University of Nottingham Malaysia

MATH4091

Optimising Inventory Management with Mixed Integer Linear Programming (Group A2)

Authors:	*Student Numbers:*
Leong Cheng Hou	20509168
Ewen Lim Yuan Khai	20510188
Atifa Mohamed	20512872
Sieh Ming Kuan	20518335
Janelle Wing Yan Liew	20517995

March 26, 2025

Abstract

Inventory management is integral to a successful business, particularly for perishable goods where stock imbalances can lead to increased costs and customer dissatisfaction. This study proposes a Mixed Integer Linear Programming (MILP) model to optimise ordering decisions for Gigi Coffee across its central warehouse and multiple retail outlets. Drawing on Bill of Materials, Master Material, and Inventory Balance datasets, the formulation incorporates supplier lead times, minimum order quantities, shelf-life constraints, holding costs, and wastage costs. The objective is to ensure demand fulfilment while minimising total inventory costs through dynamic adjustments to order sizes and frequencies. Performance is benchmarked against conventional (r, Q) and (s, S) policies. Although all approaches successfully prevent stockouts, the MILP model achieves substantially lower total inventory costs by effectively reducing both holding and purchase costs. These findings underscore the value of a data-driven optimisation approach over fixed-threshold methods, offering a more resilient and cost-effective strategy for managing perishable goods. Future work should focus on developing methods to verify the optimality of solutions and incorporate real-world constraints, such as stochastic lead times and supplier limitations, to enhance practical applicability.

Contents

Notations

P_i	Purchase cost for ingredient i.
H_i	Holding cost for ingredient i.
w_i	Wastage cost for ingredient i.
SS_i	Safety stock for ingredient i.
m_i	Minimum order quantity for ingredient i.
M	Maximum order quantity for all ingredients.
$D_{i,t}$	Demand for ingredient i in period t.
$I_{initial}$	Initial inventory level.
L_i	Lead time for ingredient i.
SL_i	Shelf life for ingredient i.
$Q_{i,t}$	Order quantity of ingredient i in period t.
$I_{i,t}$	Inventory level of ingredient i in period t.
$W_{i,t}$	Wastage quantity of ingredient i in period t.
$Y_{i,t}$	$\begin{cases} 1, & \text{if replenishment is done for ingredient } i \text{ in period } t. \\ 0, & \text{otherwise.} \end{cases}$

1 Introduction

Effective inventory planning is paramount for food and beverage businesses, ensuring consistent product availability while mitigating waste caused by perishable goods. Gigi Coffee, a rapidly expanding brand in this sector, has worked aggressively in recent months to reduce inventory levels through frequent reviews and monitoring initiatives. However, inventory planning remains an ongoing challenge, requiring a balance between maintaining sufficient stock to meet customer demand and preventing surplus accumulation.

To address these challenges, this project introduces a mathematical optimisation model designed to enhance inventory planning by integrating key factors such as supplier lead times, minimum order quantities, and shelf life of ingredients to determine the optimal ordering quantities and timing, thereby producing replenishment strategies tailored to each ingredient's requirements.

The project adopts a step-by-step methodology that utilises current inventory levels and accurate sales forecasts to enable data-driven decision-making based on real-time stock balances. Additionally, it incorporates safety stock considerations to account for demand fluctuations and supply uncertainties, ensuring operational continuity. By aligning forecasted sales projections with actual inventory data, this framework provides a structured, cost-effective approach to inventory management that streamlines stock control, minimises waste, and supports more informed decision-making.

The paper is organised as follows. Section 2 provides a review of the relevant literature. Section 3 details the available data. Section 4 describes the data preprocessing steps and the formulation of the optimisation model. Section 5 presents the performance evaluation results of the model. Finally, Section 6 concludes the paper with recommendations for future research.

2 Literature Review

2.1 Inventory Optimisation Models in Research

Inventory management has been extensively explored in operations research, with various mathematical models designed to optimise stock levels while reducing costs, playing a crucial role in enhancing decision-making and improving efficiency [1]. Traditional approaches, such as the Economic Order Quantity (EOQ) model, provide analytical solutions for balancing ordering and holding costs, while more advanced techniques, such as Mixed Integer Linear Programming (MILP), offer greater flexibility in handling complex inventory constraints. These methods form the basis of this research, which aims to develop a structured inventory model that effectively balances cost considerations and demand constraints.

2.2 Effectiveness of Optimisation Models in Research

Mathematical optimisation techniques, particularly Mixed Integer Linear Programming (MILP), are widely applied in inventory management for their ability to optimise order quantities and reorder timings while minimising overall costs, but due to their ability to handle both continuous and discrete decision variables, it is especially valuable for complex inventory decisions.

Kusuma and Hakim [2] combined MILP with ABC classification, a widely used inventory control method that categorises stock into Category A (high-value, high-priority items), Category B (moderate-value, moderate-priority items), and Category C (low-value, low-priority items). The paper aims to address the excessive material storage many companies tend to adopt to prevent shortages from occurring. However, the large amount of storage leads to high holding costs. Their model prioritised optimising inventory levels for Category A materials, which, despite making up a smaller portion of total inventory, account for the highest costs and require closer monitoring to prevent stockouts or excessive holding expenses. By integrating ABC classification with MILP, their study optimised order quantities and reorder scheduling to mitigate demand fluctuations and supplier lead time uncertainties.

Another study applied a MILP model within a multi-echelon supply chain [3] with a periodic review (s, S) policy to optimise inventory management. By integrating ordering, holding, and transportation costs into a MILP framework, this approach ensures inventory is replenished at predefined intervals, reducing the need for continuous monitoring while maintaining service levels. The study demonstrated that coordinating order timing across multiple echelons can reduce total inventory costs while improving supply chain efficiency. By leveraging MILP for structured decision-making, it effectively balances cost efficiency and demand fulfilment while minimising stock imbalances. The insights from these studies have influenced the formulation of this research's objective function, which aims to optimise inventory decisions while ensuring a cost-effective and responsive supply chain.

The findings from these studies underscore the advantages of using mathematical optimisation to enhance inventory control. By minimising excess stock and maintaining sufficient supply, this approach reduces unnecessary stockholding while ensuring a stable and reliable inventory system. These benefits reinforce MILP's effectiveness as a tool for cost-efficient inventory management, making it a valuable method for businesses seeking to optimise their supply chain operations.

However, neither study has empirical validation or real-world implementation of its models and does not include a comparative evaluation of their model's performance against traditional inventory methods or real-world data. The absence of simulation testing, sensitivity analysis or case study validation limits the assessment of their practical effectiveness.

3 Description of Data Available

The inventory optimisation model is developed using three key datasets provided by OneCreators, accessed via SQL Server Management Studio (SSMS). These datasets, namely Bill of Materials (BOM), Master Material, and Inventory Balance, offer distinct yet complementary information essential for forming a robust inventory optimisation model.

The BOM dataset outlines the ingredient composition for 364 products, distinguishing between primary ingredients (BOM Level 1) and secondary ingredients (BOM Level 2) in cases where premix formulations (PRM-XXXX) are involved. Key variables include the Product Code, Ingredient Code, Unit of Measurement (UOM), and required ingredient quantities. This dataset will be used to track and calculate the ingredient usage for each product.

The Master Material dataset contains detailed information on 181 ingredients, including purchase price, lead time, conversion rate of 1 unit of BOM UOM, and minimum order quantity (MOQ) details. This dataset defines the financial and operational constraints of the inventory management problem.

Finally, the Inventory Balance dataset provides a snapshot of inventory levels on 31 December 2024 across the main warehouse and 156 retail outlets. It details the ingredient codes and inventory levels at each location for all 181 ingredients, which will be taken as the initial inventory levels for all scenarios.

4 Methodology

4.1 Data Cleaning and Preprocessing

Effective inventory management depends on accurate and structured data. This section details the preprocessing steps applied to the **Master Material**, **Inventory Balance** and **BOM** datasets to enhance consistency and accuracy. The key steps include removing irrelevant codes, eliminating duplicates, standardising units, and updating missing values to improve data quality and streamline inventory operations.

4.1.1 Master Material Dataset

The Master Material dataset undergoes several cleaning steps to improve quality. First, unwanted ingredient codes are removed if they contain specific substrings that are not referenced in the Bill of Materials (BOM). The excluded substrings include GBK, GMD, and GRTD, along with specific ingredient codes such as GIG-0009, GIG-0063, GIG-0064, GIG-0065, GIG-0066, GIG-0067, GIG-0069, GIG-0071, GIG-0073, GIG-0087, GIG-0105, GIG-0338, GIG-0339, GIG-0340, GIG-0341, GIG-0472, and GIG-0485.

Second, duplicate entries are eliminated to maintain data consistency. When an ingredient appears multiple times with varying purchase prices, only the most recent entry is preserved.

This ensures that inventory planning and cost calculations are based on the latest and most accurate pricing information.

4.1.2 Inventory Balance Dataset

To maintain consistency with the Master Material dataset, specific cleaning actions are applied to the Inventory Balance dataset. The same ingredient codes excluded from the Master Material dataset (GBK, GMD, GRTD, ..., GIG-0485) are also removed from the Inventory Balance records to ensure uniformity across datasets. Additionally, inventory records associated with store codes HQ002 and HQ003 are excluded, as they are not relevant to the analysis.

4.1.3 Bill of Materials (BOM) Dataset

The BOM dataset is cleaned and standardised to improve inventory tracking and procurement accuracy. Any ingredient code containing the substring "GKW" is removed to eliminate irrelevant entries. Missing values within the BOM are cross-referenced with the Master Material dataset and updated to ensure completeness and consistency across all datasets. Standardising the unit of measurement (UOM) for each ingredient ensures that all entries remain consistent, maintaining uniformity and accuracy in inventory tracking and procurement. If an ingredient appears multiple times for the same product, the total quantity is summed, and duplicate rows are removed.

4.2 Preparing Cleaned Data for Model Integration

Following the preprocessing and cleaning of inventory data, the next step involves converting the structured data into code to ensure it is properly formatted and ready for integration into the inventory model.

4.2.1 Extracting Product and Ingredient Data from BOM

The first step involves extracting product codes and their corresponding ingredients from the BOM. Unique product codes are identified and each product is mapped to its respective ingredients, including the ingredient quantity and UOM.

Additionally, the extracted data is structured hierarchically, distinguishing between raw ingredients and premixes (PRM components). This categorisation ensures that premixes are properly accounted for in subsequent calculations.

4.2.2 Standardising and Converting BOM Quantities

Once product-ingredient relationships are established, ingredient quantities in the BOM are converted into standardised purchase units using data from the Master Material. Each ingredient in the BOM is matched to its corresponding record in the Master Material file to retrieve the purchase UOM and the conversion rate to purchase UOM. Using these conversion

rates, ingredient quantities are represented in their appropriate purchase units before they are used in inventory projections to ensure consistency in inventory calculations.

4.2.3 Calculating Ingredient Requirements from Sales Data

The final stage involves determining the total ingredient requirements based on store-specific sales data. This process begins by identifying the products sold and their total sales quantities. Using the standardised and converted BOM ingredient quantities, the amount of each ingredient needed per product is calculated by multiplying the sales quantity of each product by its corresponding ingredient quantities. These values are then aggregated across all products to generate a comprehensive list of ingredients required for each store.

An additional step is performed for premix components (PRM). Batch conversions are applied using predefined batch-size multipliers, ensuring accurate scaling of ingredient needs. The premix components are then further broken down into their respective ingredient compositions, allowing for precise calculation of the final ingredient requirements.

4.3 Problem Formulation

The inventory management problem is formulated as a MILP model. The model aims to minimise the overall inventory cost across all ingredients and periods for a single retailer. Each retailer places orders based on local demand, and to fulfil these orders, inventory from the central warehouse is transferred to the respective retailer. To optimise warehouse inventory levels, the combined orders from all retailers are aggregated and treated as the warehouse demand in the model. Therefore, the mathematical model is formulated as follows.

4.3.1 Assumptions

The model operates under several key assumptions that define the behaviour of demand, ordering policies, inventory flow, and other key factors in the system. It is assumed that there is no lead time between the warehouse and retailers, allowing for immediate replenishment when needed and no orders exist outside of the planning horizon. Supplier lead times are fixed and remain unchanged once established, ensuring predictable restocking intervals. The shelf life of ingredients decreases only while stored at the warehouse or retail locations. Ingredients in the Inventory Balance dataset are assumed to be fresh. New stock always arrives fresh, and inventory is managed using a first-in, first-out approach, where older stock is used before newer stock.

The model assumes that shortages do not occur, ensuring all demand is met without disruption. Suppliers operate without delivery restrictions, meaning orders can be placed and fulfilled as needed without scheduling limitations. Additionally, the model does not account for returns or defective ingredients, assuming that all received stock is in usable condition. Storage capacity at both the warehouse and retail locations is assumed to be unlimited, eliminating space constraints from inventory decision-making. These assumptions provide a structured

foundation that streamlines the optimisation process while ensuring inventory efficiency and cost minimisation.

4.3.2 Objective Function

The objective function in (1) aims to minimise total costs over the planning horizon by integrating purchasing, holding, and wastage costs. This includes the cost of acquiring the ingredient ($P_i Q_{i,t}$), the cost of carrying inventory ($H_i I_{i,t}$), and the cost associated with ingredient expiring and going to waste ($w_i W_{i,t}$). The objective function is as follows:

$$\text{Minimise Total Cost} = \sum_{i=1}^{N} \sum_{t=1}^{T} \Big(P_i Q_{i,t} + H_i I_{i,t} + w_i W_{i,t} \Big), \tag{1}$$

where i represents the ingredients ($i \in \{1, 2, \ldots, N\}$) and t represents the time periods ($t \in \{1, 2, \ldots, T\}$). The decision variables $Q_{i,t}, I_{i,t}, W_{i,t}$, and $Y_{i,t}$, as previously defined, are used to determine the optimal order quantity, inventory level, wastage, and replenishment decisions, respectively.

The model is subject to the following constraints:

$$Q_{i,t} \geq m_i \times Y_{i,t}, \quad \forall i \in N, \forall t \in T. \tag{2}$$

$$Q_{i,t} \leq M \times Y_{i,t}, \quad \forall i \in N, \forall t \in T. \tag{3}$$

$$I_{i,t} \geq SS_{i,t}, \quad \forall i \in N, \forall t \in T. \tag{4}$$

$$I_{i,0} = I_{\text{initial}} + Q_{i,0} - D_{i,0}, \quad \forall i \in N. \tag{5}$$

$$I_{i,t} = I_{i,t-1} + Q_{i,(t-L_i)} - D_{i,t} - W_{i,t}, \quad \forall i \in N, t > 0. \tag{6}$$

$$I_{i,t} \geq D_{i,t}, \quad \forall i \in N, \forall t \in T. \tag{7}$$

$$Q_{i,t},\ I_{i,t},\ W_{i,t} \geq 0, \quad \forall i \in N, \forall t \in T. \tag{8}$$

$$L_i \geq 0, \quad \forall i \in N. \tag{9}$$

$$Y_{i,t} \in \{0, 1\}, \quad \forall i \in N, \forall t \in T. \tag{10}$$

The model is further constrained by Equations (2)–(10), which ensure the decision variables

remain feasible and practical. The constraints in (2) and (3) impose minimum and maximum order quantities once an order decision has been made (via the binary variable $Y_{i,t}$). The requirement in (4) maintains a specified safety-stock level, preventing inventory from dropping below a certain threshold. Constraint (5) establishes the initial inventory balance by combining the starting inventory with any new orders and then subtracting demand. Constraint (6) continues this process each period by adding orders while deducting demand and accounting for wastage. To ensure that stock levels are sufficient to meet demand at all times, constraint (7) requires that the inventory on hand in each period be at least equal to the demand. Constraint (8) and (9) enforces non-negativity, as negative orders, inventories or lead time have no real-world interpretation and are therefore disallowed. Finally, constraint (10) ensures that $Y_{i,t}$ is binary (0 or 1), representing whether an order is placed for a given item in a given period. This constraint captures the decision to order or not.

Although shelf life is not explicitly formulated as a mathematical constraint, it is a crucial factor in the model. Incorporating shelf life directly into the mathematical formulation would require tracking the age of each inventory unit across multiple periods, significantly increasing the model's complexity. This would necessitate additional state variables and constraints, making the formulation cumbersome. Instead, the shelf-life logic is embedded in the computational implementation, where expired ingredients are flagged and accounted for dynamically when the shelf life of the ingredient is exceeded.

Thus, while the equations in (2)–(10) do not explicitly model shelf life, the approach ensures that inventory remains fresh by promoting timely replenishment and preventing overstocking.

4.3.3 Calculating Holding Costs

Holding costs are typically estimated using the following formula which assumes an annual carrying cost of 20%, a typical average for perishable goods [4].

$$\text{Inventory Holding Cost} = I \times c \times 20\%, \tag{11}$$

where
I is the average inventory level,
c is the purchase cost per unit of ingredient.

This formula has been modified to incorporate daily inventory levels instead of a fixed average. The annual carrying cost rate of 20% is converted into a daily rate by dividing it by 365 days, resulting in the adjusted formula:

$$\text{Daily Inventory Holding Cost} = I \times c \times \frac{20\%}{365}, \tag{12}$$

where
I is the daily inventory level,
c is the purchase cost per unit of ingredient.

However, it is important to note that estimating holding costs based solely on the purchase cost may not be entirely accurate, as an increase in the purchase cost does not necessarily lead to increased holding costs for the ingredient.

4.3.4 Calculating Safety Stock Levels

Since the optimisation model assumes no stockouts, it must guarantee sufficient inventory to meet the demand within the service period. Therefore, safety stock is introduced to buffer against demand fluctuations and ensure continuous ingredient availability. A common method to set safety stock level [5] is:

$$SS = z_\alpha \times \sigma_d \times \sqrt{L}, \tag{13}$$

where
z_α is the desired service factor,
σ_d is the demand deviation per period,
L is the lead time.

The z_α is set to 2.33, corresponding to a 99% service level. In other words, there is a 99% probability that the available inventory, including safety stock, will be sufficient to meet demand during the lead time. This service level is used in inventory management to reduce stockout risks but at the cost of higher holding costs. The formula also assumes that demand is normally distributed, but by the Central Limit Theorem, even if individual demand values are not normally distributed, the aggregated demand over multiple periods can be approximated by a normal distribution.

4.4 Model Implementation

The PuLP library in Python provides a high-level interface for formulating and solving linear programming (LP) and mixed-integer linear programming (MILP) problems, acting as a modelling interface that relies on third-party solvers such as CBC, GLPK, Gurobi, and CPLEX. This approach simplifies model development by allowing users to focus on problem formulation while leveraging established solver methodologies.

The objective function defined in Section 4.3.2 was implemented using PuLP, ensuring that the optimisation model aligns with the problem constraints and decision variables. The model was solved using the COIN-OR CBC (Coin-or Branch and Cut) solver, the default solver in PuLP. COIN-OR CBC is an open-source MILP solver that applies Branch-and-Cut algorithms—a hybrid method combining Branch-and-Bound and Cutting Plane techniques—to efficiently solve large-scale MILP problems.

The Cutting-plane algorithm begins by solving a simpler version of the problem, called linear relaxation, that ignores some of the integer constraints. If the solution is not a valid integer solution, additional constraints, known as fractional cuts or valid inequalities, are created

to remove the current solution while maintaining all valid integer solutions. This process is repeated until an integer solution is found.

On the other hand, the Branch-and-Bound algorithm is a divide-and-conquer algorithm. It divides the problem into subproblems (branching) and solves each one separately while removing sections that do not provide the optimal solution (bounding). The process repeats until an optimal solution is found. Combining these two algorithms allows the Branch-and-Cut algorithm to find the optimal solution through linear relaxations and strategic branching.

The optimisation model development follows an iterative workflow, where the problem is first defined, translated into a mathematical formulation, solved using an optimisation solver and then validated. If the solution does not meet the required conditions, the model is refined and re-solved until a valid outcome is achieved. Figure 1 illustrates this structured approach, highlighting the continuous refinement process to ensure model accuracy and feasibility.

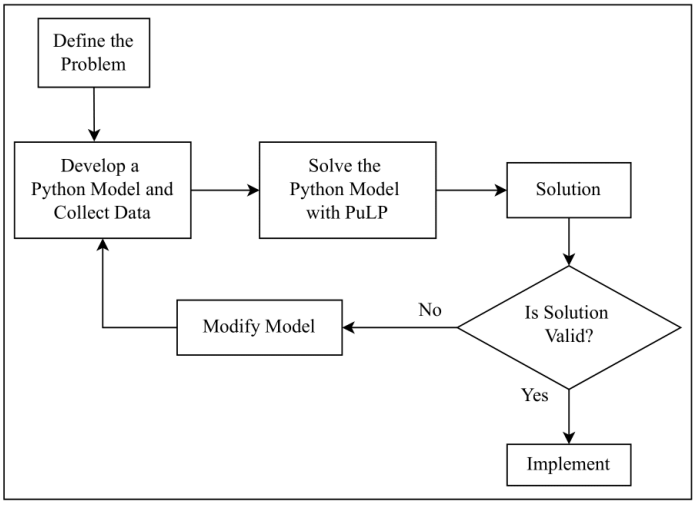

Figure 1: Development of Optimisation Model Workflow.

This iterative methodology ensures the optimisation model is systematically developed and continuously improved, leading to a reliable and effective decision-making framework. An example output of the optimisation model is shown in A.1.

4.5 Sales Forecasting to Inventory Optimisation

The process begins by obtaining a sales forecast for a defined planning horizon, establishing the expected demand for each store and forming the basis for all subsequent steps. Next, the ingredient breakdown code is executed, translating each store's product requirements into their ingredient components to identify ingredient needs. Using this breakdown, the daily ingredient demand for each store is calculated.

This daily ingredient demand is then inputted into the optimisation model, which generates an optimal ordering schedule tailored to each store. These individual store schedules are aggregated to determine the overall warehouse demand. A further optimisation step is performed at the warehouse level, resulting in a comprehensive warehouse ordering schedule. This final schedule specifies the amount and when supplies should be ordered from external suppliers to ensure that all stores can operate without disruptions. The logical flow of this process and the overall inventory network is given in Figure 2.

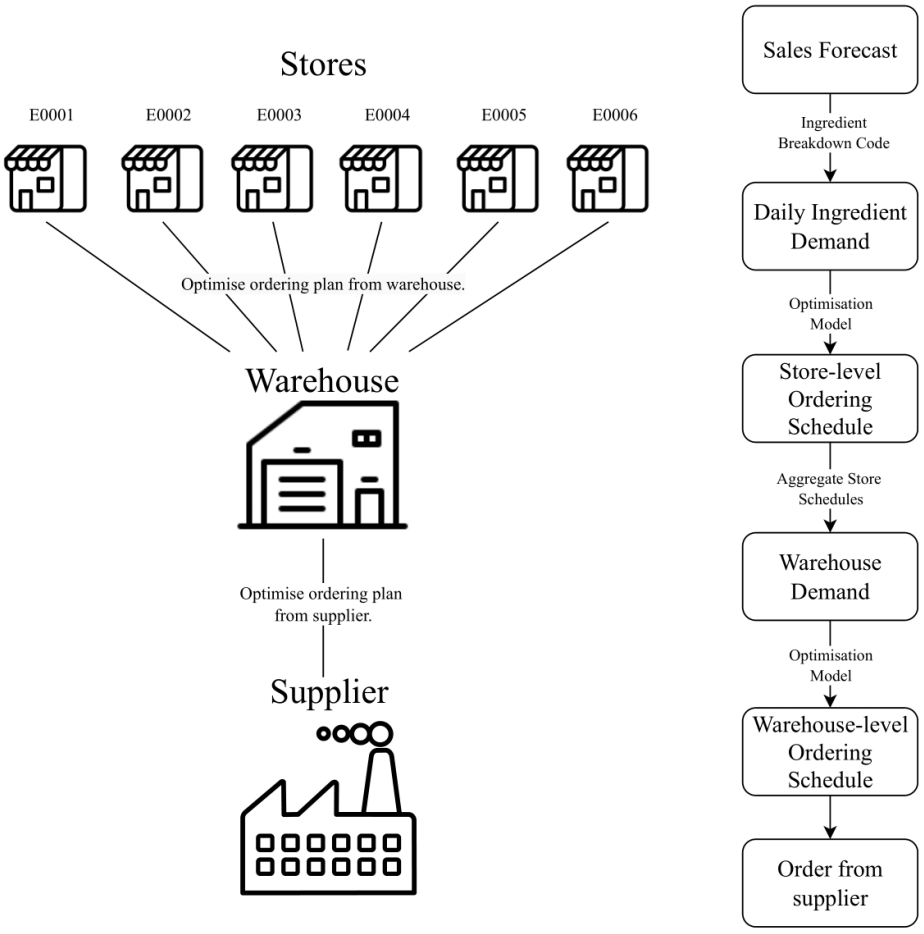

Figure 2: Sales Forecasting to Inventory Optimisation.

4.6 Performance Evaluation

To assess the performance of the optimisation model, it will be compared against two common inventory policies: the (r, Q) policy and the (s, S) policy. By benchmarking the optimisation model against these traditional methods, its ability to maintain inventory levels while minimising total inventory costs can be evaluated. The assessment will focus on Fresh Milk

(GIG-0017), an ingredient with a 60-day shelf life and no minimum order quantity, making it suitable to be used in traditional methods that do not account for shelf life or minimum order quantity.

Table 1: Parameters for Performance Evaluation on Ingredient GIG-0017.

Parameter Name	Parameter Value
Average Demand per Period (D_{avg})	13.58
Standard Deviation of Demand (D_{std})	5.9
Number of Time Periods (T)	60
Lead Time (L)	3
Purchase Cost (P)	8.9
Holding Cost (H)	Daily Inventory Level $\times P \times \frac{0.2}{365}$
Initial Inventory Level ($I_{initial}$)	$D_{avg} \times (L+1)$

Table 1 presents the parameters used across three models to ensure a fair comparison. Since the optimisation model assumes demand follows a normal distribution, D_{avg} and D_{std} were derived from historical data analysis. The planning horizon, T, was selected to prevent ingredient expiration. $I_{initial}$ was set to ensure sufficient starting inventory before new stock arrivals. Holding costs follow the same formula as Equation (12), while L and P values were sourced from the Master Material dataset.

4.6.1 (r, Q) Inventory Policy

The (r, Q) policy triggers a fixed quantity Q to order when the inventory level falls to or below the reorder point r. This ensures that replenishment occurs only when stock reaches a critical threshold, preventing stockouts while maintaining a predictable order quantity. However, it does not account for fluctuating demand patterns, which may lead to excess inventory during periods of low demand or insufficient stock during demand surges. The r and Q were computed through the Python library *stockpyl* using the parameters defined in Table 1 as inputs. Equations [6] used to calculate these values are provided in Appendix A.2.

Table 2: Parameters for (r, Q) Inventory Policy.

Parameter Name	Parameter Value
Reorder Point (r)	64.86
Order Quantity (Q)	226.09

4.6.2 (s, S) Inventory Policy

In contrast, the (s, S) policy replenishes the inventory up to a certain point S when the inventory level falls to or below the reorder point s. This ensures that inventory is restored to a target level rather than ordering a fixed quantity, allowing for more adaptive restocking based on current inventory levels. However, it may lead to variable order quantities, making it susceptible to high holding costs and potential wastage. The s and S were computed through the Python library *stockpyl* using the parameters defined in Table 1 as inputs. Equations [7] used to calculate these values are provided in Appendix A.3.

Table 3: Parameters for (s, S) Inventory Policy.

Parameter Name	Parameter Value
Reorder Point (s)	84.57
Order-up-to Level (S)	454.57

5 Results

This section presents a comparative analysis of the inventory policies using the optimisation model, the (r, Q) policy, and the (s, S) policy. The performance of these policies are evaluated based on inventory level trends over time and total inventory costs, as shown in Figure 3 and Table 4.

5.1 Inventory Policies Comparison

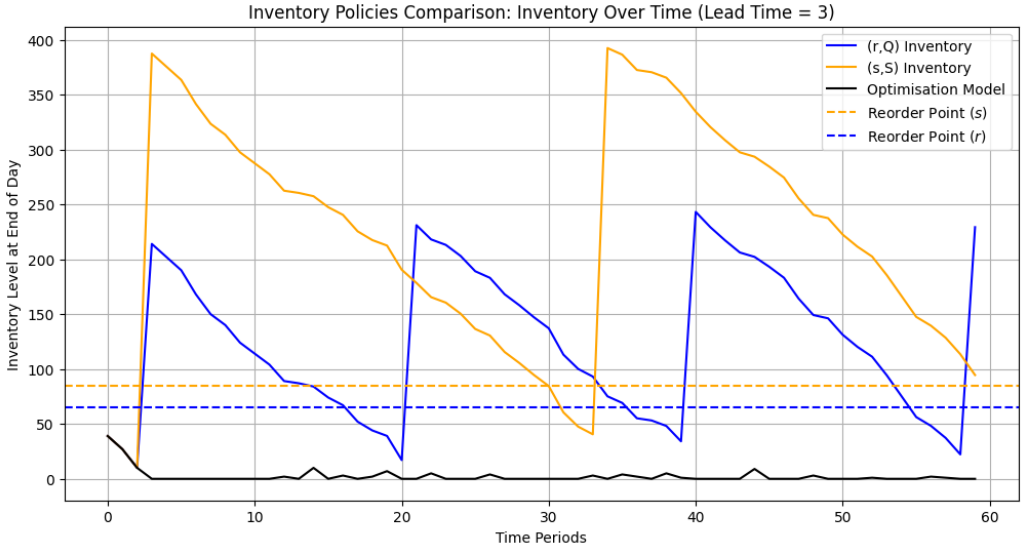

Figure 3: Inventory Policies Comparison.

Figure 3 illustrates the inventory levels at the end of the day over time for the inventory policies. All policies prevented stockouts, however, the (r, Q) policy (blue) and the (s, S) policy (orange) show high fluctuations in the inventory levels, while the optimisation model (black) maintains a more stable and lower inventory level throughout the time horizon. The (s, S) policy results in higher inventory levels due to replenishing up to S, leading to increased holding costs. The (r, Q) does not replenish as much inventory but still suffers from high inventory levels. Finally, the optimisation model maintains the lowest inventory levels while avoiding stockouts.

Table 4 shows the breakdown of the total inventory costs, which includes holding costs, stockout costs, and purchase costs for each inventory policy.

Table 4: Total Inventory Cost Breakdown for Inventory Policies.

Inventory Policy	Holding Costs	Stockout Costs	Purchase Costs	Total Costs
(r, Q)	36.03	0.00	8048.80	8084.83
(s, S)	65.01	0.00	8048.80	8113.81
Optimisation Model	0.68	0.00	6007.50	6008.18

While all policies managed to prevent stockouts, the (s, S) policy has the highest total cost ($8113.81) due to having more holding costs ($65.01). Since the inventory levels frequently reach S, storage costs accumulate over time. The (r, Q) policy has slightly lower costs ($8084.83) due to its moderate holding costs ($36.03). The optimisation model achieves the lowest total cost ($6008.18) by significantly reducing holding and purchase costs. By dynamically adjusting the order quantities, it maintains just enough inventory to prevent stockouts while minimising excessive holding costs.

These results demonstrate that the optimisation model outperforms both traditional policies by balancing order quantities and replenishment timing. The (s, S) policy is the most expensive, as it prioritises inventory availability over cost efficiency, leading to excessive stock levels. The (r, Q) policy is slightly more efficient but still relies on a fixed replenishment quantity, which does not always align with demand variations. In contrast, the optimisation model minimises costs while ensuring that inventory is maintained at optimal levels.

Additionally, none of the policies incurred stockout costs, indicating that all approaches successfully avoided shortages. However, while the traditional policies achieve this by holding surplus inventory, the optimisation model accomplishes it more efficiently by reducing excess stock while preventing shortages.

6 Conclusion and Discussion

6.1 Limitations

Despite the potential benefits of implementing an optimisation model for inventory management, several limitations affect its applicability and reliability. A key limitation is the difficulty in validating the model's outputs, as there is no definitive way to confirm that the solutions are truly optimal in real-world conditions. Without extensive testing against historical data or real-world scenarios, the model's recommendations may not always align with actual supply chain dynamics, leading to uncertainty about its effectiveness.

Additionally, the assumption of a perfect supplier that is always available, operates without restrictions and never experiences delays limits the model's realism. In reality, suppliers often face practical constraints such as production lead times, transportation disruptions, capacity restrictions, and non-working days, all of which can impact delivery reliability. Ignoring these factors creates an overly optimistic inventory plan that does not reflect real-world supply chain conditions, potentially leading to overestimated service levels and underestimated costs.

6.2 Future Considerations

Future improvements to the optimisation model should account for supplier constraints, such as work schedules, capacity limitations, and variability in lead times, to enhance its realism and practical applicability. Additionally, validating model outputs against historical data or real-world testing would improve reliability, while regularly assessing the model's performance under varying demand and supply conditions would ensure continuous relevance and accuracy.

6.3 Conclusion

This study has demonstrated the effectiveness of a Mixed Integer Linear Programming (MILP) approach in optimising inventory management for perishable goods. By incorporating supplier lead times, minimum order quantities, and shelf-life constraints, the model successfully reduces total inventory costs while ensuring demand fulfilment. Compared to traditional $(r,\ Q)$ and $(s,\ S)$ policies, the MILP model achieves superior cost efficiency by dynamically adjusting order quantities and replenishment schedules. Specifically, it reduces holding costs by maintaining lower inventory levels and minimises purchase costs through optimised ordering decisions.

Its ability to prevent stockouts while avoiding excessive inventory accumulation puts it ahead of the traditional methods that rely on fixed reorder points and quantities, which can lead to inefficient stock levels. In contrast, the MILP model leverages real-time demand fluctuations to determine optimal replenishment strategies, making it more adaptable to changing supply chain conditions.

Despite these benefits, certain limitations must be addressed to enhance the model's practical applicability. The assumption of deterministic supplier lead times and unlimited storage capacity does not fully reflect real-world constraints. Gigi Coffee should incorporate stochastic lead times, supplier capacity limits, and real-world case studies to validate the model's effectiveness under uncertain conditions. Overall, this research highlights the potential of mathematical optimisation in modern inventory management, providing a strong foundation for businesses seeking cost-effective strategies to manage perishable goods.

References

[1] Dipto Roy, Anamika Saha, and Ridita Ghosh. "A Review on Optimization Models for Inventory Management". In: *ResearchGate* (2022). URL: https://www.researchgate.net/publication/385863496_A_Review_on_Optimization_Models_for_Inventory_Management.

[2] R. I. Kusuma and I. M. Hakim. "Designing Inventory Models to Minimize Total Inventory Costs by Using Mixed Integer Linear Programming (MILP) in the Warehouse of MRO Materials". In: *IOP Conference Series: Materials Science and Engineering* 1003.1 (2020), p. 012100. DOI: 10.1088/1757-899X/1003/1/012100. URL: https://iopscience.iop.org/article/10.1088/1757-899X/1003/1/012100/pdf.

[3] J. J. Vicente. "Optimizing Supply Chain Inventory: A Mixed Integer Linear Programming Approach". In: *Systems* 13.1 (2023), p. 33. DOI: 10.3390/systems13010033. URL: https://www.mdpi.com/2079-8954/13/1/33.

[4] T. Kusomrosananan and N. Phumchusri. "Inventory Policy Improvement with Periodic Review for Perishable Goods: A Case Study of a Retail Coffee Shop in Thailand". In: *Engineering Journal* 28 (2024), p. 15. DOI: 10.4186/ej.2024.28.6.59. URL: https://engj.org/index.php/ej/article/view/4557.

[5] N. Vandeput. *Inventory Optimization: Models and Simulations*. De Gruyter, 2020. URL: https://www.researchgate.net/publication/343472734_Inventory_Optimization_Models_and_Simulations.

[6] Stockpyl Documentation. *Stockpyl RQ Approximation API*. Accessed: 24 March 2025. 2025. URL: https://stockpyl.readthedocs.io/en/latest/api/seio/rq.html#stockpyl.rq.r_q_eil_approximation.

[7] Stockpyl Documentation. *Stockpyl SS Power Approximation API*. Accessed: 24 March 2025. 2025. URL: https://stockpyl.readthedocs.io/en/latest/api/seio/ss.html#stockpyl.ss.s_s_power_approximation.

A Appendix

A.1 Example Optimisation Output

Ingredient	Order Date	Order Quantity	Arrival Date	Initial Inventory	Orders Arrived	Total Inventory	Final Inventory	Wastage Quantity
GIG-0017	01-01-2025	12	04-01-2025	55	0	55	39	0
GIG-0017	02-01-2025	12	05-01-2025	39	0	39	27	0
GIG-0017	03-01-2025	12	06-01-2025	27	0	27	10	0
GIG-0017	04-01-2025	22	07-01-2025	10	12	22	0	0
GIG-0017	05-01-2025	0	-	0	12	12	0	0
GIG-0017	06-01-2025	0	-	0	12	12	0	0
GIG-0017	07-01-2025	0	-	0	22	22	0	0

Table 5: Example Optimisation Output.

Table 5 presents an example of the inventory management process for a single ingredient, GIG-0017, without safety stock over 7 days in a single store, illustrating order placements, arrivals, and stock adjustments. The Order Date and Order Quantity columns indicate when and how many units should be requested, while the Arrival Date shows when they are expected to arrive. Initial Inventory captures the stock level at each day's start, and Orders Arrived reflects successfully delivered units. Total Inventory, the sum of the initial stock and any arrivals, provides a snapshot of all available units before consumption. Final Inventory reveals the remaining stock after usage, whereas Wastage Quantity tracks any losses from spoilage or other factors (with zero indicating no wastage). This example represents how inventory data is structured and monitored, providing insights into stock movements and enabling more effective inventory management strategies.

A.2 Computing (r, Q) Policies

The following equations calculate the r and Q parameters for (r, Q) policies, where λ is the mean demand per unit time.

$$r = F^{-1}\left(1 - \frac{Q \times \text{Holding Cost}}{\text{Stockout Cost} \times \lambda}\right) \tag{14}$$

$$Q = \sqrt{\frac{2\lambda[\text{Purchase Cost} + pn(r)]}{\text{Holding Cost}}} \tag{15}$$

These equations also define the (r, Q) policy under a normal demand distribution. Equation (14) determines the reorder point r, which is derived from the probability distribution of demand. The term $1 - \frac{Q \times \text{Holding Cost}}{\text{Stockout Cost} \times \lambda}$ represents the probability that the demand does not exceed r, effectively controlling the expected shortage risk. Equation (15) calculates the optimal

order quantity Q by considering the trade-off between ordering costs, holding costs, and stock-out costs. The equation is similar to the classical Economic Order Quantity (EOQ) models but extends further by including expected stockout costs, $pn(r)$, to better capture real-world inventory scenarios.

A.3 Computing (s, S) Policies

The following equations calculate the s and S parameters for (s, S) policies, where $\mu_L = \mu L$ and $\sigma_L^2 = \sigma^2 L$ are the mean and standard deviation of the lead-time demand.

$$Q = 1.30\mu^{0.494} \left(\frac{\text{Purchase Cost}}{\text{Holding Cost}} \right)^{0.506} \left(1 + \frac{\sigma_L^2}{\mu^2} \right)^{0.116} \tag{16}$$

$$z = \sqrt{\frac{Q}{\sigma_L} \frac{\text{Holding Cost}}{\text{Stockout Cost}}} \tag{17}$$

$$s = 0.973\mu_L + \sigma_L \left(\frac{0.183}{z} + 1.063 - 2.192z \right) \tag{18}$$

$$S = s + Q \tag{19}$$

These equations define the (s, S) policy under a normal demand distribution. Equation (16) calculates the optimal replenishment batch size Q, incorporating demand, cost parameters, and demand variability to balance ordering and holding costs. Equation (17) determines the safety stock factor z, which accounts for uncertainty in demand relative to stockout risk. Equation (18) establishes the reorder point s by considering the expected lead-time demand and variability, ensuring sufficient stock before replenishment. Finally, Equation (19) determines the order-up-to level S, which sets the maximum inventory level after an order, combining the reorder point s and the replenishment quantity Q.