Markov Decision Processes

Chapman Siu

1 Introduction

This paper will analyze two different Markov Decision Processes (MDP); grid worlds and car racing problem. These problems were implemented and provided by the MATLAB MDP toolbox from INRA[1].

1.1 Grid World

In this problem, we have an agent which resides in a world consisting of grids. The agent has a simple goal of reaching the bottom right or top left square and can start anywhere. There are four possible actions which an agent can have; left, right, up and down. The agent will move to the correct direction with probability p and in any other direction with probability $1 - p$. There is no reward for moving from one tile to another, but there is a reward of 1 for moving to the goal tile (i.e. the bottom right tile). Furthermore if the agent tries to move outside the world or into an obstacle then it will stay in the same space and get a reward of -1.

1.3 Car Race

The car race problem is where a course is given with a starting and finishing line. We are required to find the optimal sequence of acceleration and brake events, as well as the path to accomplish the course as fast as possible. Noise is added by considering that there is a probability p where a particular action is not executed; this is to imitate human beings where sometimes we may not be able to react or execute a series of commands perfectly. To motivate the agent in finding the shortest path whilst avoiding obstacles, an award of -1 is provided for every time unit that passes by, and an award of -100, whenever a collision occurs. The discount rate is set to 0.99 since we are interested in the number of time steps to reach the finish line, and not the distance traveled. Maximization of reward should lead to the optimal track time.

The car race problem is set up as follows:

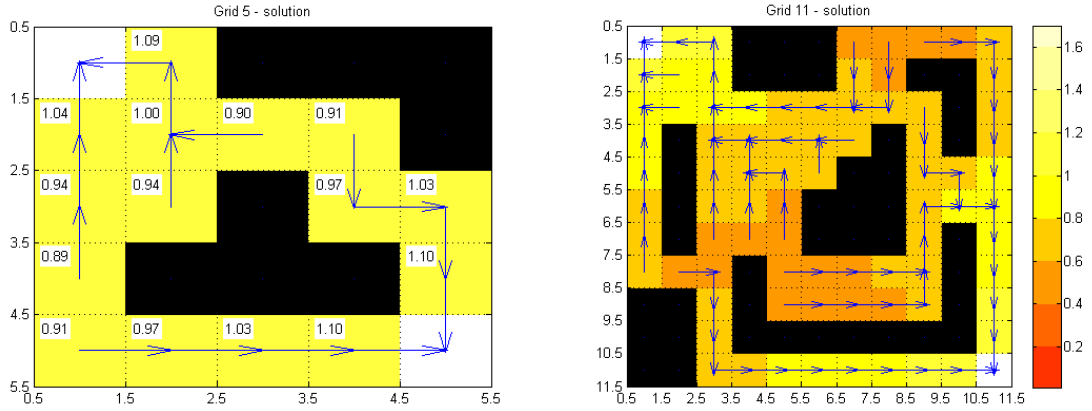
- State is modelled as a 4-tuple, containing the x, y coordinates and also the velocity in the x, y directions.
- At each step the car can accelerate, brake or keep constant by one unit in each of the x, y direction, leading to 9 possible actions.

2 Grid World

Two different grids of different size were used, a 5x5 grid and 11x11 grid. In both cases the optimal policy using policy and value iteration with a discount rate of 0.9, whilst moving in the correct direction 0.9 of the time; were exactly the same. For both of these problems, both policies were verified that they were indeed the optimal policies by solving the MDP using linear programming which is an exact method.

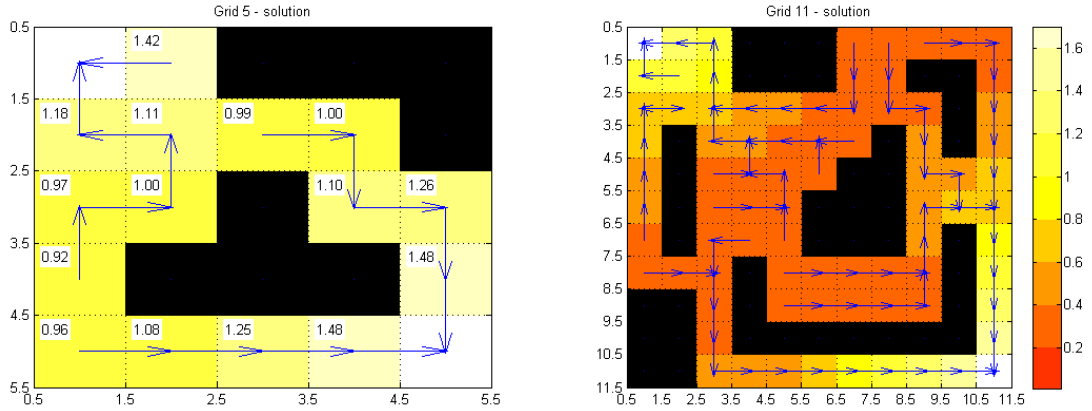
Problem	# States	Algorithm	Iterations	Time (in seconds)
5x5 Grid	25	Value	12	0.006424
		Policy	4	0.006675
		Relative Value Iteration	4	0.004178
11x11 Grid	121	Value	24	0.009119
		Policy	10	0.017132
		Relative Value Iteration	3	0.003803

From the table above, it would appear that despite having 2-3 times more iterations, the value iteration algorithm is preferable to the policy iteration approach since the execution time is marginally faster than the policy iteration approach. Graphed below are the optimal solutions, with the grid shaded according to values as determined by the value function. In comparison, using the linear programming approach to find the exact solution took five times longer for the 5x5 grid and twenty times longer for the 11x11 grid.



Grid solution with error rate of 10%, Left - 5x5 grid, Right - 11x11 grid

When error rate is 40%, both value and policy iterations converged to the same policy.



Grid solution with error rate of 40%, Left - 5x5 grid, Right - 11x11 grid

In the solution with 40% error, we can observe that there are open areas in the 11x11 grid which have the same value according to the value function. In comparison the solution with 10% error has a much more noticeable stepwise improvements in the value as we progress towards either one of the goal states.

3 Car Race

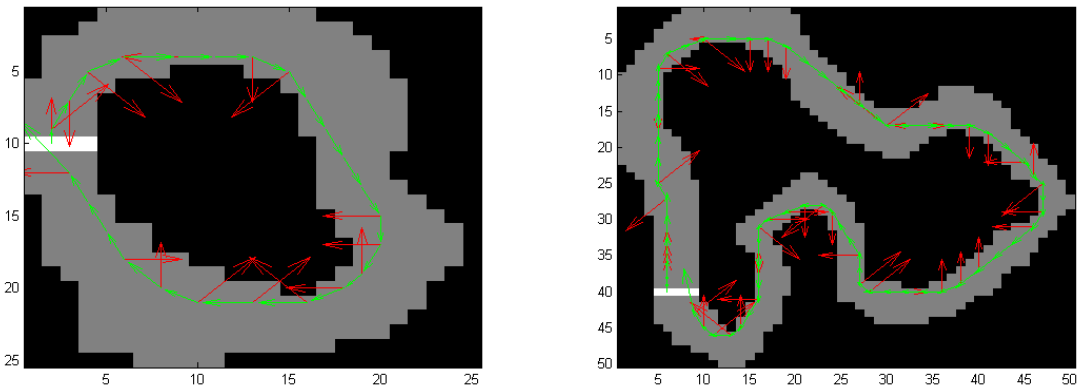
The car race problem is a much larger problem with more states, and 9 actions as oppose to 4 actions in the grid world problem. The state space chosen for this problem is over 9,000 states for one race, and over 23,000 states for the other. It is clear that the transition matrix will be quite sparse, since the space is large and the number of actions small. Hence using value iteration may take longer than policy iteration approach.

The table below shows the results when the probably of error was 0.1, with a discount rate of 0.99. For both approaches, the optimal policies converged to the same solution.

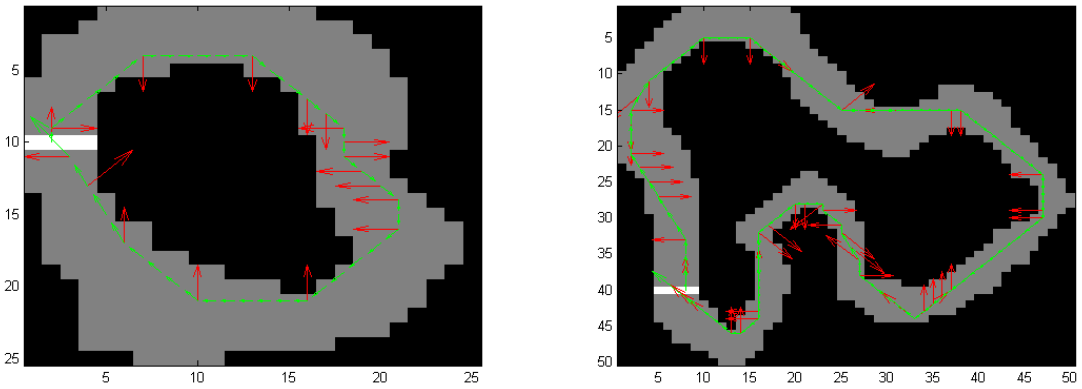
Problem	# States	Algorithm	Iterations	Time (in seconds)
25x25 Grid	9,253	Value	36	0.829661
		Policy	11	0.118273
		Relative Value Iteration	26	0.071055
50x50 Grid	23,782	Value	83	2.185702
		Policy	16	0.447792
		Relative Value Iteration	73	0.426905

Comparing the table above with the results in the grid world problem, the number of value iterations required has now increased to 8-10 times more than the policy iteration approach, resulting in 5-7 times speed improvement in solving the optimal solution.

From the graphs below, when increasing the error rate from 0.1 to 0.4, the optimal policy shifts to a more simple implementation with straighter path (in green), which in turns influences the chosen acceleration vectors (in red). Observe that in the case where the error rate is 40%, the acceleration vectors are repeated multiple times to ensure that the driver will correctly implement the intended strategy. Indeed, through simulation we can demonstrate that sometimes when the commands are not implemented correctly the agent may indeed "crash" and fail to arrive at the final state.



Car race solution with error rate 0.1, Left - 25x25 grid, Right- 50x50 grid



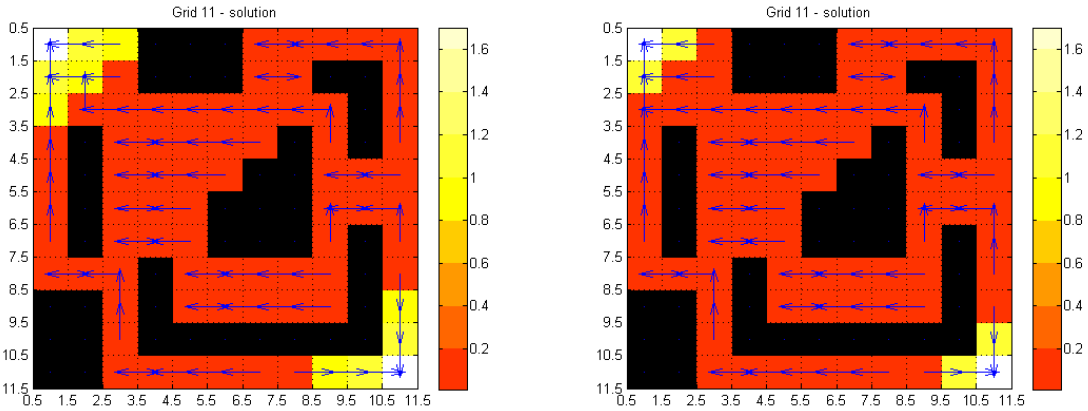
Car race solution with error rate 0.4, Left - 25x25 grid, Right- 50x50 grid

4 Relative Value Iteration

The other reinforcement learning algorithm used was the relative value iteration approach. This algorithm is another model based approach similar to value iteration. This algorithm is useful in problems where the optimal policy converges much quicker compared with the function or when there is no discount function. From the table above, we can see that the number of iterations and time taken for the relative value iteration is superior to the value and policy iteration approach. Within this algorithm we are concerned with the convergence of the difference in the value function $|v(s) - v(s^{'})|$ than we are about the values of the value function. Since the relative value iteration is simply subtraction a constant value vector from each iteration, it may affect the value vector, however this should not alter the optimal policy that relative value iteration finds[2].

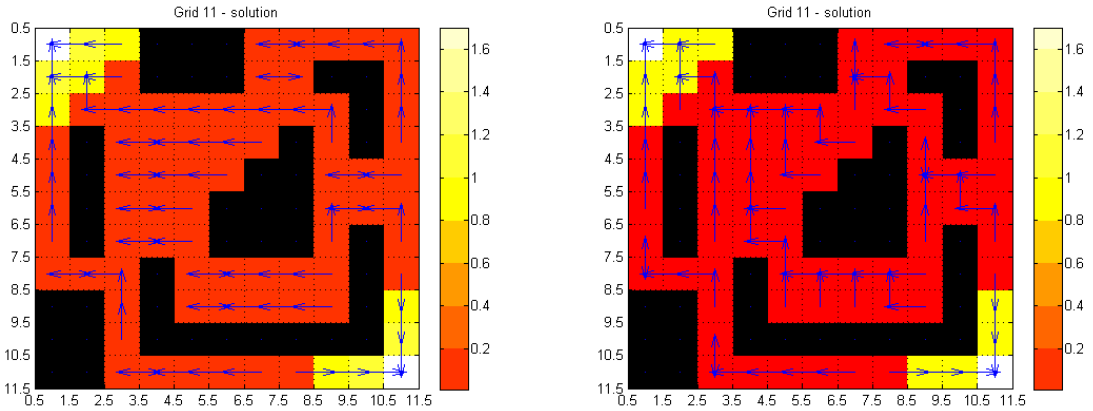
4.1 Grid World

From my analysis and the graphs below it is clear that the relative value iteration approach does not converge to the same solution as the policy and value iteration approaches. Regardless if the error rate for the agent was 0.1 or 0.4, if agent was following the policy precisely it can indeed get stuck within various places in the grid world. This appears to be related to the algorithm since by minimizing the change in the value function. Its indifference in the bottom corner demonstrates how the agent does not appear to favor either direction.



Solutions to 11x11 grid world problem, Left: error rate 0.1, Right: error rate 0.4

Due to the nature of the relative value iteration, we might be able to conclude that it is probably not suitable for such a simple problem. To further demonstrate the algorithms sensitivity to the rewards and by extension values and how this influences the optimal policy, the graphs below show the solution to the relative value iteration on one hand with no changes, and the other where instead of gaining an reward of 0 when entering a non-goal state, instead rewarded with a value of -0.01. Note that in this instance for the value and policy iteration approaches, the optimal policy converged to the same solution both times, though of course the value matrices were different.



Solutions to 11x11 grid world problem, both with error rate of 0.1. Left: No changes same set up as the policy and value iteration, Right: All non-goal states have a -0.01 reward. Notice that compared with the value and policy iteration approaches there are places in each grid where the agent can be "stuck". In both setups the solution for policy and value iterations are the same as in section 2.

This difference is more pronounced and demonstrates how intuitively the relative value iteration works when we consider the scenario where the discount rate is instead zero. Here observe how the algorithm iteratively approaches the values which minimize the difference rather than approaching the solution which by inspection would be better.

4.2 Car Race

In the car race example, using relative value iteration proved to be a useful tool since it was very fast and demonstrate a compromise in the trade-offs between value and policy iteration approaches, having lower iterations that the value iteration approach, but at the same time taking shorter amount of time to converge to a solution. This is a suitable approach since using other algorithms like linear programming would take over 15 minutes to determine an optimal policy for this problem, which is an order of magnitude larger than relative value iteration approach.

Also, this approach is much more suitable for how this problem was framed since each non-goal state had a negative value, allowing convergence of the values difference to be correctly reflected. This resulted in the same optimal policy whether the policy discount was set to 0 or 0.99.

However, in general when we take the cumulative value determined by simulating the car's path within the race for the car race problem, it is clear that under the relative iteration approach the cumulative value is lower.

Problem	Algorithm	Cumulative Value
25x25	Value/Policy Iteration	-348
	Relative value iteration	-323
50x50	Value/Policy Iteration	-2561
	Relative value iteration	-2064

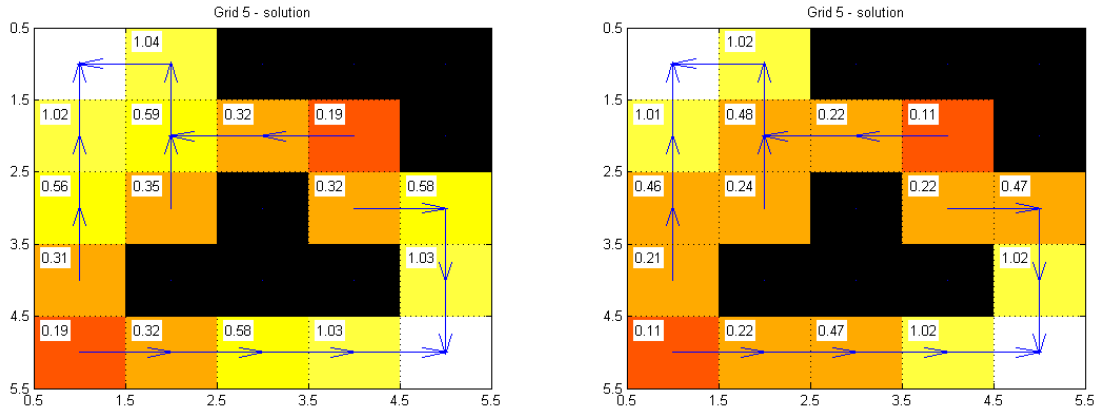
5 Discount Rate Analysis

Within MDPs, Discount rates can be defined to *not* be part of the problem definition, but instead a parameter which can be altered. Here we will have a quick look at how changing discount rates for our algorithms and problems above can affect the solution of the problem.

5.1 Grid World

For the 5x5 grid explained above with the probability of error being held constant at 0.1, changing the discount rate from 0.9 to 0.6 does not alter the solution for the policy iteration approach or the relative value iteration approach; it converges instead to the same solution as when the discount rate was set to 0.9. However, the solution for the value iteration instead converged to the same solution as the relative value iteration algorithm. In order to determine which solution is the optimal one, we can use linear programming to find the exact solution, which reveals that the solution provided by policy iteration was actually the optimal policy.

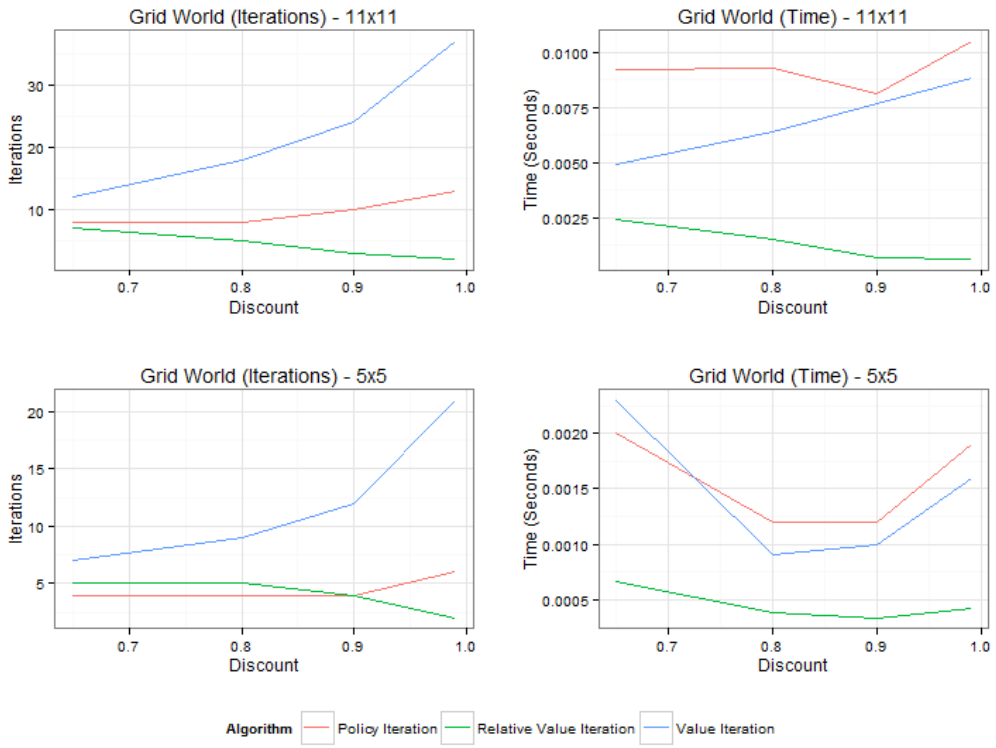
If we change the discount rate to 0.5, then the three algorithms and the linear programming approach all converge to exactly the same solution. In fact, if we were to change our discount rate from 0.1 all the way to 0.9 it appears that policy iteration approach would always converge to the optimal solution as determined by the linear programming approach, whilst relative value iteration and value iteration may not.



Car race solution for value iteration with discount rate 0.6 on left, and 0.5 on the right. Observe that both policy solution converge to the same path, with the values altered by discount rate.

For the 11x11 grid, the results were similar for the policy and value iteration. Policy iteration approach was exactly the same as the exact solution, whilst the value iteration approach did not have the same optimal solution as the linear programming or policy iteration solution.

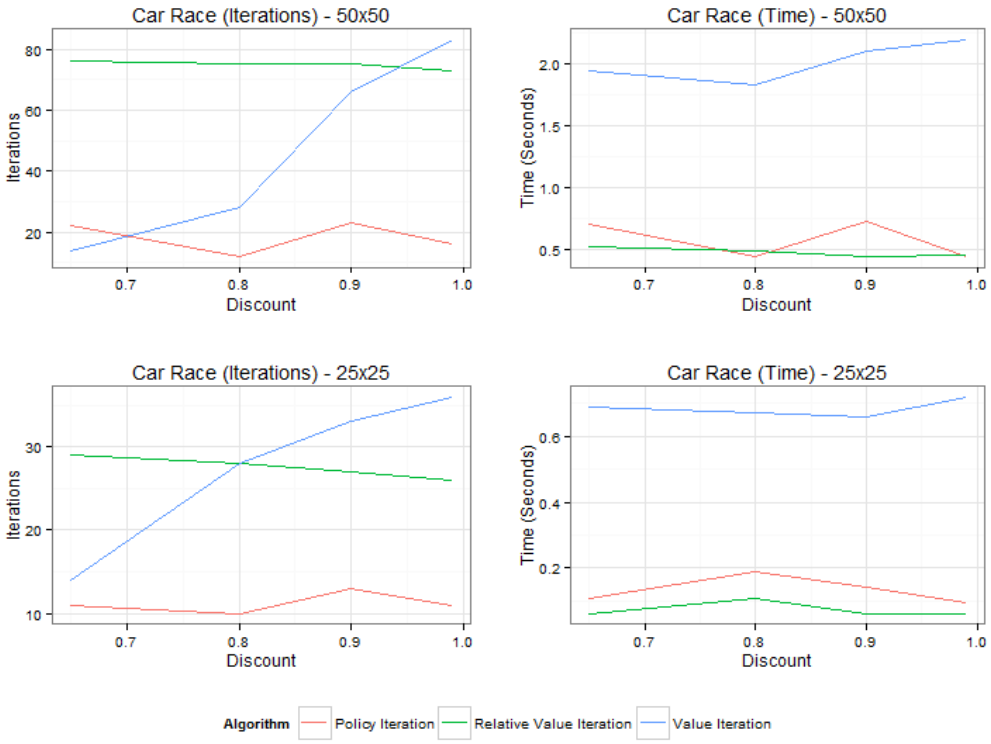
With respect to time, there was no noticeable difference in the time taken to solve any of the MDPs by changing the discount factor.



For this problem, it would appear that the relative value iteration algorithm dominates in the performance benchmarks. It has lower number of iterations, and is the fastest algorithm. However, it is also the algorithm which in general does not converge to the optimal solution derived from the exact solution of the MDP.

5.2 Car Race

The car race problem as described above, is a more difficult to analyze, simply because the time required to find the exactly solution takes over 400 times longer to find compared with the other three algorithms.



If the discount rate dropped to 0.6, then the policy iteration algorithm will not converge. This is most likely due to the reward at the finish being discounted too aggressively leading to a value close to zero. Hence the discount rate should be set as high as possible to avoid this situation from occurring.

Most interesting is the iteration graphs, which shows the number of iterations increases as the discount rate moves towards one, whilst the value iteration approach appears to be moving in the other direction. This is probably a reflection of the discount rate which directly affects the bellman equation for this problem, with lower discount rates forcing the older rewards to move towards zero quicker. This is observed in both the Car race problem and the grid world problem.

6 Conclusion

From the two problems analyzed, policy iteration approach performs better when the state space is large compared with the value iteration approach. Careful design of the transition and reward matrix can lead to optimal policies by both algorithms.

If finding an optimal policy quickly is the objective, then relative value iteration approach may be an ideal approach, especially when finding the best solution is not important. in general it would appear that policy iteration performs well in both problems presented. Further problems

could be examined using different algorithms, or formulating a MDP which consists of many more actions to determine the suitability of the various reinforcement learning algorithms.

7 References

[1] Chades I., Chapron G., Cros MJ., Garcia F., Sabbadin R. (2014). MDPtoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems. Ecography 37:916-920.

- Home page: http://www7.inra.fr/mia/T/MDPtoolbox/
- Car race home page: https://mulcyber.toulouse.inra.fr/frs/?group_id=11&release_id=579

[1] Powell, W. (2011). Approximate Dynamic Programming: Solving the Curses of Dimensionality, 2nd Edition. Chapter 3, section 3.4.2.

- Chapter excerpt: http://castlelab.princeton.edu/ORF569papers/Powell_ADP_2ndEdition_Chapter%203.pdf