Optimization of an RMG/Textile Plant through Discrete Event Simulation and Multi-Objective Genetic Algorithm: A Case Study

Submitted By

Rahin Intesar Nafim – 200012150

Nazia Tasnim – 200012141

Sidratul Muntaha – 200012124

Supervised By

Prof. Dr. A.R.M. Harunur Rashid

A Thesis submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Industrial and Production Engineering

Department of Mechanical and Production Engineering (MPE)

Islamic University of Technology (IUT)

October, 2025

Candidate's Declaration

This is to certify that the work presented in this thesis, titled, **"Optimization of an RMG/Textile Plant through Discrete Event Simulation and Multi-Objective Genetic Algorithm: A Case Study"**, is the outcome of the investigation and research carried out by us under the supervision of **Dr. A.R.M. Harunur Rashid, Professor, IUT.**

It is also declared that neither this thesis nor any part of it has been submitted elsewhere for the award of any degree or diploma.

Name of the Student: Rahin Intesar Nafim
Student No: 200012150

Name of the Student: Nazia Tasnim
Student No: 200012141

Name of the Student: Sidratul Muntaha
Student No: 200012124

Recommendation of the Thesis Supervisors

The thesis titled **"CASE STUDY: OPTIMIZATION OF AN RMG/TEXTILE PLANT THROUGH DISCRETE EVENT SIMULATION AND MULTI-OBJECTIVE GENETIC ALGORITHM**" submitted by (1) **RAHIN INTESAR NAFIM, Student No: 200012150**, (2) **NAZIA TASNIM, Student No: 200012141**, (3) **SIDRATUL MUNTAHA, Student No: 200012124** has been accepted as satisfactory in partial fulfillment of the requirements for the degree of B Sc. in Industrial and Production Engineering **on 20th October, 2025.**

Dr. A. R. M. Harunur Rashid (Supervisor)
Professor
MPE Dept., IUT, Board Bazar, Gazipur-1704, Bangladesh.

CO-PO Mapping of ME 4800 -Thesis and Project

COs	Course Outcomes (CO) Statement	(PO)	Addressed by	
CO1	Discover and Locate research problems and illustrate them via figures/tables or projections/ideas through field visit and literature review and determine/Setting aim and objectives of the project/work/research in specific, measurable, achievable, realistic and timeframe manner.	**PO2 Problem analysis**	Thesis Book	√
			Performance by research	√
			Presentation and soft skill	√
CO2	Design research solutions of the problems towards achieving the objectives and its application. Design systems, components or processes that meets related needs in the field of mechanical engineering	**PO3 Design/development of solutions**	Thesis Book	√
			Performance by research	√
			Presentation and soft skill	√
CO3	Review, debate, compare and contrast the relevant literature contents. Relevance of this research/study. Methods, tools, and techniques used by past researchers and justification of use of them in this work.	**PO4 Investigation**	Thesis Book	√
			Performance by research	√
			Presentation and soft skill	√
CO4	Analyse data and exhibit results using tables, diagrams, graphs with their interpretation. Investigate the designed solutions to solve the problems through case study/survey study/experimentation/simulation using modern tools and techniques.	**PO5 Modern tool usage**	Thesis Book	√
			Performance by research	√
			Presentation and soft skill	√
CO5	Apply moral values and research/professional ethics throughout the work, and justify genuine referencing on sources, and demonstration of own contribution.	**PO8 Ethics**	Thesis Book	√
			Performance by research	√
			Presentation and soft skill	√
CO6	Perform own self and manage group activities from the beginning to the end of the research/work as a quality work.	**PO9 Individual work and teamwork**	Thesis Book	√
			Performance by research	√
			Presentation and soft skill	√
CO7	Compile and arrange the work outputs, write the report/thesis, a sample journal paper, and present the work to wider audience using modern communication tools and techniques.	**PO10 Communication**	Thesis Book	√
			Performance by research	√
			Presentation and soft skill	√
CO8	Recognize the necessity of life-long learning in career development in dynamic real-world situations from the experience of completing this project.	**PO12 Life-long learning**	Thesis Book	√
			Performance by research	√
			Presentation and soft skill	√

Student Name /ID:

1. Rahin Intesar Nafim / 200012150

2. Nazia Tasnim / 200012141

3. Sidratul Muntaha / 200012124

Signature of the Supervisor: ……

Name of the Supervisor: Dr. A.R.M. Harunur Rashid

K-P-A Mapping of ME 4800 -Theis and Project

COs	POs	Related Ks								Related Ps							Related As				
		K1	K2	K3	K4	K5	K6	K7	K8	P1	P2	P3	P4	P5	P6	P7	A1	A2	A3	A4	A5
CO1	PO2	√	√	√	√					√											
CO2	PO3					√				√	√	√				√	√	√	√		
CO3	PO4								√	√	√	√					√	√			
CO4	PO5						√			√				√							
CO5	PO8							√													
CO6	PO9																				
CO7	PO10																				
CO8	PO12																				

Student Name /ID:

1. Rahin Intesar Nafim / 200012150

2. Nazia Tasnim / 200012141

3. Sidratul Muntaha / 200012124

Signature of the Supervisor: ……

Name of the Supervisor: Dr. A.R.M. Harunur Rashid

List of Sustainable Development Goals (SDGs) Addressed in this Project

SDG No.	Goals	Targets	Relevance to the Thesis (put √ if valid)	Remarks
1	No Poverty	1.1 Eradicate extreme poverty (people living on less than $1.25/day).		
		1.2 Reduce poverty in all its forms by at least half.		
		1.3 Implement nationally appropriate social protection systems.	√	
		1.4 Ensure equal rights to economic resources, services, property, inheritance, technology, and financial services.		
		1.5 Build resilience of the poor and reduce exposure to climate-related and other shocks.		
		1.a Mobilize resources to end poverty.		
		1.b Create pro-poor policy frameworks.		
2	Zero Hunger	2.1 End hunger and ensure access to safe, nutritious food year-round.		
		2.2 End all forms of malnutrition.		
		2.3 Double agricultural productivity and incomes of small-scale producers.		
		2.4 Ensure sustainable food production systems and resilient agricultural practices.		
		2.5 Maintain genetic diversity of seeds, plants, and animals.		
		2.a Increase investment in rural infrastructure, research, and technology.		
		2.b Correct and prevent trade restrictions/distortions in global food markets.		
		2.c Adopt measures to ensure proper functioning of food commodity markets.		
3	Good Health and Well-Being	3.1 Reduce global maternal mortality ratio.		
		3.2 End preventable deaths of newborns and under-5 children.		
		3.3 End epidemics of AIDS, tuberculosis, malaria, and neglected tropical diseases.		
		3.4 Reduce premature mortality from NCDs and promote mental health.		
		3.5 Strengthen prevention and treatment of substance abuse.		
		3.6 Halve global deaths/injuries from road traffic accidents.		
		3.7 Ensure universal access to sexual and		

		reproductive healthcare.		
		3.8 Achieve universal health coverage.		
		3.9 Reduce deaths from hazardous chemicals, pollution, and contamination.		
		3.a Strengthen tobacco control (WHO FCTC).		
		3.b Support R&D of vaccines and medicines.		
		3.c Increase health financing and workforce.		
		3.d Strengthen capacity for early warning and risk management.	√	
4	Quality Education	4.1 Ensure all complete free, equitable, quality primary and secondary education.		
		4.2 Ensure access to quality early childhood development and pre-primary education.		
		4.3 Ensure equal access to affordable technical, vocational, and higher education.		
		4.4 Increase skills for employment and entrepreneurship.		
		4.5 Eliminate gender disparities in education.		
		4.6 Ensure literacy and numeracy for youth and adults.		
		4.7 Ensure learners acquire knowledge/skills for sustainable development.		
		4.a Build and upgrade education facilities that are inclusive and safe.		
		4.b Expand scholarships for developing countries.		
		4.c Increase supply of qualified teachers.		
5	Gender Equality	5.1 End all forms of discrimination against women and girls.		
		5.2 Eliminate violence against women and girls.		
		5.3 Eliminate harmful practices (child, early, forced marriage, FGM).		
		5.4 Recognize and value unpaid care and domestic work.		
		5.5 Ensure women's participation in leadership and decision-making.		
		5.6 Ensure universal access to reproductive health and rights.		
		5.a Undertake reforms to give women equal rights to resources.		
		5.b Enhance use of enabling technology to empower women.		

		5.c Adopt and strengthen policies and laws for gender equality.		
6	Clean Water and Sanitation	6.1 Achieve universal and equitable access to safe drinking water.		
		6.2 Achieve access to adequate sanitation and hygiene.		
		6.3 Improve water quality by reducing pollution.		
		6.4 Increase water-use efficiency and sustainable withdrawals.		
		6.5 Implement integrated water resources management.		
		6.6 Protect and restore water-related ecosystems.		
		6.a Expand international cooperation in water and sanitation.		
		6.b Support participation of local communities.		
7	Affordable and Clean Energy	7.1 Ensure universal access to affordable, reliable, modern energy services.		
		7.2 Increase substantially the share of renewable energy.	√	
		7.3 Double global rate of improvement in energy efficiency.	√	
		7.a Enhance international cooperation on clean energy research/technology.		
		7.b Expand infrastructure and upgrade technology for sustainable energy.		
8	Decent Work and Economic Growth	8.1 Sustain per capita economic growth.		
		8.2 Achieve higher levels of productivity through diversification, tech, and innovation.	√	
		8.3 Promote policies for decent job creation and entrepreneurship.		
		8.4 Improve resource efficiency in production and consumption.	√	
		8.5 Achieve full and productive employment for all.	√	
		8.6 Substantially reduce youth not in employment/education/training.		
		8.7 Eradicate forced labor, modern slavery, and child labor.	√	
		8.8 Protect labor rights and safe working environments.		
		8.9 Promote sustainable tourism.		
		8.a Increase aid for trade support.		
		8.b Develop a global youth employment strategy.		

9	Industry, Innovation, and Infrastructure	9.1 Develop quality, reliable, sustainable infrastructure.		
		9.2 Promote inclusive and sustainable industrialization.		
		9.3 Increase access of SMEs to financial services and integration into value chains.	√	
		9.4 Upgrade infrastructure for sustainability and resource efficiency.		
		9.5 Enhance scientific research and technology development.	√	
		9.a Facilitate sustainable infrastructure in developing countries.		
		9.b Support domestic tech development and value addition.	√	
		9.c Increase access to ICT and internet.	√	
10	Reduced Inequalities	10.1 Achieve income growth of bottom 40%.		
		10.2 Empower and promote inclusion regardless of status.		
		10.3 Ensure equal opportunity and reduce inequalities of outcome.		
		10.4 Adopt policies for fiscal, wage, and social protection equality.		
		10.5 Improve regulation of global financial markets.		
		10.6 Ensure enhanced representation in global institutions.		
		10.7 Facilitate safe, regular, and responsible migration.		
		10.a Implement special treatment for developing countries.		
		10.b Encourage development assistance and investment in least developed areas.		
		10.c Reduce remittance costs.		
11	Sustainable Cities and Communities	11.1 Ensure access to adequate, safe, and affordable housing.		
		11.2 Provide sustainable transport systems.		
		11.3 Enhance inclusive urbanization and capacity for planning.		
		11.4 Protect cultural and natural heritage.		
		11.5 Reduce disaster impact and losses.		
		11.6 Reduce environmental impact of cities (air quality, waste).	√	
		11.7 Provide access to safe, inclusive green/public spaces.		
		11.a Support positive links between urban, peri-urban, rural.		
		11.b Increase disaster risk reduction		

		strategies.		
		11.c Support least developed countries in sustainable building.		
12	Responsible Consumption and Production	12.1 Implement 10-Year Framework on sustainable consumption/production.		
		12.2 Achieve sustainable management and use of resources.	√	
		12.3 Halve per capita global food waste.		
		12.4 Manage chemicals and waste sustainably.		
		12.5 Substantially reduce waste generation.	√	
		12.6 Encourage companies to adopt sustainable practices.		
		12.7 Promote sustainable public procurement.		
		12.8 Ensure people have relevant information for sustainable development.		
		12.a Support developing countries' scientific and technological capacity.		
		12.b Develop tools to monitor sustainable tourism impacts.		
		12.c Rationalize inefficient fossil-fuel subsidies.		
13	Climate Action	13.1 Strengthen resilience and adaptive capacity to climate-related hazards.		
		13.2 Integrate climate measures into national policies.		
		13.3 Improve education and awareness on climate change.		
		13.a Implement UNFCCC commitments (mobilize $100 billion annually).		
		13.b Promote mechanisms for capacity-building in least developed countries.		
14	Life Below Water	14.1 Reduce marine pollution.		
		14.2 Sustainably manage and protect marine ecosystems.		
		14.3 Minimize and address ocean acidification.		
		14.4 Regulate harvesting and end overfishing.		
		14.5 Conserve at least 10% of coastal and marine areas.		
		14.6 Prohibit harmful fisheries subsidies.		
		14.7 Increase economic benefits from sustainable marine resources.		
		14.a Increase scientific knowledge and marine technology transfer.		
		14.b Provide access for small-scale		

		artisanal fishers.		
		14.c Implement international law for oceans.		
15	Life on Land	15.1 Conserve terrestrial and freshwater ecosystems.		
		15.2 Promote sustainable management of forests.		
		15.3 Combat desertification and restore degraded land.		
		15.4 Ensure conservation of mountain ecosystems.		
		15.5 Take urgent action to reduce biodiversity loss.		
		15.6 Promote fair benefit-sharing from genetic resources.		
		15.7 End poaching and trafficking of protected species.		
		15.8 Prevent introduction of invasive alien species.		
		15.9 Integrate ecosystem values into policies/planning.		
		15.a Mobilize resources for biodiversity.		
		15.b Finance sustainable forest management.		
		15.c Support local communities for forest and wildlife.		
16	Peace, Justice and Strong Institutions	16.1 Reduce violence and related death rates.		
		16.2 End abuse, trafficking, and violence against children.		
		16.3 Promote rules of law and equal access to justice.		
		16.4 Reduce illicit financial/arms flows, organized crime.		
		16.5 Reduce corruption and bribery.		
		16.6 Develop effective, accountable institutions.		
		16.7 Ensure inclusive, participatory decision-making.		
		16.8 Broaden participation of developing countries in global governance.		
		16.9 Provide legal identity for all (including birth registration).		
		16.10 Ensure access to information and protect freedoms.	√	
		16.a Strengthen national institutions for prevention of violence.		
		16.b Promote/enforce non-discriminatory laws and policies.		

17	Partnerships for the Goals	17.1 Strengthen domestic resource mobilization.		
		17.2 Developed countries to implement ODA commitments.		
		17.3 Mobilize additional financial resources.		
		17.4 Assist developing countries with debt sustainability.		
		17.5 Invest in least developed countries.		
		17.6 Enhance access to science, technology, innovation.		
		17.7 Promote environmentally sound technologies.		
		17.8 Fully operationalize technology bank for LDCs.		
		17.9 Enhance international support for capacity-building.		
		17.10 Promote a universal, rules-based trading system (WTO).		
		17.11 Increase exports of developing countries.		
		17.12 Timely implementation of duty-free, quota-free market access.	√	
		17.13 Enhance global macroeconomic stability.		
		17.14 Enhance policy coherence for sustainable development.		
		17.15 Respect national policy space.		
		17.16 Enhance global partnerships.	√	
		17.17 Encourage multi-stakeholder partnerships.	√	
		17.18 Enhance data capacity of developing countries.		
		17.19 Support capacity-building for sustainable development indicators.		

Acknowledgment

Firstly, we would like to thank the Almighty Allah for giving us the courage and strength to progress in our research.

We would also like to express our deepest gratitude to our supervisor **Dr. A.R.M Harunur Rashid** for providing guidance and feedback throughout this research journey. We are extremely grateful for his support towards us.

We are also grateful to **Urmee Garments** for providing us with the opportunity to observe their manufacturing line and collect data for our study, including production capacity, SMV, and cycle time. Finally, we want to extend our thanks to **Snowtex Outerwear Ltd** for giving us the chance of touring their factory and getting a more practical understanding of the operations and production line of an RMG plant.

Abstract

The RMG and textile apparel industry are key contributors to a country's economic growth. Bangladesh ranks second among the top textile-exporting countries. Despite this achievement, the supply chain of an RMG faces various obstacles, including increased lead time, excess inventory, transportation, and maintenance costs. Additionally, there are also the problems of the inefficient utilization of resources and overworking of manpower. Moreover, even a small percentage of inefficiency can cause a loss of thousands of dollars. This study aims to create a digital twin (DT) of a real RMG plant and optimize production parameters such as throughput, lead time, bottleneck, and resource utilization. At first, we collected data from a plant and used this data to create the simulation model using a Discrete Event Simulation (DES) software. After validation of the simulated model against real factory data, we used the Multi-Objective Genetic Algorithm (MOGA) to optimize the whole production line. Through our experiment, we found that a bottleneck in an operation was causing the throughput to decrease. After the implementation of MOGA and task-clustering method, we saw a significant productivity improvement. Thus, we can conclude that by the use of advanced technologies like DES and other optimization algorithms, the performance and efficiency of a plant can be increased to match the ever-changing world market.

Summary

(for non-technical audience)

The top three textile exporting countries are China, Bangladesh, and Vietnam, respectively. But the difference in earnings between China and Bangladesh is huge. As reported in 2024, China earned $301 billion from exporting textiles. On the other hand, Bangladesh had earnings of $38.48 billion. This variation in earnings can be attributed to the use of advanced technologies and low production costs in China. In contrast, Bangladesh mainly uses traditional approaches like manual record-keeping, line balancing, and scheduling. Even most factories do not have their own Enterprise Resource Planning (ERP) software. We are conducting this study to supports the development of our RMG sectors and attract potential customers. The main goal of our thesis is to optimize production parameters like lead time, Work in Process (WIP), reduce bottlenecks, and increase output. Here in this study, we are replicating a real plant model. That is, the digital simulation model would exactly replicate the real plant. For this purpose, we are using Discrete Event Simulation (DES). As DES simulates the whole 16-hour production line and managers can visually see which operations need correction at any point in time. By using DES, the owners can also test various strategies without wasting time and resources. The second part of our study is the optimization of production parameters. After the identification of the problem parameters, we have used the Multi-Objective Genetic Algorithm (MOGA). The purpose of multi-objective optimization is to find a trade-off between conflicting parameters, for example, cost vs time. While Genetic Algorithm (GA) mimics the evolutionarily process to find the best offspring. That is to say, the optimal solution space where all our objectives are satisfied as best as possible.

The key findings of our study show that after the application of DES and MOGA methods, the production output has increased from 1487 to 1814 parts, thereby reducing the cycle time from 31.47 seconds to 25.8 seconds.

Table of Contents

List of Figures

List of Tables

Nomenclatures and Symbol

Abbreviation	Meaning
DES	Discrete Event Simulation
GA	Genetic Algorithm
MOGA	Multi-Objective Genetic Algorithm
DT	Digital Twin
KPI	Key Performance Indicators
RMG	Ready Made Garments
SCM	Supply Chain Management
MES	Manufacturing execution Systems

Chapter 1: Introduction

1.1 Objectives of the Study

The main aim of this study is to increase the operational performance of an RMG or a textile plant through the use of **Discrete Event Simulation (DES)** and **multi-objective optimization** methods. The study is motivated by two primary objectives:

1. Validation of the Simulation Model

The first objective is to validate the simulation models developed for the textile manufacturing process. This step ensuring that the simulation accurately represents the real-world system. The validation process will focus on:

- Comparing simulated outputs (such as throughput, lead time, and work-in-progress) with real historical production data from the plant. The digital twin will try and recreate output vs historical demand and some other parameters to validate the model.
- Ensuring that the simulation models are statistically reliable and reflect real-life operations within the plant, with a focus on achieving a **95% confidence level** between the simulated and actual data.

- Evaluating the accuracy of both the **Tecnomatix™** and **FlexSim™** models to determine their applicability and reliability for further optimization tasks. This ensures that both platforms are viable options for anyone looking to use Discrete Event simulation software packages.

This validation will serve as the foundation for using the simulation to make informed decisions and recommendations in the optimization phase of the study.

2. Optimization of Plant Operations

The second objective is to optimize the operations of the plant using the validated simulation models. Optimization will be conducted by applying **Genetic Algorithms (GA)** to improve key performance indicators (KPIs) such as:

- **Throughput**: This term refers to maximizing the number of products produced per unit of time.
- **Work-In-Progress (WIP)**: WIP means reducing the amount of work that is in progress at any given time, thus improving system flow.

- **Resource Utilization**: Utilization means that the resources are being used to the best of their capabilities. This ensures efficient use of machines and labor, optimizing the allocation of resources. And increased the availability of a machine.
- **Lead Time**: Lead time is an essential factor in any industry. This means minimizing the total time taken for an item to pass through the production system.

The optimization process will use **multi-objective genetic algorithms (MOGA)** to balance these KPIs and identify **optimal production parameters**. The purpose of using multi-objective is so that we can make a balance between conflicting parameters. Such as- increasing the production capacity while decreasing the bottleneck.

1.2 Structure of the Thesis

The RMG and textile sectors remain the leading contributors to Bangladesh's export earnings. According to the Bangladesh Garments Manufacturers and Exporters Association (BGMEA), earnings increased by 7.34% in 2024, reaching a revenue of $38.48 billion. This is an admirable outcome showing the resilience of our supply chain despite political and economic unrest that was happening in the country. Despite this, there are various scopes for further development with the integration of Artificial Intelligence (AI) and Machine Learning (ML). As the present RMG and textile sectors of our country focus more on traditional approaches, this is evident in some recent papers. This paper [1] studied the impact of COVID-19 on the supply chain of textiles, apparel, and fashion manufacturing sectors. Moreover, the paper suggested the development of a data-driven and technology-based supply chain for future unforeseen situations. Another study by [2] tried to understand the relation between suppliers and buyers and depended on qualitative and empirical studies, like a questionnaire survey of 66 clothing brands and human intuition, to draw up answers. Furthermore, the digital infrastructure of the surveyed factories was not developed enough. Thus, it can be noted that most Bangladeshi suppliers depend on manual and note-taking traditional approaches to collect and store data.

For most RMG plants in Bangladesh and many other countries, the integration of advanced technologies has not yet been adopted. In the supply chain management (SCM) of an RMG and textile, the employees have to focus on stages starting from the collection of raw materials to the cutting, sewing, and finishing stages. All these stages contain thousands and

thousands of lay that are the uncut fabric along with the different seams and cuts needed for a complete end product. Simply, it can be stressful to manage such a complex and interconnected process using only traditional methods when there is now the opportunity to implement technological methods to ease the production. There is no doubt that traditional methods like line balancing and scheduling greatly help to minimize production lead time and reduce bottlenecks. For instance, in this study, [3] they focused on improving the productivity of a sewing line by the implementation of line balancing. By using the line balancing technique, the study showed an improvement of efficiency to 81.78% while also reducing idle cycle time. Though they were able to receive good results, the collection method was time-consuming because of using a stopwatch. Another study[4] researched the use of hybrid line balancing in combination with Estimation of Distribution Algorithm (EDA). Here, they were able to showcase better results for multiple production lines, but a drawback is that their model was based on deterministic parameters, which means the model would not perform well under uncertain or variable conditions. One of the main characteristics of any market is that it is volatile and ever-changing, with variables that are not always visible and tangible.

The main aim and objective of our thesis is to optimize an RMG or a textile plant using a Multi-Objective Genetic Algorithm (MOGA) and Discrete Event Simulation (DES). To create a more dependable and flexible production system, these two technologies use AI and ML. Though these recent studies on these two methods are on the rise, they have not yet been implemented on a large scale. The main purpose of using DES is to replicate a plant model so that these digital twins can be used as a guinea pig for experimentation with different strategies. In this paper [5], they successfully studied the fusion of real-time synchronization of data with Internet of Things (IOT) and DES. The experiment was done by using an IMU sensor to collect data for the Digital Twin (DT), with the goal of creating a cyber-physical system. As such, in an RMG also we can use a digital twin to test various strategies without the need to expend and waste real-life resources and time. In addition, the MOGA is used to find a balance between two contradicting parameters. The use of MOGA and its various hybrid algorithms helps to model a system to find the optimal solution set in accordance with the customer's request. For this study by [6], their goal was to either reduce energy cost or energy consumption by developing a Genetic Algorithm (GA). Through the application of optimization techniques, they were able to demonstrate a more effective production time that is not only economical but also eco-friendly. And the study was even able to show a reduction

of €23.47. Furthermore, in this study [7], the authors used both DES and GA to create a simulation model using Tecnomatix™ to showcase how these techniques can help maximize daily factory production and minimize the WIP inventory. Here, through the use of GA and DES, they were able to reduce WIP inventory and even showed an increase in productivity by an average of 8.5% to 17%.

We want this study to have an impact on both academic and industrial settings. To encourage people to use different forms of simulation models and advanced methods to increase productivity and modify current processes to align with sustainable development goals and create profits for the plant. The research methodology followed a structural process. At first, we collected real factory production data from plants that we visited. Then, by the process of filtration and elimination, a dataset ranging from 28th July to 7th August was collected, spanning a total of 9 working days. Then, to make the digital twin, we used DES software. But first, a validation was done following the papers [8], [9] with the intention to see if the simulated results are similar to actual collected data. For the optimization part, various metaheuristic algorithms were considered; for instance, the linear programming (LP) method helps to find the best solution in a limited capacity. From this paper [10], where LP was used to solve an optimization problem, we can conclude that LP is best applicable for simplified problems with only one maximization or minimization of the objective function. Other optimization algorithms like Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Non-dominated Sorting Genetic Algorithm (NSGA) are also used in SCM aspects and manufacturing processes to reduce production time, increase the overall efficiency of a line or whole factory. These metaheuristics also help to estimate production capability based on stochastic (random probability distribution) data. The use of these methods builds the backbone of an agile and resilient supply chain.

Technological advancements and the proper implementation require appropriate digital infrastructure and computational capability. The paper [11] states that the Bangladesh RMG and apparel industry has a score of 1.9 on the scale of 5 for maturity in Industry 4.0. This can pose quite a challenge for any industry, but with proper and strategic implementation of the required infrastructure can help tackle this. Another obstacle could be that most SME in Bangladesh do not have proper documentation of daily production data. Furthermore, to

exactly replicate dynamic situations can be difficult and thus assumptions were made during the formulation of GA model.

The thesis is organized in an orderly manner with each chapter giving insights into the process and how the information is presented. Chapter 2 presents the literature review, summarizing and synthesizing past literature and how these papers helped draw our conclusions. And Chapter 3 details the process about how we tested the validity of the software and how data collected was used to create a digital twin. Chapter 4 discuss the results and analyses the use of GA in two-task or three-task clustering to show improvement. Lastly, Chapter 5 discuss how the implantation of our methods can help in better performance and reduce idle time. The Chapter 6 details scope for future works and recommendations.

Chapter 2: Literature Review

Bangladesh's economy heavily relies on its Ready-Made Garment (RMG) sector. This sector pursues great efficiency and innovation due to fierce international competition [12]. In such competitive landscape, a well-designed supply chain is essential for success [12]. While the broader textile industry like the ones in Brazil is also challenged by rapid changes in customer demand, fashion trend and seasonality, the severity of the challenges in Bangladesh is far beyond [8], [12]. The adoption of technology is a proven way to reduce expenses and increase revenue. However, Bangladesh RMG industry has been hesitant to adopt and embrace these technological advancements [12]. This reluctance to adopt these technologies are often caused by the limitation of the infrastructure, lack of a skilled workforce and general resistance to change [12]. Nevertheless, the industry is showing an increase in the awareness and gradual implementation of these digital solutions within the industry [12]. To improve decision making and enhance agility of production, computational tools have significant value [8]. Different modern technologies such as Artificial Intelligence (AI), the Internet of Things (IoT), blockchain and Enterprise Resource Planning (ERP) are overcoming the inefficiencies and transparency concerns of the past [12]. These modern technologies can improve both supply chain's visibility and agility. Even the responsiveness to fast market changes is better with the support of these technologies [12]. Furthermore, transformation from static manufacturing activities into dynamic ones can also be done easily with the help of these modern approaches. This helps us analyze the real-world situations with more real-world constraints which is the main purpose of using these techniques. With the combination of different management tools like ERP and Manufacturing Execution Systems (MES) with modern technologies this transformation is done [8]. An example of the application of these tools are in Discrete Event Simulation (DES) with genetic algorithms to refine existing production schedules to find an optimal solution [8][13].

A supply chain can be defined as a network of participants, including suppliers, manufacturers and distributors who work together to convert all sorts of raw materials into finished goods that are aimed to be delivered to the consumers [14]. In this competitive era, companies need to satisfy the rapid increasing customer demands while keeping the total operating costs as low as possible [15]. These constrains add more complexity in the management of a supply chain. The complexity and constant changing behavior make it difficult to manage supply chain [16]. Traditionally supply chains were managed based on

one's experience or expertise. Back then the level of complexity and dynamic behavior of supply chains were far less compared to this day. The modern supply chain and production systems are characterized by an increasing level of complexity and stochasticity [16]. An operation becomes difficult to predict and control while dealing with inconsistent factors like changing customer demand, unplanned machine failures inconsistent processing times and fluctuating lead times [7], [8]. Due to this inherent uncertainty and complexity, traditional mathematical models like linear programming or queuing theory are either too simple or too difficult to solve for designing and analyzing these systems effectively [17]. The assumptions that are needed to make these models mathematically tractable often lack the critical, real-life details. This lack of real-world constraint affects the quality of the solution directly. This causes a major difference between the theoretical answer and what works is practical [14].

The inadequacy of entirely analytical methods has made Discrete Event Simulation (DES) the leading method for modeling these kinds of complicated systems [16]. DES is a methodology that models the operation of a system as a sequence of discrete events occurring over time [16]. It is a flexible and potent tool that is capable of capturing intricate logic, resource constraints and stochastic variability found in manufacturing, logistic and supply chain operations [17]. Extensive research shows that DES is well developed, widely used tool for both analyzing existing systems and designing new ones, providing a virtual environment to test hypothesis and evaluate strategies without disrupting real-world operations [16], [17].

However, creating a high-fidelity model is only the first step. The subsequent challenge is one of optimization: identifying the single best possible setup or plan from a massive and practically endless number of possibilities. Common challenges like scheduling, facility layout or supply chain layout network design are well known to be "NP-hard". This means the computing power needed to find the optimal solution using exact methods grows exponentially with the size of the problem [14], [18]. For any realistic industrial problem, it is impossible to check every single option or using exact algorithms are simply not feasible [14], [18].

This computing limitation has led to the widespread use of metaheuristic search techniques, which aim to find near-optimal solutions as quickly as possible [18]. Among these, Genetic Algorithms (GA) are known to be especially reliable and efficient option [17]. A Genetic Algorithm is a problem-solving method based on the theory of natural selection. It uses the

mechani8sm like selection, crossover and mutation to gradually improve a set of potential answers [18]. A practical study of four algorithms (Hooke-Jeeves pattern search, Nelder-Mead simplex, simulated annealing and genetic algorithm) conducted on twenty-five (25) industrial problems showed that the GA was the most robust, consistently finding near- best solutions across all type of problems [17]. This research proves that choosing GA is not arbitrary rather is a logical, data-driven decision for handling the complex, multi-variable optimization challenges common in industrial engineering [17]. However, this robustness has a major drawback. the same study also found that the GA required most replications of all algorithms, highlighting a key trade-off between the quality of the answer and the computational cost [17]. It is now a central issue in the field of simulation-optimization [17].

Methodological Framework for Optimization through Simulation

Combining Discrete event simulation (DES) with Genetic Algorithm (GA) has created a potent methodology referred to as Simulation-Optimization or Sim-Opt [18]. This method leverages the strength of both methodologies. With the help of GA, solutions are generated through evolutionary search process and DES model analyzes those solutions [18]. The following discussion will explain the approaches and explore the evolution toward multi-objective optimization.

DES as a High- Fidelity Fitness Function

As a single, static function cannot represent the entire real-world scenario new innovation had to be made. The innovation of optimization through simulation is the replacement of a static, analytical objective function with a dynamic, high-fidelity simulation model [18]. Conventional optimization algorithms use mathematical formulas to calculate and determine the quality of a solution [18] Conversely, in the Sim-Opt framework, the discrete-event simulation (DES) model itself becomes the fitness function [18]. This shift allows for the optimization of complex, new features and interactions that are impossible to precisely define by a single mathematical equation [18]. This result is an optimization process which is grounded in a realistic, virtual representation of the physical system. This gives solutions that are far more accurate and directly applicable to real-world scenarios [18].

According to the literature, the process operates in the following manner:

1. Population Initialization: The Genetic Algorithm (GA) creates a starting set of potential solutions [8], [18]. In a scheduling problem, each “individual” or “chromosome” within this set stands for a distinct sequence of production steps [18].

2. Simulation and Evaluation: Each individual solution is passed to the DES model for testing. Using this input, the model executes one or more simulation runs based on this configuration and returns key performance indicators (KPIs) such as total throughput, average work-in-process (WIP) or machine utilization [18].
3. Fitness Calculation: The output KPIs from the simulation are then synthesized into a single fitness value for that solution. It measures the effectiveness in meeting the optimization goals [18].
4. Genetic Operations: Subsequently, the GA employs its core operations: selection, crossover and mutation to the fittest individuals of the current generation to create a new and presumably better, population of offspring [8], [18].
5. Termination: The iterative loop of evaluation and refinement is repeated until a specific condition is satisfied. This can be either reaching the number of generations or failing to find a better solution for a certain period [8], [18]. Finally, the best individual found during the entire process is then put forward as the near-optimal solution [18].

General Framework vs Integrated Software

The literature shows two primary methods for implementing Sim-Opt strategies. This reflects on the differences between theoretical research and applied industrial practice [8], [18].

1. The first approach involves creating adaptable and custom frameworks using fundamental programming and modeling software [18]. An example of this approach is creating scheduling model using MATLAB for the GA and its SimEvents toolbox for the discrete event simulation (DES) model [18]. This method gives us few advantages in solving the problem. It provides maximum flexibility, transparency in the algorithm's inner mechanics and the capacity to customize each part for exact research goal [18]. It is particularly suited for methodological research aimed at developing new algorithms. Particularly when goal is to create new algorithms or exploring theoretical foundations of the Sim-Opt process [18]. However, this path has some disadvantages as well. It requires significant expertise and development time [18].

2. The second method is the use of integrated commercial software solutions [7], [8], [9]. An example can be plant simulation tools like Tecnomatix™. These softwares

provide in-built components for not only DES modeling but also for optimization using GA. This helps getting the advantage of both methods. These are sometimes called “GAWizard” [7], [8], [9]. Compared to other methods this one reduces the initial difficulty significantly. This helps professionals in industry and academia to create and refine complex models through a visual interface with customized coding [8]. It requires less time for the development and experimentation of any model. This makes it more ideal for industrial case studies and practical problem solving [8]. The tradeoff of this method is that it can be less flexible as the included optimization algorithms often function as a “black box”. Thus, the internal processes are hidden. [8]. This limits the flexibility of these softwares. The widespread availability of these commercial software packages plays a crucial role in transferring technology from academia to industry [18].

Addressing Multi-Criteria Problems with Multi-Objective Optimization

Real-world industrial challenges are rarely characterized by a single and simple objective. It is more common to need to find a balance between multiple objectives that are often conflicting with one another. For example, a production planner might aim to maximize throughput while minimizing inventory holding costs or minimize operational cost while maximizing customer service level. This reality had driven the development from genetic algorithm which is a single objective algorithm to Multi-Objective Genetic Algorithm (MOGA) [7], [19]. MOGAs are designed to Identify not a single solution, but a set of non-dominated solutions. These are known as Pareto front [19],[20] Each point on Pareto front represents a different trade-off between the objectives. It means that no single objective can be improved without degrading another [19]. The literature provides clear examples of this method in action.

In a case study of a textile factory, MOGA was used to determine the optimal production schedule that balanced the conflicting objectives of maximizing the number of produced pieces and minimizing the total work-in-process (WIP) inventory [7]. For a more complex application, a simulation model for a municipal medical waste supply chain during COVID-19 pandemic was designed to balance minimizing total cost and minimizing the virus outbreak risk [19]. The Non-dominated Sorting Genetic Algorithm II (NSGA II) is a leading algorithm often used for such real-world problems and was successfully applied in the medical waste supply chain problem [19].

The literature has also focused on improving the GA itself to enhance performance [21]. A method called “Improved Adaptive Genetic Algorithm” was suggested to correct the flaws of traditional adaptive GAs. Here, crossover and mutation rates decrease as fitness increases [21]. This makes the solution stuck in a local optimum [21]. A fuzzy based Genetic Algorithm that incorporates fuzzy logic has been noted as a method applied by Bangladeshi manufacturers [15], [22].

The following table provides a summary of the Sim-Opt methods analyzed in the provided literature.

Table 1. Synthesis of Simulation-Optimization Approaches in the Literature

References	Problem Domain	Simulation Method	Optimization Algorithm	Optimization Objective(s)	Key Contribution / Implementation
Paper [17]	General Industrial (Buffer Sizing, Distribution, AS/RS, Job Shop)	DES	GA, Pattern Search, Simplex, Simulated Annealing	Minimize Time, Minimize Cost, Maximize Profit	Empirical comparison of search algorithms; established GA's robustness and high computational cost.
Paper [18]	Production Scheduling (Job-Shop)	DES	GA	Maximize Mill Utilization	Generalizable Sim-Opt framework developed in MATLAB/SimEvents.
Paper [8]	Production Planning (Textile Industry)	DES	GA	Minimize Overtime	Case study using Tecnomatix Plant Simulation for single-objective optimization.
Paper	Production	DES	GA (Multi-	Maximize	Case study using

[7]	Planning (Textile Industry)		Objective)	Production Output / Minimize WIP	Tecnomatix Plant Simulation for multi-objective optimization.
Paper [9]	Warehouse Management	DES	Not specified (Optimization via simulation experiments)	Solve Forklift Blockages, Determine Optimal Forklift Number	Case study demonstrating DES for tactical warehouse optimization in Tecnomatix.
Paper [14]	Supply Chain Network Design	DES	Hybrid GA (NP-OCBA)	Minimize Total Cost	Development of an efficiency-focused algorithm to reduce the computational load of Sim-Opt.
Paper [19]	Supply Chain Network Design (Medical Waste)	System Dynamics & DES	MOGA (NSGA-II)	Minimize Total Cost / Minimize COVID-19 Risk	Multi-objective optimization of a strategic supply chain network with societal factors.

The concept of digital factory model offers as unified method for improving the process involved in production [9]. A digital factory can be defined as the comprehensive digital backing for the entire sequence of operations from initial development through to the end of the production using virtual methods [9]. That's why simulation is a key technology within this concept [9].

The digital factory is a general term that includes a complex network for digital models, tools and methods all aimed to achieve integrated planning within a company's operation [9]. This idea further develops into the idea of a virtual factory centered on a digital twin which is the

exact replica of the actual production system of the factory [9]. A digital twin based virtual factory can support the entire factory lifecycle process [9]. It can be used to optimize the factory model by optimizing various systems like management of warehouse [9].

In a case study digital twin is used to analyze the model and solve mutual blockages of forklift trucks and to determine the optimal number of forklifts required [9]. The evolution of a simulation model is to transform it into a production system's digital shadow. It is done by integrating existing ERP and Manufacturing execution Systems (MES) to ensure real-time data considerations [7]. Through this integration, the model becomes a real-time dynamic operational planning tool [7]. Such a system can perform not only daily but also real-time production scheduling and optimization [7]. In a more complex application, a system dynamic simulation framework can be used to predict metrics such as the volume of waste generated during pandemic [19]. This method of using simulation provides essential data that feeds into strategic optimization models, such as designing the municipal medical waste supply chain network [19]. As mentioned previously these digital twins are optimized through genetic algorithm or multi-objective genetic algorithm. That's why the plant simulation softwares are used where a production plant be optimized through analyzing multiple objectives.

By reviewing all the papers, we can summarize the following key points:

1. In the context of Bangladesh RMG and textile, DES and GA have been rarely explored which indicates that there is scope for further research in this area.
2. The lack of use of real production data from factory and industry.
3. Lack of exploring of other algorithms like PSO (Particle Swarm Optimization), WOA (Whale Optimization Algorithm) or hybrid algorithms.
4. The modelling of plant using in-built optimization software, here we can explore more opportunities if we can input our own algorithm using Python.
5. Dependency on Tecnomatix^{TM} software this begets the question if the same model can be created and simulated in other software.

Chapter 3: Methodology and Case Study

3.1 Workflow

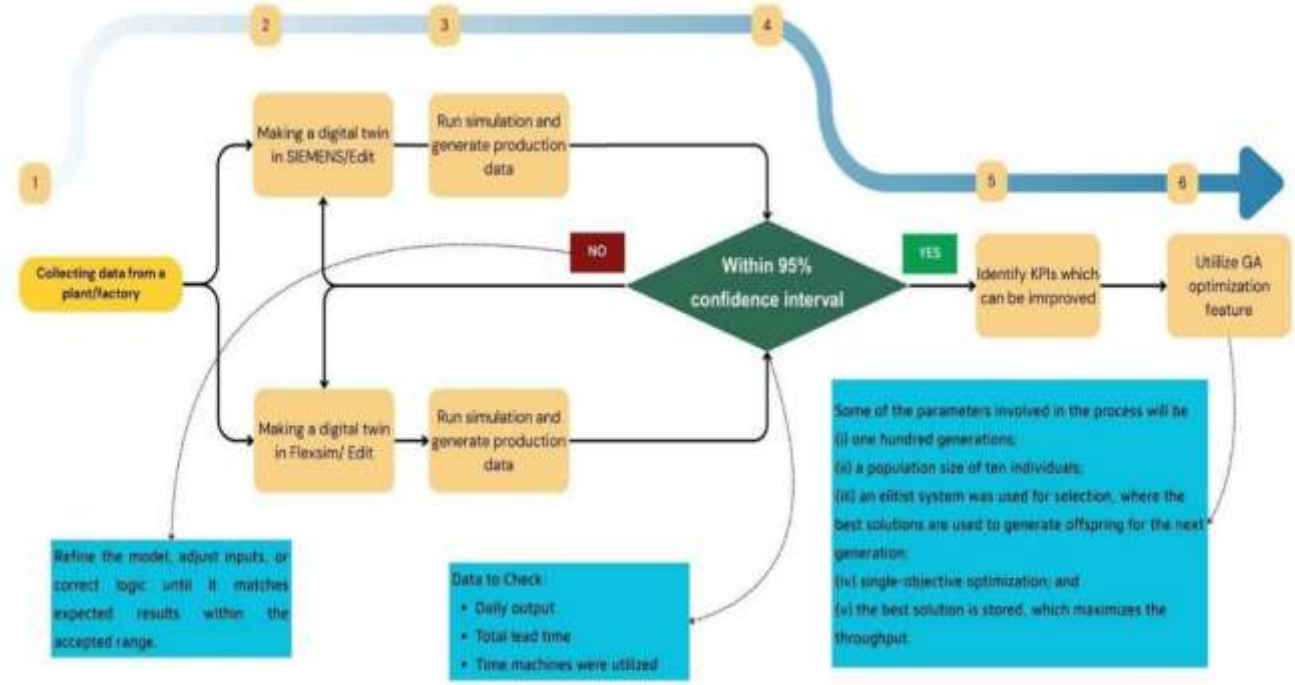

Figure 1. Workflow of Methodology

The primary step is to collect real data from any existing RMG or Textile plant. Based on this data, we will create two separate digital twins of the plant in both Tecnomatix™ and FlexSim™. We will also take different other data from the plant. Those will include:

1. Process flow
2. Operator Allocation
3. Downtime Records
4. Cycle Times

We will set these conditions to the digital model to replicate the actual plant in both the software. After that we will run the simulation for a fixed work hour (e.g. 8-hour shift for 3 months). We will set the key performance metrics as our performance measures in the simulation. Those are:

1. Lead time
2. Operator efficiency
3. Machine Efficiency
4. Throughput

The results from both the models will then be compared with the actual data taken from the plants. The main target is to achieve results that fall within the 95% confidence interval. If any of the models fails to reach the desired interval then the model or models need to be refined with new logic or input. The simulation will be rerun with the new logic or input in order to check the achieved value. When both the models meet the threshold, we will move onto the optimization phase.

We will identify the KPIs that can be improved to get a better result. Any metaheuristic approach such as genetic algorithm will then be used to optimize the process in order to reduce the lead time and increase efficiency of the system.

3.2 Model Development

We developed a simple model both in FlexSim™ and Tecnomatix™. It is a simple layout of an RMG plant consisting only the main operational section with necessary process.

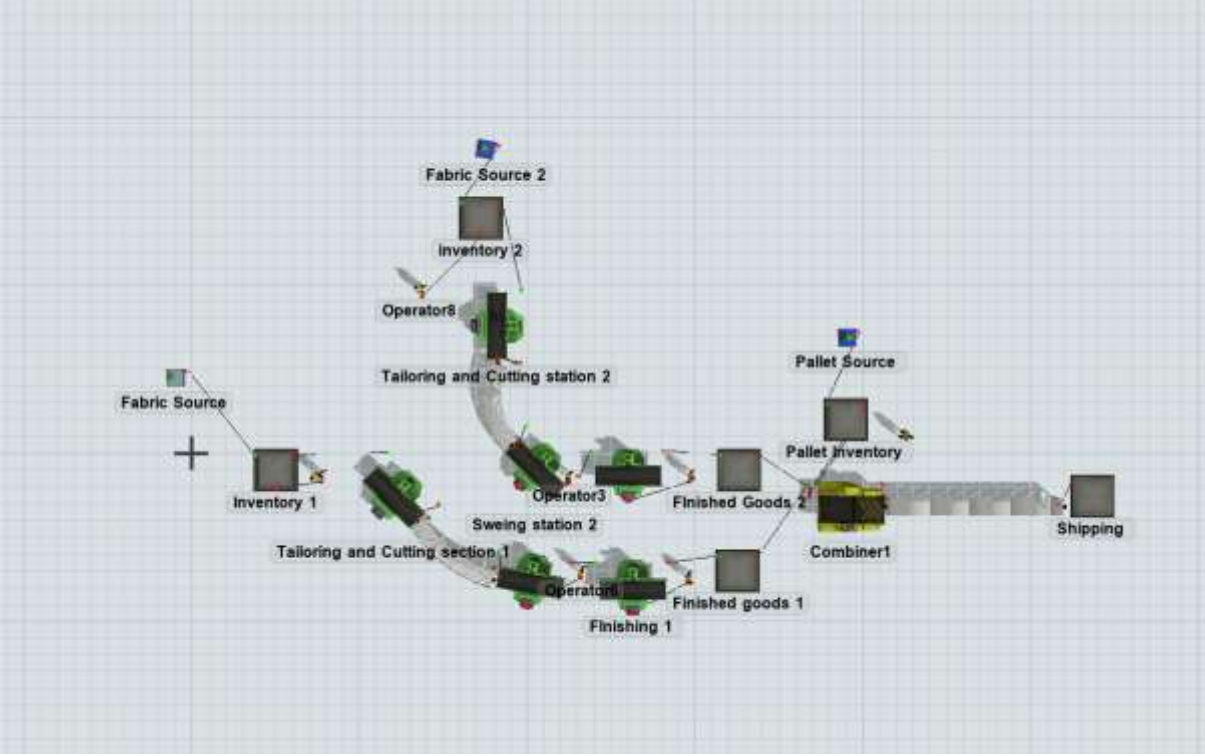

Figure 2. Model in FlexSim software

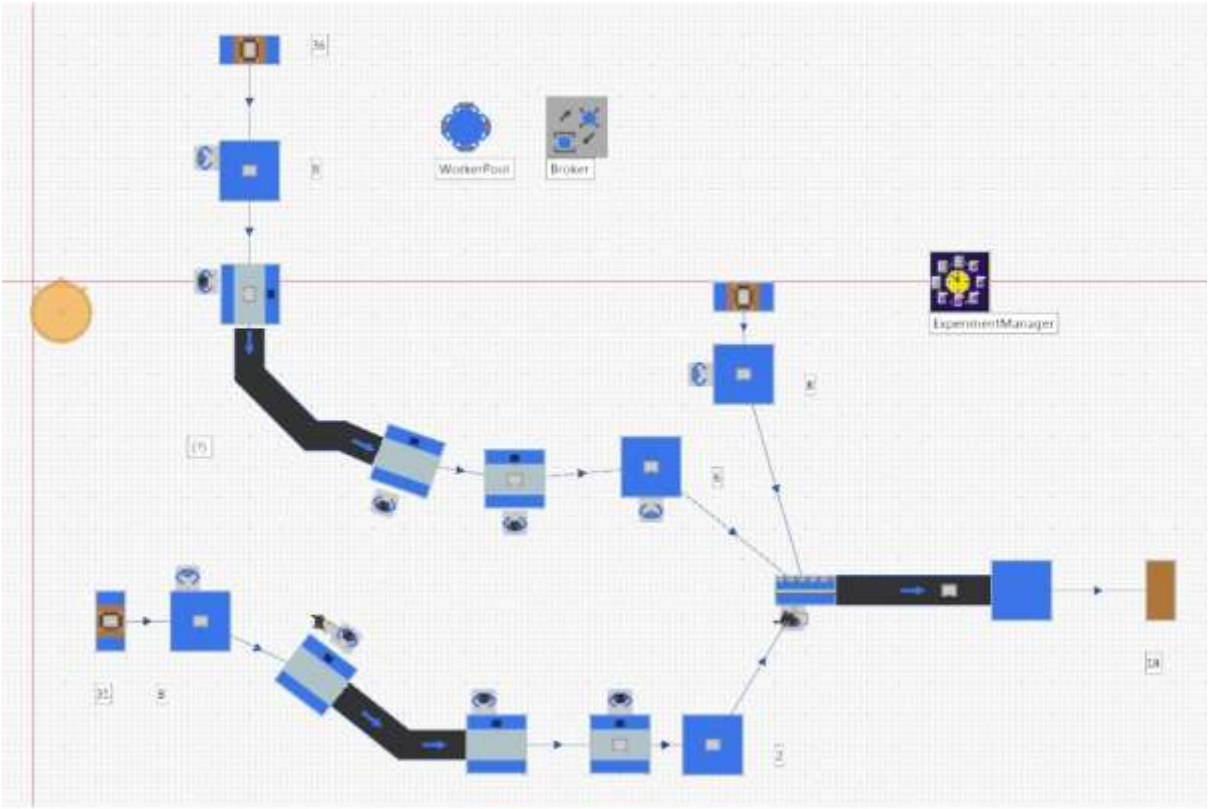

Figure 3. Model in Tecnomatix Software

For both models we have assumed the exact same data to get the exact same output. The values that we have taken here are mentioned below.

Table 2. Setting Parameters

Parameter	Value
Process Time	10 seconds
Buffer Capacity	8 units
Output Capacity	2000 units
Number of Operators	1
Run Time	8 hour-shift

3.3 Validation of the Software Models

After running the simulation under conditions mentioned in Table 3.1 we got very similar outputs from both the models.

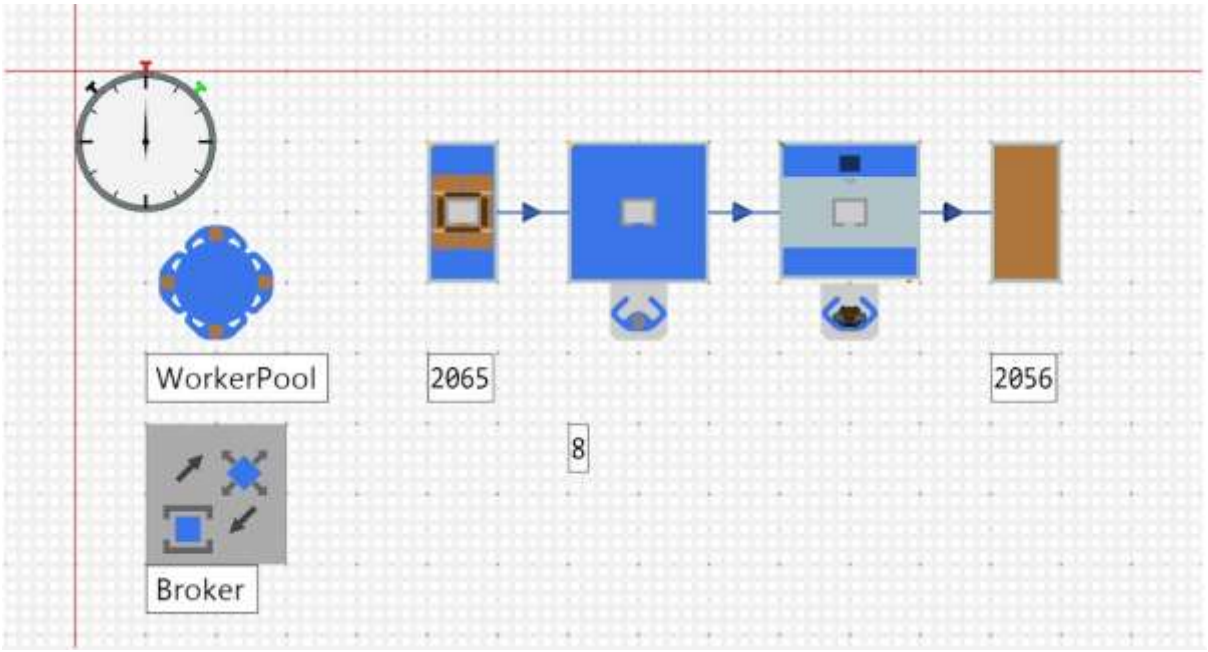

Figure 4. Output in Tecnomatix

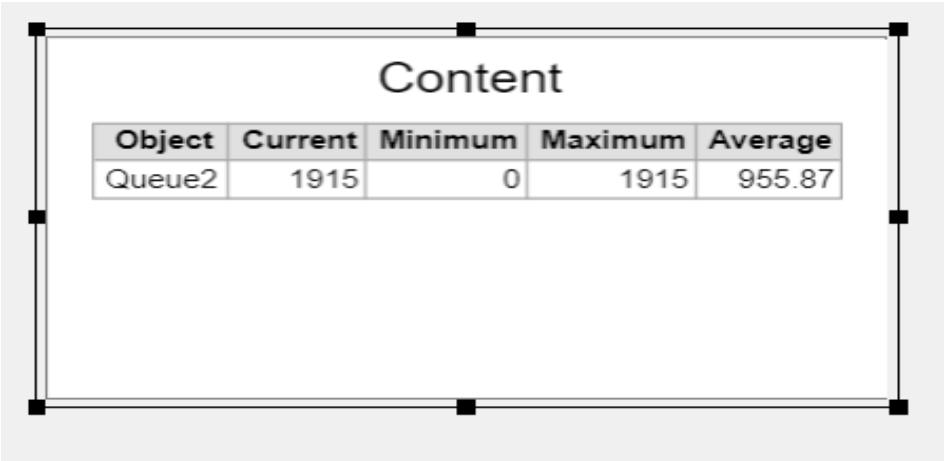

Figure 5. Output in FlexSim

As we can see that for an 8-hour shift FlexSim™ has an output of 1915 parts while Tecnomatix™ has 2056 parts. From the values of the output we understand that there is a slight difference which is **7.36%**. Although the range of difference is acceptable but to get precise and accurate data for optimizing a real plant it will be better as much as we can reduce the deviation of the outputs. There are a lot of micro level factors that needs to be fixed or looked into to reduce this difference.

3.3.1 Key Performance Measures Identification

During the simulation, when we withdrew the buffer capacity, we observed that, based on other conditions, the buffer size began to increase. To reduce buffer quantity, we tried to test the system to understand its strength and weaknesses.

We set performance measures for Tailoring and Cutting station utilization, Operator Utilization and Average queue content (content for buffer). We set the value for parameters up to 5 so that we can get 5 different scenarios to compare with each another. The scenarios represent like:

Table 3. Parameters in Different Scenarios

Parameter	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
Working Stations	1	2	3	4	5
Process Time (per unit)	10 sec	10 sec	10 sec	10 sec	10 sec
Buffer Capacity	8 units	8 units	8 units	8 units	8 units
Output Capacity	2000 units	2000 units	2000 units	2000 units	2000 units
Operators	1	1	1	1	1
Run Time	8 hours	8 hours	8 hours	8 hours	8 hours

We ran the simulation under twenty five (25) replication operation for each scenario. After the simulation is done for 8-hour shift for all of the scenarios we can view results from it. From the results of the simulation we can achieve graphs that represent the key performance measures that we have set before.

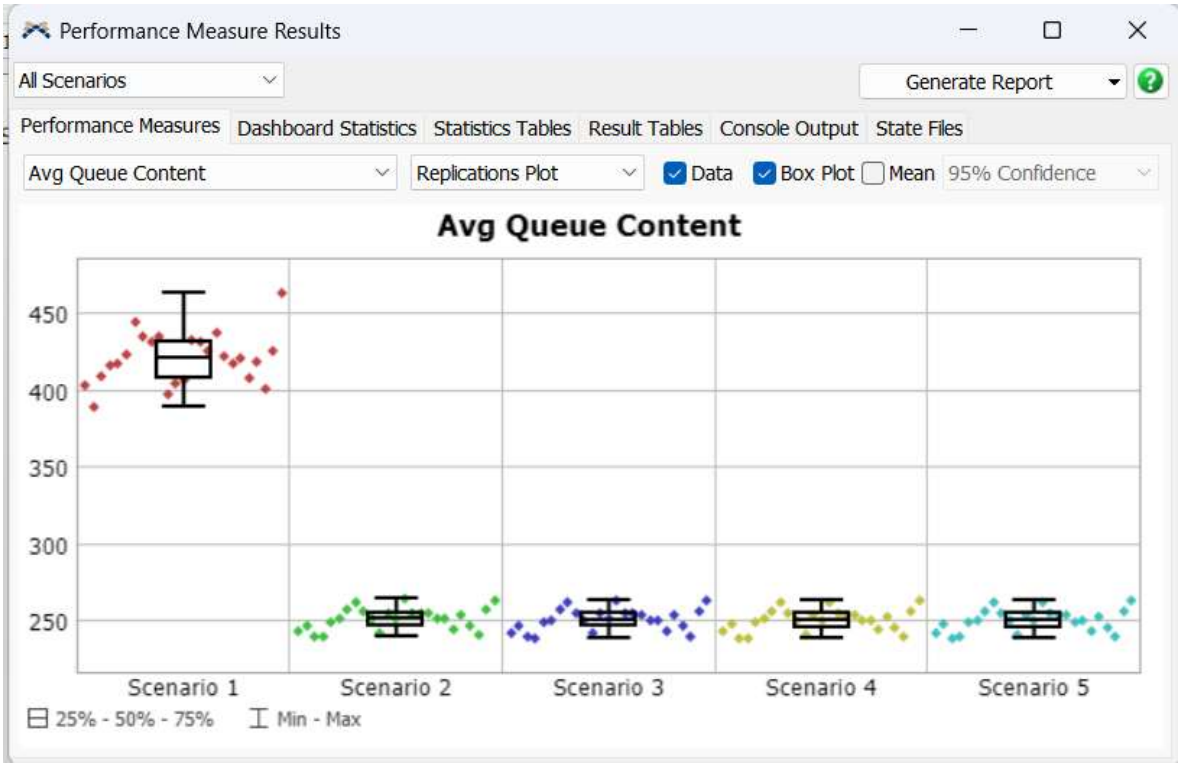

Figure 6. Average Queue Content (Buffer)

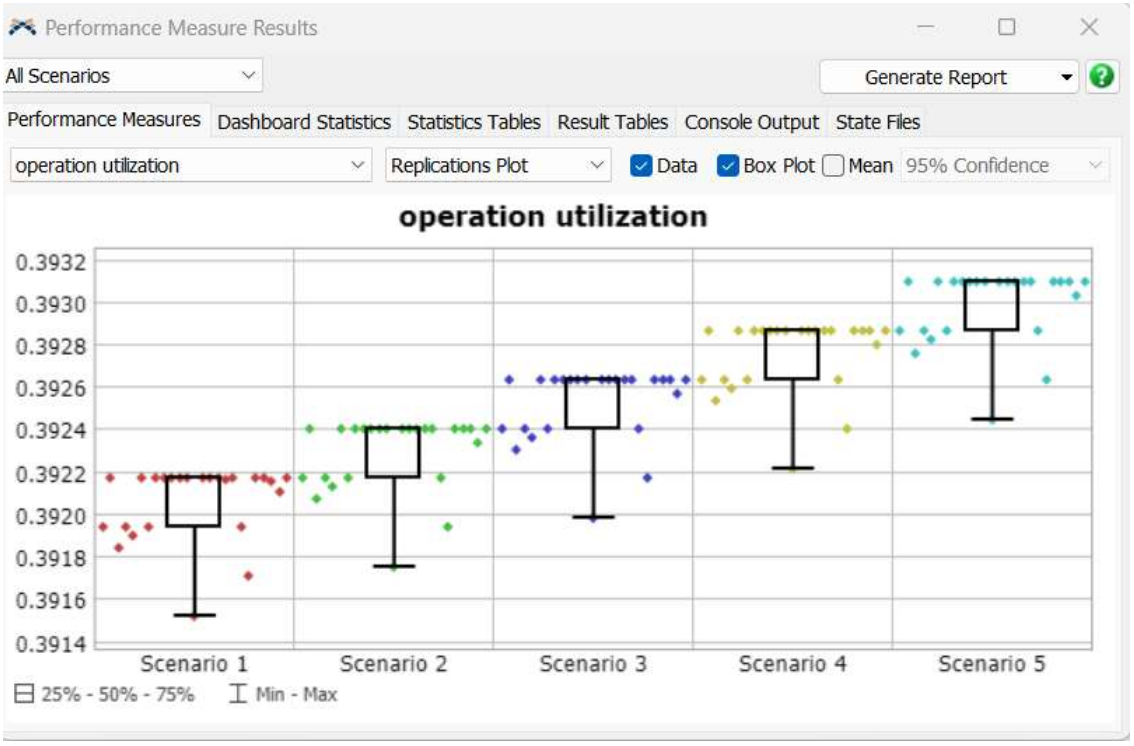

Figure 7. Operator Utilization

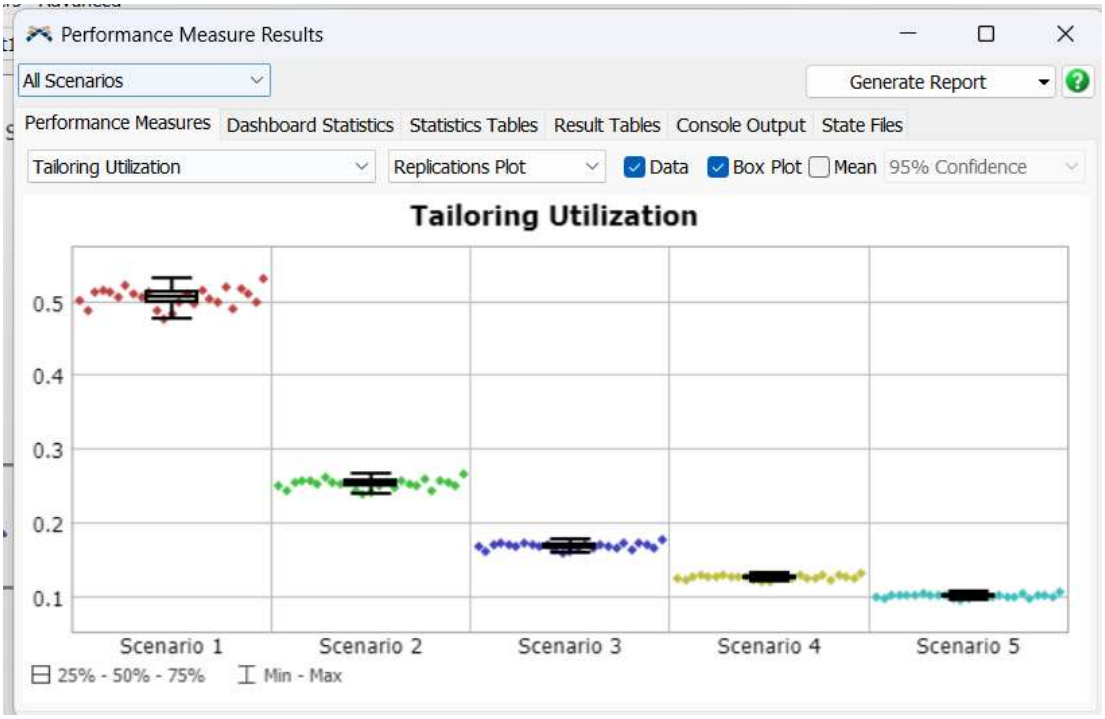

Figure 8. Tailoring Station Utilization

From these graphs, we can see that in scenario 2, the average queue content has significantly dropped also the operator efficiency has a slight increase in it. To reduce queue content in inventory that causes buffer can be reduced by applying conditions that we gained from the simulation. Only adding another testing will end up giving us the desired value and it is also very cost effective. Theoretically, if we install 5 tailoring station the average queue content will be less as the time goes by. But installing 5 workstations is not feasible in real life. The most economic decision would be adding one more testing station to reduce the buffer.

3.4 Case Study: The Production System at Urmee Garments

To demonstrate the application and benefits of a simulation-based, multi-objective optimization approach, a case study was conducted at a textile factory. The selected company is **Urmee Garments**, a prominent ready-made garment (RMG) manufacturing plant located in **Savar, Bangladesh**, an area that is a major hub for the nation's textile industry. Urmee Garments specializes in producing knitted apparel for the international market and, like many factories in the region, operates in a highly competitive, labor-intensive environment.

The production system at Urmee Garments is organized into several departments, with a primary focus on the sewing lines where garment assembly takes place. This study focuses on a specific sewing line responsible for producing a high-volume knitted garment style. The line (A) is composed of **11 distinct sequential operations**, from initial part preparation to

final assembly and inspection. Material flow is organized with a conveyor system, and buffer stocks of work-in-progress (WIP) are held between workstations to manage variability. The line is operated by **11 skilled workers** over a two-shift schedule.

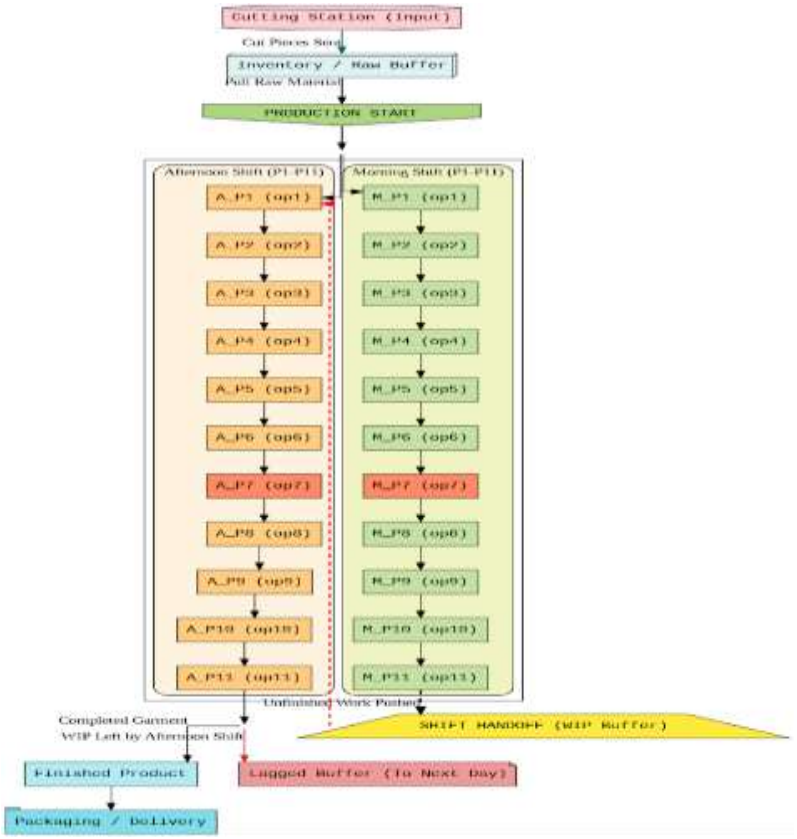

Figure 9. Production Line (A) of Urmee Garments

The primary challenge identified by the management at Urmee Garments is the optimization of daily operational production planning. The current system relies heavily on the experience of production managers and line supervisors to allocate workers and manage production flow. This approach often leads to several inefficiencies, including unbalanced production between workstations, the emergence of bottlenecks, long production lead times, and high levels of WIP inventory.

3.4.1 Data Collection and System Parameters

To build an accurate simulation model, it is essential to define the system parameters with reliable data. The product under analysis is a Tank Top for the buyer 'GU', the details of which are provided in Table 4.

Table 4. Product and Cutting Section Details

Parameter	Value
Buyer	GU
Style	60275F007A
Item	Tank top
Fabric	16 GSM S/J
Lay / Marker Quantity	100 / 1800
Output / Time	1800 pieces / 24 hr
Manpower (Cutting)	15

Table 5. Detailed Operation and Cycle Time Data (in seconds)

Op.	Actual Name of Operation	Ideal Cycle Time (s)	Jul 28 (98%)	Jul 29 (93%)	Jul 30 (93%)	Jul 31 (83%)	Aug 02 (96%)	Aug 03 (98%)	Aug 04 (97%)	Aug 06 (97%)	Aug 07 (96%)
op1	1st SLD join /TAPE/CTR/up to 5 Inch	12	12.2	12.8	12.2	14	12.5	12.2	12.4	12.4	12.5
op2	Care LBL fold and attach	15.6	15.9	16.7	15.9	18.3	16.2	15.9	16.1	16.1	16.2
op3	NK binding/up to 30 inch [Consider with extra fabric]	14.4	14.7	15.4	14.7	16.8	15	14.7	14.8	14.8	15
op4	2nd SLD TK/TM/above 2	12.6	12.9	13.5	12.9	14.7	13.1	12.9	13	13	13.1
op5	2nd shoulder join with TK/CTN/FT/SC/up to 4	16.8	17.1	18	17.1	19.7	17.5	17.1	17.3	17.3	17.5
op6	Arm Hole Binding Open /FL/CTR Up to[25*2]	24.6	25.1	26.3	25.1	28.8	25.6	25.1	25.3	25.3	25.6
op7	Side seam without lbl n without tk	30	30.6	32.1	30.6	35.1	31.2	30.6	30.9	30.9	31.2
op8	BTM hem /31 to 40 Inch	17.4	17.7	18.6	17.7	20.4	18.1	17.7	17.9	17.9	18.1
op9	BTM hem thread T/M	14.4	14.7	15.4	14.7	18.6	15	14.7	14.8	14.8	15
op10	NK 1/4 TK /TM	13.2	13.5	14.1	13.5	15.4	13.7	13.5	13.6	13.6	13.7
op11	Arm hole 1/4/TM/In-Out	20.4	20.8	21.8	21.8	23.9	21.2	20.8	21	21	21.2

For the sewing line, the primary operational data was extracted directly from the **Urmee Garments Enterprise Resource Planning (ERP) system** on **August 7, 2025**. This approach ensures that the data reflects the standardized and officially recorded performance metrics used in the factory's day-to-day management. The core data point extracted for each of the 11 operations was the **Standard Minute Value (SMV)**.

The factory's ERP system also provided the **historical operational efficiency** for the production line for nine consecutive working days. This efficiency percentage reflects how the actual performance of the workers compares against the ideal SMV. The cycle time for each operation—the actual average time taken to complete one unit—was then calculated for each day by adjusting the ideal cycle time with the day's efficiency factor. This detailed data, shown in Table 5, forms the basis for our simulation.

The simulation model was developed using **Tecnomatix Plant Simulation**, a discrete-event simulation (DES) software.

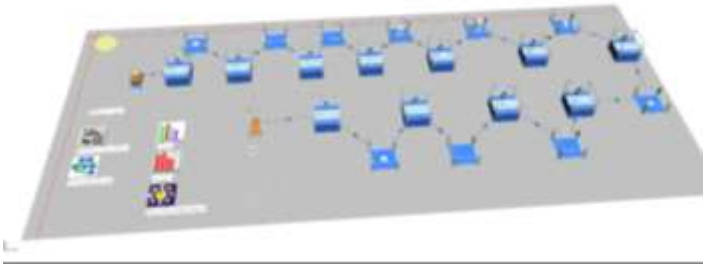

Figure 10. Simplified Model of the Sewing Line

3.4.2 Model Verification and Validation

Verification involved using low-speed graphic animation within the Tecnomatix software to visually confirm that the model's logic of movement, processing, and material flow was identical to the physical system on the factory floor.

Validation is the process of confirming that the verified model is an accurate representation of the real-world system. This was accomplished by comparing the production output from the simulation model against the actual production data from Urmee Garments for a corresponding period of nine days. A **paired t-test** was employed to statistically determine if there was a significant difference between the two sets of data.

The **Null Hypothesis (H_0)** was defined as: There is no significant difference between the DES production mean and the Real production mean.

The following calculations were performed:

1. **Mean of Differences ($\bar{d}$):**

$$\bar{d} = \frac{\sum(d_i)}{n} = \frac{86}{9} = 9.56$$

2. **Standard Deviation of Differences (s_d):**

$$s_d \approx 60.36$$

3. **Paired T-statistic (t):**

$$t = \frac{\bar{d}}{s_d/\sqrt{n}} = \frac{9.56}{60.36/\sqrt{9}} \approx 0.475$$

With **degrees of freedom (df) = n - 1 = 9 - 1 = 8**, the calculated t-statistic of **0.475** corresponds to a **p-value of 0.648**.

Since the p-value (0.648) is significantly greater than the standard alpha level of significance ($\alpha = 0.05$), we fail to reject the null hypothesis. This statistical result, supported by the visual comparison in Figure 11, provides strong evidence that there is no significant difference between the model's output and the real system's performance, thus confirming that the simulation is a **valid and accurate representation** of the actual production line.

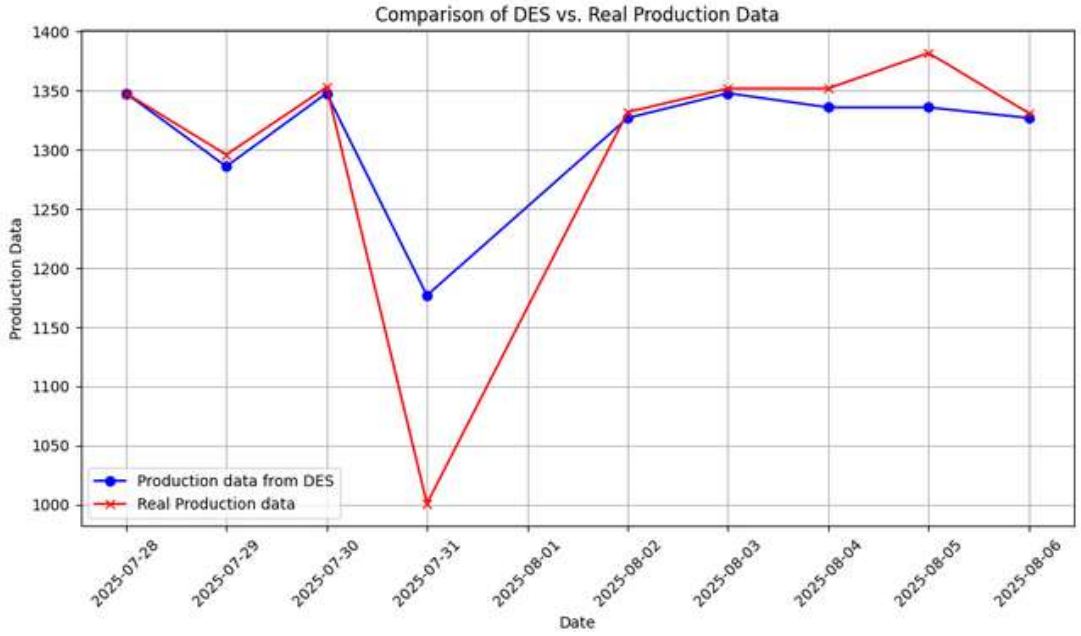

Figure 11. Comparison of Real vs Simulated Production Data

Chapter 4: Results and Analysis

4.1 Baseline Performance

The initial phase of the analysis focused on simulating the "as-is" state of the production line to establish a performance baseline. This model replicated the existing operational strategy at Urmee Garments, where one operator is assigned to each of the 11 workstations. The simulation determined that the maximum achievable daily throughput was approximately **1487 pieces**.

To confirm if a simple reallocation of operators could yield better results, both a deterministic Linear Programming (LP) model and a meta-heuristic Genetic Algorithm (GA) as well as Multi Objective Genetic Algorithm (MOGA) were applied. All methods independently concluded that the optimal allocation was the existing one: one operator for each of the 11 processes. The baseline cycle time, dictated by the slowest process, was **31.47 seconds**.

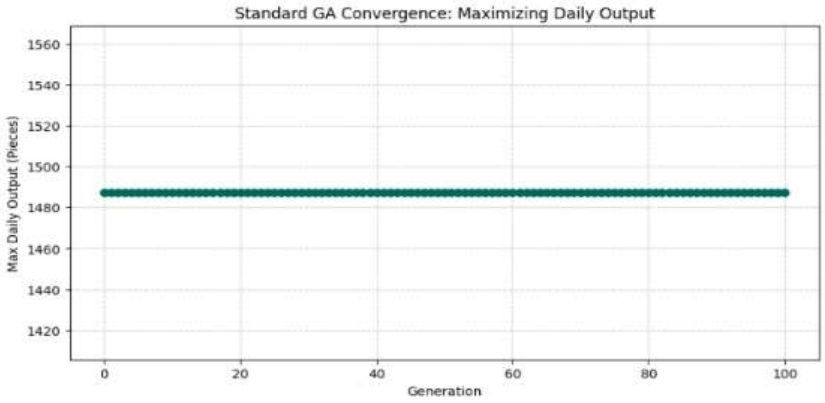

Figure 12. Baseline GA Convergence

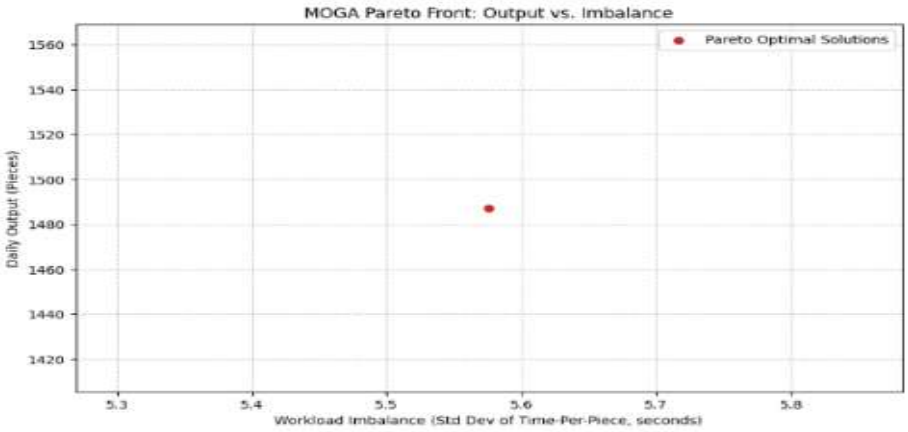

Figure 13. Baseline MOGA Convergence

4.2 Bottleneck Identification and Buffer Analysis

With the baseline performance established, the next critical step was to identify the root cause of the production limitation. A detailed analysis of the simulation model pinpointed **workstation 7 ('op7')** as the primary bottleneck of the entire production line. This conclusion was further substantiated by a two-pronged buffer analysis, which examined the state of the work-in-progress (WIP) buffers between each workstation. The physical buffers in the simulation model were set with a limited capacity of **10 pieces each**, mirroring the real-world line.

First, an analysis of buffer occupancy revealed a critical imbalance in the line's flow:

- The buffers for **workstations 1 through 6**, which precede the bottleneck, were occupied **more than 90% of the time**. This indicates that these earlier stations were producing faster than workstation 7 could process, leading to a significant pile-up of WIP and forcing these operators into idle states.

- Conversely, the buffers for **workstations 8 through 11**, which come after the bottleneck, were almost always empty. This shows that these later stations were starved of work, waiting for completed units from the slow-moving workstation 7.

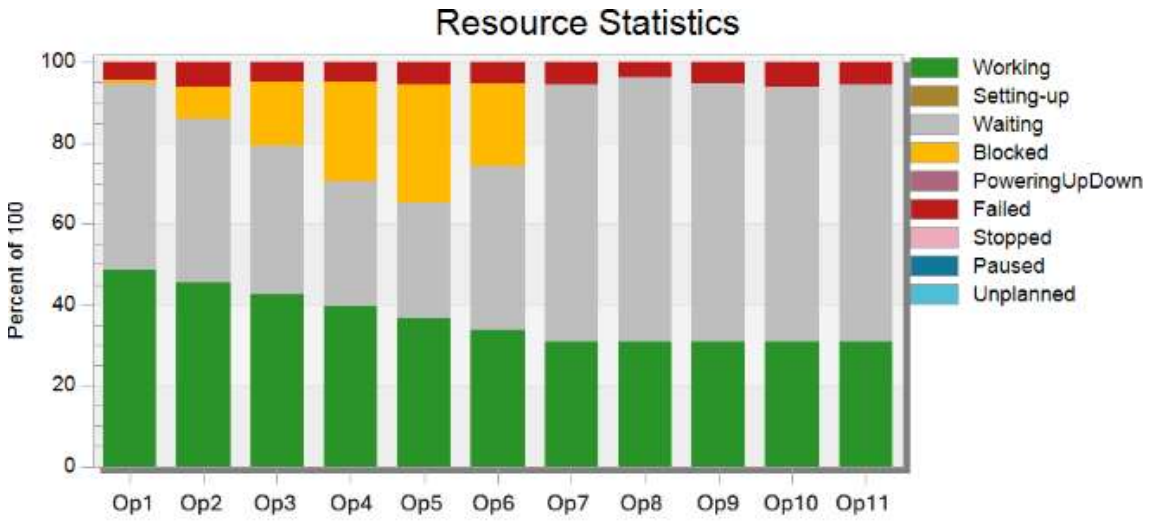

Figure 14. Bottlenecked Machine Capabilities

Second, a time-based buffer analysis was conducted on the baseline one-operator-per-station allocation. This calculation shows the time difference between each operation and the bottleneck operation (’op7’).

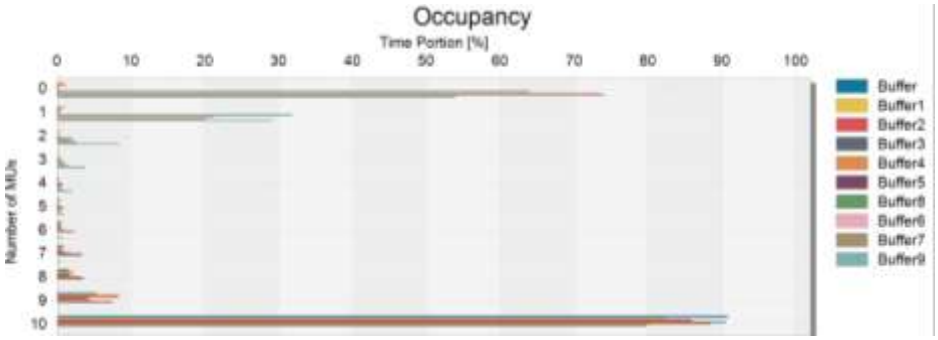

Figure 15. Buffer Capacities Showing High Occupancy Before op7

A negative value indicates spare capacity or potential idle time per piece produced. This analysis quantifies the imbalance.

Average Process Times (seconds): Average cycle times were used for the buffer analysis

{'op1': 12.578, 'op2': 16.367, 'op3': 15.10, 'op4': 13.233, 'op5': 17.624, 'op6': 25.8, 'op7': 31.465, 'op8': 18.234, 'op9': 15.10, 'op10': 13.856, 'op11': 21.389**}**

Buffer Analysis:

op1:

Buffer = -18.89 seconds per piece

op2:

Buffer = -15.10 seconds per piece

op3:

Buffer = -16.37 seconds per piece

op4:

Buffer = -18.23 seconds per piece

op5:

Buffer = -13.84 seconds per piece

op6:

Buffer = -5.67 seconds per piece

op7:

Buffer = 0.00 seconds per piece **(The Bottleneck)**

op8:

Buffer = -13.23 seconds per piece

op9:

Buffer = -16.37 seconds per piece

op10:

Buffer = -17.61 seconds per piece

op11:

Buffer = -10.08 seconds per piece

This dual analysis provided a clear visual and statistical confirmation that 'op7' was choking the flow of the entire line. The high occupancy of preceding buffers was not a sign of efficiency but a symptom of the downstream constraint. This insight was crucial, as it suggested that any meaningful improvement would have to address the imbalance around workstation 7, rather than simply focusing on individual station times.

4.3 Initial Optimization with Two-Operation Clustering

Based on the buffer analysis, a new optimization strategy was formulated: **task clustering**. A Genetic Algorithm (GA) was employed to explore this strategy, initially by testing clusters of two adjacent operations.

Table 6. Cycle Time after GA Optimization

Task Name	Time (s)	Workers	Buffer (s)
*** op3-op4 (Bundle) ***	28.33	1	0
op7 (Bottleneck Fix)	31.47	2	-12.6
op1 (Single)	12.58	1	-15.76
op2 (Single)	16.37	1	-11.97
op5 (Single)	17.62	1	-10.71
op6 (Single)	25.8	1	-2.53
op8 (Single)	18.23	1	-10.1
op9 (Single)	15.1	1	-13.23

op10 (Single)	13.86	1	-14.48
op11 (Single)	21.39	1	-6.94

The most effective two-operation bundle identified was the clustering of **'op3' and 'op4'**. The results were a new daily throughput of **1651 pieces** and a reduced cycle time of **28.33 seconds**, representing a **10.9% increase in throughput**, this can be viewed in a detailed cycle time report at Table 6

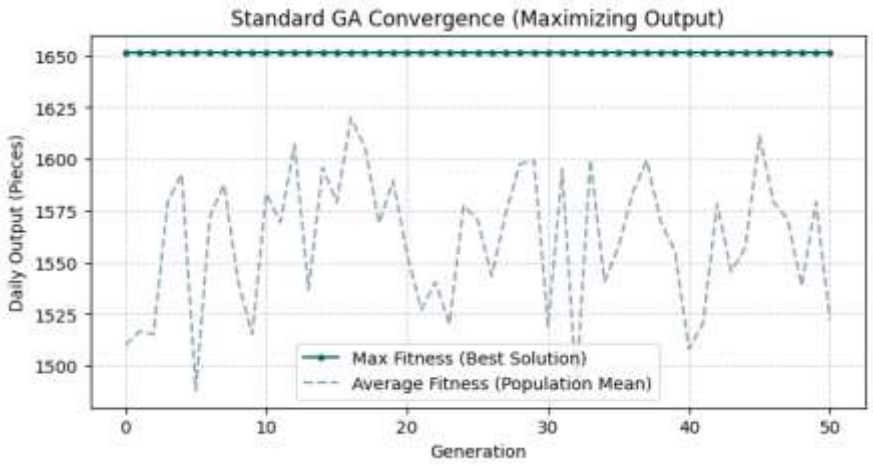

Figure 16. GA Results for Dual Clustering

Transitioning to a multi-objective paradigm, a Multi-Objective Genetic Algorithm (MOGA) was implemented to explore the solution space across two competing objectives: the maximization of throughput and the minimization of workload imbalance. Workload imbalance is defined herein as the standard deviation of the effective process times across all operator stations, where a lower value indicates a more equitable and sustainable distribution of labor. The MOGA identified a non-dominated solution on the Pareto front that achieved an identical maximum throughput of **1651** pieces by also identifying the **'op3-op4'** cluster and the reallocation of a worker to 'op7'. However, the key distinction of the MOGA solution was the significant reduction in systemic imbalance, achieving a workload variance of only **4.89 seconds**. This demonstrates that while both algorithms can find a solution for maximum output, the MOGA provides a solution that is equally productive but managerially superior due to its more stable and balanced nature.

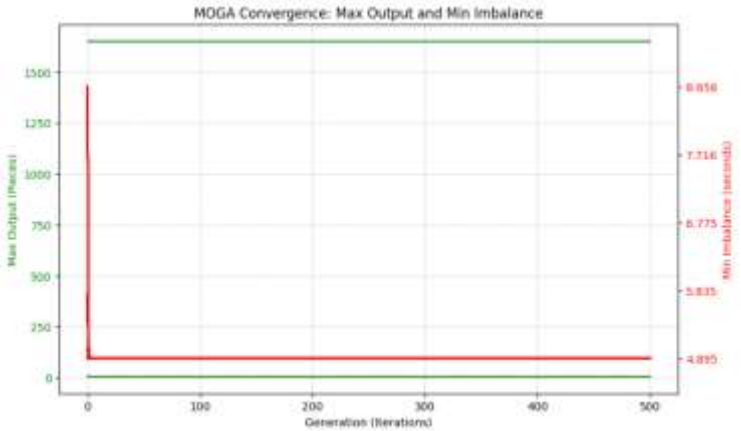

Figure 17. MOGA Pareto Front for Dual Clustering

4.4 Secondary Optimization with Three-Operation Clustering

Building on the success of the two-operation clusters, the final optimization phase explored clusters of three adjacent operations. Three possible scenarios were considered, clustering three processes to free up effective two workers, one worker goes to station 7 another worker goes to next largest bottleneck station.

The Genetic Algorithm identified the optimal task cluster for maximizing output to be the bundling of **'op3', 'op4', and 'op5'**. This three-operation cluster yielded a final daily throughput of **1814 pieces** and a final bottleneck cycle time of **25.8 seconds**. This represents a **22% increase in throughput** compared to the original baseline.

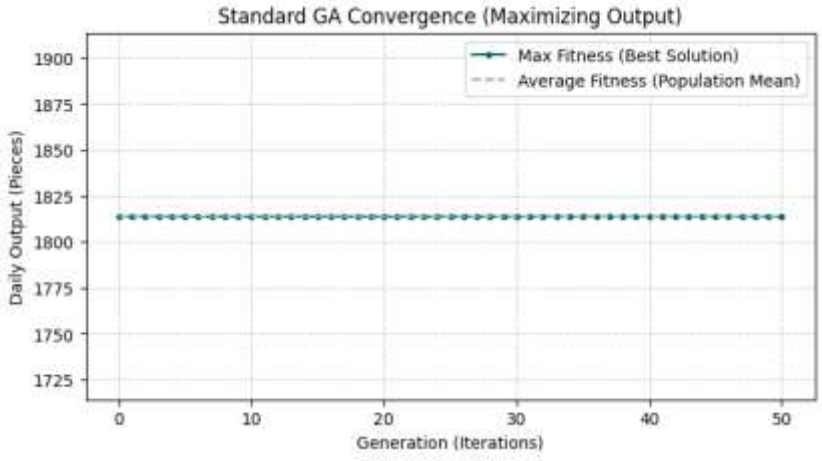

Figure 18. GA Results for Triple Clustering

The Multi Objective Genetic Algorithm Clustered **(op1, op2, op3), (op2, op3, op4), (op3, op4, op5**). The second objective was to reduce the combined cycle time of operations **9 to 11** to reduce overall lead time. The MOGA achieved a daily output of 1814 pieces and obtained total operation time of 50.34 seconds for op 9-11.

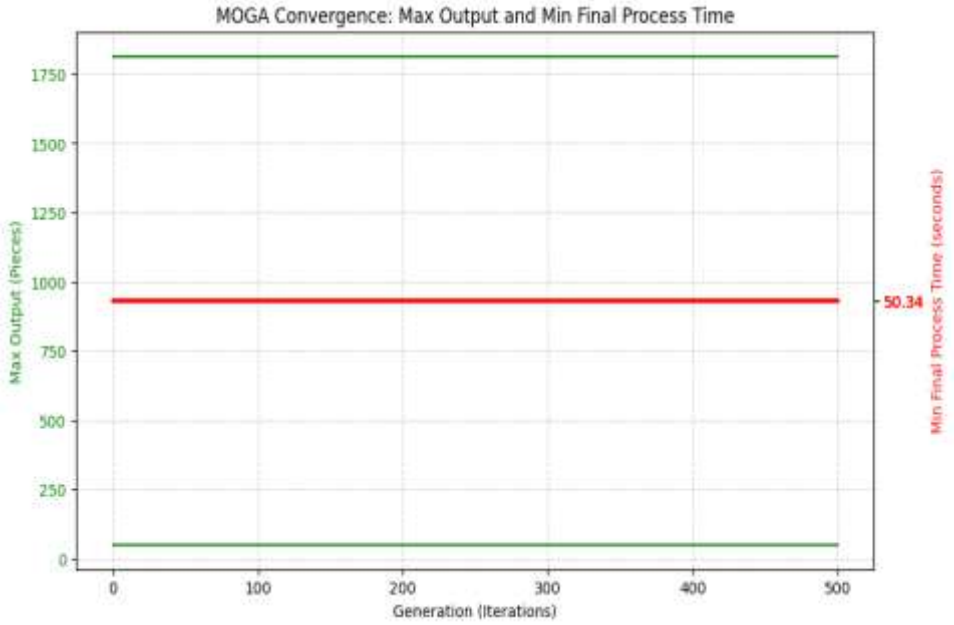

Figure 19. MOGA Pareto Front for Triple Clustering

Chapter 5: Discussion

The stepwise progression from baseline analysis to advanced optimization reveals a powerful narrative for production improvement. The true turning point was the **bottleneck and buffer analysis**. Identifying ’op7’ as the constraint and understanding its impact on the preceding buffers (over 90% occupancy) provided the critical insight that drove the subsequent strategy. The problem was not just one slow station, but a systemic imbalance.

The success of the **task clustering strategy** demonstrates the power of the simulation-optimization approach. The 22% improvement achieved with the three-operation cluster of ’op3-op4-op5’ represents the full realization of this strategy. It is a non-intuitive solution that would be difficult for a human planner to identify through trial and error, highlighting the value of the MOGA’s ability to explore a vast and complex solution space.

These findings have profound practical implications for Urmee Garments. A 22% increase in throughput achieved solely through intelligent work redistribution—with zero capital expenditure—is a massive competitive advantage.

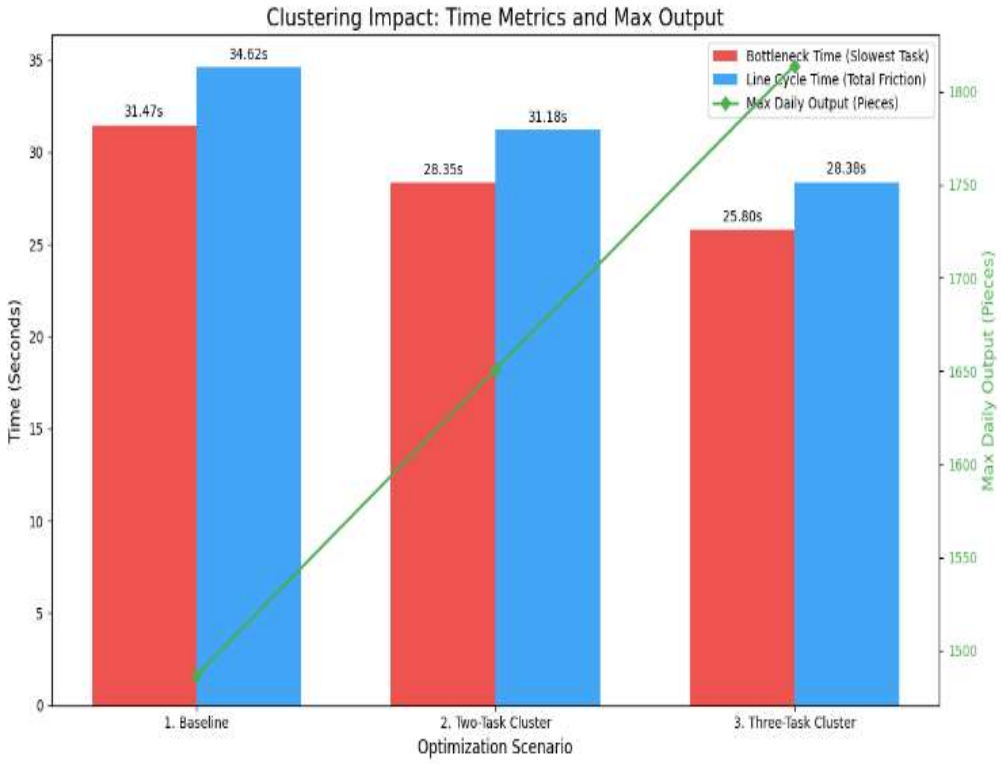

Figure 20. Optimization Result Comparison

Chapter 6: Conclusion and Future Scope

6.1 Conclusion

This thesis successfully demonstrated the application of discrete-event simulation combined with a multi-objective genetic algorithm to optimize a production line at Urmee Garments. The primary conclusions are:

- The baseline system was constrained by a bottleneck at **operation 'op7'**, limiting daily throughput to **1487 pieces**.
- A detailed buffer analysis revealed a systemic imbalance, which informed a **task clustering strategy**.
- An initial two-operation cluster increased throughput by 11% to **1651 pieces**.
- A secondary three-operation cluster (**'op3-op4-op5'**) provided the optimal solution, increasing throughput by a total of **22% to 1814 pieces**.

This research validates that a systematic, simulation-led approach can unlock substantial productivity gains.

6.2 Future Scope

The results of this study open several avenues for future research and application.

- **Implementation and Pilot Testing:** The immediate next step should be to pilot the recommended clustering strategy on the actual production line.
- **Stochastic and Dynamic Modeling:** Future research should focus on developing stochastic models that incorporate real-world uncertainties such as machine breakdowns and quality variations.
- **Integration with ERP:** The simulation tool could be integrated with the factory's ERP system for dynamic daily optimization.
- **Broader Multi-Objective Analysis:** Future studies could expand the set of objectives to include total manufacturing cost, energy consumption, and ergonomic factors.

References

[1] M. Chandra Biswas, “Samit Chakraborty Impact of COVID-19 on the textile, apparel and fashion manufacturing industry supply chain : Case study on a ready-made garment manufacturing industry MANIK CHANDRA BISWAS.” [Online]. Available: https://ssrn.com/abstract=3781228

[2] M. A. Rahman, “What Makes Bangladeshi RMG Suppliers Resilient in Global Apparel Supply Chain Management?,” *Journal of Economics, Management and Trade*, vol. 29, no. 8, pp. 153–176, Jun. 2023, doi: 10.9734/jemt/2023/v29i81128.

[3] M. A. Samad, P. P. Chowdhury, G. Rabbani, and M. N. A. Siddiqe Uzzal, “Study of a Sewing Line of an RMG Factory and Productivity Improvement through Line Balancing,” *Asian Journal of Engineering and Applied Technology*, vol. 12, no. 1, pp. 1–9, Apr. 2023, doi: 10.51983/ajeat-2023.12.1.3501.

[4] J. Pereira, “Modelling and solving a cost-oriented resource-constrained multi-model assembly line balancing problem,” *Int J Prod Res*, vol. 56, no. 11, pp. 3994–4016, Jun. 2018, doi: 10.1080/00207543.2018.1427899.

[5] G. D. Monek and S. Fischer, “DES and IIoT fusion approach towards real-time synchronization of physical and digital components in manufacturing processes,” *Reports in Mechanical Engineering*, vol. 4, no. 1, pp. 161–174, Dec. 2023, doi: 10.31181/rme040115092023m.

[6] C. Ramos, R. Barreto, B. Mota, L. Gomes, P. Faria, and Z. Vale, “Scheduling of a textile production line integrating PV generation using a genetic algorithm,” *Energy Reports*, vol. 6, pp. 148–154, Dec. 2020, doi: 10.1016/j.egyr.2020.11.093.

[7] S. Bojic, M. Maslaric, D. Mircetic, S. Nikolicic, and V. Todorovic, “Simulation and Genetic Algorithm-based approach for multi-objective optimization of production planning: A case study in industry,” *Advances in Production Engineering And Management*, vol. 18, no. 2, pp. 250–262, Jun. 2023, doi: 10.14743/apem2023.2.471.

[8] R. Ferro, G. A. Cordeiro, R. E. C. Ordóñez, G. Beydoun, and N. Shukla, “An optimization tool for production planning: A case study in a textile industry,” *Applied Sciences (Switzerland)*, vol. 11, no. 18, Sep. 2021, doi: 10.3390/app11188312.

[9] G. Fedorko, N. Mikušová, L. Čabaníková, M. Vasiľ, and P. Ondič, “Simulation in the digital factory: A case study,” *Advances in Science and Technology Research Journal*, vol. 19, no. 2, pp. 16–27, 2025, doi: 10.12913/22998624/195244.

[10] G. W. Woubante, “The Optimization Problem of Product Mix and Linear Programming Applications: Case Study in the Apparel Industry,” 2017.

[11] I. Z. Mim, M. G. S. Rayhan, and M. Syduzzaman, “Prospects and current scenario of industry 4.0 in Bangladeshi textile and apparel industry,” *Heliyon*, vol. 10, no. 11, Jun. 2024, doi: 10.1016/j.heliyon.2024.e32044.

[12] Md. M. Habib, S. Sabah, F. Chowdhury, R. Raisa, and T. F. Shuvo, “Technology Enabled Ready-Made Garments Supply Chain Management: a Literature Review,” *Revista de Gestão Social e Ambiental*, vol. 18, no. 10, p. e08842, Oct. 2024, doi: 10.24857/rgsa.v18n10-188.

[13] A. A. Tako and S. Robinson, “The application of discrete event simulation and system dynamics in the logistics and supply chain context,” *Decis Support Syst*, vol. 52, no. 4, pp. 802–815, Mar. 2012, doi: 10.1016/j.dss.2011.11.015.

[14] T. Yoo, H. Cho, and E. Yücesan, “Hybrid algorithm for discrete event simulation based supply chain optimization,” *Expert Syst Appl*, vol. 37, no. 3, pp. 2354–2361, Mar. 2010, doi: 10.1016/j.eswa.2009.07.039.

[15] S. A. Khan, A. Chaabane, and F. T. Dweiri, “Multi-Criteria Decision-Making Methods Application in Supply Chain Management: A Systematic Literature Review,” in *Multi-Criteria Methods and Techniques Applied to Supply Chain Management*, InTech, 2018. doi: 10.5772/intechopen.74067.

[16] E. Bottani and G. Casella, “Discrete-event simulation in logistics and supply chain management: a scientometric perspective,” *Prod Manuf Res*, vol. 12, no. 1, 2024, doi:

10.1080/21693277.2024.2415038.
[17] T. Lacksonen, “Empirical comparison of search algorithms for discrete event simulation.” [Online]. Available: www.elsevier.com/locate/dsw
[18] L. Fumagalli, E. Negri, E. Sottoriva, A. Polenghi, and M. Macchi, “A novel scheduling framework: integrating genetic algorithms and discrete event simulation.”
[19] P. Ghasemi, A. Goli, F. Goodarzian, and J. F. Ehmke, “Simulation-based genetic algorithm for optimizing a municipal cooperative waste supply chain in a pandemic,” *Eng Appl Artif Intell*, vol. 139, Jan. 2025, doi: 10.1016/j.engappai.2024.109478.
[20] A. Panichella, “An improved Pareto front modeling algorithm for large-scale many-objective optimization,” in *GECCO 2022 - Proceedings of the 2022 Genetic and Evolutionary Computation Conference*, Association for Computing Machinery, Inc, Jul. 2022, pp. 565–573. doi: 10.1145/3512290.3528732.
[21] P. K. Yadav, S. Singh, C. M. Tripathi, and R. Bhushan, “Supply Chain Schedule Management with Genetic Algorithm,” in *2023 International Conference on Evolutionary Algorithms and Soft Computing Techniques, EASCT 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/EASCT59475.2023.10393211.
[22] S. Mithun and M. Abdul, “Evaluating barriers to implementing green supply chain management: An example from an emerging economy.” [Online]. Available: https://shura.shu.ac.uk/32488/

Appendices

Optimization Codes

LP Code for Base Line Performance

```
import pandas as pd
from pulp import LpProblem, LpMaximize, LpVariable, LpInteger, lpSum, LpStatus

# --- 1. Data Processing ---

# The data from your Excel sheet (approximated from the image)
# op1 to op11 represent the time in seconds for each process
data = {
    'op1': [12.2, 12.8, 12.2, 14.0, 12.5, 12.2, 12.4, 12.4, 12.5],
    'op2': [15.9, 16.7, 15.9, 18.3, 16.2, 15.9, 16.1, 16.1, 16.2],
    'op3': [14.7, 15.4, 14.7, 16.8, 15.0, 14.7, 14.8, 14.8, 15.0],
    'op4': [12.9, 13.5, 12.9, 14.7, 13.1, 12.9, 13.0, 13.0, 13.1],
    'op5': [17.1, 18.0, 17.1, 19.7, 17.5, 17.1, 17.3, 17.3, 17.5],
    'op6': [25.1, 26.3, 25.1, 28.8, 25.6, 25.1, 25.3, 25.3, 25.6],
    'op7': [30.6, 32.1, 30.6, 35.1, 31.2, 30.6, 30.9, 30.9, 31.2],
    'op8': [17.7, 18.6, 17.7, 20.4, 18.1, 17.7, 17.9, 17.9, 18.1],
    'op9': [14.7, 15.4, 14.7, 16.8, 15.0, 14.7, 14.8, 14.8, 15.0],
    'op10': [13.5, 14.1, 13.5, 15.4, 13.8, 13.5, 13.6, 13.6, 13.7],
    'op11': [20.8, 21.8, 20.8, 23.9, 21.2, 20.8, 21.0, 21.0, 21.2]
}

df = pd.DataFrame(data)

# Calculate the average time for each process
avg_process_times = df.mean().to_dict()

# Total available working time per day (in seconds)
# 13 hours * 3600 seconds/hour
total_working_seconds = 13 * 3600
num_operators_per_shift = 11

# --- 2. Define the Optimization Problem ---

# Create a problem instance to maximize output
prob = LpProblem("TankTopProductionOptimization", LpMaximize)

# Decision variables: Number of operators for each process
```

```
operators = LpVariable.dicts("Operators", avg_process_times.keys(), 0,
num_operators_per_shift, LpInteger)

# New objective variable: Total number of pieces produced per day
total_pieces = LpVariable("Total_Pieces", 0, None)

# Objective: Maximize the total number of pieces produced
prob += total_pieces, "Maximize_Total_Pieces"

# Constraint 1: The total number of operators must be 11
prob += lpSum(operators.values()) == num_operators_per_shift, "Total_Operators_Constraint"

# Constraint 2: The production rate of each process must support the total number of pieces.
# This is a linear formulation: total time needed <= total time available.
# (total_pieces * time_per_piece) <= (operators * total_working_seconds)
for op, time in avg_process_times.items():
    prob += total_pieces * time <= operators[op] * total_working_seconds,
f"Capacity_Constraint_{op}"

# --- 3. Solve the Problem ---
prob.solve()

# --- 4. Display the Results ---

print("Status:", LpStatus[prob.status])

if LpStatus[prob.status] == 'Optimal':
    print("\nOptimal Operator Allocation per Process:")
    for op in sorted(avg_process_times.keys(), key=lambda x: int(x.replace('op', ''))):
        print(f"  {op}: {int(operators[op].varValue)} operators")

    # Calculate the new, optimized production rate and output
    optimized_daily_output = total_pieces.varValue
    optimized_time_per_unit = total_working_seconds / optimized_daily_output

    print("\nOptimized Results:")
    print(f"  Optimized Daily Output: {optimized_daily_output:.2f} pieces")
    print(f"  Maximum time per unit (bottleneck): {optimized_time_per_unit:.2f} seconds")

    # Analyze Buffers
    print("\nBuffer Analysis (Based on optimized allocation):")
    for op in sorted(avg_process_times.keys(), key=lambda x: int(x.replace('op', ''))):
        operators_count = int(operators[op].varValue)
        if operators_count > 0:
            process_time_per_unit = avg_process_times[op] / operators_count
            buffer = (process_time_per_unit - optimized_time_per_unit)
            print(f"  {op}: Buffer = {buffer:.2f} seconds per piece")
```

```
else:
    print("No optimal solution found.")
```

GA Code for Dual Operation Clustering

```
import random
import pandas as pd
import numpy as np
from deap import base, creator, tools, algorithms
import matplotlib.pyplot as plt # Import for plotting

# --- 1. Data Processing ---
data = {
    'op1': [12.2, 12.8, 12.2, 14.0, 12.5, 12.2, 12.4, 12.4, 12.5],
    'op2': [15.9, 16.7, 15.9, 18.3, 16.2, 15.9, 16.1, 16.1, 16.2],
    'op3': [14.7, 15.4, 14.7, 16.8, 15.0, 14.7, 14.8, 14.8, 15.0],
    'op4': [12.9, 13.5, 12.9, 14.7, 13.1, 12.9, 13.0, 13.0, 13.1],
    'op5': [17.1, 18.0, 17.1, 19.7, 17.5, 17.1, 17.3, 17.3, 17.5],
    'op6': [25.1, 26.3, 25.1, 28.8, 25.6, 25.1, 25.3, 25.3, 25.6],
    'op7': [30.6, 32.1, 30.6, 35.1, 31.2, 30.6, 30.9, 30.9, 31.2],
    'op8': [17.7, 18.6, 17.7, 20.4, 18.1, 17.7, 17.9, 17.9, 18.1],
    'op9': [14.7, 15.4, 14.7, 16.8, 15.0, 14.7, 14.8, 14.8, 15.0],
    'op10': [13.5, 14.1, 13.5, 15.4, 13.8, 13.5, 13.6, 13.6, 13.7],
    'op11': [20.8, 21.8, 20.8, 23.9, 21.2, 20.8, 21.0, 21.0, 21.2]
}

df = pd.DataFrame(data)
avg_process_times = df.mean().to_dict()

total_working_seconds = 13 * 3600
num_operators_per_shift = 11
NUM_PROCESSES = 11

# Process keys and times as a list for easy indexing
process_keys = [f'op{i}' for i in range(1, NUM_PROCESSES + 1)]
process_times_list = [avg_process_times[key] for key in process_keys]
BUNDLE_OPTIONS = 5 # P1-P2 up to P5-P6 (indices 0 to 4)
# MOGA WEIGHTS (Set 1)
weights=(1.0, -1.0)
# Goal: Maximize Output, Minimize Bundle Time (Complexity)

def evaluate_multi_objective(individual):
    # ... calculates effective_times ...

    # O1: Maximize Daily Output (using bottleneck_time)
    daily_output = total_working_seconds / bottleneck_time

    # O2: Minimize Bundle Time (Get the time of the single bundled task)
    bundle_index = individual[0]
```

```
    bundle_time = process_times_list[bundle_index] + process_times_list[bundle_index + 1]

    return daily_output, bundle_time
# --- 2. DEAP Helper Functions (Defined once) ---

def calculate_effective_times(bundle_index):
    """Helper function to calculate the time for the 10 effective stations."""
    effective_times_per_unit = []
    bundled_op_index_start = bundle_index

    for i in range(NUM_PROCESSES):
        time = process_times_list[i]

        if i == bundled_op_index_start:
            # Bundled Task Time = Op(i) + Op(i+1)
            bundle_time = time + process_times_list[i+1]
            effective_times_per_unit.append(bundle_time)
        elif i == bundled_op_index_start + 1:
            # Skip the second task in the bundle
            continue
        elif i == 6: # OP7 (index 6) is fixed at 2 workers
            effective_times_per_unit.append(time / 2)
        else:
            # All other tasks run with 1 worker
            effective_times_per_unit.append(time / 1)

    return np.array(effective_times_per_unit)


def evaluate_bundling_output(individual):
    """Calculates the total daily output (Max Fitness)."""
    effective_times = calculate_effective_times(individual[0])
    bottleneck_time = max(effective_times)
    daily_output = total_working_seconds / bottleneck_time
    return daily_output,

def evaluate_multi_objective(individual):
    """Calculates daily output (maximize) and workload imbalance (minimize)."""
    effective_times = calculate_effective_times(individual[0])

    # Objective 1: Daily Output (based on bottleneck)
    bottleneck_time = max(effective_times)
    daily_output = total_working_seconds / bottleneck_time

    # Objective 2: Workload Imbalance (standard deviation of the effective times)
    imbalance = np.std(effective_times)

    return daily_output, imbalance
```

```
# Mutation and Crossover functions (simple integer flipping/placeholder)
def mut_int_flip(individual, indpb):
    if random.random() < indpb:
        individual[0] = random.randint(0, BUNDLE_OPTIONS - 1)
    return individual,

def cx_int_placeholder(ind1, ind2):
    return ind1, ind2

# --- 3. GA Execution Functions (Isolated Setups) ---

def run_ga_bundling():
    """Runs the Standard GA (Single Objective) with full setup isolation."""

    # 1. Isolation: Clean up previous definitions if they exist
    if hasattr(creator, 'FitnessMax'): del creator.FitnessMax
    if hasattr(creator, 'FitnessMulti'): del creator.FitnessMulti
    if hasattr(creator, 'Individual'): del creator.Individual

    # 2. Define Components for Single Objective GA
    creator.create("FitnessMax", base.Fitness, weights=(1.0,))
    creator.create("Individual", list, fitness=creator.FitnessMax)

    toolbox = base.Toolbox()
    toolbox.register("attr_int", random.randint, 0, BUNDLE_OPTIONS - 1)
    toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_int,
n=1)
    toolbox.register("population", tools.initRepeat, list, toolbox.individual)

    # Register core operations
    toolbox.register("evaluate", evaluate_bundling_output)
    toolbox.register("mate", cx_int_placeholder)
    toolbox.register("mutate", mut_int_flip, indpb=0.5)
    toolbox.register("select", tools.selTournament, tournsize=2)

    POP_SIZE = 20
    NGEN = 50

    pop = toolbox.population(n=POP_SIZE)
    hof = tools.HallOfFame(1)

    # Statistics Setup for Convergence Plot
    stats = tools.Statistics(lambda ind: ind.fitness.values[0])
    stats.register("max", np.max)
    stats.register("avg", np.mean)
    logbook = tools.Logbook()

    # Run the Simple GA
```

```
    pop, log = algorithms.eaSimple(pop, toolbox, cxpb=0.1, mutpb=0.8, ngen=NGEN,
                                   stats=stats, halloffame=hof, verbose=False)

    # --- Display Results and Plotting ---
    best_bundle_index = hof[0][0]
    optimized_daily_output, = evaluate_bundling_output(hof[0])
    bundle_names = [f'op{i+1}-op{i+2}' for i in range(BUNDLE_OPTIONS)]
    chosen_bundle_name = bundle_names[best_bundle_index]

    print("\n--- OPTIMIZATION RESULTS (Genetic Algorithm) ---")
    print(f"Optimal Task Bundle to Maximize Output: {chosen_bundle_name}")
    print(f"Total Daily Output: {optimized_daily_output:.2f} pieces")

    # --- Convergence Graph Section ---
    max_fitness_values = log.select("max")
    avg_fitness_values = log.select("avg")

    plt.figure(figsize=(8, 4))
    plt.plot(range(NGEN + 1), max_fitness_values, marker='o', linestyle='-', color='#00695c',
markersize=3, label='Max Fitness (Best Solution)')
    plt.plot(range(NGEN + 1), avg_fitness_values, linestyle='--', color='#90A4AE',
label='Average Fitness (Population Mean)')

    plt.title('Standard GA Convergence (Maximizing Output)')
    plt.xlabel('Generation (Iterations)') # Clarified x-axis label
    plt.ylabel('Daily Output (Pieces)')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.legend()
    plt.show()
```

MOGA Code for Dual Operation Clustering

```
import random
import pandas as pd
import numpy as np
from deap import base, creator, tools, algorithms
import matplotlib.pyplot as plt # Import for plotting

# --- 1. Data Processing ---
data = {
    'op1': [12.2, 12.8, 12.2, 14.0, 12.5, 12.2, 12.4, 12.4, 12.5],
    'op2': [15.9, 16.7, 15.9, 18.3, 16.2, 15.9, 16.1, 16.1, 16.2],
    'op3': [14.7, 15.4, 14.7, 16.8, 15.0, 14.7, 14.8, 14.8, 15.0],
    'op4': [12.9, 13.5, 12.9, 14.7, 13.1, 12.9, 13.0, 13.0, 13.1],
    'op5': [17.1, 18.0, 17.1, 19.7, 17.5, 17.1, 17.3, 17.3, 17.5],
    'op6': [25.1, 26.3, 25.1, 28.8, 25.6, 25.1, 25.3, 25.3, 25.6],
    'op7': [30.6, 32.1, 30.6, 35.1, 31.2, 30.6, 30.9, 30.9, 31.2],
    'op8': [17.7, 18.6, 17.7, 20.4, 18.1, 17.7, 17.9, 17.9, 18.1],
    'op9': [14.7, 15.4, 14.7, 16.8, 15.0, 14.7, 14.8, 14.8, 15.0],
    'op10': [13.5, 14.1, 13.5, 15.4, 13.8, 13.5, 13.6, 13.6, 13.7],
```

```
    'op11': [20.8, 21.8, 20.8, 23.9, 21.2, 20.8, 21.0, 21.0, 21.2]
}

df = pd.DataFrame(data)
avg_process_times = df.mean().to_dict()

total_working_seconds = 13 * 3600
num_operators_per_shift = 11
NUM_PROCESSES = 11

# Process keys and times as a list for easy indexing
process_keys = [f'op{i}' for i in range(1, NUM_PROCESSES + 1)]
process_times_list = [avg_process_times[key] for key in process_keys]
BUNDLE_OPTIONS = 5 # P1-P2 up to P5-P6 (indices 0 to 4)
# MOGA WEIGHTS (Set 1)
weights=(1.0, -1.0)
# Goal: Maximize Output, Minimize Bundle Time (Complexity)

def evaluate_multi_objective(individual):
    # ... calculates effective_times ...

    # O1: Maximize Daily Output (using bottleneck_time)
    daily_output = total_working_seconds / bottleneck_time

    # O2: Minimize Bundle Time (Get the time of the single bundled task)
    bundle_index = individual[0]
    bundle_time = process_times_list[bundle_index] + process_times_list[bundle_index + 1]

    return daily_output, bundle_time
# --- 2. DEAP Helper Functions (Defined once) ---

def calculate_effective_times(bundle_index):
    """Helper function to calculate the time for the 10 effective stations."""
    effective_times_per_unit = []
    bundled_op_index_start = bundle_index

    for i in range(NUM_PROCESSES):
        time = process_times_list[i]

        if i == bundled_op_index_start:
            # Bundled Task Time = Op(i) + Op(i+1)
            bundle_time = time + process_times_list[i+1]
            effective_times_per_unit.append(bundle_time)
        elif i == bundled_op_index_start + 1:
            # Skip the second task in the bundle
            continue
        elif i == 6: # OP7 (index 6) is fixed at 2 workers
            effective_times_per_unit.append(time / 2)
```

```
        else:
            # All other tasks run with 1 worker
            effective_times_per_unit.append(time / 1)

    return np.array(effective_times_per_unit)


def evaluate_bundling_output(individual):
    """Calculates the total daily output (Max Fitness)."""
    effective_times = calculate_effective_times(individual[0])
    bottleneck_time = max(effective_times)
    daily_output = total_working_seconds / bottleneck_time
    return daily_output,

def evaluate_multi_objective(individual):
    """Calculates daily output (maximize) and workload imbalance (minimize)."""
    effective_times = calculate_effective_times(individual[0])

    # Objective 1: Daily Output (based on bottleneck)
    bottleneck_time = max(effective_times)
    daily_output = total_working_seconds / bottleneck_time

    # Objective 2: Workload Imbalance (standard deviation of the effective times)
    imbalance = np.std(effective_times)

    return daily_output, imbalance

# Mutation and Crossover functions (simple integer flipping/placeholder)
def mut_int_flip(individual, indpb):
    if random.random() < indpb:
        individual[0] = random.randint(0, BUNDLE_OPTIONS - 1)
    return individual,

def cx_int_placeholder(ind1, ind2):
    return ind1, ind2

# --- 3. GA Execution Functions (Isolated Setups) ---

def run_ga_bundling():
    """Runs the Standard GA (Single Objective) with full setup isolation."""

    # 1. Isolation: Clean up previous definitions if they exist
    if hasattr(creator, 'FitnessMax'): del creator.FitnessMax
    if hasattr(creator, 'FitnessMulti'): del creator.FitnessMulti
    if hasattr(creator, 'Individual'): del creator.Individual

    # 2. Define Components for Single Objective GA
    creator.create("FitnessMax", base.Fitness, weights=(1.0,))
    creator.create("Individual", list, fitness=creator.FitnessMax)
```

```
    toolbox = base.Toolbox()
    toolbox.register("attr_int", random.randint, 0, BUNDLE_OPTIONS - 1)
    toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_int,
n=1)
    toolbox.register("population", tools.initRepeat, list, toolbox.individual)

    # Register core operations
    toolbox.register("evaluate", evaluate_bundling_output)
    toolbox.register("mate", cx_int_placeholder)
    toolbox.register("mutate", mut_int_flip, indpb=0.5)
    toolbox.register("select", tools.selTournament, tournsize=2)

    POP_SIZE = 20
    NGEN = 50

    pop = toolbox.population(n=POP_SIZE)
    hof = tools.HallOfFame(1)

    # Statistics Setup for Convergence Plot
    stats = tools.Statistics(lambda ind: ind.fitness.values[0])
    stats.register("max", np.max)
    stats.register("avg", np.mean)
    logbook = tools.Logbook()

    # Run the Simple GA
    pop, log = algorithms.eaSimple(pop, toolbox, cxpb=0.1, mutpb=0.8, ngen=NGEN,
                                   stats=stats, halloffame=hof, verbose=False)

    # --- Display Results and Plotting ---
    best_bundle_index = hof[0][0]
    optimized_daily_output, = evaluate_bundling_output(hof[0])
    bundle_names = [f'op{i+1}-op{i+2}' for i in range(BUNDLE_OPTIONS)]
    chosen_bundle_name = bundle_names[best_bundle_index]

    print("\n--- OPTIMIZATION RESULTS (Genetic Algorithm) ---")
    print(f"Optimal Task Bundle to Maximize Output: {chosen_bundle_name}")
    print(f"Total Daily Output: {optimized_daily_output:.2f} pieces")

    # --- Convergence Graph Section ---
    max_fitness_values = log.select("max")
    avg_fitness_values = log.select("avg")

    plt.figure(figsize=(8, 4))
    plt.plot(range(NGEN + 1), max_fitness_values, marker='o', linestyle='-', color='#00695c',
markersize=3, label='Max Fitness (Best Solution)')
    plt.plot(range(NGEN + 1), avg_fitness_values, linestyle='--', color='#90A4AE',
label='Average Fitness (Population Mean)')
```

```
    plt.title('Standard GA Convergence (Maximizing Output)')
    plt.xlabel('Generation (Iterations)') # Clarified x-axis label
    plt.ylabel('Daily Output (Pieces)')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.legend()
    plt.show()


def run_moga_bundling():
    """Runs the Multi-Objective GA (MOGA) with full setup isolation."""

    # 1. Isolation: Clean up previous definitions if they exist
    if hasattr(creator, 'FitnessMax'): del creator.FitnessMax
    if hasattr(creator, 'FitnessMulti'): del creator.FitnessMulti
    if hasattr(creator, 'Individual'): del creator.Individual

    # 2. Define Components for Multi-Objective GA
    creator.create("FitnessMulti", base.Fitness, weights=(1.0, -1.0))
    creator.create("Individual", list, fitness=creator.FitnessMulti)

    toolbox = base.Toolbox()
    toolbox.register("attr_int", random.randint, 0, BUNDLE_OPTIONS - 1)
    toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_int,
n=1)
    toolbox.register("population", tools.initRepeat, list, toolbox.individual)

    # Register core operations
    toolbox.register("evaluate", evaluate_multi_objective)
    toolbox.register("mate", cx_int_placeholder)
    toolbox.register("mutate", mut_int_flip, indpb=0.5)
    toolbox.register("select", tools.selNSGA2)

    NGEN = 500
    MU = 100
    LAMBDA = 200

    pop = toolbox.population(n=MU)
    hof = tools.ParetoFront()

    # --- Statistics Setup for Convergence Plot (MOGA) ---
    stats = tools.Statistics(lambda ind: ind.fitness.values)
    stats.register("max_output", np.max, axis=0)
    stats.register("min_imbalance", np.min, axis=1)
    logbook = tools.Logbook()
    # ----------------------------------------------------

    # Run the MOGA (NSGA-II)
    pop, log = algorithms.eaMuPlusLambda(pop, toolbox, mu=MU, lambda_=LAMBDA, cxpb=0.1,
```

```
mutpb=0.8,
                                ngen=NGEN, stats=stats, halloffame=hof, verbose=False)

    # Extract Pareto Front data
    output_values = [ind.fitness.values[0] for ind in hof]
    imbalance_values = [ind.fitness.values[1] for ind in hof]

    # --- Pareto Front Plot ---
    plt.figure(figsize=(10, 6))
    plt.scatter(imbalance_values, output_values, color='#c62828', label='Pareto Optimal
Solutions')
    plt.title('MOGA Pareto Front: Output vs. Imbalance (Trade-Off)')
    plt.xlabel('Workload Imbalance (Std Dev of Time-Per-Piece, seconds)')
    plt.ylabel('Daily Output (Pieces)')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.legend()
    plt.show()

    # --- MOGA Convergence Plot (Fixed Presentation) ---
    max_output_values = log.select("max_output")
    min_imbalance_values = log.select("min_imbalance")

    fig, ax1 = plt.subplots(figsize=(10, 6))

    # Plot Max Output (Fitness 1) on primary axis
    line1 = ax1.plot(log.select("gen"), max_output_values, "g-", label="Max Output (Pieces)")
    ax1.set_xlabel("Generation (Iterations)") # Clarified x-axis label
    ax1.set_ylabel("Max Output (Pieces)", color="g")
    ax1.tick_params(axis="y", labelcolor="g")
    ax1.grid(True, linestyle='--', alpha=0.6)

    # Plot Min Imbalance (Fitness 2) on secondary axis
    ax2 = ax1.twinx()
    line2 = ax2.plot(log.select("gen"), min_imbalance_values, "r-")
    ax2.set_ylabel("Min Imbalance (seconds)", color="r")
    ax2.tick_params(axis="y", labelcolor="r")

    # Manually set y-ticks for the secondary axis to remove clutter
    imbalance_min = np.min(min_imbalance_values)
    imbalance_max = np.max(min_imbalance_values)
    imbalance_ticks = np.linspace(imbalance_min, imbalance_max, 5)
    ax2.set_yticks(imbalance_ticks)
    ax2.set_ylim(imbalance_min * 0.9, imbalance_max * 1.1)

    # Combine legends
    lines = line1 + line2
    labels = [l.get_label() for l in lines]
    #ax1.legend(lines, labels, loc="upper right")
```

```
    plt.title("MOGA Convergence: Max Output and Min Imbalance")
    plt.show()
    # ----------------------------------------------------------------------


    print("\n--- MOGA Pareto Front Solutions ---")
    print(f"Found {len(hof)} non-dominated solutions.")

    # --- Print top solutions for analysis ---
    print("Example Solutions (Output, Imbalance):")

    # Get all unique bundle indices found on the Pareto Front
    unique_bundles = sorted(list(set(ind[0] for ind in hof)))

    for i in unique_bundles:
        # Find the representative individual for this bundle type
        rep_ind = next(ind for ind in hof if ind[0] == i)
        output, imbalance = rep_ind.fitness.values
        bundle_name = [f'op{j+1}-op{j+2}' for j in range(BUNDLE_OPTIONS)][i]

        print(f"  Bundle: {bundle_name} | Output: {output:.2f} pcs | Imbalance:
{imbalance:.2f} s")
    print("-----------------------------------")


# --- 4. Main Execution Block ---

if __name__ == "__main__":

    # Run the standard GA and plot convergence
    run_ga_bundling()

    # Run the MOGA and plot the Pareto Front
    print("\n--- Running Multi-Objective Genetic Algorithm (MOGA) ---")
    run_moga_bundling()
```

GA Code for Triple Operation Clustering

```
import random
import pandas as pd
import numpy as np
from deap import base, creator, tools, algorithms
import matplotlib.pyplot as plt # Import for plotting

# --- 1. Data Processing ---
data = {
    'op1': [12.2, 12.8, 12.2, 14.0, 12.5, 12.2, 12.4, 12.4, 12.5],
    'op2': [15.9, 16.7, 15.9, 18.3, 16.2, 15.9, 16.1, 16.1, 16.2],
    'op3': [14.7, 15.4, 14.7, 16.8, 15.0, 14.7, 14.8, 14.8, 15.0],
```

```
    'op4': [12.9, 13.5, 12.9, 14.7, 13.1, 12.9, 13.0, 13.0, 13.1],
    'op5': [17.1, 18.0, 17.1, 19.7, 17.5, 17.1, 17.3, 17.3, 17.5],
    'op6': [25.1, 26.3, 25.1, 28.8, 25.6, 25.1, 25.3, 25.3, 25.6],
    'op7': [30.6, 32.1, 30.6, 35.1, 31.2, 30.6, 30.9, 30.9, 31.2],
    'op8': [17.7, 18.6, 17.7, 20.4, 18.1, 17.7, 17.9, 17.9, 18.1],
    'op9': [14.7, 15.4, 14.7, 16.8, 15.0, 14.7, 14.8, 14.8, 15.0],
    'op10': [13.5, 14.1, 13.5, 15.4, 13.8, 13.5, 13.6, 13.6, 13.7],
    'op11': [20.8, 21.8, 20.8, 23.9, 21.2, 20.8, 21.0, 21.0, 21.2]
}

df = pd.DataFrame(data)
avg_process_times = df.mean().to_dict()

total_working_seconds = 13 * 3600
num_operators_per_shift = 11
NUM_PROCESSES = 11

# Process keys and times as a list for easy indexing
process_keys = [f'op{i}' for i in range(1, NUM_PROCESSES + 1)]
process_times_list = [avg_process_times[key] for key in process_keys]
BUNDLE_OPTIONS = 5 # P1-P2 up to P5-P6 (indices 0 to 4)
# MOGA WEIGHTS (Set 1)
weights=(1.0, -1.0)
# Goal: Maximize Output, Minimize Bundle Time (Complexity)

def evaluate_multi_objective(individual):
    # ... calculates effective_times ...

    # O1: Maximize Daily Output (using bottleneck_time)
    daily_output = total_working_seconds / bottleneck_time

    # O2: Minimize Bundle Time (Get the time of the single bundled task)
    bundle_index = individual[0]
    bundle_time = process_times_list[bundle_index] + process_times_list[bundle_index + 1]

    return daily_output, bundle_time
# --- 2. DEAP Helper Functions (Defined once) ---

def calculate_effective_times(bundle_index):
    """Helper function to calculate the time for the 10 effective stations."""
    effective_times_per_unit = []
    bundled_op_index_start = bundle_index

    for i in range(NUM_PROCESSES):
        time = process_times_list[i]

        if i == bundled_op_index_start:
            # Bundled Task Time = Op(i) + Op(i+1)
```

```
            bundle_time = time + process_times_list[i+1]
            effective_times_per_unit.append(bundle_time)
        elif i == bundled_op_index_start + 1:
            # Skip the second task in the bundle
            continue
        elif i == 6: # OP7 (index 6) is fixed at 2 workers
            effective_times_per_unit.append(time / 2)
        else:
            # All other tasks run with 1 worker
            effective_times_per_unit.append(time / 1)

    return np.array(effective_times_per_unit)


def evaluate_bundling_output(individual):
    """Calculates the total daily output (Max Fitness)."""
    effective_times = calculate_effective_times(individual[0])
    bottleneck_time = max(effective_times)
    daily_output = total_working_seconds / bottleneck_time
    return daily_output,

def evaluate_multi_objective(individual):
    """Calculates daily output (maximize) and workload imbalance (minimize)."""
    effective_times = calculate_effective_times(individual[0])

    # Objective 1: Daily Output (based on bottleneck)
    bottleneck_time = max(effective_times)
    daily_output = total_working_seconds / bottleneck_time

    # Objective 2: Workload Imbalance (standard deviation of the effective times)
    imbalance = np.std(effective_times)

    return daily_output, imbalance

# Mutation and Crossover functions (simple integer flipping/placeholder)
def mut_int_flip(individual, indpb):
    if random.random() < indpb:
        individual[0] = random.randint(0, BUNDLE_OPTIONS - 1)
    return individual,

def cx_int_placeholder(ind1, ind2):
    return ind1, ind2

# --- 3. GA Execution Functions (Isolated Setups) ---

def run_ga_bundling():
    """Runs the Standard GA (Single Objective) with full setup isolation."""

    # 1. Isolation: Clean up previous definitions if they exist
```

```
    if hasattr(creator, 'FitnessMax'): del creator.FitnessMax
    if hasattr(creator, 'FitnessMulti'): del creator.FitnessMulti
    if hasattr(creator, 'Individual'): del creator.Individual

    # 2. Define Components for Single Objective GA
    creator.create("FitnessMax", base.Fitness, weights=(1.0,))
    creator.create("Individual", list, fitness=creator.FitnessMax)

    toolbox = base.Toolbox()
    toolbox.register("attr_int", random.randint, 0, BUNDLE_OPTIONS - 1)
    toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_int,
n=1)
    toolbox.register("population", tools.initRepeat, list, toolbox.individual)

    # Register core operations
    toolbox.register("evaluate", evaluate_bundling_output)
    toolbox.register("mate", cx_int_placeholder)
    toolbox.register("mutate", mut_int_flip, indpb=0.5)
    toolbox.register("select", tools.selTournament, tournsize=2)

    POP_SIZE = 20
    NGEN = 50

    pop = toolbox.population(n=POP_SIZE)
    hof = tools.HallOfFame(1)

    # Statistics Setup for Convergence Plot
    stats = tools.Statistics(lambda ind: ind.fitness.values[0])
    stats.register("max", np.max)
    stats.register("avg", np.mean)
    logbook = tools.Logbook()

    # Run the Simple GA
    pop, log = algorithms.eaSimple(pop, toolbox, cxpb=0.1, mutpb=0.8, ngen=NGEN,
                                   stats=stats, halloffame=hof, verbose=False)

    # --- Display Results and Plotting ---
    best_bundle_index = hof[0][0]
    optimized_daily_output, = evaluate_bundling_output(hof[0])
    bundle_names = [f'op{i+1}-op{i+2}' for i in range(BUNDLE_OPTIONS)]
    chosen_bundle_name = bundle_names[best_bundle_index]

    print("\n--- OPTIMIZATION RESULTS (Genetic Algorithm) ---")
    print(f"Optimal Task Bundle to Maximize Output: {chosen_bundle_name}")
    print(f"Total Daily Output: {optimized_daily_output:.2f} pieces")

    # --- Convergence Graph Section ---
    max_fitness_values = log.select("max")
```

```
    avg_fitness_values = log.select("avg")

    plt.figure(figsize=(8, 4))
    plt.plot(range(NGEN + 1), max_fitness_values, marker='o', linestyle='-', color='#00695c',
markersize=3, label='Max Fitness (Best Solution)')
    plt.plot(range(NGEN + 1), avg_fitness_values, linestyle='--', color='#90A4AE',
label='Average Fitness (Population Mean)')

    plt.title('Standard GA Convergence (Maximizing Output)')
    plt.xlabel('Generation (Iterations)') # Clarified x-axis label
    plt.ylabel('Daily Output (Pieces)')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.legend()
    plt.show()
```

MOGA Code for Triple Operation Clustering

```
import random
import pandas as pd
import numpy as np
from deap import base, creator, tools, algorithms
import matplotlib.pyplot as plt # Import for plotting

# --- 1. Data Processing ---
data = {
    'op1': [12.2, 12.8, 12.2, 14.0, 12.5, 12.2, 12.4, 12.4, 12.5],
    'op2': [15.9, 16.7, 15.9, 18.3, 16.2, 15.9, 16.1, 16.1, 16.2],
    'op3': [14.7, 15.4, 14.7, 16.8, 15.0, 14.7, 14.8, 14.8, 15.0],
    'op4': [12.9, 13.5, 12.9, 14.7, 13.1, 12.9, 13.0, 13.0, 13.1],
    'op5': [17.1, 18.0, 17.1, 19.7, 17.5, 17.1, 17.3, 17.3, 17.5],
    'op6': [25.1, 26.3, 25.1, 28.8, 25.6, 25.1, 25.3, 25.3, 25.6],
    'op7': [30.6, 32.1, 30.6, 35.1, 31.2, 30.6, 30.9, 30.9, 31.2],
    'op8': [17.7, 18.6, 17.7, 20.4, 18.1, 17.7, 17.9, 17.9, 18.1],
    'op9': [14.7, 15.4, 14.7, 16.8, 15.0, 14.7, 14.8, 14.8, 15.0],
    'op10': [13.5, 14.1, 13.5, 15.4, 13.8, 13.5, 13.6, 13.6, 13.7],
    'op11': [20.8, 21.8, 20.8, 23.9, 21.2, 20.8, 21.0, 21.0, 21.2]
}

df = pd.DataFrame(data)
avg_process_times = df.mean().to_dict()

total_working_seconds = 13 * 3600
num_operators_per_shift = 11
NUM_PROCESSES = 11

# Process keys and times as a list for easy indexing
process_keys = [f'op{i}' for i in range(1, NUM_PROCESSES + 1)]
process_times_list = [avg_process_times[key] for key in process_keys]
BUNDLE_OPTIONS = 5 # P1-P2 up to P5-P6 (indices 0 to 4)
```

```
# --- 2. DEAP Helper Functions (Defined once) ---

def calculate_effective_times(bundle_index):
    """Helper function to calculate the time for the 10 effective stations."""
    effective_times_per_unit = []
    bundled_op_index_start = bundle_index

    for i in range(NUM_PROCESSES):
        time = process_times_list[i]

        if i == bundled_op_index_start:
            # Bundled Task Time = Op(i) + Op(i+1)
            bundle_time = time + process_times_list[i+1]
            effective_times_per_unit.append(bundle_time)
        elif i == bundled_op_index_start + 1:
            # Skip the second task in the bundle
            continue
        elif i == 6: # OP7 (index 6) is fixed at 2 workers
            effective_times_per_unit.append(time / 2)
        else:
            # All other tasks run with 1 worker
            effective_times_per_unit.append(time / 1)

    return np.array(effective_times_per_unit)


def evaluate_bundling_output(individual):
    """Calculates the total daily output (Max Fitness)."""
    effective_times = calculate_effective_times(individual[0])
    bottleneck_time = max(effective_times)
    daily_output = total_working_seconds / bottleneck_time
    return daily_output,

def evaluate_multi_objective(individual):
    """Calculates Daily Output (O1) and Final Process Time (O2) for Set 3."""

    # --- O1: Maximize Daily Output ---
    effective_times = calculate_effective_times(individual[0])
    bottleneck_time = max(effective_times)
    daily_output = total_working_seconds / bottleneck_time

    # --- O2: Minimize Final Process Time (P9, P10, P11) ---
    # These tasks are not part of the bundle choices (P1-P6) and are fixed at 1 worker each.
    # Indices 8, 9, 10 correspond to OP9, OP10, OP11 in the process_times_list
    final_process_time = process_times_list[8] + process_times_list[9] +
process_times_list[10]

    # The actual optimization will be driven by the output, but this ensures the MOGA
```

```
framework is sound.
    return daily_output, final_process_time

# Mutation and Crossover functions (simple integer flipping/placeholder)
def mut_int_flip(individual, indpb):
    if random.random() < indpb:
        individual[0] = random.randint(0, BUNDLE_OPTIONS - 1)
    return individual,

def cx_int_placeholder(ind1, ind2):
    return ind1, ind2

# --- 3. GA Execution Functions (Isolated Setups) ---

def run_ga_bundling():
    """Runs the Standard GA (Single Objective) with full setup isolation."""

    if hasattr(creator, 'FitnessMax'): del creator.FitnessMax
    if hasattr(creator, 'FitnessMulti'): del creator.FitnessMulti
    if hasattr(creator, 'Individual'): del creator.Individual

    creator.create("FitnessMax", base.Fitness, weights=(1.0,))
    creator.create("Individual", list, fitness=creator.FitnessMax)

    toolbox = base.Toolbox()
    toolbox.register("attr_int", random.randint, 0, BUNDLE_OPTIONS - 1)
    toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_int,
n=1)
    toolbox.register("population", tools.initRepeat, list, toolbox.individual)

    toolbox.register("evaluate", evaluate_bundling_output)
    toolbox.register("mate", cx_int_placeholder)
    toolbox.register("mutate", mut_int_flip, indpb=0.5)
    toolbox.register("select", tools.selTournament, tournsize=2)

    POP_SIZE = 20
    NGEN = 50

    pop = toolbox.population(n=POP_SIZE)
    hof = tools.HallOfFame(1)

    stats = tools.Statistics(lambda ind: ind.fitness.values[0])
    stats.register("max", np.max)
    stats.register("avg", np.mean)

    pop, log = algorithms.eaSimple(pop, toolbox, cxpb=0.1, mutpb=0.8, ngen=NGEN,
                                   stats=stats, halloffame=hof, verbose=False)
```

```
    # --- Display Results and Plotting ---
    best_bundle_index = hof[0][0]
    optimized_daily_output, = evaluate_bundling_output(hof[0])
    bundle_names = [f'op{i+1}-op{i+2}' for i in range(BUNDLE_OPTIONS)]
    chosen_bundle_name = bundle_names[best_bundle_index]

    print("\n--- OPTIMIZATION RESULTS (Genetic Algorithm) ---")
    print(f"Optimal Task Bundle to Maximize Output: {chosen_bundle_name}")
    print(f"Total Daily Output: {optimized_daily_output:.2f} pieces")

    # --- Convergence Graph Section ---
    max_fitness_values = log.select("max")
    avg_fitness_values = log.select("avg")

    plt.figure(figsize=(8, 4))
    plt.plot(range(NGEN + 1), max_fitness_values, marker='o', linestyle='-', color='#00695c',
markersize=3, label='Max Fitness (Best Solution)')
    plt.plot(range(NGEN + 1), avg_fitness_values, linestyle='--', color='#90A4AE',
label='Average Fitness (Population Mean)')

    plt.title('Standard GA Convergence (Maximizing Output)')
    plt.xlabel('Generation (Iterations)')
    plt.ylabel('Daily Output (Pieces)')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.legend()
    plt.show()


def run_moga_bundling():
    """Runs the Multi-Objective GA (MOGA) for Objective Set 3."""

    if hasattr(creator, 'FitnessMax'): del creator.FitnessMax
    if hasattr(creator, 'FitnessMulti'): del creator.FitnessMulti
    if hasattr(creator, 'Individual'): del creator.Individual

    # WEIGHTS ADJUSTED FOR SET 3: Maximize Output (1.0) and Minimize Final Process Time (-1.0)
    creator.create("FitnessMulti", base.Fitness, weights=(1.0, -1.0))
    creator.create("Individual", list, fitness=creator.FitnessMulti)

    toolbox = base.Toolbox()
    toolbox.register("attr_int", random.randint, 0, BUNDLE_OPTIONS - 1)
    toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_int,
n=1)
    toolbox.register("population", tools.initRepeat, list, toolbox.individual)

    toolbox.register("evaluate", evaluate_multi_objective)
    toolbox.register("mate", cx_int_placeholder)
    toolbox.register("mutate", mut_int_flip, indpb=0.5)
    toolbox.register("select", tools.selNSGA2)
```

```
    NGEN = 500
    MU = 100
    LAMBDA = 200

    pop = toolbox.population(n=MU)
    hof = tools.ParetoFront()

    # --- Statistics Setup for Convergence Plot (MOGA) ---
    stats = tools.Statistics(lambda ind: ind.fitness.values)
    stats.register("max_output", np.max, axis=0)
    stats.register("min_final_time", np.min, axis=1) # Tracks O2 (Final Process Time)
    logbook = tools.Logbook()
    # ----------------------------------------------------

    pop, log = algorithms.eaMuPlusLambda(pop, toolbox, mu=MU, lambda_=LAMBDA, cxpb=0.1,
mutpb=0.8,
                                ngen=NGEN, stats=stats, halloffame=hof, verbose=False)

    # Extract Pareto Front data
    output_values = [ind.fitness.values[0] for ind in hof]
    final_time_values = [ind.fitness.values[1] for ind in hof]

    # --- Pareto Front Plot ---
    plt.figure(figsize=(10, 6))
    # Plot Final Process Time vs Output
    plt.scatter(final_time_values, output_values, color='#c62828', label='Pareto Optimal
Solutions')
    plt.title('MOGA Pareto Front: Final Process Time vs. Output (Set 3)')
    plt.xlabel('Final Process Time (OP9+OP10+OP11 sum, seconds)')
    plt.ylabel('Daily Output (Pieces)')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.legend()
    plt.show()

    # --- MOGA Convergence Plot (Fixed Presentation) ---
    max_output_values = log.select("max_output")
    min_final_time_values = log.select("min_final_time")

    fig, ax1 = plt.subplots(figsize=(10, 6))

    # Plot Max Output (O1) on primary axis
    line1 = ax1.plot(log.select("gen"), max_output_values, "g-", label="Max Output (Pieces)")
    ax1.set_xlabel("Generation (Iterations)")
    ax1.set_ylabel("Max Output (Pieces)", color="g")
    ax1.tick_params(axis="y", labelcolor="g")
    ax1.grid(True, linestyle='--', alpha=0.6)
```

```
    # Plot Min Final Process Time (O2) on secondary axis
    ax2 = ax1.twinx()
    line2 = ax2.plot(log.select("gen"), min_final_time_values, "r-", label="Min Final Process
Time (seconds)")
    ax2.set_ylabel("Min Final Process Time (seconds)", color="r")
    ax2.tick_params(axis="y", labelcolor="r")

    # Manually set y-ticks for the secondary axis to remove clutter
    time_min = np.min(min_final_time_values)
    time_max = np.max(min_final_time_values)
    time_ticks = np.linspace(time_min, time_max, 5)
    ax2.set_yticks(time_ticks)
    ax2.set_ylim(time_min * 0.9, time_max * 1.1)

    # Combine legends
    lines = line1 + line2
    labels = [l.get_label() for l in lines]
    #ax1.legend(lines, labels, loc="upper right")

    plt.title("MOGA Convergence: Max Output and Min Final Process Time")
    plt.show()

    # --- Print top solutions for analysis ---
    print("\n--- MOGA Pareto Front Solutions ---")
    print(f"Found {len(hof)} non-dominated solutions.")

    # Get all unique bundle indices found on the Pareto Front
    unique_bundles = sorted(list(set(ind[0] for ind in hof)))

    print("Example Solutions (Output, Final Process Time):")
    for i in unique_bundles:
        # Find the representative individual for this bundle type
        rep_ind = next(ind for ind in hof if ind[0] == i)
        output, final_time = rep_ind.fitness.values
        bundle_name = [f'op{j+1}-op{j+2}' for j in range(BUNDLE_OPTIONS)][i]

        print(f"  Bundle: {bundle_name} | Output: {output:.2f} pcs | Final Time:
{final_time:.2f} s")
    print("-----------------------------------")


# --- 4. Main Execution Block ---

if __name__ == "__main__":

    # Run the standard GA and plot convergence
    run_ga_bundling()

    # Run the MOGA and plot the Pareto Front
```

```
print("\n--- Running Multi-Objective Genetic Algorithm (MOGA) ---")
run_moga_bundling()
```