Computational Modelling of Algorithms Based On Cooperation of Model-Free and Model-Based Strategies

Aditya Loth
21111004
adityal21@iitk.ac.in

Harsh Agarwal
21111030
harshmag21@iitk.ac.in

Varun Vankudre
20111064
varunsv21@iitk.ac.in

Abstract

Decision-making problems are characterized by the exploration-exploitation trade-off. An agent, at all times, needs to balance between exploring new choices and exploiting known options, to reach the optimal policy and maximize the resulting cumulative reward. It was theorised that model-free and model-based compete for dominance over animal behaviour, but recent studies have shown that these models can cooperate in some circumstances. In this paper, sequential decision-making experiments have been recreated and for an agent it is shown how this cooperation between model-free and model-based strategies can be computationally represented and simulated. We implemented DYNAQ and Model-Based Pseudoreward Approximation (MBPA).

1 Introduction

Free will is anything but free. With it comes the onus of choice: not only what to do, but which inner voice to listen to - our 'automatic' response system, which some consider 'impulsive' or 'irrational', or our supposedly more rational deliberative one.[1] Research suggests that we have Dual-process system, expressing human cognition as the result of two interacting systems, such as systems that produce habits vs. goals explain a range of fundamental properties of human decision-making and judgment. These systems are Model-Free and Model-Based. In Model-Free, agent does not utilise environment information; instead, it uses trial and error feedback to update the reward associated with a state-action pair. Thus this model represents habitual behaviour as "an automatic and rapid habitual process linking reward to associated action and enabling the reflexive repetition of previously successful choices." In Model-Based, agent uses environment information to build a decision tree-like structure containing the relationship between state transitions pair and expected reward, making it computationally expensive. This model represents goal-directed behaviour as "a slow, deliberative, goal-directed process comparing the potential outcomes of each action and identifying the action most likely to generate the desired outcome."

A Model Free system is designed on Thorndike's law of effect. The system stores a look-up table that contains observed and predicted rewards for state-action pairs. Learning and prediction is computationally efficient in this system because only the table needs to be scanned and update reward values but it requires large amount of time and experience to have a reliable policy as it does not take into account the reward and transition information about the environment.

On the other had a Model Based system is designed on Tolman's cognitive map. The system takes into account information pertaining to reward and transition behavior of the environment to estimate future rewards for state action pairs and then stores the action and maximum future reward for a state in a value and policy table respectively. Although in this system requires small amount of experience and time to learn optimal policy, this is not computationally efficient and so not feasible in all scenarios.

In a Decision-Making task, for a set of states and corresponding valid action(s), the agent is always in a conundrum to choose an action in current state which will maximize cumulative future rewards. Decision making in new environments is always plagued with the conflict between agents desire to maximize its rewards and desire to collect as much information as possible about environment to see whether there is some other state with higher reward. This ubiquitous conflict is what researchers call exploration - exploitation trade off.

It is agent trying to maintain a balance between exploiting known state action pairs and exploring for new state action pairs with greater reward. Through various experiments we will show that using DynaQ and MBPA gives better performance in terms of time taken to learn optimal policy, no of steps required to learn the optimal policy and no of steps agent takes in optimal policy to maximize its cumulative reward and reach goal state (whenever present).

2 Background

2.1 Q Learning

Q Learning is one of the most famous and easy to understand Model Free Reinforcement Learning algorithm. It requires no prior knowledge of environment. In this a table called 'Q Table' is maintained for all possible state action pairs and is initialized at the beginning with suitable values. For each time the agent interacts with the environment, where it is in say state s_0 takes action a and reaches state s_1 and receives reward r then entry $Q(s_0, a)$ is updated using

$$Q(s_0, a) \leftarrow Q(s_0, a) + \alpha(r + \gamma \max_{a'} Q(s_1, a') - Q(s_0, a))$$

where α is learning rate and gamma is discount factor. The term $(r + \gamma \max_{a'} Q(s_1, a') - Q(s_0, a))$ is called temporal difference error. In most of the experiments, agent uses ϵ-greedy policy where agent chooses with probability $1 - \epsilon$ the action with maximum $Q(s_0, a)$ value and with probability ϵ chooses an action randomly with equal probability. ϵ is used to specify the exploration-exploitation tradeoff.

2.2 Dyna

DynaQ is Model Free learning combined with Model Based system that replays the experience. After each interaction with the environment the reward is received and value of state action is stored then selects randomly state action pair which previously has been seen and then replays them n times. Model Free system interacts with the environment and Model Based system replays the experience.These n planning steps updates the Q table the model based the knowledge based on the previous experience. There is also neurophysiological data with suggest the cooperation of system like DynaQ. During the sleep the hippocampal cells gets activated is correlated with and proceeded activation of the same striatal cells that encoded the value of locations.

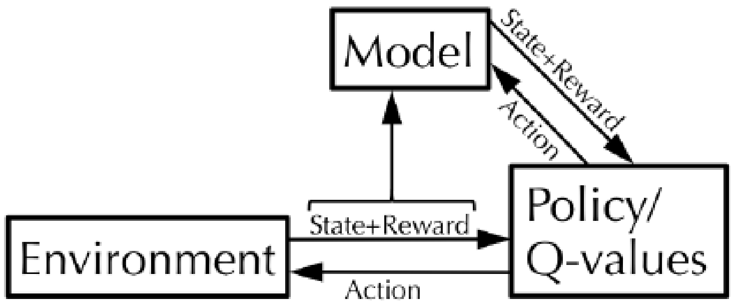


Figure 1: Dyna Architecture[2]

2.3 Model-Based Pseudo-reward Approximation

MBPA uses a different way of showing the cooperation between Model-Based and Model-Free learning system. In MBPA the dynamic programming is used to approximate the state values. New reward function is built on these state values. The shaping theorem uses these new function as augmented reward.

$$R'(s,a,s') = R(s,a,s') + F(s,a,s')$$

Here $R(s,a,s')$ states that reward from the actual environment and $F(s,a,s')$ is the shaping function for computing Pseudorewards. There is one critical property that shaping function needs to satisfy that it should be a potential-based shaping function that is optimal policy should remain unaffected by addition of Pseudorewards. Equation for this in variance property is

$$F(s,a,s') = \gamma\Phi(s') - \Phi(s)$$

where Φ is a real valued function that maps states to real numbers. If above property is not satisfied then agent may learn an sub-optimal policy which is not good. Therefore researchers suggest that simple and best Pseudorewards is the difference in optimal state values between agent's current and next state, that is

$$F(s,a,s') = \gamma V^*(s') - V^*(s)$$

Such Pseudorewards encourage agent to choose optimal action at each state and if $\epsilon = 0$ then agent directly follows the optimal policy thus going on shortest path to the goal state. However in real life scenarios, it is difficult to have such complete information and also computationally inefficient to compute optimal state values as it would require large amount of iterations of Bellman equation

$$V^*(s) = \max_a R_{\pi(s)}\left(s,a,s'\right) + \gamma \sum V^*\left(s'\right)$$

where π is the optimal policy.
The optimal policy under the value function may be costly to compute so we approximate the value function to get the optimal policy. The value function monotonically converge to the optimal value. Here we take advantage of this property, to calculate the approximate value of the states. We perform the n bellman updates to approximate the value of states, as the n increases, state values start converging to the optimal value. Choosing the n value is trade-off between approximation of optimal value state and the computation time and resources.

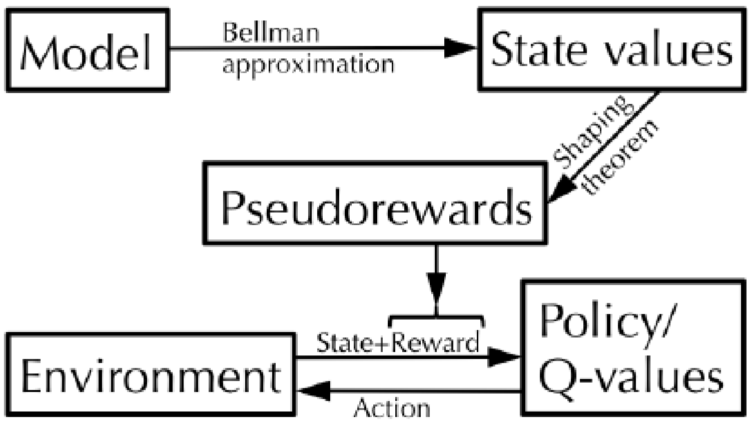


Figure 2: Model-Based Pseudoreward Approximation.[2]

3 Experiments

3.1 Two Step Decision Task

Phase 1(Pre-training phase):- The agent is given two choices, 'A' and 'B' at stage 1, which deterministically transition to stage 2 and stage 3. Instead of feedback in terms of reward, the background and choices of the respective stage are displayed

Phase 2(training phase):- Reward feedback associated with choices are revealed. Choice 'A' has a greater reward than choice 'B.' 20 trials of this phase are conducted one after the other

Phase 3(training phase):- Agent start at either stage 2 or stage 3 with uniform probability. Stage 2 choices 'C' and 'D' are rewarded less than stage 3 choices 'E' and 'F'. No of trials in this phase was variable and its effect on revaluation magnitude was observed

Phase 4(testing phase): The agent was asked to choose between stage 1 choices 'A' and 'B' without reward feedback but was instructed to make decisions based on previous experience from all three phases.

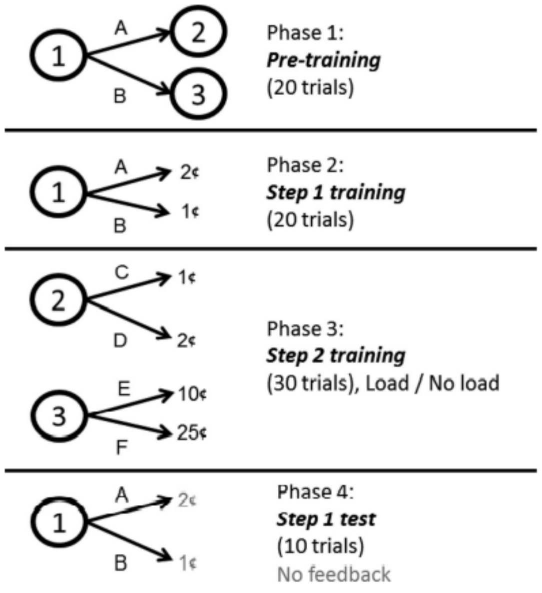


Figure 3: Two step Experiment Design. [3]

Above 4 phases were followed in the reference paper to conduct experiments with human subjects but to conduct this experiment with a computer agent, we slightly modified the experiment by removing phase 1 and directly encoding this information into our agent. Also as we are using softmax policy

$$\mathrm{P}\left(a_t = B \mid s_t = s\right) = \frac{\exp\{\beta Q(s,B)\}}{[\exp\{\beta Q(s,A)\} + \exp\{\beta Q(s,B)\}]}$$

to determine action at each state, there was no need to conduct trials at phase 4, we directly use the probability of the agent to choose higher reward action from softmax policy.

When tested with humans, they showed retrospective revaluation and chose choice 'B' against choice 'A'. When tested with model-free and model-based agents, they did not show these results. The DynaQ architecture replicates human results by integrating model-based planning with model-free learning.

3.1.1 Result

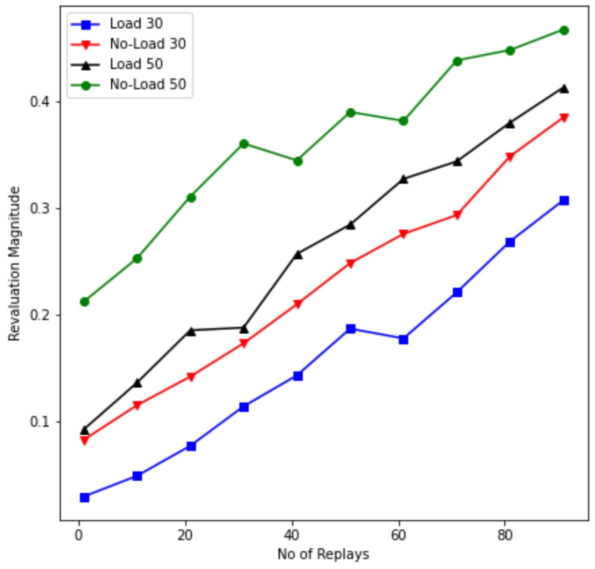


Figure 4: Revaluation Magnitude of agent using the DYNAQ algorithm for the Two-Step Task

Figure 4 represents the simulated revaluation magnitude observed as an function no of replays between Phase 3 and Phase 4 for all four conditions. It is observed that With increase in number of replays between phase 3 and phase 4 trials the revaluation magnitude increases consistently, which helps us conclude that the more time to study past experiences, more frequently optimal decision is taken. It is also observed that with increase in number of trials the revaluation magnitude increases.The revaluation magnitude of 'Load' condition is less than 'No-Load" for the same number of trials which depicts that having "Load" hampers learning in comparison to having 'No-Load'.

3.2 Maze-learning

In this problem the agent is made to learn the maze environment to reach the goal state from the start state.Agent starts at top left corner of maze denoted by 'S' in Figure 5 and has to reach in the diagonally opposite corner that is bottom right corner marked with 'G'.Agent gets a reward of '10' on reaching the goal state and '0' for all other states. We also use a discount factor (γ) 0.95 to ensure agent tries to choose shortest possible path to the goal state. We have chosen a 11x11 Square maze with 50 walls with unique path from start to goal state for this experiment. Epsilon greedy policy was used to choose action at each state with epsilon value of 0.25. We used DynaQ and MBPA to simulate this experiment.

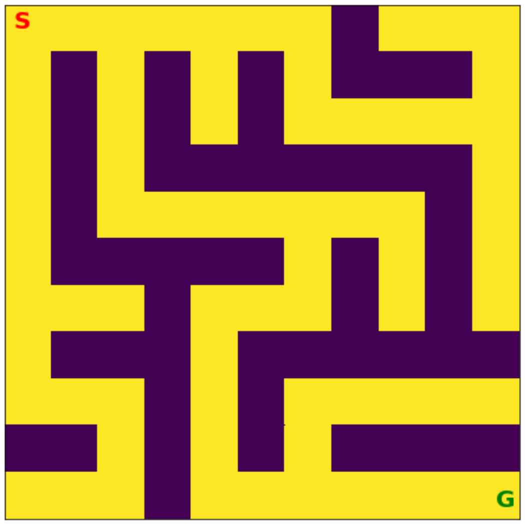
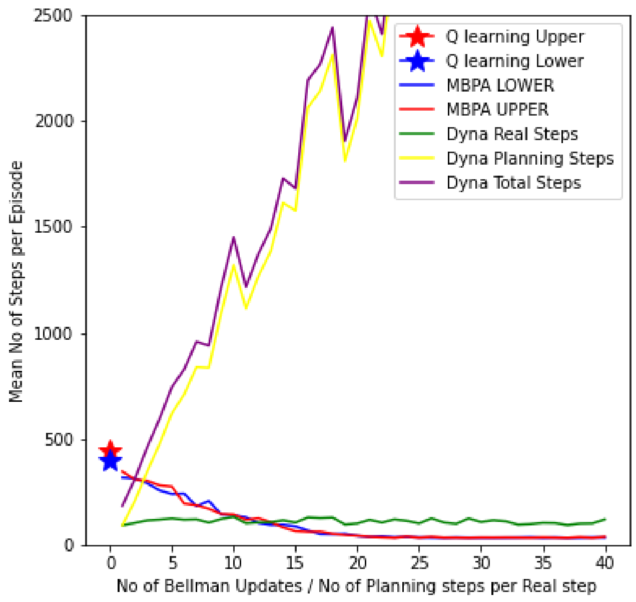
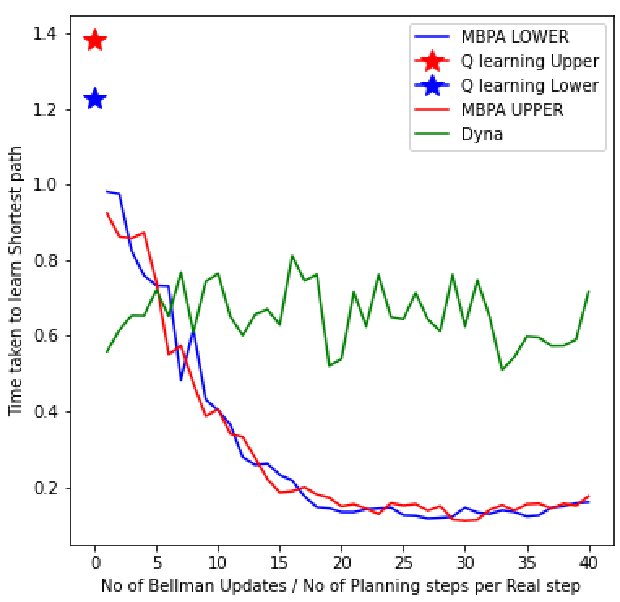


Figure 5: 11x11 Maze

3.2.1 Result



(a) No of steps.



(b) Time in seconds

Figure 6: Maze Learning.

Figure 6a shows the mean no of steps required per episode to reach goal state as no of Bellman updates performed during approximation of Pseudorewards. As can be seen that as no of Bellman updates increase, no of steps decrease and after certain no of Bellman updates, the no of steps required get almost constant. The graph also shows the No of steps required by Q learning and DYNAQ algorithms. It is also observed that after certain number of Bellman updates, DYNAQ real steps are greater than steps taken by MBPA.

Figure 6b shows that although in the beginning with small number of Bellman updates, MBPA requires more time than DYNAQ but after a certain number of Bellman Updates are done then agent reaches the goal state faster in case of MBPA in comparison to DYNAQ. It is important to note that computations performed in DYNAQ require very less computational power in comparison to computations required for MBPA but still MBPA converges faster.

3.3 Mountain-Car problem

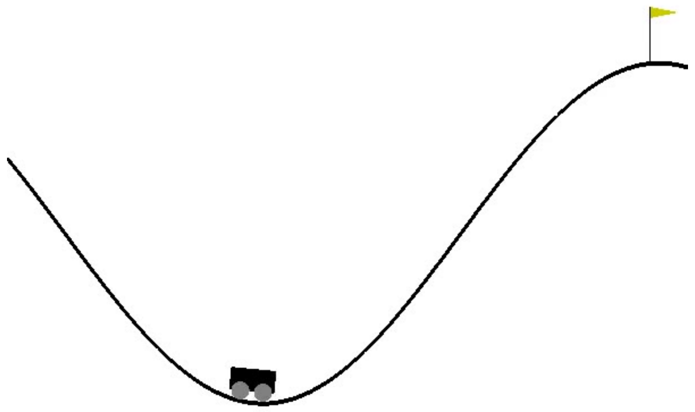
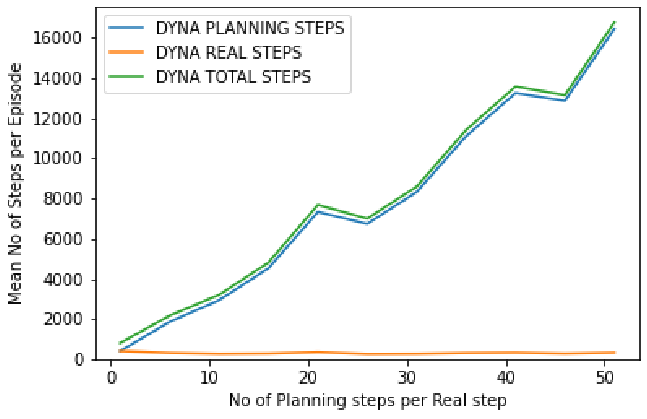


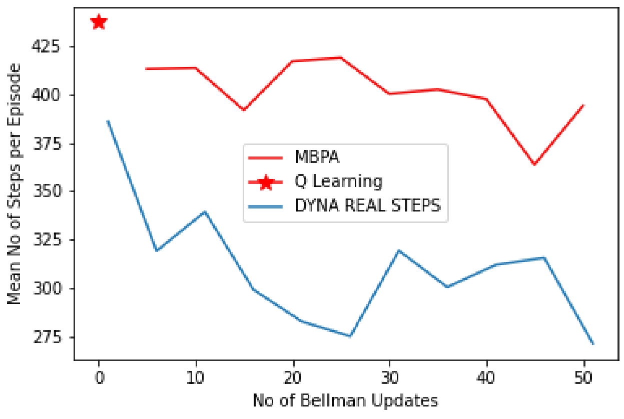
Figure 7: Mountain Car

In this experiment there is a standard mountain car environment.The agent begins from the valley and has to reach the goal state, agent needs to reach the flag placed at the top of right mountain. Agent must learn to choose proper direction of force to apply at each state to begin oscillation between two mountains to build momentum to reach the goal state. The state at each instant is position and velocities discretised. We used ϵ greedy decision policy for the Dyna and MBPA where $\epsilon = 0.01 \times 0.99^{i-1}$. For each step taken by agent in environment, if it reaches goal state then as reward 1 point is given and if not, then -1 point. This is done so agent learns a policy where it reaches the goal state in minimum number of steps,

3.3.1 Results



(a) No of steps for Dyna



(b) No of steps for DYNA AND MBPA.

Figure 8: Mountain Car Learning.

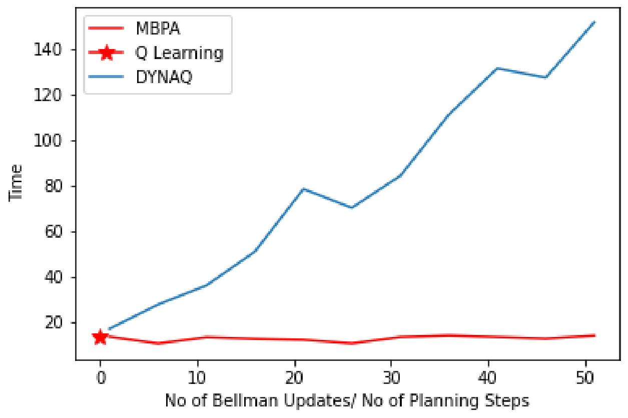


Figure 9: Time for Mountain Car

Figure 8a shows the mean no of steps required per episode to reach goal state by DYNAQ algorithm.
Figure 8b represents mean steps for Q Learning, MBPA and DYNAQ. As can be seen MBPA and DYNA both at start require less no of steps than Q Learning and DyaQ eventually drops to very small value where as MBPA more or less remains constant.
Figure 9 represents time required by Q Learning, DynaQ and MBPA in seconds. As can be seen initially all algorithms require almost same amount of time but as no of planning steps increase, so does time required by DynaQ. Whereas for MBPA, time required remains constant with respect to no of bellman updates.

3.4 Conclusion

We have seen that DYNAQ and MBPA algorithms are able to mimic the behavioural data of humans, that is, the Model Based and Model Free system cooperate with each other in certain situations. At the computational level of analysis it has been suggested that negotiating these two systems could be understood as a meta cognitive solution to a problem of resource rationality. MBPA is meta cognitive optimization which controls the trade off between the MB and MF learning. MBPA uses pseudo rewards to create a link between the MB and MF system. Results show that MBPA requires less computation than DYNAQ to converge at the optimal policy. This points to optimal resource rational solution. For human cognition we can easily draw parallel between pseudoreward and emotion, it has been observed that certain emotion almost always have the effect of changing the reward landscape. MBPA suggest that the emotions are biases that brain has learnt over past experience. These biases are used in model free learning to encourage or discourage certain actions. MBPA offers a new way to simplify complex, high-dimension environments into simpler strategies and MF heuristic-based decision.It could offer computationally cheaper way to provide knowledge to MF system through reward shaping.

References

[1] P. L. Lockwood, M. C. Klein-Fl"ugge, A. Abdurahman, and M. J. Crockett, "Model-free decision making is prioritized when learning to avoid harming others," *Proceedings of the National Academy of Sciences*, vol. 117, no. 44, pp. 27719–27730, 2020.

[2] P. M. Krueger and T. Griffiths, "Shaping model-free habits with model-based goals.," in *CogSci*, 2018.

[3] S. J. Gershman, A. B. Markman, and A. R. Otto, "Retrospective revaluation in sequential decision making: a tale of two systems.," *Journal of Experimental Psychology: General*, vol. 143, no. 1, p. 182, 2014.

[4] Y. Huang, Z. A. Yaple, and R. Yu, "Goal-oriented and habitual decisions: Neural signatures of model-based and model-free learning," *NeuroImage*, vol. 215, p. 116834, 2020.

[5] N. D. Daw, S. J. Gershman, B. Seymour, P. Dayan, and R. J. Dolan, "Model-based influences on humans' choices and striatal prediction errors," *Neuron*, vol. 69, no. 6, pp. 1204–1215, 2011.

[6] R. Bellman, *Dynamic programming.* Princeton, NJ: Princeton University Press.

[7] A. Dickinson, "Actions and habits: The development of behavioural autonomy," *Philosophical Transactions of the Royal Society: B*, vol. 308, p. 67–78.

[8] R. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Proceedings of the Seventh International Conference on Machine Learning* (B. Porter and R. Mooney, eds.), vol. 216, (San Francisco, CA), p. 224, Morgan Kaufmann.

[9] H. Aarts and A. Dijksterhuis, "Habits as knowledge structures: automaticity in goal-directed behavior," *Journal of personality and social psychology*, vol. 78, no. 1, p. 53.

[10] A. Barto, S. Bradtke, and S. Singh, "Learning to act using real-time dynamic programming," *Artificial Intelligence*, vol. 72, no. 1, p. 81–138.

[11] R. Dolan and P. Dayan, "Goals and habits in the brain," *Neuron*, vol. 80, no. 2, p. 312–325.

[12] S. Gershman, A. Markman, and A. Otto, "Retrospective revaluation in sequential decision making: A tale of two systems," *Journal of Experimental Psychology: General*, vol. 143, no. 1, p. 182.

[13] Q. Huys, N. Lally, P. Faulkner, N. Eshel, E. Seifritz, S. Gershman, and J. Roiser, "Interplay of approximate planning strategies," *Proceedings of the National Academy of Sciences*, vol. 112, no. 10, p. 3098–3103.

[14] H. McMahan, M. Likhachev, and G. Gordon, "Bounded real-time dynamic programming: Rtdp with monotone upper bounds and performance guarantees," in *Proceedings of the 22nd international conference on machine learning*, p. 569–576.

[15] K. Miller, A. Shenhav, and E. Ludvig, "Habits without values," *bioRxiv*, vol. 67603.

[16] A. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml (Vol*, vol. 99, p. 278–287.

[17] R. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *ACM SIGART Bulletin*, vol. 2, no. 4, p. 160–163.