École des Ponts
ParisTech

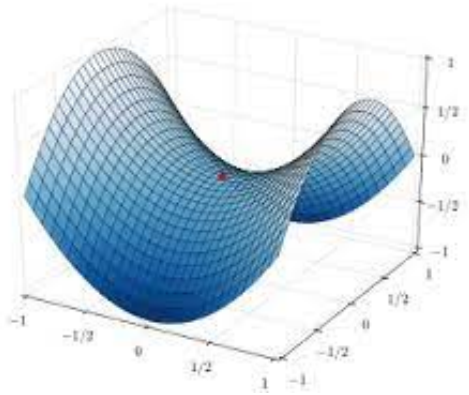
ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES

Dynamic Programming Project

Projet d'Optimisation convexe 2A

par

Mathis WAUQUIEZ Célestin HANS



Ce projet a été effectué sous la supervision de

Vincent LECLERE

1 Maintenance problem

The problem involves managing a fleet of machines over a 12-month cycle, with each machine in one of four states : new, in good shape, old, or broken. The company has several actions available each month to maintain the functionality and profitability of these machines, each with associated costs and effects.

We have to maximize the expected revenue, which makes sense because the owners of the machines wants to make as much money as possible. We then solve the problem using dynamic Programming. We created a 12x4 table whose last row is initialized with known values representing the expected earnings for each state at the final time step. The algorithm fills the table backwards, starting from the second-to-last time step and moving to the first time step. Each iteration consists in computed the expected outcome for each action, by taking into account 3 things :

- The negative cost of the action
- The earnings from the machine in its current state
- The expected future earnings from the next time step, weighted by the probabilities

After the iterations, we obtained the first row of the table equal to [110.82, 85.82, 60.82, 35.82]. For the new state we get the highest value, this suggests that starting with a new machine and managing it optimally provides the greatest potential earnings over the 12 time steps. On the contrary, we get the lowest value with the broken state, which emphasizes the cost and impact of machine breakdowns.

We then performed a simulation to compare the performance of two different strategies (the optimal strategy derived from a dynamic programming approach and the naive strategy) for managing a machine, by measuring their respective earnings and final states over 100 trials.

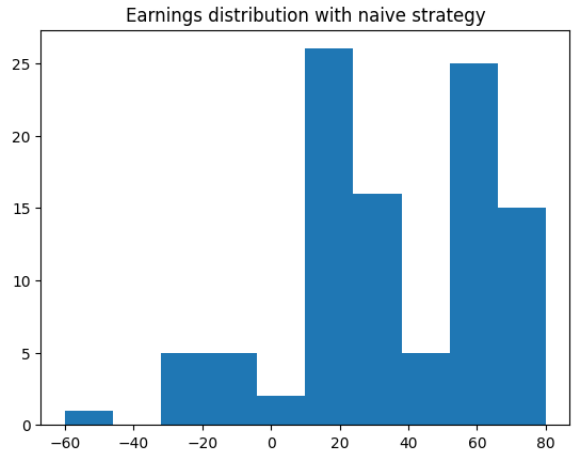


FIGURE 1 – Number of trials as a function of the earnings for the naive strategy

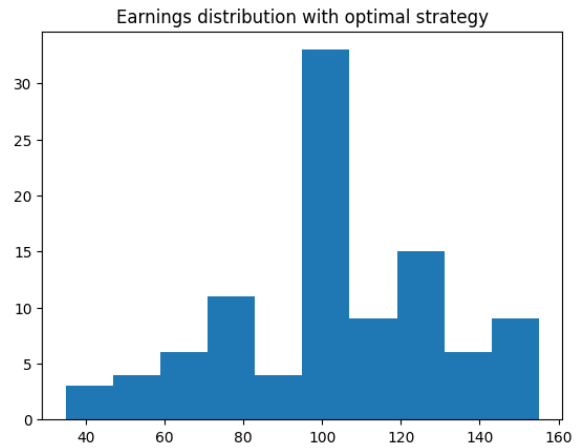


FIGURE 2 – Number of trials as a function of the earnings for the optimal strategy

With the optimal strategy, we obtain an average money earning of 102.4, which is considerably higher than the average earning for the naive strategy of 35.2.

2 Stock management

The problem is about managing a shop's inventory of a single product over 14 days. Starting with 10 units, the owner can order up to 5 more each day without exceeding 20 units in total, at a cost of 1 dollar per unit. Each unit in stock costs 0.10 dollar per day to maintain. Daily sales are driven by a random demand,which is modelled by binomial distributions. Unsold stock at the end of 14 days and unmet demand are both lost. The challenge is to optimize ordering and inventory management to maximize profits within these constraints.

The states are represented by the number of products, the control is the ordering of products, the cost is equal to 0.1 times the stock x. Let's define V(t,s) as the maximum expected profit from day t to the end of the period, given that the stock level at the beginning of day t is s. The decision variable x represents the number of units ordered at the end of day t-1 (for use on day t). The Bellman equation writes :

$$V(t,s) = \max_{x \in \{0, \min(5, 20-s)\}} \Bigg\{ -x + E_{d_t} \Big[\min(d_t, s+x) \cdot 3 - 0.1 \cdot (s + x - \min(d_t, s+x)) + V(t+1, s+x-\min(d_t, s+x)) \Big] \Bigg\}$$

A policy is a set of rules or a strategy that dictates how many units to order at the end of each day, given the current state of the inventory The policy aims to optimize the overall objective, which is to maximize profits over the 14-day period while minimizing costs from ordering and holding inventory.

We used a dynamic programming method for T=14 and obtained the optimal results summed up in the following heatmap :

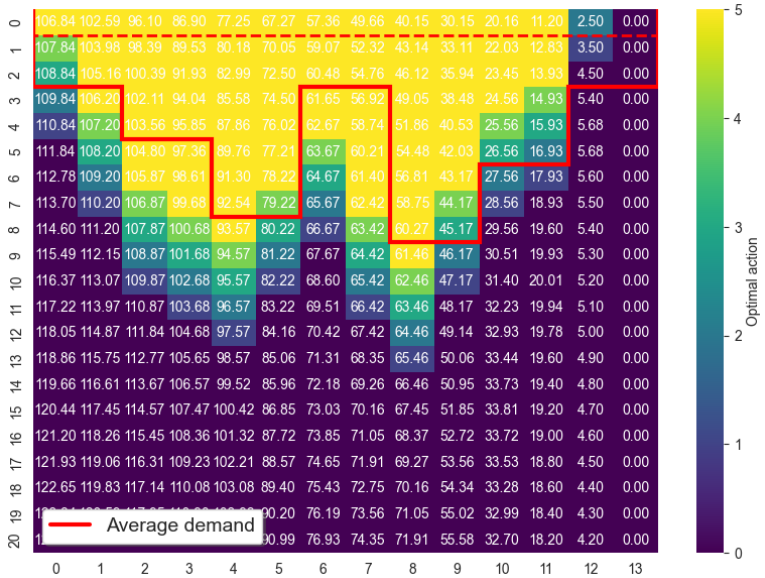


FIGURE 3 – Heatmaps showing the optimal actions and expected profits from the dynamuc programming method for T=14

The simlation with those actions gave an optimal value of 117.49 in a 95 percents confidence

interval [116.83243 ; 118.15357].

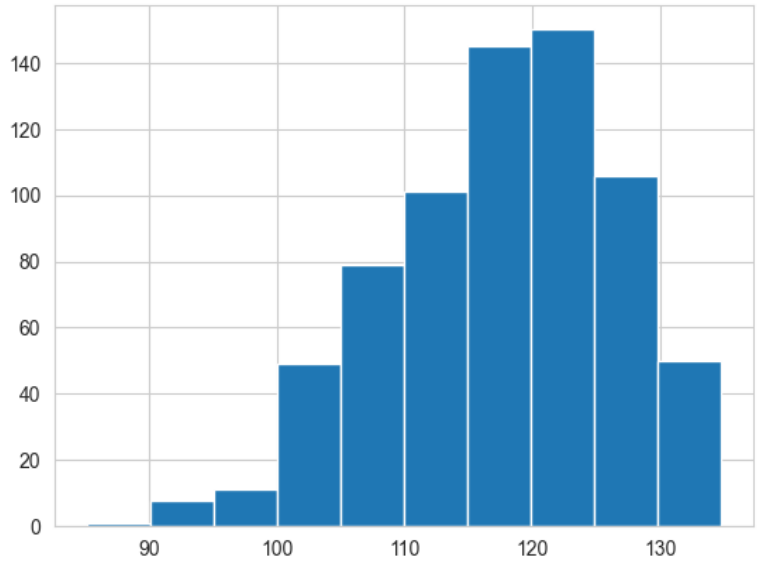


FIGURE 4 – Graph showing the number of simulations as a function of the estimated expected number of points for T=14

We made a simulation with a dynamic programming method over T = 96 days (with periodic demand). We could observe the optimal results in the following heatmap :

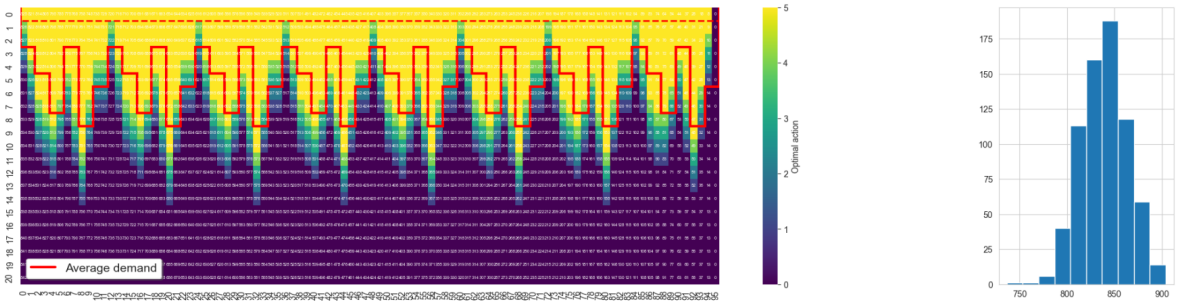


FIGURE 5 – Heatmaps showing the optimal actions and expected profits from the dynamuc programming method and a graph showing the number of simulations as a function of the estimated expected number of points, over T=96

We get an estimated expected number of points of 839.44 with the 95 percents confidence interval [837.63586 ; 841.23671].

3 Dice trading

In this problem, a player participates in a game of dice over 10 turns, aiming to maximize their total points at the end. At the beginning of each turn, the player can choose to buy a new die for 5 points if they have at least 6 points. They then roll all their dice (including any newly bought dice) with 6 faces and add the maximum value rolled to their current points. The player's objective is to strategically decide when to buy new dice and maximize their expected total points by the end of the game, considering the cost of buying dice and the randomness of the dice rolls.

The state is a tuple (p,d) where p is the number of points the player has and d the number of dice. The control is the decision of buying or not a die.

We made simulations whith a naive strategy (buy a die when you have less than two dice and when you have 3 dice, don't buy any die). For 10000 simulations, we get the following histogram :

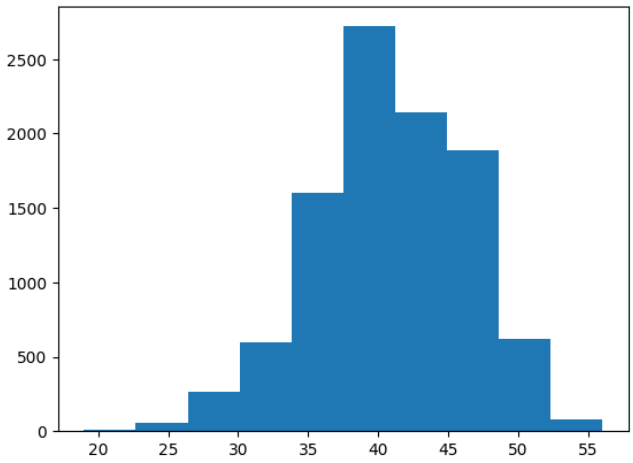
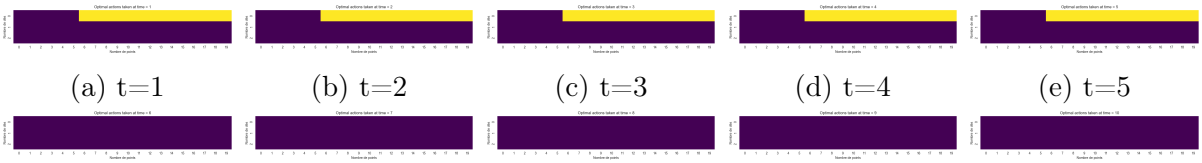


FIGURE 6 – Number of simulations as a function of the score obtained for the chosen strategy

We get an estimated expected number of points equal to 40.90 in the 95 percents confidence interval (40.79, 41.00).

We implemented then a dynamic programming method to compute an optimal value and the associated strategy, using the laws of maximum for 1, 2 and 3 dice. We obtained an optimal value of 40,97, which is higher than the naive strategy taking into account the 95 percents interval. We can then plot the heatmaps for different times.



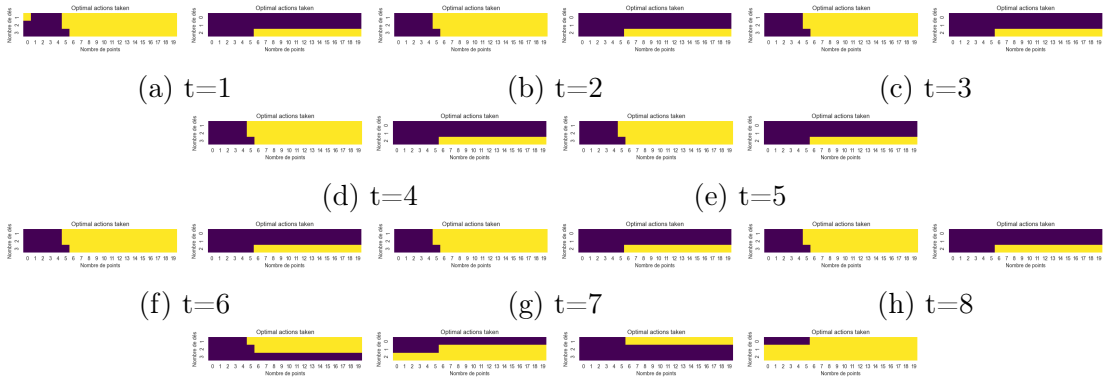
(a) t=1 (b) t=2 (c) t=3 (d) t=4 (e) t=5

(f) t=6 (g) t=7 (h) t=8 (i) t=9 (j) t=10

FIGURE 7 – Heatmaps representing the optimal actions for several times

According to those heatmaps, we conclude that it is not interesting to buy a die for T greater than 5.

Then we add a new rule : at any turn, once the dice are thrown, the player, if he has at least 2 dice, can spend a die to double the gain of the throw. Implementing a new action to the previous method leads to the following results :



(a) t=1 (b) t=2 (c) t=3

(d) t=4 (e) t=5

(f) t=6 (g) t=7 (h) t=8

(i) t=9 (j) t=10

FIGURE 8 – Heatmaps representing the optimal actions (action of buying a die in the left, and sacrificing a die in the right) for several times

We can see that the player tends to sacrifice a die when he has a many dice which is intuitive. We obtain an optimal value of 50.28, which is relevant as it is higher than the optimal value obtained with the previous rules.