

DPRL-Assignment 1

Federico Donati, Filippo Donghi

November 2025

1 Problem Explanation and Assumptions

This assignment considers a finite-horizon dynamic programming problem over a selling season of 150 time periods. We start with an initial inventory of 5 items. In each time period t , demand is binary, $D_t \in \{0, 1\}$, and the probability of demand increases linearly over time:

$$P(D_t = 1) = \frac{t}{150}.$$

This means demand is very unlikely early in the season, and almost certain near the end.

In each time period, we may choose to order one additional unit. Delivery, however, is unreliable: the order arrives immediately, but only with probability 0.5. Selling one item yields a profit of 1, and each item in inventory incurs a holding cost of 0.1 per time unit. Leftover inventory at the end of the horizon has no value, and unmet demand is not backordered or penalized. The objective is to maximize the expected total revenue minus inventory costs.

We assume holding cost is charged at the end of each period. This is consistent with the idea that any newly arrived item is counted as being stored into the next step.

State and Action Space

The state space is defined by:

$$X = \{(x, t) \mid x \in \{0, 1, \dots, M\}, t \in \{1, \dots, 150\}\},$$

where x is the current number of inventory units and M is a chosen maximum inventory level. Since starting with 5 items and ordering at most one item per period with a 50% success rate leads to slow growth in inventory, M can be chosen reasonably to avoid unnecessary computational, and is equal to 15 in this case.

The action space is:

$$A = \{a_0, a_1\},$$

where a_0 means no order is placed and a_1 means ordering one item.

Dynamic Programming Formulation

Let $V_t(x)$ denote the maximum expected total reward from time t to the end of the horizon when the inventory level is x at the start of period t . Demand occurs with probability $p_t = \frac{t}{150}$, and when we choose to order (a_1), the item arrives with probability 0.5. Let the per-unit holding cost be $h = 0.1$, charged on the *end-of-period* inventory.

Terminal condition:

$$V_{151}(x) = 0 \quad \text{for all } x,$$

which simply closes the recursion after period 150.

To compute $V_t(x)$ we evaluate both actions:

1. Do not order (a_0):

$$Q_0(x, t) = p_t \left(\mathbb{I}(x > 0) \cdot 1 - h \max(x - 1, 0) + V_{t+1}(\max(x - 1, 0)) \right) + (1 - p_t) \left(-h x + V_{t+1}(x) \right).$$

2. Order one unit (a_1): Delivery succeeds with probability 0.5.

$$\begin{aligned} Q_1(x, t) = & \frac{1}{2} \left[p_t (\mathbb{I}(x + 1 > 0) \cdot 1 - h x + V_{t+1}(x)) + (1 - p_t) (-h(x + 1) + V_{t+1}(x + 1)) \right] \\ & + \frac{1}{2} \left[p_t (\mathbb{I}(x > 0) \cdot 1 - h \max(x - 1, 0) + V_{t+1}(\max(x - 1, 0))) + (1 - p_t) (-h x + V_{t+1}(x)) \right]. \end{aligned}$$

The dynamic programming recursion is:

$$V_t(x) = \max\{Q_0(x, t), Q_1(x, t)\}, \quad \pi_t(x) = \arg \max\{Q_0(x, t), Q_1(x, t)\}.$$

2 The Solution

Once the state and action spaces and parameters were defined, the solution was computed using two main functions:

1. **Expected Value Computation:** The function `compute_expected_value` determines the expected value of a state (inventory, time) given a particular action. For each action, we consider all possible outcomes:
 - whether demand occurs or not,
 - whether an attempted delivery succeeds or not (if ordering).

Each outcome is weighted by its probability, and the immediate reward (sales revenue minus holding cost) is added to the continuation value of the resulting state. This function allows us to evaluate how good a given action is in a particular state.

2. **Dynamic Programming Recursion:** The function `solve_dp` runs a backward recursion from $t = 150$ down to $t = 1$. For each state, it evaluates both possible actions using `compute_expected_value` and selects the action with the higher expected value. The output is a value table and a corresponding table of optimal actions (the optimal policy).

The result is a complete mapping from every (x, t) state to an optimal action.

3 The Simulation

To validate and interpret the optimal policy, we simulate the system behavior over 1000 episodes:

1. `simulate_single_episode` takes the system from $t = 1$ to $t = 150$, applying the optimal policy and sampling random demand and delivery outcomes.
2. `run_simulations` executes the above procedure 1000 times and collects the total realized reward for each run.

The resulting distribution of rewards can be used to estimate the expected revenue under the optimal policy and to visualize variability across supply and demand outcomes.

4 Results

The expected revenue under the optimal policy obtained from dynamic programming is

$$\mathbb{E}[R^*] = 33.61514015200629.$$

Over 1000 simulated episodes, the sample statistics are:

- Average simulated revenue: 33.431699999999999
- Standard deviation: 6.364002287083180
- Absolute difference from expected: 0.183440152006291

These values indicate that the simulation average is very close to the theoretical expectation, with a small discrepancy that is entirely consistent with sampling variability given the observed spread.

The optimal ordering policy over time exhibits a clear threshold pattern: ordering is preferred at low on-hand inventory and at later periods when demand probability is higher, while no-order is preferred when inventory is sufficient or time is early to avoid holding costs.

5 Discussion

Agreement between DP and simulation

The gap between the dynamic-programming expected revenue 33.61514015200629 and the simulated average 33.431699999999999 is modest relative to the standard deviation 6.364002287083180. Given $n = 1000$ runs and the observed dispersion, this deviation is expected from finite-sample effects rather than a modeling or coding error. The histogram in Figure 1 is unimodal and centered close to the DP benchmark, which supports the numerical correctness of the value function and policy.

Policy structure and intuition

The policy's qualitative shape matches intuition for a setting with increasing demand over time and positive holding costs:

1. When inventory is low and time is late (higher demand probability), ordering reduces stockout risk and increases expected sales, so the optimal action is to order.
2. When inventory is bigger or time is early (lower demand probability), ordering provides little marginal benefit and only increases expected holding costs, so the optimal action is to refrain from ordering.

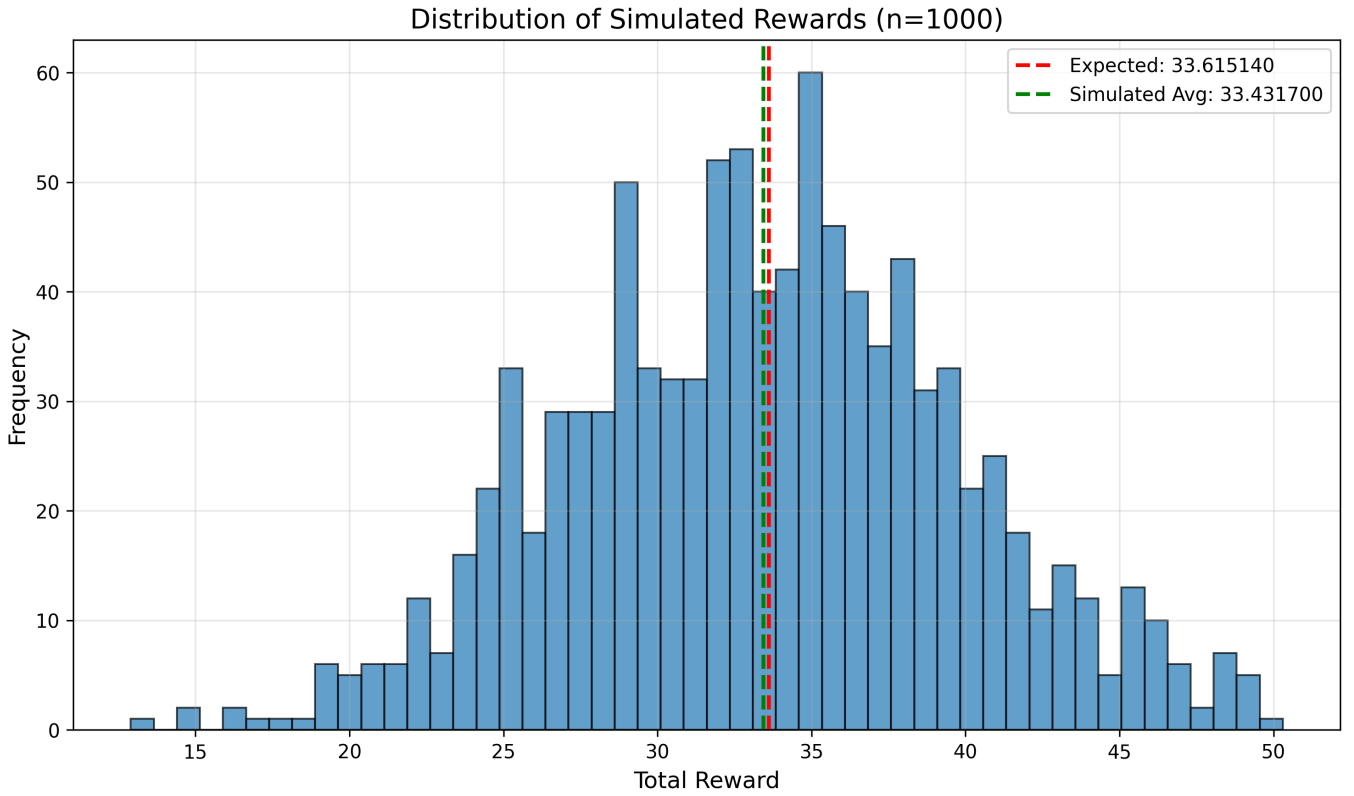


Figure 1: Distribution of simulated total rewards over $n = 1000$ episodes. The red dashed line marks the DP expected value 33.61514015200629; the green dashed line marks the simulated average 33.431699999999999.

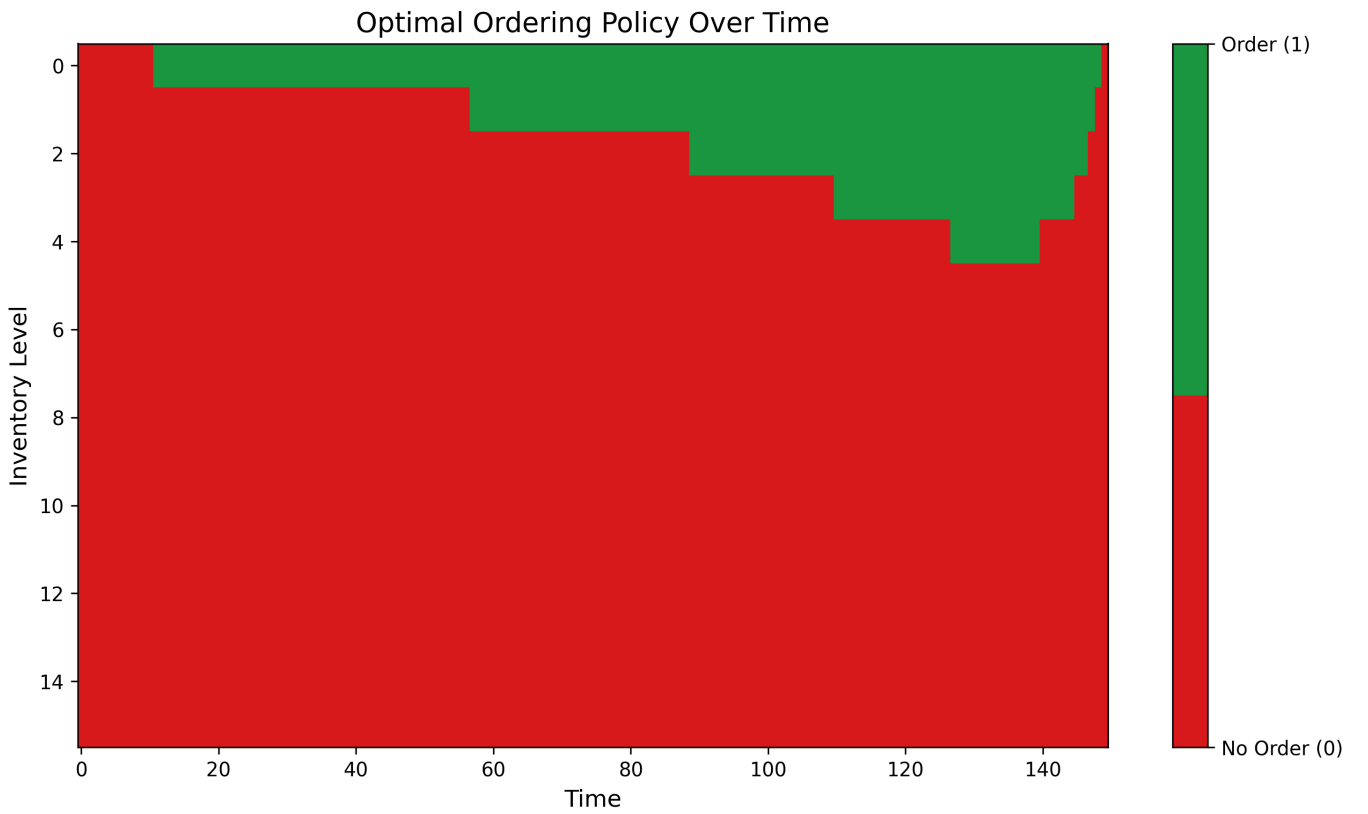


Figure 2: Optimal ordering policy heatmap (green = order, red = no order). Time increases left-to-right; inventory increases top-to-bottom.