1. Introduction

Reinforcement Learning (RL) is an area of Machine Learning that uses experiences to determine the next optimal decision. Typical the reinforcement learning process can be broken down into a few key steps: Observe the environment, decide how to act based on a policy, act based on that policy, be rewarded or penalized for that action, update the policy, iterate until the best policy is produced. Two major type of RL algorithms are the model-based and model-free algorithms. In the model-based algorithms a the function used to issue the reward is used to determine the best policy.

The objective of this paper is to explore three different reinforcement learning algorithms, Value Iteration, Quality Learning or Q-Learning and the State Action Reward State Action (SARSA) algorithm. These algorithms will be explored and implemented through the racetrack problem. In the racetrack problem, the objective is to get a car from the starting line to the finish line in as little time as possible. The racetrack will be described as a two dimensional Cartesian plane, where the location of the car on the racetrack can be described as a point pair (x, y). The active control exerted over the car will be acceleration a_x and a_y. The range of values that can be assigned to a_x and a_y are -1,0,1. The velocity of the car will be capped to values between -5 and 5, any acceleration values that cause the velocity to go beyond these bounds will be ignored. Additional the probability of the acceleration control being applied on each time step is 80%, meaning there is a 20% chance the accelerate control is issued no acceleration updates will occur. The experiments for these algorithms will occur on three uniquely shaped racetracks, the R-Shaped, O-Shaped and L-Shaped tracks.

An additional constraint that will be applied to this problem, is the consequence of a crash. A crash is defined as the car intersecting with the walls of the racetrack, in one variation if a crash happens, the car will be placed back to a location on the track nearest to the crash site with a zero velocity component. In the second, harsher variation, if a crash happens, the car will be placed back at the start of the racetrack with a zero velocity component. These variations will only be applied to the R-Shaped race track, the O-Shaped and L-Shaped tracks will not implement the harsher variation of the crash, instead if a crash occurs on these tracks the car will be placed at a location nearest to the crash site.

The hypothesis presented in this paper predicts that the SARSA algorithm will perform better compared to the Q-Learning or Value Iteration algorithms. The reason for this hypothesis is that the SARSA algorithm is an on-policy algorithm as it updates the policy based on the current choices of the policy that is being implemented. The Q-Learning algorithm, however, does not update the same policy used to explore the world. The SARSA algorithm seems better suited for the racetrack problem, since the actions taken by the model to drive the car the fastest around the racetrack should update the policy as the world is better understood, if the model understand there is a wall or turn coming up, it will have the environment baked into the policy, where as the Q-Learning or Value Iteration algorithms, do not account for updates to the environment each time the policy is updated.

Section 1 has provided the introduction, problem statement and hypothesis in regards to three major learning algorithms that will be explored. Section 2 will provide an in-depth explanation of the algorithms. Section 4 will present the results obtained by the different algorithms. Section 5 will discuss the results that were obtained and compare them to the hypothesis that was outlined in the introduction. This report will conclude in Section 6 with a discussion of lessons learned and areas of possible future work.

2. Algorithms and Experimental Methods

Value Iteration

Value Iteration is one type of Reinforcement Learning. In Value Iteration the Value $V(s)$ represents how good a state the agent is currently in, typically the higher the value the best position the agent is in. The value function is dependent on the policy that the agent used to get into a given state. For a given set of states (s) that belong to the set State Space (S) there is a maximum value where the value of that state is highest among all states in the State Space. This value has an optimal policy associated with it. The Q Function is a representation of the total expected reward received by an agent in state s and picking an action (a) out of the whole Action Space (A). In Value Iteration the agent is not involved in the process of learning, instead the optimal state is determined by iterating and improve the estimate of $V(s)$ until the different between iterations on $V(s)$ is within a certain value, at this point the algorithm is considered converged and the car can be run using the policy obtained during the training. The general algorithmic steps for the Value Iteration Algorithm are shown below:

Value Iteration

```
Train the Car:
    For all states ($s$) in the overall State Space (S):
        Set $V(s)$ to 0
    While $max_{sofS}(V_t(s) - V_{t-1}(s)) < \epsilon$:
        Number of Iterations = Number of Iterations + 1
        For all states ($s$) in the overall State Space (S):
            For all actions ($a$) in the overall Action Space (A):
                $ExpectedValue = 0.8*(V(AccApplied))+0.2*(V(AccNotApplied))$
                $Q_t(s,a) = Reward + \eta * (ExpectedValue) * V_{t-1}(s')$
            $\pi_t(s) = max_{aofA}(Q_t(s,a))$
            $V_t(s) = Q_t(s, \pi_t(s))$

Test the Car:
    While the Car has not reached the finish line:
        Get Best Policy (Acceleration) based on Current State
        Apply Acceleration to Car and Update Velocity and Position
        Check Car if the Car has hit a wall
        If Wall Hit has Occurred:
            If the Crash is Type 1:
                Place the Car back to the near position of the crash
                and set velocity to zero.
            If the Crash is Type 2:
                Place the Car back to at starting position and set velocity to zero.
```

Q-Learning
INSERT GENERAL DESCRIPTION.
The general algorithmic steps for the Q-Learning Algorithm are shown below:

Q-Learning

```
Train the Car:
    Initialize all $Q(s,a)$ to -1000.
    Set a number of Training Iterations (Episodes)
    For each Training Iteration in the total number of Training Iterations:
        Initialize s by setting the car to the starting position.
        While the Car has not has not reached the finish line or the maximum number of moves has not been reached
            Chose an Action (a) from Q(s,a) based the current state using an $\epsilon$ Greedy Approach
            $\epsilon$ Greedy Approach:
                Chose action a based on the best policy Or
                Randomly pick the action from the Action Space
            Take action (a) observe the reward and update the cars state.
            Update $Q(s,a)$:
                $Q(s,a) = Q(s,a) + \eta * (reward + \gamma * max_{a'} Q(s',a') - Q(s,a))$

Test the Car:
    While the Car has not reached the finish line:
        Get Best Policy (Acceleration) based on Current State
        Apply Acceleration to Car and Update Velocity and Position
        Check Car if the Car has hit a wall
        If Wall Hit has Occurred:
            If the Crash is Type 1:
                Place the Car back to the near position of the crash
                and set velocity to zero.
            If the Crash is Type 2:
                Place the Car back to at starting position and set velocity to zero.
```

SARSA

State-Action-Reward-State-Action (SARSA) is the final type of Reinforcement Learning algorithm that will be discussed in this paper. SARSA is an on-policy learning algorithm, it is on-policy because it chooses the action for each state while the learning is occurring and does so by following the policy. This is the biggest different between Q-Learning and SARSA, SARSA learns and updates while following the policy, Q-Learning does not. Stated differently

SARSA

```
Train the Car:
    Initialize all $Q(s,a)$ to -1000.
    Unless state position is a finish spot, then sent $Q(finish,a)$ to zero.
    Set a number of Training Iterations (Episodes)
    For each Training Iteration in the total number of Training Iterations:
        Initialize s by setting the car to the starting position.
        Chose an Action (a) from Q(s,a) based the current state using an $\epsilon$ Greedy Approach
        While the Car has not has not reached the finish line or the maximum number of moves has not been reached
            Take action (a) observe the reward and update the cars state to s'.
            Chose a second action (a') from state s' using using an $\epsilon$ Greedy Approach
            $Q(s,a) = Q(s,a) + \eta * (reward + \gamma * Q(s',a') - Q(s,a))$

Test the Car:
    While the Car has not reached the finish line:
        Get Best Policy (Acceleration) based on Current State
        Apply Acceleration to Car and Update Velocity and Position
        Check Car if the Car has hit a wall
        If Wall Hit has Occurred:
            If the Crash is Type 1:
                Place the Car back to the near position of the crash
                and set velocity to zero.
            If the Crash is Type 2:
                Place the Car back to at starting position and set velocity to zero.
```

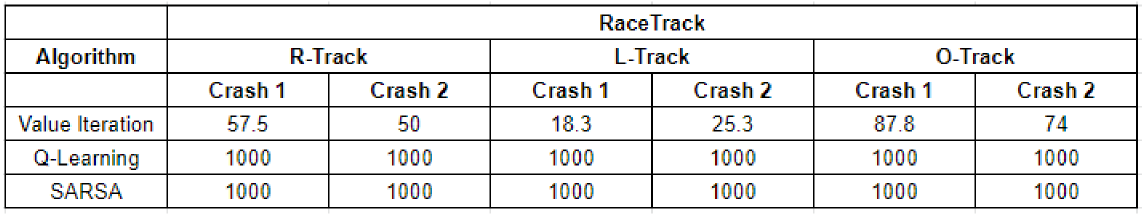
3. Data Sets

For this paper there were no explicate data-sets provided. Instead the data provided was in the form of the shape of the Racetracks. There were three uniquely shaped racetracks, the R-Shaped, O-Shaped and L-Shaped tracks provided for this problem. Each taking on the shape of the letter in their respective names.

4. Results

Tables 1-3 display the learning for each of the algorithms across both types of crash variations. For brevity of this report the processing of tuning the various tune parameters (Learning Rate, Discount Factor, ect) have been omitted.
Table 1 shows the Learning Curve for Value Iteration, the optimal values used to obtain these results were: $\epsilon = 0.000001$, Discount = 0.7, Limit On Iterations = 1000. Note the Limit on Iterations was set just in case the algorithm values different did not converge in a reasonable amount of time, in all runs obtained for this report the Limit of Iterations was never hit.
Table 2 shows the Learning Curve for Q-Learning, the optima values used to obtain these results were: $\epsilon = 0.02$, Discount = 0.9, Learning Rate = 0.6, Maximum Number of Moves = 1000, Number of Training Runs = 10,000. The Maximum Number of Moves was put in place to limit how long the car was allowed to explore the environment during each training run.
Table 3 shows the Learning Curve for SARSA, the optima values used to obtain these results were: $\epsilon = 0.02$, Discount = 0.9, Learning Rate = 0.6, Maximum Number of Moves = 1000, Number of Training Runs = 10,000. The Maximum Number of Moves was put in place to limit how long the car was allowed to explore the environment during each training run.
Table 4 shows the results of the Test Runs for each of the algorithms and the crash variations, to get a statstical analysis the car was run using the policy obtained during training 10 times. The number of moves required to complete the track was then averaged to provide the numbers in Table 4. Note that Q-Learning and SARSA both had a hard cap of 1,000 moves that they were allowed to utilized while the test ran.

Table 1: Value Iteration, Q-Learning and SARSA Average Moves Taken Across 10 Cars



	RaceTrack					
Algorithm	**R-Track**		**L-Track**		**O-Track**	
	Crash 1	**Crash 2**	**Crash 1**	**Crash 2**	**Crash 1**	**Crash 2**
Value Iteration	57.5	50	18.3	25.3	87.8	74
Q-Learning	1000	1000	1000	1000	1000	1000
SARSA	1000	1000	1000	1000	1000	1000

5. Discussion

The hypothesis presented in this report was that the SARSA algorithm will perform better compared to the Q-Learning or Value Iteration algorithms.

6. Conclusion

Overall, the results could not provide meaningful data towards the hypothesis that SARSA would out perform Q-Learning or Value Iteration. The only comparisons that can be drawing is by examining the data obtained for the learning curves during the training runs. In general, the difference between the two crash variations did cause the cars to learn different behavior. In Q-Learning and SARSA, when the car was placed back at a nearest location to the crash, the car was more often able to finish the racetrack during the training iterations. During the course of these experiments it was noticed that the seed values for Epsilon, Q and Learning Rate also produced drastically different results. A lesson learned from this project is to have a fun understanding of what part of the algorithm these parameters impact prior to running the algorithm, as a bad seed value can lead to lost time in trying to debug an implementation issue that simply isn't there, instead poorly chosen values are making it appear as if the algorithm is not running correctly.

A suggested area of future work, aside from fixing the implementation issues that might have been altering the results obtained for the Q-Learning and SARSA algorithms, would be to explore the impact of varying the starting position of the car on each training iteration of the Q-Learning and SARSA algorithms. If the initialized position of the car was varied on each iteration or the initialized position was placed closure to the finish line and then worked back towards the start lining, it is possible that the Values of Q would have percolated backwards towards the start, providing the car with the motivation (value insecntive) to move towards the finish line. This could also be the root cause of the implementation issue that was impacting the Q-Learning and SARSA algorithms.

7. References

1. Alpaydin, E. (2004). Introduction to machine learning (Oip). Mit Press.

Table 2: Value Iteration Learning Curves All Race Tracks

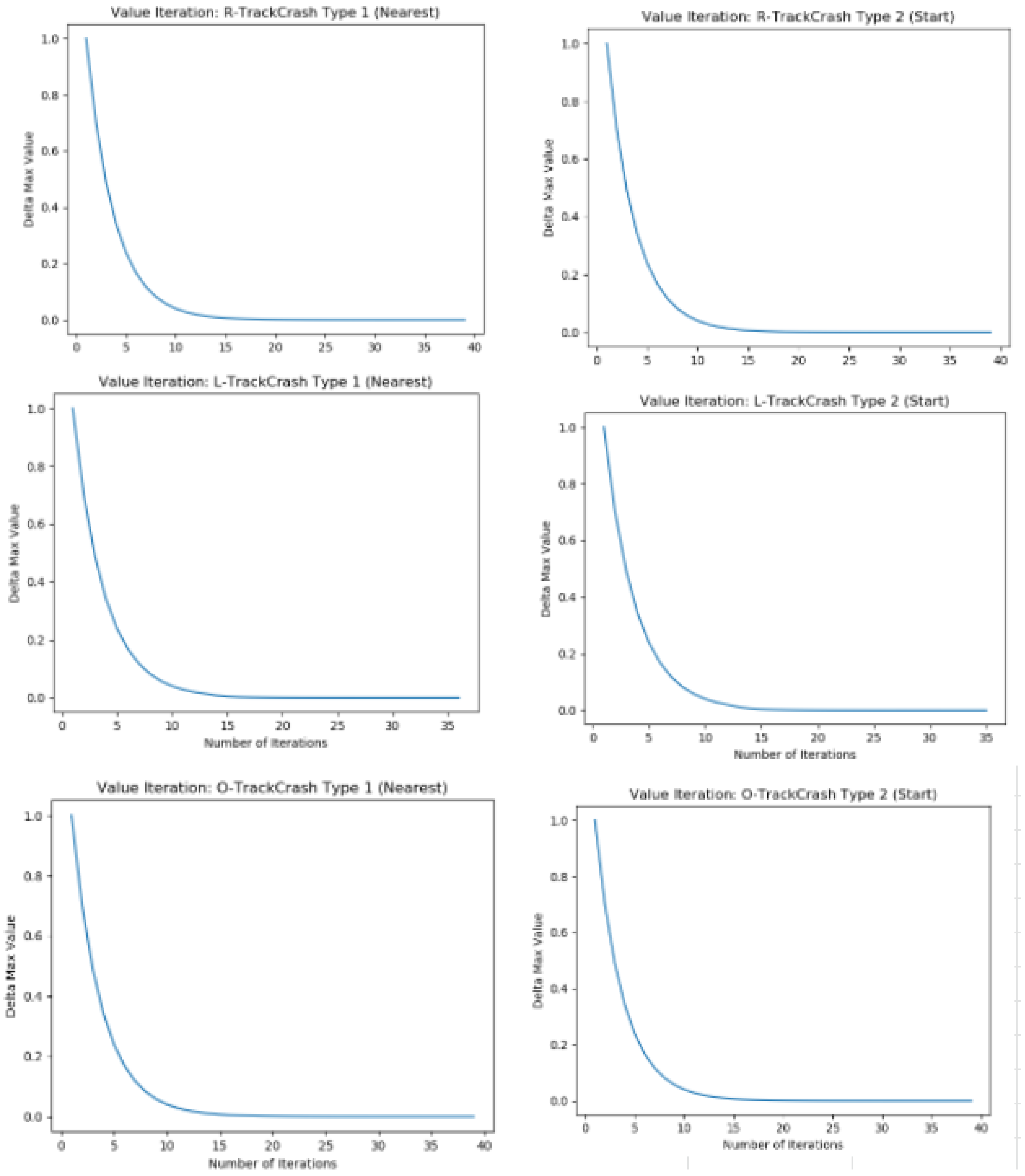


Table 3: Q Learning Learning Curves All Race Tracks

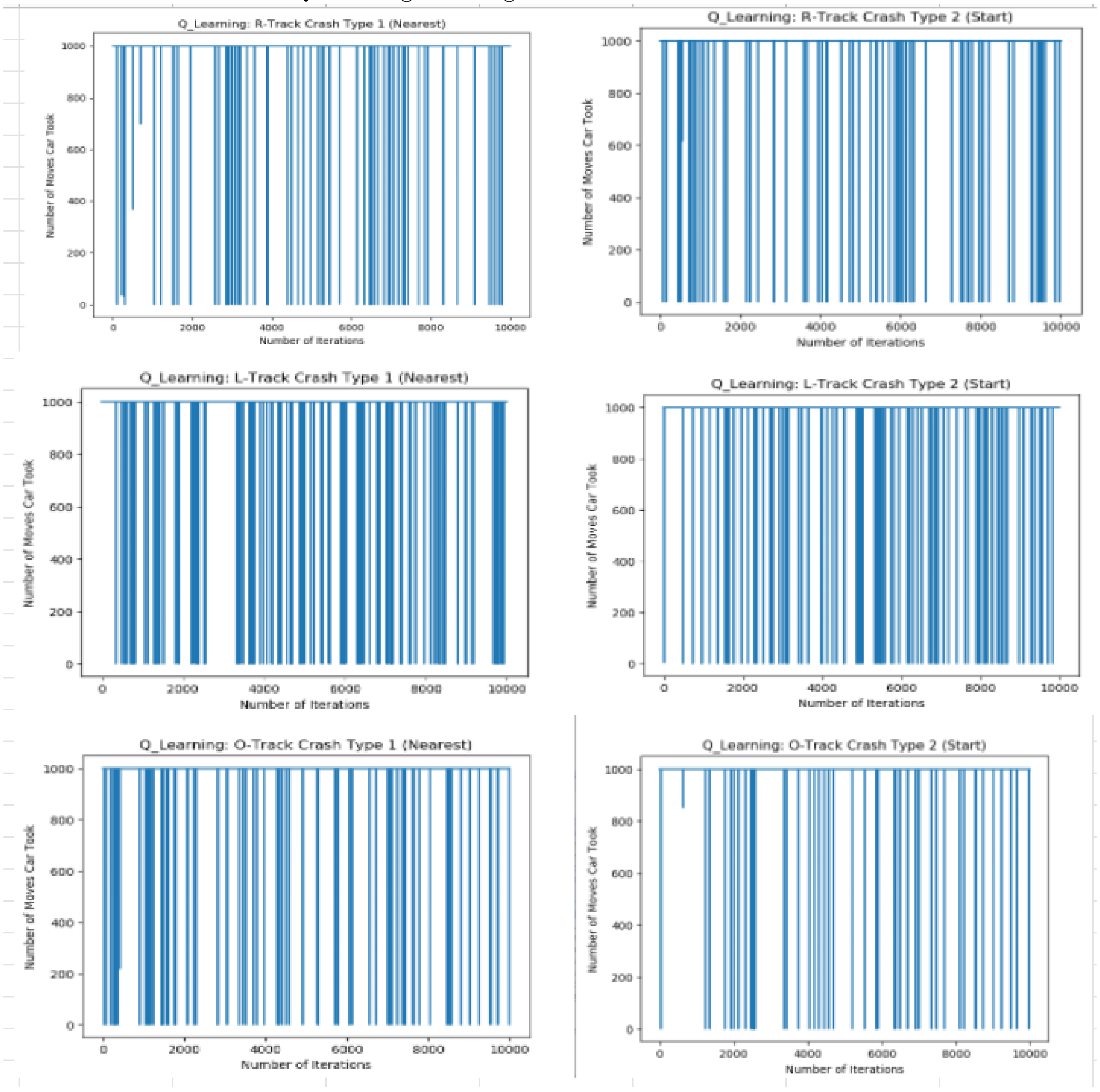
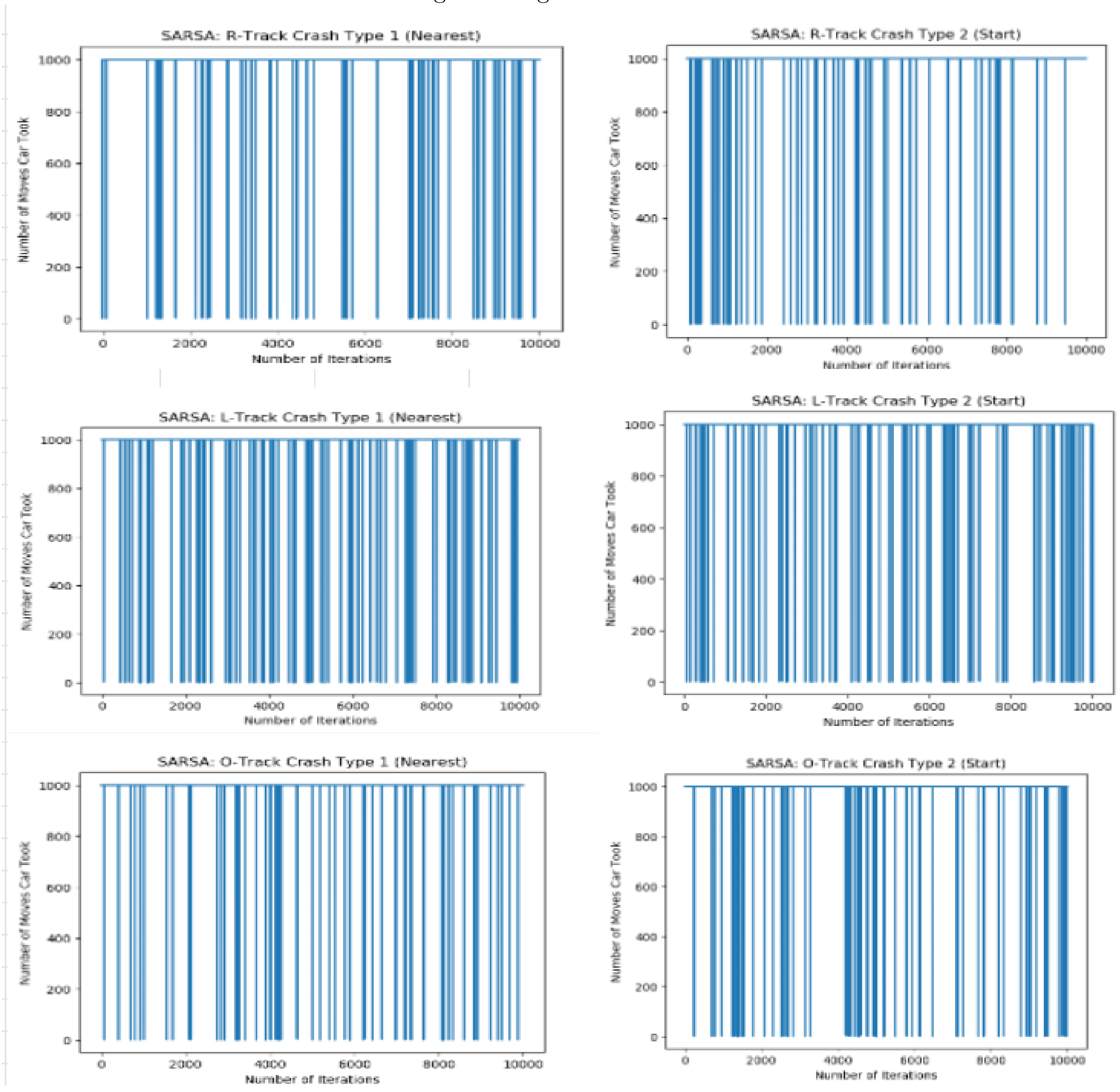


Table 4: SARSA Learning Learning Curves All Race Tracks


SARSA: R-Track Crash Type 1 (Nearest)
SARSA: R-Track Crash Type 2 (Start)
SARSA: L-Track Crash Type 1 (Nearest)
SARSA: L-Track Crash Type 2 (Start)
SARSA: O-Track Crash Type 1 (Nearest)
SARSA: O-Track Crash Type 2 (Start)
Number of Moves Car Took
Number of Iterations