Notes for chapter 5: Dynamic Programming

Introduction and Terminology

- MDP Control Problem: Compute the Optimal Value Function
- MDP Prediction Problem: Compute the Value Function when the Agent bases it's decision on a fixed policy. That is, compute the Value Function of the implied MRP.

Planning Algorithms
Algorithms with access to the MDP model environment are known as planning algorthms. This is to reflect the fact that the AI needs to interact with the environment and projects probabilistic scenario's of future states/rewards for various actions and solves for the Value Function based on these projected outcomes.

Principle of Optimality
An Optimal Policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions.
I.e in order to achieve optimality we need to act optimal at all future states resulting from the current state.

We will use the term Dynamic Programming in the narrow context of planning algorithms that solve the MDP Prediction Problem. And have the following 2 properties:

1. The state space is finite, the action space is finite and the set of pairs of the of next state and reward is also finite.
2. We have explicit knowledge of model specifications (either $\mathcal{P}_R$ or both $\mathcal{P}$ and $\mathcal{R}$)

In this section Dynamic Programming algorithms solve both MDP Prediction and MDP Control problems exactly. Meaning that as iterations increase the Value Function converges to the true Value Function.

Fixed-Point Theory

Each of the 3 algorithms we're going to cover is based on Fixed-Point theory and updates the computed Value Function towards the fixed point.

Definition (Fixed Point)**.** The Fixed Point of a function $f : \mathcal{X} \to \mathcal{X}$ is a value $x \in \mathcal{X}$ that statisfies $x = f(x)$.

Theorem (Banach Fixed-Point Theory)**.** Let $\mathcal{X}$ be a non-empty set equipped with a complete metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Let $f : \mathcal{X} \to \mathcal{X}$ be a function s.t. $\exists L \in [0, 1)$ s.t. $d(f(x_1), f(x_2)) \leq Ld(x_1, x_2)$, $\forall x_1, x_2 \in \mathcal{X}$ (this property is called contraction, we shall refer to f as a contraction function). Then,

1. There exists an unique Fixed Point $x^* \in \mathcal{X}$, i.e $x^* = f(x^*)$.
2. $\forall x_0 \in \mathcal{X}$, and sequence $[x_i | i = 0, 1, ...]$ defined as $x_{i+1} = f(x_i)$, $\lim\limits_{i \to \infty} x_i = x^*$.
3. $d(x^*, x_i) \leq \frac{L^i}{1-L} d(x_1, x_0)$

The theorem basically says that for any contraction function f there not only exists an unique fixed point x^* but one can arrive at x^* by repeated application of f from any starting point x_0.

$$f(f(...f(x_0)...)) \to x^*$$

Define $f^i : \mathcal{X} \to \mathcal{X},\ i = 0, 1, ...$ as:

$$f^{i+1}(x) = f(f^i(x))$$

In this notation the fixed point computation can be written as:

$$\lim_{i \to \infty} f^i(x) = x^*,\ \forall x \in \mathcal{X}$$

The algorithm is:

$$x_{i+1} = f(x_i),\ \forall i = 0, 1, ...$$

Bellman Policy Operator and Policy Evaluation Algorithm

The Policy Evaluation Algorithm solves the problem of computing the Value Functionn of a Finite MDP with fixed policy π. We want to find a numerical algorithm that would solve the MRP Bellman Equation:

$$V^\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi V^\pi$$

Where $\mathcal{R}^\pi \in \mathbb{R}^m$ is a column vector and $\mathcal{P}^\pi$ is a $m \times m$ matrix.

Definition (Bellman Policy Operator)**.** $B^\pi(V) = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi V$

So the MRP Bellman equation can be expressed as $V^\pi = B^\pi(V^\pi)$. Which means V^π is a Fixed Point of $B^\pi : \mathbb{R}^m \to \mathbb{R}^m$. Note that B^π is just a linear transformation.

Now we want to find some complete metric $d : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ s.t. we can show that B^π is contraction function. Because then we found our algorithm by the Banach Fixed-Point Theorem.

For any $V \in \mathbb{R}^m$, we shall express the value for any state $s \in \mathcal{N}$ as $V(s)$. Our metric $d : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ shall be the L^∞ norm:

$$d(X, Y) = ||X - Y||_\infty = \max_{s \in \mathcal{N}} |(X - Y)(s)|$$

B^π is a contraction function under the L^∞ norm, because for all $X, Y \in \mathbb{R}^m$:

$$\begin{aligned} d(B^\pi(X), B^\pi(Y)) &= \max_{s \in \mathcal{N}} |(B^\pi(X) - B^\pi(Y))(s)| \\ &= \max_{s \in \mathcal{N}} |(\mathcal{R}^\pi + \gamma \mathcal{P}^\pi X - \mathcal{R}^\pi - \gamma \mathcal{P}^\pi Y)(s)| \\ &= \gamma \max_{s \in \mathcal{N}} |\mathcal{P}^\pi (X - Y)(s)| \\ &\leq \gamma \max_{s \in \mathcal{N}} |(X - Y)(s)|,\ \text{since } \mathcal{P}^\pi : \mathcal{N} \times \mathcal{S} \to [0, 1] \end{aligned}$$

So invoking the the Banach Fixed-Point Theory proves the following theorem:

Theorem (Policy Evaluation Convergence Theorem)**.** For a Finite MDP with $|\mathcal{N}| = m$, and $\gamma < 1$ if $V^\pi \in \mathbb{R}^m$ is the value function of the MDP. When evaluated with fixed policy $\pi : \mathcal{N} \times \mathcal{A} \to [0, 1]$ then V^π is the fixed point of the Bellman Policy Operator and can be computed by repeated applications of the Bellman Policy Operator. That is,

$$\lim_{i \to \infty} (B^\pi)^i(V_0) \to V^\pi,\ \text{for all starting Value Functions } V_0$$

Greedy Policy

This section is a stepping stone from Prediction to Control problems. We'll define a function that will improve a Value Function or improve a policy with a greedy technique.

Formally, the Greedy Policy Function

$$G : \mathbb{R}^m \to (\mathcal{N} \to \mathcal{A})$$

is a function mapping a value function V to a deterministic policy $\pi'_D : \mathcal{N} \to \mathcal{A}$. Defined as:

Definition (Greedy Policy Function)**.**

$$G(V)(s) = \pi'_D(s) = \arg\max_{a \in \mathcal{A}} \{\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{N}} \mathcal{P}(s,a,s')V(s')\}, \; \forall s \in \mathcal{N}$$

If 2 or more actions achieve maximization then we use an arbitrary rule assigning 1 to the output. Since our code for Finite MDP's uses $\mathcal{P}_R$ instead of $\mathcal{P}$ and $\mathcal{R}$ we can use the following equivalent expression for G:

$$G(V)(s) = \arg\max_{a \in \mathcal{A}} \{\sum_{s' \in \mathcal{N}} \sum_{r \in \mathcal{D}} \mathcal{P}_R(s,a,r,s')(r + \gamma W(s'))\}, \; \forall s \in \mathcal{N}$$

Where,

$$W(s') = \begin{cases} V(s') & \text{, if } s' \in \mathcal{N} \\ 0 & \text{, if } s' \in \mathcal{T} = \mathcal{S} - \mathcal{N} \end{cases}$$

Policy Improvement

Definition (Value Function Comparison)**.** We say $X \geq Y$ for Value Functions $X, Y : \mathcal{N} \to \mathbb{R}$ if and only if

$$X(s) \geq Y(s), \; \forall s \in \mathcal{N}.$$

That is, a value function is better then another one if and only if it's no worse then the other one.

Theorem (Policy Improvement Theorem)**.** For a Finite MDP, for any policy π

$$V^{\pi'_D} = V^{G(V^\pi)} \geq V^\pi$$

Proof. It follows directly from the Policy Iteration Theorem that $\lim_{i \to \infty} (B^{\pi'_D})^i(V^\pi) = V^{\pi'_D}$. So the proof is complete if we prove that:

$$(B^{\pi'_D})^{i+1}(V^\pi) \geq (B^{\pi'_D})^i(V^\pi)$$

Let us prove this statement by induction. The base case ($i = 0$) is that

$$B^{\pi'_D}(V^\pi) \geq V^\pi$$

$$B^{\pi'_D}(V^\pi(s)) = \mathcal{R}(s, \pi'_D(s)) + \gamma \sum_{s' \in \mathcal{N}} \mathcal{P}(s, \pi'_D(s), s')V^\pi(s')$$

We know that $\pi'_D(s) = G(V^\pi(s))$ is the action that maximizes $\{\mathcal{R} + \gamma \sum_{s' \in \mathcal{N}} \mathcal{P}(s,a,s')V^\pi(s')\}$. Therefore $B^{\pi'_D}(V^\pi(s)) = \max_{a \in \mathcal{A}} Q^\pi(s,a), \; \forall s \in \mathcal{N}$. Since $V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(s,a)Q^\pi(s,a), \; \forall s \in \mathcal{N}$ is the weighted average of $Q^\pi(s,a)$ over all possible actions and $B^{\pi'_D}(V^\pi(s))$ is the maximum of $Q^\pi(s,a)$ over all possible actions. We have that $B^{\pi'_D}(V^\pi(s)) \geq V^\pi(s), \; \forall s \in \mathcal{N}$, this proves the base case.

Now to complete the prove we need to show that

$$\text{If } (B^{\pi'_D})^{i+1}(V^\pi) \geq (B^{\pi'_D})^i(V^\pi) \text{ then } (B^{\pi'_D})^{i+2}(V^\pi) \geq (B^{\pi'_D})^{i+1}(V^\pi).$$

Let $s \in \mathcal{N}$,

$$(B^{\pi'_D})^{i+2}(V^\pi(s)) = \mathcal{R}(s, \pi'_D(s)) + \gamma \sum_{s' \in \mathcal{N}} \mathcal{P}(s, \pi'_D(s), s')(B^{\pi'_D})^{i+1}(V^\pi(s'))$$

$$(B^{\pi'_D})^{i+i}(V^\pi(s)) = \mathcal{R}(s, \pi'_D(s)) + \gamma \sum_{s' \in \mathcal{N}} \mathcal{P}(s, \pi'_D(s), s')(B^{\pi'_D})^{i}(V^\pi(s'))$$

$$(B^{\pi'_D})^{i+2}(V^\pi(s)) - (B^{\pi'_D})^{i+1}(V^\pi(s)) = \gamma \sum_{s' \in \mathcal{N}} \mathcal{P}(s, \pi'_D(s), s')((B^{\pi'_D})^{i+1}(V^\pi(s')) - (B^{\pi'_D})^{i}(V^{\pi'_D}(s')))$$
$$\geq 0.$$

Since $\gamma \in [0,1]$ and $\mathcal{P}$ maps only to positive values and since we assumed that $(B^{\pi'_D})^{i+1}(V^\pi) \geq (B^{\pi'_D})^i(V^\pi)$. Thus $(B^{\pi'_D})^{i+2}(V^\pi) \geq (B^{\pi'_D})^{i+1}(V^\pi)$. This completes our proof by induction. □

Policy Iteration Algorithm

- Start with any Value Function $V_0 \in \mathbb{R}$
- For each iteration $j = 1, 2, ...$ calculate
  - Deterministic Policy $\pi_{j+1} = G(V_j)$
  - Value Function $V_{j+1} = \lim_{i \to \infty} (B^{\pi_{j+1}})^i (V_j)$
- Terminate when $d(V_j, V_{j+1}) = \max_{s \in \mathcal{N}} |(V_j - V_{j+1})(s)|$ is adequately small.

So the algorithm terminates when there is no further improvement to the Value Function, and the following statement should hold:

$$V_j = (B^{G(V_j)})^i (V_j) = V_{j+1},\ \forall i = 0, 1, 2, \dots$$

In particular, this equation should hold for $i = 1$

$$V_j(s) = (B^{G(V_j)})(V_j)(s) = \mathcal{R}(s, G(V_j)) + \gamma \sum_{s' \in \mathcal{N}} \mathcal{P}(s, G(V_j), s') V_j(s),\ \forall s \in \mathcal{N}$$

We know that $G(V_j)$ is the action the maximizes the Bellman Policy Operator thus,

$$V_j(s) = \max_{a \in \mathcal{A}} \{\mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{N}} \mathcal{P}(s, a, s') V_j(s)\},\ \forall s \in \mathcal{N}$$

This is the MDP State-Value Function Bellman Optimality equation, which would mean that if $V_j = V_{j+1}$ then $V_j = V^*$.

The deterministic policy π_j is therefore also an optimal policy by the Optimal Value Function & Optimal Policy theorem.

This proves the following theorem:

Theorem (Policy Iteration Convergence Theorem)**.** For a Finite MDP with $|\mathcal{N}| = m$ and $\gamma < 1$. The Policy Iteration Algorithm converges to the optimal Value Function $V^* \in \mathbb{R}^m$, along with a deterministic policy $\pi_D^* : \mathcal{N} \to \mathcal{A}$, no matter which value function $V_0 \in \mathbb{R}^m$ we choose to start with.

Bellman Optimality Operator and Value Iteration Algorithm

Definition (Bellman Optimality Operator)**.**

$$B^* : \mathbb{R}^m \to \mathbb{R}^m,\ B^*(V)(s) = \max_{a \in \mathcal{A}} \{\mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{N}} \mathcal{P}(s, a, s') V_j(s)\},\ \forall s \in \mathcal{N}$$

Or equivalently

$$B^*(V)(s) = \max_{a \in \mathcal{A}} \{\sum_{s' \in \mathcal{N}} \sum_{r \in \mathcal{D}} \mathcal{P}_R(s, a, r, s')(r + \gamma W(s'))\},\ \forall s \in \mathcal{N}$$

Now we want to show that B^* is a contraction function under the L^∞ norm. So we can take advantage of the Banach Fixed-Point Theorem and solve the MDP Control Problem by iterative applications of B^*.

We want to prove that $\forall X, Y \in \mathbb{R}^m$:

$$\max_{s \in \mathcal{N}} |(B^*(X) - B^*(Y))(s)| \leq \gamma \max_{s \in \mathcal{N}} |(X - Y)|(s),\ \forall s \in \mathcal{N}$$

For this prove we need to utilize 2 key properties of B^*:

1. Monotonicity Property: $\forall X, Y \in \mathbb{R}^m,\ X(s) \geq Y(s),\ \forall s \in \mathcal{N} \implies B^*(X)(s) \geq B^*(Y)(s),\ \forall s \in \mathcal{N}$
2. Constant Shift Property: $\forall X \in \mathbb{R}^m,\ \forall s \in \mathcal{N},\ \forall c \in \mathbb{R}:\ B^*(X+c)(s) = B^*(X)(s) + \gamma c$

Let's now prove that B^* is indeed a contraction function. Let $X, Y \in \mathbb{R}^m$, and assume that $\max\limits_{s \in \mathcal{N}} |(X - Y)(s)| = c$, then $X(s) - c \leq Y(s) \leq X(s) + c$. Now by the monotonicity and constant shift properties we obtain:

$$\begin{aligned} & B^*(X(s) - c) \leq B^*(Y(s)) \leq B^*(X(s) + c) \\ \iff & B^*(X(s)) - \gamma c \leq B^*(Y(s)) \leq B^*(X(s)) + \gamma c \\ \implies & \max_{s \in \mathcal{N}} |(B^*(X) - B^*(Y))(s)| \leq \gamma c = \gamma \max_{s \in \mathcal{N}} |(X - Y)|(s),\ \forall s \in \mathcal{N} \end{aligned}$$

Since $\gamma < 1$ we conclude that B^* is a contraction function. So invoking Banach's Fixed-Point Theorem proves the following theorem:

Theorem (Value Iteration Convergence Theorem)**.** For a Finite MDP with $|\mathcal{N}| = m$ and $\gamma < 1$. If $V^* \in \mathbb{R}^m$ is the optimal value function, then V^* is the unique fixed point of the Bellman Optimality Operator $B^* : \mathbb{R}^m \to \mathbb{R}^m$, and

$$\lim_{i \to \infty} (B^*)^i (V_0) \to V^*,\ \forall V_0 \in \mathbb{R}^m$$

This gives us the following algorithm, known as the Value Iteration Algorithm,

- Start with any value function $V_0 \in \mathbb{R}^m$
- Iterating over $i = 0, 1, 2, \ldots$, calculate in each iteration

$$V_{i+1}(s) = B^*(V_i(s)),\ \forall s \in \mathcal{N}$$

- Terminate when $d(V_i, V_{i+1}) = \max\limits_{s \in \mathcal{N}} |(V_i - V_{i+1})(s)|$ is adequately small.

Optimal Policy from Optimal Value Function

When we converge to the optimal value function with the Policy Iteration Algorithm in stop $j, V_j = V^*$, the algorithm has a deterministic policy π_j associated with V_j s.t.

$$V_j = V^{\pi_j} = V^*$$

and we refer to π_j as the optimal policy π^*, that yields the optimal value function

$$V^{\pi_j} = V^{\pi^*} = V^*$$

But value iteration has no such policy, since it only operates on value function. So how do we extract the optimal policy from the optimal value function. The answer lies in the greedy policy function G^*, we know that

$$B^{G(V)}(V) = B^*(V),\ \forall V \in \mathbb{R}^m$$

Specializing for V^* and using the fact that V^* is the fixed point B^*, we obtain:

$$B^{G(V^*)}(V^*) = B^*(V^*) = V^*,\ \forall V \in \mathbb{R}^m$$

The above equation tells us that V^* is the unique fixed point of $B^{G(V^*)}$ but $B^{G(V^*)}$ has unique fixed point $V^{G(V^*)}$, therefore

$$V^{G(V^*)} = V^*$$

This states that $G(V^*) = \pi^*$ is the (Deterministic) Optimal Policy.

Generalized Policy Iteration

Let us start by looking at a 2D layout of the progression of a value function in the policy iteration algorithm from the starting value function V_0 to the optimal value function V^*:

$$\begin{aligned}
\pi_1 = G(V_0), V_0 &\rightarrow B^{\pi_1}(V_0) \rightarrow (B^{\pi_1})^2(V_0) \rightarrow \cdots \rightarrow V^{\pi_1} = V_1 \\
\pi_2 = G(V_1), V_1 &\rightarrow B^{\pi_2}(V_1) \rightarrow (B^{\pi_2})^2(V_1) \rightarrow \cdots \rightarrow V^{\pi_2} = V_2 \\
\cdots\cdots\cdots&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
\pi_{j+1} = G(V_j), V_j &\rightarrow B^{\pi_{j+1}}(V_j) \rightarrow (B^{\pi_{j+1}})^2(V_j) \rightarrow \cdots \rightarrow V^{\pi_{j+1}} = V^*
\end{aligned}$$

Each row starts with the creation of the policy using the greedy policy function, then successive applications of the Bellman Policy Operator until convergence to the value function for that row's policy.

So each row starts with policy improvement followed by policy evaluation.

The greedy policy operator and the Bellman policy operator apply to all states $s \in \mathcal{N}$. Therefore the policy iteration has 3 nested loops:

- The outermost loop loops over each row where each row creates an improved policy.
- The loop within the outermost loop loops over each column for each row successively applying the Bellman policy operator, evaluating the value function for that row's policy.
- The innermost loop loops over each state in $\mathcal{N}$ since we need to sweep through all non-terminal states when updating the value function, by applying the Bellman policy operator to a value function.

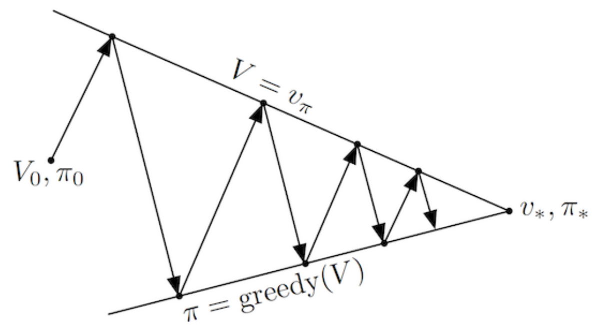

Figure 1: Progression of value function and policy in policy iteration

The idea behind this image is that the lower line represents the progression of policy improvement and the upper line represents the progression of value function improvement as the algorithm moves along. Arrows pointing towards the upper line represent policy evaluation and arrows pointing towards the lower line represent policy improvement.

The key takeaway is that policy improvement and policy evaluation push in different direction, even as they aim to get the value function and policy consistent with each other. Note that there are actually 2 notions of consistency between a policy and a value function:

1. The notion of the value function V being consistent with / close to the value function V^π of the policy π.
2. The notion of the policy π being consistent with / close to the greedy policy $G(V)$ of the value function V.

Policy Evaluation aims for improvement of the 1st notion but in turn worsens the 2nd one. Policy Improvement does the opposite.

Generalized Policy Iteration is the idea that we can evaluate the Value Function for a policy with any Policy Evaluation method, and we can improve a policy with any Policy Improvement method.

Definition (Generalized Policy Iteration)**.** GPI is any algorithm to solve MDP Control problems that alternates some form of policy evaluation and some form of policy improvement.

Asynchronous Dynamic Programming

All previous discussed algorithms are qualified as synchronous DP algorithms. Synchronous refers to 2 things:

1. All states' values are updated in each iteration.
2. The mathematical description of the algorithm corresponds to all states' values being updated simultaneously. However when implementing in code, this simultaneous update would be done by creating a new copy of the value function and sweeping through all states from the old function one-by-one to update the new copy.

In practice DP algorithms are generally asynchronous where these 2 constraints are relaxed. A benefit of this approach is that we can maintain a single vector for the value function and update it's value in-place.

Another feature is that we can evaluate te states in any way we'd like, meaning we can assign priorties for evaluating certain states.

A simple but effective way of prioritizing state value updates is known as prioritized sweeping. We maintain a queue of the states, sorted by their value function gaps $g : \mathcal{N} \to \mathbb{R}$:

$$g(s) = |V(s) - \max_{a \in \mathcal{A}} \{\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{N}} \mathcal{P}(s,a,s')V(s')\}|, \; \forall s \in \mathcal{N}$$

After each states' value get's updated by the Bellman Optimality Operator, we update the value function for each state where it's value changes as a result of this update. These are exactly the states from which we have probabilistic transistion to the state whose value just got updated.

So, after each state update the queue of states are resorted. We always pull out the state with the largest value function gap, and update the value function for that state. This prioritizes states with the largest function gaps and ensures we quickly get to a point where the value function gaps are low enough.

A specialization of MDP's is when each state is encountered not more then once in sequence of state occurences, and when all of these random sequences terminate. This structure can be conceptualized as a Directed Acyclic Graph (DAG). Wherein each non-terminal node represents a tuple of non-terminal state and action and each terminal node represents a terminal state. The edges of the graph represent the probabilistic transistions of the MDP.

Here the MDP control and prediction problems can be solved by walking backwards on the DAG fron the terminal nodes and setting the value function for each visited node using the Bellman optimality equation(for control) or Bellman policy equation(for prediction).

Finite Horizon MDP's

In this section we consider Finite Horizon MDP's, which are DAG-structrued MDP's, meaning we can model each non-terminal node of the DAG as non-terminal state - action pair and each terminal node as a terminal state. With the addition that each sequence terminates within a fixed number of time steps T and each time step has a seperate (from other time steps) set of countable states.

So all states at time step T are terminal states and all states before time step T could be terminal states. Now denote $\mathcal{S}_t,\ \forall t = 0, 1, \ldots, T$ as the set of non-terminal states at time step t, $\mathcal{T}_t$ as the set of terminal states at time step t and $\mathcal{N}_t$ as the set of non-terminal states at time step t.

When the MDP isn't time-homogenoeus we augment each state to include the index of it's time step. it's state space now becomes: $\mathcal{S} = \{(t, s_t) \mid t = 0, 1, \ldots, T, s_t \in \mathcal{S}_t\}$. And the MDP's set of terminal states: $\mathcal{T} = \{(t, s_t) \mid t = 0, 1, \ldots, T, s_t \in \mathcal{T}_t\}$. We denote the set of rewards receivable at time step t as $\mathcal{D}_t$ (countable subset of $\mathbb{R}$) and we denote the set of allowable actions at time t from $\mathcal{N}_t$ as $\mathcal{A}_t$. Then the entire action space becomes: $\mathcal{A} = \cup_{t=1}^{T-1} \mathcal{A}_t$. The state-reward transistion probability function is given by:

$$\mathcal{P}_\mathcal{R} = \begin{cases} (\mathcal{P}_\mathcal{R})_t(s_t, a_t, r_{t+1}, s_{t+1}) \text{ if } t = t+1,\ s_{t+1} \in \mathcal{S}_{t+1} \text{ and } r_{t+1} \in \mathcal{D}_{t+1} \\ 0, \text{ otherwise} \end{cases}$$

For all $t = 0, 1, \ldots, T-1,\ s_T \in \mathcal{S}_t,\ a_t \in \mathcal{A}_t,\ t+1 \in 0, 1, \ldots, T$. Where

$$(\mathcal{P}_\mathcal{R})_t : \mathcal{N}_t \times \mathcal{A}_t \times \mathcal{D}_{t+1} \times \mathcal{S}_{t+1} \to [0, 1]$$

are the seperate state-reward transistion probability functions for each time step $t = 0, 1, \ldots, T-1$, such that

$$\sum_{r_{t+1} \in \mathcal{D}_{t+1}} \sum_{s_{t+1} \in \mathcal{S}_{t+1}} (\mathcal{P}_\mathcal{R})_t(s_t, a_t, r_{t+1}, s_{t+1}) = 1$$

for all $t = 0, 1, \ldots, T-1,\ s_t \in \mathcal{N}_t,\ a_t \in \mathcal{A}_t$.

Likewise it's convenient to represent any policy of the MDP: $\pi : \mathcal{N} \times \mathcal{A} \to [0, 1]$ as:

$$\pi((t, s_t), a_t) = \pi_t(s_t, a_t)$$

Where $\pi_t : \mathcal{N}_t \times \mathcal{A}_t \to [0, 1]$ are the seperate policies for each time-step $t = 0, 1, \ldots, T-1$. So essentially we can interpert π as being composed of the sequence $(\pi_0, \pi_1, \ldots, \pi_{T-1})$. Consequently the value function for a given policy π: $V^\pi : \mathcal{N} \to \mathbb{R}$ can be represented in terms of a sequence of value functions

$$V_t^\pi : \mathcal{N}_t \to \mathbb{R},\ V_t^\pi(s_t) = V^\pi((t, s_t)),\ \forall t = 0, 1, \ldots, T-1,\ \forall s_t \in \mathcal{S}_t$$

Then the Bellman Policy Equation can be written as:

$$V_t^\pi(s_t) = \sum_{r_{t+1} \in \mathcal{D}_{t+1}} \sum_{s_{t+1} \in \mathcal{S}_{t+1}} (\mathcal{P}_\mathcal{R}^{\pi_t})_t(s_t, r_{t+1}, s_{t+1})(r_{t+1} + \gamma W_{t+1}^\pi(s_{t+1})),$$

for all $t = 0, 1, \ldots, T-1$ and $s_t \in \mathcal{S}_t$.
Where,

$$W_t^\pi(s_t) = \begin{cases} V_t^\pi(s_t) \text{ if } s_t \in \mathcal{N}_t \\ 0 \text{ if } s_t \in \mathcal{T}_t \end{cases} \quad \forall t = 1, 2, \ldots, T$$

and

$$(\mathcal{P}_\mathcal{R}^{\pi_t})_t(s_t, r_{t+1}, s_{t+1}) = \sum_{a_t \in \mathcal{A}_t} \pi_t(s_t, a_t)(\mathcal{P}_\mathcal{R})_t(s_t, a_t, r_{t+1}, s_{t+1})$$

for all $t = 0, 1, \ldots, T-1$.

So for a finite MDP this yields a simple algorithm to calculate V_t^π for all t by simply decrementing down from $t = T-1$ to $t = 0$ and using the rewritten Bellman Policy Equation to calculate V_t^π for all $t = 0, 1, \ldots, T-1$ from the known values of W_{t+1}^π (since we're stepping back in time). This solves the Finite Horizon MDP Prediction problem.

We now move on to the Control Problem − to calculate the Optimal Value Function and Opimal Policy. Similair to the pattern we've seen before we can write the optimal value function $V^* : \mathcal{N} \to \mathbb{R}$ as a sequence of value functions $V_t^* : \mathcal{N}_t \to \mathbb{R}$ for each $t = 0, 1, \ldots, T-1$ defined as:

$$V_t^*(s_t) = V^*((t, s_t))$$

. Thus the MDP State-Value Function Bellman Optimality Equation can be written as:

$$V_t^*(s_t) = \max_{a_t \in \mathcal{A}_t} \{(\mathcal{P}_\mathcal{R})_t(s_t, a_t, r_{t+1}, s_{t+1})(r_{t+1} + \gamma W_{t+1}^*(s_{t+1}))\}$$

for all $t = 0, 1, \ldots, T-1$.
Where,

$$W_t^*(s_t) = \begin{cases} V_t^*(s_t) \text{ if } s_t \in \mathcal{N}_t \\ 0 \text{ if } s_t \in \mathcal{T}_t \end{cases} \quad \forall t = 1, 2, \ldots, T$$

the associated optimal deterministic policy $(\pi_D^*) : \mathcal{N}_t \to \mathcal{A}_t$ is defined as:

$$(\pi_D^*)_t(s_t) = \arg\max_{a_t \in \mathcal{A}_t} \{(\mathcal{P}_\mathcal{R})_t(s_t, a_t, r_{t+1}, s_{t+1})(r_{t+1} + \gamma W_{t+1}^*(s_{t+1}))\}$$

for all $t = 0, 1, \ldots, T-1,\ , \forall s_t \in \mathcal{N}_t$. So for a finite MDP this yields a simple algorithm to calculate V_t^* for all t by simply decrementing from $t = T-1$ to $t = 0$, using the rewritten MDP State-Value Function Bellman Optimality Equation to calculate V_t^* and the last equation to calculate $(\pi_D^*)_t$ for all $t = 0, 1, \ldots, T-1$ from the known values of W_{t+1}^*.

Note that these algorithms for Finite Horizon MDP's don't require any "iterations to convergence" like we had for regular Policy Evaluation and Value Iteration. Rather in these algorithms we simply walk back in time and immediately obtain the Value Function for each time step to the next time step's Value Function. These algorihtms are called "Backward Induction Algorithms".