Contents

1. Background

Water is one of the most valuable commodities all around the world, and reliable streamflow forecasts are critical for sustainable water resources management. These forecasts are widely used to inform downstream decisions, including water allocation across sectors in ways that balance both economic and environmental objectives.

In the Phase 1 of this project, we developed machine learning-based models to produce accurate probabilistic streamflow forecasts. While forecast accuracy is crucial, an equally important challenge lies in how these forecasts are effectively utilized for water resource management across different actors. Phase 2 of the project addresses this challenge directly. It involves evaluating water regulation policies within an agent-based modeling framework that simulates water withdrawals at two French river monitoring stations. Distinct actors (e.g., agriculture, industry, energy) use the streamflow forecasts from Phase 1 to inform their water allocation decisions, aiming to satisfy ecological requirements while maximizing economic returns.

With this objective in mind, and aligned with the goals of the Hackathon, we designed and implemented water management incentive and quota policies. These policies were tested across a range of scenarios that vary in forecast error profiles, hydrological conditions, actor compositions, and water crisis levels. Our proposed strategies aim to provide safeguard against ecological systems by maintaining environmental flows above critical thresholds while also ensuring maximum economic benefits among different actors.

2. Problem Statement

We were provided two functions to modify and optimize to improve the ecological and economical factors.

Design of quota policy:

The quota policy determines how much water each actor is permitted to withdraw, balancing their assigned priority with current hydrological stress conditions (relative to DOE and DCR).

Design of incentive policy:

Incentive policy is designed in such a way that penalties are given for high usage during stress and subsidies are provided for underuse.

We were provided data on five canonical water-user archetypes, each defined by distinct demand patterns, economic values, and priority levels. These archetypes help classify different sectors based on how they utilize water resources and the value they derive from them.

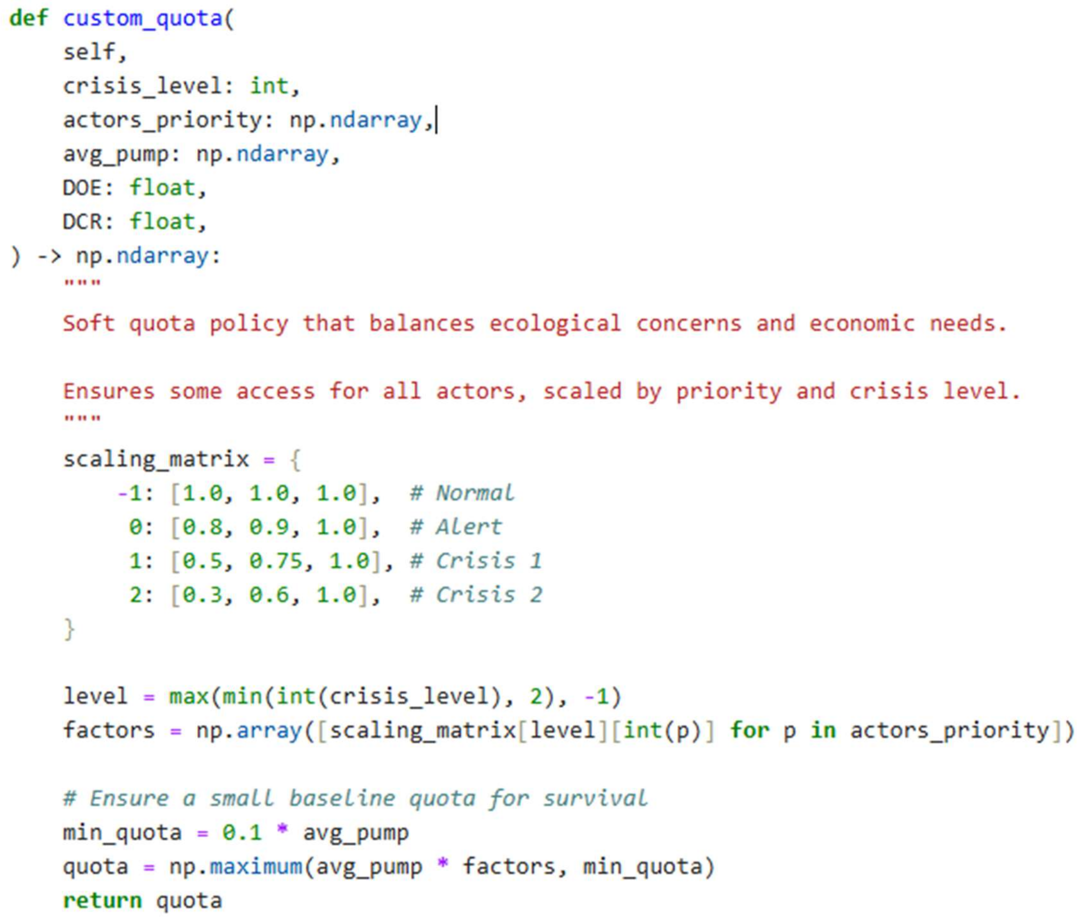
The **agriculture** sector is categorized as a medium-priority user (P_1). It has a low economic value per unit of water but is characterized by high overall consumption. **Industry** falls under the low-priority category (P_0), with moderate economic value per unit and low water consumption. **Energy** is designated as a high-priority user (P_2) due to its critical role in society. It has a medium level of water consumption. **Drinking water** is given the highest priority (P_2), also considered critical to societal functioning, and has medium consumption. Finally, **leisure** uses—such as recreational and aesthetic water use—are categorized as low priority (P_0), with high economic value per unit but very low consumption.

Water User	**Priority Level**	**Economic Value per Unit**	**Consumption Level**
Agriculture	Medium (P_1)	Low	High
Industry	Low (P_0)	Moderate	Low
Energy	High (P_2)	Critical societal value	Medium
Drinking Water	Highest (P_2)	Critical societal value	Medium
Leisure	Low (P_0)	High	Very Low

Based on the priority level, economical value, and consumption level, we were expected to develop the ***quota*** and ***incentive*** policy that gave optimize values for economical (more is better) and ecological (less is better) for multiple scenarios while considering uncertainties.

3. Our Custom Policies

3.1 Custom Quota Policy



```python
def custom_quota(
    self,
    crisis_level: int,
    actors_priority: np.ndarray,
    avg_pump: np.ndarray,
    DOE: float,
    DCR: float,
) -> np.ndarray:
    """
    Soft quota policy that balances ecological concerns and economic needs.

    Ensures some access for all actors, scaled by priority and crisis level.
    """
    scaling_matrix = {
        -1: [1.0, 1.0, 1.0],  # Normal
         0: [0.8, 0.9, 1.0],  # Alert
         1: [0.5, 0.75, 1.0], # Crisis 1
         2: [0.3, 0.6, 1.0],  # Crisis 2
    }

    level = max(min(int(crisis_level), 2), -1)
    factors = np.array([scaling_matrix[level][int(p)] for p in actors_priority])

    # Ensure a small baseline quota for survival
    min_quota = 0.1 * avg_pump
    quota = np.maximum(avg_pump * factors, min_quota)
    return quota
```

Figure: Snapshot of our quota policy.

Description:

This function implements a **soft quota allocation policy** that ensures equitable and priority-aware access to water resources, adapting to varying levels of water crisis.

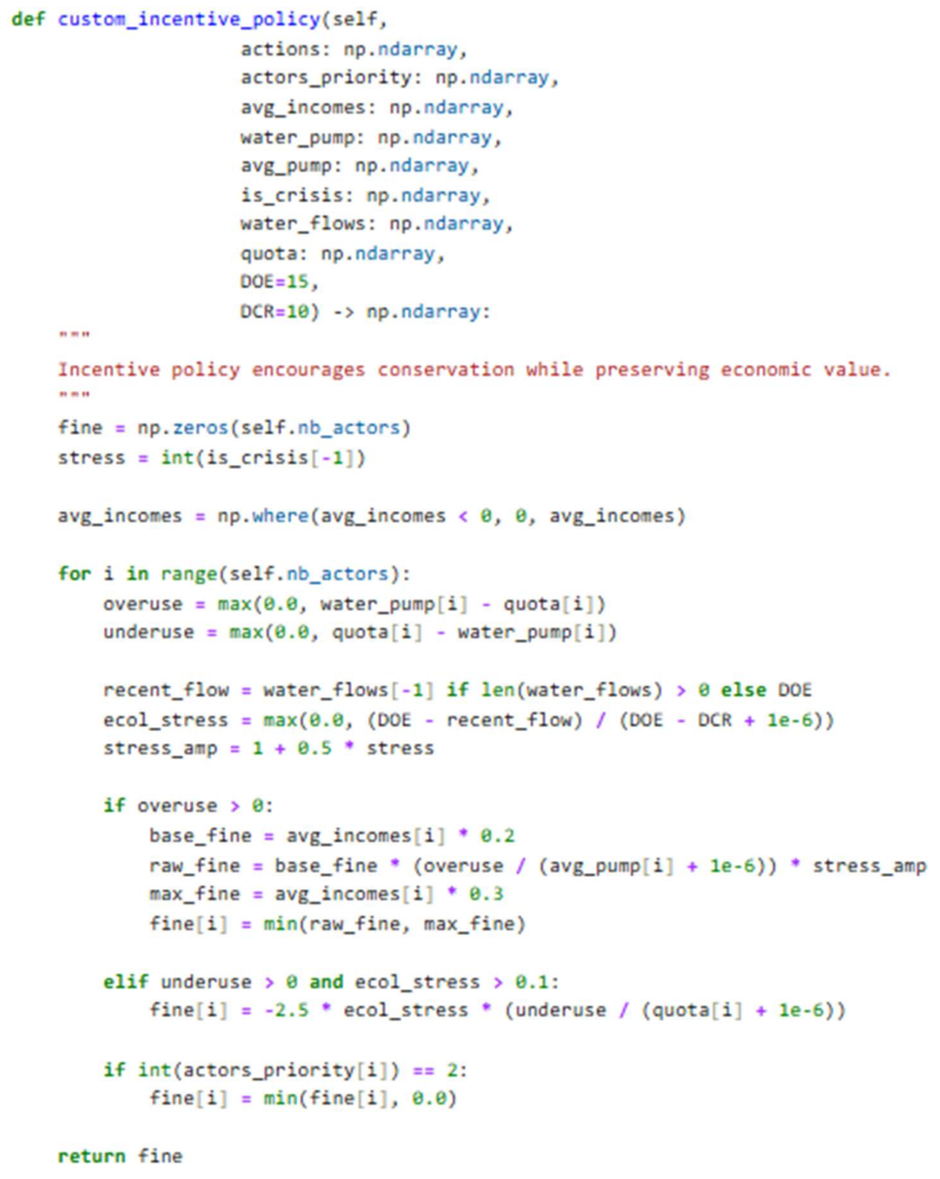
Key Features:

1. **Priority-based scaling**: Quotas are scaled by a predefined matrix based on each actor's priority and the current crisis level.
2. **Guaranteed minimum access**: Each actor receives at least 10% of their average pump volume to avoid total cutoff, even under severe crisis.
3. **High-priority protection**: High-priority actors (priority = 2) retain their full water quota regardless of crisis level.

Quota Scaling Matrix:

Crisis Level	Low Priority	Medium Priority	High Priority
Normal (-1)	100%	100%	100%
Alert (0)	80%	90%	100%
Crisis 1 (1)	50%	75%	100%
Crisis 2 (2)	30%	60%	100%

3.2 Custom Incentive Policy



```python
def custom_incentive_policy(self,
                    actions: np.ndarray,
                    actors_priority: np.ndarray,
                    avg_incomes: np.ndarray,
                    water_pump: np.ndarray,
                    avg_pump: np.ndarray,
                    is_crisis: np.ndarray,
                    water_flows: np.ndarray,
                    quota: np.ndarray,
                    DOE=15,
                    DCR=10) -> np.ndarray:
    """
    Incentive policy encourages conservation while preserving economic value.
    """
    fine = np.zeros(self.nb_actors)
    stress = int(is_crisis[-1])

    avg_incomes = np.where(avg_incomes < 0, 0, avg_incomes)

    for i in range(self.nb_actors):
        overuse = max(0.0, water_pump[i] - quota[i])
        underuse = max(0.0, quota[i] - water_pump[i])

        recent_flow = water_flows[-1] if len(water_flows) > 0 else DOE
        ecol_stress = max(0.0, (DOE - recent_flow) / (DOE - DCR + 1e-6))
        stress_amp = 1 + 0.5 * stress

        if overuse > 0:
            base_fine = avg_incomes[i] * 0.2
            raw_fine = base_fine * (overuse / (avg_pump[i] + 1e-6)) * stress_amp
            max_fine = avg_incomes[i] * 0.3
            fine[i] = min(raw_fine, max_fine)

        elif underuse > 0 and ecol_stress > 0.1:
            fine[i] = -2.5 * ecol_stress * (underuse / (quota[i] + 1e-6))

        if int(actors_priority[i]) == 2:
            fine[i] = min(fine[i], 0.0)

    return fine
```

Figure: Snapshot of custom incentive policy

Description:

This function applies **dynamic economic incentives** (fines and subsidies) based on water use behavior, income, and ecological stress. It encourages actors to stay within quotas and rewards conservation during times of environmental strain.

Key Features:

1. **Fines for overuse**:

Proportional to how much the actor exceeds their quota.

Scaled by the actor's income and the current crisis level.

Capped at 30% of the actor’s average income to avoid excessive penalties.

2. **Subsidies for underuse**:

Applied only if ecological stress exists (i.e., river flow is below the ecological threshold).

Scaled by the amount of water saved and the level of ecological stress.

3. **High-priority protection**:

High-priority actors are **never fined** and may still receive subsidies if they conserve water.

Additional notes to avoid penalty:

- *The system ensures that **medium-priority actors always perform better than low-priority** actors by:*
- *Giving them more water in the quota function.*
- *Reducing their likelihood of fines due to larger quota buffers.*
- *Optionally, you can scale subsidies by priority to further boost their performance.*

4. Why is our policy better than the original?

We initially ran the model using both the original policies and our custom policies. The original policy served as a baseline for comparison, allowing us to evaluate and improve the performance of our custom approach.

Additionally, in order to qualify scenarios for comparison, we ensured that the ***satisfaction criteria were met for qualification***, as outlined in the provided description.

Summary of Overall Performance

Original Policy:

Percentage of scenarios meeting satisfaction criteria: **61.11%**

Median Ecological factor (Qualified only): **0.822**

Median Economic factor (Qualified only): **0.378**

Custom Policies:

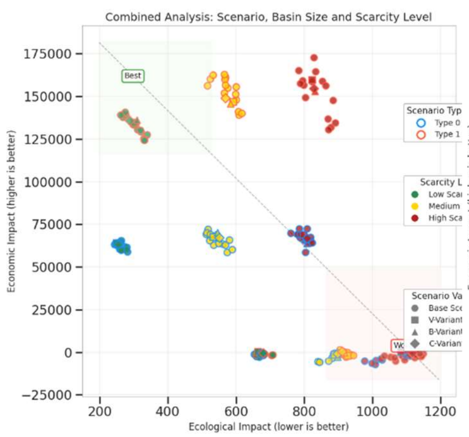
Percentage of scenarios meeting satisfaction criteria: **83.80%** ***(Highly improved)***

Median Ecological factor (Qualified only): **0.956** ***(Slight Declined)***

Median Economic factor (Qualified only): **0.901** ***(Highly improved)***

Based on these results, we observe that our custom policy significantly increased the percentage of scenarios satisfying the criteria. While the ecological factor is slightly lower compared to the original policy, the economic factor shows substantial improvement. This reflects a strategic trade-off in favor of economic performance while still maintaining strong ecological outcomes.

Let's compare our plots:



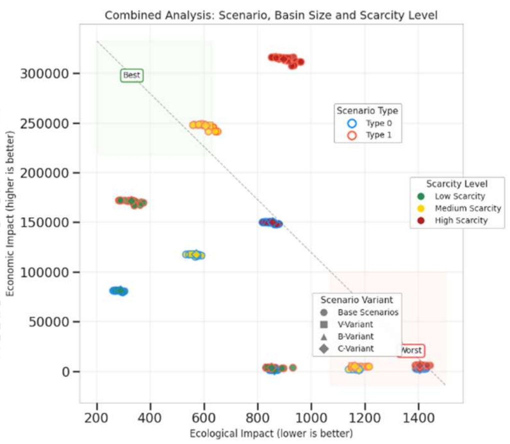
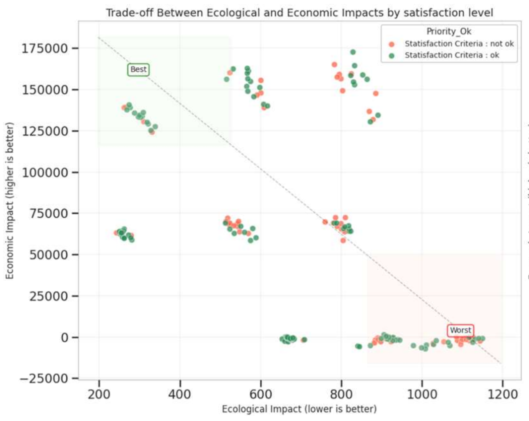


Figure: Comparing the economical and ecological impact of original and custom policies under multiple scenarios.

Based on the above figures (left is the original and the right is the custom), we can clearly see that the economic impact has highly improved for these multiple scenarios.



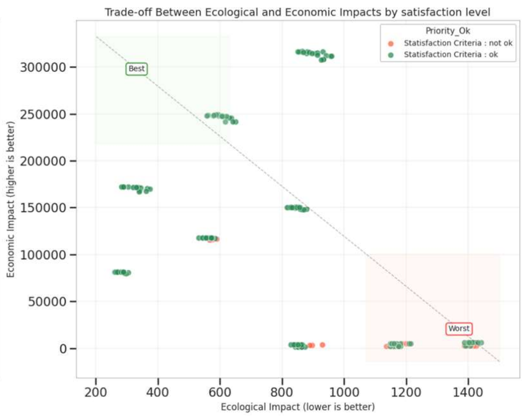


Figure: Comparing the satisfaction level of original and custom policies for different scenarios.

Based on the figure (left is original and the right is the custom), we can clearly see that our custom policy satisfies the requirement in most scenarios (which is important to qualification).