Markov Decision Processes, Planning Algorithms, and Reinforcement Learning

Writeup for Assignment 03 - CS 6741

Magahet Mendiola

ABSTRACT

An empirical analysis of planning and reinforcement learning algorithms in the context of Markov decision processes.

1. PERFORMANCE METRICS

A number of simple metrics were used to evaluate the performance of our planning algorithms. These include the number of iterations and elapsed time required for utility values at each state to converge. However, the goal of planning is to find the optimal policy for an MDP, which does not require complete value convergence. Therefore, we should also compare time time and iterations to discover the optimal policy for each state.

The rate at which a policy converges toward the optimum is also important. If a partial plan can be created quickly, it may be good enough for practical use on a given MDP. This evolution can be measured with the hamming distance between the current policy and the optimal policy at each iteration. We will evaluate our planning and learning algorithms based on this metric.

Another derivation of the hamming distance metric is the time/iterations required to find the policy which will successfully guide a deterministic agent to the goal state. The idea behind this metric is that there may be many states in an MDP that are never visited. Running value or policy iteration until enough of the states have optimal policies to guide the agent could greatly reduce planning time. Creating a planning algorithm with this stopping criteria would be similar to q-learning in regards to limiting exploration. It would also be similar to a* path finding, with search areas radiating omni-directionally from non-zero reward states.

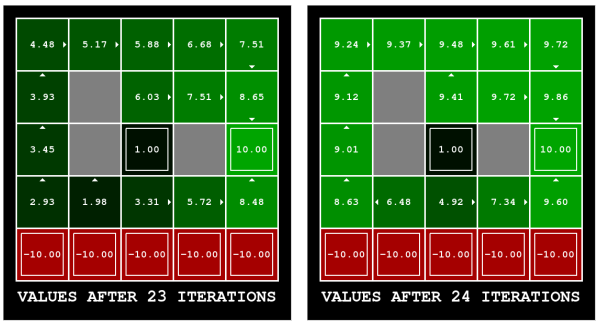
2. EXAMPLE MDPS - GRID WORLDS

Each algorithm was tested against contrived grid world type MDPs. These worlds were setup to accentuate the strengths and weaknesses of each algorithm. They also illustrate the impact of rewards, discounts, and the stochastic nature of the process on the behavior of our planning and learning algorithms. States are referenced in these grids as coordinate pairs enumerated from west to east and south to north. For example, the bottom left corner (state) of a grid world is referenced as (1, 1). The state directly to the right (east) of this is referenced as (2, 1).

2.1 Discount Grid

This grid world, titled Discount Grid [1], is useful for illustrating the effects of utility discounting, transition function probabilities, and state rewards on optimal policies. It includes two terminal states with positive rewards (+1 closer, +10 farther) and five negative terminal states at the bottom edge of the grid. This setup allows us to explore what set of problem and solution parameters affect the goals defined by an optimal policy.

Figure 1a shows the resulting values of the MDP after running value iteration with a γ of 0.9 and a reward of zero in all the non-terminal states. The transition probabilities are 0.8 for desired action and a combined 0.2 for all orthogonal movements. In this case, the policy directs the agent to seek out the farther, higher valued, terminal state. It also avoids the high penalty cliff except when the alternative is moving into the lower reward terminal state.



(a) 0.9 (b) 0.99

Figure 1: discount grid - various γ settings

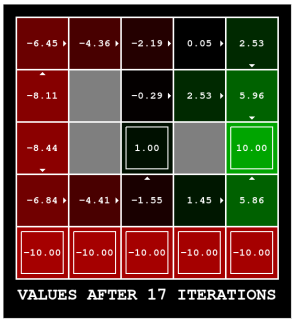
Time matters.

With a γ of 0.99, which reduces the effects of discounting, the policy shifts slightly (Figure 1b). The policy in state (2, 2) now risks a direct movement to the west, given the higher expected utility of that action. This is due to the higher relative value of state (1, 2) after being discounted less than before. Also, state (3, 4) now avoids the risk of falling into the lower reward for the same reason. The expected value of attempting a northward movement is now relatively more valuable, despite the indirect route to the terminal state. This shows the implications of adjusting delayed reward and how an agent could be lead to take greater risks or follow an otherwise sub-optimal route to gain a larger reward farther in the distance.

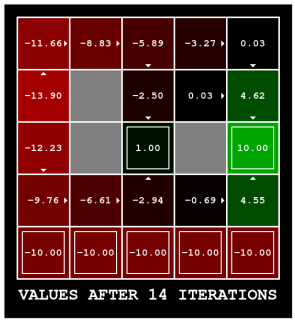
Rewards matter.

With a shift in rewards, we can affect how the result-

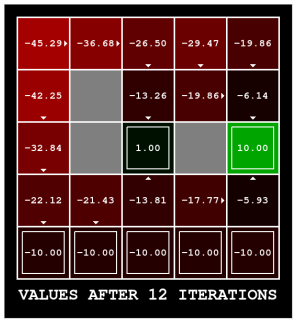
ing policy directs the agent. Figure 2a shows the result of changing the reward in all non-terminal states to -2. The agent now believes that the closer terminal state is a better alternative at state (3, 2). The negative rewards, and risk of falling off the cliff, push the agent toward the closer end point. With the non-terminal rewards set to -3, this early termination policy extends to the northern path as well (Figure 2b). With a sufficiently negative non-terminal reward, we can even entice the agent to end the process as quickly as possible, by intentionally jumping off the cliff (Figure 2c).



(a) -2



(b) -3



(c) -11

Figure 2: discount grid - various non-terminal reward settings

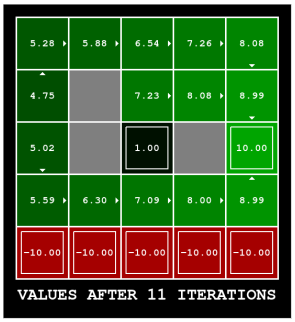
The problem matters.

Policies are not only affected by planning algorithm parameters, but also by the problem definition, such as the probabilities in the transition function. If movement in the MDP is deterministic, or the intentional action probabilities are sufficiently high, otherwise risky paths would be considered optimal. Figure 3a shows a policy of cliff-walking given only a 1% change of falling off the ledge. Conversely, the policy in Figure 3b shows how our planning algorithm would approach a high risk environment (inebriated agent). In the later case, the agent will avoid any east-west movement, even at the expense of exiting at the low reward terminal state.

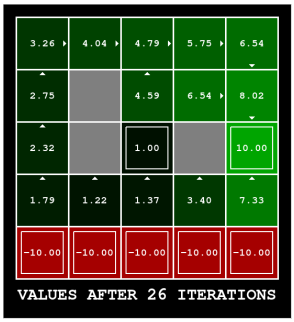
2.2 Tunnel Grid

Tunnel grid is a one dimensional grid with the starting state on one end and the only terminal state on the other. Although this grid is simplistic, it highlights the distinctions between our planning and learning algorithms well.

For example, Table 1 shows the elapsed time and the number of iterations/episodes each algorithm took to converge. In the case of Q-learning, an episode is determined as a single



(a) 0.01



(b) 0.4

Figure 3: discount grid - various unintentional movement probabilities

(a) value iteration

(b) policy iteration

(c) q-learning

Figure 4: tunnel grid

training run from the starting state until the agent reaches a terminal state. Also, convergence for the Q-learning agent is defined as the state when setting ϵ to zero and following the best q-score at each state would result in the optimal policy.

	Elapsed Time	Iterations/Episodes
Value	0.04	29
Policy	0.04	19
Q-Learning	0.390	18

Table 1: tunnel grid - performance

We see from the first two rows that value iteration requires nearly 50% more training cycles to converge. However, policy iteration and value iteration are identical in terms of elapsed time. This is due to the greater computational requirements of policy evaluation at each interval. Q-learning required a substantially longer amount of time to train.

In the case of the tunnel grid, Q-learning will perform identically to an agent that guesses randomly (random walk) until it eventual reaches a non-zero reward state. Until then, none of the states it visits will produce a meaningful Q-value. This is highly inefficient, as the expected number of states the agent will visit before reaching the end is $n^2 + n$ with n being the number of states between the start and terminal point. Once this state is eventually reached, the Q-values will be updated only for that state. The agent would have to perform another random walk to the penultimate state for Q-values to propagate further. In total, the expected number of steps the agent would have to move before a full

path is created in this MDP is:

$$\sum_{n=1}^{\#ofstates} n^2 + n \tag{1}$$

In a 19 state tunnel grid world, that comes out to 2,660 expected steps before a full path of q-values are propagated. Value and polity iteration would propagate utility values to the starting state in only 19 iterations, making our learning algorithm's performance look terrible in this example. However, Q-learning cannot be directly compared to our planning algorithms as it does not have prior knowledge of the MDP and must explore each state to build it's own modal of the world. This does highlight opportunities for improvement in the learning algorithm.

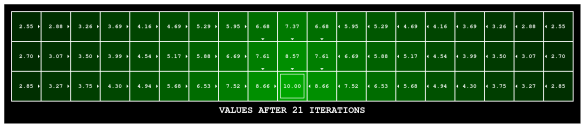
A more intelligent agent would keep track of previously visited states that resulted in all zero q-scores and place greater action probability on moving to previously unexplored states. Another enhancement would be to remember the path taken by the agent and update q-values for the entire path once a non-zero reward state is encountered, rather than only updating the closest state.

If the grid world is modified slightly, the strengths of Q-learning become evident. With the starting state on the north wall of the tunnel and the terminal state on the south wall, the Q-learning agent does not need to randomly explore very far before finding the terminal state. In fact, it takes only three episodes and 141 total agent moves (0.01 sec) to discover the optimal path (Figure 5b). If the agent was allowed to continue running, it would eventually discover the entire MDP and assign Q-scores and policies. However, there is no need in this case. The majority of the states are superfluous and exploring them would not improve the agent's optimal path in the context of this MDP.

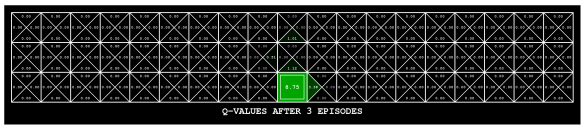
Figure 5a shows the comparative waste with value iteration. It does a comprehensive evaluation of all the states and stops only when the furthest state update falls below the stopping threshold. It's analogous to searching an entire house before trusting that the front door will get you outside. Domain knowledge would be very helpful in guiding such algorithms. For example, the stopping condition for value or policy iteration could be set such that it triggers once a path is revealed between predetermined states. Taking this further, each of these algorithms could be modified to behave directionally; only propagating value/policy towards a target state. This would tailor them more toward the specific use case of path finding, but should work well in the context of grid worlds. It does, however, suppose special knowledge of the MDP regarding a known target state. As we saw in the discount grid example, there are many situations where the environmental conditions will dictate what should be considered the target state.

2.3 Simple Grid

To evaluate value and policy iteration performance empirically as the number of states increase, each algorithm was run to convergence on grids ranging from 3 by 3 (9 states) to 30 by 30 (900 states). A single, positive valued, terminal state was set at one corner. Figure 6 shows the results of this experiment. Within these grid sizes, iterations seem to grow linearly with respect to the grid edge size, which seems reasonable considering the radiating propagation behavior. Elapsed time for each shows greater than



(a) value iteration - 0.11 sec



(b) q-learning - 0.01 sec

Figure 5: side tunnel grid

linear growth, which can be partially attributed to the polynomial growth of the number of states. Relatively, policy iteration elapsed time seems to grow at a greater rate than value iteration. At 900 states, the difference is as great as 40% slower.

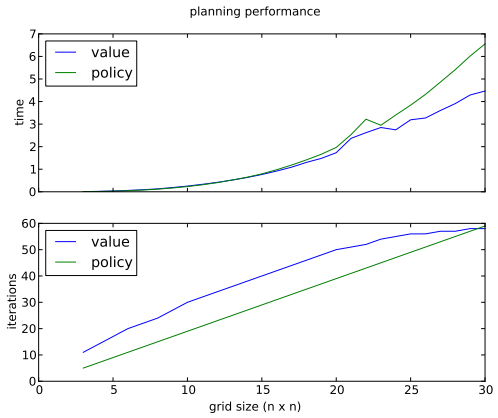


Figure 6: planning performance by grid size

Hamming distance over time is shown in Figure 7. This plot shows how each algorithm converges toward an optimal policy. Policy iteration, although slower overall, shows a smooth transition as it updates the policy at each adjacent state during subsequent iterations of the algorithm. Value iteration updates may not update state policies each time, and we see from the plot that with around 300 non-optimal state polices, the performance curve makes a significant behavioral shift. This is probably due to change in the time complexity of updating each state as non-zero values are propagated outward from the terminal state. The policy iteration curve likely smooths the effects of these state value updates over the entire curve, as fully converged values are assigned to each state during every iteration.

Q-Learning.

For our reinforcement learning agent, this simple grid provides an opportunity to explore the effects of various α (learning rate) and ϵ (exploration rate) settings. First, we can look at how quickly the utility value at the agent's starting point accumulates over time. Figure 8 shows the progress of five agents with various learning rates as they explore a 30x30 simple grid. A slight negative reward is set at each

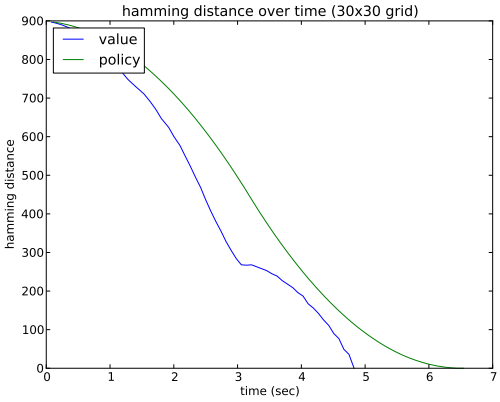


Figure 7: algorithm convergence over time

non-terminal state to penalize non-optimal movement. With an α of 1, we see that q-scores propagate quickly to the starting state as the agent puts the most weight on recently computed q-scores. However, we do notice a dip and an ongoing variance after reaching a stable value. This is due to the fact that the agent is acting in a stochastic environment, and deviations from the optimal path will have a greater affect on the computed q-scores at each (state, action) pair visited during an episode. There are two methods we could implement to improve this. First, we could start with α at 1 and decay this value over time. The idea being to transition the agent from exploration to exploitation. The second strategy would be to maintain a separate α for each (state, action) pair, and to decay the value based on the number of times that pair has been observed by the agent. This would make the exploration-exploitation transition local to well visited states.

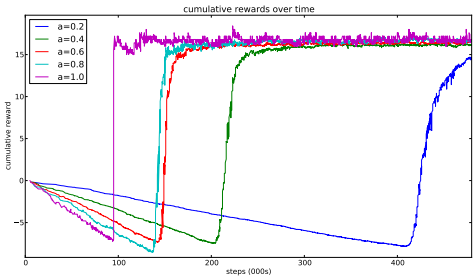


Figure 8: cumulative rewards over time (various α)

Figure 9 shows the progress of our Q-learning agent on a 30x30 simple grid, with various ϵ values. We see that with a higher exploration incentive, the agent takes more steps to converge on an optimal path. However, we also see less sacrifice in reward during exploration, as many episodes of random exploration will ignore previously learned suboptimal paths. The advantage of epsilon-greedy exploration can be seen clearly with a slight modification to the grid. In Figure 10 we see the result of running the same agents on a grid with a trap state (lower reward) in the center of the grid. We see that agents with low ϵ get stuck at the lower reward state. More adventurous agents will continue searching for better rewards, eventually finding the higher valued terminal state.

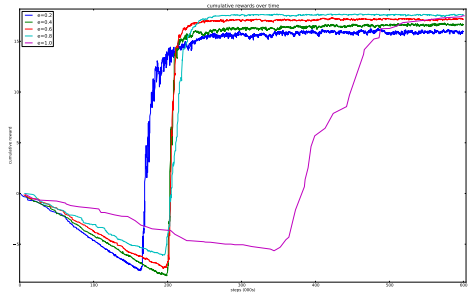


Figure 9: cumulative rewards over time (various ϵ)

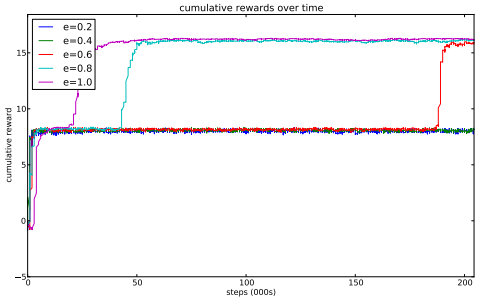


Figure 10: cumulative rewards over time - trap grid (various ϵ)

Figures 11 and 12 show a smaller version if this trap grid world. Figure 11 shows an agent with a small exploration incentive after over 12,000 episodes. The resulting policy and computed state values show how the agent did not explore the farther state enough to influence the established q-scores. Instead the agent repeats the path from the starting state to the first positive state, focusing almost exclusively on exploiting it's current knowledge of the world.

Figure 12 shows an agent with a higher ϵ value. This agent effective learns to bypass the low reward state in favor of the farther state. It also finds this optimal policy with less than 1,500 episodes. This exploration driven agent has one major disadvantage. All those moves spent exploring are wasted if the farther, more valuable, state doesn't exist. As with the learning rate, an agent that decreases ϵ over time would be able to gradually transition between exploring and exploiting.

2.4 Random Grid

Theoretically, performance of both value and policy iteration are determined by the number of states and actions. To validate this empirically, an experiment was conducted which generated random grid worlds of a fixed size. A random number of positive and negative terminal states were scattered throughout the grid, along with a random number of unreachable states. Figure 13 shows an example of one of these grid worlds. Value and policy iteration was performed on each of these grids to determine if iteration or time per-

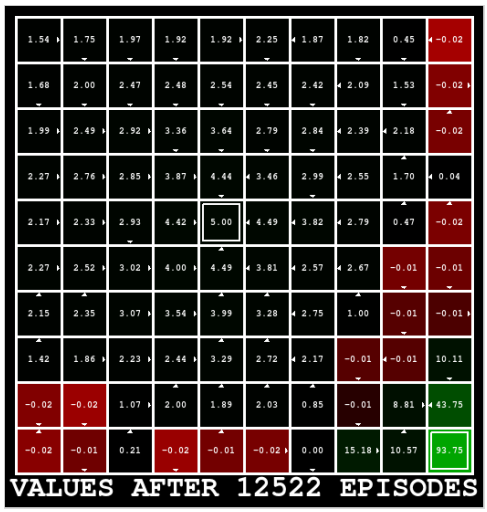


Figure 11: trap grid - (0.2 ϵ)

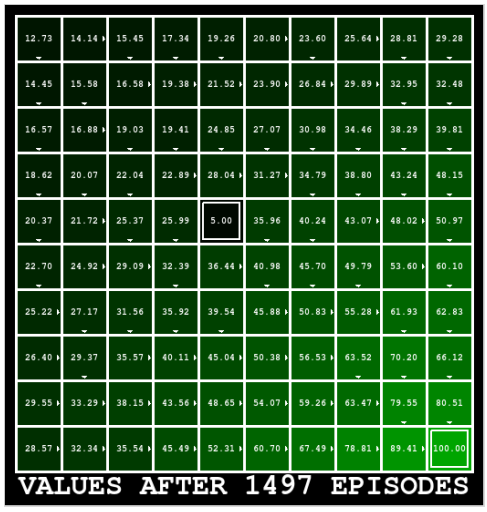


Figure 12: trap grid - (0.8 ϵ)

formance would vary with each layout. Through thousands of such random generations, performance was almost identical between each planning process. The only variance being accounted for by random isolated states (surrounded by walls).

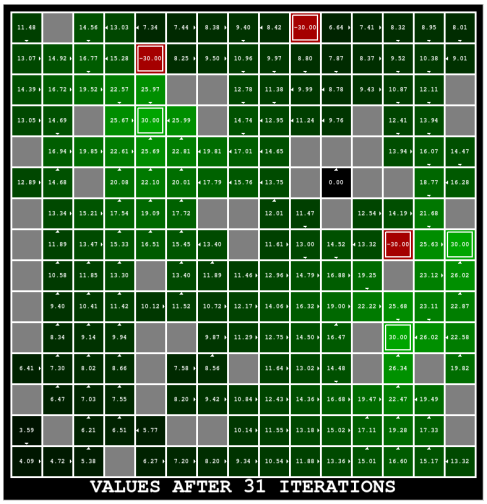


Figure 13: random grid

Q-learning on these random grids performed with mixed results, depending on the starting state of the agent. In some cases, the agent was restricted to an isolated section of the grid by walls. In these cases, exploration of the reachable area was completed quickly and an optimal policy was discovered to the nearest positive terminal state. Performance was also affected by the general shape of the grid. As we noted with the tunnel grid example, Q-learning performed well when positive terminal states were close to the starting state and large sections of the grid could essentially be ignored. In these situations, better stopping criteria would be helpful. Something similar to the update threshold in value and approximate policy iteration. In fact, a reverse of epsilon greedy search could work well. The process would involve decreasing epsilon until an established path was determined. Epsilon would then be increase to encourage the agent to focus on exploration. Given that that optimal policy is already determined, there is less need for improving the already established path and more effort can go into finding alternative paths and reward states.

3. REFERENCES

[1] D. Klein. *Project 3: Reinforcement Learning*, 2009 (accessed April, 2014). http://inst.eecs.berkeley.edu/ cs188/fa09/ projects/reinforcement/reinforcement.html.

[2] S. J. Russell, P. Norvig, J. F. Candy, J. M. Malik, and D. D. Edwards. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.