Journal of Machine Learning Research Project 4 (2025) 1-8 Submitted 12/5; Published 12/5

Learning to Drive: Reinforcement Learning Approaches to the Racetrack Problem

Taron Moe-Stull TARONMOE@ICLOUD.COM
School of Computing
Montana State University
Barnard Hall, 357, Bozeman, MT 59718, USA

Editor: Text Editor (Overleaf)

Abstract

The racetrack problem is a common benchmark for reinforcement learning where an agent must drive from a start region to a finish region as efficiently as possible under constraints. In this work, three approaches are implemented and compared for the racetrack domain: model-based Value Iteration, Q-learning, and SARSA. The environment is defined as an ASCII grid, where position is discrete and velocity is two-dimensional and bounded. Actions are accelerations in each axis with values in *-1, 0, 1*. Stochasticity is introduced via a 20% probability that an attempted acceleration fails, leaving velocity unchanged. Two crash handling strategies are evaluated: resetting the car to the nearest valid track cell *(NRST)*, or returning to the original start line with zero velocity *(STRT)*. The hypothesis is that Value Iteration will produce strong policies given full model knowledge; Q-learning will tend to learn shorter, more aggressive policies than SARSA (especially under *NRST*); and SARSA will converge toward safer policies due to its on-policy updates. All three methods are evaluated on the provided tracks and compared by steps-to-finish, learned path structure, and stability under crash and stochastic dynamics.

Keywords: Reinforcement Learning, Value Iteration, Q learning, SARSA, Racetrack Problem

1 Problem Statement

The goal is to find efficient solutions to a racetrack control problem using three algorithms: Value Iteration, Q-learning, and state-action-state-reward-action (SARSA). The environment consists of a car (the agent) moving on a racetrack with a start region, barriers, and a finish region. The agent has partial direct control over acceleration and indirect control over velocity and position, in both the x and y directions. The objective is to reach the finish with the minimum number of acceleration-action steps. Collisions apply a penalty: velocity is set to 0, and the agent either moves to the previous valid position or restarts the track entirely, depending on a user-selected rule.

All algorithms are evaluated on the same tracks and crash handling rules. Performance is compared using step count to finish, consistency of successful paths, and crash/navigation behavior. Because the environment includes stochastic acceleration failure and crash penalties, learning behavior is expected to be shaped significantly by punishment and uncertainty. (CSCI 446 Course Materials, 2025)

Hypothesis

It is expected that the agent will generally seek more risk-averse paths when crashes force a full restart rather than only zeroing velocity. Although the optimal policy is guaranteed only via Value Iteration, the crash penalty should also influence Q-learning and SARSA behavior. Value Iteration is expected to produce the strongest and most consistent policies due to full model knowledge. For model-free methods, Q-learning is expected to converge faster and produce more aggressive policies due to off-policy updates, while SARSA is expected to behave more cautiously because updates follow actions actually taken during exploration. Finally, the 20% acceleration failure probability is expected to introduce variability in learned policies, especially for Q-learning and SARSA. (Russell and Norvig, 2020)

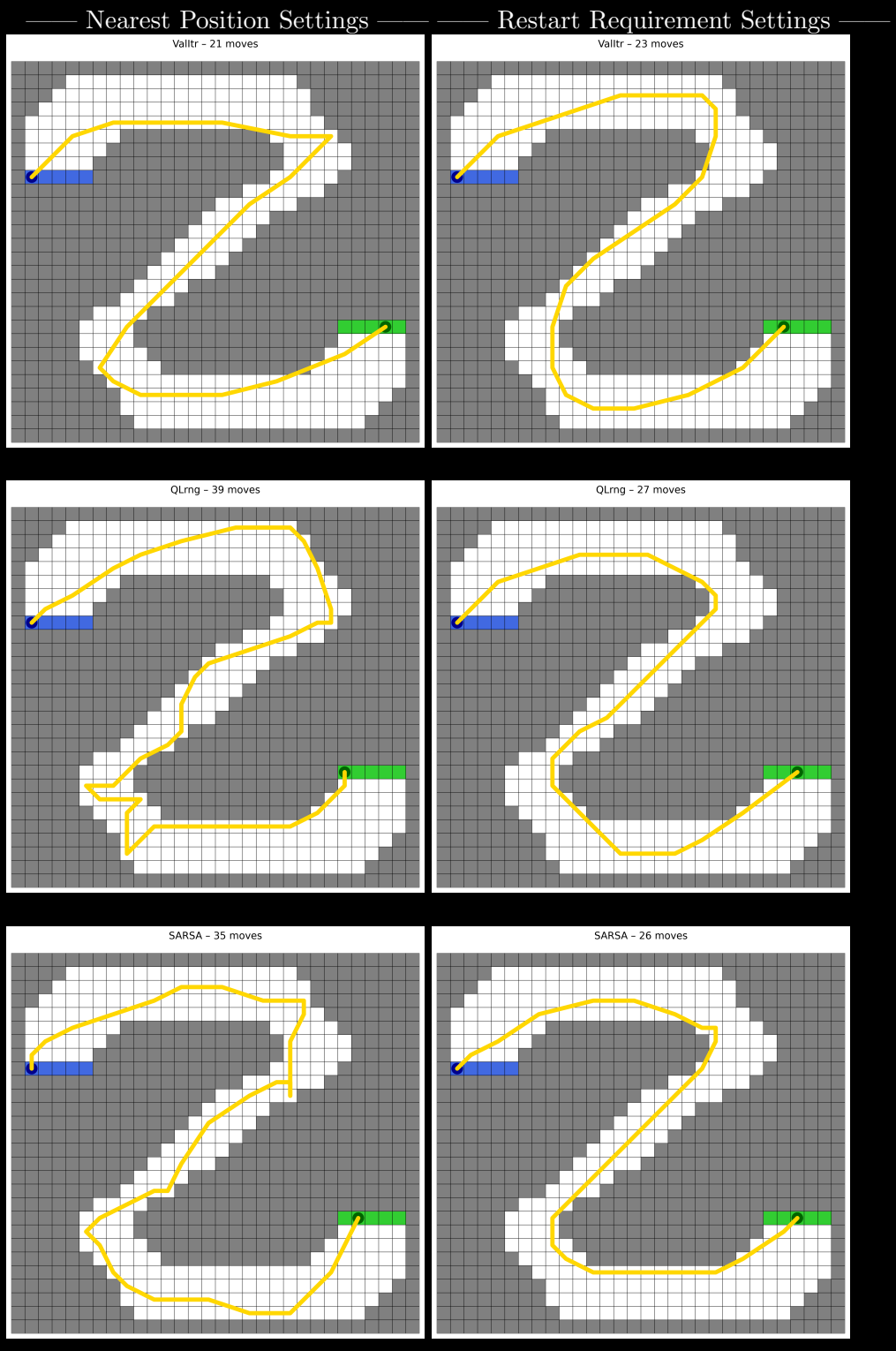
2 Experimental Approach

To evaluate and compare the three approaches, each method was run on the same tracks with identical starting parameters. Value Iteration serves as a baseline due to complete model knowledge. Q-learning and SARSA are trained through repeated interaction with the environment, learning via trial and error. For both learning-based methods, multiple training runs are required before evaluating the final policy. Final policies are evaluated separately from training to reduce exploration bias.

All methods operate over a state space defined by the agent's position and velocity. Actions are acceleration updates represented by {-1, 0, 1} per axis. The 20% acceleration failure rate introduces meaningful stochasticity into transitions. Q-learning and SARSA use greedy exploration, with learning rate and discount factor chosen to yield stable convergence without exhaustive tuning. Crash handling is treated as an environment rule rather than additional reward shaping: crashes reset the agent (either to nearest valid cell or to start) and naturally affect learning through experience. Using this setup, differences between algorithms are examined and summarized in the results. (Sutton and Barto, 2015)

3 Results

In line with the initial hypothesis, the STRT requirement did consistently result in finding solutions that mitigate risk against collision. However, it also resulted in substantially better solutions (requiring fewer moves) in both Q-learning and SARSA. Images of the 2-track solutions convey this trend relatively clearly:



Figures 0 a-f: 2-Track Found Pathways From All 3 Algorithms and Both Rule-sets

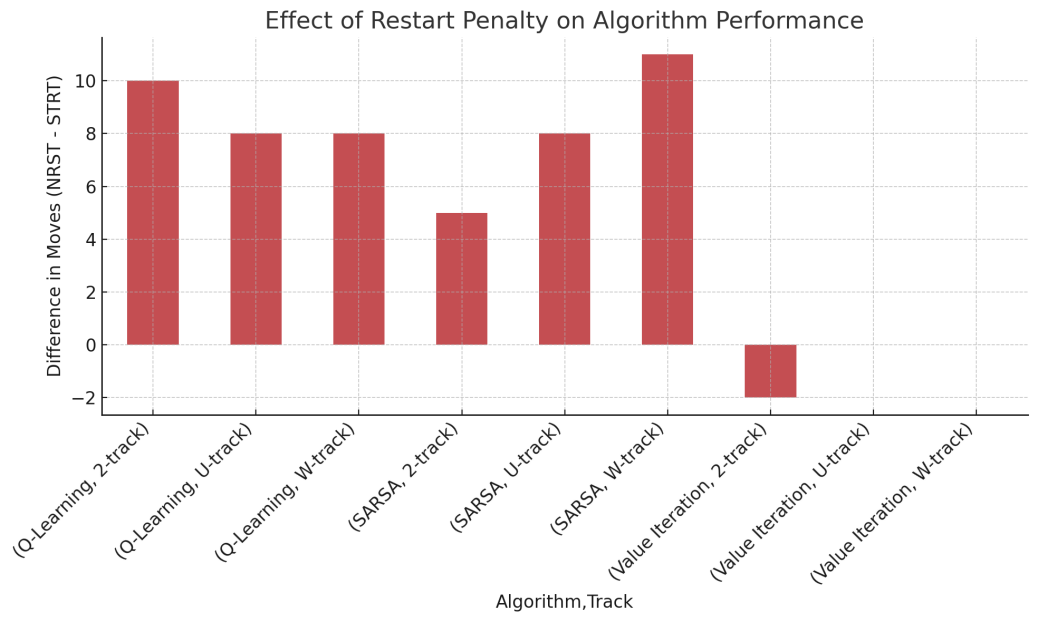
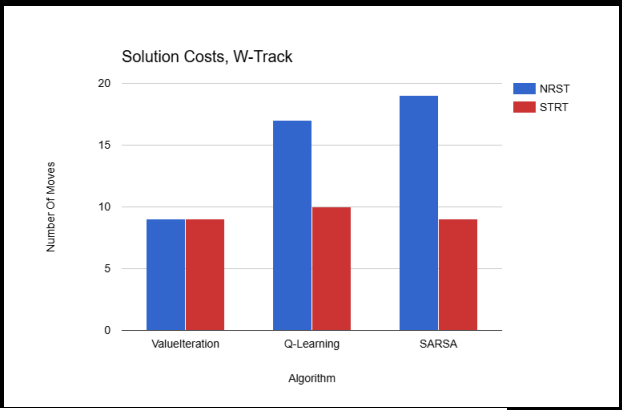
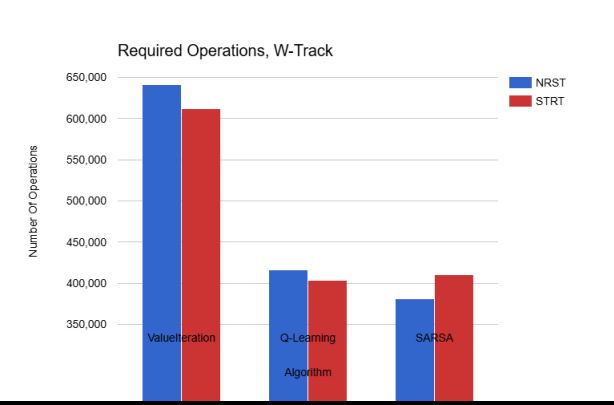
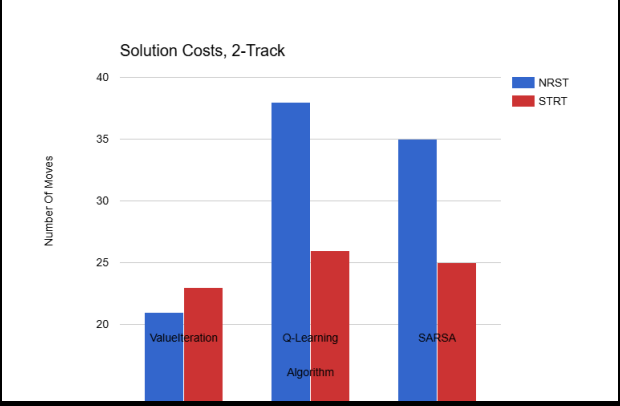


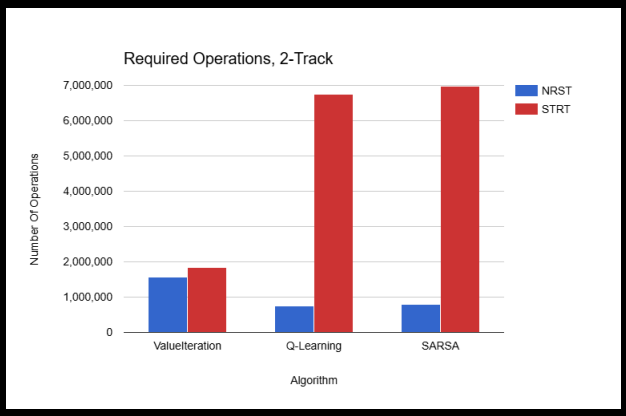
Figure 1: Effect of crash handling strategy on performance.

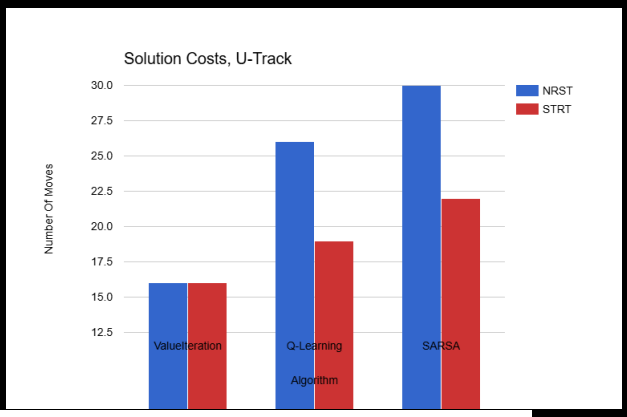
As previously stated and confirmed via observation, the STRT requirement generally results in cleaner found solutions in Q-learning and SARSA. The reason for this likely relates both to algorithm behavior and implementation details. The stricter rule-set typically requires longer training to find reliable solutions, producing more total moves per training run early on and effectively increasing exploration of action trees. In the implementation, training requires the agent to either find a solution or fail (exceeding 500 moves results in failure and the run resets) a fixed number of times before training concludes. Under STRT, early runs tend to reset more often, producing many more moves per run and substantially greater exploration. This is reflected in the number of operations required to complete training under each crash rule:

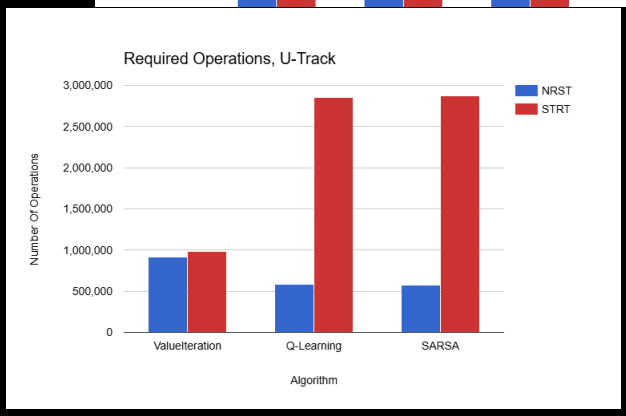

Solution Costs, W-Track
NRST
STRT
Number Of Moves
20
15
10
5
0
ValueIteration
Q-Learning
SARSA
Algorithm


Required Operations, W-Track
NRST
STRT
Number Of Operations
650,000
600,000
550,000
500,000
450,000
400,000
350,000
ValueIteration
Q-Learning
SARSA
Algorithm


Solution Costs, 2-Track
NRST
STRT
Number Of Moves
40
35
30
25
20
ValueIteration
Q-Learning
SARSA
Algorithm







Figures 2 a-f: Comparison Between Number of Moves in Found Solution and Runtime Operations Required in Computation

In the short W-track, there is no immediately discernible difference between runtime operations and rule selection. However, the more complex 2-track shows a clear difference, with an order-of-magnitude increase in computational demand (7×10^5 compared to 7×10^6). This supports the conclusion that a stricter rule-set demands more exploration. As a note: Value Iteration performed relatively consistently across rule-sets both in runtime operations and moves required.

4 Discussion

The hypothesis is generally supported by observed behavior across all three algorithms, but the results also highlight important nuances. On the W-track, the best solution found by Value Iteration is identical for both crash handling modes. For Q-learning and SARSA, the harsher STRT rule-set often produces better solutions at similar computational cost, likely because it more strongly punishes wasted movement and stabilizes learning direction.
On the 2-track, Value Iteration produces similar policies under both rule-sets, with STRT producing a slightly more conservative path and requiring more moves. Q-learning and SARSA find better and safer pathways under STRT, but at the cost of substantially increased computational demand. The U-track shows similar behavior but with less extreme scaling effects.
To address runtime discrepancies between rule-sets for Q-learning and SARSA, several adjustments could reduce wasted early exploration. One option is tuning existing parameters based on rule-set (training rate, move cap before reset, number of training runs). Another option is modifying scoring to incorporate progress toward the finish, such as initializing with a positional heuristic and softening penalties for moves that improve proximity to the finish. This could reduce the number of purely random early runs required before the agent finds a trajectory worth reinforcing.
This approach introduces trade-offs on more maze-like tracks where naive distance heuristics may bias toward suboptimal paths. A practical improvement would be to gradually reduce reliance on the heuristic as training stabilizes, allowing the learned value function to dominate over time.
Overall, the trends reinforce the expected behavior of deterministic vs. non-deterministic approaches. Deterministic methods provide optimal solutions when the state space is tractable, while model-free reinforcement learning methods can find effective solutions with less explicit computation, but may require careful rule and reward design to avoid worst-case exploration behavior.

5 Summary

This work implements and compares reinforcement learning and dynamic programming approaches to a racetrack control problem under two crash handling modes: Value Iteration, Q-learning, and SARSA. Using Value Iteration as a baseline, performance and behavior were analyzed in terms of solution quality, computational demand, and the effect of crash rules on learning.
The observed behavior supports the core hypothesis: STRT encourages safer, more stable learned policies in Q-learning and SARSA, often producing better solutions but requiring substantially more training computation on complex tracks. Value Iteration remains consistent across rule-sets, producing strong policies given complete model knowledge.
In conclusion, the experiments demonstrate how crash handling rules and environment stochasticity strongly shape learned strategies, and how algorithm choice impacts both reliability and computational cost.

References

Montana State University CSCI 446 Course Materials. Project 4 reinforcement learning and the racetrack problem. https://montana.instructure.com/courses/19506/files/, 2025. Accessed: December 2025.

Overleaf. Overleaf, online LaTeX editor. https://www.overleaf.com. Accessed December 2025.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, Hoboken, NJ, 4th edition, 2020. Accessed: December 2025.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2015. Accessed December 2025.