Revenue Management - Retailer Game

Juan David Ovalle Felix Veaux Jean Pool Nieto
Mirza Abubacker

May 12, 2025

1 Introduction

In retail, a markdown refers to a deliberate price reduction on a product, typically aimed at accelerating inventory turnover while minimizing the overhead costs of holding unsold goods. Retailers must carefully balance the timing and depth of these markdowns—reducing prices too early or too aggressively can lead to profit loss, while delaying markdowns or overestimating demand may result in excess inventory that is costly to store or eventually unsellable.

This report explores the Retailer Game, developed by Ramandeep S. Randhawa, as a simulation tool to experiment with various markdown strategies. The objective is to identify an approach that minimizes the gap between actual revenue generated in each run and the revenue achievable under a Perfect Foresight Strategy—a theoretical benchmark where future demand is known in advance.

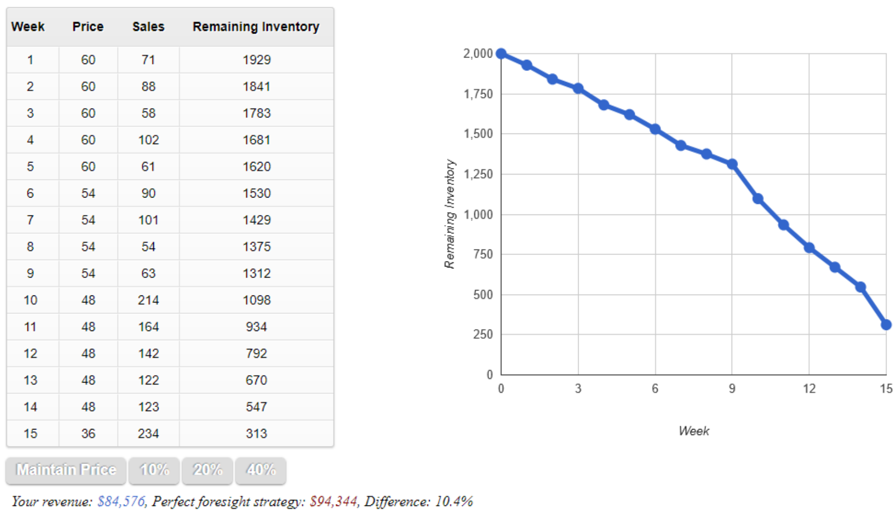
1.1 Objective

Develop a model to determine the optimal price markdown strategy that **maximizes total revenue** over a 15-week selling period.

1.2 Rules of the Game

1. **15 weeks** to sell all of the inventory (starting at 2000 in stock), leftover inventory is considered lost.
2. Price always starts at **$60**
3. Discounts can be applied at **10%, 20%, or 40%** only once. Applying 40% discount ends the game.
4. The price can only be maintained or marked **down**, not up

A general assumption here is that each run of the game represents a different item with different demands that affect sales resulting from the markdowns.



Week	Price	Sales	Remaining Inventory
1	60	71	1929
2	60	88	1841
3	60	58	1783
4	60	102	1681
5	60	61	1620
6	54	90	1530
7	54	101	1429
8	54	54	1375
9	54	63	1312
10	48	214	1098
11	48	164	934
12	48	142	792
13	48	122	670
14	48	123	547
15	36	234	313

Figure 1: Sample Run of Retailer Game

In this sample run, the difference between the actual revenue generated and the revenue generated with the perfect foresight strategy is 10.4%.

2 Approaches

2.1 Exhaustive Search of The Optimal Combination

Using the Selenium Python library, an exhaustive search was conducted by simulating every possible combination of price markdowns over the 15-week selling horizon. With three markdown options available—10%, 20%, and 40%—and 14 weeks in which markdown decisions could be applied (excluding the fixed initial price), a total of 680 unique price trajectories were generated. Each combination was simulated five times to account for variability, and the results were averaged to obtain a reliable performance estimate for each strategy.

The table below summarizes the aggregated results across all combinations:

Mean Score of Different %	11.61%
95% CI Score	(11.32%, 11.91%)
Standard Deviation	4.90

Table 1: Average Metrics Across All Combinations

The best combination was determined and produced these results across five runs:

Mean Score of Different %	2.68%
Standard Deviation	1.90

Table 2: Combo ID 366 Produced The Best Results In the Exhaustive Search

2.2 Linear Optimization

DynamicProgramming.ipynb

This approach involved defining an objective function, decision variables, and constraints and using an optimization model to find the optimal markdown strategy. The historical demand data excel file provided by the Retailer Game website was used to determine the median lifts of each price change. While the assumption was that the demand for each item varied, the demand lift would be static across all items. Using this assumption, the median lift was calculated for each markdown:

Markdown	Median Lift
10%	1.30
20%	1.79
40%	2.81

Table 3: Median lifts determined for markdown

Thus, the demand at each price point could be calculated using the initial demand observed on week 1 when the price is $60.

Prices	Demand
$60	Demand at 60 (Let's call it d60)
$54	1.30 * d60
$48	1.79 * d60
$36	2.81 * d60

Table 4: Demand calculated at each price point anchored by initial demand on week 1

2.2.1 Objective Function

$$\max_x \sum_{t=1}^{15} [60 \cdot d_{60} \cdot y_{t,60} + 54 \cdot d_{54} \cdot y_{t,54} + 48 \cdot d_{48} \cdot y_{t,48} + 36 \cdot d_{36} \cdot y_{t,36}]$$

- $d_{t,p}$: Demand at time t for price p
- $y_{t,p}$: Binary decision variable indicating whether price p is chosen at week t

2.2.2 Constraints

Based on the rules of the game, the constraints were defined as follows:

$$\sum_{p \in P} y_{t,p} = 1 \qquad \forall t \in T \quad \text{(Only one price per week)} \tag{1}$$

$$y_{1,60} = 1 \qquad \text{(Week 1 must be at \$60)} \tag{2}$$

$$\sum_{t \in T} \sum_{p \in P} p \cdot y_{t,p} \leq 2000 \qquad \text{(Total price level budget constraint)} \tag{3}$$

$$\sum_{\substack{p \in P \\ \text{idx}(p) \leq j}} y_{t,p} \geq \sum_{\substack{p \in P \\ \text{idx}(p) \leq j}} y_{t+1,p} \quad \forall t \in \{1, \ldots, 14\},\ \forall j \in \{1, 2, 3\} \quad \text{(Monotonicity constraint)} \tag{4}$$

2.2.3 Running the Model

The `optimize_monotonic_schedule(d60)` function builds and solves the optimization model using Gurobi. The function derives an initial demand value denoted by `d60` from the first week of the run, the model then chooses the optimal strategy for the rest of the weeks. The model uses binary decision variables $y_{t,p}$ to indicate whether the price $p \in \{60, 54, 48, 36\}$ is chosen in week $t \in \{2, \dots, 15\}$. The function returns the optimal price schedule across the 15 weeks and the corresponding total revenue under that strategy.

This model was run in batch on 100 simulated runs, and the results are summarized below.

Mean Score of Different %	2.31%
95% CI Score	(1.92%, 2.70%)

Table 5: Linear Optimization Model Metrics

2.3 Reinforcement Learning

Reinforcement learning (RL) model to optimize markdown pricing by choosing between the current price or markdowns, aiming to maximize revenue. It uses a Q-network trained offline with Conservative Q-Learning (CQL), which balances learning from historical data and avoiding overfitting to rare, uncertain strategies. Key input features include the current week, inventory, current price, and lift tiers which reflects how demand increases when prices drop.

The model adapts pricing based on the item's lift tier, allowing more aggressive markdowns for high-response items and more conservative actions otherwise. While effective in learning nuanced strategies, its performance is bounded by the quality and diversity of historical data. Simpler pricing strategies like rule-based or linear markdowns often show higher average performance, but they may leave more inventory unsold. Figures in the appendix show reward and inventory distributions across strategies, as well as the revenue gap between the RL model and the optimal benchmark.

The results of the RL model have an average of 25,000 gap against the optimal solution. The limitation of this strategy is lacking a hybrid approach to allow the model interact with the game directly instead of learning from historical data. Additionally, include a dynamic model to change the lift tier in each week based on previous results.

3 Optimal Approach

When comparing the results of all three approaches, the best results come from the **linear optimization model** with a mean score of **2.31%** across 100 runs. This model also allows for dynamic shifting of the strategy based on the demand observed in the first week, allowing for a more consistent performance when compared to the one-size-fits-all strategy of the best combination from the exhaustive search.

For illustrative purpose, a set of strategies devised by the linear optimization model based on the demand from the first week in intervals of 10 units is presented below.

Demand at Full Price ($60)	# $60	# $54	# $48	# $36
30	1	0	0	14
40	1	0	0	14
50	1	0	1	13
60	1	0	7	7
70	1	0	12	2
80	2	1	12	0
90	4	3	8	0
100	5	6	4	0
110	11	0	4	0
120	13	0	2	0
130	14	1	0	0

Table 6: Number of times each price is set versus demand at full price

4 Conclusion

This project explores multiple approaches to optimize markdown pricing strategies within the constraints of the Retailer Game. The exhaustive and optimization-based methods leverage deterministic assumptions about demand lifts, enabling near-optimal solutions under fixed conditions. In contrast, the reinforcement learning (RL) model demonstrates strong adaptability by learning from historical data, though its performance lags behind the theoretical optimum by an average revenue gap of $25,000. Its limitations stem from static lift assumptions and the lack of real-time interaction. Future improvements could focus on hybrid strategies that incorporate live feedback and dynamic demand estimation, enabling more responsive and effective markdown planning. By far, the linear optimization is the best approach to deal with markdown pricing strategies for future improve can be added dynamic optimization to adapt its respond based on each week results.

Appendix: Evaluation Figures

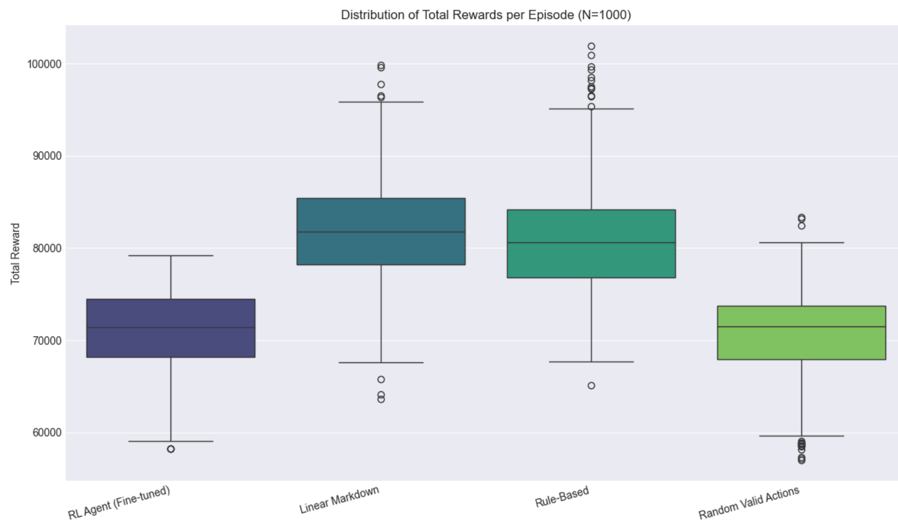


Figure 2: Distribution of Total Rewards per Episode (N=1000) in RL model

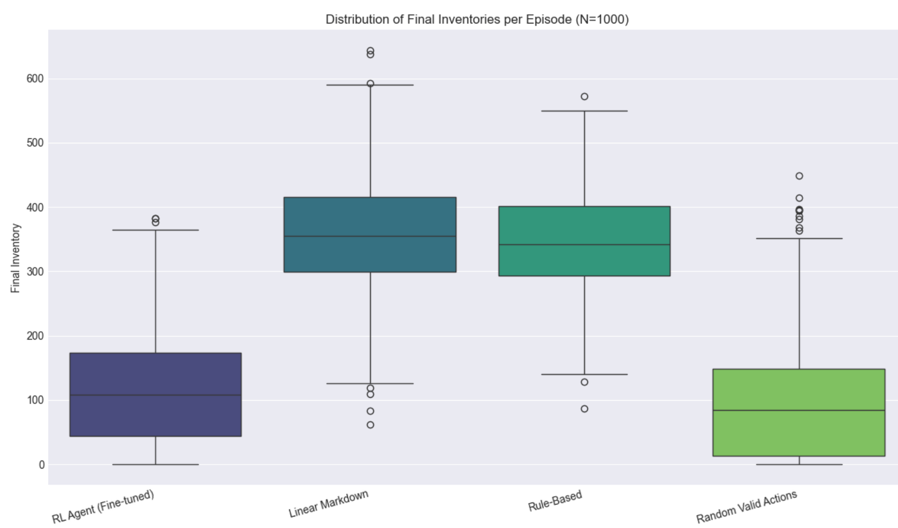


Figure 3: Distribution of Final Inventories per Episode (N=1000) in RL model

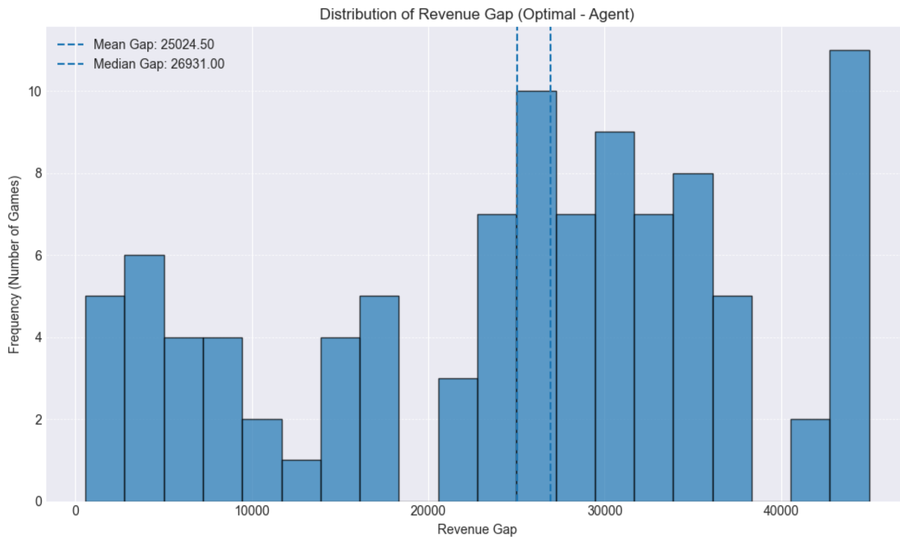


Figure 4: Distribution of Revenue Gap (Optimal - Agent) for 100 Games in RL model for Tier Lift 1