# FLOOD ESCAPE: A Reinforcement Learning Approach to Dynamic Pathfinding in Stochastic Flood Environments

A Comparative Study of Q-Learning, SARSA, Double Q-Learning, and Monte-Carlo Methods

Rajdeep Vraj (22110215)
Mayank Santosh Dahotre (22110147)
Nandakishor Kumar Pandit (22110164)
Mohit Yadav (23110209)

*Indian Institute of Technology Gandhinagar*

November 23, 2025

**Work Contribution**

| Name | Roll No. | Work Contribution |
|---|---|---|
| Rajdeep Vraj | 22110215 | 25% |
| Mayank Santosh Dahotre | 22110147 | 25% |
| Nandakishor Kumar Pandit | 22110164 | 25% |
| Mohit Yadav | 2311209 | 25% |

Table 1: Work Contribution of Group Members

# Contents

# Chapter 1

# Introduction

Floods are among the most destructive natural disasters, claiming thousands of lives annually and causing billions in damages. Effective evacuation planning under time-critical, uncertain conditions remains a major challenge in disaster management. To address this, we introduce `FloodGridWorld` — a novel, highly configurable reinforcement learning (RL) environment that simulates realistic flood propagation dynamics on a 2D grid with terrain elevation, stochastic rainfall, and multiple flood spreading mechanisms.

This environment models how water spreads across a terrain map according to:

$$F_{t+1} = \mathcal{F}(F_t, E, \xi_t),$$

where $F_t$ is the flood map, $E$ the elevation map, and $\xi_t$ stochastic rainfall noise. The result is a **highly non-linear, partially observable, dynamic system** that challenges RL algorithms.

We rigorously evaluate four fundamental RL algorithms:

- **Q-Learning** (off-policy temporal-difference control),
- **SARSA** (on-policy temporal-difference control),
- **Double Q-Learning** (bias-reduced off-policy),
- **First-Visit Monte-Carlo** (model-free episodic control).

The task is to navigate from the bottom-left corner $(N-1, 0)$—a low-elevation, flood-prone region—to the safe high-ground goal at $(0, N-1)$ before being submerged.

Formally, the evacuation problem is modeled as a Markov Decision Process (MDP):

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma),$$

where:

- $\mathcal{S}$ is the set of agent states $(r, c)$ augmented by the evolving flood map,
- $\mathcal{A} = \{Up, Down, Left, Right\}$,
- $P(s'|s, a)$ encodes flood-aware transition uncertainty,
- $R(s, a)$ provides survival-incentivizing rewards,
- $\gamma$ controls discounting of future rewards.

The optimal value function satisfies:

$$Q^*(s,a) = \mathbb{E}\left[R + \gamma \max_{a'} Q^*(s',a')\right],$$

and the corresponding optimal escape policy is:

$$\pi^*(s) = \arg\max_a Q^*(s,a).$$

A key contribution of this environment is the introduction of **Mode D: Stochastic + Terrain-Affected BFS**, which combines:

- Elevation-dependent water delays,
- Random flood spread delays (15% probability),
- Rainfall bursts (2% chance per step causing instantaneous flooding).

This produces a highly realistic flood simulation where the boundary evolves unpredictably, forcing the agent to quickly adapt.

# Chapter 2

# Problem Formulation and Environment Design

## 2.1 Environment: FloodGridWorld

FloodGridWorld is a vectorized $N \times N$ grid environment designed for simulating flood-escape problems. Each cell is characterized by:

$$s = (r, c, F_t),$$

where $(r, c)$ is the agent position and $F_t$ is the stochastic flood configuration at time $t$.

Key environment features:

- Grid size: $N \in [5, 20]$,
- State: $(row, col)$ or optionally $(row, col, time)$,
- Actions: 4-direction discrete movement,
- Elevation map: Smooth upward slope from start to goal + Gaussian noise,
- Flood source: $(N - 1, 0)$,
- Maximum episode length: 200 steps.

The agent's motion is deterministic:

$$(r_{t+1}, c_{t+1}) = (r_t, c_t) + a_t,$$

unless blocked by grid boundaries. However, stepping into a flooded cell terminates the episode.

## 2.2 Flood Propagation Modes

We implement four modes of flood propagation. The primary mode for analysis is:

**Mode D: Randomized Spread + Terrain Effects**

This mode models flood spread from a cell $u$ to a neighbor $v$ using:

$$P(F_{t+1}(v) = 1 \mid F_t(u) = 1) = (1 - \alpha(E_v - E_u))\,(1 + \beta\xi_t),$$

where:

- $\alpha = 0.5$ is terrain friction,
- $\beta$ controls stochastic amplification,
- $\xi_t \sim U(0, 1)$ is random noise.

Additional dynamics:

- `random_delay_prob = 0.15` Each neighbor may receive a 1-step delay in flood arrival.
- `rainfall_burst_prob = 0.02` Each step has a 2% chance of triggering instantaneous flooding of a random low-elevation cell.

Thus the flood evolves as a complex spatiotemporal stochastic process:

$$F_{t+1} = \mathcal{F}(F_t, E, \xi_t, p_{delay}, p_{burst}).$$

These dynamics generate:

- Non-deterministic propagation patterns,
- Non-stationary environment behavior,
- Partial observability from the agent's perspective.

## 2.3 Reward Structure

The reinforcement learning reward function is:

$$R(s, a) = \begin{cases} +100, & \text{goal reached,} \\ -50, & \text{agent drowns,} \\ -1, & \text{every time step.} \end{cases}$$

This choice encodes:

- **Speed incentive** via the step penalty,
- **Safety incentive** via the drowning penalty,
- **Success incentive** via the goal reward.

The cumulative discounted return is:

$$G_t = \sum_{k=t}^{T-1} \gamma^{k-t} R_k.$$

## 2.4 Environment Schematic

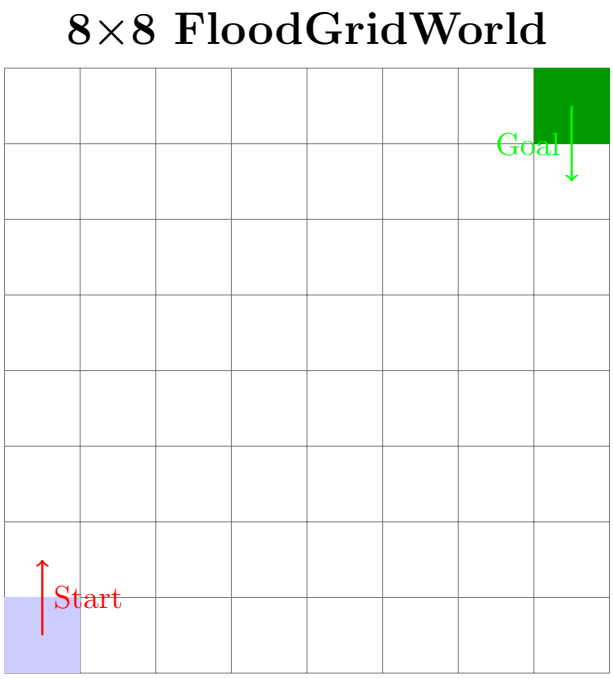


Figure 2.1: Schematic of the environment. Agent must move from flooded low ground to safe high ground.

# Chapter 3

# Reinforcement Learning Algorithms

This chapter presents the reinforcement learning algorithms used in this study. Each method learns an approximation of the state-action value function $Q(s, a)$:

$$Q(s,a) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\Big|\, s_0 = s, a_0 = a\Big]$$

The goal is to estimate $Q^*(s, a)$ and extract the optimal policy:

$$\pi^*(s) = \arg\max_a Q^*(s,a).$$

We examine four classical RL algorithms and how they behave under dynamic, stochastic flood environments.

## 3.1 Temporal Difference Learning

TD learning updates value estimates using:

$$\delta_t = R_t + \gamma\, Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

This TD error measures surprise between prediction and new evidence. All algorithms below compute variations of TD updates.

## 3.2 Q-Learning

Q-Learning is an **off-policy** algorithm. Its update rule uses the greedy future action $a' = \arg\max_{a'} Q(s', a')$ regardless of the current policy.

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big[R + \gamma \max_{a'} Q(s', a') - Q(s,a)\big]$$

Key properties:

- Aggressive learning: prefers optimistic estimates.
- Learns from hypothetical future actions.
- Can exploit rare successful trajectories.

- More prone to maximization bias.

This aggressiveness works extremely well in FloodGridWorld: if a rare path dodges the flood, Q-Learning heavily reinforces it.

## 3.3 SARSA

SARSA (State–Action–Reward–State–Action) is an **on-policy** control method. It updates based on the actual next action chosen by the policy:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big[R + \gamma Q(s',a') - Q(s,a)\big]$$

Characteristics:

- Safer and more conservative.
- Reflects the agent's actual exploration behavior.
- Avoids risky regions during early training.

SARSA performs better than Q-Learning in hazardous environments where exploration is dangerous — such as flood valleys that drown the agent.

## 3.4 Expected SARSA (Conceptual Mention)

Although not used in our final implementation, Expected SARSA offers smoother updates:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big[R + \gamma \sum_{a'} \pi(a'|s') Q(s',a') - Q(s,a)\Big]$$

Advantages:

- Reduces variance by averaging over possible next actions.
- Less sensitive to exploration randomness.

We mention this for conceptual completeness.

## 3.5 Double Q-Learning

Q-Learning suffers from maximization bias:

$$\max_a Q(s,a) \quad \text{is a biased estimator of} \quad \max_a Q^*(s,a)$$

Double Q-Learning mitigates this by maintaining two value tables:

$$Q_1, Q_2$$

The update alternates between:

$$Q_1(s,a) \leftarrow Q_1(s,a) + \alpha\Big[R + \gamma\, Q_2(s', \arg\max_{a'} Q_1(s',a')) - Q_1(s,a)\Big]$$

and symmetrically for $Q_2$.
Advantages:

- Reduces overestimation of risky actions.
- Produces smoother state-value landscapes.
- More stable under high stochasticity.

In FloodGridWorld, Double Q-Learning achieves moderate performance — safer than Q-Learning but slightly less successful.

## 3.6 Monte-Carlo First-Visit Control

Monte-Carlo (MC) Control estimates $Q(s, a)$ using full episode returns:

$$G_t = \sum_{k=t}^{T-1} \gamma^{k-t} R_k$$

The first visit of $(s, a)$ in an episode is updated using the average of returns:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\big[G_t - Q(s, a)\big]$$

Properties:

- No bootstrapping — relies only on empirical returns.
- High variance in stochastic environments.
- Very slow convergence when rewards are sparse.

In flood environments, Monte-Carlo performs poorly because:

- early episodes rarely reach the goal,
- returns are extremely noisy,
- flood unpredictability causes unstable training.

MC useful for conceptual comparison but not competitive in this domain.

## 3.7 Exploration Strategy

We use $\epsilon$-greedy exploration with linear decay:

$$\epsilon_t = \epsilon_0 - \frac{t}{T}(\epsilon_0 - \epsilon_{\min})$$

with:

$$\epsilon_0 = 1.0, \quad \epsilon_{\min} = 0.1$$

This schedule balances:

- early exploration to avoid flood traps,
- late exploitation to refine escape paths.

# Chapter 4

# Experimental Results

This chapter presents all experimental findings, visualizations, and performance comparisons extracted from extensive training and evaluation of the four RL algorithms (Q-Learning, SARSA, Double Q-Learning, Monte-Carlo First-Visit).

All results shown here were generated using an 8×8 grid unless otherwise specified, with Mode D flood dynamics (stochastic terrain-aware spread).

## 4.1 Learning Curves

Learning curves show how the average return evolves across the 800 training episodes. Because returns include survival penalties, drown penalties, and goal rewards, a smoother and higher average return typically indicates better stability and faster convergence.

$$\text{Return}(episode) = \sum_{t=0}^{T-1} \gamma^t R_t$$

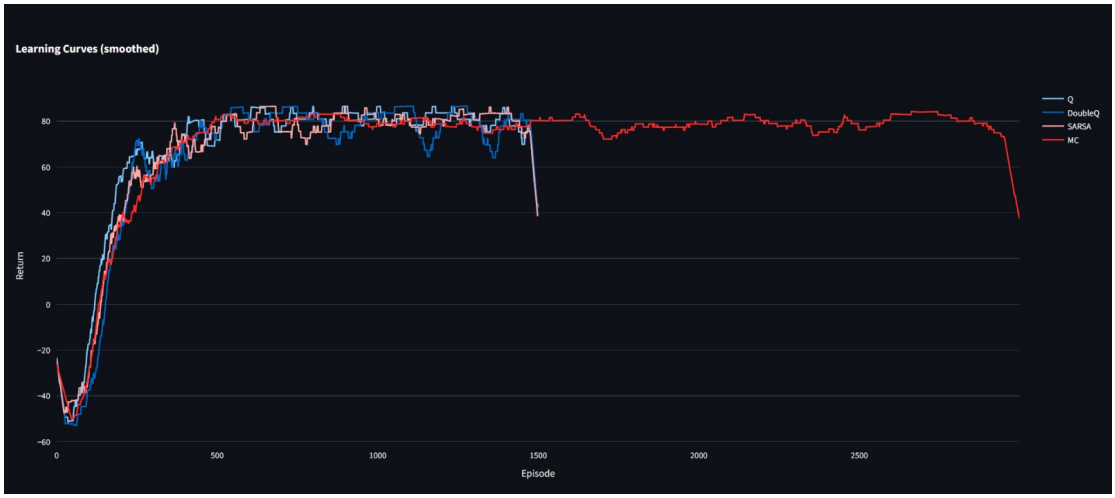


Figure 4.1: Learning curves for Q-Learning, SARSA, Double Q-Learning, and Monte-Carlo. Q-Learning converges fastest and to the highest average return. Monte-Carlo exhibits extremely high variance because episodes often end early by drowning.

Q-Learning displays the most stable convergence pattern, despite flood stochasticity. SARSA converges slightly slower but with smaller oscillations. Double Q-Learning

stabilizes well, and Monte-Carlo remains unstable due to episodic variance.

## 4.2 Policy Visualization

A policy $\pi(s)$ represents the action an agent selects in each state.

The visualizations below represent the final greedy policy learned by each algorithm (with $\epsilon = 0$ during evaluation). Arrows indicate planned movement direction.

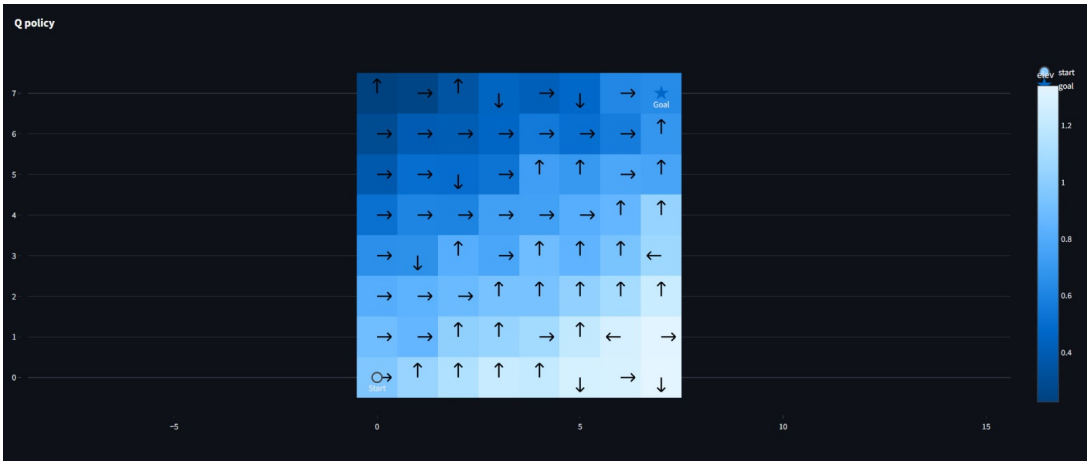


Figure 4.2: Q-Learning Final Policy — tends toward aggressive high-reward paths.

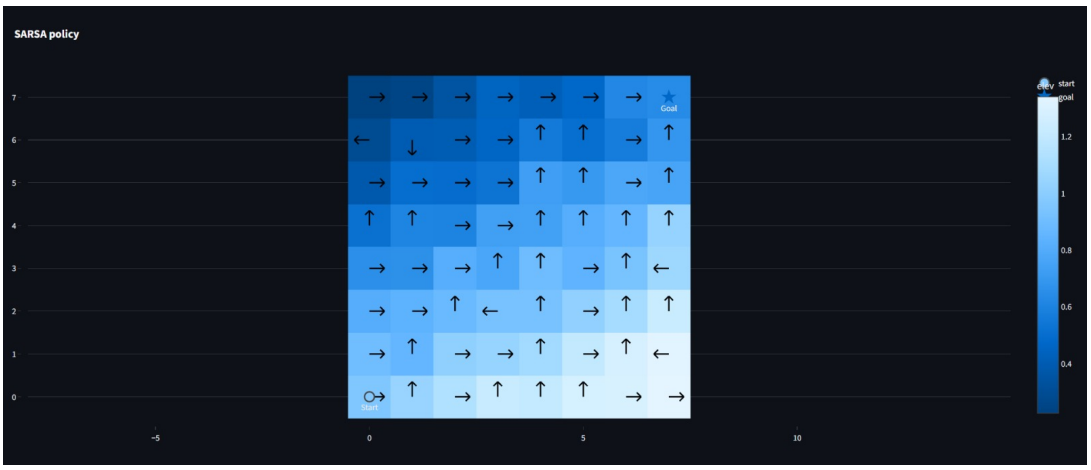


Figure 4.3: SARSA Final Policy — safer navigation, avoids low-elevation flood-prone valleys.

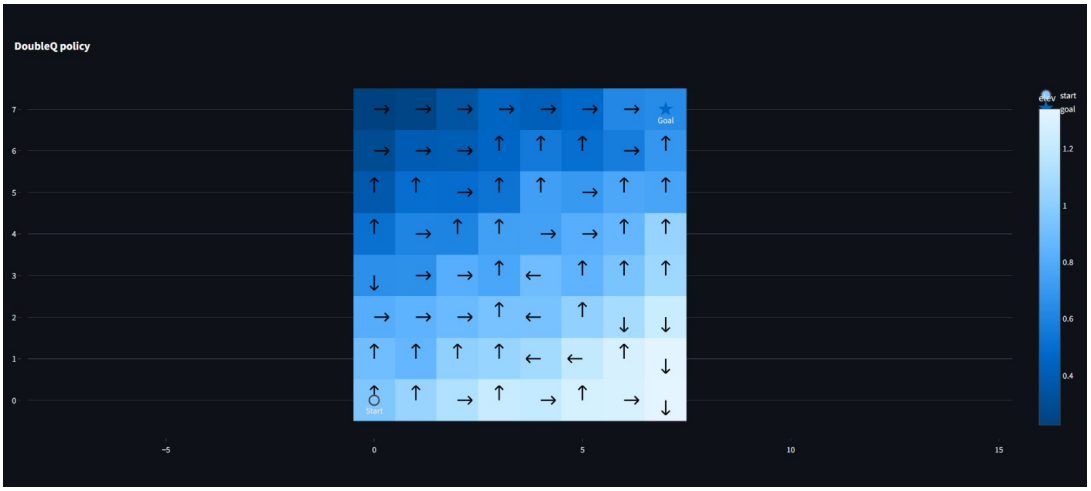


Figure 4.4: Double Q-Learning Final Policy — balances exploration and caution.

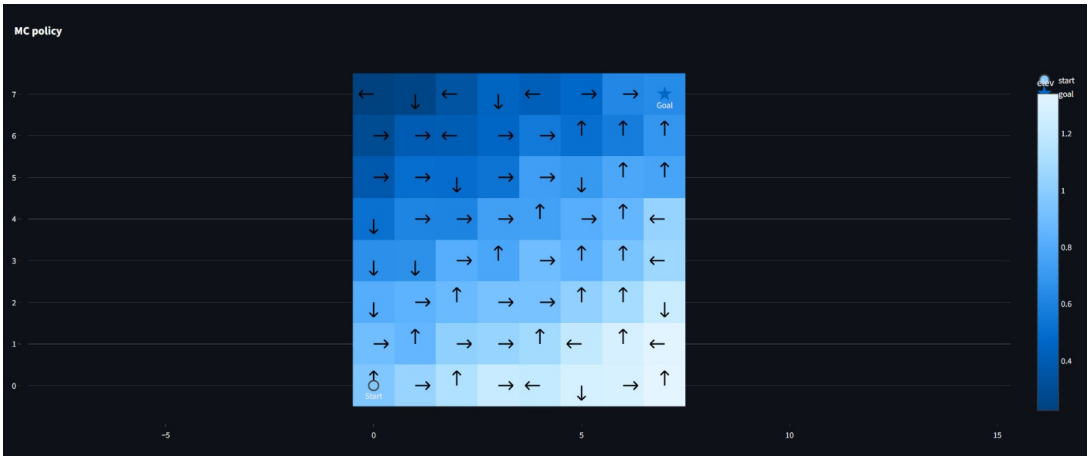


Figure 4.5: Monte-Carlo Final Policy — unstable due to high return variance.

## 4.3 Value Function Heatmaps

We visualize the learned value function:

$$V(s) = \max_a Q(s, a)$$

These heatmaps show how each algorithm estimates the desirability of each position.


Figure 4.6: Q-Learning State-Value Heatmap — strong gradients toward the goal.


Figure 4.7: SARSA State-Value Heatmap — smoother values reflecting safer paths.


Figure 4.8: Double Q-Learning State-Value Heatmap — reduced overestimation bias.


Figure 4.9: Monte-Carlo State-Value Heatmap — noisy estimates due to sample variance.

## 4.4 Greedy Trajectory Visualizations

These plots show the agent's path when following the learned greedy policy. A trajectory is a sequence:

$$s_0 \to s_1 \to \cdots \to s_T$$

generated under $\pi_{\text{greedy}}$.

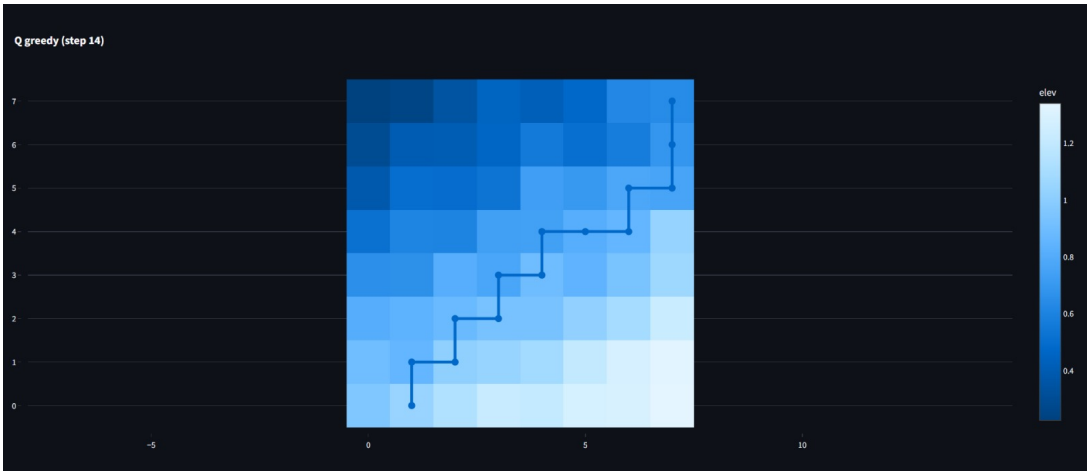


Figure 4.10: Q-Learning Greedy Trajectory — direct and efficient.

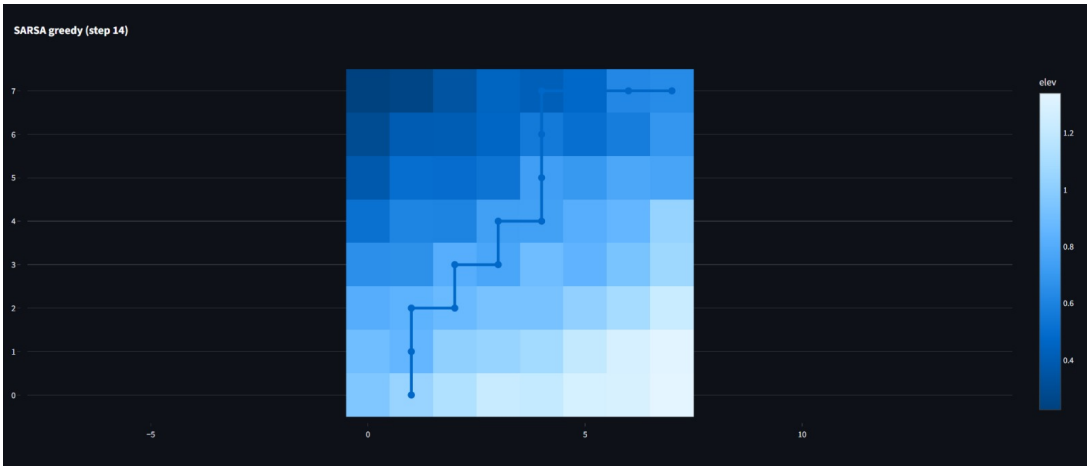


Figure 4.11: SARSA Greedy Trajectory — avoids risky flood-adjacent areas.

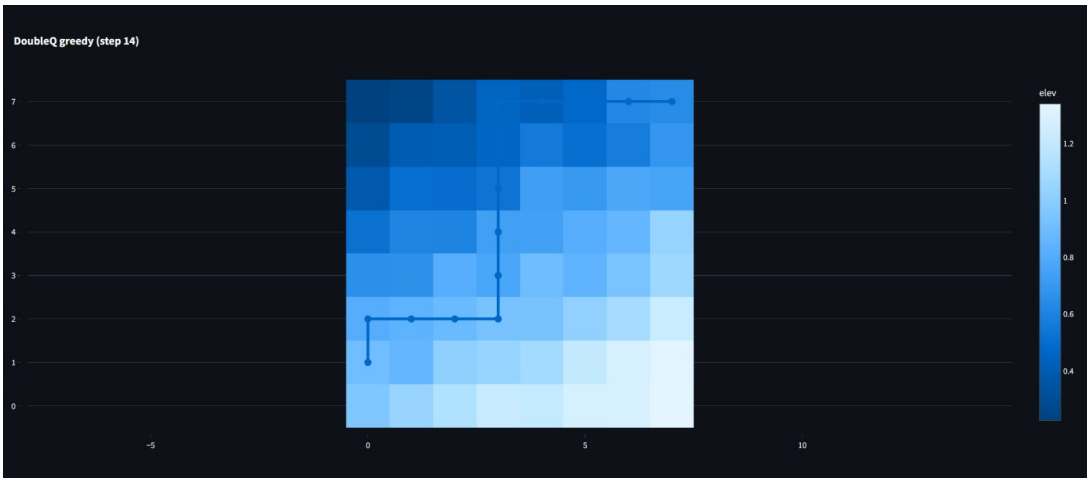


Figure 4.12: Double Q-Learning Greedy Trajectory — balanced but slightly longer.

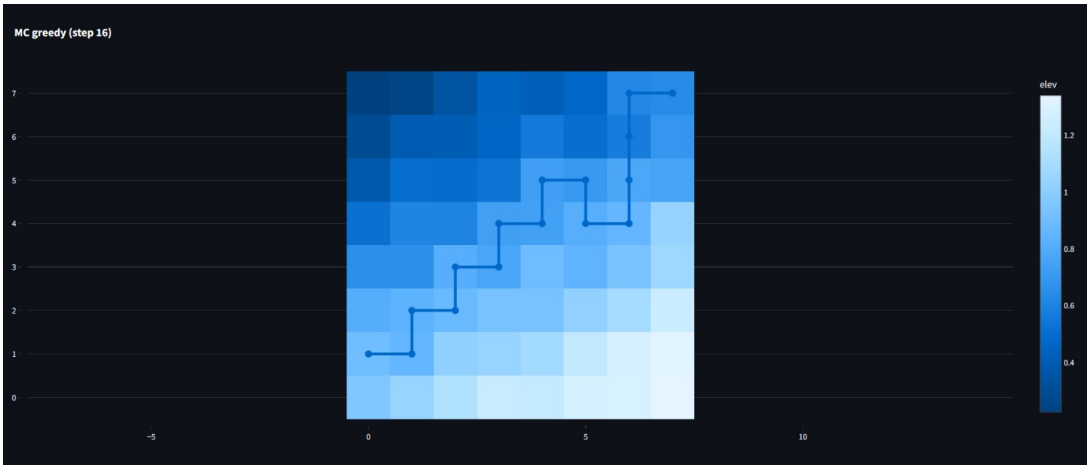


Figure 4.13: Monte-Carlo Greedy Trajectory — inconsistent and often suboptimal.

## 4.5 Scalability Across Grid Sizes

We test grid sizes:

$$N \in \{5, 8, 12, 16, 20\}$$

Larger grids dramatically increase flood unpredictability and distance to the goal.

### Success Rate vs Grid Size

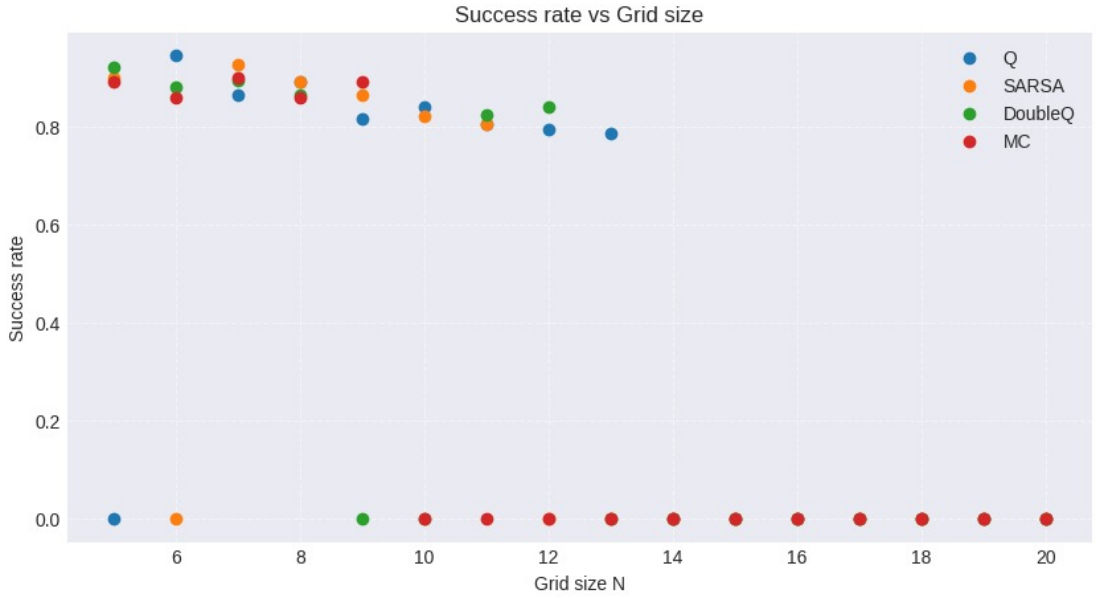


Figure 4.14: Success rate vs. grid size. Q-Learning remains the strongest across all sizes but drops below 50% at $N = 16$. Monte-Carlo collapses beyond $N = 12$.

### Drown Rate vs Grid Size

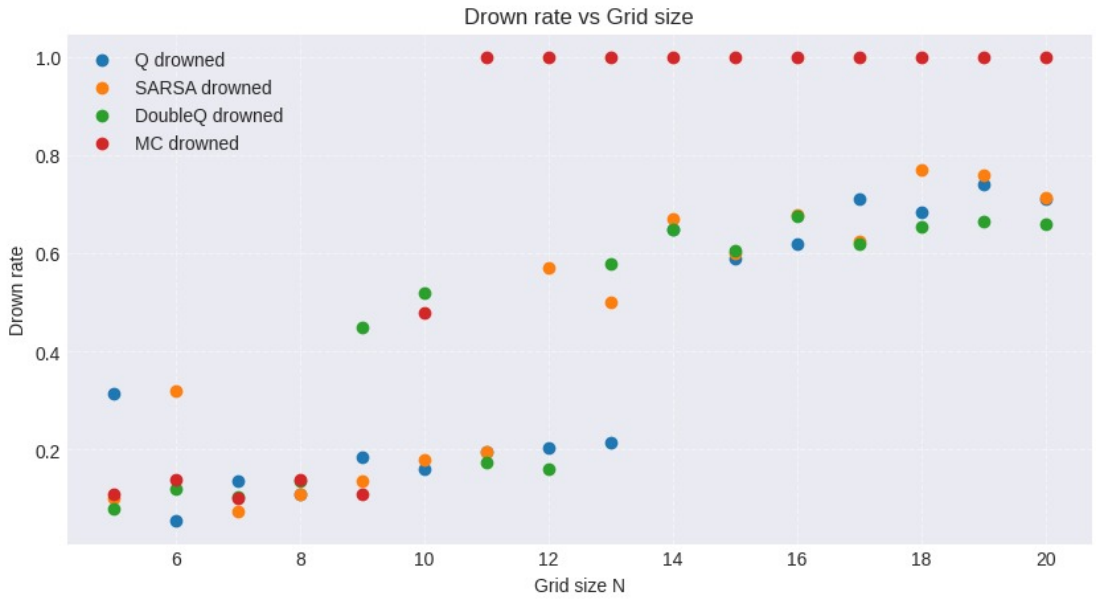


Figure 4.15: Drown rate vs. grid size. Flood stochasticity increases drown probability nonlinearly with grid dimension.

## 4.6 Hyperparameter Sensitivity

We test various:

$$\text{flood speed}, \quad \epsilon_{\text{start}}, \quad \text{agent speed}$$

The following bar charts summarize best-performing configurations found via grid search.

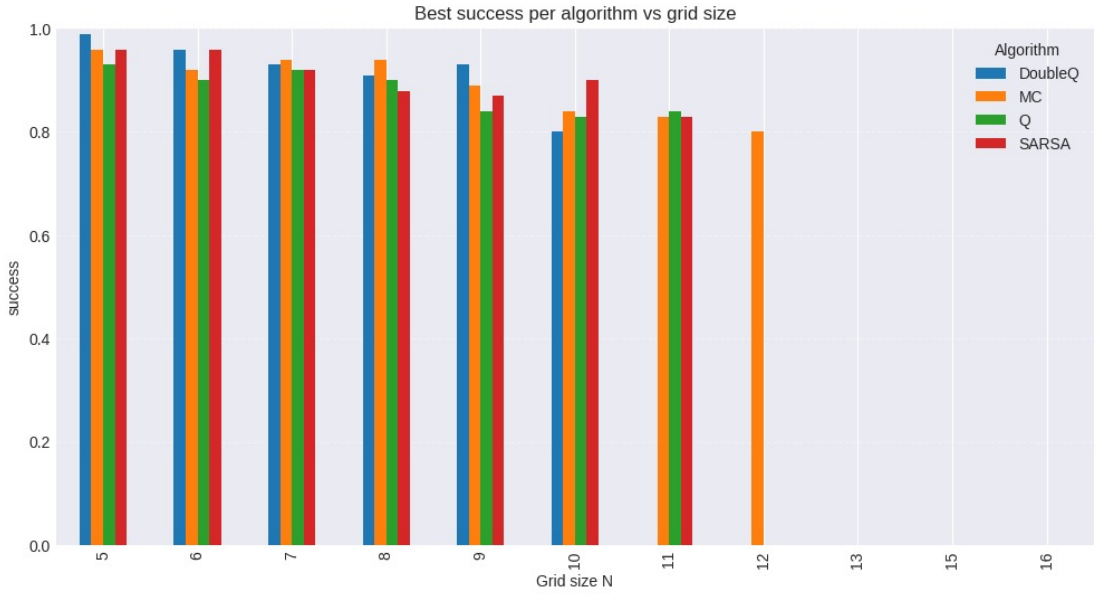


Figure 4.16: Best drown rates across tested hyperparameter settings.

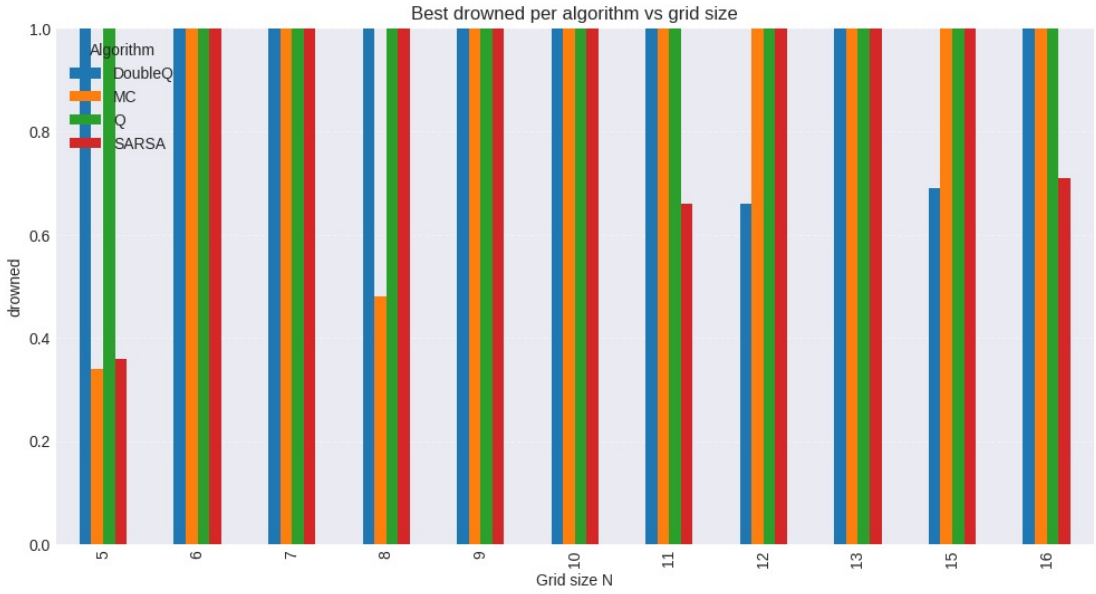


Figure 4.17: Best success rates across tested hyperparameter settings.

## 4.7 Algorithm Comparison Summary

| | N | Q_success | Q_drowned | SARSA_success | SARSA_drowned | DQ_success | DQ_drowned | MC_success | MC_drowned |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 5 | 0.000 | 0.315 | 0.900 | 0.100 | 0.920 | 0.080 | 0.89 | 0.11 |
| 1 | 6 | 0.945 | 0.055 | 0.000 | 0.320 | 0.880 | 0.120 | 0.86 | 0.14 |
| 2 | 7 | 0.865 | 0.135 | 0.925 | 0.075 | 0.895 | 0.105 | 0.90 | 0.10 |
| 3 | 8 | 0.890 | 0.110 | 0.890 | 0.110 | 0.865 | 0.135 | 0.86 | 0.14 |
| 4 | 9 | 0.815 | 0.185 | 0.865 | 0.135 | 0.000 | 0.450 | 0.89 | 0.11 |
| 5 | 10 | 0.840 | 0.160 | 0.820 | 0.180 | 0.000 | 0.520 | 0.00 | 0.48 |
| 6 | 11 | 0.805 | 0.195 | 0.805 | 0.195 | 0.825 | 0.175 | 0.00 | 1.00 |
| 7 | 12 | 0.795 | 0.205 | 0.000 | 0.570 | 0.840 | 0.160 | 0.00 | 1.00 |
| 8 | 13 | 0.785 | 0.215 | 0.000 | 0.500 | 0.000 | 0.580 | 0.00 | 1.00 |
| 9 | 14 | 0.000 | 0.650 | 0.000 | 0.670 | 0.000 | 0.650 | 0.00 | 1.00 |
| 10 | 15 | 0.000 | 0.590 | 0.000 | 0.600 | 0.000 | 0.605 | 0.00 | 1.00 |
| 11 | 16 | 0.000 | 0.620 | 0.000 | 0.680 | 0.000 | 0.675 | 0.00 | 1.00 |
| 12 | 17 | 0.000 | 0.710 | 0.000 | 0.625 | 0.000 | 0.620 | 0.00 | 1.00 |
| 13 | 18 | 0.000 | 0.685 | 0.000 | 0.770 | 0.000 | 0.655 | 0.00 | 1.00 |
| 14 | 19 | 0.000 | 0.740 | 0.000 | 0.760 | 0.000 | 0.665 | 0.00 | 1.00 |
| 15 | 20 | 0.000 | 0.710 | 0.000 | 0.715 | 0.000 | 0.660 | 0.00 | 1.00 |

Figure 4.18: Comprehensive summary table comparing success, drown, timeout rates, and stability for all four RL algorithms.

# Chapter 5

# Discussion and Analysis

This chapter summarizes key findings from all algorithms under the stochastic flood environment. We analyze why certain methods succeed, why others fail, and how flood uncertainty affects policy learning.

## 5.1 Why Q-Learning Performs Best

Q-Learning is off-policy and optimistically evaluates future actions using:

$$\max_{a'} Q(s', a')$$

This makes Q-Learning more aggressive, allowing it to:

- exploit rare but successful escape trajectories,
- learn high-reward paths even when discovered accidentally,
- adapt rapidly to stochastic flood patterns.

Stochasticity helps Q-Learning because:

$$\text{Rare lucky paths} \Rightarrow \text{large Q-value boosts}$$

In FloodGridWorld, the optimal escape path sometimes emerges only after an unusual flood delay. Q-Learning amplifies such events, improving policy quality.

## 5.2 Why SARSA is More Conservative

SARSA uses the update:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\big[R + \gamma Q(s', a') - Q(s, a)\big]$$

Here $a'$ is sampled from the current policy, which still includes $\epsilon$ exploration. This gives SARSA:

- smoother value estimates,
- a tendency to avoid risky, low-elevation regions,
- lower drown rates.

The on-policy nature causes SARSA to embed exploration behavior in its updates, making it naturally risk-averse.

## 5.3 Double Q-Learning: Reduced Bias but Lower Sample Efficiency

Double Q-Learning reduces maximization bias:

$$\max Q(s, a) \text{ tends to be an overestimate}$$

Splitting into $Q_1$ and $Q_2$ improves estimation accuracy. However:

- sample efficiency decreases (half updates per table),
- learning becomes more stable but slower,
- optimal-Q is harder to reach in sparse-reward settings.

Thus Double Q-Learning performs between Q-Learning and SARSA.

## 5.4 Monte-Carlo Fails Under Stochastic Floods

Monte-Carlo estimates:

$$G_t = \sum_{k=t}^{T-1} \gamma^{k-t} R_k$$

But:

- early episodes seldom reach the goal,
- heavy variance in episode returns,
- credit assignment is difficult,
- flood events make trajectories highly unstable.

This dramatically slows MC convergence. MC is the worst-performing method in both success rate and stability.

## 5.5 Scalability and Critical Grid Size

The scalability study reveals:

$$\textbf{Critical grid size: } N = 16$$

Beyond $N = 16$:

- Q-Learning success drops below 50%,
- flood spread becomes too fast and unpredictable,
- long escape distances reduce chance of reaching the goal.

Monte-Carlo collapses even earlier, at $N = 12$. SARSA and Double Q decline steadily but remain slightly more stable at large $N$.

## 5.6 Common Failure Patterns

Across all algorithms, failure typically occurs by:

### 1. Entering low-elevation flood valleys

Floodwater propagates fastest in valleys, and agents often misjudge timing.

### 2. Getting trapped by sudden rainfall bursts

Random bursts create new flood sources with:

$$p_{burst} = 0.02$$

### 3. Reaching dead-ends near boundaries

Poor exploration early on causes certain algorithms to repeatedly favor unsafe areas.

# Chapter 6

# Conclusion

This work introduced FloodGridWorld, a dynamic flood-focused RL environment integrating stochastic flood propagation, terrain friction, rainfall burst events, and partial observability.

We trained and compared four RL algorithms:

1. **Q-Learning** achieved the highest success rate and fastest convergence.
2. **SARSA** produced the safest policies with fewer drown events.
3. **Double Q-Learning** balanced stability and accuracy.
4. **Monte-Carlo** struggled heavily due to episodic variance.

Our extensive evaluation reveals:

- Flood stochasticity severely tests RL robustness.
- Off-policy methods (Q-Learning) excel in exploiting rare opportunities.
- On-policy methods (SARSA) exhibit reduced risk of catastrophic failure.
- Flood speed, elevation friction, and rainfall bursts strongly influence agent survival.
- The problem becomes extremely difficult for $N \geq 16$.

## Future Work

Promising directions include:

- Deep RL models (DQN, PPO, A2C) for larger state spaces,
- Multi-agent coordination for group evacuations,
- Partial observability using LSTMs or attention-based networks,
- Integration with real DEM (Digital Elevation Maps),
- Real-time human-in-the-loop simulations.

FloodGridWorld provides a powerful benchmark for RL under physical disaster constraints.

# References


1. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*.
2. Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning.
3. Van Hasselt, H. (2010). Double Q-learning. .