An Agent-based approach to Routing Optimisation for Santa's Gift Delivery

John McCabe and Kwun Ho Ngan
School of Mathematics, Computer Science & Engineering

1. Introduction

1.1 Domain Space and Agent Tasks

Adapting from the Kaggle competition 'Santa's Stolen Sleigh', it is aimed to apply an agent-based reinforcement learning model to assist Santa in his gift delivery route planning. The initial challenge has multiple stages – batch packing by weight and route optimization per trip. To confine our problem, the scope of work presented in this paper will be constrained in the route optimization aspect. The domain (i.e. environment) is an interactive world map with geographical location (longitude and latitude coordinates) and assigned weights of a single batch of gifts. The goal is to deliver the assigned list of gifts in the batch in the most efficient manner. Efficiency is defined by achieving through multiple objectives – (1) Weight control and (2) Travel distance minimization. These objectives have real-life significance such as fuel efficiency in drone delivery.

While a simplified version is presented for better visual illustration, the code is optimized for scaling (i.e. larger domain space) with a manual adjustment the batch quantity of gifts for simulation (Default: 9).

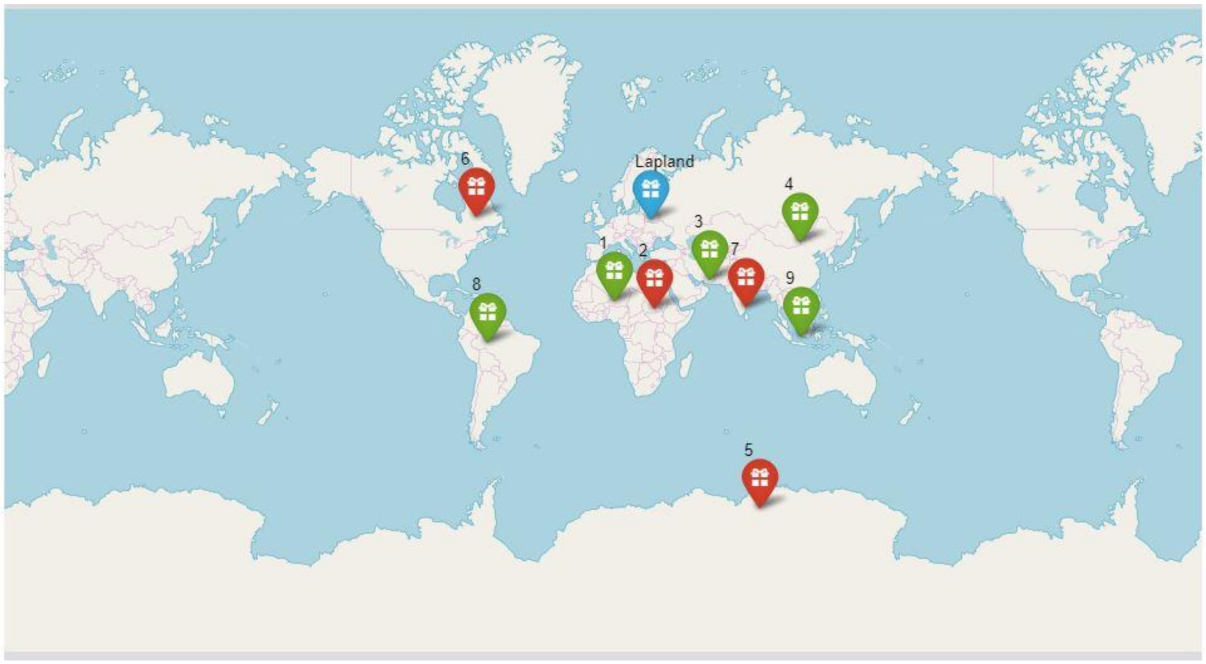

Figure 1. Illustration of domain; showing 10 locations (including starting point).

1.2 State Transition (function) during Delivery Routing

The problem proposed in this paper stems from a similar problem of the classical travelling salesperson (TSP). A decision-making process (i.e. action) is made for the transition from the current location (i.e. state) to the next location (new state). Each of the state is a set of geographical coordinates (longitude and latitude). Each route begins from Lapland (LL) with a fixed number of locations (gifts (G)) to be delivered. No single location can be visited twice.

In a scenario of 4 gifts {G1, G2, G3, G4}, a sample delivery route will have 4 state transitions as shown below. There are 24 possible routes and the optimal route will be determined based on an action selection policy with predetermined criteria defined in a reward function (discussed in the next section).

An example of sequence of states: {LL, G1, G3, G4, G2} and can be formally defined as: $s_{t+1} = \delta(s_t, a_t)$

1.3 Reward Function

The decision-making process (i.e. action) is rewarded for bringing the agent from one state to the next. A positive outcome will be given a reward while a negative outcome is punished. The aim of agent-based learning is to ensure the agent makes its decision based on learning from cumulative rewards. For example in Q-Learning, the agent would choose its action to maximize its long term reward.

In this problem set, there are 2 key objectives for achieving the desired outcome – (1) Minimizing the weight carried over the distance of delivery, and (2) identify the shortest delivery route. The reward function is thus defined by a base reward for each location delivered, a positive reward for dropping the heaviest gift first and punishing long state-to-state distance as shown in Eq. 1.

$$R(s) = base\_weight + weight\_weight * \log\big(Weight(s)\big) - distance\_weight * \log(Distance(s_{start}, s_{end})) \qquad (1)$$

The weight and distance are measured in kilogram and kilometers respectively. To adjust for the curvature of the globe, the great circle distance is used in the calculation. The logarithmic function for the weight and distance allows both terms to be scaled to similar magnitude whilst maintaining perspective (due to its monotonic nature). Base_weight, weight_weight and distance_weight are hyperparameters for refining the reward. The effect of these hyperparameters will be presented in Section 2.4.

1.4 Action Selection (Policy)

A deterministic policy for action selection is highly inefficient as all possible state transitions must be evaluated. Watkins [5] introduced Q- Learning that aims to avoid the need of a policy function by picking the best future action through reward maximization from a given state and action pair. Four Q-learning policies for choosing the next delivery location will be investigated in this paper.

(1) Random: Random choice of next location from all possible remaining locations
(2) Greedy: Always pick the location with maximum reward (in Q Matrix) whenever possible.
(3) ε-greedy: An hybrid between (1) and (2) where ε controls the proportion of reward exploitation and random exploration.
(4) Boltzmann: Probability of action selection is based on a Boltzmann distribution with a hyperparameter, τ, controlling the proportion of reward exploitation and random exploration.

In this paper, ε is initialized as 1 (fully random in selection) with a decay of 0.01 on each episode until it reaches a predefined minimum of 0.01 (i.e. a notion of exploration remains with the agent). A similar initialization is made for τ (i.e. 1 at the beginning and decay by 0.01 each episode).

1.5 Graphical Representation of States in Domain

A graphical representation of the domain with an illustrative batch of 3 gifts is shown in Figure 1. Each state is a landmark in the map with Lapland is set to be always the first point of travel.

As the reward is dependent on the weight of the gift being delivered to the next state, the rewards between states are not symmetrical (clearly shown in Figure 2); this is a break from the traditional TSP problem.

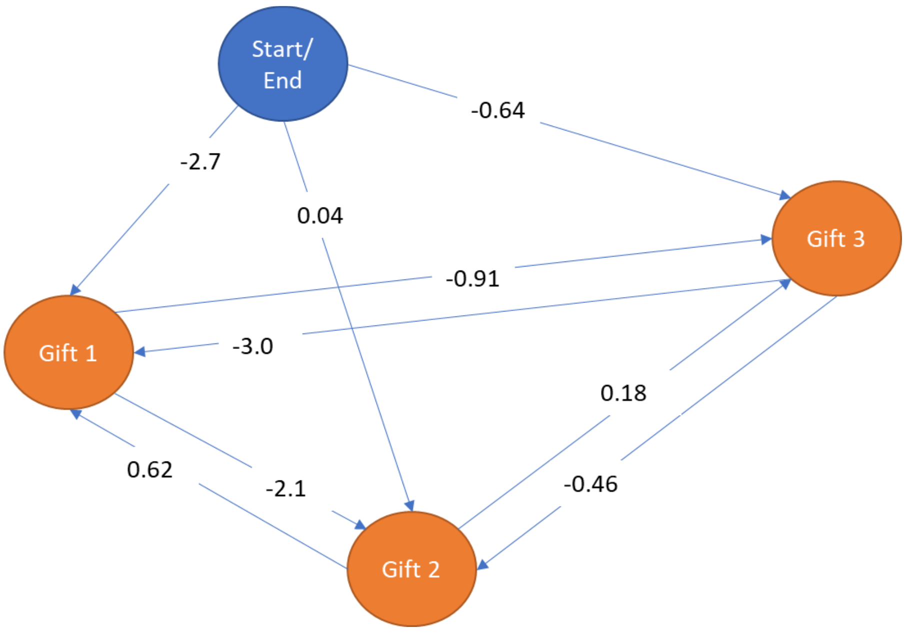

Figure 2: A graphical representation of the domain with a batch of 3 gifts (Red denotes heavy gifts. Green denotes light gifts). The possible state transition with the associated reward are also shown.

1.6 Reward Matrix (R-Matrix)

The reward matrix is constructed as a table of predefined reward when an agent transits from one state to another. The calculation of reward at each state transition is based on the reward function detailed in Eq. 1. In order to avoid any illegal move (e.g. staying at the same location/heading to Lapland), these state transitions are assigned with a negative infinity reward (-Inf). This allows the agent to learn about these constraints instead of a rule-based control. Table 1 illustrates the calculated R-Matrix of the first 3 gifts.

		Destination			
Start Point		Lapland	G1	G2	G3
	Lapland	-Inf	-2.7	0.04	-0.64
	G1	-Inf	-Inf	0.62	-0.91
	G2	-Inf	-2.1	-Inf	-0.46
	G3	-Inf	-3.0	0.18	-Inf

Table 1: Calculated R-Matrix based on reward allocation in Eq. 1. All illegal moves are assigned a negative infinite reward (-Inf).

1.7 Parameters in Q-learning

Two key parameters are found in the Q-learning algorithm [Sutton] are the learning rate, α, and the discount factor, γ. Q-value is updated based on the temporal difference (TD) prediction error correction (2nd term in Eq. 2). The two Q-Learning hyperparameters are highlighted in red and will be evaluated for their significance in model performance.

$$Q(S_t, A_t)_{new} \leftarrow Q(S_t, A_t)_{old} + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{2}$$

The learning rate, α, controls the rate of updating the TD error correction. It has a range between 0 (no learning) and 1 (aggressive update). This parameter should be optimised such that progressive update is achieved without large fluctuation in prediction error. A nominal value of 0.5 is used for policy evaluation. A grid search for the following α values (0.1, 0.3, 0.6, 0.9) will be investigated on its effect on Q-Learning.

The discount factor, γ, controls the long-term effect of current reward. It also has a range between 0 (No effect) and 1 (Full effect across all future steps in a run). Similar nominal value of 0.5 will be used for policy evaluation while a grid search over (0.1, 0.3, 0.6, 0.9) for its effect.

1.8 Update of Q-Matrix in a Learning Episode

To illustrate the update in Q-Matrix in a learning episode using Eq. 2, the same simplified batch of 3 gifts will be used. Nominal values for Q-Learning hyperparameters are set at α = 0.5 and γ = 0.5. The subsequent tables in this section will present the step-by-step update of the Q-Matrix.

Initial Q-Matrix (with R-Matrix)

The Q-Matrix is initialised with all zeros. The first episode is primarily executed by random selection of action while the Q-Matrix is updated. Assuming the random choice determines the next move (S_{t+1}) to G2, R_{t+1} is taken to be 0.04.

	Lapland	QG1	QG2	QG3	RG1	RG2	RG3
Lapland	0	0	0	0	-2.7	**0.04**	-0.64
G1	0	0	0	0	-inf	0.62	-0.91
G2	0	0	0	0	-2.1	-inf	-0.46
G3	0	0	0	0	-3.0	0.18	-inf

Based on Eq. 2, the numerical calculation is shown below:

$$Q(S_t, A_t)_{new} \leftarrow Q(S_t, A_t)_{old} + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

$$0.02 \leftarrow 0 + 0.5[0.04 + 0.5 \max_a [0, 0,0,0] - 0]$$

After 1st Update

	Lapland	QG1	QG2	QG3	RG1	RG2	RG3
Lapland	-Inf	0	0.02	0	-2.7	0.04	-0.64
G1	0	0	0	0	-inf	0.62	-0.91
G2	0	0	0	0	-2.1	-inf	**-0.46**
G3	0	0	0	0	-3.0	0.18	-inf

Assume the next move is randomly chosen to be G3, the Q-Matrix calculation is shown below.

$$Q(S_t, A_t)_{new} \leftarrow Q(S_t, A_t)_{old} + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

$$-0.23 \leftarrow 0 + 0.5[-0.46 + 0.5 \max_a[0,0,0,0] - 0]$$

After 2nd Update

	Lapland	QG1	QG2	QG3	RG1	RG2	RG3
Lapland	-Inf	0	0.02	0	-2.7	0.04	-0.64
G1	0	0	0	0	-inf	0.62	-0.91
G2	-Inf	0	-Inf	-0.23	-2.1	-inf	-0.46
G3	0	0	0	0	-3.0	0.18	-inf

Assume the next random move is selected to G1, the next Q-Matrix calculation is shown below.

$$Q(S_t, A_t)_{new} \leftarrow Q(S_t, A_t)_{old} + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

$$-1.5 \leftarrow 0 + 0.5[-3.0 + 0.5 \max_a[0,0,0,0] - 0]$$

After 3rd Update (1-episode)

	Lapland	QG1	QG2	QG3	RG1	RG2	RG3
Lapland	-Inf	0	0.02	0	-2.7	0.04	-0.64
G1	0	0	0	0	-inf	0.62	-0.91
G2	-Inf	0	-Inf	-0.23	-2.1	-inf	-0.46
G3	-Inf	-1.5	0	-Inf	-3.0	0.18	-inf

The first move in Episode 2 will begin in Lapland. the next Q-Matrix calculation is shown below.

$$Q(S_t, A_t)_{new} \leftarrow Q(S_t, A_t)_{old} + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

$$-1.35 \leftarrow 0 + 0.5[-2.7 + 0.5 \max_a[-Inf, 0, -Inf, -0.23] - 0]$$

After 1st Update (2-episode)

	Lapland	QG1	QG2	QG3	RG1	RG2	RG3
Lapland	-Inf	-1.35	0.02	0	-2.7	0.04	-0.64
G1	0	0	0	0	-inf	0.62	-0.91
G2	-Inf	0	-Inf	-0.23	-2.1	-inf	-0.46
G3	-Inf	-1.5	0	-Inf	-3.0	0.18	-inf

1.9 Performance of Q-Learning across Episodes

In the illustrative performance evaluation from the associated Jupyter Notebook (e.g. 10 gift locations), Q-Learning performance is evaluated based on the total accumulation of reward over episode. Figure 3 shows the learning phase (high exploration) and the habit formed (high exploitation).

The learning phase consists of the first few episodes where the Q-matrix is being constructed or an optimal solution has not been reached. As such, the reward accumulation over the early episode shows large variation. Once an optimal Q-matrix is established, the best route with maximum reward will always be chosen by the default greedy policy. At this phase, it can be considered a habit formed (i.e. no variation in reward is expected).

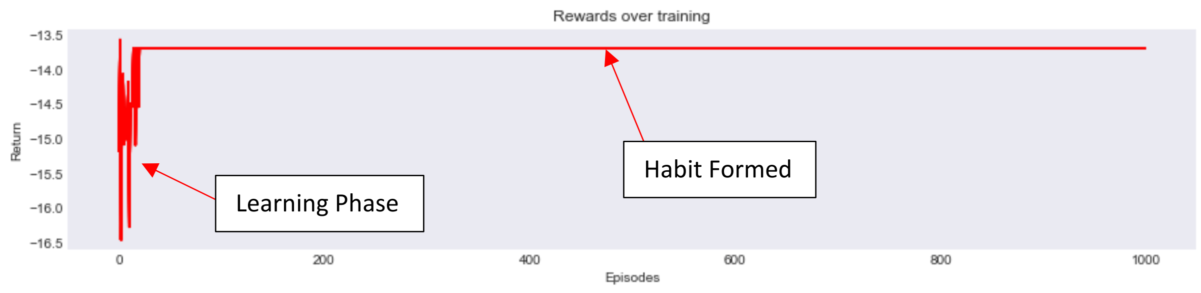

Figure 3: Performance of Q-Learning across episodes

2 Analysis of Results

In this section, a reinforcement learning simulation of 9 gift locations is used for comparison. Nominal values for hyperparameters can be assumed unless otherwise stated. The optimal route (domain best) has also been calculated based on 362880 possible combination (factorial of 9).

2.1 Policy Evaluation

The mechanism for action selection (i.e. policy) used in this study has successfully demonstrated how learning is acquired. In a random selection, it is expected that no learning will be achieved. Reward accumulation will also be random (high variation). For a greedy policy, learning is done during the first few episodes where reward is accumulated based on random choice. Once an optimal Q-matrix is established, action selection is solely based on the maximum Q-value, resulting in rapid convergence but not necessarily the optimal policy. It is not expected to observe any variation in accumulated reward (i.e. a flat line). There is also a significant step up between the lowest accumulated reward and the optimal solution. The same magnitude of step up can also be observed for ε-Greedy and Boltzmann policies. The variation in reward accumulation (after habit formed at later episodes) is due to the activation of random exploration. All policies except random selection is very close to the best solution in this domain space as shown in Figure 5.

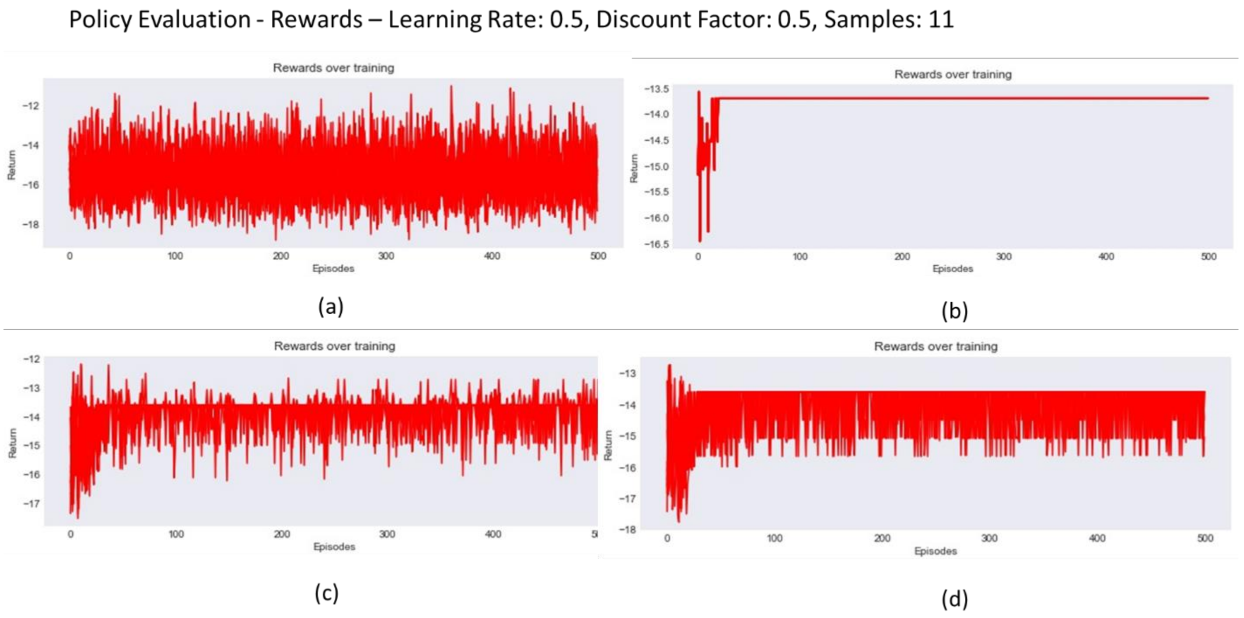

Figure 4: Reward accumulation across different policies (a) Random (b) Greedy (c) ε-Greedy (d) Boltzmann across 500 episodes

Policy Evaluation – Against Optimal – Learning Rate: 0.5, Discount Factor: 0.5, Samples: 11

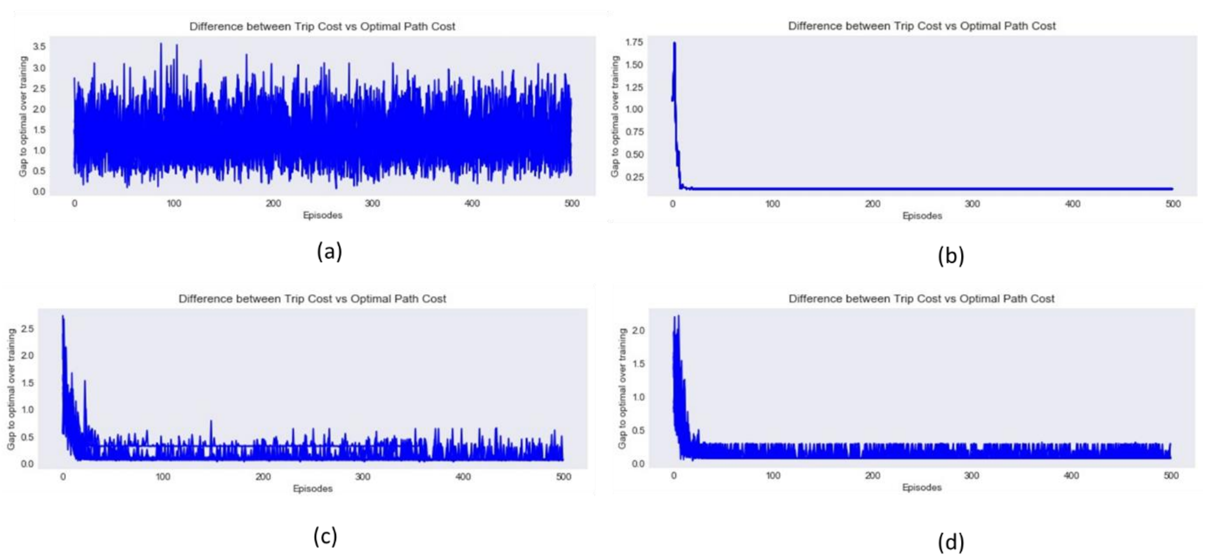

Figure 5: Difference of reward accumulation from domain best solution across different policies (a) Random (b) Greedy (c) ε-Greedy (d) Boltzmann across 500 episodes

It is unclear from the plots which policy is best performing. Table 2 shows provides insight into the how each of the policies scored against the optimal policy. The ranking is between 1 and 362880, where 1 is the optimal route and 362880 is the worst. Reviewing these scores, the ε-greedy policy provides the (marginally) best performance. Taking the plots into account the ε-greedy policy was determined to be the best policy.

Policy	Best Ranking	Worst Ranking	Median Ranking
Greedy	280	280	280
Random	1134	336097	237311
ε-greedy	98	280	98
Boltzman	98	3108	98

Table 2: Statistics of ranking scores from 11 samples for each policy. Rank is in range [1,362880].

2.2 Learning Rate

There is an interesting trend of convergence as the learning rate, α, increases. At learning rate of 0.1, it does not appear to have converged after 50 episodes. However, the optimal solution appears to converge as soon as the approximately 15th episode when learning rate is set at 0.9.

Learning Rate Evaluation – Rewards – Policy: ε-greedy, Discount Factor: 0.5, Samples: 11

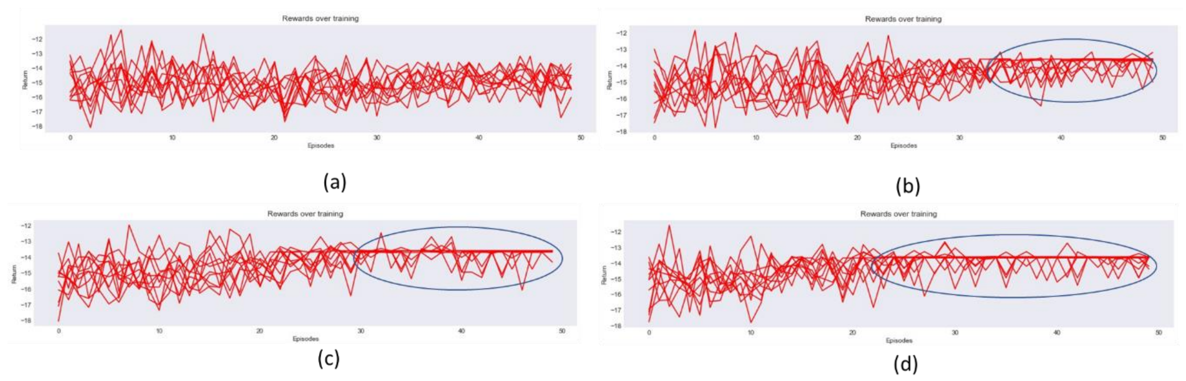

Figure 6: Convergence of Reward accumulation to optimal solution of an ε-Greedy policy by varying the learning rate from 0.1 (a), 0.3 (b), 0.6 (c) and 0.9 (d).

Table 3 shows the resulting metrics; note that these tests are run over a short number of episodes for evaluation purposes only. Each configuration converges over a large number of episodes. It is clear from the metrics that there appears to be a sweet-spot of around the 0.6 for learning rate.

Learning Rate	Best Ranking	Worst Ranking	Median Ranking
0.1	112	4565	463
0.3	98	5404	98
0.6	98	2991	98
0.9	98	2991	112

Table 3: Statistics of ranking scores from 11 samples for each learning rate. Rank is in range [1,362880].

2.3 Discount Factor

As expected, the discount rate, γ, has not shown any significant effect on the exploitation of Q-learning reward maximization. In all discount rates, the converged optimal solutions have shown similar gain in reward with same level of random variation due to exploration.

Discount Rate Evaluation – Rewards– Policy: ε-greedy, Learning Rate: 0.6, Samples: 11

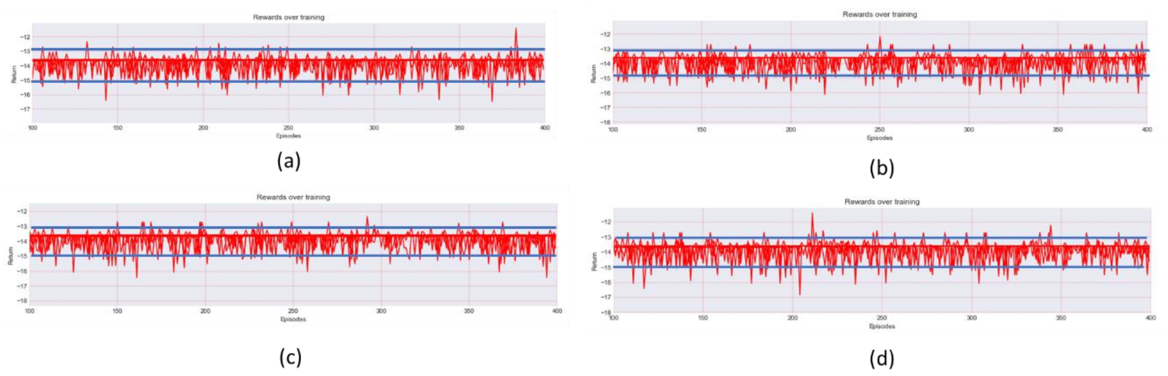

Figure 7: Reward accumulation across different discount factors (a) 0.1 (b) 0.3 (c) 0.6 (d) 0.9 across 400 episodes

The impact of changing the discount rate is inconclusive from the plots. However, the metrics provide some insight to the impact. Although for all models of discount factors have converged over the shorter 400 episodes, there appears to be a clear pattern with regards to the worst ranking score. With the higher the discount rate, it seems more likely the agent oscillates around the true optimal solution.

Discount Factor	Best Ranking	Worst Ranking	Median Ranking
0.1	98	7749	98
0.3	98	9474	98
0.6	98	11566	98
0.9	98	40833	98

Table 4: Statistics of ranking scores from 11 samples for each discount rate. Rank is in range [1,362880].

2.4 Hyperparameter in Reward Function

The effect of hyperparameters in Reward function (i.e. the component weightage) is best observed qualitatively with the optimised route. The change in component weights has a significant effect on the learning of the agent. By applying only the weight component as reward, the agent will deliver all the heaviest gifts first regardless of distance. Similarly, the distance component has successfully delivered the gifts by distance only. The base weight has no influence on delivery route by either distance and weight. It is expected that the optimised route is purely based on its learning of the random action selection in early episodes. As such, this route will change with a different training cycle. When all weights are set to 1 (i.e. top left), it has the most balanced approach where the multiple objectives are met. Gifts are delivered with preference of weight over distance as the shortest route will be achieved whenever possible.

Reward Function Evaluation – Policy: ε-greedy, Learning Rate: 0.6, Discount Factor: 0.6, Samples: 11

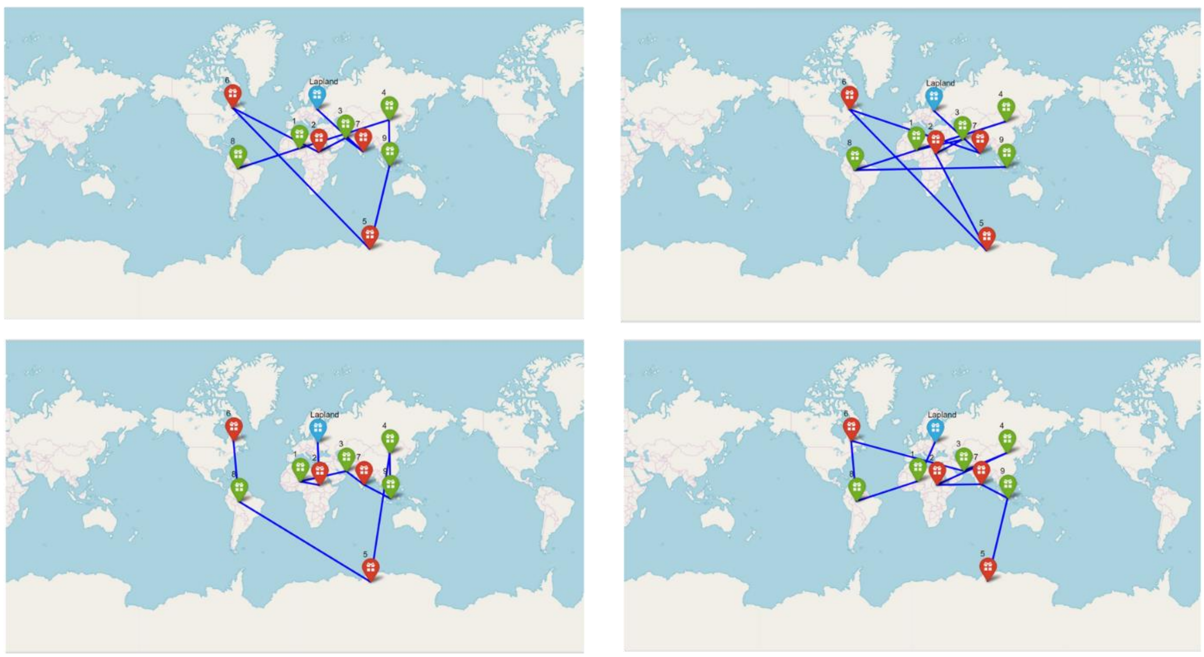

Figure 7: Effect of Hyperparameters in Reward Function (base_weight, weight_weight, distance_weight) - All Weights (1,1,1) (top left) Weight only (0,1,0) (top right) Distance only (0,0,1) (bottom left) Base only (1,0,0) (bottom right).

Having demonstrated the impact of changing the reward function parameters, the next step is to determine if there is an optimal combination of parameters. Table 5 shows the top 10 results of running the following parameter combinations:

- base_weights = [1,2,3]
- weight_weights = [0.1, 1, 2]
- distance_weights = [0.1, 1, 2]

Base_weight	Weight_weight	Distance_weight	Rank	Diff2Opt
2	2	2	98	0.083196
2	1	2	99	0.083786
1	1	1	117	0.087842
1	2	2	173	0.093586
3	1	2	244	0.097474
3	2	2	556	0.114242
1	1	2	784	0.121325
3	1	1	999	0.128803

Table 5. Rank and difference to optimal results against combinations of reward parameters.

The results appear to be quite noisy, but all the best performing combinations point to having an equal balance weight and distance. The base_weight can be viewed as a bias and as such it seems reasonable to deduce that setting the base_weight to 2 would be preferable.

2.5 The Heuristic Best Model

Based on the above parametric analysis, the heuristic best model is found using a grid search of all the parameters evaluated. It is found that the combination of an ε-Greedy policy with a learning rate and discount rate of 0.6 offers the closest resemblance to the domain best model. This best model will now be scaled to a scenario with a batch of 50 gift delivery locations. Figure 8 & 9 shows the reward accumulation and the optimal route respectively. While it takes more episodes to converge, the outcome for the optimal solution is very promising. No computation limitation is expected for even larger batch.

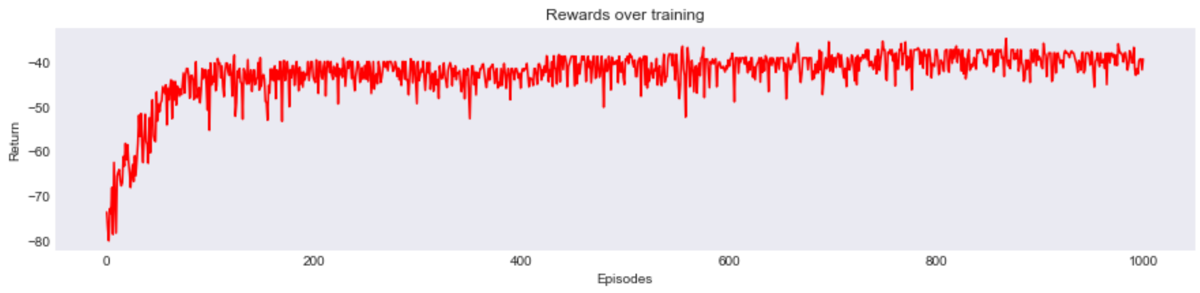

Figure 8: Reward accumulation for the best heuristic model (an ε-Greedy policy with a learning rate and discount rate of 0.6).

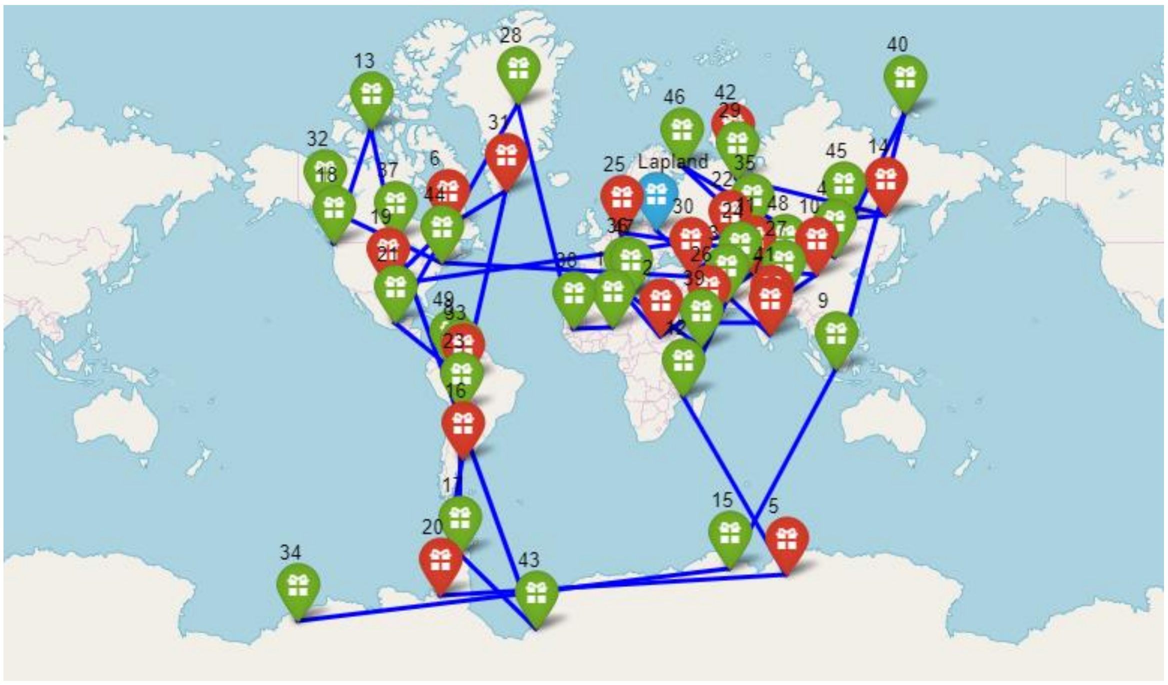

Figure 9: Optimal delivery route for the best heuristic model (an ε-Greedy policy with a learning rate and discount rate of 0.6).

3 Alternative Agent Algorithm (SARSA)

An alternative reinforcement learning model based on an on-policy SARSA algorithm. It is a slight deviation from Q-learning such that the update of Q-Matrix is based on the next state and action pair. SARSA tends to be more conservative as it will always assess the reward from all the exploratory actions. In contrary, Q-Learning will have taken the optimal path as soon as possible.

$$Q(S_t, A_t)_{new} \leftarrow Q(S_t, A_t)_{old} + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \quad (3)$$

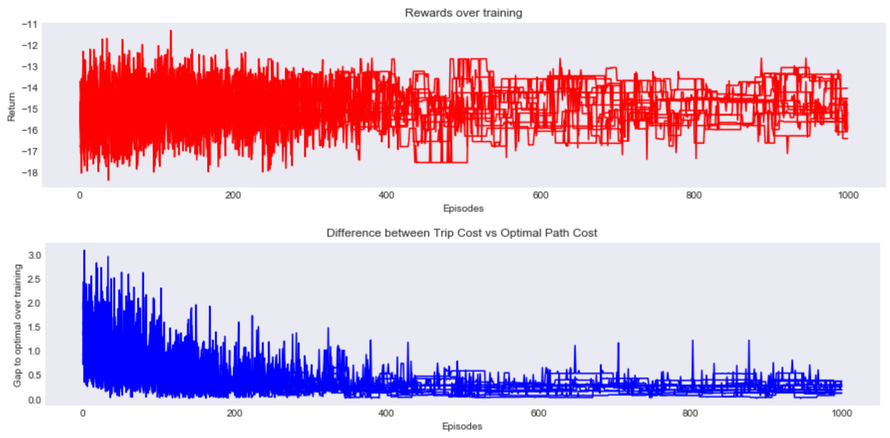

Figure 10: Reward accumulation (top) and Difference with Domain Best Model (bottom) for the SARSA-based agent (an ε-Greedy policy with a learning rate and discount rate of 0.6).

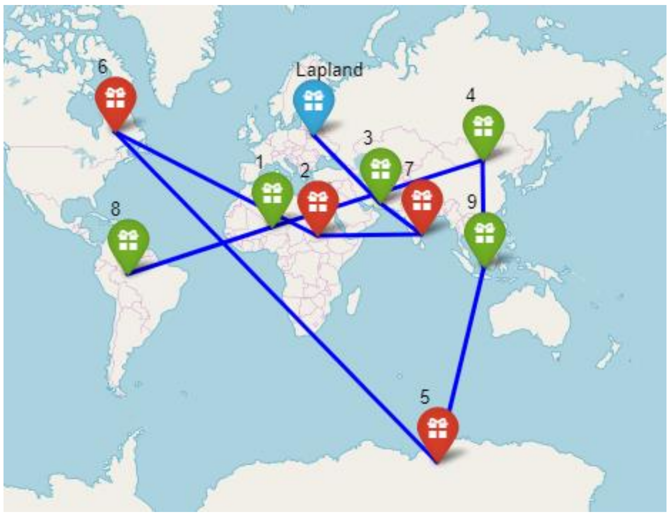

Figure 11: Optimal delivery route for the SARSA-based agent (an ε-Greedy policy with a learning rate and discount rate of 0.6).

As shown in Figure 10, the SARSA agent is able to converge close to an optimal solution (comparable to the domain best). It however does not try to take on the reward maximization path as quickly as possible. It takes on the approach to learn from each action taken with the notion to explore the domain space deeper. In a qualitative observation from the map (Figure 11), the outcome of SARSA is highly acceptable as it seems to balance the importance of distance and weight very well instead of simply putting weight as preference.

4 Multi Agent Q-Learning Algorithm

A multi-agent Q-Learning scenario is implemented. In this model, an additional state parameter is used to allow communicate between the 2 agents such that they will not pick up the same item for delivery. The code has been written for scalability and it is possible to deploy more agents to complete the tasks of delivering a large set of gifts from Santa (e.g. 100k gifts as original Kaggle challenge) in a shorter period.

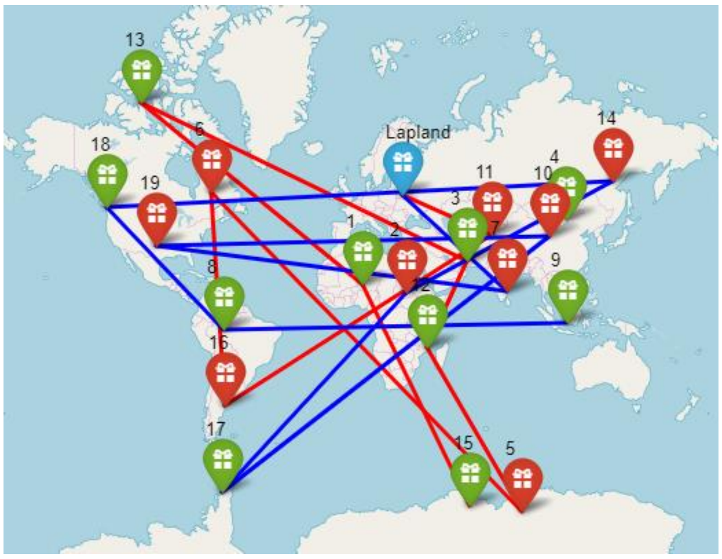

Figure 12: Optimal delivery route for multiple Q-Learning agent (parameter of individual agent is the same as best model for single agent).

5 Conclusion and Future Work

This work has presented a prototype of agent-based approach for delivery route optimisation. It has also demonstrated its scalability though the use of Q-matrix can be memory expensive with a huge list of possible states and action pairs. The outcome of the study has achieved its goal of minimizing weight in the shortest possible journey.

Through exploring the hyperparameters it has been shown that the model efficiency i.e. how quickly it reaches convergence can be optimised. This is highly valuable for larger or production systems though its effectiveness is less prominent for a small set of gifts with convergence under a few hundred episodes.

The implementation of the reward function can completely change how an agent reacts to the domain environment. This is expected as the update of Q-learning lies in the TD prediction error generated primarily based on the reward function. It is therefore recommended to have a well-defined reward function tailored to the specific optimization objective.

The modification of Q-Learning agent to a SARSA-based agent has shown how an agent can balance quick exploitation of past learning to a deeper exploratory of domain space. This is crucial especially if the objective is mission critical or life threatening (e.g. in autonomous vehicles).

As future work, the outcome of this work can be applied and extended to a multi-agent drone delivery network with different objectives for each agent. Other constraints can be included to illustrate blocking effect of component combination (e.g. weight with bad weather). A deep Q-network can also be applied for more complex state conditions for action selection.

Reference

1. Y. Niv, Reinforcement learning in the brain, The Journal of Mathematical Psychology, Vol 53(3), pp. 139 - 154, 2009.
2. I.P. Pavlov, Conditioned Reflexes: An investigation of the physiological activity of the cerebral cortex (translated), Classics in the History of Psychology, https://psychclassics.yorku.ca/Pavlov/, Viewed on 1st April 2019.
3. B. F. Skinner, The generic nature of the concepts of stimulus and response, The Journal of General Psychology, Vol 12(1), pp. 40 - 65, 1935.
4. R. S. Sutton & A. G. Barto, Reinforcement learning: An introduction, MIT Press, 1998.
5. C.J.C.H. Watkins, Learning from delayed rewards, PhD Thesis, University of Cambridge, England, 1989.

Appendix A:

Essence of Psychology in the Development of Q-Learning

Computational reinforcement learning is built upon a wealth of knowledge from behavioural psychology research in associative learning. Based on a comprehensive review by Yael Niv [1], this paper will aim to extract the key findings from stimuli conditioning theories in psychology and establish the parallelism of behavioural psychology with the ongoing development in computer agent learning (i.e. reinforcement learning). With the access of powerful modern computer, it is also interesting that computational simulation can now be used as a platform to validate some of the empirical results in behavioural conditioning experiments. While it is beyond the author's expertise to discuss the research in psychology at length, the Pavlovian (classical) conditioning and instrumental (operant) conditioning concepts will be discussed as the outcoming from these researches has formed the cornerstone for modern computational reinforcement learning development.

Pavlovian conditioning [2] highlights the empirical evidence of associative learning in that a neutral stimulus (e.g. the tone of a bell) can be conditioned to a desired outcome (e.g. the anticipatory salivation of dogs to food) through repeated conditioning experiments. Instrumental conditioning [3] provides the mean of conditioning that learning of a specific responsive behaviour (action) can be strengthened or weakened by a rewarding/punishment mechanism. Sutton & Barto [4] formulated these concepts into a computational framework to allow a more structured investigation and analysis in the conditioning of behaviour. It also provides a methodology for optimisation of decision making (i.e. best action) based on the expected outcome of long-term consequences through a process of reward maximization (or punishment reduction).

By comparing with the development of agent-based reinforcement learning, an agent should take into account the current state (i.e. stimuli) at which it extracts from the present environment and generate the appropriate decision making (i.e. agent policy) to reach the next state through learning from the expectation of long-term rewards. An example of this approach will be elaborated in the next section where an error correction mechanism presents how learning is achieved algorithmically and the formation of habits by continual selection of the optimal policy for action.

Error correction mechanism in associative learning

From the field of neuroscience, the neuromodulator (dopamine) provides a phasic signal of prediction error in rewards. The correction in the prediction error through the continual firing of dopamine is the core mechanism for influencing the learning process and the selection of appropriate action. It forms the basis of habit formation. Rescorla-Wagner model formulates this mechanism in associative learning as shown in Equation 1.

$$V_{new} = V_{old}(CS_i) + \eta[\lambda_{US} - \textstyle\sum_i V_{old}(CS_i)] \tag{1}$$

V is the estimation of the cumulative strength in conditional stimuli (CS) and the strengthening/weakening of stimuli (i.e. learning) is based upon the error correction (i.e. the difference) between actual strength of unconditional stimuli (λ_{US}) and the predicted cumulated stimuli ($\Sigma V_{old}(CS_i)$).

The model assumes that learning occurs when the actual stimuli differs significantly from previous stimulation (i.e. unexpected events). It also assumes that the strength of different stimuli is an arithmetic summation to form the expectation of outcome (i.e. neglecting the effect of blocking/inhibition). It also suffers from ignoring the second order conditioning (i.e. an indirect stimulus to the outcome) and, similarly, the temporal impact from previous conditions.

Sutton and Barton (1990) expanded on the temporal relationship to Rescorla-Wagner model.

$$\begin{aligned} V(S_t) &= E[r_t|S_t] + \gamma \sum_{S_{t+1}} P(S_{t+1}|S_t)(E[r_{t+1}|S_{t+1}] + \gamma E[r_{t+2}|S_{t+1}] + \ldots) \\ &= P(r|S_t) + \gamma \textstyle\sum_{S_{t+1}} P(S_{t+1}|S_t)V(S_{t+1}) \end{aligned} \quad (2)$$

The value cumulation at current state, $V(S_t)$, is determined by a reward function, $P(r|S_t)$, and expected value accumulation in future based on the probability of state transition, $\Sigma P(S_{t+1}|S_t)V(S_{t+1})$. The recursive valve function exhibits the long-term effect of current state and reward, and is an essential building block for temporal difference (TD) learning. Contrasting with the Rescorla-Wagner model (Eq. 1), learning occurs with the correction of temporal difference prediction error (δ_t) as shown in Equation 3.

$$V(S_t)_{new} = V(S_t)_{old} + \eta \cdot \delta_t \text{where } \delta_t = P(r|S_t) + \gamma \textstyle\sum_{S_{t+1}} P(S_{t+1}|S_t)V(S_{t+1}) - V(S_t) \quad (3)$$

It is also challenging to estimate $P(r|S_t)$ and $P(S_{t+1}|S_t)$ unless prior knowledge in entirety of the domain space. A 'model-free' approach to estimate this prediction error stochastically and incrementally is proposed by Barto et. al (1989), and Berteskas and Tsktiskili (1996) without prior assumption of the dynamics in the environment. In addition, it is more important to understand and optimise the selection of responsive action as the assignment of reward is based on the action rather than the state transition.

Q-Learning is one of the model free approaches to apply TD learning to evaluate the value accumulation (known as the Q function) without generating a defined policy (π), $\pi(S,a) = P(a|S)$. The Q-learning function is updated as shown in Eq. 4 and 5 to be used as 'memory' for subsequent decision making. The choice of best future action (through the max operator) ensures temporal information is applied during the TD prediction error correction process.

$$Q(S_t, a_t)_{new} = Q(S_t, a_t)_{old} + \eta\delta_t \quad (4)$$

$$\delta_t = r_t + max_a \gamma Q(S_{t+1}, a) - Q(S_t, a_t) \quad (5)$$

By learning through many iterations, Q-learning can be ensured to converge to the true optimal (similar to the formation of habit). As a close, the Rescorla-Wagner model in behavioural psychology is shown mathematically to have close resemblance with the widely used Q-learning model in computational reinforcement learning. Interestingly, recent electrophysiological findings (Morries et al. 2006, Roesch et al. 2007) have also validated the same learning path by dopaminergic neurons.