

# Markov Decision Processes Applications

Giovanni Bianco

September 2024

## 1 Introduction to Finite Markov Decision Processes (MDPs)

Markov Decision Processes (MDPs) are a fundamental framework in the study of decision-making under uncertainty, particularly in the fields of reinforcement learning, operations research, and control theory. Formally, a finite MDP is completely defined by:

- $S$  is a finite set of states within the environment.
- $A$  is a finite set of actions available to the agent. The choice of action affects the transition between states.
- $P : S \times A \times S \rightarrow [0, 1]$  is the transition probability function, where  $P(s'|s, a)$  specifies the probability of transitioning from state  $s$  to state  $s'$  and therefore receiving the associated reward  $r$  after taking action  $a$ .
- $R : S \times A \times S \rightarrow R$  is the reward function, where  $r = R(s, a, s')$  provides the immediate reward obtained after transitioning from state  $s$  to state  $s'$  as a result of taking action  $a$ . The expected reward for taking action  $a$  in state  $s$  is defined by the equation:

$$R(s, a) = \sum_{s'} P(s'|s, a) \cdot R(s, a, s')$$

Here,  $P(s'|s, a)$  is the transition probability, and  $R(s, a, s')$  is the reward associated with ending up in  $s'$  when action  $a$  is taken in state  $s$ .

The objective in an MDP is to find a policy  $\pi : S \rightarrow A$ , a mapping from states to actions, that maximizes the expected sum of discounted rewards over time, known as the *return*. This can be expressed as:

$$\max_{\pi} E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right],$$

where  $s_t$  and  $a_t$  denote the state and action at time step  $t$ , respectively, and the expectation is taken with respect to the policy  $\pi$  and the transition probabilities  $P$ .

MDPs are forgetful processes, characterized by the *Markov property*, which states that the future state of the system depends only on the current state and the chosen action, not on the sequence of events that preceded it.

## 2 Solving the MDP

Solving an MDP involves finding an optimal policy  $\pi^*$  that maximizes the expected cumulative reward over time, often referred to as the *return*. There are two primary methods for solving MDPs: **Value Iteration** and **Policy Iteration**. Both algorithms leverage the Bellman optimality equations to iteratively compute the optimal value function and policy.

### 2.1 Value Iteration

Value iteration directly approximates the optimal value function  $V^*(s)$ , which represents the maximum expected return from any state  $s$ , given that the agent acts optimally from that state forward. The algorithm updates the value function iteratively using the Bellman equation:

$$V_{k+1}(s) = \max_{a \in A} \left[ R(s, a) + \gamma \sum_{s'} P(s'|s, a) V_k(s') \right]$$

Where:

- $V_k(s)$  is the value of state  $s$  at iteration  $k$ .
- $R(s, a)$  is the expected reward for taking action  $a$  in state  $s$ .
- $P(s'|s, a)$  is the transition probability to state  $s'$  from state  $s$  after taking action  $a$ .
- $\gamma$  is the discount factor.

The algorithm proceeds as follows:

1. Initialize  $V_0(s)$  arbitrarily for all  $s \in S$ .
2. For each state  $s$ , update  $V_{k+1}(s)$  using the Bellman equation.
3. Repeat the process until  $V_k(s)$  converges to  $V^*(s)$  within a small threshold  $\epsilon$ .
4. Once  $V^*(s)$  is found, the optimal policy  $\pi^*(s)$  is obtained by taking the action that maximizes the value function:

$$\pi^*(s) = \arg \max_{a \in A} \left[ R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^*(s') \right]$$

## 2.2 Policy Iteration

Policy iteration alternates between *policy evaluation* and *policy improvement*. Instead of directly computing the value function, this method iteratively improves a policy by evaluating its performance.

**Policy Evaluation:** Given a policy  $\pi$ , the value function  $V^\pi(s)$  is calculated by solving the following system of linear equations:

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s'|s, \pi(s)) V^\pi(s')$$

This step determines the value of each state under the current policy  $\pi$ .

**Policy Improvement:** Once the value function  $V^\pi$  is computed, the policy is updated by acting greedily with respect to  $V^\pi$ :

$$\pi'(s) = \arg \max_{a \in A} \left[ R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^\pi(s') \right]$$

This ensures that the new policy  $\pi'$  is at least as good as the previous policy  $\pi$ .

The algorithm proceeds as follows:

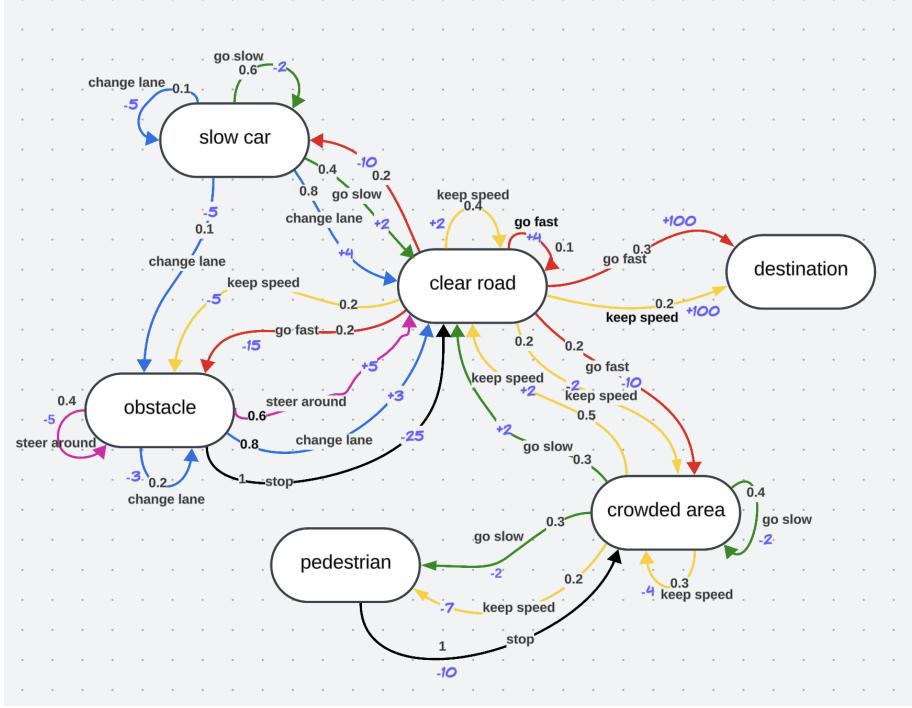
1. Initialize an arbitrary policy  $\pi_0$ .
2. Perform policy evaluation to compute  $V^{\pi_0}(s)$ .
3. Perform policy improvement to update the policy to  $\pi_1$ .
4. Repeat until the policy converges, i.e., when no further improvement is possible:  $\pi_{k+1}(s) = \pi_k(s)$  for all states  $s \in S$ .

## 2.3 Comparison of Value Iteration and Policy Iteration

- **Value Iteration** typically converges faster in practice, as it directly approximates the optimal value function, but may require more iterations than policy iteration.
- **Policy Iteration** converges in fewer iterations because it computes exact values for a given policy, but policy evaluation (solving a system of linear equations) can be computationally expensive, especially for large state spaces.

Both methods are guaranteed to converge to the optimal policy  $\pi^*$ , provided the MDP is finite and the discount factor  $\gamma < 1$ .

### 3 First scenario: Autonomous car



In my model of autonomous car, the ultimate goal is for the car to reach its destination, which, when achieved, provides a reward of 100 points. To discourage lingering on the road, I will implement a discounting mechanism in the reward system. The states I chose are: Destination, Clear road, Crowded area, Pedestrian, Obstacle, Slow car. The actions are: Go Fast, Go Slow, Keep Medium Speed, Change Lane, Steer around and Stop.

**Clear road** As the journey begins on a clear road, the car has two pacing options: fast or medium. Opting to drive fast increases the likelihood of reaching the destination to 40%, as spending less time on the road reduces the probability of running into problems. However, if the car continues at a fast pace and remains on the clear road, which has a 10% probability, it receives a small reward of +4, since it is still moving rapidly towards the destination without incurring any problem. Despite the advantages of speed, there are notable risks: entering an crowded area at a high speed (20% probability) leads to a penalty of -10 due to potential difficulties in adjusting speed in time, which might cause accidents. Encountering an obstacle at this speed (also with a 20% probability) poses even greater challenges, reflected in a harsher penalty of -15. Additionally, if the car ends up behind a slower vehicle (20% probability), it receives a -10 reward since it's somewhat easier to handle than an obstacle but still requires careful maneuvering.

Conversely, choosing a medium pace lowers the probability of a quick arrival to 10%. However, it allows the car to manage encounters more effectively due to the reduced speed. If it stays on the clear road at this pace (40% probability), the reward is +2, acknowledging steadier but slower progress towards the destination. Encounters with obstacles or entering crowded areas, each with a 20% probability, incur lower penalties of -5 and -2, respectively, as the situations are easier to handle at a medium speed. In this way I emphasized the trade-offs between speed and safety, highlighting how strategic choices on the road impact the car's ability to navigate and ultimately reach its destination efficiently and safely.

**Crowded area** Let's now explore how the car would handle different situations. For instance, if it enters a crowded area, it could either decide to maintain a moderate speed or slow down.

In the first scenario, if the car keeps a moderate pace, it has a higher likelihood of passing through the crowded area and returning to a clear road (50% chance). However, if a pedestrian appears (with a probability of 0.2), the outcome would lead to a negative reward of -7. Additionally, if the car remains in the crowded area (with a 20% chance), the reward would be -4 due to prolonged exposure to suboptimal conditions. On the other hand, if the car chooses to slow down, it is more likely to stay in the crowded area longer (40% chance to remain and 30% to go back to clear road). Despite this, the risk is reduced, and the reward of remaining in the crowded area is a more manageable -2. In this case, the likelihood of encountering a pedestrian increases to 30%, but since the car is moving slowly, it can stop in time, resulting in a smaller penalty of -3.

**Pedestrian** When a pedestrian is encountered, the system transitions to a 'pedestrian' state, where the only possible action is stopping. This action brings the car back to the crowded area with 100% certainty, though at a cost. The act of stopping delays the car further, resulting in a reward of -10. When transitioning from the crowded area back to the clear road, we receive a reward of +2 due to the improvement in driving conditions.

**Slow car** On the clear road, as previously mentioned, if we choose to drive fast, we might encounter a slower vehicle. In this case, the car has two options: slow down or change lanes. If the car slows down, there is a 60% chance it will remain behind the slower vehicle, resulting in a negative reward of -2. However, there is a 40% chance the slower car will either accelerate or move away from our car, returning to the clear road with a reward of +2. If our car attempts to change lanes, there are several possible outcomes. With an 80% probability, the lane will be empty, allowing us to return to the clear road and earn a reward of +4. However, if the lane is not empty, the car will recognize this and avoid changing lanes, leaving it stuck behind the slower car without reducing speed. This scenario may increase risk and keeps us in the slower car state, therefore the reward will be -5. Additionally, with a 10% probability, the lane might be

available for a change, but there could be a minor obstacle present. In this case, the car transitions to the 'obstacle' state, resulting in a reward of -5.

**Obstacle** The 'obstacle' state encompasses a variety of situations, such as a minor object on the road, a stopped vehicle, a closed lane, or even an accident that requires a complete stop. I've put together all these cases to avoid over-complicating the representation. In this scenario, the car has several options. It may attempt to steer around the obstacle. If the obstacle is minor (with a 60% probability), the action will succeed, returning the car to the clear road and yielding a reward of +5, as the car avoids slowing down or performing unnecessary maneuvers. On the other hand, if the obstacle is unavoidable, we would need to select another action, losing time and resulting in a reward of -5. If the car tries to change lanes, there is an 80% chance it will successfully avoid the obstacle, leading to a reward of +3. However, if the lane change is unsuccessful—either because it's not possible or doesn't allow us to clear the obstacle—the car will remain in the obstacle state obtaining a reward of -5. If the obstacle is unavoidable, the best choice might be stopping and wait for the situation to improve (this would lead us with a 1% probability to the clear road). But the obstacle might require a long stop (e.g., due to an accident), which would slow the car down significantly, therefore the associated reward will be -25.

### 3.1 Matrix Creation and Reward Structuring

To efficiently guide the agent's actions within the desired behavioral framework, specific methodologies were employed in constructing the transition matrix and defining the reward system:

- **Focusing on Desired Actions:** The transition probabilities were primarily defined for actions considered beneficial or necessary for the agent's objectives. This selective focus aids in streamlining the decision-making process, ensuring the agent prioritizes valuable actions.
- **Handling Undesired Actions:** Actions not aligned with the agent's goals were assigned a deterministic outcome, leading to a specific state (e.g., "pedestrian") with a transition probability of 1. This assignment simplifies the state space by reducing the consideration of non-essential actions.
- **Reward Assignment:** To deter the agent from selecting undesirable actions, a punitive reward of -1000 was assigned to these actions. This substantial penalty is designed to strongly discourage the agent from executing any non-optimal actions, aligning its behavior with the intended strategy.

The approach ensures that the agent's learning and decision-making processes are both efficient and aligned with the strategic objectives, by focusing on reinforcing beneficial actions while penalizing detrimental ones.

### 3.1.1 Discount factor

I decided to use a discount factor of 0.9 to effectively discourage actions that delay reaching the destination unnecessarily, like excessive detouring. A discount factor less than 1 reduces the present value of future rewards, thus making immediate rewards more appealing compared to those received later. This choice helps ensure that strategies favoring quicker paths to the destination are preferred, as actions leading to prolonged routes become less beneficial due to the compounded reduction in reward value over time.

## 3.2 Solving the MDP

I implemented Policy and Value iteration making each run for 1000 episodes, with 100 steps per episode. As previously announced, the discount factor gamma used was 0.9 and the theta was 1e-3. I obtained the following results:

### Value Iteration

**Convergence Time:** 0.002246 s  
**Average Reward:** -609.924

#### Value Function

State	Value
Pedestrian	-1956.925
Clear Road	-274.506
Destination	0.0
Crowded Area	-1063.251
Obstacle	-238.834
Slow Car	-215.699

#### Policy

State	Action
Pedestrian	Stop
Clear Road	Go Fast
Destination	Stop
Crowded Area	Keep
	Medium
	Speed
Obstacle	Change
	Lane
Slow Car	Go Slow

### Policy Iteration

**Convergence Time:** 0.002511 s  
**Average Reward:** -617.656

#### Value Function

State	Value
Pedestrian	-1956.924
Clear Road	-274.506
Destination	0.0
Crowded Area	-1063.251
Obstacle	-238.833
Slow Car	-215.698

#### Policy

State	Action
Pedestrian	Stop
Clear Road	Go Fast
Destination	Stop
Crowded Area	Keep
	Medium
	Speed
Obstacle	Change
	Lane
Slow Car	Go Slow

## Comparative Analysis

Difference Type	Value
Value Function Difference (Sum of Absolute Differences)	0.00422
Policy Difference (Number of Different Actions)	0

### 3.3 Commentary on the Results

The results from both the algorithms are remarkably similar, showing very little variation in terms of convergence times, value functions, and policies. Both methods converge to identical policies and value functions with negligible differences, indicating that they are equally effective for this specific model. The slight discrepancy in average rewards and convergence times is minor, suggesting that either algorithm could be suitably efficient and effective in similar settings. These findings highlight the robustness and reliability of both algorithms in determining optimal policies for decision-making under the given conditions. The results make sense and resemble the decisions a human driver would take in the situations given. The policies generated reflect a direct response to the model's specific formulation. For example, the action to 'Stop' in the presence of a 'Pedestrian' or at the 'Destination' are forced by my model choices but adhere strictly to safety and goal completion criteria inherent in the model. Actions such as 'Change Lane' for 'Obstacles' and 'Keep Medium Speed' in 'Crowded Areas' demonstrate the model's capability to incorporate realistic traffic conditions and decision-making nuances required for safe and efficient navigation. The choice of the model to go fast on a clear road reflects the need to reach the destination as quickly as possible, even at the cost of higher risks. In summary, while highly simplified, this scenario effectively illustrates how a real-world project could be approached and completed by a reinforcement learning agent. It provides a clear sense of the task structure and the process an RL agent would follow to accomplish such objectives, offering valuable insight into the potential execution of a more complex project.

## 4 Task 2: Modelling Single Stock Trading Agent as an MPD

In this scenario we will model an agent able to trade on a single stock in the market. The bot can execute one of three actions: buy, sell, or hold the stock. We consider the scenario where we are a small investor, whose trading actions do not significantly influence market prices. Assuming no impact from external factors such as breaking news or global market trends, given the forgetfulness of MDPs, the available information is typically insufficient for accurately predicting future market movements. In such a model, following, for example a price increase, the probabilities of further increases, decreases, or market stability are

assumed to be equally likely.

#### 4.1 Model Formulation and State Definitions

To enhance the model's realism, we consider the duration of price changes by defining two states for each market condition. Using price increases as an example:

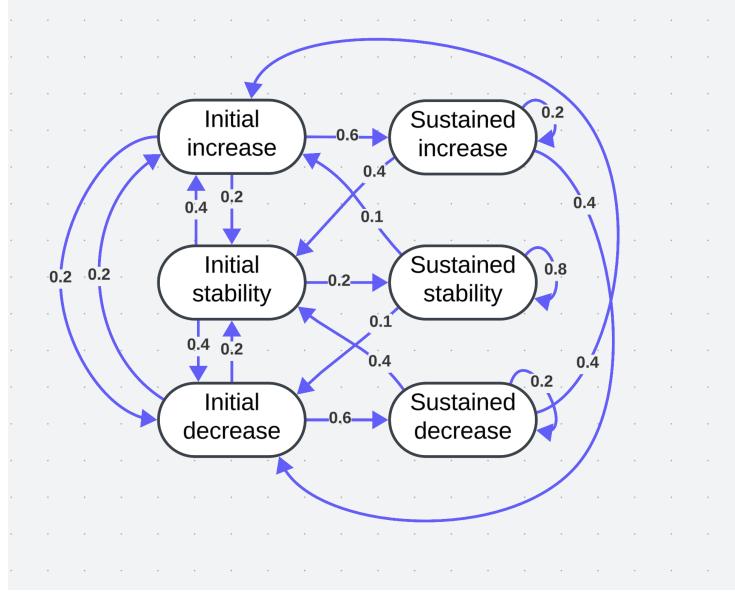
1. **Initial Increase:** Indicates the market has just begun to rise.
2. **Sustained Increase:** Represents a sustained upward trend in the market.

We assume that if a price increase has just commenced (*Initial Increase*), it is more likely to continue for some time. Therefore, the probability of transitioning from *Initial Increase* to *Sustained Increase* is modeled as 0.6, while the probabilities of transitioning to an *Initial Decrease* or to *Stability* are 0.2 each, respectively. However, if the price has been increasing for an extended period (*Sustained Increase*), the likelihood of remaining in this state decreases to 0.2, with a transition probability of 0.4 to any other initial state, representing a change in market conditions. The same reasoning applies to price decreases.

The only exception is when the market is stable. In economics, there is a notion of market equilibrium, which states that markets continuously fluctuate and adjust until they reach an equilibrium state. Therefore, from an initial stability, we modeled a 0.4 probability of transitioning to an initial increase and a 0.4 probability of transitioning to an initial decrease. However, with a 0.2 probability, when demand meets supply, the price stabilizes and remains in this state for an extended period, until an external shock disrupts the equilibrium.

Since we are disregarding external interferences in our model, the break of this equilibrium is not considered very likely. Therefore, once sustained stability is reached, the model assumes we will remain stable with a probability of 0.8, and transition to another initial state with a probability of only 0.1 each.

## 4.2 Transition Diagram



## 4.3 Rewards and discount factor

The reward strategy is the following: executing a 'buy' action which is followed by a market increase results in a reward of +10, given the profitable transaction. Conversely, a market decrease following a 'buy' results in a penalty of -10, reflecting a financial loss. If the market remains stable post-purchase, no reward or penalty is applied, indicating a neutral outcome where the position could be maintained without financial impact. The reward structure can be adjusted to encourage either risk-averse or risk-loving behaviors by varying the magnitude of rewards and penalties. For example, reducing the penalty for losses might encourage more frequent buying and selling actions under uncertainty, promoting a risk-seeking behavior. The transition probabilities between states in our model do not depend on the actions taken, thus our transition matrix simply maps the likelihood of moving from one state to another and is therefore the same for all the actions. In contrast, the reward matrix will have the rows connected with the action 'hHold', with only rewards of 0, while the lines of 'Buy' and 'Sell' will be one the inverse of the other. This separation of transition and reward matrices allows our model to flexibly adapt to different trading strategies without complex modifications.

Even in this scenario, a discount factor is inherently present. The concept of discounting originates from finance, reflecting the principle that future money holds less value than the same amount today. This is due to factors such as inflation, the uncertainty of life, and the opportunity cost of not having immediate access to the funds. Applying this to our case, we assume the perspective of

short-term focused traders who aim to maximize immediate profits. Therefore, we set the discount factor to 0.75. This gradual decrease in value over time encourages the model to prioritize actions that yield faster results, discouraging delays in reaching optimal states.

#### 4.4 Solving the MDP and results

I implemented Policy and Value iteration making each run for 1000 episodes, with 100 steps per episode. As previously announced, the discount factor gamma used was 0.75 and the theta was 1e-3. I obtained the following results:

##### Value Iteration

**Convergence Time:** 0.00105 s  
**Average Reward:** -44891.77

##### Value Function

State	Value
Initial Increase	-1422.435
Sustained	-1629.069
Increase	
Initial Decrease	-1422.435
Sustained	-1629.069
Decrease	
Initial Stability	-1866.597
Sustained	-2780.911
Stability	

##### Policy

State	Action
Initial Increase	Buy
Sustained	Sell
Increase	
Initial Decrease	Sell
Sustained	Buy
Decrease	
Initial Stability	Buy
Sustained	Buy
Stability	

##### Policy Iteration

**Convergence Time:** 0.00084 s  
**Average Reward:** -45296.92

##### Value Function

State	Value
Initial Increase	-1422.435
Sustained	-1629.069
Increase	
Initial Decrease	-1422.435
Sustained	-1629.069
Decrease	
Initial Stability	-1866.597
Sustained	-2780.911
Stability	

##### Policy

State	Action
Initial Increase	Buy
Sustained	Sell
Increase	
Initial Decrease	Sell
Sustained	Buy
Decrease	
Initial Stability	Buy
Sustained	Buy
Stability	

## Comparative Analysis

Difference Type	Value
Value Function Difference (Sum of Absolute Differences)	0.0
Policy Difference (Number of Different Actions)	0

### 4.5 Commentary on the Results

The policy obtained from my problem formulation reveals a rational and effective trading strategy within the constraints of the model I designed. By constructing distinct states for the market's behavior—such as "Initial Increase" and "Sustained Stability"—I was able to capture the progression of market trends more realistically, allowing the agent to adapt its actions accordingly.

The policy highlights a few key behaviors that align with typical trading strategies. For example, the agent chooses to **buy during an Initial Increase** and **sell during a Sustained Increase**, which reflects the common practice of capitalizing on upward momentum early and selling before the trend reverses. This makes sense given the model's structure, where sustained trends are less likely to persist indefinitely, and the agent effectively mitigates risk by exiting the market before the price starts to fall.

The **decision to buy in stable markets** is particularly interesting. In the absence of market movement, the agent's choice to purchase indicates a forward-looking strategy, perhaps based on the notion that stability precedes future movement. Given the discount factor applied in the model, this action is understandable as the agent prioritizes short-term gains over delayed rewards.

One strength of this formulation is how the transition probabilities between states capture the duration and persistence of market conditions. For instance, in my model, an Initial Increase is more likely to lead to a Sustained Increase, but with diminishing likelihood as the trend continues. This helps the agent navigate uncertainty by balancing short-term optimism with the risk of market reversal.

Both algorithms—Value Iteration and Policy Iteration—converge to the same optimal policy and value function, confirming the robustness of the problem formulation. However, Policy Iteration provided a slightly better average reward, which suggests that while both methods are effective, Policy Iteration's path to convergence might explore policies that yield marginally better results, despite taking longer to converge.

In summary, the policy makes logical, human-like decisions in various market states, supporting the effectiveness of my MDP formulation. By emphasizing both short-term rewards and the risk of prolonged negative conditions, I was able to design an agent that performs well under uncertainty, reflecting the complexity of real-world trading behavior.