# Comparison of optimized Markov Decision Process using Dynamic Programming and Temporal Differencing – A reinforcement learning approach



# Comparison of optimized Markov Decision Process using Dynamic Programming and Temporal Differencing – A reinforcement learning approach

**Annapoorni Mani[1], Shahriman Abu Bakar[2], Pranesh Krishnan[3], Sazali Yaacob[4]**

[1]Ph.D. Research Scholar, British Malaysian Institute Universiti Kuala Lumpur, Gombak, Selangor, Malaysia
[2]Professor Madya, Faculty of Mechanical Engineering Technology, Universiti Malaysia Perlis, Perlis, Malaysia
[3]Talent and Content AI Instructor, Skymind Holdings Berhad, Kuala Lumpur, Malaysia
[4]Professor and Dean, Malaysian Spanish Institute Universiti Kuala Lumpur, Kulim, Kedah, Malaysia

Email: shahriman@unimap.edu.my; annapoorni.pranesh@gmail.com

**Abstract**. Reinforcement learning is one of the promising approaches for operations research problems. The incoming inspection process in any manufacturing plant aims to control quality, reduce manufacturing costs, eliminate scrap, and process failure downtimes due to non-conforming raw materials. Prediction of the raw material acceptance rate can regulate the raw material supplier selection and improve the manufacturing process by filtering out non-conformities. This paper presents a Markov model developed to estimate the probability of the raw material being accepted or rejected in an incoming inspection environment. The proposed forecasting model is further optimized for efficiency using the two reinforcement learning algorithms (dynamic programming and temporal differencing). The results of the two optimized models are compared, and the findings are discussed.

## 1. Introduction

Under the umbrella of Artificial Intelligence, machine learning is the hub for several promising algorithms in the recent past which has helped to restructure the problem-solving approach. In contrast to supervised learning and unsupervised learning techniques, reinforcement learning (RL) interacts with the environment. RL tries to model the problems like the way that humans live their lives [1]. Based on the learning methods, the RL can be grouped into value-based, policy-based, model-based, and imitation learning. The methods have their own variants and incremental improvements and are driven by a major set of challenges. Planning, exploration, generalization, data efficiency, temporal abstractions, training stability, scalability, lifelong learning, credit assignment, and sparse rewards are some of the common challenges in the reinforcement learning field [2-3].

Based on the policy search, the RL algorithms can be broadly divided into model-based and model-free. As per the names, model-based methods use a model to solve the problem, whereas model-free methods do not use any models [4]. Dynamic Programming, Temporal Difference Learning, Q-Learning, Monte Carlo Learning, and Actor-Critic Learning are some of the common RL algorithms.

DP is a model-based approach where it already has the MDP and only needs to find out what to do, whereas TP is a model-free approach where it does not require the knowledge of a model of the world. TP is simulation-based and learns by bootstrapping [5]. The advent of deep learning has significantly accelerated research in RL using deep learning algorithms, creating a subfield of deep reinforcement learning [6-7]. RL algorithms that are inspired by behavioral psychology learn from trial and error by interacting with the stochastic environment. RL problems are built upon the concept of the Markov decision process (MDP).

RL learns how to optimize the agent's behavior based on the presence or absence of some reward. In the past decade, due to the advancement of computing power and mathematical algorithms, there is more light thrown on reinforcement learning. RL has been studied in many fields such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, and statistics.

Paraschos et al. [6], in their proposed RL-based approach, presented an optimal joint control policy. In the manufacturing field, quality inspection is estimated using supervised machine learning models. This paper attempts to apply reinforcement learning to optimize a quality prediction model developed based on the RFID data gathered in an automotive manufacturing pipeline. The incoming inspection process in any manufacturing plant aims to control quality, reduce manufacturing costs, eliminate scrap, and process failure downtimes due to the non-conforming raw materials. Prediction of the raw material acceptance rate can regulate the raw material supplier selection and improves the manufacturing process by filtering out non-conformities.

MDP model for part quality inspection is optimized using two reinforcement learning techniques. The remainder of this paper is organized as follows. Section 2.1 introduces to reinforcement learning and the Markov decision process. Section 2.2 details the incoming inspection case study along with the finite state machine diagram. Section 2.3 describes the dynamic programming (DP) algorithm, value iteration, and policy iteration. Section 2.4 details the temporal difference (TD) learning approach. Section 3 tabulates and discusses the results of the DP and TD models, and finally, Section 4 concludes the paper with the findings.

## 2. Model Development

This section describes the steps involved in optimizing the Markov decision process using reinforcement algorithms.

### *2.1. Reinforcement learning and Markov decision process*

RL, often known as a semi-supervised learning technique, is a subset of machine learning. RL as a framework assists model decision-making and has widespread attention in game-playing as shown in Figure 1. The RL can be described as an MDP where it consists of a set of states, actions, transition dynamics, reward functions, and discount factors. When the MDP is episodic (states are reset after each episode), then the states, actions, and rewards in an episode form a trajectory of the policy. The goal of the RL was to find the optimal policy which achieves the maximum expected Return from all states [7-8]. The fundamental units of RL are Environment, Agent, Action, State, and Reward.

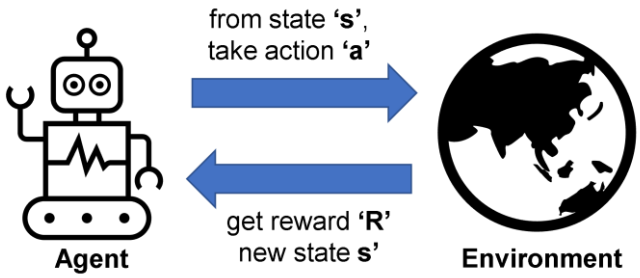


**Figure 1.** Reinforcement learning.

The very fundamental reinforcement algorithms used to solve any MDP problem are (1) Dynamic programming (DP), (2) Monte Carlo (MC), and (3) Temporal difference (TD) as shown in Figure 2. When the model dynamics such as transition probability and reward probability are known, the go-to algorithm is DP. In DP, we require a complete and accurate model of the environment. It requires all the previous states. The MDP problems can be both episodic and non-episodic or continuous tasks. Episodic tasks have a terminal or end state. The DP can be applied to both episodic and non-episodic or continuous tasks. MC algorithm requires the only experience such as sample sequences of states, actions, and rewards from online or simulated interaction with an environment. Unlike DP, it requires no prior knowledge of the environment. TD algorithms require no model, and hence the agent learns to predict the expected value of a variable occurring at the end of a sequence of states. The TD method allows the learned state-values to guide actions that subsequently change the environment state. The case study chosen to implement the algorithm is non-episodic, and hence the MC approach is not explored in this research work. The case study is solved in both the DP and using the TD approach.

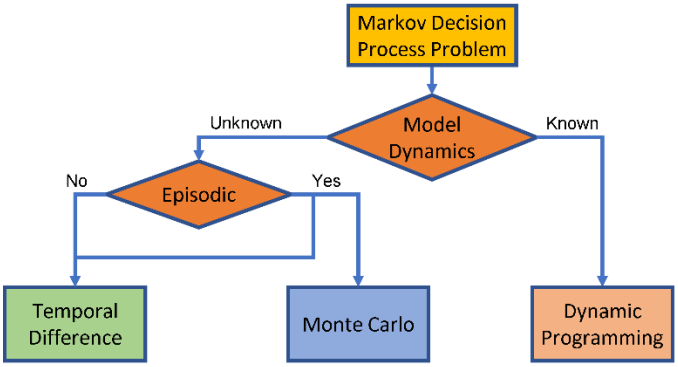


**Figure 2.** Topology of reinforcement learning algorithms

*2.2. An incoming inspection case study*

Original Equipment Manufacturers (OEM) procure material from outside have agreements with the vendor to supply material or products as per the required dimension. These agreements, in general, specify the quality of the component both in terms of aesthetics (appearance, color, texture), fit, and function. In any OEM, the very initial process is to make the incoming raw materials undergo a set of Quality assurance process known as Incoming inspection (IIP). This process ensures that the material from the vendor meets the expected quality defined by the OEM and accepted by the vendor. In this case study, seven stations in a typical IIP are considered, namely: Package and store (PI), Visual Inspection (VI), Gauge inspection (GI), Rework 1 (RW1), Rework 2 (RW2), Return to vendor (RT) and Pack & store (PS).

*2.3. Dynamic Programming and Bellman Equation*

When the model dynamics (transition probability and reward probability) are known, dynamic programming may be used to break down an optimization issue into smaller sub-problems and store the solution to each sub-problem so that each sub-problem is only solved once.

$$Q(s,a) = P^{a}_{ss'} \times (R^{a}_{ss'} + \gamma Vs) \quad (1)$$

where $Q(s,a)$= Value of state
$P^{a}_{ss'}$= transition probability moving to next state(s') from state (s) for action (a)
$R^{a}_{ss'}$ = reward probability for getting into the state (s') from state (s) for action (a)
$\gamma$= discounted factor
$Vs$ = value function

The optimal value function and the optimal policy must be estimated to solve the MDP problem using dynamic programming. In value iteration, the value function is initialized randomly in the beginning. The numerical values that the agent obtains for completing some action at some state(s)

in the environment are referred to as rewards. Based on the agent's activities, the numerical value might be positive or negative. In real life, we are more concerned with enhancing the cumulative reward (all of the rewards the agent receives from the environment) than the reward the agent receives from the current situation (also called immediate reward). The entire amount of reward received by the agent from the environment is referred to as returns [9-10].

*2.3.1. Bellman Equation*

The Bellman equation is the fundamental step in solving reinforcement learning problems. By estimating the optimal policy and value functions, the bellman equation helps us to solve the Markov decision process. The bellman equations can be solved using two types of algorithms, namely: value iteration and policy iteration. To solve the Bellman equation, we use dynamic programming.

$$V*(s) = max_{\pi} V^{\pi}(s) \quad (2)$$

where $V*(s)$ = optimum value function
$\pi$ = policy
$V^{\pi}(s)$ = value function for policy $\pi$

*2.3.2. Value iteration*

The value function or V-function or otherwise referred to as state-value function, simply measures the goodness of the agent in any given state. The value function is initialized randomly, and through iteration, the value function is optimized and is updated in the value table.

*2.3.3. Policy iteration*

In policy iteration, the actions to be taken by the agent need to be decided or initialized first. The value table is then created according to the policy. The process involves policy evaluation and policy improvement. After computing the optimal value function, the optimal policy is estimated using the following steps.

1. The policy is initialized randomly.
2. Calculate the value function V(s) for the random policy.
3. If the policy is optimum, stop the process.
4. Else, continue until the policy is improved.

*2.4. Temporal Difference Learning*

Temporal Difference learning combines the benefits of both DP and MC algorithms. DP uses bootstrapping technique to calculate the value of a state using the Bellman equation. However, to apply DP, we need to know the model dynamics of the environment. In contrast, MC is a model-free approach but cannot be applied for non-episodic tasks. In TD, bootstrapping is applied to compute the state value or Q-value, and unlike the MC methods, TD can be applied for non-episodic tasks [11-12].

*2.4.1. TD prediction using update rule*

In TD prediction, the estimated new policy is the difference between the target and the old estimate multiplied with the learning rate and finally summed up with the old estimate. TD follows the TD update rule.

$$V(s) = V(s) + \alpha(r + \gamma V(s') - V(s)) \quad (3)$$

Where V($s$)= new estimate
$\alpha$ = learning rate
$r$ = reward
$\gamma$ = discounted factor
$V(s')$= old estimate

*2.4.2. TD control*

In TD control, the goal is to find the optimal policy in the absence of the policy as input. Therefore, a random policy is initialized, and the optimal policy is found iteratively.

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \times maxQ(s',a) - Q(s,a)) \quad (4)$$

Where $Q(s,a)$= value of state
$\alpha$ = learning rate
$r$ = reward
$\gamma$ = discounted factor
$Q(s',a)$= value of next state

Table 1 signifies the assumed reward and discounted rewards values for optimum value estimation.

**Table 1.** Reward and Discounted reward

| ***Station*** | | ***Reward*** | ***Discounted Reward*** |
|---|---|---|---|
| Package Inspection | **PI** | 0.6 | 0.7 |
| Visual Inspection | **VI** | 0.7 | 0.8 |
| Rework 1 | **RW1** | -0.3 | 0.4 |
| Gauge Inspection | **GI** | 0.8 | 0.9 |
| Rework 2 | **RW2** | -0.2 | 0.2 |
| Pack and Store | **PS** | 0.9 | 0.9 |
| Return | **RT** | -0.4 | 0.5 |

For every state-action pair, the Q(s, a) is computed, and the maximum of the Q(s, a) is chosen. This process is continued for all the states and their state-action pairs, and the results are tabulated in Table 14. From the final q – function, the action with maximum value is chosen, which is the optimal policy.

Table 14 shows the results of the Q(s, a) for the initial iteration. After running several iterations, the results of the maximum Q(s, a) are compared

## 3. Results and Discussion

This section contains information on the results of the Q-learning algorithm to estimate the optimal strategy to get the maximum reward. In the DP approach, we first compute value iteration to find the optimum value function. Further to that, a random policy is initialized and using the optimal value function. The iteration is carried until the optimum policy is reached. To achieve the optimum value function and the optimal policy through the DP approach, we need to have the model dynamics such as transition probability and reward probability. In the TD approach, both the optimum value function and optimal policy are estimated without any prior knowledge of the model dynamics. In both approaches, the learning rate was fixed at 0.3 for ease of explanation. In Table 1 presented in section 2.5, the reward values and discounted reward values are assumed arbitrarily.

**Table 2.** Estimation of q-value using dynamic programming – value iteration

| ***State*** | ***Action*** | | ***Next state*** | ***Reward Probability*** | ***Discounted Reward*** | ***Q(s,a)*** | ***After 'n' iterations*** |
|---|---|---|---|---|---|---|---|
| PI | 1 | Accept | VI | 0.7 | 0.7 | 0.595 | 0.54 |
| | 0 | Reject | RT | 0.1 | 0.1 | 0.055 | 0.05 |
| VI | 1 | Accept | GI | 0.8 | 0.8 | 0.72 | 0.67 |
| | 0 | Reject | RW1 | -0.3 | 0.4 | -0.05 | 0.19 |
| **GI** | **1** | **Accept** | **PS** | 0.9 | 0.8 | 0.81 | **0.75** |
| | 0 | Reject | RW2 | -0.2 | 0.4 | 0.02 | 0.12 |
| RW1 | 1 | Accept | VI | 0.7 | 0.6 | 0.5 | 0.5 |
| | 0 | Reject | RT | 0.1 | 0.1 | 0.055 | 0.05 |
| RW2 | 1 | Accept | GI | 0.8 | 0.6 | 0.58 | 0.51 |
| | 0 | Reject | RT | 0.1 | 0.1 | 0.055 | 0.05 |

Table 2 showcases the results of the q-values estimated using dynamic programming. After estimating the Q(s, a) with the model dynamics, the process is continued through a number of iterations to finally achieve the optimal value function. For the assumed reward and discounted reward values, the DP algorithm reached the optimum value function at the seven iterations. The optimality is assessed when there is no or less change between the iterated values function.

**Table 3.** Estimation of optimum policy using policy iteration

| *State-Action pair* | *Optimum q - value* |
|---|---|
| PI-VI | 0.54 |
| PI-RT | 0.05 |
| VI-GI | 0.67 |
| VI-RW1 | 0.19 |
| **GI-PS** | **0.75** |
| GI-RW2 | 0.12 |
| RW1-VI | 0.5 |
| RW1-RT | 0.05 |
| RW2-GI | 0.51 |
| Rw2-RT | 0.05 |

As detailed in section 2.4.3, policy iteration is a process of finding the value function for the optimum policy. Thereby after finding the value iteration, the optimum policy is estimated by step-by-step evaluation of the randomly initialized policy. From the optimal value function, the maximum value (0.75) is chosen. This signifies that the agent taking action to move from GI to PS is the optimum policy.

In the attempt to calculate the optimal policy for acceptance from the Start (Package Inspection) and to reach the destination (Pack and Store), Table 4 shows the cell mapping for the substations. Four possible policies are estimated to move from Start to Destination and are tabulated from Table 5 until Table 8.

**Table 4.** TD cell mapping table for Pack and Store

| | *1* | *2* | *3* | *4* |
|---|---|---|---|---|
| *1* | PI | VI | GI | PS |
| *2* | - | RW1 | RW2 | - |

**Table 5.** TD Update rule table – policy 1

| **Cell** | (1,1) | (1,2) | (1,3) | (1,4) |
|---|---|---|---|---|
| **State** | PI | VI | GI | PS |
| **Value NS** | 0.18 | 0.336 | 0.4752 | **0.60264** |

**Table 6.** TD Update rule table – policy 2

| **Cell** | (1,1) | (1,2) | (2,2) | (1,2) | (1,3) | (1,4) |
|---|---|---|---|---|---|---|
| **State** | PI | VI | RW1 | VI | GI | PS |
| **Value NS** | 0.18 | 0.336 | 0.1452 | 0.39228 | 0.514596 | **0.6302172** |

**Table 7.** TD Update rule table – policy 3

| **Cell** | (1,1) | (1,2) | (1,3) | (2,3) | (1,3) | (1,4) |
|---|---|---|---|---|---|---|
| **State** | PI | VI | GI | RW2 | GI | PS |
| **Value NS** | 0.18 | 0.336 | 0.4752 | 0.27264 | 0.559152 | **0.664064** |

**Table 8.** TD Update rule table – policy 4

| **Cell** | (1,1) | (1,2) | (2,2) | (1,2) | (1,3) | (2,3) | (1,3) | (1,4) |
|---|---|---|---|---|---|---|---|---|
| **State** | PI | VI | RW1 | VI | GI | RW2 | GI | PS |
| **Value NS** | 0.18 | 0.336 | 0.1452 | 0.39228 | 0.514596 | 0.3002172 | 0.58909296 | **0.682365072** |

Further, to calculate the optimal policy for rejection using the TD approach, Table 9 details the cell mappings from the Start (Packaging Inspection) and to reach the Goal (Return) substation. Table 10 until Table 13 display the values of the previous state.

**Table 9.** Temporal Distance cell mapping table for Return

| | ***1*** | ***2*** | ***3*** |
|---|---|---|---|
| ***1*** | PI | VI | RW1 |
| ***2*** | - | GI | - |
| ***3*** | - | RW2 | RT |

**Table 10.** TD Update rule table – policy 5

| **Cell** | (1,1) | (3,3) |
|---|---|---|
| **State** | PI | RT |
| **Value NS** | 0.18 | **0.006** |

**Table 11.** TD Update rule table – policy 6

| **Cell** | (1,1) | (1,2) | (1,3) | (3,3) |
|---|---|---|---|---|
| **State** | PI | VI | RW1 | RT |
| **Value NS** | 0.18 | 0.336 | 0.1668 | **0.00324** |

**Table 12.** TD Update rule table – policy 7

| **Cell** | (1,1) | (1,2) | (2,2) | (3,2) | (3,3) |
|---|---|---|---|---|---|
| **State** | PI | VI | GI | RW2 | RT |
| **Value NS** | 0.18 | 0.336 | 0.4752 | 0.30288 | **0.092016** |

**Table 13.** TD Update rule table – policy 8

| **Cell** | (1,1) | (1,2) | (1,3) | (1,2) | (2,3) | (3,2) | (3,3) |
|---|---|---|---|---|---|---|---|
| **State** | PI | VI | RW1 | VI | GI | RW2 | RT |
| **Value NS** | 0.18 | 0.336 | 0.1668 | 0.35676 | 0.489732 | 0.3149208 | **0.10044456** |

Prediction of the outcome at the time (t+1) is better than the prediction at the time(t). Hence use the later prediction (t+1) to adjust the earlier prediction at the time (t). Prediction of outcome (judgment of the package reaching the Pack and Store) at gauge inspection (GI) is better than the prediction at package inspection (PI). Hence use the score or judgment at GI to adjust the prediction at PI. When a raw material tends to move from PI to VI, there is an equal probability (initial move, hence no experience is applied) that it would either qualify at station VI to go to GI or get disqualified and sent to RT. So, the intuition that the raw material would get qualified at VI (t+1) is better than the intuition at PI (t). Meaning, the reality or the fact about the quality of the raw material is more evident at each of the later stages than the initial stage.

To be more precise, consider Table 14 to explain the estimation of policy 2 displayed in Table 6. If we assume that the value of the previous state is '0' when the raw material enters VI from PI, but when the same raw materials re-enter station VI from Rework 1, the value of the state changes from '0' to '0.336'. As the value of state at (t+1) is improved, the value of state at (t) should also get improved. The results of the DP and TD algorithms arrive at the optimum policy.

**Table 14.** Estimation of policy 2 using TP algorithm.

| ***State*** | ***V(s)*** | ***Reward (r)*** | ***State Value V(s')*** | ***Discounted Reward (ɣ)*** | ***V(s)*** |
|---|---|---|---|---|---|
| PI | 0 | 0.6 | 0 | 0.7 | 0.18 |
| VI | 0.18 | 0.7 | **0** | 0.8 | 0.336 |
| RW1 | **0.336** | -0.3 | 0 | 0.4 | 0.1452 |
| VI | 0.1452 | 0.7 | **0.336** | 0.8 | 0.39228 |
| GI | 0.39228 | 0.8 | 0 | 0.9 | 0.514596 |
| PS | 0.514596 | 0.9 | 0 | 0.9 | 0.6302172 |

By comparing the results of the dynamic programming model and the Temporal Differencing models presented above, it is evident that the models produce similar results. However, the choice of the model depends on the availability of the model dynamics. DP algorithm is chosen when all the model parameters are known in advance, while TD algorithm is chosen otherwise when the model dynamics are unknown.

## 4. Conclusion

This paper presented the Markov model developed to estimate the probability of the raw material being accepted or rejected in an incoming inspection environment. The proposed forecasting model is further optimized for efficiency using the two reinforcement learning algorithms (dynamic programming and temporal differencing). The TD models exploit the Markov property where the future states depend on the current state. This nature of the TD is more efficient in Markov environments. The results of the two optimized models are compared, and the findings are discussed. When the model dynamics are not known, and when we do not have any information about the environment and when there is a need to explore all possible policies, the go-to algorithm is the Monte Carlo approach. In MC, all the possible paths are attempted, and the search is extensive.

## Acknowledgments

The research has been carried out under the Malaysian Technical University Network (MTUN) Research Grant by Ministry of Higher Education of Malaysia (MOHE) under a grant number of (9028-00005) & (9002-00089) with the research collaboration with thanks to Center of Excellence Automotive & Motorsport and Faculty of Mechanical Engineering Technology, Universiti Malaysia Perlis (Malaysia) for their productive discussions and input to the research.

## References

[1] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.
[2] Barto, A. G., Thomas, P. S., & Sutton, R. S. (2017). Some recent applications of reinforcement learning. In Proceedings of the Eighteenth Yale Workshop on Adaptive and Learning Systems.
[3] Szepesvári, C. (2010). Algorithms for reinforcement learning. Synthesis lectures on artificial intelligence and machine learning, 4(1), 1-103.
[4] Deisenroth, M. P., Neumann, G., & Peters, J. (2013). A survey on policy search for robotics. Foundations and trends in Robotics, 2(1-2), 388-403.
[5] Fenjiro, Y., & Benbrahim, H. (2018). Deep reinforcement learning overview of state of the art. Journal of Automation Mobile Robotics and Intelligent Systems, 12.
[6] Paraschos, P. D., Koulinas, G. K., & Koulouriotis, D. E. (2020). Reinforcement learning for combined production maintenance and quality control of a manufacturing system with deterioration failures. Journal of Manufacturing Systems, 56, 470-483.
[7] Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). A brief survey of deep reinforcement learning. arXiv preprint arXiv:1708.05866.
[8] Li, Y. (2017). Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274.
[9] Bertsekas, D. P. (2007a). Dynamic Programming and Optimal Control, volume 1. Athena Scientific, Belmont, MA, 3 edition. x, 1
[10] Bertsekas, D. P. (2007b). Dynamic Programming and Optimal Control, volume 2. Athena Scientific, Belmont, MA, 3 edition. x, 1, 24
[11] Gosavi, A. (2017). A tutorial for reinforcement learning. The State University of New York at Buffalo.
[12] Ravichandiran, S. (2018). Hands-on reinforcement learning with Python: master reinforcement and deep reinforcement learning using OpenAI gym and TensorFlow. Packt Publishing Ltd.