The Thirty-First International Florida
Artificial Intelligence Research Society Conference (FLAIRS-31)

A Reinforcement Learning Approach to Autonomous Speed Control in Robotic Systems

Nima Aghli, Marco Carvalho
Harris Institute for Assured Information, School of Computing
Florida Institute of Technology
150 W. University Blvd., Melbourne, FL - 32901
naghli2014@my.fit.edu, carvalho@fit.edu

Abstract

Model-free reinforcement learning techniques have been successfully used in diverse robotic applications. In this paper, we design and implement the Q-learning algorithm, a widely used model-free algorithm to find the optimal speed control function for a fast moving train on a fixed track. The goal is to allow for the train to learn the fastest speed profile it may use on a track, without derailment. We contrast the performance of the learning algorithm with the performance of the human controlling trying to perform the same task. In order the test the proposed algorithm, a complete hardware and software testbed has been designed and implemented, allowing for the evaluation of the learning models over a physical environment. We conclude that in simple tasks, the performance on humans in speed control is similar to the performance of the reinforcement learning algorithm, but when a more complex track is considered, the proposed reinforcement learning learning models outperforms the humans.

1 Introduction

Reinforcement learning (RL) is one of popular techniques used in field of artificial intelligence, control, and machine learning. It has proved to be a robust learning tool applied in diverse fields. Its advantageous from the theoretical and algorithmic perspective led to using RL for solving complex tasks from job-shop scheduling (Zhang and Dietterich 1995) to easy daily tasks like riding a bicycle (Randløv and Alstrøm 1998).

Generally speaking, in reinforcement learning an agent seeks to discover the state-space by trial and error, based on a reward/cost function. The goal is to learn sequences of actions associated with a given that that yield a greater accumulated reward in the long term. In reinforcement learning, the state represents the current situation of the agent in the world, while actions are mappings of state transitions and associated rewards. The result of an RL learning algorithm is a a policy, which maps the best sets of actions to given state.

1.1 Markov Decision Process

In reinforcement learning, we assume that our environment satisfy the Markov property, which states that the conditional probability of a future stat of a system based on all its prior states depends only on its current state, which means that the system is memoryless. This model is called the Markov Decision Process (MDP) (Tsitsiklis 1994) (Dayan 1992) (Howard 1960).The MDP model is described as follows, where (s) presents as given state, (a) represents a given action, and (r) the reward:

- S : a set of states $s \in S$.
- A : a set of actions $a \in A$.
- $R(s, a) \rightarrow r$: a reward function.
- $T : S \times A \rightarrow \Pi(\mathcal{S})$ where $\Pi(S)$ is the probability distribution of state transitions for a given action.

The transition function probabilistically defines the next state of an agent in the environment based on its current state and the action it takes. The Reward function defines the expected reward the agent receives based on the action it takes in the current state. This formulation requires no assumptions for a finite space of state or actions, however, in this paper, we limit our problem to a finite set of states and actions that is defined by the environment.

2 Experimentation Environment

The environment proposed for this design and experimentation of the learning algorithm is a set of connected rail tracks illustrated a railroad system (Figure 2). The environment also provides an analog for a simple flexible manufacturing system, where robotic units would be represented by the model trains.

Agents are model trains with customized hardware that move on these rails. They have WIFI connectivity, sensors used to detect failure or success when they take actions, a motor speed controller, and the main processing unit. We call this model trains as an agent in our RL problem. Figure 1 shows the agent running on track with customized hardware.

2.1 Problem Description

In our experiment, each agent in the environment is capable of controlling its speed from very slow to very fast. We described it as values between 0 and 255. Considering that the agent receives a task as moving from one point to another point on the tracks, our goal is to minimize the traveling time for the agent. That means the faster agent will go

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

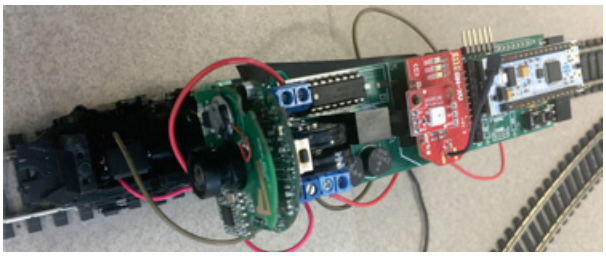

Figure 1: Model trains as the agents with sensors on them.

forward earliest it will reach its destination, but the problem is that if we set the max speed for the agent, it is very likely that it will flip over or derail due to the fact that there are different sharp angles on tracks, anomalies on connections and joints, uneven connections between track segments, switching segments with sensitive tracks, and so on. That will make it impossible for the agent to move forward with max speed all the way until it reaches the goal.On the other hand, setting a slow, safe speed for the train will make sure that the agent will reach the destination. However,it will lose the optimal condition we are looking for, which is to reach the goal as fast as possible.

In general, we can define our problem as a behavior optimization problem in which our agent almost knows how to reach its goal, but it needs to tune its behavior somehow to reach the destination in optimal time. This behavioral knowledge can be gained from prior experiences by the agent or can be a solution suggested by a human. There have been studies in behavior optimization of robots in reinforcement learning problems like using reinforcement learning to refine robot motor control (Franklin 1988) or use human-generated control strategies to bootstrap a robot controller using Q-learning. (Smart and Kaelbling 2000).

In addition to minimizing the travel time, we would also like to minimize the number of times that the agent will fail to complete the path (flip over or derail) during the learning process.

3 Comparing To Human Performance

There have been many types of research in the field of reinforcement learning to compare human performance to reinforcement learning performance (Knox and Stone 2012) and also how human experience can help to boost an RL algorithm performance. (Mnih et al. 2015) (Thomaz, Breazeal, and others 2006) Also, to demonstrate our experimental results in reinforcement learning, as the second part of our experiments, we compare the human learning rate and performance results where humans will manually control the speed of trains via a joystick.

It is commonly stated that reinforcement learning algorithms take a large number of observations and samples in comparison to a human (Doshi-Velez and Ghahramani). As the second part of our experimentation, we investigate how humans perform in controlling the speed of an agent to finish the lap in the minimum amount of time and how the error rate is compared to the RL algorithms.

3.1 Test Cases

We have run our learning algorithm on two different paths that we have selected on tracks. The learning procedure starts with the agent moving forward from a designated starting point and finishing up at the end of the loop. This procedure repeats until the the agent learns optimal policies. In test case one we first test our approach in simple path 2 and demonstrate our results. In addition, we asked humans to control the agent manually on the same path and compared the results.

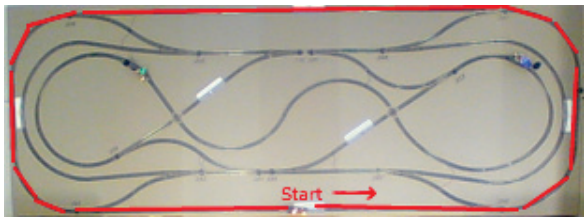

Figure 2: Simple path taken by the agent in test case 1.

For our second test case, we use a long and complex path as it is shown in Figure 3, which includes additional turns in both directions.

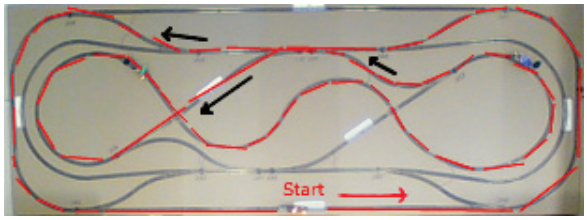

Figure 3: Complex path taken by the agent in test case 2.

In our problem set, we divided all tracks to segments and labeled them as different states that the agent can be in at each time step. Figure 4 shows each state with a different color for for test case 1. Agents can take actions by changing their speed from 0 to 255. The lower the value the lower the speed would be. Zero speed turns off the motor, but the train will slide based on its previous speed.

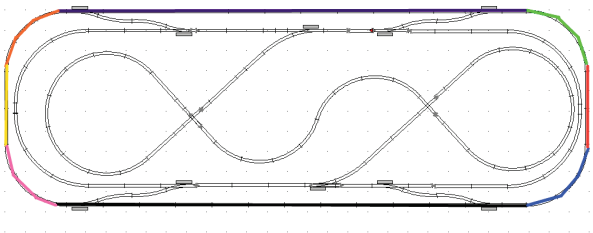

Figure 4: States as partition of segments the agent can be in each time step.

3.2 Feedback from the Environment

Each agent is equipped with gyroscope sensor on it. We use this sensor as our state signal from the environment based on the action we perform. A gyroscope is widely used in robotics for stabilization (Fuller et al. 2014) (Brown and Xu 1997) and motion control (Nagasaka et al. 2004). It returns Yaw, Roll, Pitch values of the robot.

Since we have the robot on a flat surface, we can ignore the pitch values while it will never change. The yaw value gives us the direction that the agent is facing which we will not use it as our state-action feedback. Hence, we are only interested in roll values from the gyroscope that describe if the train is flipping over. Trains have a flexibility of some degree that they can push left and right as they are on curves on tracks. The yaw value has a significant change between $90°$ to $-90°$ when the train flips over. This indicates to the learner that the action taken on a specific state leads to failure. These values are observed by plotting roll values of the agent and running it from a lower speed to the maximum speed on a simple path shown in 2 with curves until it loses its control and goes off the track and flips over.

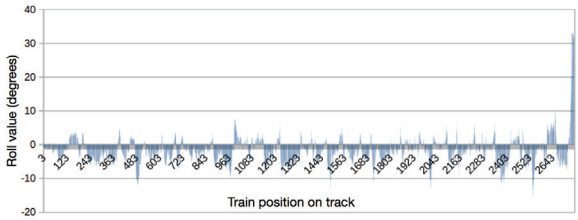

Figure 5: Gyro roll value readings on loop path with increasing speed.

Figure 5 shows the plotting of roll values. From that, we can observe that the roll values of more than 20 indicate a flip-over state. In addition to that, we can predict that before the agent flips over at time t, it has experienced some high roll values at time $t-1$. This can be used as a warning for the agent that a little bit more speed on the same state is going to cause a failure. This is very important for us in designing the reward function to reduce the number of times the agent flips over significantly. If we define R as our gyro roll reading value we have:

$$\text{Feedback State} \begin{cases} \text{Critical Pass} & 7 > R > 6 \\ \text{Non-Critical Pass} & R < 6 \\ \text{Flip over} & R > 7 \end{cases} \quad (1)$$

3.3 Reward Function

It has been proven that choosing a good reward function can have strong effects on how long the learner will spend the time to learn the optimal policy in an environment (Mataric 1994) (Randløv and Alstrøm 1998).At the beginning, we simply defined our reward function as follows:

$$R(s,a) \begin{cases} +5 & \text{if state } s \text{ is passed successfully} \\ -5 & \text{if state } s \text{ pass is fails (train is flipped over)} \end{cases} \quad (2)$$

The agent will receive 5 rewards if it reaches the destination without flipping over and 5 penalties if along the path it flips over. In that case, the agent must start the learning process from the beginning. From our observations in Section 3.2, for the critical roll values that the agent experiences, This will be a warning for the agent. If it increases its speed in the same state in next iteration, it will be flipped over by a minor change in reward function:

$$R(s,a) \begin{cases} +5 & \text{if state } s \text{ is passed without critical gyro value} \\ -10 & \text{if state } s \text{ is passed with critical gyro value} \\ -20 & \text{if state } s \text{ pass fails (train is flipped over)} \end{cases} \quad (3)$$

We were able to decrease the number of times the agent flips over in the learning process from 10 to 2 in test case 1,

3.4 Modeling the Problem as MDP

We use the standard Markov decision process (MDP) Formulation (Bertsekas and Tsitsiklis 1995) for representing the fully observable environment.

$$\begin{aligned} &S: \text{a set of states (position of agent on track) } s \in S \\ &A: \text{a set of actions (speed range 0-255) } a \in A \\ &R(s,a) \rightarrow r: \text{reward received by taking action } a \text{ at states } s \\ &R(s,a) \begin{cases} +5 & \text{if state } s \text{ is passed without critical gyro value.} \\ -10 & \text{if state } s \text{ is passed with critical gyro value.} \\ -20 & \text{if state } s \text{ pass fails (train is flipped over).} \end{cases} \end{aligned} \quad (4)$$

4 The Q-Learning Algorithm

As we do not have a model of our world based on a probability distribution function $T(s,a,s')$ which defines the probability of transitioning from state s to state s' using action a. In addition, modeling mechanical characteristics of the trains is very hard task to do. Hence, we selected one of the most widely used model-free reinforcement learning techniques called Q-Learning (Kaelbling, Littman, and Moore 1996) (Watkins and Dayan 1992) due to its algorithmic simplicity and also the ease of transitioning from state-value function to an optimal control policy (Martinson, Stoytchev, and Arkin 2001) by choosing every state-action with the highest value. Q-Learning has been used in many robotic applications from stabilization of a Biped robot (Park, Jo, and Kim 2004) to path planning for multi-agent soccer robots (Kim et al. 2000).

In our experiment, we implement Q-Learning to find the best policy for the agent that maps the state to the optimal action, which in our case is the fastest and safest speed for the agent in each state. We define $Q^*(s,a)$ as discounted reinforcement of taking action a at state s. Afterward taking actions optimally, we also define $V^*(s)$ as the value of s assuming that the best action is taken initially. Hence, we define it as:

$$V^*(s) = \max_a Q^*(s,a) \quad (5)$$

We can define our optimal policy as :

$$\pi^*(s) = \arg\max_a Q^*(s,a) \quad (6)$$

where $Q^*(s, a)$ is formulated as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))\right] \quad (7)$$

where γ is the discount factor between zero and one that defines how much future rewards will be more substantial than the reward that the agent will receive from the very next state. α is called the learning rate, which defines how much the learner values the new experience above experiences from previous actions. In our implementation, α is set to 1.

5 Learning Process

Beginning from the starting point, the agent will move forward with a slow speed while calculating Q-values and saving them in the Q-matrix. At the end of each loop, the agent increases its speed by 5. This continues process ends when the agent reaches maximum speed and all Q-values are updated. Obviously, we will have flip-overs while learning. In such cases, we put the agent back to the starting point and resume the learning process.

The output of our reinforcement learning implementation would be the list of optimal actions. In our case this is the speed for the agent for the list of the states, which are the track segments in our environment.

6 Experimental Results

In this section, we show the results of our implementation running on two different paths. Our first test was to run a Figure 6 shows the result of running the learning process on Test case 1. Red plot shows the finish time of the lap with no learning where we have increased the speed of the agent in each loop until it flips over. This indicates the best time that the agent can finish the lap without any learning algorithm. Blue plot shows the lap finish time by the agent when we run our learning algorithm where we can see a significant improvement in finishing the lap.

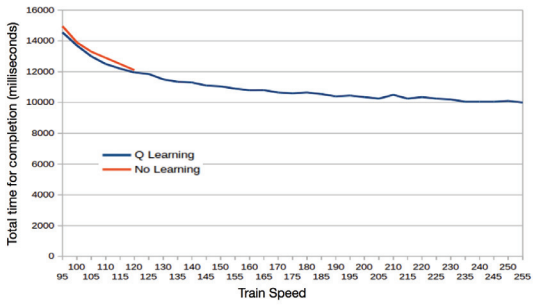

Figure 6: Q-Learning and no learning results on path1.

Table 1 shows the results of test case 1 for humans and the RL algorithm, as well as average performance of five humans and RL algorithm.

As shown in our results, for a simple path covering the perimeter of the track, humans learned how to control the agent in much fewer laps than the learning algorithm did. The finish time was very close to what Q-Learning achieved but RL algorithm still recorded better lap finish time. Without dynamic speed change, the agent could finish at 12 seconds without flipping over which, was 2 seconds more than Q-Learning and human finish time. Hence, we can assume that Q-learning achieves better results compared to setting fixed speed to the agents. The standard deviation the RL algorithm for five runs was 0.092 seconds, while the standard deviation for humans was 0.135. A t-test ((Student 1908)) with 99% confirmed the statistical difference between average finish time taken by humans and by the Q-Learning algorithm..

Table 1: Q-Learning results for Test Case 1.

	# of Laps	# of Failures	Finish(sec)
Learning Best	33	2	10
Human Best	6	1	10.44
Learning Average	10	5	10.54
Human Average	33	3	10.16

For our second experiment, we picked a more complex path with more curves. Figure 3 shows the path for Test case 2. Figure 7 shows the comparison of the finishing time of the loop on Test case 2 by the agent using our learning algorithm.

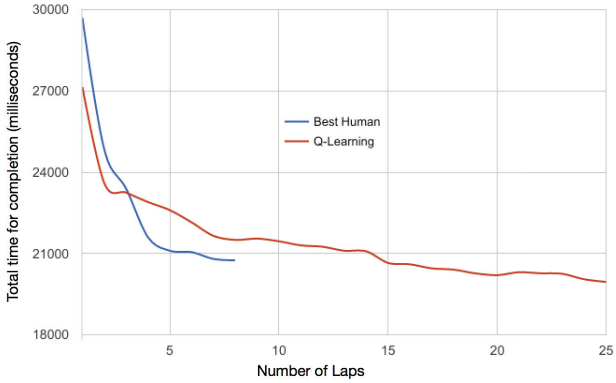

Figure 7: Lap finish time for the best results of human and RL algorithm.

Similarly to our approach in test case 1, we again ran our experiment with humans manually controlling the speed of the trains through a joystick. We have recorded the best results for five different participants, all given some initial time to familiarize themselves with the controls. Table 2 compares the results of the best finish time by humans and our learning algorithm.

Unlike our first experiment, on a simple path that the human lap finish time was almost the same as our learning algorithm. As we changed the path to a more complex path, the number of errors by the manually controlled agent increased and also the reinforcement learner recorded better results in finishing time.

Table 2: Q-Learning results for Test Case 2.

	# of Laps	# of Failures	Finish(sec)
Learning Best	26	2	19.9
Human Best	8	5	20.6
Learning Average	12	8	20.97
Human Average	33	3	20.11

We again conducted a t-test for test case 2, evaluating the is a statistical difference between time required for a person to complete the more complex path (test case 2), and the time required by the learning algorithm.

The variance of the reinforcement learning algorithm measurements over five runs was 0.158 seconds, while the variance of measurements across five participants was 0.431 seconds. Using a 99% level of confidence our t-test has rejected the null (H_0) hypothesis that both measurements (between humans and the RL algorithm) were equal.

Figure 7 shows the best performance of humans on manual speed control with learning algorithm over five runs. Our results show that humans need a smaller number of trials to learn in comparison to a machine, but the learning algorithm has a better finishing time than the humans after 25 laps.

Figures 8 and 9 shows the actions selected by the agent and human for each lap in the learning process. Figure 10 demonstrates the best actions selected in the best lap for humans and the RL algorithm.

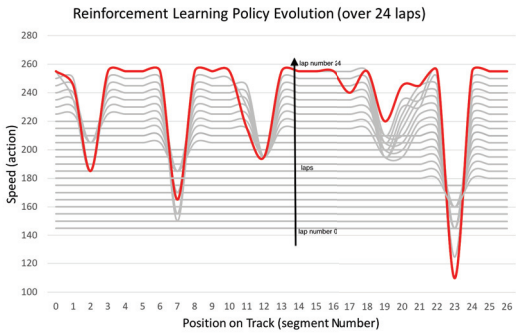

Figure 8: Actions (speeds) selected by the agent for each lap (best result plotted in red).

Figure 10 shows that the human and the learning algorithm has a similar reaction to the different states but machine picks actions in a more optimal manner and it has better performance in finishing time. In addition, human decreased speed in some states which was not a critical state.

7 Conclusions

In this paper, a physical testbed to run our proposed model-free reinforcement learning algorithm to minimize the travel time of the model trains operating on tracks has been developed. By finding the best safe speeds for different states, our reinforcement learning algorithm will prevent trains from flipping over or derailing. Also it is shown in our results that

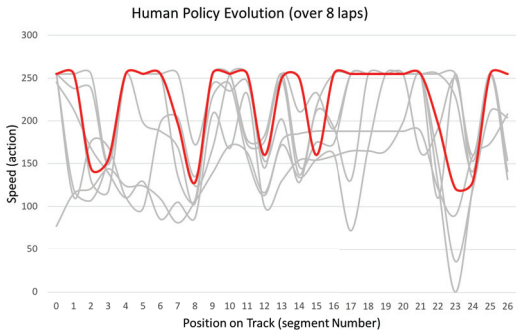

Figure 9: Actions (speeds) selected by human for each lap (best result plotted in red).

we can minimize travel time significantly by using reinforcement learning compared to setting a fixed speed to trains.

The second part of our experimentation was to compare human controller behavior to the reinforcement learning method from the learning rate and optimal action selection perspective. We have asked humans to control the speed of trains manually by a joystick, and our results show that humans can learn faster in simpler tasks with almost close finishing time performance in comparison to the reinforcement learning algorithm. On the other hand, when we have selected a complicated path to be tested, we witnessed that human performance decreases in comparison to learning algorithm and the number of errors made by the human controller increases as the travel path gets more complicated.

Finally, we used t-test statistical examination method to proof the statistical difference between humans and the reinforcement learning algorithm results for both test cases.

References

Bertsekas, D. P., and Tsitsiklis, J. N. 1995. Neuro-dynamic programming: an overview. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 1, 560–564. IEEE.

Brown, H. J., and Xu, Y. 1997. A single wheel, gyroscopically stabilized robot. *IEEE Robotics & Automation Magazine* 4(3):39–44.

Dayan, P. 1992. The convergence of td (λ) for general λ. *Machine learning* 8(3-4):341–362.

Doshi-Velez, F., and Ghahramani, Z. A comparison of human and agent reinforcement learning in partially observable domains. Citeseer.

Franklin, J. A. 1988. Refinement of robot motor skills through reinforcement learning. In *Proceedings of the 27th IEEE Conference on Decision and Control*, 1096–1101 vol.2.

Fuller, S. B.; Helbling, E. F.; Chirarattananon, P.; and Wood, R. J. 2014. Using a mems gyroscope to stabilize the attitude of a fly-sized hovering robot. In *IMAV 2014: International Micro Air Vehicle Conference and Competition 2014, Delft, The Netherlands, August 12-15, 2014*. Delft University of Technology.

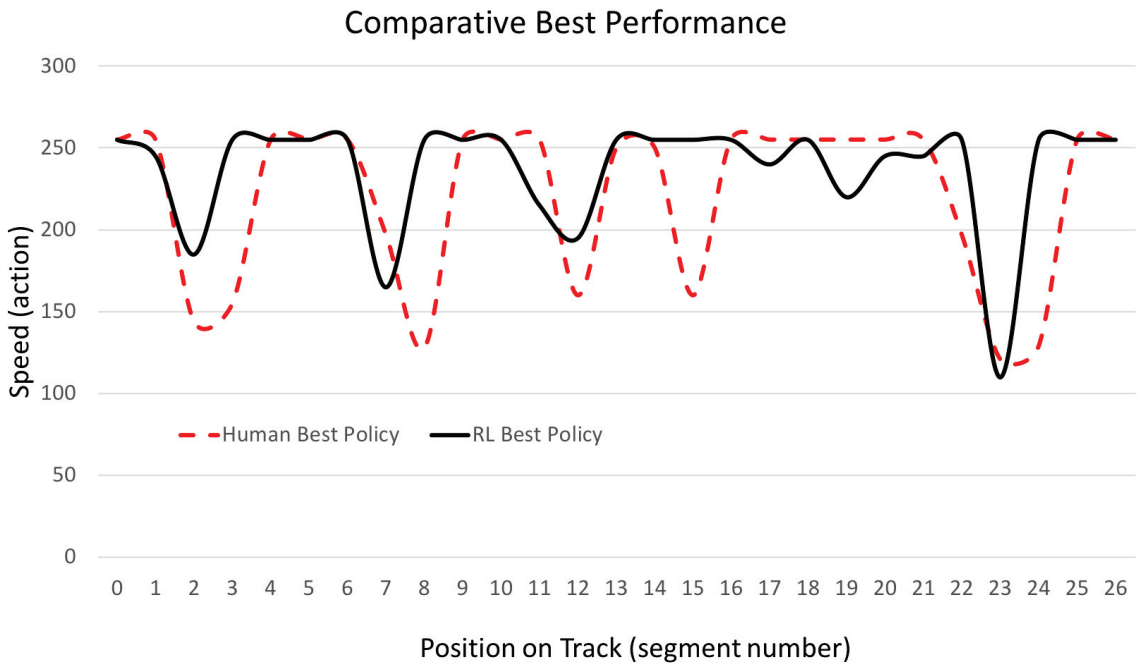

Figure 10: Comparison between human best action selection lap and Q-learning best action selection lap.

Howard, R. 1960. *Dynamic Programming and Markov Processes*. Published jointly by the Technology Press of the Massachusetts Institute of Technology and.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. W. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* 4:237–285.

Kim, D.-H.; Kim, Y.-J.; Kim, K.-C.; Kim, J.-H.; and Vadakkepat, P. 2000. Vector field based path planning and petri-net based role selection mechanism with q-learning for the soccer robot system. *Intelligent Automation & Soft Computing* 6(1):75–87.

Knox, W. B., and Stone, P. 2012. Reinforcement learning from human reward: Discounting in episodic tasks. In *ROMAN, 2012 IEEE*, 878–885. IEEE.

Martinson, E.; Stoytchev, A.; and Arkin, R. C. 2001. Robot behavioral selection using q-learning. Technical report, Georgia Institute of Technology.

Mataric, M. J. 1994. Reward functions for accelerated learning. In *Machine Learning: Proceedings of the Eleventh international conference*, 181–189.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Nagasaka, K.; Kuroki, Y.; Suzuki, S.; Itoh, Y.; and Yamaguchi, J. 2004. Integrated motion control for walking, jumping and running on a small bipedal entertainment robot. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, 3189–3194. IEEE.

Park, K.-H.; Jo, J.; and Kim, J.-H. 2004. Stabilization of a biped robot based on two mode q-learning. In *2nd International Conference on Autonomous Robots and Agents*, 13–15.

Randløv, J., and Alstrøm, P. 1998. Learning to drive a bicycle using reinforcement learning and shaping. In *ICML*, volume 98, 463–471. Citeseer.

Smart, W. D., and Kaelbling, L. P. 2000. Practical reinforcement learning in continuous spaces. In *ICML*, 903–910.

Student. 1908. The probable error of a mean. *Biometrika* 1–25.

Thomaz, A. L.; Breazeal, C.; et al. 2006. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Aaai*, volume 6, 1000–1005.

Tsitsiklis, J. N. 1994. Asynchronous stochastic approximation and q-learning. *Machine Learning* 16(3):185–202.

Watkins, C. J., and Dayan, P. 1992. Q-learning. *Machine learning* 8(3-4):279–292.

Zhang, W., and Dietterich, T. G. 1995. A reinforcement learning approach to job-shop scheduling. In *IJCAI*, volume 95, 1114–1120. Citeseer.