MARKOV DECISION PROCESSES - APPLICATIONS IN FINANCE

JOHN TWEEDIE

Abstract. Markov Decision Processes are a modelling tool used to derive a set of sequential decisions which maximize the utility of a stochastic system. Applications range from finance and insurance, to agriculture and natural resource management, among many other. Building upon the foundational concept of the Markov property, Markov Decision Processes have been studied extensively since their introduction in the early 1950s. This paper provides a survey of Markov Decision Processes, with commentary on the developmental history, construction, and application. A simple example application is provided to illustrate the process.

Keywords. Markov Decision Process, Markov Process, Markov Chain, Stochastic Process, Expected Value

Date: 2023-12-08
Student Number: 400550023
e-mail: tweedij@mcmaster.ca

Contents

1. Introduction

Decision-making in the face of uncertainty and dynamic environments is a ubiquitous problem across nearly any perceivable discipline. Markov Decisions Processes (MDPs) offer a powerful framework for which to formally predict optimal action sequences when uncertainty is inherent to the problem at hand. Building upon the mechanics of Markov chains and Markov reward processes, MDPs produce a conceptual, dynamic model of a stochastic process for which to optimize in regard to an expected terminal reward value, such as the wealth of a portfolio at the end of a specified period. MDPs have extensive applications, often employed in the fields of medicine, manufacturing (Alagoz et al., 2010), communication, signal processing (Hu and Yue, 2008), and financial management (Bäuerle and Rieder, 2011). This paper provides a survey of MDPs, including description of developmental history, general formulations and underlying mechanics, extensions and adaptations, and example applications. A case-study of a simple financial portfolio management example problem is provided to illustrate the process of formulating and applying a MDP.

2. Overview of Markov Decision Processes

2.1. **Definition.**

While many variations and extensions exist, a Markov decision process in its most general form is defined by a model of a discrete-time stochastic process that includes an element of control by the agent. The common goal of MDPs is to solve a problem involving dynamic and repeated decision making in systems that evolve with an element of stochasticity and consequent uncertainty regarding behaviour prediction. The stochastic system is modelled such that sequences of *states* are achieved by transitioning between states for both finite or infinite periods. Transitions between states are stochastic, and the probability of transitioning from one state to another is based on a prescribed stochastic transition matrix. What distinguishes MDPs from other similar Markov models is the reward and control elements. *Rewards* simply represent a real-valued object (e.g. dollars) incurred to the agent upon transitioning from one state to another, thereby introducing a objective measure for which to assess the value or utility of a given system, or subset of its states. An *action* set is introduced to the system to achieve a level of control for which to 'steer' the stochastic system towards a goal of optimality. A given action taken from a state may influence the probability of transitioning to another, and will always influence the reward incurred upon transitioning (else, we would have a trivial solution). The construction and conceptualization of MDPs is best described in a sequential and cumulative manner, generally following the theories that comprise their historical development and formulation. In this paper, we begin with the foundational principles of MDPs introduced by Andrey Markov (see section 2.2) and proceed thereon, iteratively introducing concepts to our model that allow the agent to model and solve increasingly complex stochastic problems.

2.2. **History.**

Since their formal introduction in the 1960s, MDPs have been studied extensively and applied across many disciplines (Hu and Yue, 2008). The foundational mechanistic underpinnings of MDPs can be traced back much further to the works of their namesake; Russian mathematician Andrey Markov. In the 1900s, Markov introduced a concept relating to the probabilistic convergence properties of dependant random sequences, wherein a state-transition in a stochastic system is only dependant on the current state. His work was subsequently termed as the *Markov*

Property with the related extension describing sequences of random variables exhibiting the property being the *Markov Process* (aka *Markov Chains*) (see section 3.3). In 1954, Richard Bellman adopted the concepts introduced by Markov and extended the ideas to develop and introduce the *Markov Decision Process* (Bäuerle and Rieder, 2011). Through later works, Bellman introduced his principal of optimality and related dynamic programming methods for which to optimize MDP models in terms of expected value (Bellman, 1957, 1958). Important subsequent developments in the field came through Ronald Howards works on decision theory, offering an alternative optimal solution method for MDPs (Howard, 1960), and through David Blackwell (Blackwell, 1965), who provided several contributions regarding discounted state-occupancy measures and optimal policies, and introduced the MDP model formulation we most-commonly see today.

2.3. **Applications.**

From their introduction the 1950s, MDPs have since seen fairly wide-spread adoption throughout many fields. In some sense, whenever there is a need for decision making under uncertainty, a MDP model can be employed as a useful tool. Notably, MDPs are used extensively in the field of finance, where one can model for an optimal decision-making policy to help guide a portfolio management (i.e. re-balancing and asset allocation) or trading strategy (Bäuerle and Rieder, 2011). Applications exist in precision agriculture to optimize crop yields under uncertain predictions of climatic conditions by modelling for an optimal irrigation and fertilization regime (White, 1993). Therapeutic decision making, regarding estimates of prognosis under various treatment plans available to a patient, often employ MDPs to determine an optimal plan in terms of life expectancy and quality (Beck and Pauker, 1983). MDPs are frequently deployed for operations research and supply chain management purposes, where one can model for policies to guide optimal inventory control with regard to uncertain projections of demand (White, 1993). Further applications are noted in the fields of water/natural resource management, queuing theory, marketing, automotive insurance, autonomous manufacturing, energy grid management, and sports, amongst many others (Alagoz et al., 2010; Hu and Yue, 2008; White, 1993).

2.4. **Extensions.**

While this paper focuses on discrete-time formulations of MDPs, it is noted that a plethora of variants, extensions, and adaptations of MDP models exist to employ in relevant contexts. Notably, there exists *continuous-time* MDPs, whereby state-transitions are described by differential equations and can be conceptualized as rates, as opposed to discrete jumps. Here, the agent can model decisions at any given time, rather than at discrete intervals. *Partially observable* MDPs arise when an agent does not have a complete set of information pertaining to the state space, wherein a subset of states remains unobserved. For instance, this may arise when deriving a MDP model where the longer-term drift (i.e. trend) of a stock price is not feasibly known (Bäuerle and Rieder, 2011). Two special cases of partially observable MDPs exist; *Bayesian* models, where the unobserved process is parameterized and assumed not to change over time, and *Hidden Markov* models, where the unobserved portion is assumed to evolve with the properties of a Markov Process. A *piecewise* *deterministic* MDP model is one in which the underlying process exhibits periodic random jumps in state-evolution equilibria; for instance, when modelling over a change in economic states (i.e. recessionary vs. expansionary conditions).

3. Construction of Markov Decision Processes

3.1. **The Markov Process.**

The Markov process (also referred to as Markov chains) forms the fundamental mechanistic framework from which MDPs are constructed. The Markov process is a memoryless stochastic process that produces a sequence of random variables describing the evolution of states, each exhibiting the Markov property.

Definition 3.1. The Markov property states that the future evolution of state sequences is independent of the prior state sequence leading to the current state. In other words, outcomes are only dependant on the current state. To satisfy the Markov property, a sequence of random variables $\{S_n\}$ must exhibit:

$$P\{S_n = s_n \mid S_{n-1} = s_{n-1}, \dots, S_0 = s_0\} = P\{S_n = s_n \mid S_{n-1} = s_{n-1}\} \tag{3.2}$$

If true, the sequence $\{S_n\}$ forms a Markov process (Markov chain).

More formally, the Markov process can be formulated as a tuple:

$$(S, Q_n(s)) \tag{3.3}$$

Where:

- S denotes the finite state space, with individual states denoted by $s \in S$.
- $Q(s)$ denotes the stochastic transition matrix, which describes the probability of moving from state s to state s'.

A simple case of a Markov process can be conceptualized by Figure 3.1, below.

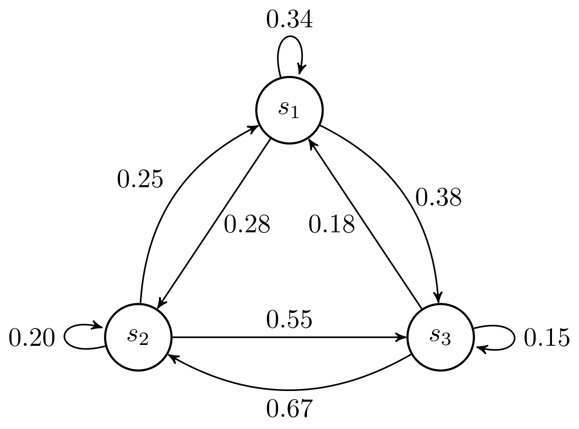

Figure 1. Conceptual example of a 3-state Markov process, with transition probabilities displayed on transition arrows.

The conceptual chain in Figure 3.1 produces the following transition matrix $Q_n(s)$:

$$Q_n(s) = \begin{vmatrix} 0.34 & 0.28 & 0.38 \\ 0.25 & 0.20 & 0.55 \\ 0.18 & 0.67 & 0.18 \end{vmatrix}$$

where the rows represent the current state s, and the columns represent the state s' in which to transition to. As such, stochastic transition matrices are square with row vectors containing probabilities summing to 1. A Markov process describes the probability of the occurence of a sequence of states $\{S_0, S_1, ...S_n\}$ over a period n; for instance, the sequence $\{S_0 = s_2, S_1 = s_3, S_2 = s_1, S_3 = s_2, S_4 = s_3\}$ will occur with probability $\prod Q_{ss'}(s) = 0.152$. While this may be useful information in certain contexts, it tells us little in regards to the expected real-world 'value' of achieving a given sequence. The proceeding section will introduce a value function to rectify this, thus providing Markov process models with a tool to improve contextual relevance and objective applicability.

3.2. **Markov Reward Processes.**

A Markov Reward Process can be conceptualized as a Markov chain with 'reward' values incurred upon transitioning between states. In its general case, a MRP is formulated by a 4-tuple, as follows:

$$(S, Q(s), r(s), g(s)) \tag{3.4}$$

Where:

- S denotes the finite state space, with individual states denoted by $s \in S$.
- $Q(s)$ denotes the stochastic transition matrix, which describes the probability of moving from state s to state s'.
- $r(s)$ denotes the reward function, describing the reward incurred by moving from state (s) to state s' at time n.
- $g_N(s)$ denotes the expected terminal reward of the system at time N, proceeding from state s at time n, where future state-transition rewards are reduced by a discount factor $\gamma \in$[0,1].

Evidently, the main difference between a Markov process and MRP is simply the inclusion of the reward function $r(s)$. This tells us that when the system transitions from state s to $s' \in S$ with probability $Q_{s,s'}(s)$, we will incur the reward specified by $r(s')$. Rewards are real-valued objects; for instance, a dollar amount or the yield of a crop in kilograms per square kilometer. The inclusion of a reward function gives the user an objective measure to maximize over a period by finding an optimal finite set of state transitions, or simply to find the expected reward of a system over a specified period. Rewards can be set in various manners; e.g. as a function of time n or of some characteristic of state sequences. However, in most contexts, they are prescribed in a static and deterministic sense, with unique values corresponding to specific state transitions.

Extending from the example given in Figure 3.1, a conceptualization of a simple Markov reward process is given in Figure 3.2, below.

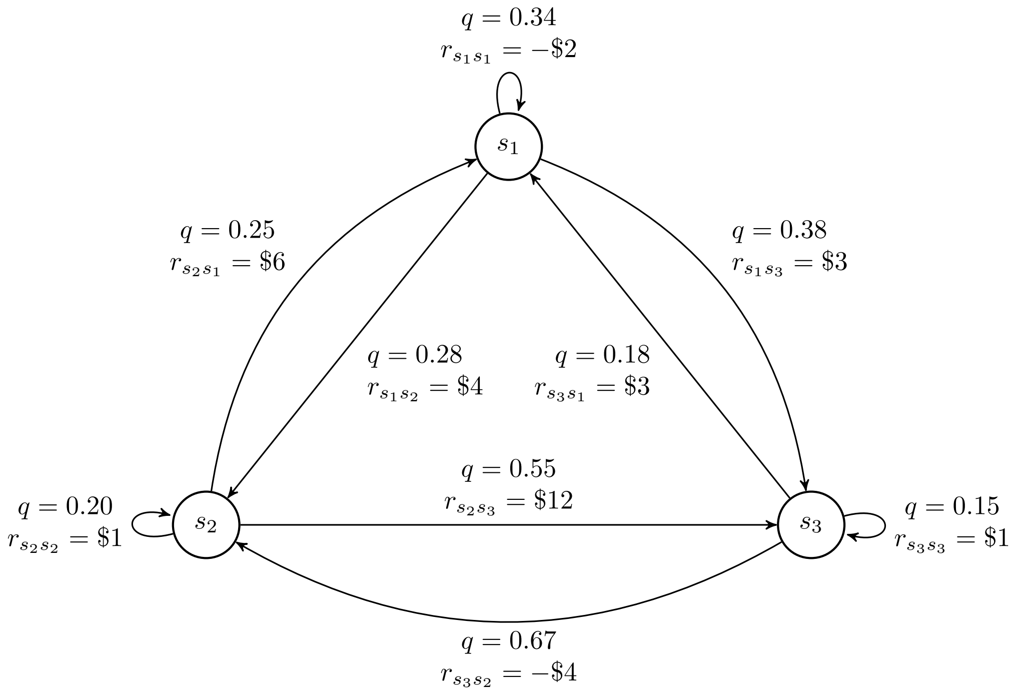

Figure 2. Conceptual example of a 3-state Markov reward process, with transition probabilities and reward values displayed on transition arrows.

The conceptual dollar-value reward function described in Figure 3.2, with state-transitions coinciding with the same $Q(s)$ as before, produces the following matrix:

$$r(s) = \begin{vmatrix} -\$2 & \$4 & \$3 \\ \$6 & \$1 & \$12 \\ \$3 & -\$4 & \$1 \end{vmatrix}$$

Considering the above, solving for an optimal state-transition sequence $\{S_0, S_1, ...S_n\}$ over a period of $n = 4$ would yield \$43 from the sequence: $\{S_0 = s_2, S_1 = s_3, S_2 = s_1, S_3 = s_2, S_4 = s_3\}$. However, since the system is stochastic, the probability of achieving this specific optimal sequence is only 1.52% ($\prod Q_{ss'}(s)$). We will see in the proceeding section how the addition of controls to the system can aid in achieving an optimal sequence (or higher expected reward) with higher probability.

3.3. **Markov Decision Processes.**

Continuing with the iterative construction, we will introduce the mechanistic concept of controlled *actions* to the Markov reward process, thereby producing a Markov decision process. We will focus here only on discrete time MDPs, although variations for continuous time exist. A Markov Decision Process, in its general case, is formulated by a 5-tuple, as follows:

$$(S, A(s), Q(s, a), r(s, a), g(s)) \tag{3.5}$$

Where:

- S denotes the finite state space, with individual states denoted by $s \in S$.
- $A(s)$ denotes the action space, with the individual actions denoted by $a \in A$.
  - $D_n(s) = \{a \in A \mid (s,a) \in D_n\}$ denotes the subset of possible actions from a given state s at time n, where D_n is the set of all possible actions across states, and is a measureable subset of $S \times A$.
- $Q(s,a)$ denotes the stochastic transition matrix, which describes the probability of moving from state s to state s', given action $a \in D_n(s)$ has been performed at state s.
- $r(s,a)$ denotes the reward function, describing the reward incurred by moving from state-action pair (s,a) to state s'.
- $g_N(s)$ denotes the expected terminal reward of the system at time N, proceeding from state s at time n, where future state-transition rewards are reduced by a discount factor $\gamma \in$[0,1].

Functionally, a MDP tells us that when the system is in state $s \in S$ and a permissible action $a \in A$ is taken, the system will transition to state $s' \in S$ with probability $Q_{ss'}(a,s)$ and incur the reward specified by $r(s',a)$. The inclusion of the action set A adds an additional layer of complexity to the model by way of a decision function; i.e. a specified control function to specify an action a to take while at state s.

Definition 3.6. A decision function is defined by a mapping $f : S \to A$ that satisfies $f(s) \in A(s)$ for $s \in S$, where A is the union of all action sets $A(s)$, and F is the set of all decision functions.

Decision functions tell us what action a is to be chosen when the system is in state s. Considering this linkage, the stochastic transition matrix is now populated by state-action pairs (s,a), rather than states alone. We can define the set of all possible state-action pairs by $\Gamma = \{(s,a) \mid s \in S, a \in A\}$. Continuing on with the conceptualized simple example, a Markov decision process framework with state-action pairs is displayed in Figure 3.3, below.

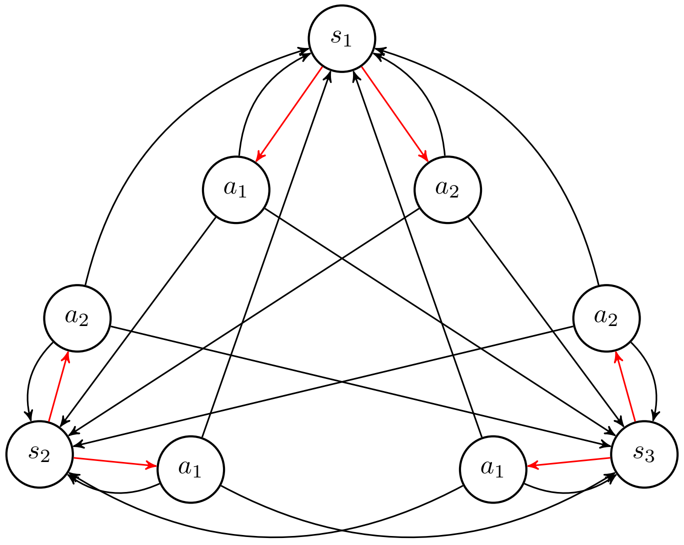

Figure 3. Conceptual example of a 3-state, 2-action MDP model. Transition probabilities are not displayed due to complexity, but would reside on transition

arrows stemming from each action towards a state. Mappings from states to actions (shown in red) are specified by a decision function $f(s)$.

3.3.1. *The Value Function.*

Let us introduce the key concept of value in the context of MDPs. Specifically, we consider the *value function*, which allows one to derive the expected reward (or 'value') of an MDP model and further determine state-action sequences that may produce a maximal reward. In essence, solving for the maximal value of the value function is, commonly, the objective of Markov decision processes. Solutions, while not achievable in a closed-form, may be obtained through various recursive algorithmic methods, outlined in section 3.3.4. The value of a state is denoted by $V(s)$, and refers to the total expected discounted reward one would obtain by following a sequence of actions and states when starting from state s. Here, the sequence of actions to be taken are specified by a policy, which is explained further in the proceeding sections. To construct the value function, let us first consider the following relation describing the total reward return G_n at time n incurred by taking an arbitrary set of actions in our MDP model:

$$\begin{aligned}
G_n(s) &= \sum_{k=n+1}^{\infty} \gamma^{k-n-1} \cdot r_k \\
G_n(s) &= r_{n+1}(s,a) + \gamma \cdot r_{n+2}(s',a') + \gamma^2 \cdot r_{n+3}(s'',a'') + \dots \\
G_n(s) &= r_{n+1}(s,a) + \gamma(\cdot r_{n+2}(s',a') + \gamma \cdot r_{n+3}(s'',a'') + \dots) \\
G_n(s) &= r_{n+1}(s,a) + \gamma(G_{n+2}(s'))
\end{aligned} \tag{3.7}$$

We can see that the total reward for a given state is determined by the immediate reward upon taking action a plus the discounted reward value of the proceeding state s' transitioned to by taking said action. This gives us a 'look-ahead' recursive relationship describing a state value. The expected value of state s is determined by summing over all possible sequences of actions, prescribed by a probability distribution π, times the respective state-transition probabilities. Thus, the expected value of state s is described as follows:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[G_n \mid S_n = s\right], \quad \forall s \in S \tag{3.8}$$

We can similarly decompose this relation into the immediate and future discounted reward values, and introducing our stochastic transition matrix $Q_n(s)$ and probability distribution π on permissible actions at state s.

$$V^{\pi}(s) = \sum_{a \in D_n(s)} \pi(s,a) \cdot r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} Q_n(s') \cdot V^{\pi}(s') \tag{3.9}$$

The above (Eq. 3.9) represents the *state-value function* form of the Bellman Equation (Bellman, 1957), which specifies that the total expected reward for a decision process, given a starting state $s \in S$, is equal to the immediate reward incurred upon taking an initial action plus the sum of the discounted future rewards incurred thereafter. Let us now consider a finite case where the system proceeds from state s until a terminal state s_N, occurring at time N. When we consider an N-stage sequence of actions, we can simplify and express the state-value function as follows:

$$V^{\pi}(s) = \mathbb{E}_s^{\pi}\left[\sum_{k=n}^{N-1} r_n(s,a) + g_N\left(s_N\right)\right], \quad s \in S, \tag{3.10}$$

The above represents the conditional expectation of the total reward in the period $[n, N]$, given we begin in state s and follow the actions prescribed by π.

We can express the value of actions at a state in a similar manner to the value of states by further decomposing Equation 3.9 and considering all possible states s' to transition to given we take action $a \in D_n(s)$. Conversely, the state-value form of this relation (Equation 3.9) considers to expected value over all possible actions we can take from a given state. This relation, termed as the *action-value function*, can be expressed as follows:

$$U^{\pi}(s,a) = r(s,a) + \gamma \cdot \sum_{s' \in S} Q_{s,s'}(s,a) \cdot \sum_{a' \in D_n(s')} \pi(s',a') \cdot U^{\pi}(s,a) \tag{3.11}$$

3.3.2. *Markov Decision Process Policies.*

We now introduce the concept of a MDP policy, denoted by $\pi \in \Pi$, and not to be confused with the often similarly-notated stationary distribution of a Markov chain. A policy can be viewed as a set of decision rules that is used to determine which action to take for any given system history and any given observation period. Policies can be constructed as a set of decision functions $f_n \in F_n$ to map states to actions based on the agent's optimization objective. Typically, it is the goal of the agent to find a policy that takes a stochastic system through a series of states (via actions) that maximizes value, as we have seen in Equation **??**.

In essence, a policy forms a probability distribution on actions. To see this, we will define an arbitrary policy $\pi = \{f_0, f_1, ...f_n\} \in \Pi$, and let $H_n = \Gamma^{n-1} \times S$ be the set of a system's history up to time n, where $H_0 = S$. Then, for any $n \geq 0$ we have $h_n = (s_0, a_0, s_1, a_1, ...a_{n-1}, s_n) \in H_n, \pi_n(f(s_n) \mid h_n)$, which forms the probability distribution on the set of actions $A(s_n)$. In other words, if system history h_n occurs at time n, then the actions are prescribed according to policy $\pi_n(f(s_n)|h_n)$. Given a system's history, policies can exhibit a variety of characteristics. For instance, we have a *Markov policy* if $\pi_n(f(s_n)|h_n) = \pi_n(f(s_n)|s_n)$, a *stationary policy* if $\pi_n(f(s_n)|s_n) = \pi_0(f(s_0)|s_n)\ \forall n \in N, s \in S$, and a *deterministic policy* if $\pi_n(f(s_n)|s_n) = 1\ \forall n \in N, s \in S$. Typically, it is the objective to derive a preferable deterministic policy, which tells the agent to take a certain action a when at a state s. Herein, we make the distinction that all reference to policies is in the context of deterministic policies, and thus rid ourselves of denoting π with respect decisions functions and states $(\pi(s, f(s))$ in favour of the commonly-adopted notation with respect to states and deterministically-selected actions $(\pi(s,a))$.

3.3.3. *Optimal Policies.*

Let us now revisit the concept of state- and action-value function in the context of Bellman's *Principle of Optimally* Bellman, 1957, which proposes that an optimal (in terms of maximal value) sequence of actions will yield a solution to a MDP wherein whichever initial state and action are selected, the actions taken from the resulting state will constitute an optimal sequence. This principle forms the general objective for MDPs; to derive the sequential set of states and actions (prescribed by a policy π) that produce the maximum expected reward for a given period. Such a policy is termed as an *optimal policy*, and is denoted as π^*. If an optimal policy π^* is adopted, the state-value function (denoted in this context as $V^*(s)$) will be optimized in regard to expected reward return for all non-terminal states:

$$V^*(s) = \max_{\pi \in \Pi} V^\pi(s) \quad \forall s \in S \tag{3.12}$$

Similarly, the optimal action-value function $U^*(s)$, under optimal policy π^*, will satisfy:

$$U^*(s,a) = \max_{\pi \in \Pi} V^\pi(s) \quad \forall s \in S, a \in A \tag{3.13}$$

When we consider all possible actions $a \in A$ available at state s, picking the action that maximizes our expected future return will thereby maximize $U^*(s)$. As such, equation 3.14 can be expressed as:

$$V^*(s) = \max_{a \in A} U^*(s,a) \quad \forall s \in S \tag{3.14}$$

Decomposing these relations further, we can express the concept of optimality in a step-wise manner. For a policy to be optimal, it must prescribe an optimal action upon arriving at each state throughout the simulated period. As such, we can express the optimal state-value as follows:

$$V^*(s) = \max_{a' \in A} \left[r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} Q_{s,s'}(s,a) \cdot V^*(s') \right] \quad \forall s \in S \tag{3.15}$$

and, for a finite MDP over period $[n, N]$, can be expressed alternatively as:

$$V^*(s) = \max_{\pi} \mathbb{E}^\pi \left[\sum_{n=0}^{N-1} r_{n+1}(s,a) + g_N(s_N) \right] \quad \forall s \in S, \tag{3.16}$$

Next, the optimal action-value can be expressed as:

$$U^*(s,a) = r(s,a) + \gamma \cdot \sum_{s' \in S} Q_{s,s'}(s,a) \cdot \max_{a' \in A} U^*(s', a') \quad \forall s \in S, a \in A \tag{3.17}$$

To finish our conceptualization of optimal policies, we will reconsider the overall objective of MDPs; to maximize our expected value for all states of the modelled system. To do so, an optimal policy π^* must maximize $V^*(s)$ over all states. Put formally:

Definition 3.18. A MDP policy $\pi^* \in \Pi$ is considered optimal if $V^{\pi^*}(s) \geq V^\pi(s) \quad \forall s \in S, \pi \in \Pi$.

To ensure a non-trivial exercises, it suffices to prove that such a policy even exists. To do so, we must first consider the following lemma.

Lemma 3.19. *For any pair of optimal policies* π_1^* *and* $\pi_2^*, V^{\pi_1^*}(s) = V^{\pi_2^*}(s)\ \forall s \in S$

Proof. The proof for the above lemma simply follows by the definition of optimal MDP policies. If π_1^* is an optimal policy, it follows that the respective state-value function $V^{\pi_1^*}(s)$ is greater than or equal to the state-value function for all other policies, including that of optimal policy π_2^*, for all $s \in S$. Similarly, the relation is implied in the inverse order by definition of optimal policies, i.e. $V^{\pi_2^*}(s) \geq V^{\pi_1^*}(s) \quad \forall s \in S$. Thus, it is implied that $V^{\pi_1^*}(s) = V^{\pi_2^*}(s) \quad \forall s \in S$. □

Theorem 3.20. *For any discrete-time Markov decision process, the following will hold true:*

(1) *There exists an optimal policy $\pi^* \in \Pi$, such that $V^{\pi^*}(s) \geq V^{\pi}(s) \quad \forall s \in S$*
(2) *All optimal policies will satisfy the Optimal State-Value Function, wherein $V^{\pi^*}(s) = V^*(s) \quad \forall s \in S$*
(3) *Similarly, all optimal policies will satisfy the Optimal Action-Value Function, wherein $U^{\pi^*}(s,a) = U^*(s,a) \quad \forall s \in S, a \in A$*

Proof. (Rao and Jelvis, 2022) To prove the above theorem, we first consider the proof of Lemma 3.19. Since we have proven that $V^{\pi_1^*}(s) = V^{\pi_2^*}(s)\ \forall s \in S$ for any two optimal policies, it follows that we must establish some deterministic optimal policy and prove it will satisfy the optimal state-value and optimal action-value functions. We will consider an arbitrary candidate optimal policy, denoted by π^c. By definition of optimal action-value, such a policy will prescribe an action a to take at state s which maximizes $U^*(s,a)$. More formally:

$$\pi^c(s) = \underset{a \in A}{\arg\max} U^{\pi}(s,a) \quad \forall s \in S$$

We will also consider the optimal state-value function $V^*(s) = \max_{a \in A} U^*(s,a)$, and can thus infer the following relation:

$$V^*(s) = U^*\left(s, \pi^c(s)\right)$$

As such, the optimal value will be obtained when we take the action prescribed by π^c, which, by our prior definition, maximizes the action-value $U^*(s,a)$. Further, this relation infers that if we take this optimal action, and subsequently transition to state s, we can then take an optimal action at state s' to again achieve the optimal state-value. This is true for each state that is not terminal. More formally, this is constructed as follows:

$$V^c(s) = V^*(s) \quad \forall s \in S$$

Similar to the proof of Lemma 3.19, the same is true when we consider the inverse order in steps; i.e. for each step, when we take an actions a prescribed by π^c, we will satisfy the optimal action-value function:

$$Q^c(s,a) = Q^*(s,a) \quad \forall s \in S, a \in A$$

The proof is completed by contradiction, where we assume our candidate policy π^c is not, in fact, optimal. Under this assumption, there must therefore exist an alternative policy (say, π^z) where at a given state s, $V^z(s)$ is greater than $V^c(s)$. Since we have previously established that $V^c(s) = V^*(s)$, the definition of the optimal value function ($V^*(s) = \max_{\pi \in \Pi} V^{\pi}(s) \quad \forall s \in S$) is contradicted when $V^z(s) > V^c(s)$. As such, our candidate policy π^c, which achieves the optimal state-value $V^c(s) = V^*(s)$, is, in fact, an optimal policy.

□

3.3.4. *Optimal Policy Solutions.*

As no closed-form solutions to Markov Decision Process value-optimization problems exist, one must obtain a solution through recursive algorithmic methods, such as the dynamic programming approaches described by Bellman (Bellman, 1958) or Howard (Howard, 1960). The approach described by Bellman is that of a backwards-induction value-iteration method. Howard takes a somewhat similar approach, but instead solves by iteratively updating policies. Various alternatives exist, as well as modifications and extensions to the value- and policy-iteration methods; however, we will focus here on these two seminal methods.

The value-iteration method is described as follows (Bellman, 1958; Rao and Jelvis, 2022): We will first consider the input as our MDP model: $(S, A(s), Q(s,a), r(s,a), g(s))$ and our desired output: the state-value function $V(s)$, which will eventually converge to an optimized condition $(V*(s))$ for all states. The agent begins by specifying an arbitrary outcome of the value function for all states. Typically, this would be zero, or a low enough value such that a true outcome of the MDP model's value function could feasibly exceed it. Next, the agent iterates over all states, for a specified number of trials or until a convergence tolerance is met, wherein for each state (iteration) the Bellman optimal state-value equation (Equation 3.16) is computed. In other words, the maximal expected reward across all permissible actions is found for each state during an iteration. As such, the values of $V(s)$ are updated upon each iteration that yields a higher value (i.e. state-value improvement, otherwise known as 'greedy' policy improvement). Eventually, $V(s)$ will converge to $V*(s)$, observed when a tolerance threshold for the change in value updates is reached. The resulting set of actions, which yield the optimal value across all states, thus comprises the optimal policy π^*.

The policy-iteration method is described as follows (Howard, 1960; Rao and Jelvis, 2022): We begin with the MDP model as before, but our desired output is now the optimal policy itself, as opposed to the optimal value function across states. The agent starts by specifying an arbitrary initial deterministic policy, which involves estimating the subsequent initial value of each state-value function under the arbitrary starting policy. In this instance, value is computed by conditioning on the iterated policy using the relation described in Equation 3.10. Here we are not computing the expected reward value at a state across all actions; rather, we are computing the expected reward value given we take the actions prescribed by the policy at hand. Policy improvement is achieved when we iterate over states, where we compute respective action-values $U^\pi(s,a)$, and update the policy with the action that achieves the maximum value:

$$U^\pi(s,a) = \underset{a \in A}{\arg\max} \left[\sum_{s' \in S} Q_{s,s'}(s) \cdot \left(r\left(s,a\right) + \gamma \cdot V^\pi\left(s'\right)\right) \right] \tag{3.21}$$

Eventually, the policy will converge to the optimal policy π^*; observable when the policy does not change (update) through successive iterations.

4. Financial Applications of MDPs

A highly common use of MDPs is for financial applications, as the evolution of asset prices is reasonably considered to be a stochastic process in itself. The classic use of a MDP in this context is to maximize the terminal wealth of some portfolio consisting of both stochastically-evolving and stationary financial assets. One can model for an optimal portfolio management strategy in regard to periodic rebalancing of assets, asset allocation among classes under specific market conditions, deployment of capital, and maximizing returns on a risk-adjusted basis (Bäuerle and Rieder, 2011; White, 1993). Strategies can be derived for both one-period and infinite cases.

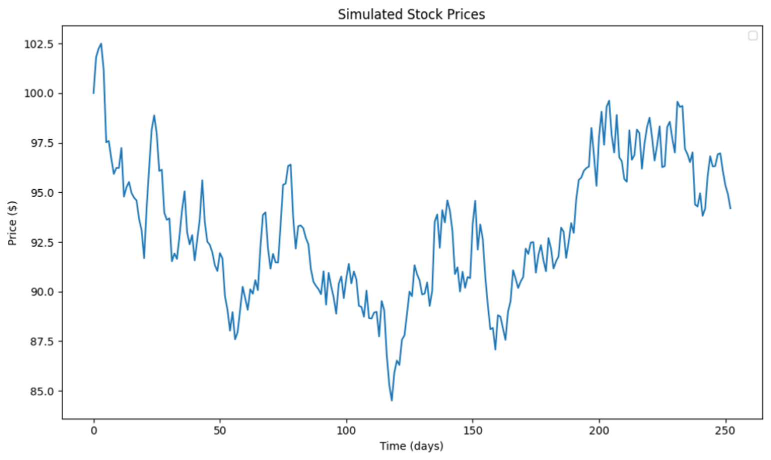

FIGURE 4. Conceptual example of a simulated asset prince using the Black-Scholes-Mertion equation.

4.0.1. *Modelling Asset Price Evolution.*

The crux of any financial (or, for that matter, any other field) application lies in modeling and constructing the stochastic nature of the system of interest; e.g. the future evolution of an asset price. Specifically, this refers to populating the stochastic transition matrix for the context of MDPs. Herein lies the source of most uncertainty in regard to validity of the MDP outcomes, as these problems are inherently difficult to accurately predict due to multitude of seemingly unpredictable factors that impact financial markets. A modeller can take one of two general categories of approaches here; i) observation of past behaviour and extrapolation into future periods, or ii) utilization of established predictive models. For i), one could presumably observe the price of an asset over a historical period and populate a stochastic transition matrix, with states being asset prices ranges, accordingly. For ii), one could employ rely an established model to generate the probability distribution of asset prices changes over discrete time-intervals, where parameters can be introduced to model price evolutions under specific economic conditions in accordance with the modeller's specific objectives. Perhaps the most well-known and widely-adopted such model is the Black-Scholes-Merton equation, which assumes an asset price S_n will evolve continuously by:

$$d(S_n) = S_n \cdot (\mu \cdot ds + \sigma \cdot B_s) \tag{4.1}$$

where μ and σ are given parameters describing the mean and variance of the asset price, and B_s is Brownian Motion. Discretized over a time-step Δt, we have:

$$S_{n+1} = S_n \cdot \exp\left(\left(\mu - \frac{1}{2} \cdot \sigma^2\right) \cdot \Delta t + \sigma \cdot B_s\right) \tag{4.2}$$

Described more simply, we have at each discretized time-step:

$$S_{n+1} = S_n \cdot (relative\ change_{n+1}) \tag{4.3}$$

A example of the outputs of the Black-Scholes-Merton equation for a stock under typical market volatility conditions is presented in Figure 4.

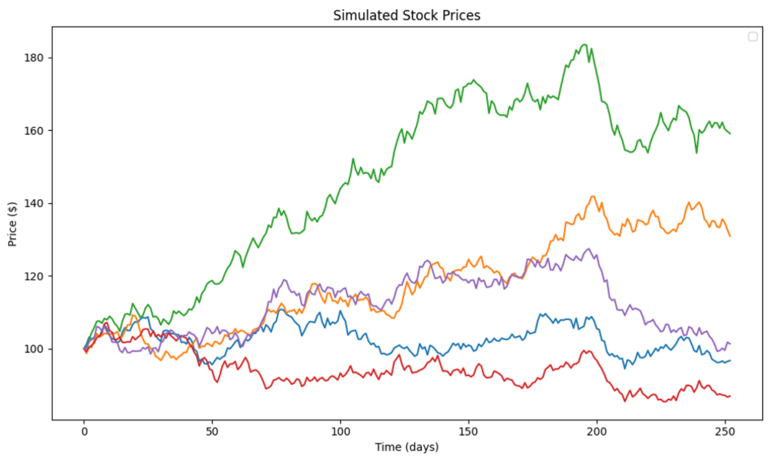

FIGURE 5. Conceptual example of a multiple correlated simulated asset princes using the Black-Scholes-Mertion equation.

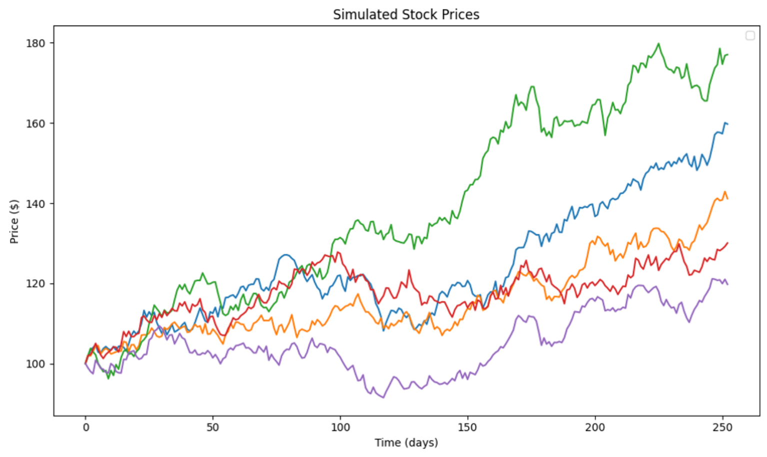

FIGURE 6. Conceptual example of a multiple correlated simulated asset princes using the Black-Scholes-Mertion equation and a positive drift factor.

One can factor in multicollinearity to more-accurately model the behaviour of multiple stocks in a portfolio (Figure 5). Further, one can model asset prices tending to a certian end price by inclusion of *drift*, thereby allowing one to investigate portfolio strategies under specific market conditions (i.e. bull-markets, bear-markets, or neutral-markets) (Figure 6).

4.1. **Example MDP Application - Portfolio Strategy.**

Here, we present a simple example to illustrate the process of developing and applying a MDP to produce a policy that optimizes the rebalancing strategy of a financial portfolio. Consider a portfolio consisting of exactly one stock or one bond

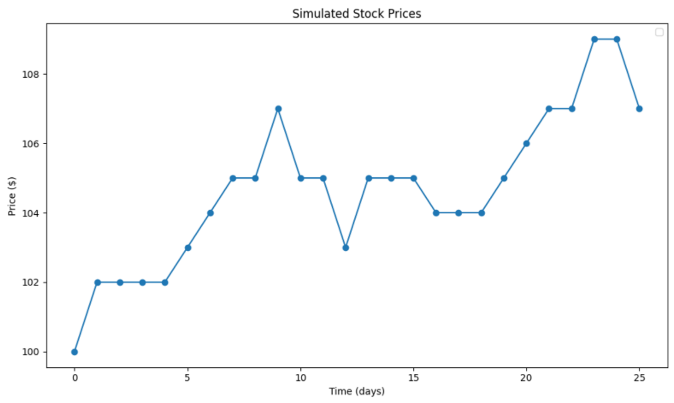

FIGURE 7. Simulated example stock price over n=25.

holding at any given time. Assume no arbitrage opportunity, no transaction costs, and for simplicity, a discount factor γ of 1 (i.e. no discounting). Rebalancing will occur after each trading day, and the rebalancing actions available to the agent are either convert all holdings to either the stock or the bond. Actions will not influence the state-transition probabilities, but will influence the rewards incurred.

Now, we must construct out stochastic transition matrix $Q_n(s)$. To do so, we will employ a somewhat trivial method of simulating our relative stock price over 25 trading days using Equation 4.2 with an arbitrary volatility condition. This will serve as the observational history from which to derive $Q_n(s)$ by the relative frequencies of each transition occurrence. To simplify, we round the resulting prices to the nearest dollar. The states of $Q_n(s)$ are then implied by each unique price in the simulated period. Of course, it is noted that we could simply populate $Q_n(s)$ directly from Equation 4.2; however, this would generate a stationary policy solution. Here, we are interested in having a policy that differs in actions among states; perhaps a certain price-point of the asset exhibits observable market-force resistance to price changes above or below. Such effects may be captured by observing the (simulated) price history and assuming the behaviour continues into the future.

The results of the stock price simulation are provided in Figure 7, and the subsequent transition matrix is presented below:

$Q_n(s) =$

s_n	$s_0 : 100$	$s_1 : 102$	$s_2 : 103$	$s_3 : 104$	$s_4 : 105$	$s_5 : 106$	$s_6 : 107$	$s_7 : 109$
$s_0 : 100$	0.0	1.00	0.000	0.000	0.000	0.000	0.000	0.000
$s_1 : 102$	0.0	0.75	0.250	0.000	0.000	0.000	0.000	0.000
$s_2 : 103$	0.0	0.00	0.000	0.500	0.500	0.000	0.000	0.000
$s_3 : 104$	0.0	0.00	0.000	0.500	0.500	0.000	0.000	0.000
$s_4 : 105$	0.0	0.00	0.125	0.125	0.500	0.125	0.125	0.000
$s_5 : 106$	0.0	0.00	0.000	0.000	0.000	0.000	1.000	0.000
$s_6 : 107$	0.0	0.00	0.000	0.000	0.333	0.000	0.333	0.333
$s_7 : 109$	0.0	0.00	0.000	0.000	0.000	0.000	0.500	0.500

Now, we will consider our reward function. First, we assume the agent will incur a reward of \$0.10 for holding a bond for a trading day. If the agent holds the stock for a trading day, they will incur a reward equal to the subsequent change in the stock's price. The reward matrices, with entries expressed as dollars, for each action will therefore be as follows.

$r_{bond}(s) =$

s_n	$s_0 : 100$	$s_1 : 102$	$s_2 : 103$	$s_3 : 104$	$s_4 : 105$	$s_5 : 106$	$s_6 : 107$	$s_7 : 109$
$s_0 : 100$	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
$s_1 : 102$	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
$s_2 : 103$	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
$s_3 : 104$	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
$s_4 : 105$	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
$s_5 : 106$	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
$s_6 : 107$	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
$s_7 : 109$	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10

$r_{stock}(s) =$

s_n	$s_0 : 100$	$s_1 : 102$	$s_2 : 103$	$s_3 : 104$	$s_4 : 105$	$s_5 : 106$	$s_6 : 107$	$s_7 : 109$
$s_0 : 100$	0.0	2.0	3.0	4.0	5.0	6.0	7.0	9.0
$s_1 : 102$	−2.0	0.0	1.0	2.0	3.0	4.0	5.0	7.0
$s_2 : 103$	−3.0	−1.0	0.0	1.0	2.0	3.0	4.0	6.0
$s_3 : 104$	−4.0	−2.0	−1.0	0.0	1.0	2.0	3.0	5.0
$s_4 : 105$	−5.0	−3.0	−2.0	−1.0	0.0	1.0	2.0	4.0
$s_5 : 106$	−6.0	−4.0	−3.0	−2.0	−1.0	0.0	1.0	3.0
$s_6 : 107$	−7.0	−5.0	−4.0	−3.0	−2.0	−1.0	0.0	2.0
$s_7 : 109$	−9.0	−7.0	−6.0	−5.0	−4.0	−3.0	−2.0	0.0

We now move to find an optimal policy solution. We will assume a short finite period of $n = 25$, equal to our price simulation period. From here, we run 5000 Monte-Carlo iterations of the policy iteration algorithm, where the policy is optimized by finding the actions at each state that maximize the action-value function. The resulting deterministic optimal policy $\pi*$ is as follows:

s_n	*Action*
$s_0 : 100$	Hold Stock
$s_1 : 102$	Hold Bond
$s_2 : 103$	Hold Stock
$s_3 : 104$	Hold Stock
$s_4 : 105$	Hold Bond
$s_5 : 106$	Hold Stock
$s_6 : 107$	Hold Stock
$s_7 : 109$	Hold Bond

Since we have assumed our stock price moves exactly according to the simulated history, it follows that an optimal action when the price is at the lower bound of \$100 is to hold a stock (i.e. we assume prices only go up from here). Similarly, the reverse is true when the price is at the upper bounding state of \$109; here it pays to hold a bond, as the stock price can only decrease. Less trivially, the policy for the remaining interior states prescribes an action that maximizes our expected reward over the period of $n = 25$ thereafter. While simple, and with many broad assumptions, this illustrative example shows how one can construct and apply a MDP to optimize a portfolio rebalancing strategy.

References

Alagoz, O., Hsu, H., Schaefer, A. J., & Roberts, M. S. (2010). Markov Decision Processes: A Tool for Sequential Decision Making under Uncertainty. *Medical decision making : an international journal of the Society for Medical Decision Making*, *30*(4), 474–483. https://doi.org/10.1177/0272989X09353194

Bäuerle, N., & Rieder, U. (2011). *Markov Decision Processes with Applications to Finance*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-18324-9

Beck, J., & Pauker, S. G. (1983). The Markov Process in Medical Prognosis. *Medical Decision Making*, *3*(4), 419–458. https://doi.org/10.1177/0272989X8300300403

Bellman, R. (1957). A Markovian Decision Process. *Indiana University Mathematics Journal*, *6*(4), 679–684. https://doi.org/10.1512/iumj.1957.6.56038

Bellman, R. (1958). Dynamic programming and stochastic control processes. *Information and Control*, *1*(3), 228–239. https://doi.org/10.1016/S0019-9958(58)80003-0

Blackwell, D. (1965). Discounted Dynamic Programming [Publisher: Institute of Mathematical Statistics]. *The Annals of Mathematical Statistics*, *36*(1), 226–235. https://doi.org/10.1214/aoms/1177700285

Howard, R. A. (1960). *Dynamic programming and Markov processes* [Section: 136 pages illustrations 24 cm]. Technology Press of Massachusetts Institute of Technology [Cambridge].

Hu, Q., & Yue, W. (2008). *Markov decision processes with their applications*. Springer.

Rao, A., & Jelvis, T. (2022, October). *Foundations of Reinforcement Learning with Applications in Finance* (1st ed.). Chapman; Hall/CRC. https://doi.org/10.1201/9781003229193

White, D. J. (1993). A Survey of Applications of Markov Decision Processes. *44*(11).