13	8	11	10	11	10
J6	13	7	15	10	13	5

ตาราง 4.4 ข้อมูลเวลาในการผลิตของเครื่องจักร สุ่มครั้งที่ 4

Job	S1-M1_1	S1-M1_2	S2-M2_1	S2-M2_2	S3-M3_1	S3-M3_2
J1	11	5	13	9	14	7
J2	14	10	12	5	12	7
J3	14	10	12	7	13	6
J4	13	10	11	9	13	10
J5	11	7	14	8	14	6
J6	13	8	14	5	14	7

ตาราง 4.5 ข้อมูลเวลาในการผลิตของเครื่องจักร สุ่มครั้งที่ 5

Job	S1-M1_1	S1-M1_2	S2-M2_1	S2-M2_2	S3-M3_1	S3-M3_2
J1	12	10	14	5	12	9
J2	14	6	14	10	11	7
J3	15	7	12	6	15	7
J4	15	9	11	9	14	8
J5	12	8	14	6	15	5
J6	12	10	15	7	13	6

ตาราง 4.6 ข้อมูลเวลาในการผลิตของเครื่องจักร สุ่มครั้งที่ 6

Job	S1-M1_1	S1-M1_2	S2-M2_1	S2-M2_2	S3-M3_1	S3-M3_2
J1	13	9	11	8	12	6
J2	13	6	11	5	13	5
J3	13	10	13	9	12	8
J4	14	9	13	8	13	10
J5	14	10	14	10	11	8
J6	11	8	15	5	15	5

ตาราง 4.7 ข้อมูลเวลาในการผลิตของเครื่องจักร สุ่มครั้งที่ 7

Job	S1-M1_1	S1-M1_2	S2-M2_1	S2-M2_2	S3-M3_1	S3-M3_2
J1	14	6	12	10	14	6
J2	14	9	15	10	11	9
J3	14	10	14	7	13	7
J4	13	9	11	10	11	5
J5	14	7	15	9	14	5
J6	12	5	14	9	15	8

ตาราง 4.8 ข้อมูลเวลาในการผลิตของเครื่องจักร สุ่มครั้งที่ 8

Job	S1-M1_1	S1-M1_2	S2-M2_1	S2-M2_2	S3-M3_1	S3-M3_2
J1	11	6	12	5	11	10
J2	12	6	14	5	14	8
J3	11	6	14	6	14	7
J4	14	8	11	8	13	6
J5	13	9	15	10	13	10
J6	15	6	14	6	14	5

ตาราง 4.9 ข้อมูลเวลาในการผลิตของเครื่องจักร สุ่มครั้งที่ 9

Job	S1-M1_1	S1-M1_2	S2-M2_1	S2-M2_2	S3-M3_1	S3-M3_2
J1	13	7	11	9	15	8
J2	12	6	11	7	13	9
J3	15	8	11	10	15	8
J4	12	6	13	5	14	7
J5	13	6	15	5	13	10
J6	12	10	14	10	13	5

ตาราง 4.10 ข้อมูลเวลาในการผลิตของเครื่องจักร สุ่มครั้งที่ 10

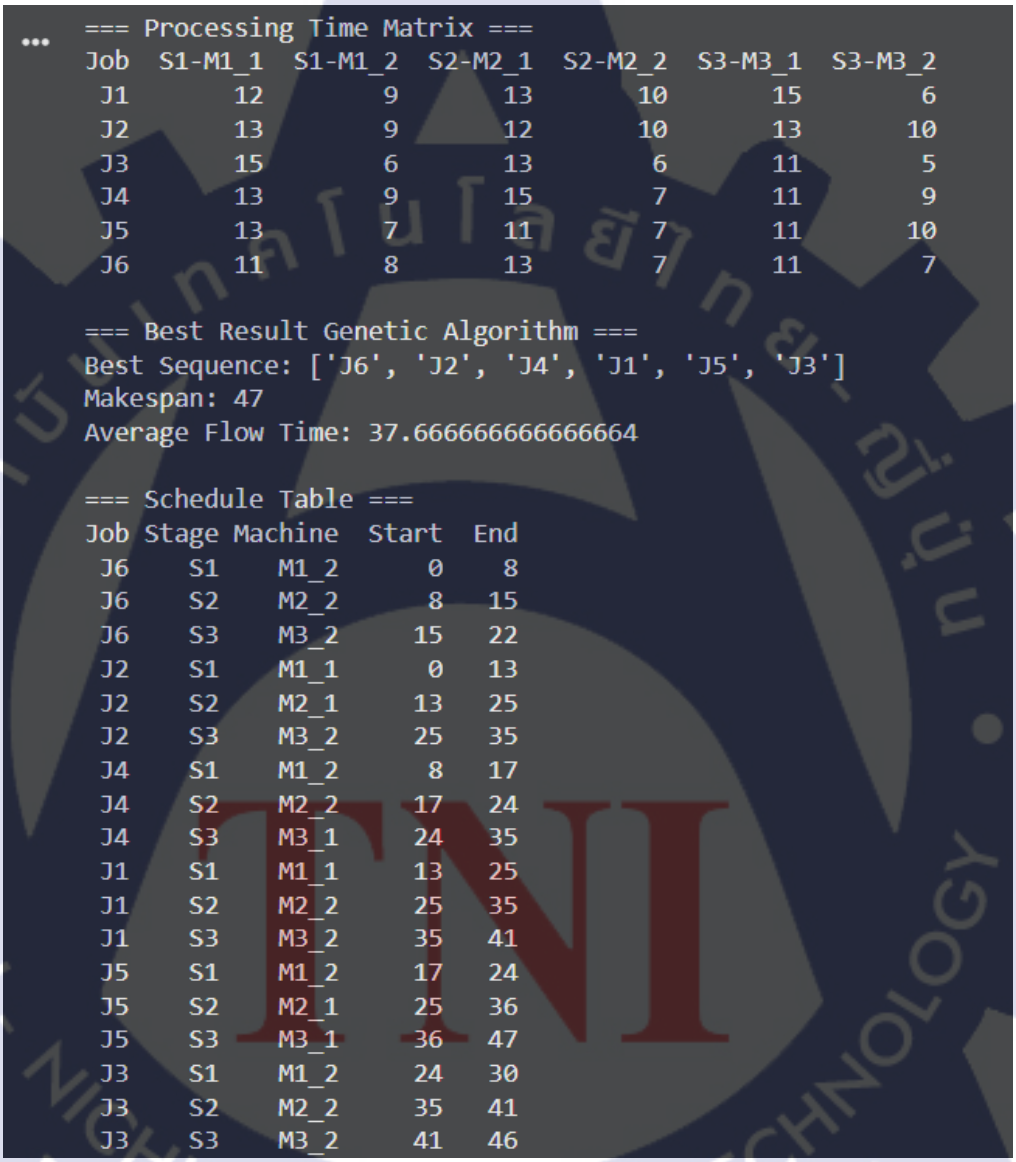
Job	S1-M1_1	S1-M1_2	S2-M2_1	S2-M2_2	S3-M3_1	S3-M3_2
J1	12	9	13	10	15	6
J2	13	9	12	10	13	10
J3	15	6	13	6	11	5
J4	13	9	15	7	11	9
J5	13	7	11	7	11	10
J6	11	8	13	7	11	7

ตารางข้างต้นที่กล่าวมานี้เป็นเวลาการผลิตของเครื่องจักรที่ถูกกำหนดในรูปของช่วงค่า (Range) เพื่อสะท้อนลักษณะความไม่แน่นอนของกระบวนการผลิตจริง โดยกำหนดให้เครื่องจักรกลุ่มแรกมีช่วงเวลาในการผลิตระหว่าง 11–15 หน่วยเวลา และเครื่องจักรกลุ่มที่สองมีช่วงเวลาในการผลิตระหว่าง 5–10 หน่วยเวลา ซึ่งค่าดังกล่าวจะถูกสุ่มในแต่ละรอบของการทดลอง

4.2 ผลการทดลอง

ผลการทดลองของการแก้ปัญหาการจัดตารางการผลิตในระบบ Flexible Flow Shop ซึ่งมีหลายขั้นตอนการผลิตและเครื่องจักรแบบขนาน โดยมีวัตถุประสงค์เพื่อประเมินประสิทธิภาพของ Genetic Algorithm (GA) ในการค้นหาลำดับงานที่เหมาะสม และเปรียบเทียบกับวิธี Brute Force Algorithm (BF) ซึ่งให้คำตอบที่เหมาะสมที่สุด การทดลองดำเนินการโดยใช้ชุดข้อมูลเวลาในการผลิตของเครื่องจักร จำนวน 10 กรณี โดยในแต่ละกรณีมีการกำหนดเวลาการผลิตที่แตกต่างกัน เพื่อสะท้อนสภาพแวดล้อมการผลิตที่หลากหลาย

ตัวชี้วัดที่ใช้ในการประเมินผลประกอบด้วยค่า Makespan ซึ่งแสดงถึงเวลาการผลิตรวมของระบบ และค่า Average Flow Time ซึ่งแสดงถึงประสิทธิภาพการไหลของงานในระบบ ผลลัพธ์จากการทดลองจะถูกนำเสนอในรูปแบบตาราง และนำมาวิเคราะห์เพื่อเปรียบเทียบข้อดี ข้อจำกัด และแนวโน้มของแต่ละวิธี เพื่อสนับสนุนการเลือกใช้วิธีที่เหมาะสมสำหรับปัญหาการจัดตารางการผลิตในระบบจริง

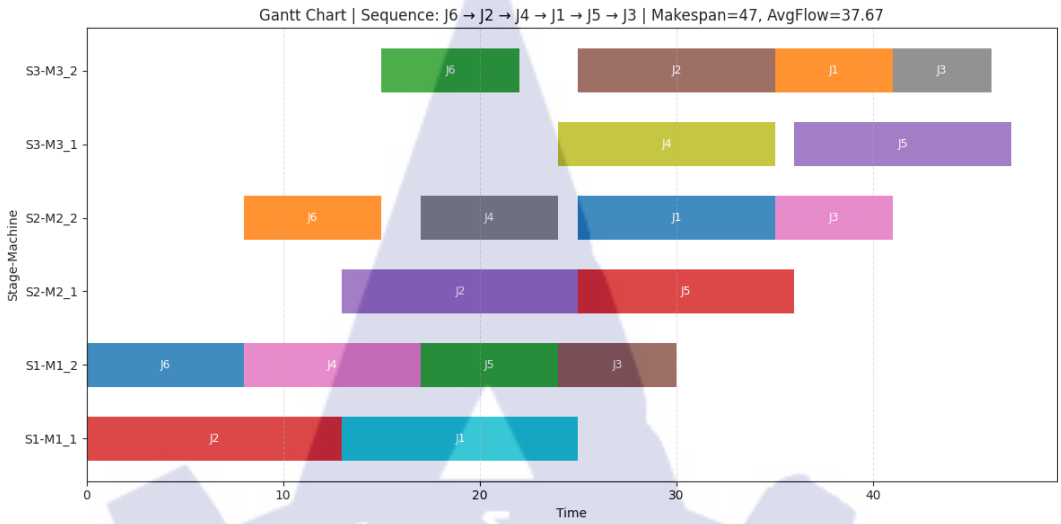


```
=== Processing Time Matrix ===
Job  S1-M1_1  S1-M1_2  S2-M2_1  S2-M2_2  S3-M3_1  S3-M3_2
 J1       12        9       13       10       15        6
 J2       13        9       12       10       13       10
 J3       15        6       13        6       11        5
 J4       13        9       15        7       11        9
 J5       13        7       11        7       11       10
 J6       11        8       13        7       11        7

=== Best Result Genetic Algorithm ===
Best Sequence: ['J6', 'J2', 'J4', 'J1', 'J5', 'J3']
Makespan: 47
Average Flow Time: 37.666666666666664

=== Schedule Table ===
Job Stage Machine  Start  End
 J6    S1    M1_2      0    8
 J6    S2    M2_2      8   15
 J6    S3    M3_2     15   22
 J2    S1    M1_1      0   13
 J2    S2    M2_1     13   25
 J2    S3    M3_2     25   35
 J4    S1    M1_2      8   17
 J4    S2    M2_2     17   24
 J4    S3    M3_1     24   35
 J1    S1    M1_1     13   25
 J1    S2    M2_2     25   35
 J1    S3    M3_2     35   41
 J5    S1    M1_2     17   24
 J5    S2    M2_1     25   36
 J5    S3    M3_1     36   47
 J3    S1    M1_2     24   30
 J3    S2    M2_2     35   41
 J3    S3    M3_2     41   46
```

รูปที่ 4.1 ตัวอย่างผลการทดลองโดยวิธี Genetic Algorithm



รูปที่ 4.2 ตัวอย่าง Gantt Chart ของผลการทดลองโดยวิธี Genetic Algorithm

จากผลการทดลองโดยใช้ ข้อมูลเวลาในการผลิตของเครื่องจักร จำนวน 10 กรณี พบว่า Genetic Algorithm สามารถหาลำดับงานที่เหมาะสมคือ J2, J5, J6, J1, J3 และ J4 โดยให้ค่า Makespan เท่ากับ 47 หน่วยเวลา และค่า Average Flow Time เท่ากับ 36.17 หน่วยเวลา จากตารางการจัดตารางการผลิตพบว่า งาน J2 ซึ่งอยู่ลำดับแรกสามารถดำเนินการได้อย่างต่อเนื่องในทุกขั้นตอน ส่งผลให้มีเวลาเสร็จสิ้นเร็วที่สุด นอกจากนี้ยังพบว่าการมีเครื่องจักรแบบขนาน (Parallel Machines) ช่วยให้บางงาน เช่น J6 สามารถเริ่มการผลิตได้พร้อมกันกับงานอื่น แม้ว่าจะไม่ได้อยู่ในลำดับแรก

จากผลการทดลองโดยใช้ ข้อมูลเวลาในการผลิตของเครื่องจักร จำนวน 10 กรณี พบว่า Genetic Algorithm สามารถหาลำดับงานที่เหมาะสมคือ J6, J2, J4, J1, J5 และ J3 โดยให้ค่า Makespan เท่ากับ 47 หน่วยเวลา และค่า Average Flow Time เท่ากับ 37.67 หน่วยเวลา จากตารางการจัดตารางการผลิต พบว่างาน J6 ซึ่งถูกจัดให้อยู่ลำดับแรก สามารถดำเนินการได้อย่างต่อเนื่องในทุกขั้นตอน โดยไม่มีเวลารอคอย ส่งผลให้มีเวลาเสร็จสิ้นเร็วที่สุด

นอกจากนี้ การมีเครื่องจักรแบบขนานช่วยให้หลายงานสามารถเริ่มต้นการผลิตได้พร้อมกัน ซึ่งช่วยลดเวลาในการรอคอยของระบบโดยรวม แต่ยังมีในขั้นตอนที่ 3 พบว่าเกิดคอขวด (Bottleneck) เนื่องจากมีหลายงานเข้าสู่ขั้นตอนนี้พร้อมกัน ส่งผลให้ค่า Makespan ถูกกำหนดโดยงาน J5 ซึ่งเสร็จสิ้นที่เวลา 47 หน่วยเวลา

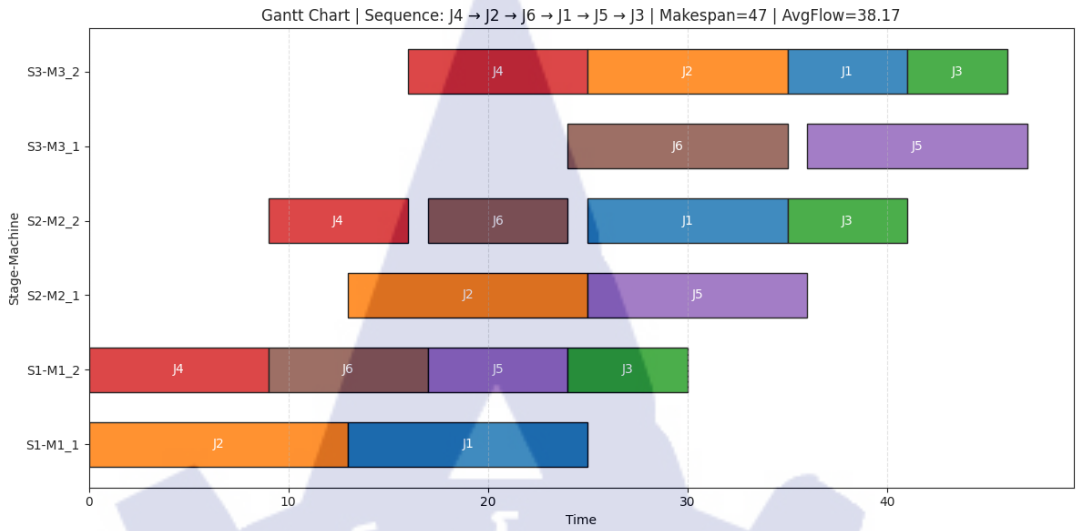
ดังนั้น สามารถสรุปได้ว่า Genetic Algorithm สามารถจัดลำดับงานที่ช่วยเพิ่มประสิทธิภาพของการไหลของงานในระบบได้ดี แม้ว่าจะยังมีข้อจำกัดจากคอขวดในขั้นตอนสุดท้าย

```
=== Processing Time Matrix ===
Job  S1-M1_1  S1-M1_2  S2-M2_1  S2-M2_2  S3-M3_1  S3-M3_2
 J1       12        9       13       10       15        6
 J2       13        9       12       10       13       10
 J3       15        6       13        6       11        5
 J4       13        9       15        7       11        9
 J5       13        7       11        7       11       10
 J6       11        8       13        7       11        7

=== Best Result Brute Force ===
Best Sequence: ['J4', 'J2', 'J6', 'J1', 'J5', 'J3']
Makespan: 47
Average Flow Time: 38.2

=== Schedule Table ===
Job Stage Machine  Start  End
 J4    S1    M1_2      0    9
 J4    S2    M2_2      9   16
 J4    S3    M3_2     16   25
 J2    S1    M1_1      0   13
 J2    S2    M2_1     13   25
 J2    S3    M3_2     25   35
 J6    S1    M1_2      9   17
 J6    S2    M2_2     17   24
 J6    S3    M3_1     24   35
 J1    S1    M1_1     13   25
 J1    S2    M2_2     25   35
 J1    S3    M3_2     35   41
 J5    S1    M1_2     17   24
 J5    S2    M2_1     25   36
 J5    S3    M3_1     36   47
 J3    S1    M1_2     24   30
 J3    S2    M2_2     35   41
 J3    S3    M3_2     41   46
```

รูปที่ 4.3 ตัวอย่างผลการทดลองโดยวิธี Brute Force



รูปที่ 4.4 ตัวอย่าง Gantt Chart ของผลการทดลองโดยวิธี Brute Force

จากผลการทดลองโดยใช้ ข้อมูลเวลาในการผลิตของเครื่องจักร จำนวน 10 กรณี พบว่า Brute Force สามารถจัดลำดับงานที่ดีที่สุดคือ J4, J2, J6, J1, J5 และ J3 โดยให้ค่า Makespan เท่ากับ 47 หน่วยเวลา และค่า Average Flow Time เท่ากับ 38.20 หน่วยเวลา วิธี Brute Force สามารถหาคำตอบที่เหมาะสมที่สุดได้ เนื่องจากทำการพิจารณาทุกลำดับของงานที่เป็นไปได้ทั้งหมด อย่างไรก็ตาม จากผลการทดลองพบว่าค่า Average Flow Time ยังอยู่ในระดับที่สูงเมื่อเปรียบเทียบกับ Genetic Algorithm

จากการวิเคราะห์ตารางการจัดตารางการผลิต พบว่ามีการเกิดคอขวดในขั้นตอนที่ 3 ซึ่งมีหลายงานเข้าสู่ขั้นตอนดังกล่าวพร้อมกัน ส่งผลให้ค่า Makespan ถูกกำหนดโดยงาน J5 ซึ่งเสร็จสิ้นที่เวลา 47 หน่วยเวลานอกจากนี้ การเลือกงาน J4 เป็นงานแรก ส่งผลให้เวลาการไหลของงานโดยรวมสูงกว่ากรณีของ Genetic Algorithm ซึ่งเลือกงาน J6 เป็นงานแรก ดังนั้น แม้ว่า Brute Force จะให้ผลลัพธ์ที่เหมาะสมที่สุดในแง่ของ Makespan แต่ไม่ได้ให้ประสิทธิภาพที่ดีที่สุดในแง่ของ Average Flow Time

ตารางที่ 4.11 ตารางเปรียบเทียบ Makespan และ Average Flow Time โดยใช้วิธี Genetic Algorithm และวิธี Brute Force

Case	GA makespan	BF Makespan	GA Avg Flow Time	BF Avg Flow Time
1	46	46	34.6	34.7
2	47	47	34.8	35.5
3	47	47	36.5	36.2
4	49	47	36.6	37.2
5	48	48	35.2	36.2
6	46	49	39.5	39.5
7	48	48	36.5	37.5
8	47	47	35.8	35.8
9	47	47	36.2	37.7
10	47	47	37.6	38.2

จากตารางผลการทดลองจำนวน 10 กรณี พบว่า Genetic Algorithm สามารถให้ค่า Makespan เท่ากับ Brute Force ได้ถึง 8 กรณี คิดเป็นร้อยละ 80 ของทั้งหมด ซึ่งแสดงให้เห็นว่า Genetic Algorithm มีประสิทธิภาพในการค้นหาคำตอบที่ใกล้เคียงกับค่าที่เหมาะสมที่สุด ในบางกรณี เช่น Case 4 Genetic Algorithm ให้ค่า Makespan สูงกว่า Brute Force เล็กน้อย ซึ่งเป็นลักษณะของวิธีเชิงประมาณ (Heuristic) และในทางกลับกัน พบว่าใน Case 6 Genetic Algorithm ให้ค่า Makespan ต่ำกว่า Brute Force ซึ่งอาจเกิดจากข้อจำกัดของการจัดสรรเครื่องจักรในวิธี Brute Force

เมื่อพิจารณาค่า Average Flow Time พบว่า Genetic Algorithm ให้ค่าต่ำกว่า Brute Force ใน 6 กรณี และมีค่าเท่ากันใน 2 กรณี แสดงให้เห็นว่า Genetic Algorithm สามารถปรับปรุงการไหลของงานในระบบได้ดีกว่า ดังนั้นสามารถสรุปได้ว่า Genetic Algorithm เป็นวิธีที่มีประสิทธิภาพในการแก้ปัญหาการจัดตารางการผลิต โดยสามารถให้ผลลัพธ์ที่ใกล้เคียงวิธี Brute Force ทั้งในด้าน Makespan และ Average Flow Time

4.3 ผลการทดลองภายใต้ข้อจำกัดเวลา

จากการทดลองในหัวข้อก่อนหน้า เป็นการเปรียบเทียบประสิทธิภาพของ Genetic Algorithm (GA) และ Brute Force (BF) โดยใช้จำนวนงานขนาดเล็ก ซึ่งสามารถหาคำตอบที่เหมาะสมที่สุดได้ครบทุกกรณี อย่างไรก็ตาม ในสถานการณ์จริง ปัญหาการจัดตารางการผลิตมักมีขนาดใหญ่และไม่สามารถใช้วิธี Brute Force ในการค้นหาคำตอบทั้งหมดได้ภายในเวลาที่จำกัด

ดังนั้น งานวิจัยนี้จึงได้ทำการทดลองเพิ่มเติมโดยกำหนดเงื่อนไข เวลาการรันสูงสุด 2 นาที สำหรับทั้งสองวิธี โดยใช้ปัญหาขนาดใหญ่ขึ้นที่จำนวน 20 งาน เพื่อจำลองสถานการณ์การใช้งานจริง และประเมินความสามารถของอัลกอริทึมในการค้นหาคำตอบภายใต้ข้อจำกัดด้านเวลา

ตารางที่ 4.12 ผลการเปรียบเทียบค่า Makespan และ Average Flow Time ของวิธี Genetic Algorithm และวิธี Brute Force จากการทดลองจำนวน 10 กรณี โดยใช้เวลารัน 2 นาที

Case	GA makespan	BF Makespan	GA Avg Flow Time	BF Avg Flow Time
1	106	107	66.55	64.25
2	105	115	66.20	71.80
3	112	117	71.00	73.60
4	106	113	67.45	72.50
5	106	113	67.25	71.85
6	108	115	69.55	74.25
7	105	113	67.30	72.65
8	111	117	70.00	73.70
9	104	110	64.90	69.40
10	111	117	71.30	73.15

ผลการทดลองจำนวน 10 กรณี พบว่า Genetic Algorithm ให้ผลลัพธ์ที่ดีกว่าในภาพรวม โดยมีค่า Makespan เฉลี่ยเท่ากับ 107.40 เมื่อเปรียบเทียบกับ Brute Force ที่มีค่าเฉลี่ย 113.70 ซึ่งสามารถลดระยะเวลาการผลิตรวมลงได้ประมาณ 5.54% นอกจากนี้ ค่า Average Flow Time ของ Genetic Algorithm มีค่าเฉลี่ยเท่ากับ 68.15 ซึ่งต่ำกว่าวิธี Brute Force ที่มีค่าเฉลี่ย 71.72 หรือปรับปรุงได้ประมาณ 4.97% แสดงให้เห็นว่า Genetic Algorithm สามารถเพิ่มประสิทธิภาพของระบบการผลิตได้ทั้งในด้านระยะเวลารวมและความรวดเร็วของการไหลของงาน

จากผลการทดลองพบว่า ค่า Makespan ของ Genetic Algorithm ต่ำกว่า Brute Force ในทุกกรณี โดยมีค่าประมาณอยู่ในช่วง 104–112 หน่วยเวลา ขณะที่ Brute Force อยู่ในช่วง 107–117 หน่วยเวลา แสดงให้เห็นว่า GA สามารถค้นหาคำตอบที่มีประสิทธิภาพได้ดีกว่าภายใต้ข้อจำกัดด้านเวลา สำหรับค่า Average Flow Time พบว่า Genetic Algorithm ให้ค่าต่ำกว่า Brute Force ใน 9 จาก 10 กรณี ซึ่งสะท้อนว่า GA สามารถจัดลำดับงานให้มีการไหลของงานในระบบที่มีประสิทธิภาพมากกว่า แต่ในบางกรณี เช่น Case 1 ค่า Average Flow Time ของ GA สูงกว่าเล็กน้อย แสดงให้เห็นว่าแม้ GA จะเน้นการลด Makespan แต่ยังอาจมีผลกระทบต่อการกระจายเวลาของงานบางส่วน

ผลการทดลองนี้แสดงให้เห็นว่า ภายใต้ข้อจำกัดของเวลาในการคำนวณ วิธี Brute Force ไม่สามารถค้นหาคำตอบที่เหมาะสมที่สุดได้ครบถ้วน เนื่องจากจำนวนลำดับงานที่เพิ่มขึ้นแบบแฟกทอเรียล (Factorial Growth) ในขณะที่ Genetic Algorithm สามารถให้คำตอบที่มีคุณภาพสูงได้ภายในเวลาอันจำกัด ดังนั้น Genetic Algorithm จึงเหมาะสมสำหรับการแก้ปัญหาการจัดตารางการผลิตในระบบขนาดใหญ่

จากการทดลองทั้งหมดสามารถสรุปได้ว่า Genetic Algorithm มีประสิทธิภาพเหนือกว่า Brute Force อย่างชัดเจนในกรณีปัญหาขนาดใหญ่ภายใต้ข้อจำกัดด้านเวลา โดยสามารถให้ผลลัพธ์ที่ดีกว่าทั้งในด้าน Makespan และ Average Flow Time จึงเป็นวิธีที่เหมาะสมสำหรับการประยุกต์ใช้ในระบบการผลิตจริง

บทที่ 5
สรุปผล อภิปรายผล และข้อเสนอแนะ

5.1 สรุปผลการวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนาแบบจำลองและแนวทางการแก้ปัญหาการจัดตารางการผลิตในระบบ Flexible flow shop (FFS) ซึ่งเป็นปัญหาที่มีความซับซ้อนสูงและจัดอยู่ในกลุ่ม NP-hard โดยมุ่งเน้นการลดเวลาการผลิตรวมของระบบ (Makespan) และค่าเฉลี่ยเวลาการไหลของงาน (Average Flow Time) จากบทที่ 3 ได้มีการพัฒนาแบบจำลองทางคณิตศาสตร์ของระบบการผลิต ซึ่งประกอบด้วยงานจำนวน 6 งาน และ 3 ขั้นตอนการผลิต โดยในแต่ละขั้นตอนมีเครื่องจักรแบบขนานที่มีประสิทธิภาพแตกต่างกัน (Non-uniform Machines) พร้อมทั้งได้ออกแบบอัลกอริทึม Genetic Algorithm (GA) สำหรับใช้ในการค้นหาลำดับการผลิตที่เหมาะสม รวมถึงพัฒนาโปรแกรมด้วยภาษา Python เพื่อใช้ในการจำลองและประเมินผลลัพธ์ ในบทที่ 4 ได้ทำการทดลองโดยใช้ข้อมูลเวลาในการผลิตจำนวน 10 กรณี เพื่อประเมินประสิทธิภาพของ Genetic Algorithm และเปรียบเทียบกับวิธี Brute Force ซึ่งเป็นวิธีที่ให้คำตอบที่เหมาะสมที่สุด

ผลการทดลองพบว่า

- Genetic Algorithm สามารถให้ค่า Makespan เท่ากับ Brute Force ได้ใน 8 จาก 10 กรณี (คิดเป็นร้อยละ 80)

- ในบางกรณี Genetic Algorithm ให้ค่า Makespan สูงกว่าเล็กน้อย ซึ่งเป็นลักษณะของวิธีเชิงประมาณ (Approximation Method)

- ในด้าน Average Flow Time พบว่า Genetic Algorithm ให้ค่าต่ำกว่าหรือเทียบเท่าวิธี Brute Force ในส่วนใหญ่ของกรณีศึกษา ซึ่งแสดงให้เห็นว่าอัลกอริทึมสามารถจัดลำดับงานได้อย่างมีประสิทธิภาพ ส่งผลให้งานแต่ละงานใช้เวลารอคอยภายในระบบลดลง และสามารถเข้าสู่กระบวนการผลิตในขั้นตอนถัดไปได้รวดเร็วมากขึ้น

ผลลัพธ์ดังกล่าวแสดงให้เห็นว่าแม้ Genetic Algorithm จะเป็นวิธีเชิงประมาณ แต่ยังสามารถให้ผลลัพธ์ที่ดีทั้งในด้านการลดเวลาในการผลิตรวม (Makespan) และการลดระยะเวลาการไหลของงาน (Average Flow Time) ซึ่งมีความเหมาะสมต่อการประยุกต์ใช้ในระบบการผลิตจริงที่ต้องการทั้งความรวดเร็วและประสิทธิภาพในการจัดการสายการผลิต

นอกจากนี้ ค่า Average Flow Time ที่ต่ำยังสะท้อนถึงการไหลของงานภายในระบบการผลิตที่มีความต่อเนื่องมากขึ้น ลดปัญหาการสะสมของงานระหว่างขั้นตอนการผลิต (Work in Process: WIP) และช่วยเพิ่มประสิทธิภาพการใช้ทรัพยากรของเครื่องจักรในแต่ละขั้นตอนการผลิต

เพื่อประเมินประสิทธิภาพของอัลกอริทึมภายใต้เงื่อนไขที่ใกล้เคียงกับการใช้งานจริง งานวิจัยนี้ได้ทำการทดลองเพิ่มเติมโดยกำหนดเวลาในการประมวลผลของแต่ละวิธีเท่ากันที่ 2 นาที และเพิ่มขนาดของปัญหาเป็น 20 งาน ซึ่งส่งผลให้จำนวนคำตอบที่เป็นไปได้เพิ่มขึ้นอย่างมากในระดับแฟกทอเรียล ผลการทดลองพบว่า วิธี Brute Force ไม่สามารถค้นหาคำตอบที่เหมาะสมที่สุดได้ภายในเวลาที่กำหนด เนื่องจากต้องทำการพิจารณาทุกลำดับความเป็นไปได้ของงาน ซึ่งมีจำนวนมหาศาลและไม่สามารถคำนวณได้ทันในทางปฏิบัติ ในขณะที่ Genetic Algorithm สามารถให้คำตอบได้ภายในเวลา 2 นาทีอย่างสม่ำเสมอ และให้ค่าผลลัพธ์ในระดับที่มีคุณภาพเหมาะสม โดยค่า Makespan ที่ได้มีความใกล้เคียงกับค่าที่ดีที่สุด และค่า Average Flow Time มีแนวโน้มต่ำกว่าในหลายกรณี

ผลการทดลองนี้แสดงให้เห็นอย่างชัดเจนว่า เมื่อขนาดของปัญหาเพิ่มขึ้น วิธี Brute Force ไม่เหมาะสมต่อการนำไปใช้งานจริง ในขณะที่ Genetic Algorithm สามารถรักษาสมดุลระหว่างคุณภาพของคำตอบและเวลาในการคำนวณได้อย่างมีประสิทธิภาพ

5.2 อภิปรายผลการวิจัย

จากผลการทดลองสามารถอภิปรายประเด็นสำคัญได้ดังต่อไปนี้

5.2.1 ประสิทธิภาพของ Genetic Algorithm

ผลการทดลองแสดงให้เห็นว่า Genetic Algorithm มีความสามารถในการค้นหาคำตอบที่มีค่า Makespan ใกล้เคียงกับวิธี Brute Force ซึ่งเป็นคำตอบที่ดีที่สุดเชิงทฤษฎี ทั้งนี้เนื่องจากกลไกของอัลกอริทึมที่ใช้กระบวนการคัดเลือก การผสมพันธุ์ และการกลายพันธุ์ ทำให้สามารถสำรวจพื้นที่คำตอบได้อย่างมีประสิทธิภาพ นอกจากนี้ Genetic Algorithm ยังสามารถให้ค่า Average Flow Time ที่ต่ำกว่าในหลายกรณี ซึ่งสะท้อนถึงความสามารถในการจัดลำดับงานที่ช่วยลดระยะเวลาการรอคอยในระบบการผลิต

5.2.2 ผลกระทบของโครงสร้างระบบการผลิต

จากการวิเคราะห์ Gantt Chart พบว่า ขั้นตอนการผลิตสุดท้าย (Stage 3) เป็นคอขวดของระบบ (Bottleneck) เนื่องจากมีงานหลายงานเข้าสู่ขั้นตอนดังกล่าวในเวลาใกล้เคียงกัน ส่งผลให้เครื่องจักรในขั้นตอนนี้มีการใช้งานสูง และเป็นตัวกำหนดค่า Makespan ของระบบ ผลการศึกษานี้ชี้ให้เห็นว่า แม้จะใช้วิธีการจัดตารางการผลิตที่มีประสิทธิภาพ แต่ข้อจำกัดด้านโครงสร้างของระบบ เช่น จำนวนเครื่องจักรหรือความสามารถในการผลิต ยังคงมีผลต่อประสิทธิภาพโดยรวมของระบบ

5.2.3 ความเหมาะสมของ Genetic Algorithm สำหรับปัญหา Flexible flow shop

เนื่องจากปัญหา FFS มีจำนวนคำตอบที่เป็นไปได้เพิ่มขึ้นแบบแฟกทอเรียลเมื่อจำนวนงานเพิ่มขึ้น การใช้วิธี Brute Force จึงไม่เหมาะสมสำหรับปัญหาขนาดใหญ่ ในทางตรงกันข้าม Genetic Algorithm สามารถให้คำตอบที่มีคุณภาพดีโดยไม่จำเป็นต้องค้นหาทุกคำตอบที่เป็นไปได้ จึงเหมาะสมสำหรับการประยุกต์ใช้ในระบบการผลิตจริงที่มีความซับซ้อนสูง

5.2.4 ความไม่แน่นอนของผลลัพธ์

เนื่องจาก Genetic Algorithm เป็นอัลกอริทึมที่มีลักษณะการค้นหาแบบสุ่ม (Stochastic Search) ผลลัพธ์ที่ได้ในแต่ละรอบการทดลองอาจแตกต่างกัน ดังนั้นการทดลองซ้ำหลายรอบจึงมีความจำเป็น เพื่อให้สามารถประเมินเสถียรภาพของอัลกอริทึมได้อย่างเหมาะสม

5.3 การอภิปรายประโยชน์ของ Genetic Algorithm

จากผลการทดลองสามารถสรุปประโยชน์ของการใช้ Genetic Algorithm ในการแก้ปัญหาการจัดตารางการผลิตได้ดังนี้

5.3.1 ช่วยลดเวลาในการคำนวณอย่างมีนัยสำคัญ เนื่องจากไม่จำเป็นต้องค้นหาทุกคำตอบที่เป็นไปได้เหมือนวิธี Brute Force ส่งผลให้สามารถแก้ปัญหาขนาดใหญ่ได้ภายในระยะเวลาที่จำกั

5.3.2 สามารถให้คำตอบที่มีคุณภาพใกล้เคียงกับคำตอบที่เหมาะสมที่สุด โดยจากผลการทดลองในบทที่ 4 พบว่าสามารถให้ค่า Makespan เท่ากับวิธี Brute Force ได้ในส่วนใหญ่ของกรณีศึกษา

5.3.3 มีความสามารถในการปรับปรุงการไหลของงานในระบบ โดยให้ค่า Average Flow Time ที่ต่ำกว่าในหลายกรณี ซึ่งสะท้อนถึงการจัดลำดับงานที่ช่วยลดเวลาการรอคอยในระบบการผลิต

5.3.4 มีความเหมาะสมสำหรับระบบการผลิตที่มีความซับซ้อน เช่น ระบบ Flexible Flow Shop ที่มีเครื่องจักรขนานหลายเครื่องและมีประสิทธิภาพแตกต่างกัน

5.4 ข้อจำกัดของ Genetic Algorithm

แม้ว่า Genetic Algorithm จะมีข้อดีหลายประการ แต่ยังคงมีข้อจำกัดที่ควรพิจารณา ดังนี้

5.4.1 ไม่สามารถรับประกันได้ว่าจะให้คำตอบที่เหมาะสมที่สุดในทุกกรณี เนื่องจากเป็นวิธีการเชิงประมาณ โดยจากผลการทดลองพบว่าในบางกรณีค่า Makespan ที่ได้อาจสูงกว่าวิธี Brute Force เล็กน้อย

5.4.2 เนื่องจากเป็นอัลกอริทึมที่มีลักษณะการค้นหาแบบสุ่ม ผลลัพธ์ที่ได้ในแต่ละรอบการทดลองอาจแตกต่างกัน จึงจำเป็นต้องทำการทดลองซ้ำหลายครั้งเพื่อประเมินความเสถียรของผลลัพธ์

5.4.3 ประสิทธิภาพของ Genetic Algorithm ขึ้นอยู่กับการกำหนดค่าพารามิเตอร์ เช่น ขนาดประชากร อัตราการผสมพันธุ์ และอัตราการกลายพันธุ์ หากกำหนดค่าไม่เหมาะสมอาจส่งผลให้การค้นหาคำตอบไม่มีประสิทธิภาพ

5.5 เหตุผลในการเลือกใช้ Genetic Algorithm แทน Brute Force

แม้ว่าวิธี Brute Force จะสามารถให้คำตอบที่เหมาะสมที่สุดเชิงทฤษฎีได้ แต่เนื่องจากปัญหาการจัดตารางการผลิตแบบ Flexible Flow Shop เป็นปัญหาประเภท NP-hard ซึ่งจำนวนคำตอบจะเพิ่มขึ้นอย่างรวดเร็วเมื่อจำนวนงานเพิ่มขึ้น ส่งผลให้ไม่สามารถนำวิธีดังกล่าวไปใช้แก้ปัญหาขนาดใหญ่ได้ในทางปฏิบัติ ในทางตรงกันข้าม Genetic Algorithm เป็นวิธีการเชิงเมตาฮิวริสติกที่สามารถค้นหาคำตอบที่มีคุณภาพดีได้ภายในระยะเวลาที่จำกัด โดยไม่จำเป็นต้องพิจารณาทุกคำตอบที่เป็นไปได้

จากผลการทดลองเพิ่มเติมที่กำหนดเวลาในการคำนวณเท่ากัน พบว่า Genetic Algorithm สามารถให้คำตอบได้อย่างมีประสิทธิภาพ ในขณะที่วิธี Brute Force ไม่สามารถให้คำตอบที่ใช้งานได้ภายในเวลาที่กำหนด ดังนั้นจึงสามารถสรุปได้ว่า Genetic Algorithm เป็นวิธีที่มีความเหมาะสมมากกว่าในเชิงปฏิบัติ สำหรับการแก้ปัญหาการจัดตารางการผลิตในระบบ Flexible Flow Shop ที่มีความซับซ้อนและมีขนาดใหญ่

5.6 ข้อจำกัดของงานวิจัย

แม้ว่างานวิจัยนี้จะสามารถบรรลุวัตถุประสงค์ที่ตั้งไว้ แต่ยังมีข้อจำกัดบางประการ ได้แก่

5.6.1 ขนาดของปัญหาที่ศึกษาอยู่ในระดับขนาดเล็ก (6 งาน)

5.6.2 ไม่ได้พิจารณาเวลาในการปรับตั้งเครื่องจักร (Setup Time)

5.6.3 ไม่ได้พิจารณาความไม่แน่นอนของเครื่องจักร เช่น การหยุดทำงาน (Machine Breakdown)

5.7 ข้อเสนอแนะ

เพื่อพัฒนางานวิจัยในด้านนี้ให้มีความสมบูรณ์มากยิ่งขึ้น สามารถดำเนินการต่อยอดได้ดังนี้

5.7.1 เพิ่มขนาดของปัญหาเช่น เพิ่มจำนวนงานและจำนวนเครื่องจักร เพื่อทดสอบประสิทธิภาพของอัลกอริทึมในระบบที่มีความซับซ้อนมากขึ้น

5.7.2 พัฒนาอัลกอริทึมแบบผสม (Hybrid Algorithm) เช่น การรวม Genetic Algorithm กับ Local Search หรือ Metaheuristic อื่น ๆ เพื่อเพิ่มประสิทธิภาพในการค้นหาคำตอบ

5.7.3 พิจารณาปัจจัยเพิ่มเติมในระบบการผลิต เช่น Setup Time, Machine Breakdown, หรือ Due Date เพื่อให้แบบจำลองมีความใกล้เคียงกับระบบจริงมากขึ้น

5.7.4 ปรับปรุงฟังก์ชันวัตถุประสงค์ โดยเพิ่มตัวชี้วัดอื่น เช่น Machine Utilization, Total Tardiness หรือ Energy Consumption

5.7.5 ประยุกต์ใช้กับข้อมูลจริงจากภาคอุตสาหกรรม เพื่อทดสอบความสามารถของแบบจำลองในสถานการณ์จริง และเพิ่มคุณค่าทางวิชาการและการใช้งาน

บรรณานุกรม

บรรณานุกรม

[1] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, New York: Springer, 2016.

[2] K. R. Baker and D. Trietsch, *Principles of Sequencing and Scheduling*, Hoboken, NJ: John Wiley & Sons, 2009.

[3] A. S. Jain and S. Meeran, "Deterministic job-shop scheduling: Past, present and future," *European Journal of Operational Research*, vol. 113, no. 2, pp. 390-434, March 1999.

[4] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," *Annals of Operations Research*, vol. 41, no. 3, pp. 157-183, March 1993.

[5] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press, 1975.

[6] S. Kirkpatrick et al., "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671-680, May 1983.

[7] E. Nowicki and C. Smutnicki, "A fast taboo search algorithm for the job shop problem," *Management Science*, vol. 42, no. 6, pp. 797-813, June 1996.

[8] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Optimization*, New York: John Wiley & Sons, 2000.

[7] E. Nowicki and C. Smutnicki, "A fast taboo search algorithm for the job shop problem," *Management Science*, vol. 42, no. 6, pp. 797-813, June 1996.

[9] R. Ruiz and C. Maroto, "A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility," *European Journal of Operational Research*, vol. 169, no. 3, pp. 781-800, March 2006.

[10] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," *Annals of Operations Research*, vol. 41, no. 3, pp. 157-183, March 1993.

[11] J. Gao et al., “A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems,” *Computers & Operations Research*, vol. 35, no. 9, pp. 2892-2907, September 2008.

[12] I. Kacem et al., “Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic,” *Mathematics and Computers in Simulation*, vol. 60, no. 3-5, pp. 245-276, November 2002.

[13] F. Pezzella et al., “A genetic algorithm for the flexible job-shop scheduling problem,” *Computers & Operations Research*, vol. 35, no. 10, pp. 3202-3212, October 2008.

[14] X. Li and L. Gao, “An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem,” *International Journal of Production Economics*, vol. 174, pp. 93-110, April 2016.

[15] G. Zhang et al., “An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem,” *Computers & Industrial Engineering*, vol. 56, no. 4, pp. 1309-1318, May 2009.

[16] A. S. Jain and S. Meeran, “Deterministic job-shop scheduling: Past, present and future,” *European Journal of Operational Research*, vol. 113, no. 2, pp. 390-434, March 1999.

[17] สุภาวดี บุญช่วย และคณะ, “การประยุกต์ใช้วิธีการทางพันธุกรรมแบบปรับปรุงสำหรับปัญหาการจัดตารางการผลิตแบบยืดหยุ่น,” *วารสารวิศวกรรมอุตสาหการ*, ปีที่ 12, ฉบับที่ 2, หน้า 45-53, กรกฎาคม – ธันวาคม 2562.

[18] พิสิฐ พงษ์ประเสริฐ, *“การจัดตารางการผลิตแบบ Flexible Flow Shop โดยใช้วิธีฮิวริสติกและโปรแกรมเชิงเส้นจำนวนเต็ม,”* วิทยานิพนธ์ วศ.ม. (สาขาวิศวกรรมอุตสาหการ), มหาวิทยาลัยเชียงใหม่, เชียงใหม่, 2561.

[19] ณัฐวุฒิ ศรีสมบัติ และคณะ, “การจัดตารางการผลิตสำหรับเครื่องจักรขนานแบบไม่สัมพันธ์กัน,” *วารสารวิชาการวิศวกรรมศาสตร์*, ปีที่ 15, ฉบับที่ 1, หน้า 88-97, มกราคม – มิถุนายน 2563.

[20] ธนภัทร ศรีสุข, “การจัดตารางการผลิตที่มีเวลา Setup Time โดยใช้วิธี Forward และ Backward Scheduling,” วิทยานิพนธ์ วศ.ม. (สาขาวิศวกรรมอุตสาหการ), มหาวิทยาลัยขอนแก่น, จังหวัดขอนแก่น, 2560.

[21] กิตติพงษ์ แสงทอง และคณะ, “การประยุกต์ใช้กฎการจัดลำดับงานสำหรับระบบการผลิตแบบ Job Shop,” *วารสารเทคโนโลยีอุตสาหกรรม*, ปีที่ 14, ฉบับที่ 3, หน้า 101-110, กันยายน – ธันวาคม 2561.

ภาคผนวก

ภาคผนวก ก.

Genetic Algorithm Code

ภาคผนวก ก. Genetic Algorithm Code

```
import random
import pandas as pd
import matplotlib.pyplot as plt
# ==========================
# 1) Problem Data
# ==========================
jobs = ["J1", "J2", "J3", "J4", "J5"]
stages = ["S1", "S2", "S3"]
machines = {
    "S1": ["M1_1", "M1_2"],
    "S2": ["M2_1", "M2_2"],
    "S3": ["M3_1", "M3_2"]
}
# ==========================
# 2) Generate random processing times
#    _1 -> 11 to 15
#    _2 -> 5 to 10
# ==========================
```

```
def generate_processing_time():
    processing_time = {}
    for job in jobs:
        processing_time[job] = {}
        for stage in stages:
            processing_time[job][stage] = {}
            for machine in machines[stage]:
                if machine.endswith("_1"):
                    p = random.randint(11, 15)
                else:
                    p = random.randint(5, 10)
                processing_time[job][stage][machine] = p
    return processing_time
processing_time = generate_processing_time()
# ========================
# 3) Calculate Makespan
# ========================
def calculate_makespan(sequence):
    job_ready = {job: 0 for job in jobs}
    machine_free = {
```

```
        "S1": {"M1_1": 0, "M1_2": 0},
        "S2": {"M2_1": 0, "M2_2": 0},
        "S3": {"M3_1": 0, "M3_2": 0}
    }
    for job in sequence:
        for stage in stages:
            best_machine = None
            best_end = float("inf")
            for machine in machines[stage]:
                p = processing_time[job][stage][machine]
                start = max(job_ready[job], machine_free[stage][machine])
                end = start + p
                if end < best_end:
                    best_end = end
                    best_machine = machine
            job_ready[job] = best_end
            machine_free[stage][best_machine] = best_end
    makespan = max(job_ready.values())
    return makespan
# ========================
```

```
# 4) Decode full schedule
# ========================
def decode_schedule(sequence):
    job_ready = {job: 0 for job in jobs}
    machine_free = {
        "S1": {"M1_1": 0, "M1_2": 0},
        "S2": {"M2_1": 0, "M2_2": 0},
        "S3": {"M3_1": 0, "M3_2": 0}
    }
    schedule = []
    for job in sequence:
        for stage in stages:
            best_machine = None
            best_start = None
            best_end = float("inf")
            for machine in machines[stage]:
                p = processing_time[job][stage][machine]
                start = max(job_ready[job], machine_free[stage][machine])
                end = start + p
                if end < best_end:
```

```
                best_end = end
                best_start = start
                best_machine = machine
            job_ready[job] = best_end
            machine_free[stage][best_machine] = best_end
            schedule.append([job, stage, best_machine, best_start, best_end])
    makespan = max(job_ready.values())
    avg_flow_time = sum(job_ready.values()) / len(job_ready)
    return schedule, makespan, avg_flow_time
# ========================
# 5) Create initial population
# ========================
def create_population(size):
    population = [ ]
    for _ in range(size):
        chromosome = jobs[:]
        random.shuffle(chromosome)
        population.append(chromosome)
    return population
# ========================
```

```
# 6) Order Crossover (OX)
# ========================
def crossover(parent1, parent2):
    n = len(parent1)
    start, end = sorted(random.sample(range(n), 2))
    child = [None] * n
    # copy slice from parent1
    child[start:end] = parent1[start:end]
    # fill remaining from parent2 in order
    pointer = 0
    for gene in parent2:
        if gene not in child:
            while child[pointer] is not None:
                pointer += 1
            child[pointer] = gene
    return child
# ========================
# 7) Swap Mutation
# ========================
def mutation(chromosome):
```

```
    i, j = random.sample(range(len(chromosome)), 2)
    chromosome[i], chromosome[j] = chromosome[j], chromosome[i]
    return chromosome
# ==========================
# 8) Genetic Algorithm
# ==========================
def genetic_algorithm(pop_size=50, generations=200, mutation_rate=0.1):
    population = create_population(pop_size)
    best_solution = None
    best_makespan = float("inf")
    for g in range(generations):
        fitness = [ ]
        for chromosome in population:
            ms = calculate_makespan(chromosome)
            fitness.append((chromosome, ms))
            if ms < best_makespan:
                best_makespan = ms
                best_solution = chromosome[:]
            fitness.sort(key=lambda x: x[1])
        selected = [chrom for chrom, _ in fitness[:pop_size // 2]]
```

```
        new_population = [ ]
        while len(new_population) < pop_size:
            p1, p2 = random.sample(selected, 2)
            child = crossover(p1, p2)
            if random.random() < mutation_rate:
                child = mutation(child)
            new_population.append(child)
        population = new_population
    return best_solution, best_makespan
# =========================
# 9) Print Processing Time Matrix
# =========================
def print_processing_time_matrix(processing_time):
    rows = [ ]
    for job in jobs:
        row = {
            "Job": job,
            "S1-M1_1": processing_time[job]["S1"]["M1_1"],
            "S1-M1_2": processing_time[job]["S1"]["M1_2"],
            "S2-M2_1": processing_time[job]["S2"]["M2_1"],
```

```
            "S2-M2_2": processing_time[job]["S2"]["M2_2"],
            "S3-M3_1": processing_time[job]["S3"]["M3_1"],
            "S3-M3_2": processing_time[job]["S3"]["M3_2"],
        }
        rows.append(row)
    df_pt = pd.DataFrame(rows)
    print("=== Processing Time Matrix ===")
    print(df_pt.to_string(index=False))
    return df_pt
# ========================
# 10) Plot Gantt Chart
# ========================
def plot_gantt(schedule, sequence, makespan, avg_flow_time):
    fig, ax = plt.subplots(figsize=(14, 8))
    machine_y = {
        "S1-M1_1": 0,
        "S1-M1_2": 1,
        "S2-M2_1": 2,
        "S2-M2_2": 3,
        "S3-M3_1": 4,
```

```
    "S3-M3_2": 5
}
job_colors = {
"J1": "tab:blue",
"J2": "tab:orange",
"J3": "tab:green",
"J4": "tab:red",
"J5": "tab:purple",
"J6": "tab:brown",
}
for job, stage, machine, start, end in schedule:
    label = f"{stage}-{machine}"
    y = machine_y[label]
    ax.barh(
        y,
        end - start,
        left=start,
        height=0.6,
        color=job_colors[job],
        edgecolor="black"
```

```
        )
        text_color = "black"
        ax.text(
            start + (end - start) / 2,
            y,
            job,
            ha='center',
            va='center',
            color=text_color,
            fontsize=16
        )
    ax.set_yticks(list(machine_y.values()))
    ax.set_yticklabels(list(machine_y.keys()), fontsize=14)
    ax.set_xlabel("Time", fontsize=14)
    ax.set_ylabel("Stage-Machine", fontsize=14)
    ax.set_title(
        f"Gantt Chart | Sequence: {' ⟶ '.join(sequence)} | "
        f"Makespan={makespan}, AvgFlow={avg_flow_time:.2f}",
        fontsize=16
    )
```

```
    ax.tick_params(axis='x', labelsize=14)
    ax.grid(True, axis='x', linestyle='--', alpha=0.4)
    plt.tight_layout()
    plt.show()
# ==========================
# 11) Run everything
# ==========================
df_pt = print_processing_time_matrix(processing_time)
best_sequence, best_makespan = genetic_algorithm(
    pop_size=50,
    generations=200,
    mutation_rate=0.1
)
schedule, makespan, avg_flow_time = decode_schedule(best_sequence)
print("\n=== Best GA Result ===")
print("Best Sequence:", best_sequence)
print("Makespan:", makespan)
print("Average Flow Time:", avg_flow_time)
# Schedule table
```

```
df_schedule = pd.DataFrame(schedule, columns=["Job", "Stage", "Machine", "Start",
"End"])

print("\n=== Schedule Table ===")

print(df_schedule.to_string(index=False))

# Gantt Chart

plot_gantt(schedule, best_sequence, makespan, avg_flow_time)
```

Brute Force Code

```
import itertools

import pandas as pd

import matplotlib.pyplot as plt

# ==========================

# Brute Force  processing_time

# ==========================

def brute_force(processing_time):

    best_sequence = None

    best_makespan = float("inf")

    best_avg_flow_time = None

    best_schedule = None

    all_sequences = list(itertools.permutations(jobs))
```

```
    for sequence in all_sequences:
        schedule, makespan, avg_flow_time = decode_schedule(sequence)
        if makespan < best_makespan:
            best_makespan = makespan
            best_sequence = sequence
            best_avg_flow_time = avg_flow_time
            best_schedule = schedule
    return list(best_sequence), best_makespan, best_avg_flow_time, best_schedule
# ========================
# Run Brute Force
# ========================
bf_sequence, bf_makespan, bf_avg_flow_time, bf_schedule =
brute_force(processing_time)
print("\n=== Brute Force Best Result ===")
print("Best Sequence:", bf_sequence)
print("Makespan:", bf_makespan)
print("Average Flow Time:", bf_avg_flow_time)
df_bf_schedule = pd.DataFrame(
    bf_schedule,
    columns=["Job", "Stage", "Machine", "Start", "End"]
```

```
)

print("\n=== Brute Force Schedule Table ===")

print(df_bf_schedule.to_string(index=False))

# ==========================

# Plot Gantt Chart

# ==========================

plot_gantt(

    bf_schedule,

    bf_sequence,

    bf_makespan,

    bf_avg_flow_time

)
```