Finding Optimal Strategies over Families of Tasks in Reinforcement Learning

Brown University Applied Math - Computer Science
Sc.B Honors Thesis

Advisor: Professor George Konidaris
Second Reader: Professor Michael L. Littman
Support: Graduate students David Abel and Yuu Jinnai

Yue (Sophie) Guo

May 2018

1 Introduction

1.1 Overview

In reinforcement learning, the transition and reward functions are typically unknown but stationary. The agent must interact with the environment to gather information about the unknown transition and reward functions to maximize overall reward.

However, there are some situations where the transition function is not fixed, and policies must be adapt to changes in transition functions. For example, in inventory management, ideally there are only fixed transition functions based on the fact of product's popularity, market size, strength of advertisement, and company's financial status etc, but this is not always successful because some accidents might occur such as a sudden natural disaster, political disturbance, or emergence of a more advanced product, leading to unpredictable results and a modified transition function. Reinforcement learning in delivery systems also has a similar issue when the environment becomes unpredictable, such as the deterioration of vehicles, health condition of the delivery man, distinct requirement on delivery and so on. In addition, when using reinforcement learning in power system to control solar panels, the transition functions are fixed until new panels are installed and heat energy is transferred to them. The transition function would also change based on new settings [1]. In these cases, it is more reasonable to model a sequence of Reinforcement Learning problems, each has a stationary transition function, to represent all possible situations.

When we are faced with a new problem with an unknown transition function, which is drawn from the same distribution that prior problems were drawn from, we want to speed learning by using knowledge of prior problems. We also want to focus on building the whole policy in advance, i.e., using past knowledge to compute (perhaps using a lot of computation time) a policy that we can then deploy in real life easily.

In order to explore, we further simplify the problem by assuming the changing transition functions are deterministic and of unknown but fixed distribution, with the reward function R unknown but fixed. We hope that through examining this situation, we could gain some insights into dealing with changing transition functions in real world. We also hope that by finding optimal strategies in this situation, we could further explore the situation when both R and T change.

1.2 Background

Reinforcement learning is an area in artificial intelligence concerned with how agents choose actions to maximize cumulative rewards. It is commonly formulated as a Markov Decision Process (MDP), composed of a set of states, a set of actions, a transition function, a reward function, and a discount factor.

An agent selects actions using a policy, which is a function that maps states to actions. The goal is to find a policy that maximizes reward in the term, which means we want to maximize the expected reward $\arg\max_{\pi\in\Pi} \mathbb{E}_{M\in\mathbb{M}}[V_T^\pi(s_0)]$ given the MDP, where π is the policy, M is an MDP, s_0 is the initial state, and $V_T^\pi(s_0)$ is the discounted sum of rewards received under π for s_0. The Bellman Equation is used to determine what reward is received under the policy π, using the

formula:

$$V_T^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathrm{S}} T(s, \pi(s), s') V_T^{\pi}(s')$$

, where $R(s, \pi(s))$ is the reward received for state s under policy π, γ is the discount factor, S is the set of all states, and $T(s, \pi(s), s')$ gives the probability that the state observed after executing $\pi(s)$ in state s is s'.

1.3 Previous Work Summary

This paper is an extension on the research that I conducted with graduate students David Abel, Yuu Jinnai, Prof. George Konidaris, and Prof. Michael Littman on the topic "which knowledge should be transferred in lifelong RL". In the work, I mainly contributed to the first family of results, the jumpstart policies, where we experimented with four agents exploring their relative performance over a distribution of tasks. The cases that we have conducted experiments on are where the goal and reward functions are varied, but we have not yet explored anything on the varied transition function as it is more complicated. Therefore, I form the undergraduate honors thesis on this topic in order to work on this specific question, and hope to provide some inspiration for the next step in our research.

2 Research Specification

2.1 What is the Goal

Given some fixed S, A, s_0, Π, a fixed but unknown R and a distribution D over deterministic transition functions, we would like to find the optimal strategy to maximize the expected reward over D, which means for each choice of policy space Π, we need to solve:

$$\arg\max_{\pi \in \Pi} \mathbb{E}_{T \in \mathbb{T}} \left[V_T^{\pi}(s_0) \right]. \tag{1}$$

We are especially interested in three classes of policy space Π:

1. Fixed Deterministic Policy Space, Π_d: $S \mapsto A$. Mapping each state $s \in S$ to a single action $a \in A$. The solution to a single MDP with a stationary transition function is always of this form.

2. Fixed Stochastic Policy Space, Π_s: $S \mapsto \Pr(A)$. Mapping each state $s \in S$ to a probability distribution of actions over A. Here we generalize the policy to a stochastic policy, a more powerful class which accounts for uncertainty over the transition functions.

3. Belief Space Policy Space, Π_b: $S \times \Pr(T) \mapsto A$. Mapping each state $s \in S$ to a single action $a \in A$ based on its current belief on the unknown probability distribution D over deterministic transition functions. This models the full learning setting, where the agent uses all the knowledge it has observed so far about the new T to make decisions. However, this model is computationally expensive.

2.2 Methods

We used proofs to find the optimal policies in the Fixed Deterministic Policy Space Π_d: $S \mapsto A$ and the Fixed Stochastic Policy Space, Π_s: $S \mapsto \Pr(A)$, and programmed experiments to find the optimal policy respectively in the Belief Space Policy Space, Π_b: $S \times \Pr(T) \mapsto A$.

2.3 Representation of Transition Functions

We represent the deterministic transition functions as a matrix for each action, with rows for input states and columns for output states. Denote the entire space of deterministic transition functions as $Space_T$. For entry of position (i, j) of matrix A_a in $Space_T$, the value is either 1 or 0, with 1 indicating the state i will transit into state j if we choose this action a, and 0 for not transiting into state j. Thus, the matrix is sparse as each row will only have one non-zero entry. D is the distribution of these matrices over $Space_T$.

T_{action_0}	s_0	s_1	s_2	...
s_0	0	1	0	...
s_1	0	0	1	...
s_2	1	0	0	...
...	...	...	...	...

Thus we should have this property:

$$\forall s_0 \in S, \forall T_k \in Space_T, \ \sum_{s_i} T_k(s_0, s_i) = 1 \text{ and } \max T_k(s_0, s_i) = 1$$

3 Theoretical Work

3.1 Fixed Deterministic Policy Space

In Fixed Deterministic Policy Space Π_d: $S \mapsto A$, we know each given state is mapped to only one action. As the transition function is deterministic, $T(s_k, \pi_d(s_k), s_{k+1})$ is simply an entry (s_k, s_{k+1}) in the transition matrix of the action $a = \pi_d(s_k)$, with value either 0 or 1.

As we want to maximize the expected reward, we would like to calculate the $\arg\max_{\pi_d \in \Pi_d} \mathbb{E}_{T \in \mathbb{T}} [V_T^{\pi_d}(s_0)]$ indicated in the previous section.

$$\arg\max_{\pi_d \in \Pi_d} \mathbb{E}_{T \in \mathbb{T}} [V_T^{\pi_d}(s_0)] = \arg\max_{\pi_d \in \Pi_d} \sum_{T \in \mathbb{T}} Pr(T) V_T^{\pi_d}(s_0)$$

Substituting the Bellman equation, $V_T^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathrm{S}} T(s, \pi(s), s') V_T^{\pi}(s')$, we have

$$\arg\max_{\pi_d \in \Pi_d} \sum_{T \in \mathbb{T}} Pr(T)[R(s_0, \pi_d(s_0)) + \gamma \sum_{s' \in S} T(s_0, \pi_d(s_0), s') V_T^{\pi_d}(s')]. \quad (2)$$

Since we are only considering deterministic transition functions, and for each pair of current state and action, there is only one state to be transitioned into. In the transition matrix, this is the only entry equals one among the entire row. We can define this target state to be

$$s_{i,T} := \arg\max_{s' \in S} T(s_{i-1,T}, \pi_d(s_{i-1,T}), s').$$

Note the $s_{i-1,T}$ is the previous state based on T, and as the start state is fixed, we have $s_{0,T} = s_0$. Plug in (2), we have

$$\arg\max_{\pi_d \in \Pi_d} \sum_{T \in \mathbb{T}} Pr(T)[R(s_0, \pi_d(s_0)) + \gamma T(s_0, \pi_d(s_0), s_{1,T}) V_T^{\pi_d}(s_{1,T})].$$

As we know the $T(s_0, \pi_d(s_0), s_{1,T})$ is either 0 or 1, and we only keep the transition entries that are 1, then we could plug in $T(s_0, \pi_d(s_0), s_{1,T}) = 1$, yielding

$$\arg\max_{\pi_d \in \Pi_d} \sum_{T \in \mathbb{T}} Pr(T)[R(s_0, \pi_d(s_0)) + \gamma V_T^{\pi_d}(s_{1,T})]$$

$$= \arg\max_{\pi_d \in \Pi_d} \sum_{T \in \mathbb{T}} Pr(T)[R(s_0, \pi_d(s_0)) + \gamma[R(s_{1,T}, \pi_d(s_{1,T})) + \gamma[R(s_{2,T}, \pi_d(s_{2,T})) + \gamma[...]]]]$$

$$= \arg\max_{\pi_d \in \Pi_d} \sum_{T \in \mathbb{T}} Pr(T) \sum_{s_0, s_{1,T} ... s_{inf,T}} \gamma^i R(s_{i,T}, \pi_d(s_{i,T})). \tag{3}$$

In addition, if we only have the reward of the goal state to be one, while all other states have zero reward, we could further have

$$\arg\max_{\pi_d \in \Pi_d} \sum_{T \in \mathbb{T}} Pr(T)\gamma^k$$

where k is the number of steps we need to get to the goal state from the initial state, and k depends on T. Therefore, to calculate the optimal policy in Fixed Deterministic Space, we only need to consider the probability distribution of transition functions, the discount factor, and the needed number of steps it takes from the initial state to the goal state for each transition function.

3.2 Fixed Stochastic Policy Space

In Fixed Stochastic Policy Space, Π_s: $S \mapsto \Pr(A)$, we know each given state is mapped to a probability distribution of actions. Thus $T(s_k, \pi_s(s_k), s_{k+1})$ is a probability distribution of entries (s_k, s_{k+1}) in the transition matrices of actions in the distribution $Pr(a) = \pi_s(s_k)$, with value either 0 or 1. Note that all the entries have the same position: s_kth row and s_{k+1}th column, and the probability that an entry is chosen is the same as the probability that the corresponding action is chosen.

Same as in the fixed deterministic policy space, we want to maximize the expected reward, and calculate the $\arg\max_{\pi_s \in \Pi_s} \mathbb{E}_{T \in \mathbb{T}}[V_T^{\pi_s}(s_0)]$ indicated in the previous section.

$$\arg\max_{\pi_s \in \Pi_s} \mathbb{E}_{T \in \mathbb{T}}[V_T^{\pi_s}(s_0)] = \arg\max_{\pi_s \in \Pi_s} \sum_{T \in \mathbb{T}} Pr(T) V_T^{\pi_s}(s_0)$$

Substituting the Bellman equation, $V_T^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathrm{S}} T(s, \pi(s), s') V_T^{\pi}(s')$, we have

$$\arg\max_{\pi_s \in \Pi_s} \sum_{T \in \mathbb{T}} Pr(T)[R(s_0, \pi_s(s_0)) + \gamma \sum_{s' \in S} T(s_0, \pi_s(s_0), s') V_T^{\pi_s}(s')]. \tag{4}$$

However, note that $\pi_s(s_0)$ is a distribution of actions, thus to calculate the expectation, we need to modify (4) into

$$\arg\max_{\pi_s \in \Pi_s} \sum_{T \in \mathbb{T}} Pr(T) \sum_{a \sim \pi_s(s_0)} Pr(a)[R(s_0, a) + \gamma \sum_{s' \in S} T(s_0, a, s') V_T^{\pi_s}(s')]. \tag{5}$$

Although we are only considering deterministic transition functions, the case is different from the fixed deterministic policy space because given a current state, there is a probability distribution of actions can be taken. Under the chosen action, there is only one state to be transitioned into. In the transition matrix of the chosen action, this resulting state is the only entry equals one among the entire row. Then we can define this target state to be

$$s_{i,T} := \arg\max_{s' \in S} T(s_{i-1,T}, a, s')$$

where $a \sim \pi_s(s_{i-1,T})$, and in addition, for $s_{i,T} \sim D^{s_{i,T}}$, the probability distribution of succeeding states at step i, $Pr(s_{i,T}) = Pr(a)$. Same as in the deterministic policy space, $s_{i-1,T}$ is the previous state based on T, and as the start state is fixed, we have $s_{0,T} = s_0$.

Plug in (5), we have

$$\arg\max_{\pi_s \in \Pi_s} \sum_{T \in \mathbb{T}} Pr(T) \sum_{a \sim \pi_s(s_0)} Pr(a)[R(s_0, a) + \gamma T(s_0, a, s_{1,T}) V_T^{\pi_s}(s_{1,T})].$$

Same as in the fixed deterministic policy space, the $T(s_0, \pi_d(s_0), s_{1,T})$ is either 0 or 1, and we only keep the transition entries that are 1, then we could plug in $T(s_0, \pi_d(s_0), s_{1,T}) = 1$, yielding

$$\arg\max_{\pi_s \in \Pi_s} \sum_{T \in \mathbb{T}} Pr(T) \sum_{a \sim \pi_s(s_0)} Pr(a)[R(s_0, a) + \gamma V_T^{\pi_s}(s_{1,T})]$$

$$= \arg\max_{\pi_s \in \Pi_s} \sum_{T \in \mathbb{T}} Pr(T) \sum_{a_{0,T} \sim \pi_s(s_0)} Pr(a_{0,T})[R(s_0, a_{0,T}) + \gamma[\sum_{a_{1,T} \sim \pi_s(s_{1,T})} Pr(a_{1,T})[R(s_{1,T}, a_{1,T}) + \gamma[...]]]]$$

$$= \arg\max_{\pi_s \in \Pi_s} \sum_{T \in \mathbb{T}} Pr(T) \sum_{[s_0, s_{1,T} ... s_{inf,T}] \in \text{all possible paths}} \sum_{a_{i,T} \sim \pi_s(s_i)} [\Pi_{a_{0,T}, a_{1,T} .... a_{i,T}} Pr(a_{i,T})] \gamma^i R(s_{i,T}, a_{i,T}). \tag{6}$$

Therefore, it is clear that finding the optimal policy in the fixed stochastic policy space is more complicated than doing that in the fixed deterministic policy space. Even if we only consider the simplified case where we have reward one at the goal state while zero at all the other states, it requires multiplying the probability of each action along the path from the initial state to the goal state, and summing the product for all possible paths.

4 Experimental Work

4.1 Experiment Design

4.1.1 Constructing MDPs

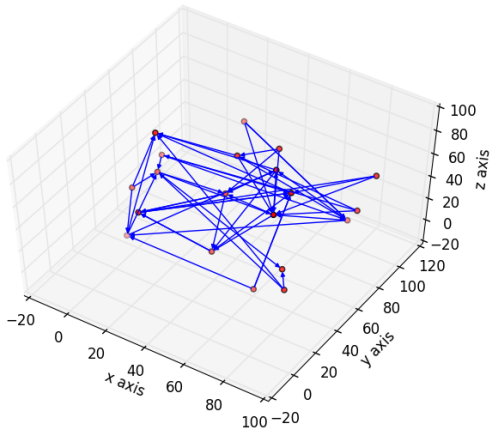
We want to construct MDPs that satisfy the following constraints in order to perform experiments on a distribution of MDPs with a distribution of deterministic transition matrices:

- Given any action and state, it is reasonable to have a transition to any state in the state space, though we will only pick one state to be transitioned into.
- It is easy to generate a distribution of transition functions, and all the generated transition functions are applicable and utilizable. In addition, it is easy to calculate the probability that transition function appears in the distribution.
- Use a reward function that only has a reward to be one in the goal state, and zero reward for all other states, in order to simplify the model. Thus the reward function is only associate with which state we choose as the goal state.

Therefore, we aim at avoiding all irrelevant parameters and context, and could construct our MDPs as follows:

- States: Represented simply as integers, as their indices.
- Start State: Randomly pick a non-absorbing state to be the starting state.
- Actions: Represented simply as integers, as their indices.
- Reward Function: 1 at goal state, 0 at other states. The goal state is fixed.
- Discounting Rate: A constant, γ
- Transition Function: A list of matrices, with each matrix corresponding to the action of the same index. The matrices representation were previously described in section 2.3 - Representation of transition functions.

To visualize this construction, we could represent each state as a node, and the directed lines indicate the deterministic transition functions. For any $s_1 \rightarrow s_2$, it means under this action, the agent will reach state s_2 from state s_1. It is possible to reach the same state given different actions, and to have some absorbing states as well. Thus, not all the starting states are guaranteed to have a path towards the goal

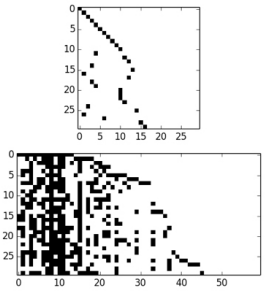


state.

In order to view the construction more clearly, we plot the states and their transitions in 3d as follows. The arrows indicate the possible transitions between states. Here is a sample with 20 states and 3 actions.

4.1.2 Generating Transition Matrices

In order to generate a distribution of MDPs with a distribution of transition matrices, we need to find a method to generate transition matrices. Both the Chinese Restaurant Process (CRP) and Indian Buffet Process (IBP) can model the distributions of matrices because the matrices generated are all composed only of zeros and ones. For each row in the matrix, both CRP and IBP pick the entry(s) to be one based on the ones already picked in previous rows. All the other entries in the row are remained to be zero. This works with our target, as our transition matrices are deterministic, and thus only have zeros and ones in them.



However, we choose CRP instead of IBP because it guarantees to have only one entry for each row to be one, and the entry picked will be within the nth column of the matrices (n is the number of states). As matrices of our transition functions are n-by-n matrices, this property is very important. For IBP, we cannot make sure the entries picked are within the nth column of the matrices, and thus our transition functions could be invalid. Moreover, it has multiple entries for each row to be ones, so it is better to model matches of stochastic transition functions, instead of deterministic ones. So we hope we could keep IBP for future generalization.

To generate transition matrices in CRP, we pick the first entry of the first row to be one. Then for the ith row, we pick the first unoccupied column with probability $\frac{\alpha}{n-1+\alpha}$, and an occupied column with probability $\frac{c}{n-1+\alpha}$, where c is the number of entries that are one in this column.

The graphs above are sampled CRP and IBP transition matrices that we have generated. (1st is CRP, 2nd is IBP, n=30)

4.1.3 Control the Level of "Known"

We want to control how much we know about the transition function selected as well. To deal with this, we fix an original transition function generated by the Chinese Restaurant Process and call it the source matrix. We then pick i columns of a transition function to change under the given distribution. i ranges from 0 to n, where n is the number of states. Thus, if $i = 0$, then we have all the information about the transition function, and if $i = n$, then the transition function is

completely unknown. Based on that, we compare results for $i = 0$, $i = 0.1n$, $i = 0.2n$, $i = 0.3n$, $i = 0.4n$, $i = 0.5n$, $i = 0.6n$, $i = 0.7n$, $i = 0.8n$, $i = 0.9n$, $i = n$, gradually from a completely known transition function to a completely unknown transition function. As each action should have a different transition function, we assume their independence and generate transition functions separately for each of them, and fix i for all the actions.

We would like to compare the performances of different agents taking i from 0 to n. Intuitively, as we are gradually getting more "unknown" to the distribution, the performances of all agents should get worse and worse. We will discuss the actual experiment work in the next section soon.

4.1.4 Predictions

As this work is an extension for the research on "What to Transfer in Lifelong Reinforcement Learning", we are using the same agents and compare their performances.

- π_{avg}: The optimal policy in the average MDP.
- π_{prior}: The action-prior policy, which stochastically chooses actions according to the probability distribution over action optimality: $\pi_{prior}(a \mid s) = \Pr_{M \sim D}\left(a = \arg\max_a Q^*_M(s, a) \mid s\right)$.
- π^u : The uniform random policy.
- π^*_b: The optimal belief-space policy—the policy that maximizes value over the Belief MDP defined by the MDP distribution.

The optimal belief-space agent, Π^*_b, should have the optimal behavior across the distribution of MDPs, though it is computationally difficult to solve the belief space. This agent should perform as our upper bound among the four agents.

π^u, the uniform random policy should perform the worst, as it does not take in too much about the information about the distribution of MDPs, but simply explore randomly.

It is reasonable to predict that the action-prior policy would perform better than the optimal policy in the average MDP because it might not be meaningful to calculate the average MDP. For example, if we have two transition matrices that are vertically symmetric, would our average transition matrices be the matrices that have entries in the central column all be ones? Indeed, this averaged MDP might be meaningless, and we doubt whether it is utilizing the information it receives correctly.

4.2 Experiment Results

4.2.1 Settings

In this experiment, for each level of "unknown" i, (i =0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90), we sampled MDPs from a distribution of MDPs, and run each policy for 100 steps. The MDPs are constructed as described in section 4.1.1 and section 4.1.3. The parameters we set for MDPs are 100 states, 4 actions, $\gamma = 0.99$. The concentration parameter we use in the Chinese Restaurant Process is 20.

We varied the number of goals, the distribution size, and the number of samples drawn based on the distribution size. The three experiments we conducted are (1) 2 goals, 150 samples, 30 MDPs in the distribution, (2) 2 goals, 300 samples, 50 MDPs in the distribution, and (3) 3 goals, 150 samples, 30 MDPs in the distribution. The reason we are using multiple goals is that our transition matrices are sparse, and thus it is very easy for exchanging rows to make the MDP unsolvable. If we are using stochastic transition matrices, and our transition matrices are not sparse, 1 goal is also reasonable.

This means, for each i, we first use the Chinese Restaurant Process to generate 4 sparse matrices (size $100*100$) to be our source transition matrices, and additionally we guarantee it is solvable and needs at least 2 or 3 steps from the initial state to the goal state, so that the MDP is hard enough as not to be solved by luck immediately. Then we create 30 or 50 copies of the source transition matrices, for each copy, for each of the 4 matrices, randomly pick i rows of the matrix. Based on the property that the customers are exchangeable in one Chinese Restaurant process, we exchange the rows randomly for all the selected rows. These 30 or 50 transition matrices form our distribution of transition matrices. Then we initialize the starting state based on the transition matrices created, by randomly picking a non-absorbing state. As other parameters are not changed, this forms our distribution of MDPs for i.

The goal states and the initial state are fixed for each family of transition matrices that is generated by the source transition matrices, as they are generated randomly once, and fixed among all the MDPs in the distribution. For the reward function, we only have the reward equals one at the goal state, while zero reward for all the other states. Our reward function is therefore fixed.

Thus, we have a distribution of MDPs for each level of "unknown" i, and we want to plot for each step, the accumulative rewards of each agent with respect to i.

4.2.2 Result Graphs

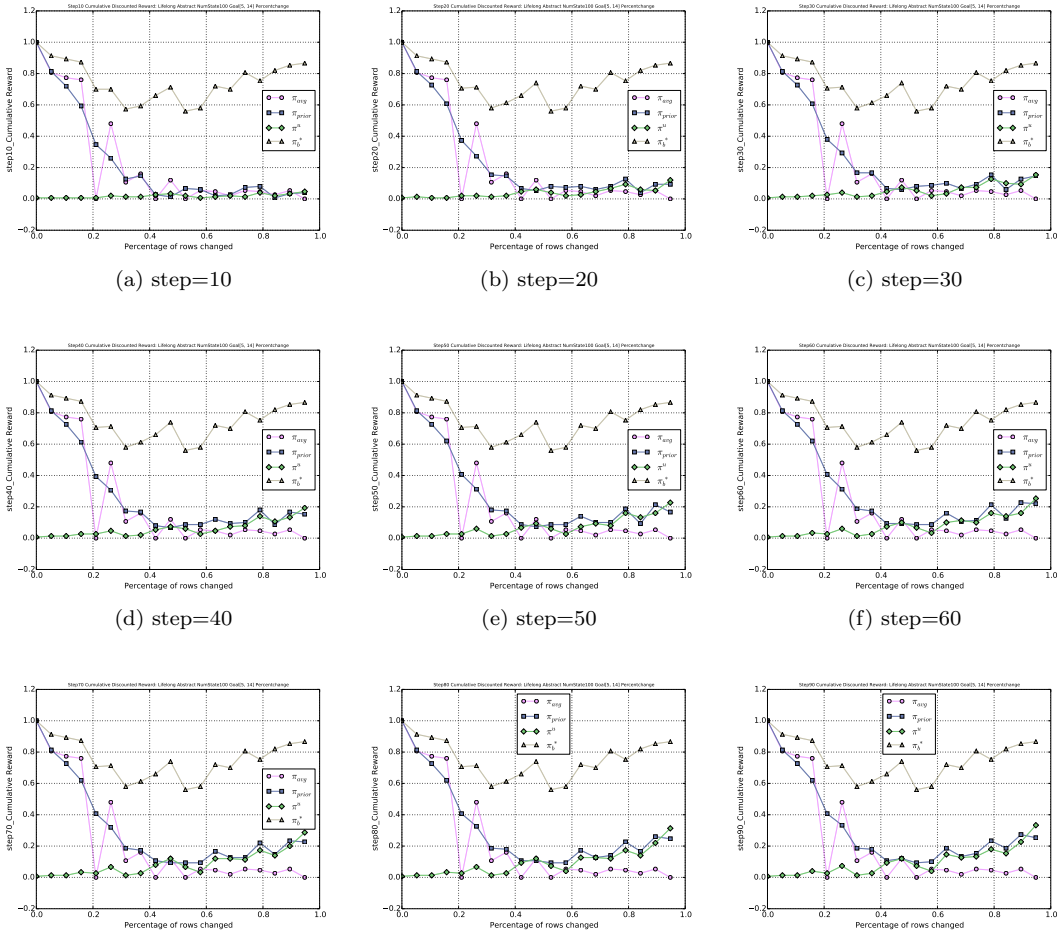
Below are the graphs of performances that the four agents have in the three experiments. For each experiment, there are nine graphs, arranged from step 10 to step 90 out of the 100 steps. The y-axis is the cumulative reward averaged among the samples drawn, and the x-axis is the percentage of rows that are unknown to the agents, which means it is our level of "unknown".

Indeed, for each sample and for each agent, reward of each single step is recorded, and these graphs are created based on the data recorded on file, instead of created simultaneously with the experiments. In addition, due to randomness, we do not have the same graphs every time, but the results here are typical. We will discuss the general trend in observation in the next discussion section.

Note that here we are using percentage of rows changed, which means the percentage of rows that are exchanged from the source matrices.

4.2.3 Results Discussions

In general, the performances of all agents are getting better and better from step = 10 to step = 90, which is intuitively correct because in some samples, all agents need to have longer steps in order

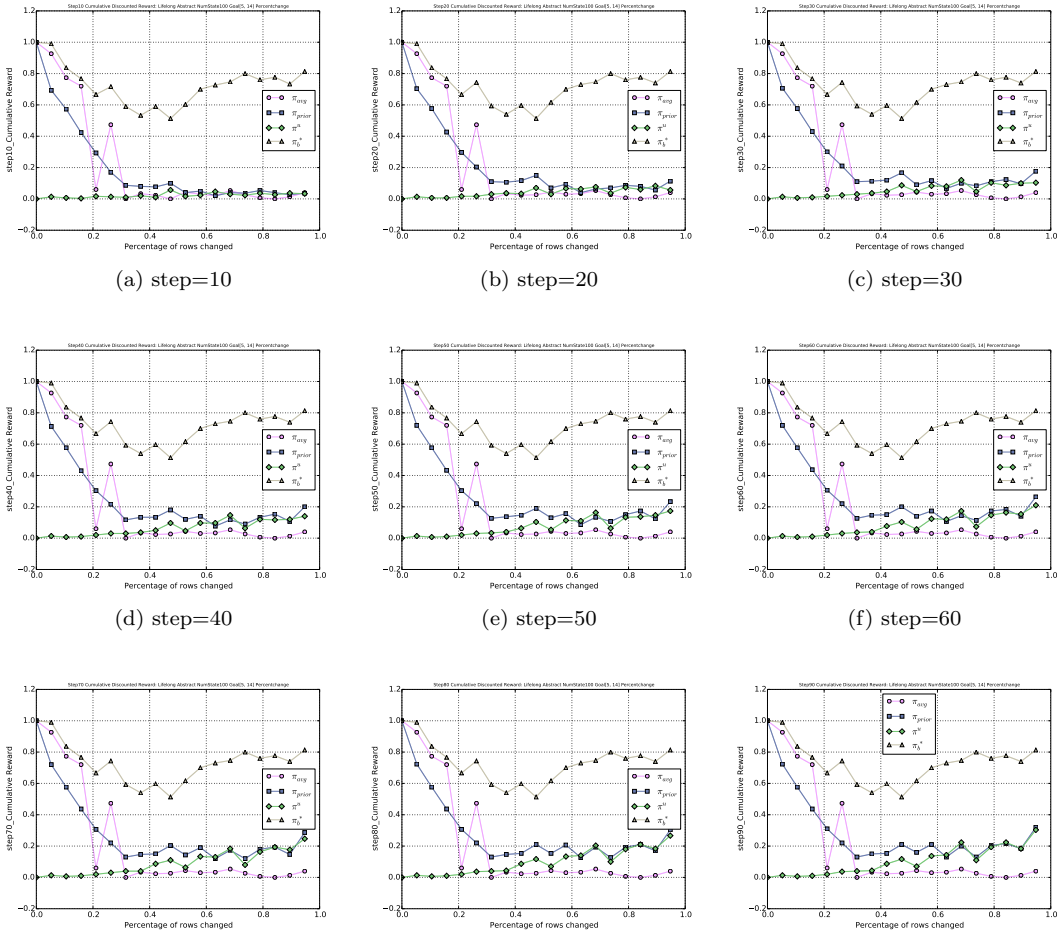


(a) step=10 (b) step=20 (c) step=30

(d) step=40 (e) step=50 (f) step=60

(g) step=70 (h) step=80 (i) step=90

Figure 1: Cumulative Rewards with respect to level of "unknown", 2 goals and 150 samples, from a distribution of 30 MDPs

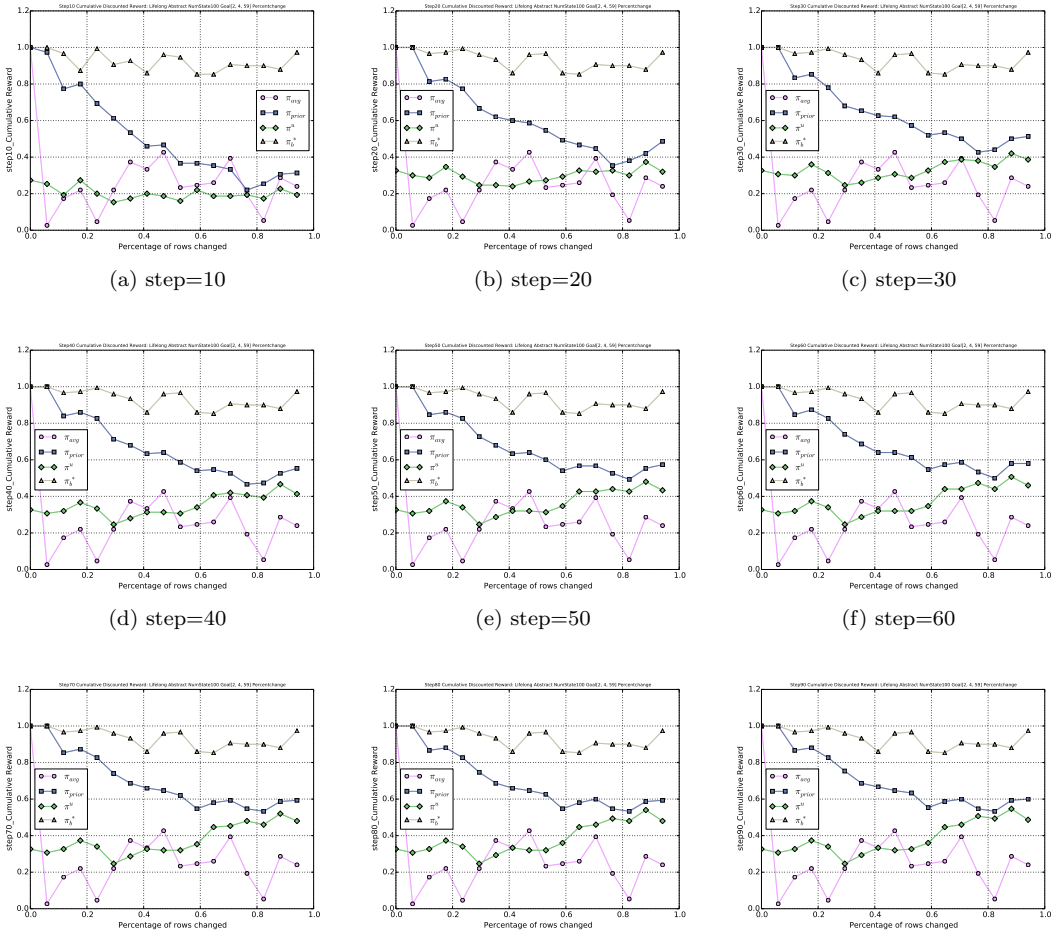


(a) step=10 (b) step=20 (c) step=30

(d) step=40 (e) step=50 (f) step=60

(g) step=70 (h) step=80 (i) step=90

Figure 2: Cumulative Rewards with respect to level of "unknown", 2 goals and 300 samples, from a distribution of 50 MDPs



(a) step=10 (b) step=20 (c) step=30

(d) step=40 (e) step=50 (f) step=60

(g) step=70 (h) step=80 (i) step=90

Figure 3: Cumulative Rewards with respect to level of "unknown", 3 goals and 150 samples, from a distribution of 30 MDPs

to get to the goal state by solving hard MDPs, while in a few lucky samples, it is easier to solve the MDPs. Thus we can see a growing cumulative rewards as the general trend from the first graph to the ninth for all the three experiments. However, the cumulative rewards for some agents stops increasing at relatively earlier steps, while the others might keep increasing until relatively later.

Besides, the performances of all agents become worse when we exchange more rows from the source matrices for all the three experiments. This indicates the distribution of MDPs become harder for the agents to solve when it becomes more and more "unknown". However, for each agent, the performances vary even we could be confident on the general trend. We will discuss the performances of the four different agents separately as follows.

Among all the three experiments, the belief agent is able to reach the goal state the fastest, and has the highest cumulative reward among all the agents. Although it takes long to learn, the belief agent is definitely the best, which corresponds to our prior expectations. In addition, it is very stable from step 10 to step 90, because it reaches the goal state very early if it could, and thus not changed a lot even it is given more steps to explore. Besides, depending on the hardness of the MDP distribution, its cumulative reward is not stable with respect to the percentage of rows exchanged, especially for experiments with 2 goals (150 samples from a distribution of 30 MDPs and 300 samples from a distribution of 50 MDPs), as it forms a valley shape. While for the experiment with 3 goals, it oscillates between 0.8 and 1.0.

The performance of the action prior agent in general becomes worse when the percentage of rows increases. When there is no row changing, it has the same performance as the belief agent, achieving 1.0 cumulative rewards very quickly. However, the cumulative rewards gradually decrease while more and more rows are exchanged from the source transition matrices. In addition, the action prior agent tends to achieve the goal states more slowly especially when more rows are changed, as the performance is getting better if we observe at the bigger time steps. This result makes sense because we expect the distribution becomes more unknown to the agent when more rows are exchanged, and thus it should take longer time for the agent to learn and the performance should become worse. Although the action prior agent is sometimes outperformed by the average MDP agent and the uniform random agent, especially when small number of row exchanges outperformed by the average MDP agent, and when big number of row exchanged outperformed by the uniform random agent, the action prior agent in general is the second best. In addition, when there are many rows exchanging ($i = 0.7 \sim 0.9$), the performance is not strictly getting worse, as it oscillates a little, which might be due to the hardness of the distribution.

The average MDP agent is not performing well compared to our prediction. When there is no row changing, it has the same performance as the belief agent and the action prior agent, but when more and more rows change from the source transition matrices, it is not stable, and is sometimes even outperformed by the uniform random agent. The most interesting thing is this agent does not improve when the time step gets bigger, which means it sticks to the existing policy at an early stage, not having any progress in updating afterwards. There are multiple times that it almost hits 0.0 as its cumulative rewards, indicating it is very hard for the agent to find the goal state at all no matter how many times it tries. The possible reason behind this might be the averaged deterministic transition matrices. As there is only one entry to be 1 in each row of the transition matrices, when calculating the average, it is most reasonable to pick the entry that is most commonly valued

1 for each row among all the transition matrices in the distribution. However, as the matrices are very sparse, this could direct the agent into a completely wrong transition if the entry fails to be the picked correctly. We would expect an improvement in the stochastic transition matrices, but not for the deterministic ones.

The uniform random agent in general has low cumulative rewards no matter how many rows have been changed from the original source matrices. However, the rewards grow when the steps get bigger. This means the agent is learning, and able to find the goal states sometime, but not as fast as other agents. For the cases that lots of rows have been exchanged from the original source matrices, when given lots of steps, the uniform random agent could even surprisingly be almost as good as the action prior agent. This is because when we are generating the source matrices and exchanging rows in making the distributions, we have sure all the MDPs are solvable for our fixed initial state and goal states. Thus, it is possible that the uniform random agent could perform better if we have bigger time steps. It is also interesting that when more rows are exchanged, the uniform random agent tens to have better performances. The only possible reason to explain would be as more rows are random, the transitions are not fixed, so getting into absorbing state is less possible if we randomly pick one action, however, there is no solid proof to support this guess.

Although the three experiments have almost the same result in terms of general trend increasing or decreasing in performances, there is still variance between the three different settings. Recall that our three settings are (1) 2 goals, 150 samples, 30 MDPs in the distribution, (2) 2 goals, 300 samples, 50 MDPs in the distribution, and (3) 3 goals, 150 samples, 30 MDPs in the distribution. We would like to discuss the variance in the results of three settings as follows.

By comparing (1) 2 goals, 150 samples, 30 MDPs in the distribution, and (2) 2 goals, 300 samples, 50 MDPs in the distribution, we could gain some insights in the larger sample distribution sizes. These two experiments are using the same source transition function, the same initial state and goal states. The performance of average MDP agent has only very slight difference. This makes sense because simply expanding the distribution and sample size would not change the average MDP, as the only average needs to be calculated is the average transition matrices, and this should not change. The belief agent has the same "shape" in performances, but the position of the valley-like portion is different, which might be subject to the hardness of the distributions constructed. However, the larger sample and distribution size cause the belief agent to have more smooth (less zig-zag) performances with respect to the percentage of rows exchange. The action prior agent has relatively better performance when there are more samples and the distribution is larger. When more rows are exchanged, the performance both oscillates in the two experiments, but in general, with more samples and larger distribution, the performance is flatter in trend. The uniform random agent has almost the same performance in the two experiments, and seems only to have fewer oscillations with larger sample and distribution size. Therefore, increasing the sample and distribution size could result in a more smooth performance, but the basic shape is not changed.

If we compare (1) 2 goals, 150 samples, 30 MDPs in the distribution, and (3) 3 goals, 150 samples, 30 MDPs in the distribution, we will understand how increasing the number of goals could affect the performances. If there are more goals in an MDP, then it means the MDP is easier to solve as reaching either goal state will give back the same reward (=1). Thus ideally all the agents should have better performance if there are more goals. The belief agent, action prior agent, and

the uniform random agent are all having better performances. Additionally, the action prior agent behaves dominantly better than the uniform random agent no matter how many rows are changed or time step increases, while it can be caught up if the number of goals decreases. The valley-like shape in the performance of the belief agent is also flatter, thus makes it more uniform. Although the average MDP agent has a better performance as well, it could still hit near-zero cumulative reward occasionally. However, when it is not hitting zero, the cumulative reward increases when there are more goals. Therefore, increasing the number of goals would definitely make solving the MDPs easier, but the basic shape is still preserved.

Through discussing the settings and results in the three experiments, we could confirm that the overall performance of the four agents are indeed stable among the changing in transition matrices. It is certain that these results are only very small portion of the experiments that we repeat, but they are typical results and worth analyzing. Based on these, we are able to gain some knowledge on the cumulative rewards achieved by the four agents.

5 Conclusion

5.1 Work Summary

In this paper, we have explored what is the optimal policy for the three policy spaces: fixed deterministic, fixed stochastic, and belief space. The first two are purely theoretical work, and we conduct experiments for the belief space, which is the main portion of this paper.

We have provided two optimal policies as well as their proofs for finding the optimal policy to solve the fixed deterministic and fixed stochastic spaces. The theoretical results are very straightforward and clean in the final format, as we have applied the property of deterministic transition function into the proofs.

For the experiments part, we provided the entire design and the results, along with the discussions, to explore the optimal policy for the belief space. We have run three experiments of different settings in order to make comparison and control the parameters of distribution size, sample size, and goal state numbers. We have also utilized a parameter, the percentage of rows exchanged, to indicate how far the distribution of transition matrices is from our source transition matrices. From the discussion, we analyzed the performances of all the agents separately, and how parameters change could affect the experiment. In general, the results corresponds with our original discussion, that the belief agent performs the best among the all four agents we used, and as the percentage of rows exchanging increases, we expect the agents to learn more slowly and have relatively worse results. The experimental results are the graphs for each step plotting the performances (cumulative rewards) with respect to the percentage of rows change, and they are in the section of result graphs.

This paper explores the specific question: what is the optimal strategies over distributions of deterministic transition functions in reinforcement learning. We hope the solution provided in this paper could not only pave the road for exploring unknown transition functions from fixed distributions, but could also inspire the research work in the next stage to explore the case where all the variables are not fixed in lifelong reinforcement learning.

5.2 Future Extensions

There remain lots of potential future explorations. One of them is to work on the distributions of non-deterministic transition functions. Then it could be reasonable to use the Indian Buffet Process (IBP) to generate transition matrices instead of using the Chinese Restaurant Process (CRP) as there could be more than one possible state to be transitioned into given the current state under the transition function. In addition, the calculation in average transition function would probably make more sense as we could calculate the mean of the probability that a state could be transitioned into, while not restricted to only picking one entry to be one for each row in the transition matrices.

Implementing more agents in the experiment part is also one of the possible future explorations, as the four agents we chose is only very few among all possible agents available. In addition, it would be nice to have the agents calculated in the theoretical work run as well, though they are only optimal in their own policy space, fixed deterministic policy space and fixed stochastic policy space, while in the experiment we are testing is the belief space policy space. It would be exciting to see how different optimal agents in different policy spaces behave.

It would also be possible to explore other meaningful tasks, as for now the task setting is very simple in order to get rid of all irrelevant factors. However, there should be more complicated models utilizing a distribution of deterministic transition functions that fit into or beyond the scope of this paper. We hope this paper could provide some insights for the potential extensions as well.

6 Resources and Acknowledgement

[1] Abel, David and Williams, Edward C. and Brawner, Stephen and Reif, Emily and Littman, Michael. Bandit-Based Solar Panel Control, Innovative Applications of Artificial Intelligence, 2018.

[2] Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a Unified Theory of State Abstraction for MDPs, ISAIM, 2006.

[3] Abel, David and Hershkowitz, David and Littman, Michael. Near Optimal Behavior via Approximate State Abstraction, International Conference on Machine Learning 2915–2923, 2016.

[4] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning, Artificial Intelligence 112, 1998.

[5] George Konidaris and Andrew Barto. Building Portable Options: Skill Transfer in Reinforcement Learning, IJCAI, 2007.

[6] Thomas G. Dietterich. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition, Journal of Artificial Intelligence Research 13, 2000.

[7] George Konidaris. Constructing Abstraction Hierarchies Using a Skill-Symbol Loop, IJCAI, 2016.

[8] Nakul Gopalan, Marie desJardins, Michael L. Littman, James MacGlashan, Shawn Squire, Stefanie Tellex, John Winder, and Lawson L.S. Wong. Planning with Abstract Markov Decision Processes, 2017.

[9] Nicholas K. Jong, Todd Hester, and Peter Stone. The Utility of Temporal Abstraction in Reinforcement Learning, AAMAS, 2008.

[10] Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. A Laplacian Framework for Option Discovery in Reinforcement Learning, arXiv, 2017.

[11] George Konidaris and Andrew Barto. Skill Discovery in Continuous Reinforcement Learning Domains using Skill Chaining, NIPS, 2009.

[12] Diederik M. Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A Survey of Multi-Objective Sequential Decision-Making, Journal of Artificial Intelligence Research 48, 2013, 67-113.

[13] Susanne Still and Doina Precup. An information-theoretic approach to curiosity-driven reinforcement learning, 2012.

[14] Jan Storck, Sepp Hochreitery, and Jurgen Schmidhuber. Reinforcement Driven Information Acquisition In Non-deterministic Environments, In Proc. ICANN'95, p. 159-164.

[15] Shakir Mohamed and Danilo J. Rezende. Variational Information Maximisation for Intrinsically Motivated Reinforcement Learning, NIPS, 2015.

[16] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method, arXiv, 2000.

[17] Michael Kearns and Satinder Singh. Near-Optimal Reinforcement Learning in Polynomial Time, Machine Learning, 49, 209–232, 2002.