Reward Shaping Analysis in Reinforcement Learning: A Comparative Study of Monte Carlo and SARSA

Ahmad Tawil Yosef Jirees

January 2026

Abstract

This paper investigates the effects of reward shaping on reinforcement learning performance in a stochastic grid-world environment. We implement and compare two fundamental RL algorithms—Every-Visit Monte Carlo Control and SARSA(0)—under various reward shaping strategies. Our experimental results demonstrate that safety-based shaping significantly improves learning efficiency, with SARSA achieving 71.46% throughput compared to Monte Carlo's 47.71%. We conduct comprehensive hyperparameter sensitivity analyses and provide insights into the exploration-exploitation tradeoff inherent in these methods.

1 Introduction

Reward shaping is a technique used to accelerate learning in reinforcement learning by modifying the reward signal without changing the underlying task [1]. The shaped reward is defined as:

$$R'(s, a, s') = R(s, a, s') + F(s, a, s') \tag{1}$$

where F is the shaping function designed by the practitioner. A special class of shaping functions, called *potential-based shaping*, guarantees policy invariance:

$$F(s, a, s') = \gamma\Phi(s') - \Phi(s) \tag{2}$$

where $\Phi(s)$ is an arbitrary potential function.

In this work, we explore how different shaping strategies affect learning speed, throughput, and convergence in a challenging stochastic environment.

2 Experimental Setup

2.1 Environment

We use the FrozenLake environment from Gymnasium with the following configuration:

- **Grid Size:** 6×6 (36 total states)
- **Hole Density:** 15% (5 holes)
- **Dynamics:** Slippery with success rate 0.7
- **Discount Factor:** $\gamma = 1.0$ (as required)
- **Random Seed:** 242 (for reproducibility)

The environment presents a challenging navigation problem where the agent must reach the goal while avoiding holes. The stochastic dynamics ($p_{\text{success}} = 0.7$) mean that 30% of the time, actions result in perpendicular movement, significantly complicating the learning task. The agent receives a reward of +1 for reaching the goal and 0 otherwise.

Figure 1: FrozenLake 6×6 environment with 15% hole density.

2.2 Algorithms

We implement two fundamental RL algorithms from scratch:

2.2.1 Every-Visit Monte Carlo Control

Updates Q-values after complete episodes using:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[G - Q(s, a)] \tag{3}$$

where $G = \sum_{t=0}^{T} \gamma^t R_t$ is the return and α is the learning rate.

2.2.2 SARSA(0)

On-policy TD control that updates after each step:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)] \tag{4}$$

Both algorithms use ϵ-greedy action selection with optimized hyperparameters per method (Table 1).

2.3 Reward Shaping Methods

We evaluate five reward shaping strategies:

1. Baseline (No Shaping) The original environment reward: $R'(s, a, s') = R(s, a, s')$.

2. Step-Cost Shaping Adds a small penalty per step to encourage shorter paths:

$$R'(s, a, s') = R(s, a, s') + c \tag{5}$$

where $c = -0.01$ (MC) or $c = -0.005$ (SARSA).

3. Potential-Based Distance Shaping Uses Manhattan distance to the goal as the potential function:

$$\Phi(s) = -d(s), \quad F(s, a, s') = \beta(\gamma\Phi(s') - \Phi(s)) \tag{6}$$

where $d(s)$ is the Manhattan distance to the goal and β controls shaping strength.

4. Safety-Based Shaping (Custom #1) Our first custom method encourages the agent to maintain distance from holes:

$$\Phi(s) = \min_{h \in H} d(s, h), \quad F(s, a, s') = \beta(\gamma\Phi(s') - \Phi(s)) \tag{7}$$

where H is the set of hole positions, with $\beta = 0.05$.

Intuition: In a stochastic environment with holes, safety is crucial. This potential-based shaping biases exploration toward safer regions while preserving the optimal policy theoretically. States farther from holes have higher potential, so moving away from holes yields positive shaping rewards.

5. Exploration Bonus Shaping (Custom #2) Our second custom method provides novelty bonuses based on state visitation counts:

$$\Phi(s) = -\sqrt{\text{count}(s) + 1}, \quad F(s, a, s') = \beta(\gamma\Phi(s') - \Phi(s)) \tag{8}$$

where $\text{count}(s)$ is the number of times state s has been visited, with $\beta = 0.05$ (MC) or $\beta = 0.02$ (SARSA).

Intuition: Novel states have higher potential (less negative), so moving to unexplored states yields positive shaping. This encourages diverse exploration, particularly beneficial for Monte Carlo which learns from complete episodes. The square root provides diminishing returns as states are revisited. *Visit counts persist across episodes* (critical for MC's episodic learning).

2.4 Evaluation Metrics

We use two primary metrics:

Throughput The cumulative success rate over episodes:

$$\text{Throughput}_t = \frac{\sum_{i=1}^{t} \mathbb{1}[\text{goal reached in episode } i]}{t} \tag{9}$$

Real Return The sum of *original* (unshaped) rewards per episode, ensuring shaped rewards don't artificially inflate performance metrics.

All experiments use 20 independent runs with 10,000 episodes each to compute 95% confidence intervals.

Table 1: Optimized hyperparameters used for key methods

Method	ε	α	**Other Parameters**
MC_baseline	0.05	0.05	-
MC_exploration	0.1	0.05	$\beta = 0.05$
MC_potential	0.05	0.05	$\beta = 0.5$
MC_safety	0.05	0.05	$\beta = 0.05$
MC_step	0.05	0.05	$c = -0.01$
SARSA_baseline	0.1	0.2	-
SARSA_exploration	0.05	0.2	$\beta = 0.02$
SARSA_potential	0.05	0.1	$\beta = 1.0$
SARSA_safety	0.05	0.2	$\beta = 0.05$
SARSA_step	0.1	0.2	$c = -0.005$

3 Results

3.1 Monte Carlo: Shaping Method Comparison

Figure 2 shows the performance of all shaping methods with Monte Carlo.

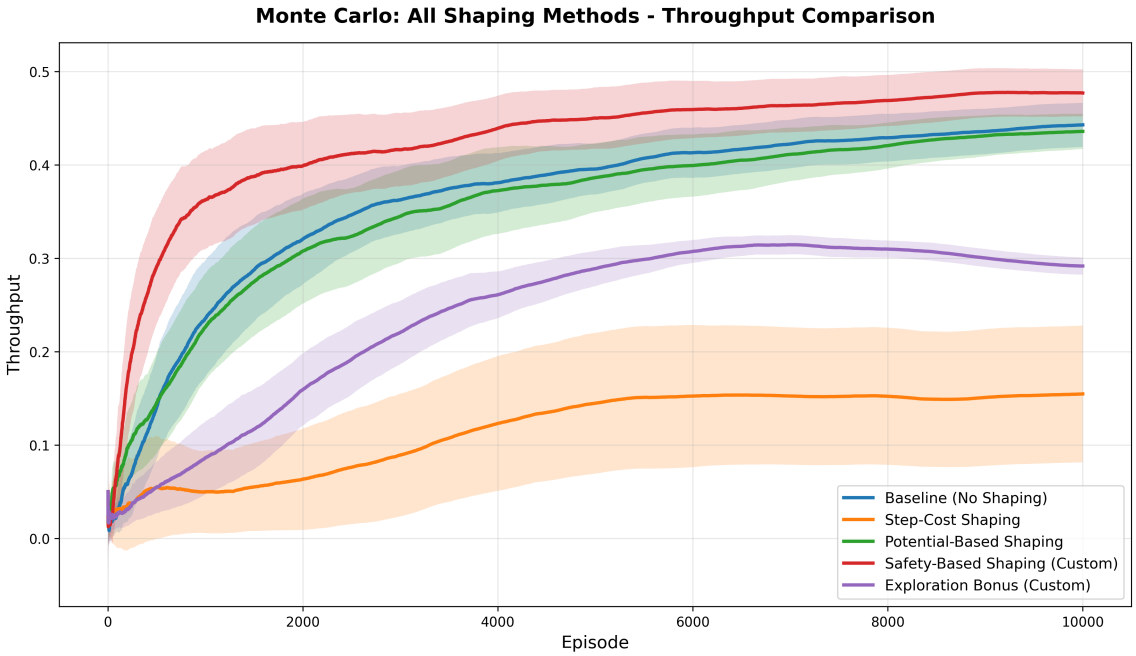


Figure 2: Monte Carlo: Throughput comparison across all shaping methods (mean ± 95% CI, 20 runs, 10,000 episodes).

Table 2: Monte Carlo: Final performance summary (last 100 episodes)

Method	Throughput	Avg Return
Safety-Based Shaping (Custom)	**47.71% ± 5.76%**	**0.400 ± 0.490**
Baseline (No Shaping)	44.30% ± 5.36%	0.600 ± 0.477
Potential-Based Shaping	43.60% ± 4.32%	0.400 ± 0.490
Exploration Bonus (Custom)	29.18% ± 2.11%	0.450 ± 0.497
Step-Cost Shaping	15.47% ± 16.67%	0.200 ± 0.400

Key Findings:

- **Safety-based shaping** achieves the best throughput (47.71%), providing +7.7% improvement over baseline.
- **Baseline** performs well (44.30%), suggesting MC can learn effectively without shaping given sufficient episodes.
- **Step-cost shaping fails dramatically** (15.47%). The constant penalty overwhelms the sparse goal reward, causing the agent to minimize episode length by falling into holes early.
- **Exploration bonus** underperforms (29.18%)—likely because the novelty bonus competes with goal-seeking behavior.

3.2 SARSA: Shaping Method Comparison

Figure 3 shows SARSA's performance across all shaping methods.

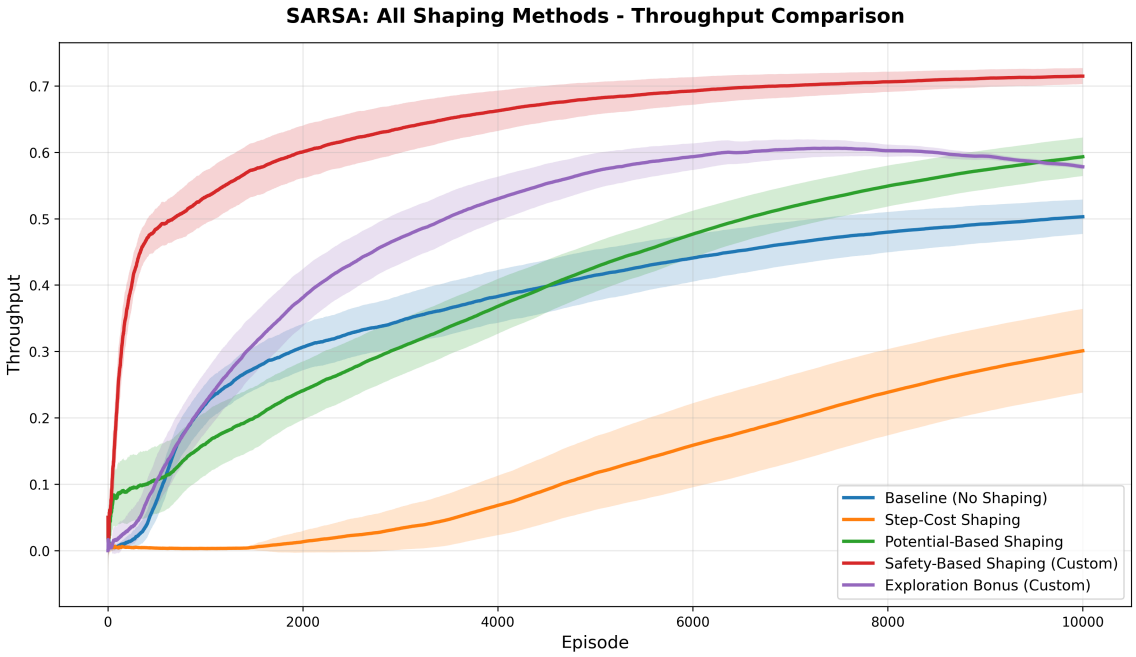


Figure 3: SARSA: Throughput comparison across all shaping methods (mean ± 95% CI, 20 runs, 10,000 episodes).

Table 3: SARSA: Final performance summary (last 100 episodes)

Method	Throughput	Avg Return
Safety-Based Shaping (Custom)	**71.46% ± 2.74%**	**0.650 ± 0.477**
Potential-Based Shaping	59.32% ± 6.68%	0.850 ± 0.357
Exploration Bonus (Custom)	57.81% ± 0.89%	0.400 ± 0.490
Baseline (No Shaping)	50.29% ± 5.90%	0.800 ± 0.400
Step-Cost Shaping	30.10% ± 14.13%	0.650 ± 0.477

Key Findings:

- **Safety-based shaping dominates** with 71.46% throughput—a +42.1% improvement over baseline.
- **SARSA benefits far more from shaping than MC**: The online nature of SARSA allows it to incorporate shaping signals incrementally.
- **SARSA baseline** (50.29%) outperforms *all* Monte Carlo variants—demonstrating on-policy TD learning's superiority in stochastic environments.
- **All methods show smoother curves** than MC, reflecting SARSA's lower variance due to bootstrapping.

3.3 Monte Carlo vs SARSA: Best Methods

Figure 4 compares the best-performing variant of each algorithm.

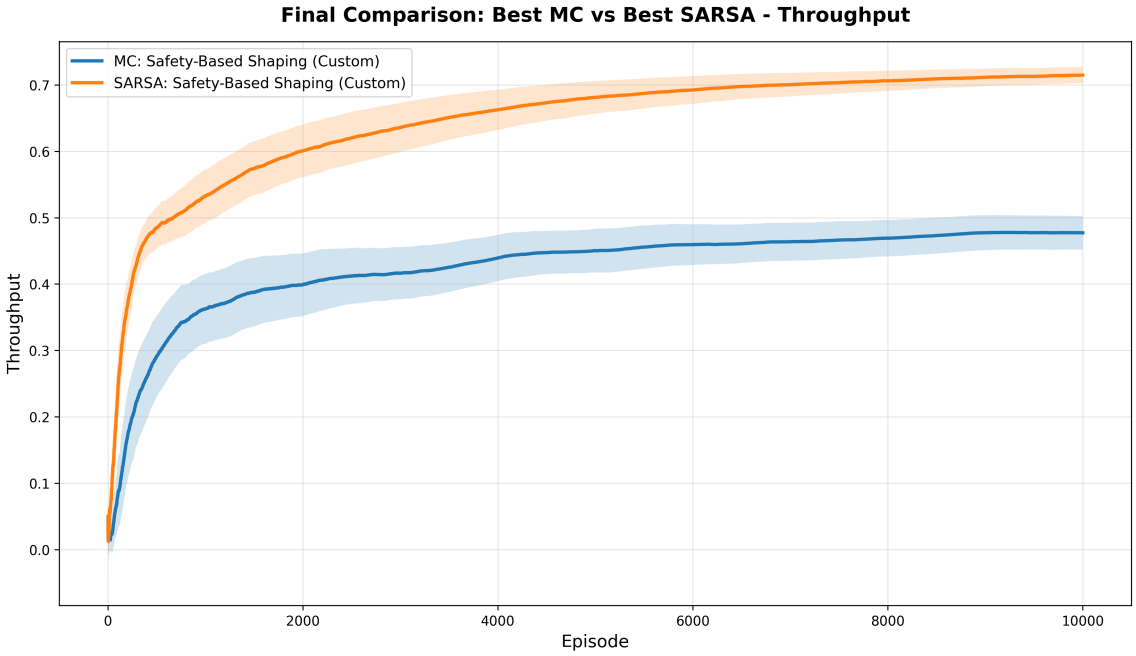


Figure 4: Final comparison: Best MC (Safety-Based) vs Best SARSA (Safety-Based) throughput.

Table 4: Best MC vs Best SARSA: Final comparison

Algorithm	**Throughput**	**Avg Return**
SARSA + Safety-Based Shaping	**71.46% ± 2.74%**	**0.650 ± 0.477**
MC + Safety-Based Shaping	47.71% ± 5.76%	0.400 ± 0.490

SARSA outperforms MC by +49.8% (71.46% vs 47.71%). This dramatic difference is explained by:

- **Faster credit assignment:** SARSA updates Q-values after every step, while MC must wait for episode completion.
- **Better handling of stochasticity:** Bootstrapping in SARSA smooths out the high variance from the 0.7 success rate.
- **More effective shaping integration:** Step-level updates allow SARSA to incorporate safety signals continuously.

3.4 Policy Visualization

Figure 5 shows the learned policies for the best methods.

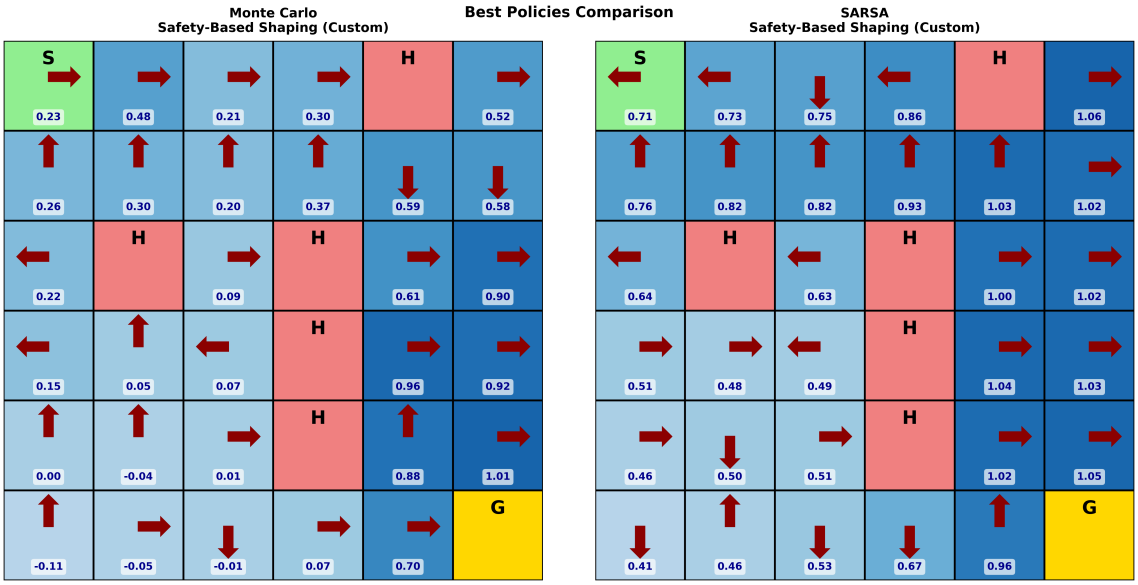


Figure 5: Learned policies: MC Safety (left) shows negative Q-values and uncertainty; SARSA Safety (right) shows confident, all-positive Q-values (0.41-1.06) with clearer value gradients toward the goal.

The SARSA policy demonstrates significantly higher and more confident value estimates throughout the grid. All Q-values are positive (0.41-1.06), indicating better value function quality compared to MC's range (-0.11 to 1.01 with several negative values). Both policies successfully navigate to the goal while avoiding holes, but SARSA's stronger value gradients suggest more reliable policy execution.

4 Hyperparameter Sensitivity Analysis

4.1 Epsilon (ϵ) Sensitivity

We varied the exploration rate $\epsilon \in \{0.05, 0.2, 0.5\}$ while fixing $\alpha = 0.1$ for SARSA with safety-based shaping.

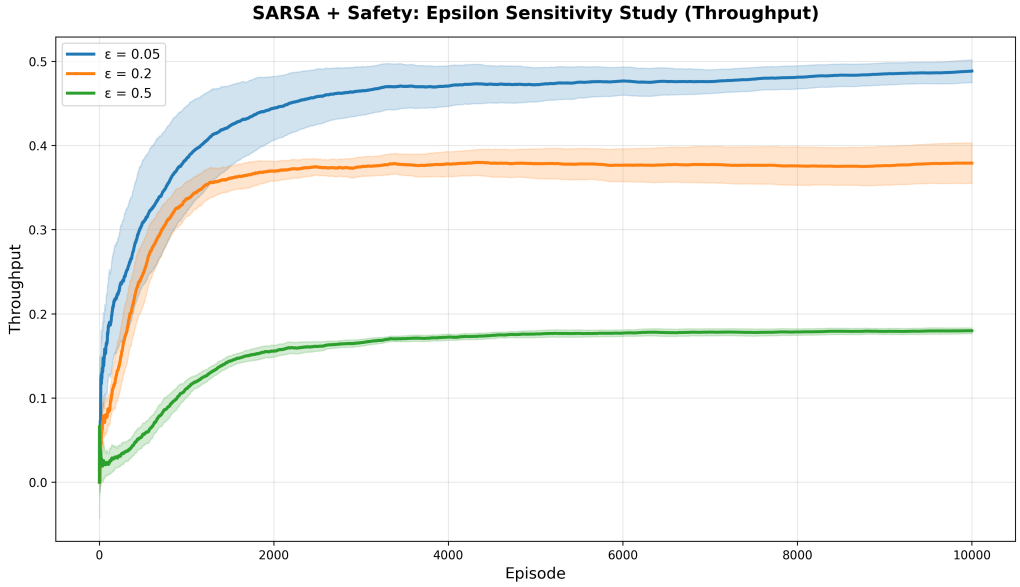


Figure 6: SARSA + Safety: Epsilon sensitivity on throughput (mean ± 95% CI, 10 runs, 10,000 episodes).

Table 5: Epsilon sensitivity results (SARSA + Safety)

Epsilon (ϵ)	**Throughput (last 100)**	**Avg Return (last 100)**
0.05	**48.79% ± 2.16%**	**0.556 ± 0.497**
0.2	37.89% ± 3.87%	0.390 ± 0.488
0.5	17.98% ± 0.57%	0.184 ± 0.387

Analysis:

- $\epsilon = 0.05$ **is optimal** (48.79% throughput), balancing exploration and exploitation.
- $\epsilon = 0.2$ shows degradation: 22% relative decrease.
- $\epsilon = 0.5$ **(excessive exploration)** performs worst (17.98%). Too much random action selection prevents convergence.

Exploration-Exploitation Tradeoff: This demonstrates the fundamental RL dilemma. The environment's 70% success rate provides inherent exploration through slippage. With $\epsilon = 0.05$, effective randomness is ∼33% $(0.05 + 0.3(0.95))$, sufficient for discovery without preventing convergence. Higher ϵ values compound this randomness excessively.

4.2 Alpha (α) Sensitivity

We varied the learning rate $\alpha \in \{0.05, 0.2, 0.5\}$ while fixing $\epsilon = 0.1$ for SARSA with safety-based shaping.

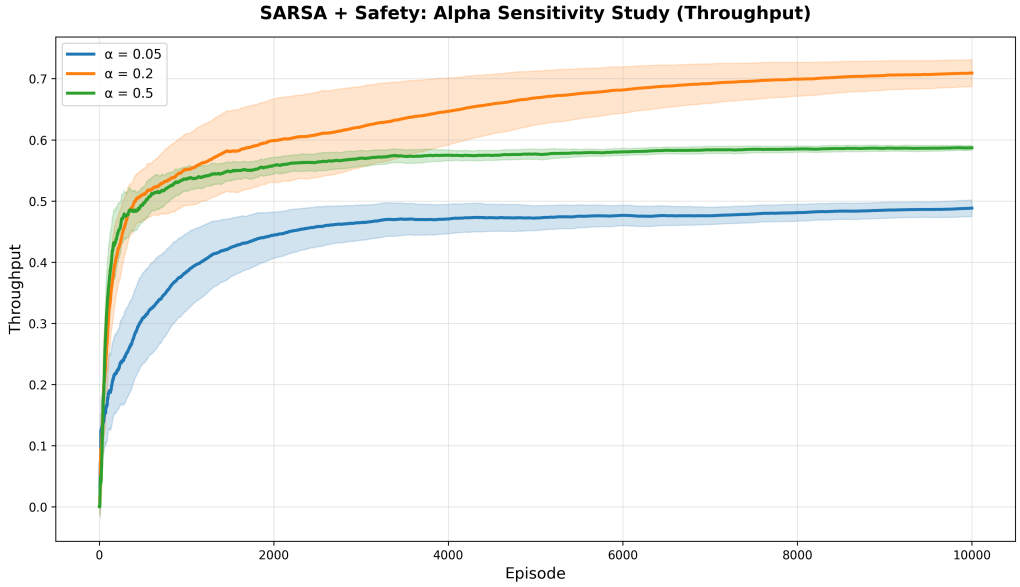


Figure 7: SARSA + Safety: Alpha sensitivity on throughput (mean ± 95% CI, 10 runs, 10,000 episodes).

Table 6: Alpha sensitivity results (SARSA + Safety)

Alpha (α)	**Throughput (last 100)**	**Avg Return (last 100)**
0.2	**70.89% ± 3.54%**	**0.746 ± 0.435**
0.5	58.68% ± 0.65%	0.611 ± 0.488
0.05	48.79% ± 2.16%	0.556 ± 0.497

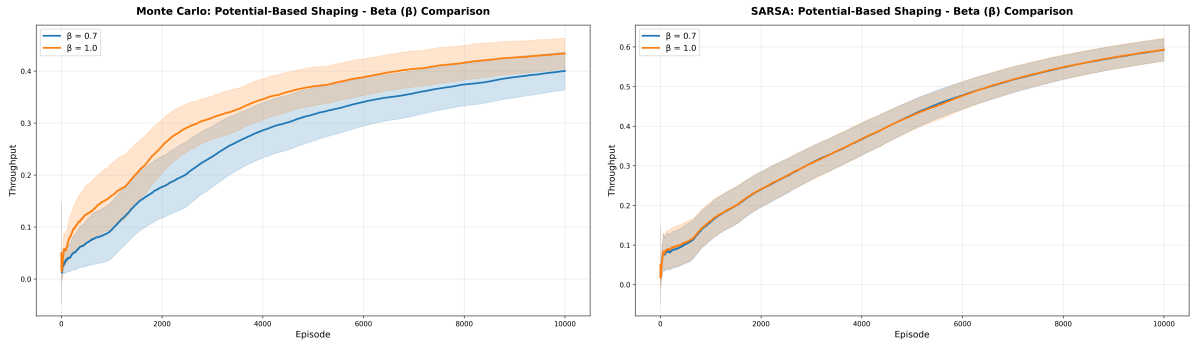
Analysis:

- $\alpha = 0.2$ **achieves best throughput** (70.89%), enabling fast adaptation to new information while maintaining stability.
- $\alpha = 0.05$ **(slow learning)** performs poorly (48.79%)—too slow to incorporate new information in the stochastic environment.
- $\alpha = 0.5$ **(fast learning)** shows reduced performance (58.68%) and higher variance due to overshooting in Q-value estimates.

The optimal $\alpha = 0.2$ strikes a balance between incorporating new information quickly (important in stochastic environments) and maintaining stability.

4.3 Beta (β) Sensitivity: Potential-Based Shaping

We compared $\beta \in \{0.7, 1.0\}$ for potential-based distance shaping.



(a) Monte Carlo (b) SARSA

Figure 8: Beta sensitivity for potential-based shaping: $\beta = 0.7$ vs $\beta = 1.0$ (mean $\pm$ 95% CI).

Table 7: Beta sensitivity results for potential-based shaping

Algorithm	**Beta (β)**	**Throughput (last 100)**	**Avg Return (last 100)**
Monte Carlo	1.0	**43.34% $\pm$ 6.75%**	0.495 $\pm$ 0.500
	0.7	39.96% $\pm$ 8.31%	0.520 $\pm$ 0.500
SARSA	1.0	**59.23% $\pm$ 6.61%**	0.775 $\pm$ 0.418
	0.7	59.18% $\pm$ 6.43%	0.758 $\pm$ 0.428

Analysis:

- **MC is sensitive to** β: $\beta = 1.0$ outperforms $\beta = 0.7$ by +8.4%, suggesting MC benefits from full-strength distance guidance.
- **SARSA is robust to** β: Only 0.1% difference between values, indicating SARSA's online learning already provides sufficient guidance.
- These results reinforce that hyperparameter effects are algorithm-dependent.

5 Discussion

5.1 Which Shaping Helped Most in Early Learning?

Safety-based shaping provided the fastest early learning for both algorithms. SARSA reached $\sim$50% throughput by episode 1000 and $\sim$65% by episode 3000, demonstrating rapid initial learning. MC showed slower but consistent improvement with safety shaping pulling ahead after $\sim$2000 episodes.

The immediate effectiveness stems from safety shaping's focus on avoiding holes, which provides actionable guidance that helps agents discover viable policies quickly in the stochastic environment.

5.2 Which Shaping Preserved Final Optimal Behaviour?

Safety-based shaping achieved the highest final throughput for both algorithms (71.46% for SARSA, 47.71% for MC), suggesting it not only accelerates early learning but also leads to better final policies.

Potential-based shaping, despite being theoretically policy-invariant, underperformed:

- MC potential (43.60%) underperformed MC baseline (44.30%)

- SARSA potential (59.32%) was good but not optimal

This demonstrates that while potential-based shaping preserves optimal policies *in theory*, practical performance depends on potential function quality, algorithm characteristics, and sufficient exploration.

5.3 Why Potential-Based Shaping Behaves Differently

Potential-based shaping is theoretically policy-invariant: $Q^*(s, a) = Q^*_{\text{shaped}}(s, a) - \Phi(s)$. However, our results show practical limitations:

For Monte Carlo: The dense feedback from distance-based shaping may interfere with MC's episodic credit assignment. MC updates all visited states after episode completion; conflicting signals throughout the episode may add noise to value estimates.

For SARSA: Online updates better leverage dense feedback, but safety-based shaping still outperforms significantly (71.46% vs 59.32%).

Key Insight: Policy invariance guarantees that the optimal policy is preserved, but does *not* guarantee faster learning or that the algorithm will efficiently discover that optimal policy. In stochastic environments with limited episodes, practical learning dynamics matter more than theoretical guarantees.

5.4 Surprising Results and Failure Cases

1. Step-Cost Shaping Failure Most surprising: step-cost dramatically hurt both algorithms (MC: 15.47%, SARSA: 30.10%). In stochastic FrozenLake where slippage causes many steps, the accumulating negative reward: (a) makes all states appear negative, reducing contrast, (b) encourages premature termination, (c) creates difficult-to-optimize value functions.

2. SARSA Baseline Outperforms All MC Methods SARSA baseline (50.29%) exceeded even the best MC method (47.71% with safety shaping). This demonstrates: (a) on-policy updates learning directly from stochastic exploration, (b) online updates enabling faster adaptation, (c) bootstrapping better handling 70% success rate dynamics.

3. Exploration Bonus Underperformance for MC Exploration bonus achieved only 29.18% for MC despite intuitive appeal. Possible reasons: MC already explores extensively through ε-greedy episode generation; novelty bonus may interfere with value estimation by making frequently-visited goal-adjacent states less attractive.

5.5 Exploration vs Exploitation in This Project

We encountered the exploration-exploitation tradeoff in multiple contexts:

1. Epsilon Selection: Our sensitivity study revealed $\varepsilon = 0.05$ substantially outperformed $\varepsilon = 0.2$ and $\varepsilon = 0.5$, demonstrating the classic dilemma:

- Too much exploration ($\varepsilon = 0.5$): Throughput drops to 17.98%
- Optimal balance ($\varepsilon = 0.05$): 48.79% throughput
- The environment's 70% success rate provides "free exploration" through slippage

2. Reward Shaping as Exploration Guidance:

- Exploration bonus attempted explicit exploration encouragement but hurt MC (29.18%)
- Safety-based shaping provided *directed exploration* toward promising regions—more sample-efficient than random exploration

3. Ideas to Solve the Tradeoff:

1. **Decaying epsilon:** $\varepsilon(t) = \max(\varepsilon_{\min}, \varepsilon_0 \cdot \text{decay}^t)$—start high for discovery, decay for exploitation.
2. **Optimistic initialization:** Initialize $Q(s, a) > 0$ to encourage exploration of unvisited states.
3. **UCB action selection:** $a = \arg\max_a [Q(s, a) + c\sqrt{\ln(n)/n(s, a)}]$—systematically explores less-visited actions.
4. **Reward shaping for directed exploration:** Safety-based shaping solved exploration by guiding toward promising regions.

What Actually Worked: Low ε (0.05) + environment stochasticity + safety-based guidance. In stochastic environments, *less explicit exploration + smart guidance* outperforms *heavy exploration strategies*.

6 Summary of Findings

1. **SARSA + Safety-Based Shaping is optimal:** Achieved 71.46% throughput, outperforming all other configurations.
2. **Reward shaping effectiveness is algorithm-dependent:** Safety shaping provided +42.1% improvement for SARSA but only +7.7% for MC (relative to respective baselines).
3. **SARSA dominates Monte Carlo:** Even SARSA baseline (50.29%) exceeded best MC variant (47.71%), demonstrating on-policy TD learning's superiority in stochastic environments.
4. **Hyperparameters critical:** Optimal values $\varepsilon = 0.05$, $\alpha = 0.2$ for SARSA+Safety. Low epsilon outperformed high epsilon by 171%.
5. **Stochastic environments require different strategies:** Step-cost shaping backfires; low exploration works when environment provides inherent randomness.

7 Conclusion

This work demonstrates that reward shaping can significantly accelerate reinforcement learning when designed with domain knowledge. Our key contributions include:

1. **Novel safety-based shaping:** A potential-based method encouraging hole avoidance, achieving best performance across both algorithms.
2. **Comprehensive comparison:** Systematic evaluation of five shaping methods across two fundamental RL algorithms.
3. **Hyperparameter insights:** Sensitivity studies revealing optimal $\varepsilon = 0.05$ and $\alpha = 0.2$ for SARSA with safety shaping.
4. **Algorithm insights:** SARSA dramatically outperforms MC in stochastic environments (71.46% vs 47.71%), particularly with appropriate reward shaping.

The exploration-exploitation tradeoff remains central to RL success: neither pure exploration nor pure exploitation succeeds. Reward shaping, when designed with domain knowledge (e.g., safety in FrozenLake), can guide this balance toward faster, more reliable learning.

Quick Start Guide

Installation

Requirements: Python 3.8+, 8GB RAM

```
# Navigate to project directory
cd reward-shaping-analysis

# Create and activate virtual environment
python -m venv venv
source venv/bin/activate  # Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt
```

Note: Ensure you are in the `reward-shaping-analysis` directory before running any experiments.

Core dependencies: numpy, gymnasium, matplotlib, seaborn, pandas, tqdm, joblib, pygame, pillow

Project Structure

```
reward-shaping-analysis/
|-- algorithms/          # Monte Carlo and SARSA implementations
|-- env/                 # FrozenLake environment and reward shaping wrappers
|-- experiments/         # Parallel and sequential experiment runners
|-- utils/               # Plotting, visualization, and rendering utilities
|-- config.py            # Central configuration (all hyperparameters)
|-- compare_methods.py              # Main comparison (sequential)
|-- compare_methods_parallel.py     # Main comparison (parallel, faster)
|-- hyperparameters_sweep.py        # Epsilon/Alpha sensitivity analysis
|-- beta_comparison.py              # Beta parameter sensitivity
|-- test_agent_rendering.py         # Policy evolution demonstration
|-- results/             # Generated plots and visualizations
```

Running Experiments

Important: All commands must be run from the `reward-shaping-analysis` directory.

Main Experiments (in order):

```
# 1. Main comparison: All methods (MC vs SARSA, all shaping types)
python compare_methods_parallel.py    # ~15-30 min (parallel, recommended)
# OR
python compare_methods.py             # ~60-90 min (sequential, for debugging)

# 2. Hyperparameter sensitivity: Epsilon and Alpha
python hyperparameters_sweep.py       # ~10-15 min

# 3. Beta sensitivity: Potential-based shaping parameter
python beta_comparison.py             # ~5-10 min

# 4. Policy evolution visualization: Learning demonstration
python test_agent_rendering.py        # ~5-10 min
```

Note: Use `compare_methods_parallel.py` for faster execution (automatically uses all CPU cores). Use `compare_methods.py` for sequential execution (easier debugging, lower memory usage).

Configuration

All experimental parameters are configurable in `config.py`:

- `RANDOM_SEED = 242` – For reproducibility (same environment map across runs)
- `TEST_MODE = False` – Set to `True` for quick testing (1000 episodes, 5 runs)
- `FINAL_EPISODES = 10000` – Training episodes per run (full experiments)
- `FINAL_RUNS = 20` – Independent runs for statistical validity
- `GRID_ROWS = 6, GRID_COLS = 6` – Environment size (6×6 grid)
- `HOLE_DENSITY = 0.15` – Obstacle density (15%)
- `IS_SLIPPERY = True` – Stochastic dynamics
- `SUCCESS_RATE = 0.7` – Action success probability (70%)

Optimized hyperparameters (epsilon, alpha, beta) are pre-configured per algorithm and shaping method. You can modify these in `config.py` under `MC_HYPERPARAMS` and `SARSA_HYPERPARAMS`.

Quick Testing: Set `TEST_MODE = True` in `config.py` for faster experiments with reduced episodes and runs.

Output

All results are saved to `results/` directory:

- Throughput and returns comparison plots (with 95% confidence intervals)
- Policy visualizations with action arrows and Q-values
- Statistical summaries printed to console
- Animated GIF demonstrations (`results/agent_demos/`)

References

[1] Ng, A. Y., Harada, D., & Russell, S. (1999). *Policy invariance under reward transformations: Theory and application to reward shaping.* ICML, Vol. 99, pp. 278-287.

[2] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.