Coursework Report: Modeling and Simulation in Logistics Planning

Assignment 01 - Production planning with discrete event simulation (DES) and optimization

Siddharthan Somasundaram (Matriculation number : 254304)[a,1]

[a] *Otto von Guericke University, Magdeburg, Germany*

Submitted as part of the coursework for Modeling and Simulation in Logistics Planning (Summer 2025).

Abstract—This report documents the development and optimization of a discrete-event simulation model for a medium-sized manufacturing company's production system, as part of an assignment in the Modelling and Simulation in Logistics Planning course. The objective is to improve delivery reliability by reducing total tardiness and makespan across 100 customer orders. Using Siemens Plant Simulation, a five-stage production process with varying product routes and setup times was modeled. A Genetic Algorithm (GA) was applied to optimize the order sequence and production lot sizes based on a multi-objective function. The impact of different weightings for tardiness and makespan was also explored. The results demonstrate that targeted optimization can significantly enhance production efficiency and delivery performance.

Contents

1. Introduction

Improving delivery reliability is a key objective for manufacturing companies, especially when delays significantly impact customer satisfaction and production efficiency. In this context, a simulation-based approach was adopted to analyze and optimize the production planning process of a medium-sized company that manufactures five different product types.

The company's production system consists of five sequential stages: Press, Laser Processing, Bending, Profiling, and Welding — each type of product follows a specific route and setup requirement. To evaluate and improve the current scheduling strategy, a discrete event simulation model was developed using Siemens Plant Simulation.

This report presents the structured approach followed to implement and analyze the simulation model. The work is divided into three main tasks:

- *Task 1:* focuses on building the baseline model using a fixed production sequence and evaluating performance using makespan and total tardiness.
- *Task 2:* introduces a Genetic Algorithm (GA) to optimize both the production order and the lot sizes by minimizing a weighted objective function:

$$\text{Min } z = 0.8 \times \text{Total Tardiness} + 0.2 \times \text{Makespan}.$$

- *Task 3:* investigates the effect of adjusting the weights in the objective function to explore potential improvements in the balance between delivery time and efficiency.

The following sections provide a detailed explanation of the modeling assumptions, the steps taken for implementation, the applied optimization strategy, and the insights gained from the simulation results. The simulation model used is also submitted alongside this report.

2. System Overview

The manufacturing system under study consists of a five-stage sequential production line designed to process a variety of metal components. Each stage represents a specific operation in the production workflow:

- *Press*, with a capacity of 2.
- *Laser Processing*, with a capacity of 2.
- *Bending*, with a capacity of 2.
- *Profiling*, with a capacity of 2.
- *Welding*, with a capacity of 2.

In the simulation, each stage is decoupled using buffers. Buffer capacities were assumed to be unlimited, and thus do not constrain the production flow. The process begins with raw materials entering through the Press station and ends when finished products are stored in product-specific buffers. These final buffers serve as the delivery point for customer orders, and products are considered fully processed once they arrive there.

3. Production Program and Product Routing

In the developed simulation model, five different product types were considered, each with its own routing through the production stages. These differences reflect real-world variations in manufacturing processes and setup requirements. The routing behavior of each product type was modeled as follows:

- *Product_A*, skips Laser Processing.
- *Product_B*, skips Profiling.
- *Product_C*, skips Profiling.
- *Product_D*, skips Profiling.
- *Product_E*, skips Welding.

These routing rules were integrated into the model to ensure accurate representation of processing logic.

4. Processing Times for Each Production Stage

Stage	Product_A	Product_B	Product_C	Product_D	Product_E
Press	120	150	300	180	150
Laser	0	240	300	300	300
Bending	360	390	360	60	240
Profiling	360	0	0	0	30
Welding	300	360	120	180	0

Table 1. Processing times (in seconds) for each production stage

5. Machine Setup Times

5.1. Press Setup Time

From/To	Product_A	Product_B	Product_C	Product_D	Product_E
–	600	600	600	600	600
Product_A	0	600	600	600	600
Product_B	600	0	600	600	600
Product_C	600	600	0	600	600
Product_D	600	600	600	0	600
Product_E	600	600	600	600	0

Table 2. Setup times for Press machine (in seconds)

5.2. Laser Setup Time

From/To	Product_A	Product_B	Product_C	Product_D	Product_E
–	0	0	0	0	0
Product_A	0	0	0	0	0
Product_B	0	0	240	120	240
Product_C	0	240	0	360	240
Product_D	0	480	240	0	360
Product_E	0	360	480	360	0

Table 3. Setup times for Laser machine (in seconds)

5.3. Bending Setup Time

From/To	Product_A	Product_B	Product_C	Product_D	Product_E
–	0	0	0	0	0
Product_A	0	0	0	0	0
Product_B	0	0	240	240	240
Product_C	0	240	0	120	360
Product_D	0	240	360	0	480
Product_E	0	360	360	120	0

Table 4. Setup times for Bending machine (in seconds)

5.4. Profiling Setup Time

From/To	Product_A	Product_B	Product_C	Product_D	Product_E
–	0	0	0	0	0
Product_A	0	0	0	0	360
Product_B	0	0	0	0	0
Product_C	0	0	0	0	0
Product_D	0	0	0	0	0
Product_E	240	0	0	0	0

Table 5. Setup times for Profiling machine (in seconds)

5.5. Welding Setup Time

From/To	Product_A	Product_B	Product_C	Product_D	Product_E
–	0	0	0	0	0
Product_A	0	360	360	120	0
Product_B	120	0	240	240	0
Product_C	360	480	0	120	0
Product_D	240	480	480	0	0
Product_E	0	0	0	0	0

Table 6. Setup times for Welding machine (in seconds)

Order Generation Logic

As part of the simulation model, 100 customer orders were generated programmatically at the beginning of the simulation using the **init** method. Each order consists of a minimum of one and a maximum of five different product types. The number of items per order follows a triangular distribution with an expected value of 3.

For every product type within an order, the demand quantity is randomly assigned using a uniform distribution ranging from 1 to 3 units. Additionally, each order is assigned a due time, randomly selected from a uniform distribution between 4 hours and 3 days.

To reflect realistic scheduling conditions, the orders are sorted in ascending order based on their completion time before being processed.

6. Task 1

6.1. Simulation Model Structure

The simulation model was built in **Siemens Plant Simulation** to replicate the company's production and order fulfillment system. The key components used in the model are listed and described below:

Data Tables:

- **CustomerOrders**: Store customer order data generated from **init** method including product types, quantities, and due dates.
- **ManufacturingTime**: Contains product-specific processing times for each production stage.
- **PressSetup**, **LaserSetup**, **BendingSetup**, **ProfilingSetup**, **WeldingSetup**: Tables containing machine-specific setup times for product transitions.
- **ProductSequence**, **OrderSequence**: Store sequences used for production and order execution.

Sources:

- **OrderItems**: Source for the flow that initiates the raw material for the production process.
- **Order**: Triggers delivery orders as per the **OrderSequence** data table.

ParallelStation:

- **Press**, **Laser**, **Bending**, **Profiling**, **Welding**: Simulate individual processing stages with Processing capacity 2 and processing times as per the product type.

AssemblyStation:

- **OrderProcessingUnit**: Collects completed order items and performs final assembly before delivery.

Buffers:

- **BufferLaser**, **BufferBending**, **BufferProfiling**, **BufferWelding**: Decouple each processing stage to allow asynchronous operation.
- **Product_A** to **Product_E**: Product-specific output buffers used to store finished products after completion of all processing stages. Each buffer holds completed units until they are retrieved for customer order fulfillment.

Methods:

- **Init**: Generates random customer orders at the start of the simulation.
- **Endsim**: Performs the final calculation of the fitness value after all orders have been processed.
- **OrderProcessing**, **ProductAttribute**, **Move**: Custom logic controlling order matching, attribute assignment, and product movement logic.

Global Variables:

- **MakeSpan**: Tracks the total time required to process all orders.
- **TotalTardiness**: Accumulates delay across all late orders.
- **Fitness**: Used during GA optimization to evaluate solution quality.
- **Total_Quantity_A** to **Total_Quantity_E**: Track the total quantity required per type of product for production.

These components collectively model the entire flow of materials and information from order creation to final delivery. Refer Siddharthan_Somasundaram_Assignment_01_DES.ssp file for the implementation.

6.2. Task 1: Model evaluation

Based on the implementation of the initial production program and by setting the random seed to 1 in the **EventController**, the simulation produced the following performance metrics:

- **Makespan**: `01:00:19:00` (1 day, 0 hours, 19 minutes, 0 seconds)
- **Total Tardiness**: `12:19:49:07` (12 days, 19 hours, 49 minutes, 7 seconds)
- **Fitness**: `10:11:07:06` (10 days, 11 hours, 7 minutes, 6 seconds)
- **Total Setup Time**: `01:38:00:00` (1 hour, 38 minutes, 0 seconds)

These results reflect the time required to complete all customer orders, the cumulative delay beyond the specified due dates, the overall fitness value of the scheduling solution, and the total time spent on setup operations.

7. Task 2: Optimization Using a Genetic Algorithm

To minimize the overall **makespan** and **total tardiness**, the Genetic Algorithm (GA) assistant was used in Siemens Plant Simulation. The optimization objective function is defined as:

$$\text{Minimize } z = 0.8 \cdot \text{Total Tardiness} + 0.2 \cdot \text{Makespan}$$

This weighted objective prioritizes timely order fulfillment (tardiness) while also considering total production time (makespan).

7.1. Optimization Strategy and Parameters

Two distinct optimization strategies were developed:

- **Approach 1:** Optimized `OrderSequence` and `LotSize`.
- **Approach 2:** Extended Approach 1 by also optimizing `ProductSequence`, allowing dynamic reordering of product types.

Each approach was tested in three experimental GA configurations to study the effect of varying the population size and number of generations:

- **Case 1:** 100 generations, 5 individuals
- **Case 2:** 500 generations, 5 individuals
- **Case 3:** 200 generations, 100 individuals

This setup allowed comparison across quick evaluations (case 1), deeper convergence attempts (Case 2), and high-diversity large population runs (Case 3).

7.2. Approach 1

Optimization Parameters

- **Order Sequence:** `root.OrderSequence` with 100 elements.
- **Lot Size:** `root.Lotsize`, ranging from 2 to 50.

A product type-based generation strategy was used instead of order-by-order generation. The products were produced in fixed alphabetical order ($A \to B \to C \to D \to E$), and their total required quantities were divided into lots. This reduced product switching and minimized setup time.

7.3. Approach 2

Optimization Parameters

- **Order Sequence:** `root.OrderSequence` with 100 elements.
- **Lot Size:** `root.Lotsize`, ranging from 2 to 50.
- **Product Sequence:** `root.ProductSequence`, allowing flexibility in switching and production of the product.

The inclusion of `ProductSequence` enabled more flexible sequencing, potentially reducing tardiness and increasing the diversity of the GA solution space.

7.4. Task 2: Model evaluation

7.4.1. Baseline (Task 1):

The unoptimized production plan yielded a makespan of `01:00:19:00` and a total tardiness of `12:19:49:07`, indicating poor scheduling performance and significant delivery delays.

7.4.2. Tardiness and Makespan Comparison:

Comparing the best-performing cases from both approaches, **Approach 1 – Case 3** achieved `00:00:00:00` total tardiness and a makespan of `01:02:10:30`. In contrast, **Approach 2 – Case 3** resulted in `03:10:12:37` tardiness and a slightly longer makespan of `01:03:41:30` (see Tables 7 and 8). This highlights that Approach 1 not only eliminated delivery delays but also maintained a more compact production schedule.

7.4.3. Fitness Evaluation:

The fitness value, which reflects the combined cost of makespan and tardiness, offers a clear comparison of overall performance between the two best cases:

- Approach 1 – Case 3: `00:05:14:06`
- Approach 2 – Case 3: `02:23:18:23`

Here, **Approach 1 – Case 3** significantly outperforms Approach 2 – Case 3. The zero tardiness and lower makespan led to the lowest fitness score overall, reinforcing its suitability under the defined optimization objective.

7.4.4. Setup Time (Only for Reference):

Although not directly included in the optimization, setup time provides further insight into process efficiency:

- Approach 1 – Case 3: `00:12:58:00`
- Approach 2 – Case 3: `01:01:44:00`

Among the best cases, Approach 1 also required considerably less setup time. This further supports its advantage, as it reduces overall operational effort and transition delays between lots.

7.4.5. Lot Size Influence on Optimization

An important trend observed from the comparison tables (Tables 7 and 8) is the impact of lot size on optimization outcomes. In both approaches, larger lot sizes do not consistently lead to better fitness; instead, optimal fitness depends on a balance between lot size and resulting setup time.

- In **Approach 1**, **Case 3** (lot size = 6) achieved the lowest fitness value of `00:05:14:06`, with zero tardiness and a reduced setup time of `00:12:58:00`, compared to **Case 2** (lot size = 11) which had a higher fitness of `00:10:46:44` and longer setup time (`00:22:12:00`).
- In **Approach 2**, **Case 3** (lot size = 10) achieved the best fitness of `02:23:18:23`, better than Cases 1 and 2 (lot size = 5) which showed significantly higher fitness values (`04:10:50:33` and `04:01:10:36`, respectively).

This pattern suggests that an optimal lot size—rather than simply a larger one—can help reduce setup time and align better with due dates, thus improving the combined objective of minimizing makespan and tardiness in the fitness function.

Case	Fitness (D:H:M:S)	Makespan (D:H:M:S)	Tardiness (D:H:M:S)	Setup Time (D:H:M:S) (for reference)	Lot Size	Best Gen/Ind
1	01:15:05:39	01:03:16:00	01:18:03:03	00:22:12:00	6	Gen 094 / Ind 05
2	00:10:46:44	01:03:16:00	00:06:39:26	00:22:12:00	11	Gen 448 / Ind 07
3	00:05:14:06	01:02:10:30	00:00:00:00	00:12:58:00	6	Gen 096 / Ind 2

Table 7. Task 2: Approach 1 - Comparison of Optimization Results for GA Cases

Case	Fitness (D:H:M:S)	Makespan (D:H:M:S)	Tardiness (D:H:M:S)	Setup Time (D:H:M:S) (for reference)	Lot Size	Best Gen/Ind
1	04:10:50:33	01:01:58:00	05:07:03:42	00:11:08:00	5	Gen 036 / Ind 03
2	04:01:10:36	01:01:58:00	04:18:58:46	00:11:08:00	5	Gen 296 / Ind 06
3	02:23:18:23	01:03:41:30	03:10:12:37	01:01:44:00	10	Gen 156 / Ind 60

Table 8. Task 2: Approach 2 - Comparison of Optimization Results for GA Cases

7.4.6. Conclusion:

Approach 1 – Case 3 is the clearly optimal configuration when comparing the best scenarios from each approach. It achieved:

- **Zero tardiness**,
- **Shortest makespan** (`01:02:10:30`),
- **Lowest fitness value** (`00:05:14:06`),
- **Lowest setup time** (`00:12:58:00`).

7.5. Experimental Determination of GA Parameters

To identify suitable parameters (population size and number of generations), three configurations were tested:

- **Case 1:** 100 generations, population size = 5 (low computational effort, rapid convergence).
- **Case 2:** 500 generations, population size = 5 (emphasizes the depth of generational refinement).
- **Case 3:** 200 generations, population size = 100 (high diversity, balanced exploration/exploitation).

Rationale for Parameter Selection

- **Population Size:** A small size (5) risks premature convergence but is computationally efficient. A larger size (100) improves diversity, allowing a broader exploration of the solution space.
- **Number of Generations:** Higher generations (500) allow deeper convergence but may overfit. Fewer generations (100–200) balance runtime and solution quality.
- **Trade-offs:** Larger populations require more computational resources per generation but reduce the risk of stagnation. More generations improve refinement, but demand longer run-times.

Results and Analysis

From the results presented in Tables 7 and 8, it is evident that Case 3 consistently yields the best fitness performance across both approaches. This supports the effectiveness of using a larger population size (100) and a moderate generation count (200) for achieving high-quality solutions.

Case 3 (population size = 100, generations = 200) outperformed others due to:

- **Diversity:** Larger populations explored more solutions, avoiding local optima.
- **Balanced refinement:** 200 generations provided sufficient iterations for convergence without excessive run-time.

8. Task 3: Objective Weight Adjustment for Better Balance

In Task 2, we observed that the optimal configuration (**Approach 1 – Case 3**) completely eliminated total tardiness (`00:00:00:00`), which ensured all orders were delivered on time. However, this came at the cost of a relatively high makespan (`01:03:16:00`), largely due to frequent product switching and fragmented lot processing.

This raised an important question: *Can we reduce the makespan further while still maintaining a reasonable level of tardiness?* To investigate this, we modify the weightings in the objective function as follows:

$$\min z = \alpha \cdot \text{Tardiness} + (1 - \alpha) \cdot \text{Makespan}$$

The goal is to shift the optimization balance—first equally, and then more heavily toward makespan reduction—to assess the impact on scheduling efficiency.

8.1. Approach

The same setup used in Task 2's optimal configuration (**Approach 1 – Case 3**) was used here to ensure comparability. Only the weightings of the objective function were varied. Two different configurations were tested:

- A balanced weighting: $z = 0.5 \cdot \text{Tardiness} + 0.5 \cdot \text{Makespan}$
  This aims to reduce makespan while giving equal priority to avoiding tardiness.
- A makespan-focused weighting: $z = 0.2 \cdot \text{Tardiness} + 0.8 \cdot \text{Makespan}$
  This prioritizes minimizing the production duration, even if slight tardiness is introduced.

8.2. Balanced Weighting Result ($\alpha = 0.5$)

Since the configuration in Task 2 already achieved zero tardiness, this first test explores whether an equal weighting between tardiness and makespan could reduce the makespan further while still controlling delays.

Generation	Fitness	Makespan	Total Tardiness	Setup Time
Gen 121 / Ind 169	02:17:13:33	01:03:28:00	04:06:59:07	00:18:42:00

Table 9. Result with Balanced Weighting ($\alpha = 0.5$)

As seen in Table 9, the makespan did not improve when compared to Task 2 (`01:03:28:00` vs `01:03:16:00`), and total tardiness significantly increased to over 4 days. This indicates that a 50/50 weighting does not yield a better balance than the original setup.

8.3. Makespan-Focused Weighting Result ($\alpha = 0.2$)

Since the balanced weighting did not yield an improvement in makespan, the next test further increased the emphasis on makespan (80%) while reducing the weight for tardiness (20%). The objective was to evaluate whether greater pressure on minimizing makespan would lead to a faster production schedule.

Generation	Fitness	Makespan	Total Tardiness	Setup Time
Gen 103 / Ind 120	00:21:17:12	01:02:36:30	00:00:00:00	00:14:42:00

Table 10. Result with Makespan-Focused Weighting ($\alpha = 0.2$)

Interestingly, although the objective function emphasized makespan more heavily, the result shows that the makespan (`01:02:36:30`) was not significantly reduced when compared to the baseline in Task 2 (`01:03:16:00`).

One possible explanation is the increase in **setup time** caused by frequent switching between products. The optimization may have scheduled production sequences that minimize delay but result in higher transition overhead, thereby offsetting the makespan gain expected from the weighting adjustment.

This highlights a key insight: *increasing the weight on makespan does not guarantee shorter schedules, as the optimizer may still produce sequences with frequent product switches. This leads to higher setup times, which in turn offset the expected reduction in makespan.*

9. Final Conclusion

This simulation and optimization study across Tasks 1–3 demonstrates how strategic scheduling decisions significantly affect production efficiency and delivery reliability.

In **Task 1**, the baseline production program revealed poor delivery performance with a makespan of `01:00:19:00` and a total tardiness of `12:19:49:07`, exposing inefficiencies in the default order execution.

In **Task 2**, the Genetic Algorithm (GA) was used to optimize both the order sequence and lot sizes. Among all tested configurations, **Approach 1 – Case 3** emerged as optimal, achieving *zero total tardiness*, a makespan of `01:02:10:30`, and a significantly reduced setup time. This proves that a well-designed lot-sizing strategy can eliminate delays and minimize setup-related inefficiencies. In contrast, Approach 2 added flexibility via product sequencing, but this led to longer setup times and reduced overall performance.

In **Task 3**, different objective function weightings were explored to assess whether a better trade-off between makespan and tardiness could be achieved. The results showed that:

- A balanced weighting ($\alpha = 0.5$) increased tardiness without improving makespan.
- A makespan-focused weighting ($\alpha = 0.2$) slightly improved the makespan (`01:02:36:30`) but at the cost of more complex production sequences and longer setup times.

Therefore, it was concluded that a better balance between makespan and total tardiness could *not* be achieved through weight adjustment alone, as setup time remains a limiting factor. Simply increasing the emphasis on makespan in the optimization objective does not guarantee shorter production schedules due to the added setup overhead. This suggests that setup times become the bottleneck, and once tardiness is eliminated.