Reinforcement Learning for Hydroelectric Dam Control

January 31, 2026

Abstract

This report presents a Reinforcement Learning (RL) approach to optimize the operation of a hydroelectric dam for electricity production. We describe the problem, two baseline heuristics, the RL solution, the validation evaluation, and the performance evaluation. Additional research directions and ablation studies are also discussed.

1 Introduction

Pumped-storage hydroelectric dams function as large-scale energy storage systems, buying electricity to pump water uphill during low-price periods and generating during price peaks [1]. Electricity spot prices exhibit predictable daily and seasonal patterns but remain stochastic due to supply-demand dynamics and weather impacts [2], making traditional optimization impractical without accurate forecasts.

This project develops a Reinforcement Learning control strategy to maximize revenue under price uncertainty. We formulate dam operation as a Markov Decision Process where the controller decides hourly whether to pump, generate, or hold based on current conditions—reservoir level, price, and time—without future price knowledge. Tabular Q-learning learns an optimal policy from three years of historical price data, balancing immediate profit with long-term value through reward shaping and potential-based incentives.

We validate the learned policy on two years of unseen data, comparing performance against random and rule-based baselines while analyzing learned arbitrage strategies through Q-table visualization.

2 Problem Formulation

A Markov Decision Process (MDP) provides a mathematical framework for modeling sequential decision-making under uncertainty. In this section, we formally define the state space, action space, reward function, and constraints that characterize our dam control problem.

We model dam operation as a Markov Decision Process (S, A, P, R, γ):

2.1 State Space

The agent does not observe the full raw environment observation vector. Instead, a compact engineered state is used, consisting of four features:

$$s_t = [V_t,\ \mathbb{I}(p_t > \bar{p}_{t,24}),\ h_t,\ \mathbb{I}(m_t \in \mathcal{M}_{\text{cold}})], \tag{1}$$

where V_t is the amount of water in the reservoir, $h_t \in \{1, \ldots, 24\}$ is the hour of day, and m_t is the month. The binary indicator $\mathbb{I}(p_t > \bar{p}_{t,24})$ is 1 if the current price p_t is above the rolling 24-hour average $\bar{p}_{t,24}$, and 0 otherwise. The last binary indicator is 1 if it is a cold month and 0 otherwise. The cold months are defined as

$$\mathcal{M}_{\text{cold}} = \{10, 11, 12, 1, 2\}.$$

For Tabular Q-learning, the reservoir volume is discretized in five bins, and the hours of the day are 24 discrete values.

2.2 Action Space

The environment supports a continuous action space $A = [-1, 1]$, where:

- $a < 0$: generate electricity (release water),
- $a > 0$: pump water (store energy),
- $a = 0$: no operation.

For Tabular Q-learning, this space is discretized to $\{-1, 0, 1\}$.

2.3 Reward Function

There are two different rewards used. The first is the raw environment reward. This reward is actual money spent on pumping or made by generating. This reward is only used for evaluating.

For training, a shaped reward is used. The stored energy, E_t, and its cumulative cost, C_t are tracked. Pumping actions update (E_t, C_t) but give no immediate reward. When energy is generated and sold, the realized profit is

$$r_t^{\text{profit}} = 0.9\, p_t\, (-\Delta E_t) - (-\Delta E_t) \frac{C_t}{E_t}, \qquad \Delta E_t < 0, \tag{2}$$

Here, p_t is the electricity price. The factor 0.9 accounts for energy losses, as 10% of the water's potential energy is lost during conversion to electricity. This factor is therefore incorporated into the reward function.

To help the agent better associate actions with delayed economic outcomes and stabilize learning, we apply potential-based reward shaping [4]. The potential is as follows.

$$\Phi_t = 0.9\, p_t\, E_t - C_t, \tag{3}$$

For the final training reward, we took the difference in potential and added it to the profit reward:

$$r_t^{\text{train}} = r_t^{\text{profit}} + (\gamma \Phi_{t+1} - \Phi_t). \tag{4}$$

This reward function encourages actions that increase the future value of the inventory. The reward is divided by 1000 for numerical stability.

2.4 Optimization Objective

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t) \mid \pi\right] \tag{5}$$

The highest commutative reward with discount factor $\gamma = 0.99$ for long-term planning.

3 Data analysis

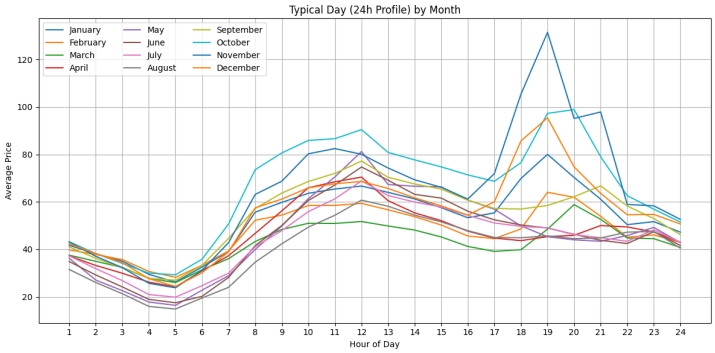


Figure 1: Typical Day (24h Profile) by Month.

The training data consists of hourly electricity prices (€/MWh) from 2007-2009. Additional descriptive Figures are provided in the Appendix. Figure 1 shows the average daily price trajectory for each month. The prices are consistently lowest overnight and peak consistently for every month at midday. What is interesting is the seasonality effect on these peaks. In the winter months, a second price peak is formed in the afternoon. This can explained by the fact that people are more active and need heating inside their homes during cold evenings. Although the intraday pattern is stable, absolute price levels vary over a longer time. The data shows inflation as well as extended periods of generally higher or lower prices (See Appendix A). As a result, fixed price thresholds are unsuitable. Dynamic or relative price features should be integrated to help with generalization.

Weekend prices follow the same intraday structure as weekdays but with a lower amplitude as seen in Figure 9. Price volatility is highest during the peak hours, and the dataset does contain extremely high and low outliers, shown in the appendix plots.

Overall, the data show stable intraday patterns with seasonal modulation of peak times. The absolute price level varies but shows good relative trends. The properties motivate a sequential decision-making approach using dynamic, relative price signals.

4 Heuristic Baseline

To check if our problem benefits from Reinforcement Learning, a heuristic is constructed as a baseline. The final results can be compared to this baseline to see if reinforcement learning helped our problem.

4.1 Random Heuristic

With the provided code and environment, a baseline random heuristic can be run. In this heuristic, the agent takes a random action every step. This heuristic was run 100 times and the mean total cumulative reward is: -96610.98.

4.2 Rule-Based Heuristic Policy

This heuristic is a simple rule-based approach that defines when to pump, generate, or remain idle. It relies on the hour of the day, whether the current month is classified as cold or not, and whether the current price is above or below the 24-hour rolling average.

It selects a continuous action $a_t \in [-1, 1]$, where $a_t > 0$ corresponds to pumping, $a_t < 0$ to generating, and $a_t = 0$ to holding.

The cold months, modeled after Figure 1 $\mathcal{M}_{\text{winter}} = \{10, 11, 12, 1, 2\}$.
The policy is defined as:

$$a_t = \begin{cases} 1, & \text{if } h_t \in [2,7] \ \wedge \ p_t < \bar{p}_t, \\ -1, & \text{if } m_t \in \mathcal{M}_{\text{winter}} \ \wedge \ h_t \in [10,12] \ \wedge \ p_t > \bar{p}_t, \\ -1, & \text{if } m_t \in \mathcal{M}_{\text{winter}} \ \wedge \ h_t \in [18,21] \ \wedge \ p_t > \bar{p}_t, \\ -1, & \text{if } m_t \notin \mathcal{M}_{\text{winter}} \ \wedge \ h_t \in [9,14] \ \wedge \ p_t > \bar{p}_t, \\ 0, & \text{otherwise.} \end{cases}$$

For the full utilization, we pump six hours a day and generate six hours a day. Pumping always happens in the morning between 02:00-07:00. Energy is only sold if the price is higher than the average of the last 24 hours and only used when it is lower than the last 24 hours. In the summer everything is sold between 09:00-14:00. To utilize the extra price peak in the winter, discussed in chapter 3 the energy is generated and sold between 10:00-12:00 and 18:00-21:00. A plot of the pumping and generating times is in Appendix B.

This results in a total reward on the validation data of 72,886. The algorithm behaves as expected shows in Appendix C.

5 Reinforcement Learning Solution

This section describes our Tabular Q-learning approach to learning an optimal dam control policy. We begin with an overview of the algorithm choice, followed by details on state discretization, the learning procedure, training setup, and analysis of the learned policy behavior.

5.1 Approach Overview

We selected Tabular Q-learning as our Reinforcement Learning algorithm for several reasons. First, Q-learning is a model-free method that learns directly from experience without requiring knowledge of the environment's transition dynamics or reward function—critical given the stochastic nature of electricity prices [3]. Second, as an off-policy algorithm, Q-learning can learn the optimal policy while following an exploratory behavior policy, enabling efficient learning from our finite historical dataset. Third, the Tabular formulation provides interpretability: the learned Q-table can be directly inspected to understand which state-action pairs the agent values most highly.

The algorithm maintains a table $Q(s, a)$ that estimates the expected cumulative reward for taking action a in state s. Through iterative updates based on observed transitions, the Q-values converge to the optimal action-value function, from which the optimal policy can be extracted via greedy action selection.

5.2 State Representation and Discretization

As described in Section 2.1, our state representation consists of four engineered features: reservoir volume V_t, a binary price indicator relative to the 24-hour rolling average, hour of day h_t, and a binary cold-month indicator.

Tabular Q-learning requires finite and discrete state and action spaces. The continuous state variables are discretized as follows:

- **Volume**: 5 bins spanning [0, 100,000] m^3
- **Price indicator**: 2 bins (binary: above/below 24h average)
- **Hour**: 24 bins (one per hour of day)
- **Cold month**: 2 bins (binary: winter/non-winter)

This yields a total state space of $5 \times 2 \times 24 \times 2 = 480$ discrete states. The discretization balances expressiveness with tractability.

The action space is discretized to three values: $\{-1, 0, 1\}$, corresponding to maximum generation, no action, and maximum pumping respectively.

5.3 Q-learning Algorithm

Q-learning updates state-action values iteratively using:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right] \tag{6}$$

with learning rate $\alpha = 0.03$ and discount factor $\gamma = 0.99$.

5.4 Training Setup and Hyperparameters

The model is trained on data from 2007 to 2009, consisting of 1,096 daily observations. A validation set covering the period 2010–2011 (730 days) is used to evaluate the algorithm and explore how well it generalizes. This set is only used after training.

The model is trained for a fixed number of episodes. Model selection is performed by choosing the episode with the highest validation performance, which prevents overfitting. Also, a relatively high number of training episodes (50) is used. This proved to find better solutions early on in the process than a lower number of episodes. This is expected to be due to the epsilon decay function.

The learning rate is set to $\alpha = 0.03$ throughout training. After 80% of the episodes, exploration is minimal, and then dividing the learning rate by 10 helps stabilize Q-value updates.

Once exploration is reduced, a smaller learning rate helps stabilize Q-value updates. The discount factor is set to $\gamma = 0.99$, giving high importance to future rewards while still valuing immediate profit.

5.5 Training Dynamics

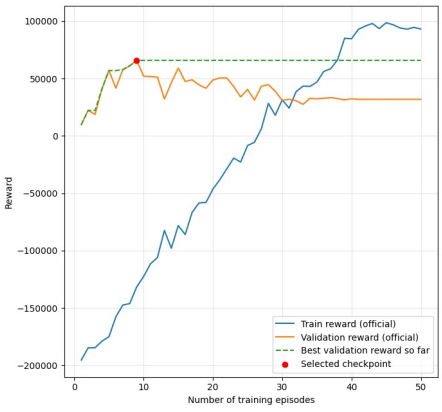


Figure 2: Cumulative reward per episode

Figure 2 shows the cumulative reward per episode for both the training and validation datasets. The validation curve reaches its maximum around episode 8, after which performance degrades. This indicates that additional training episodes do not improve generalization.

Note that training and validation rewards are not directly comparable in magnitude, as they are accumulated over datasets of different lengths. The validation curve is therefore used solely

for model selection, not for absolute performance comparison. Also, the train reward plotted is rolled out using a epsilon exploration policy while the validation reward is plotted using a greedy policy.

5.6 Learned Policy Interpretation

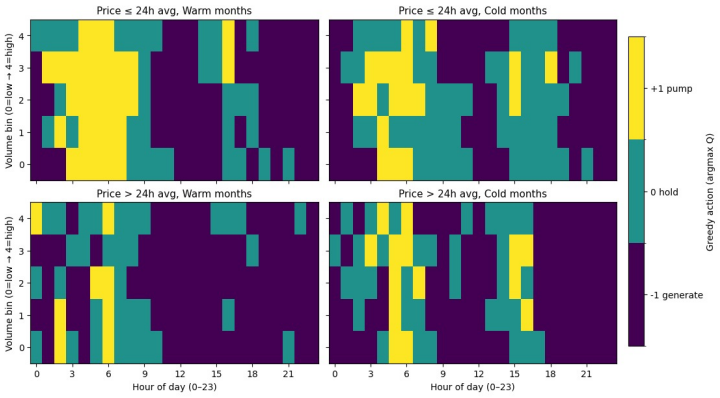


Figure 3: Q-table preferred action for all states

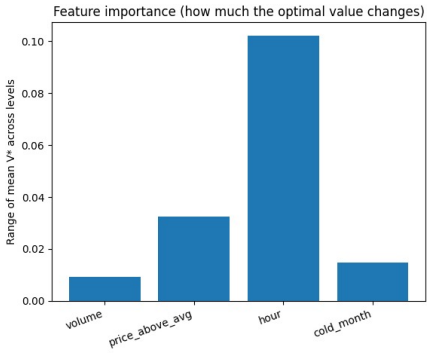


Figure 4: Feature importance

The learned policy is analyzed by visualizing the greedy action selection ($\arg\max_a Q(s,a)$) across different state dimensions. Heatmaps are constructed by conditioning on price regime (above or below the 24-hour average), hour of day, cold month, and reservoir volume in Figure 3.

These visualizations illustrate that the policy prefers to pump longer and more when the price is below the average. The policy generates more when it's above the price average. Also, when the month is cold, it holds its storage for longer so that it can sell at the peak in the afternoon. The volume does not seem to show a very clear pattern of preferred action.

Figure 4 shows the average change in value for a state while varying the feature value. This also shows that the volume was the least important feature for the algorithm. The hour of the day is the most important.

6 Validation and Performance

The final policy is evaluated on an unseen validation dataset. Evaluation is performed using a greedy policy ($\epsilon = 0$) with the learned Q-table. Performance is compared against the random and rule-based heuristic baselines.

6.1 Cumulative Reward Comparison

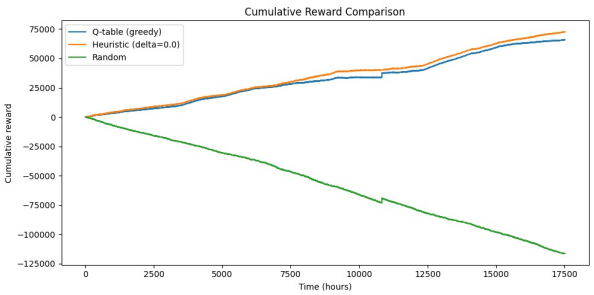


Figure 5: Cumulative Reward Plot

The cumulative reward on the validation set is **€65,902**, substantially outperforming the random baseline but about 10% lower than the rule-based heuristic (See Figure 5). The result confirms that the learned policy captures exploitable price patterns.

6.2 Policy Behavior on Validation Data

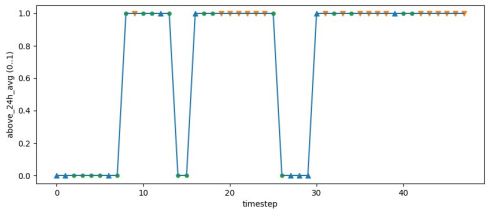


Figure 6: Higher/Lower price than last 24h vs action

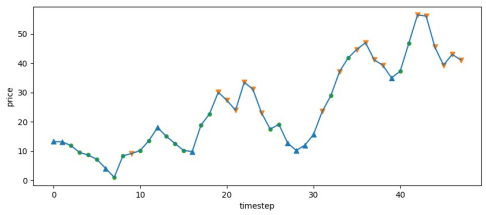


Figure 7: Electricity price vs action

In these plots, the actual behavior of the algorithm on validation data is plotted for a random 48h period. The blue triangle is pumping water up. The green dot is holding and the orange triangle is generating electricity. The algorithm behaves as expected. Figure 6 shows that the algorithm usually pumps when the price is lower than the last 24 hours and generates when it is higher. In the price-plot in Figure 7 can be seen that it pumps at lower prices and sells at higher ones.

6.3 Robustness and Generalization

The policy behaves consistently across the full validation set. Performance degrades smoothly during atypical price periods, indicating robustness rather than overfitting to isolated price spikes.

6.4 Validation Summary

In summary, the learned Q-learning policy generalizes to unseen data but achieves slightly lower cumulative reward compared to the rule-based heuristic. The results demonstrate that Q-learning is a viable approach for hydroelectric dam control under price uncertainty. However, it is not necessarily the best option.

7 Additional Experiments and Ablation

The following experiments were exploratory in nature and aimed at informing the design of the final agent rather than providing a comprehensive statistical comparison. To assess the effect of the state and action representation on the learning performance, several variations were tested. These included adding weekday/weekend indicators, replacing the binary cold-month indicator with four seasonal categories, and modifying discretization by changing the bin sizes for volume and time (hour). Finally, beyond simply experimenting with bin size for the price, percentile-based, relative, and normalized bins were implemented to account for the effects of inflation.

By experimenting there was also found that removing temporal structure from the state, such as hour of day or relative price information, consistently degraded performance, indicating that intraday price dynamics are critical for effective control. Adding additional features, such as weekday/weekend indicators, yielded no consistent improvement and in some cases increased variance during training.

Finally, several action space variants were also tested. Expanding the action space beyond the basic [-1, 0, 1] to include intermediate values (e.g., [-1, -0.5, 0, 0.5, 1] or [-1, -0.67, -0.33, 0, 0.33, 0.67, 1]) did not improve performance gains but resulted in an increase in Q-table size. Therefore, the simpler action set was retained.

Based on these experiments, the optimal feature set and binning strategy were determined as follows: volume (5 bins), hour (24 bins), price_above_average (2 bins), and is_cold_month (2 bins). This configuration maximized the agent's learning effectiveness while keeping the Q-table as small as possible.

7.1 Learning Rate and Discount Factor

The hyperparameter experiments focused on the learning rate and discount factor. Higher learning rates (≥ 0.3) accelerated convergence but caused occasional instability. Therefore, reducing the learning rate from 0.03 to 0.003 after 80% of training was crucial to stabilize the final policy.

Variations in the discount factor influenced the agent's balance between short-term and long-term learning. Values in the range of 0.95 to 0.99 promoted longer-term strategic behavior, while a discount factor of 1 hindered convergence due to the excessive emphasis on distant rewards.

Training initially began with a discount factor of 0.95 to encourage exploration. In the final training run, the discount factor was increased to 0.99, which provided a better balance between stability and long-term reward optimization.

7.2 Exploration Strategy Variants

Several exploration strategies were evaluated to optimize the balance between exploration and exploitation. Using a constant epsilon of (0.05) led to slow exploration.

A decaying ϵ-greedy strategy proved most effective. Exploration was gradually reduced over approximately 80% of training episodes using an initial epsilon of $\epsilon = 1.0$, allowing sufficient early exploration while enabling stable exploitation later. Ending exploration prematurely (i.e. 50% of episodes) reduced the agent's ability to adequately explore the state space, leading to lower performance.

8 Conclusion, Discussion and Future Work

This work demonstrates that Tabular Q-Learning can successfully control a hydroelectric dam under stochastic electricity prices. Using a compact, interpretable state representation and historical price data, the learned policy achieves strong validation performance and exhibits economically intuitive behavior, such as pumping during low-price periods and generating during price peaks.

Although Tabular Q-learning demonstrated the ability to learn patterns in this problem and generate profit, it required significantly more implementation effort and complexity than the rule-based heuristic. Despite outperforming the heuristic in some cases, it was also less predictable. Although reinforcement learning offers the advantage of adaptability without manual rule engineering, this flexibility did not lead to an improvement over our best-performing solution.

Future work could try to optimize the reward function or use other engineered features which have not been tried in this report. Also, this framework can be extended in several directions. First, incorporating stochastic inflows and uncertainty in water availability would increase realism. Second, multi-reservoir or multi-plant coordination could be studied to assess scalability. Third, function approximation methods such as linear value function approximation or deep reinforcement learning could be explored to handle richer state representations, provided interpretability and stability are preserved.

Appendix

Appendix A: Additional Data Analysis Figures

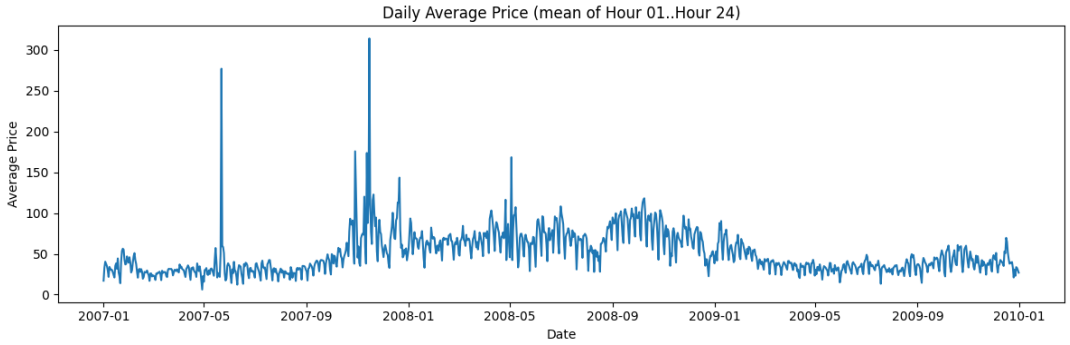


Figure 8: Daily Average Price (mean of Hour 1-24)

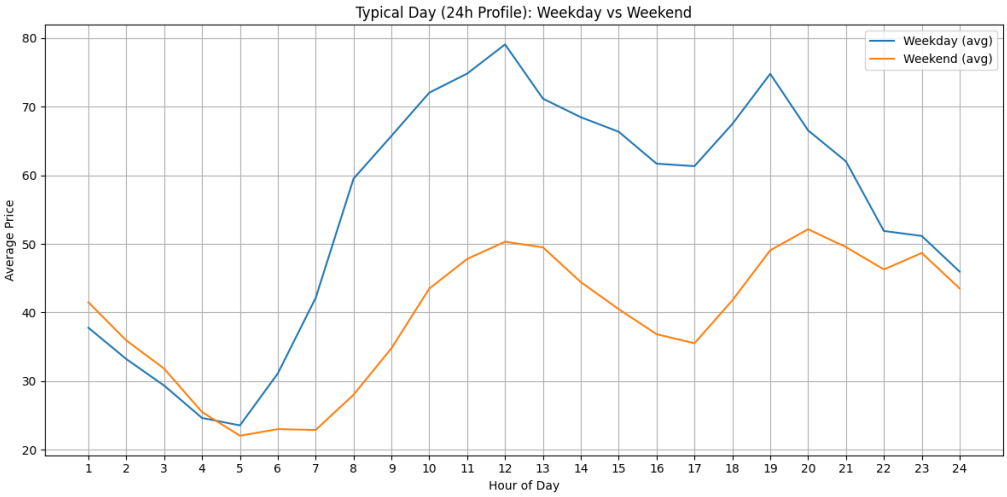


Figure 9: Typical Day (24h Profile): Weekday vs Weekend.

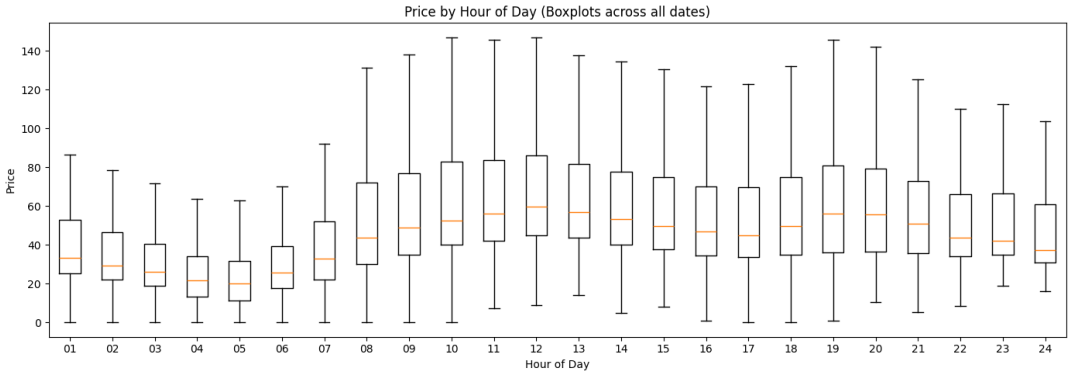


Figure 10: Price by Hour of Day (boxplots across all dates).

Appendix B: Heuristic policy

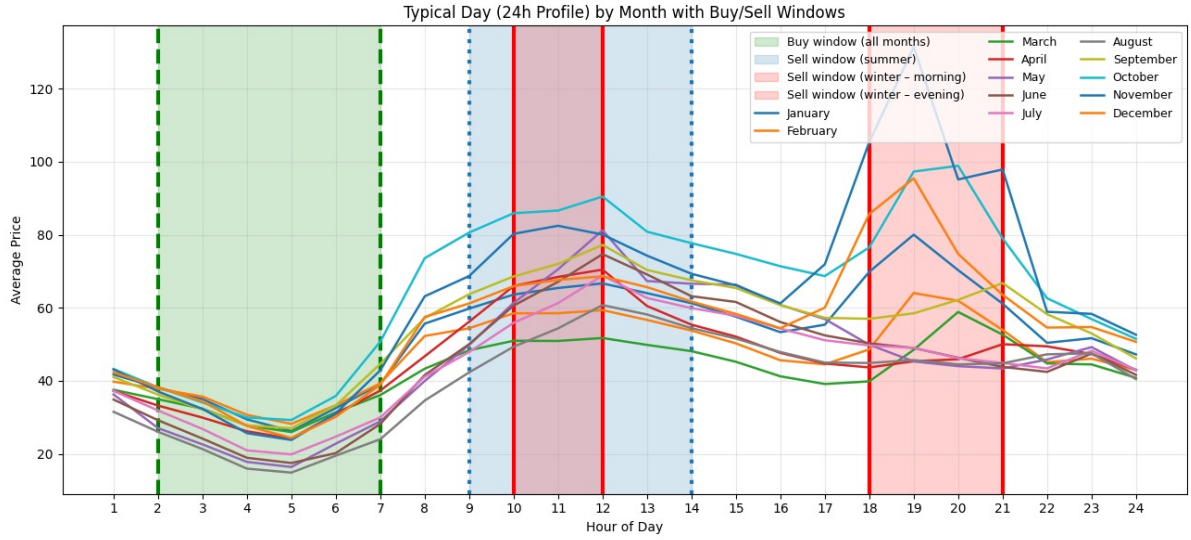
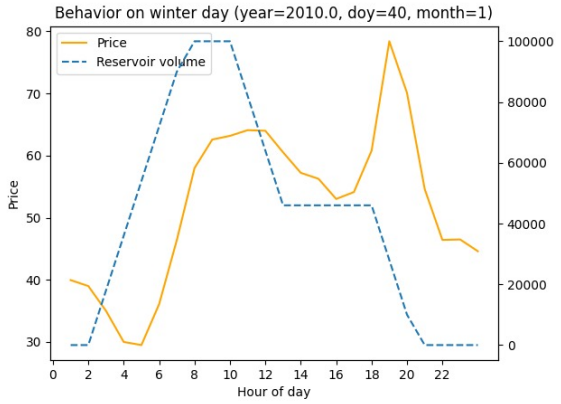
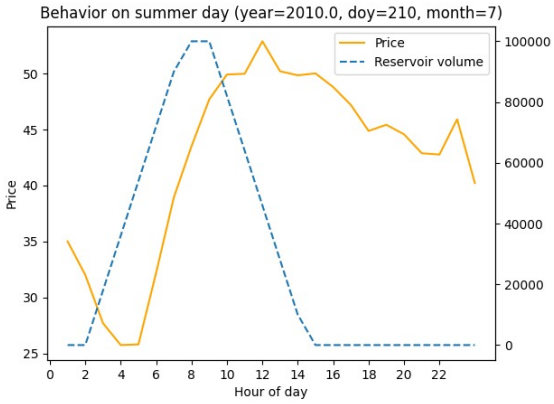


Figure 11: Typical daily price by month, showing average hourly prices and heuristic buy and sell windows. The shaded regions indicate pumping during low-price night hours and generation during peak-price periods, with separate windows for cold and not cold months.

Appendix C: Heuristic behavior



(a) Winter day (year 2010, day-of-year 40, January)



(b) Summer day (year 2010, day-of-year 210, July)

Figure 12: Heuristic behavior under cold and warm seasonal conditions. Electricity price is shown on the left axis (solid line) and reservoir volume on the right axis (dashed line). Here can be seen that the full volume is used and that in cold months the inventory is sold in two periods while in the non-cold month it is sold in one.

References

[1] International Energy Agency, *Hydropower Special Market Report*, IEA, Paris, 2021.

[2] R. Weron, "Electricity price forecasting: A review," *International Journal of Forecasting*, vol. 30, no. 4, pp. 1030–1081, 2014.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, Cambridge, MA, 2018.

[4] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations," in *Proceedings of the 16th International Conference on Machine Learning (ICML)*, pp. 278–287, 1999.