Deceptive Reinforcement Learning

Yen Sheng Miao and **Yanlong Pan**
849388 906250
{yenm, yanlongp}@student.unimelb.edu.au

Abstract

Deceptive reinforcement learning involves finding a policy such that it will be difficult for an observer to identify which goal the agent is moving towards with precision. In this paper, a model-free reinforcement learning method of generating a deceptive policy via q-table initialization is proposed. By utilizing the method proposed by Masters and Sardina [2017] to analyze the deceptiveness of a path, the efficiency of the method presented in this paper is examined by carrying out experimentations with different map configurations and a varying number of fake goals. Furthermore, the result from the model-based method presented in Masters and Sardina's [2017] research is used as an upper benchmark while the naïve reinforcement learning as a lower benchmark when analyzing the efficiency of the method proposed in this paper.

1 Introduction

Machine learning is an area in artificial intelligence that enables a system to learn how to solve a problem autonomously. Generally, this is done through learning and exploring similar types of problems until the best solution for the problem in-hand is obtained or the time limit for training/learning expires. There are three categories of machine learning – supervised learning, unsupervised learning and reinforcement learning (RL). However, unlike supervised or unsupervised machine learning that learns from a set of data, RL learns a solution for a problem by employing an agent to interact with the environment and learn from the agent's experience. While interacting with the environment, a positive reward is added to the agent if the agent performs an action that moves closer to the goal and penalize the agent for any actions that move the agent away from the goal. An optimal solution for a problem is that set of actions that maximizes/minimizes the cumulative rewards/penalties. One of the simplest forms of RL is q-learning, which uses dynamic programming to solve a task. There are also several other variants of q-learning such as SARSA, approximate Q-learning etc. that differs only by the reward structure used to facilitate learning.

This paper introduces a new form of RL algorithm known as deceptive RL. Instead of arriving at an optimal policy to the real goal, which is what conventional RL algorithm does, this algorithm finds a policy to the real goal that is as deceptive as possible to an observer of the environment. To translate the conventional algorithm from finding an optimal policy into an algorithm that finds a deceptive policy, deceptive steps that were taken by the agent to the real goal also need to be rewarded. Additionally, the size of the reward allocated to each location needs to vary according to the location's deceptive magnitude to allow a learning agent to differentiate between locations on the environment that are considerably more deceptive than others. Nevertheless, the overall deceptive behavior is dependent on how the relative magnitude of the reward between locations is determined, which will result in either a simulative or dissimulative policy. A prominent application of deceptive RL will be for privacy protection mainly in the area of route planning, whereby the final destination of a path is disguised by the route the agent takes.

Before the discussion on deceptive RL, two simple but essential assumption about an observer need to be clarified. For design simplicity's sake, it is assumed that an observer is naïve and rational. A naïve and rational observer implies that he/she was not told the possibilities of being deceived and infer the agent's final destination only through the series of observations on the actions performed by the agent. Although RL can be designed to take into accounts the observer's adaptive nature on deception after being deceived some number of times, an implication of these assumptions is that adjustment for the deceptive path is not needed.

This paper is structured as follows. Section 2 describes the essential background for RL, particularly for q-learning, the idea of reward shaping on q-learning as well as deception and a standard of measure to analyze the degree of deception for a policy. Then in the next section, the method and implementation for generating a deceptive policy using RL are illustrated. Section 4 and 5 describes the experiment carried out using the devised method on different scenarios

and the results. Finally, section 6 presents other related works in the area of deception, as well as deception detection and section 7, provides a conclusion for this paper.

2 Background

2.1 reinforcement learning (RL)

RL is a framework that enables an agent to come up with an optimal policy to a goal from a start location in an initially unknown state space. The agent is able to uncover a policy that guides the agent to the goal by interacting with the environment and learning from these interactions.

In RL, the model of a domain is defined as a Markov decision process (MDP). The MDP is a discrete time, stochastic process that can be described by the tuple <S, A, T, γ, R> where [Ng *et al.*, 1999; Sutton and Barto, 1998]:

- S is a state space with a finite discrete set of states $\{s_0, s_1, s_2 \dots\}$;
- A is a finite discrete set of actions that the agent is able to perform $\{a_0, a_1, a_2 \dots\}$;
- T is the state transition probability function from performing an action a ∈ A in state s ∈ S, $T: S \times A \longrightarrow S$;
- γ is the discount factor that takes the value $0 \leq \gamma \leq 1$, depending on the importance of future rewards; and
- R is the reward function from performing an action a ∈ A in state s ∈ S, $R: S \times A \longrightarrow S$.

An optimal solution for the MDP is attained when the expected discounted accumulated reward at every state from the initial state s_0 is maximized. The expected reward is maximized by always performing the best action that provides the highest reward over an indefinite horizon.

Q-value is used to describe the value associated to an action a ∈ A at certain state s ∈ S. A q-value is derived from the Bellman equation and represent the total expected reward associated with an action a ∈ A in state s ∈ S followed by any actions specified in optimal policy to reach a goal.

$$Q(s,a) = r(s,a,s') + \gamma \sum_{s' \in S} P_a(s'|s) \max_{a \in A(s)} Q(s',a') \quad [1]$$

After all actions A for all states S have been calculated from solving the bellman equation recursively, a q-table is used to display the value for each of the q-value for each action at each state. Similarly, choosing the maximum q-value for each state creates a policy for the agent that map states to actions, $\pi : S \rightarrow A$.

2.2 Q-learning

Q-learning is a type of model-free RL method that learns a policy using the q-value of the states. A model-free algorithm implies that the agent requires no explicit knowledge of the environment and learns from its environment through trial and error, in contrast to a model-based algorithm which requires information of the environment to be provided as part of the agent's learning process. The algorithm for q-learning is:

$$Q(s,a) \leftarrow Q_{t-1}(s,a) + \alpha TD_t(a,s) \quad [2]$$

$0 \leq \alpha \leq 1$ is the learning rate that determines the significance of recent information against older information and $TD_t(a,s)$ is the temporal difference which can be further expanded:

$$TD_t(a,s) = Q_t(s,a) - Q_{t-1}(s,a) \quad [3]$$

The $TD_t(a,s)$ term along with appropriate learning rate progressively updates the q-value of the state-action pair until it converges to 0, which implies convergence of the q-value. Algorithm 1 is a brief pseudocode to illustrate the overall process of q-learning which will be important in understanding the method created for deceptive RL in section 3.

```
Initialize Q (s, a) arbitrarily

Repeat (for each episode):

  Initialize s

  Repeat (for each step of episode):

    Choose a from s using policy derived
    from Q

    Take action a, observe r, s'

    Update Q value using equation [3]

    s ← s';

until s is terminal
```

Algorithm 1 Q-learning Algorithm [Sutton and Barto, 2018].

Minor trade-off between exploration and exploitation can increase the efficiency of the learning agent and was introduced in q-learning. The ε-greedy learning mechanism, where ε specifies the exploration rate, allows the agent to explore unknown states ε probability of the time instead of constantly exploiting actions with the highest q-value.

2.3 Reward Shaping

Reward shaping is the act of incorporating additional procedural knowledge during RL, thereby improving the performance of the agent. A reliable shaping reward provides supplementary information with regards to the state of the environment which allows the agent to reduce the number of suboptimal actions performed. These rewards are generally derived from domain knowledge, typically from the modeler of the environment or in the form of historical data. The shaping reward can be integrated into Q-learning by bootstrapping additional rewards (into equation [2]) for actions that indicated a positive progress towards the goal.

However, there are evidence that occasionally the introduction of shaping rewards deteriorates the performance of learning, diverting the agent from their primary goals of the environment. Furthermore, an ill-defined shaping rewards will result in an agent converging optimally to a path that maximizes the shaping rewards, however suboptimal with respect to the original task [Wiewiora, 2003].

The primary objective of reward shaping is to provide additional information for the agent so as to reduce the number of explorations performed before reaching the goal state for the first time. Instead of altering the RL algorithm to incorporate shaping rewards, the q-table can be initialized to achieve the same effect since an action is chosen based on the differences in the q-value of that current state. Koenig and Simmons [1996] also showed that initializing the q-values will have large influence on the efficiency on goal. Further research by Wiewiora [2003] also revealed that shaping reward can be integrated into learning via q-value initialization. Wiewiora [2003] also proves the effect of initializing the q-table using state potential function is analogous to potential based reward shaping which is a reward shaping method that provides theoretical guarantees [Ng *et al.*, 1999].

2.4 Deception

One of the main objectives of deception in the context of grid-world is to generate a path that ensures any rational spectators in the environment is unable to pinpoint the final destination of the path with certainty. However, deceptiveness is a subjective notion and therefore a suitable method of evaluation is needed to enable the classification of paths based on their deceptiveness. Masters and Sardina's [2017] research on deceptive path planning presented a method of ranking paths based on its potential to deceived. In their research, a path's potential to deceived is measured by evaluating the path from the start position to the last deceptive point (LDP) using the idea of path completion [Masters and Sardina, 2017]. Consequently, this method of calculating deceptiveness will be used as a standard of measurement to examine the deceptiveness of an agent in this paper.

Simulation and dissimulation are two major components that makes up deception. Simulation is defined as presenting the false, whereas dissimulation is defined as hiding the real [Whaley, 1982]. In the context of grid-world, simulation misleads an observer into selecting the wrong goal as the agent's final destination. To perform simulation, the agent (P) intentionally travels towards the fake goal (F) then turns back and move towards the real goal (R) after getting close to the fake goal. This behavior is depicted in figure 1.

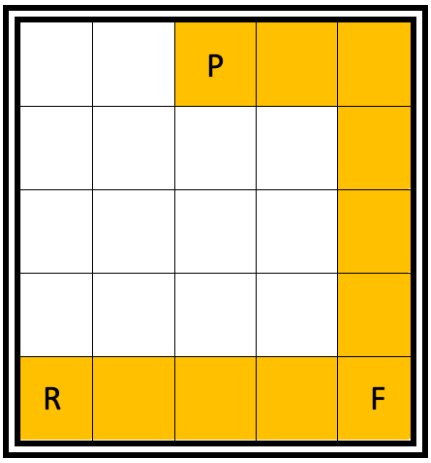


Figure 1 Agent practicing simulation.

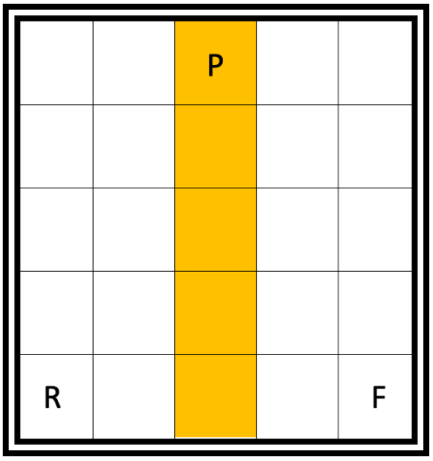


Figure 2 Agent practicing dissimulation.

Dissimulation on the other-hand ensures that the observer is unable to differentiate between which goals the agent is traveling towards until the last minute. Dissimulation occurs when the agent travels along a path where it is both optimal to go either to the fake goal or the real goal. In figure 2, the orange path is a dissimulative path since the agent is unable to tell which goal, real or fake, the agent is traveling towards. Furthermore, the orange path is also an optimal path if the agent, after traveling along the orange path, decided to move towards the real or the fake goal.

2.4.1 Last Deceptive Point

In a multi-goal environment that is transparent to the observer, an agent traveling deceptively through a path will eventually arrive at a node that will no longer be deceptive to an observer. The last node of the deceptive path is defined by Masters and Sardina [2017] as the last deceptive step. In figure 3, the last deceptive point is located at the position marked (LDP), since an observer is unable to determine the goal the agent is travelling to before the agent performs the next step.

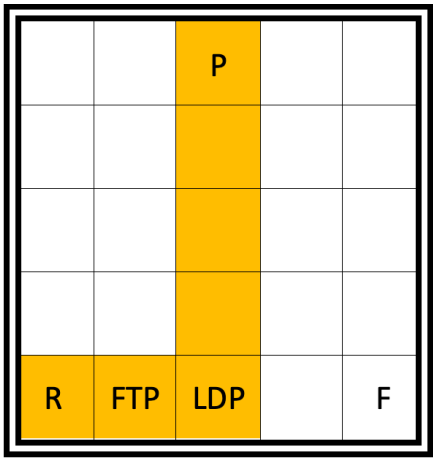


Figure 3 Position of the Last Deceptive Point (LDP) and the First Truthful Point in this environment setup.

Whereas the first node after the last deceptive point is known as the first truthful point (marked as FTP on figure 3). All the nodes, including the first truthful point after the last deceptive point are truthful. From figure 3, it is also apparent that the closer the LDP is to R, the more deceptive the path is to the observer. Furthermore, if the agent travels optimally to the LDP from the start state (P) followed an optimal path from the LDP to the final goal, the overall path is guaranteed to be both dissimulative and optimal.

2.4.2 Measuring Deception

Masters and Sardina [2017] generalize deception as a probabilistic goal recognition problem. A probabilistic goal recognition problem in the context of a multi-goal environment, is a subject of assigning probabilities to each of the goals based on a series of actions performed by the agent. Thus, a probability distribution of goals across all G for each series of observations can be generated: Pr(g|O) which represents the probability of goal g ∈ G being the real goal given the sequence of observation O.

The probability distribution Pr(g|O) can be further simplified by only referencing the final observation in that sequence of observations denoted by Pr(g|n) where n represents the agent's current location. Pr(g|n) can be calculated by calculating the cost difference between the cheapest plan to the goal from the final observation in the sequence, and the cheapest plan that could have been used to reach the goal from the start position [Masters and Sardina, 2017]. Consequently, the lower the cost difference between the two paths, the higher the probability for that goal being the real goal.

$$costdiff(s, g, n) = optc(n, g) - optc(s, g) \qquad [4]$$

The degree of deception for a path can be managed accordingly based the node along the path the agent takes going to the real goal.

Master and Sardina [2017] further formalize the idea of probabilistic goal recognition in deceptive path planning by devising three units of measurement to evaluate the ability for a path to deceive, which is magnitude, density and extent of deception. Magnitude, density and extent can be estimated using information from the environment along with the concept of probabilistic theory as discussed. Each of these properties examines specific part of a path and will be discussed.

2.4.2.1 Magnitude, Density and Extent

To assess the deceptiveness of a path, the probability of each node along the path, which also known as the deceptive magnitude, has to be examined. The result from this assessment is a binary outcome on whether or not the node is a truthful step performed by the agent. By calculating the cost difference between each node and a goal, a node is considered to be truthful if the probability of the agent heading towards the real goal given the node is greater than the probability the agent heading towards any other goal $Pr(g_r|O) > Pr(g|O)$, otherwise the step is considered to be deceptive.

The relative deceptive strength of any two paths can then be ranked using the deceptive density. The deceptive density of a path can be calculated by taking the inverse on the number of truthful steps performed by the agent. Thus, the smaller the number of truthful steps, the greater the density value, the lower the possibilities an observer correctly identifying the location of the real goal.

$$density(\pi) = \frac{1}{|N_t|} \qquad [5]$$

Based on the definition of last deceptive point and the deceptive density, a path can either be a strongly deceptive path or a weakly deceptive path. A path can be considered to be strongly deceptive if the agent travels from the initial state to the LDP without any truthful step. On the contrary, a path is considered as weakly deceptive if there are some truthful

steps present in between the path from the initial state to the LDP.

An important feature that determines the deceptive ability of a path is the length of the LDP measured in terms of path completion. This is also defined as the extent of the deception for a path [Masters and Sardina, 2017]. Recall that LDP signifies the point in a path at which an observer ceases to be deceived. Therefore, it is of interest for the agent to abstain from reaching the LDP until later in the path without sacrificing any progress to the goal being made.

Path completion is an appropriate method to measure the extent of deception as it uses the optimal costs from the start point to the agent's current position to measure progress made by the agent to the goal. By using path completion, it allows the differentiation between a path are able to deceive further from path that can deceive longer. The extent of deception for a path can be calculated using path completion as,

$$pcomp(n, g, s) = optc(s, g) - optc(LDP, g) \qquad [6]$$

According to the formula, extent specifies how close the LDP is to the goal or how far left does the agent need to travel truthfully after the LDP to the goal. The smaller the path completion of the LDP, the shorter the path an agent have to travel truthfully to the goal, the greater the extent of deception of the path.

2.4.3 Maximizing the LDP

The LDP that maximizes the extent of deception occurs at a deceptive node that is closest to the real goal's radius of maximum probability. A radius of maximum probability for a goal g_r is defined as the tipping point where the probability of an agent traveling to that goal g_r is the same as the probability of that agent traveling to any other goals g' on the map, denoted by $costdif(s, g_r, n) = costdif(s, g', n)$. The probability of nodes inside the radius of maximum probability of goal g_r signifies that there is a high probability that the agent is traveling to that goal.

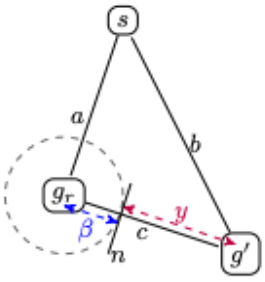


Figure 4 Labels a, b, and c stand for optimal cost between nodes [Masters and Sardina, 2017].

Since $costdif(s, g_r, n) = costdif(s, g', n)$, Thus

$$\beta = \frac{c + a - b}{2} \qquad [9]$$

Thus, an LDP which has the greatest extent of deception should be at a node on or closest to the radius of maximum probability of a node. Masters and Sardina [2017] presented a formula that calculates this radius for a real goal using the position of the real goal, fake goal and the agent's start position.

3 Deceptive Reinforcement Learning Method

The objective of model-free RL is to allow an agent to learn an optimal policy that is able to avoid obstacles while accomplishing the goals set out by the environment. A model-free learning agent does not require an accurate representation of the environment to learn a policy, as oppose to a model-based method, thus making it more robust against various scenarios. However, an agent that can demonstrate the characteristics of being able to deceive while also completing the goal will often result in a sub-optimal policy. Deceptive RL can therefore be described as having two goals, an explicit goal that is set out by the environment and an implicit goal to deceive.

In a model-free environment where the states are unknown to the agent, to ensure that the agent produces a policy with the capability to deceive, some form of heuristic is needed in order to provide the agent with hints on the locations on the map that are deceptive. Furthermore, for an agent to uncover any information about the deceptive points on the map, information regarding the policy for the real as well as the fake goals need to be known. After these policies have been identified, an impression of deception can be created by augmenting the q-table of the fake/real goal to skew the policy towards the fake goal. In this section, two approaches of designing the reward heuristics for the deceptive RL agent will be described. Both of these approaches were built based on the idea of combining multiple q-tables together to arrive at a policy that is able to convey information on all the goals' location on the map, directing the agent to the real goal deceptively.

3.1 Negated Q-table Approach

In a model-free environment, the policy of the agent traveling optimally to the real and fake goal can only be learnt by exploring the environment and learning through these explorations. After the agent has all the optimal policy to each of the goal, the agent can take into consideration the position of goals by simply inferring to the pre-trained policy. The

policy of the goals can be inferred by the agent through reward shaping using q-table initialization, initializing the q-table with a goal's policy.

```
Perform q-learning to real goal using
algorithm from Algorithm 1 to obtain policy
P

For each Q (s, a) in P:
    Q (s, a) ← - Q (s, a)

Perform q-learning to real goal using
algorithm from Algorithm 1, initializing Q
(s, a) with P(negated).
```

Algorithm 2 Algorithm to create a simulative policy.

Recall that a deceptive path minimizes the number of truthful steps to the real goal. A deceptive policy to the real goal can be generated by negating the q-table of the agent's optimal policy going to the real goal, then using it to initialize the table of a new RL agent as shown by the algorithm in algorithm 2. From figure 5, the negated optimal policy that will be used to initialize the q-table exhibits a policy that leans towards traveling to the fake goal. After training, a simulative policy to the real goal as shown in figure 6 is created. Although the policy generated is deceptive, a policy that is repeatedly simulative will be predictable in the long term. Therefore, a dissimulative path is preferred.

	0	1	2	3	4
4	0.64 >	^ 0.00	^ START :0.00	1.06 >	0.90 v
3	0.40 >	−381.41 >	−354.69 >	0.79 >	^ 0.87
2	0.20 >	^ −402.52	0.65 >	0.68 v	< 0.72
1	< 0.00	−424.54 >	0.44 >	0.59 v	^ 0.59
0	REAL GOAL	0.20 >	^ 0.40	< 0.56	FAKE GOAL

Figure 5 Negated optimal policy to the real goal.

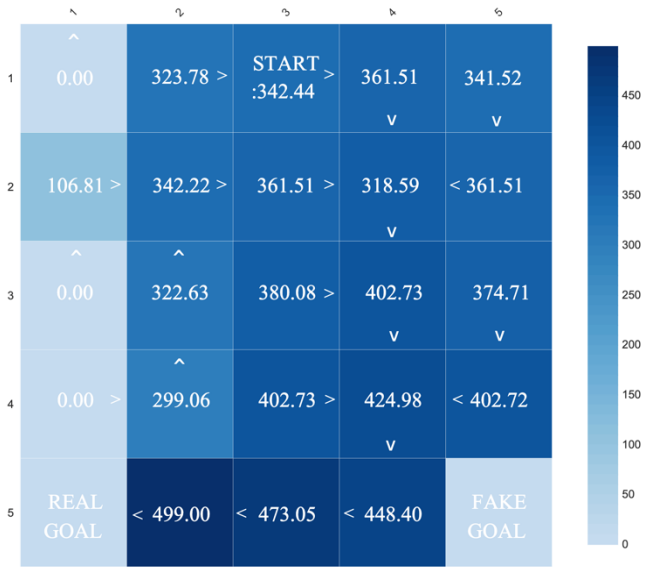


Figure 6 Policy after training with q-table initialized with the negated optimal policy to the real goal (figure 5).

To create a dissimulative policy, the policy of the fake goal (F) needs to be taken into account. Improving on the approach of creating a simulative policy, the sum of the negated real goal ($R_{negated}$) and fake goal's ($F_{negated}$) policy is used to initialize the q-table.

$$Q(s,a) \leftarrow R_{negated} + F_{negated}, \{a \in A, s \in S\} \quad [10]$$

This will create two regions on the map with values that is undesirable for the agent to travel towards the real goal as shown in figure 7 (with the double negative sign). "REGION R" on the figure is created by the negated optimal policy of the agent going to the real goal while "REGION F" is the negated optimal policy of the agent going to the fake goal.

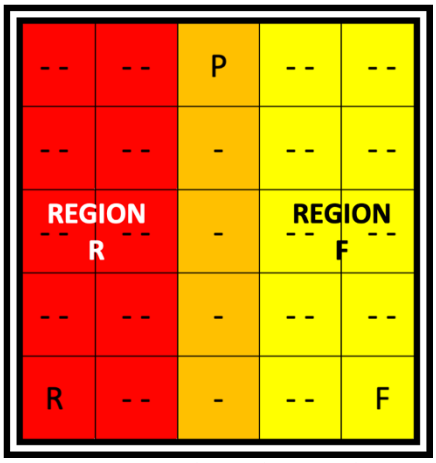


Figure 7 Attempt to create a dissimulative path using Negated Q-table Approach.

The ε-greedy reinforcement agent that chooses actions with the largest q-value (1-ε of the time) will travel using a path

between the real goal and the fake goal ("with the single negative sign") that has a q-value that is relatively larger than region R and F, thereby creating a dissimulative path. However, this method only works well with cases where the agent is located in between the real and the fake goal. Furthermore, the q-table that was used to initialize the policy does not always display signs of noticeable deceptive pattern when two or more negated policy's q-table are added together and used to initialize a q-table. Rather, the q-table display some form of random initialization. In this case, not only will the time needed to train the model became longer, the number truthful steps on the path generated has also increased.

3.2 Iterative Q-table Combination Approach

To improve on the above model, a new approach of combining q-tables is devised. Assuming there is only a single real goal and a single fake goal on the map, the new approach of combining q-tables is as follows,

1. Train the agent with the fake goal but initialize the q-table with the real goal's policy (q-table initially have the value initialize to 0, after that uses real goal's policy from stage n-1).

2. Train the agent with the real goal having the q-table initialized to the fake goal's policy (from step 1) minus the agent's optimal path's policy to the real goal (pre-trained, constant).

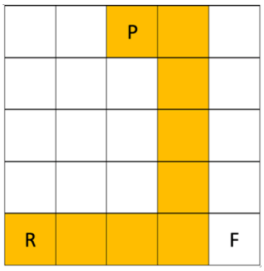
3. By treating step 1 followed by step 2 as a stage, the process (step 1 ⇒ step 2) is repeated until the policy created by the real and fake goal in stage (n - 1) converges to the policies created in stage (n).

The idea behind step 1 and 2 is to allow the agent to take policies of different goals into account when generating a deceptive policy to the real goal. Furthermore, a negated optimal policy to the real goal is also added during initialization of q-table to in step 2 to avoid agent taking the optimal path to the real goal thereby lengthening the time needed for convergence. Adding a negated optimal policy is also analogous to supplying additional information to the deceptive RL agent, thus shaping the path. This is also an idea based on creating the simulative path in the negated q-table approach as discussed. Algorithm 3 shows the pseudocode for the implementation of the new q-table combination method step 1-3.

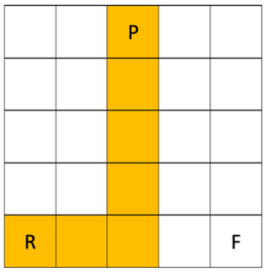
Additionally, the hyperparameters of the learning agent, particularly the discount factor should also be tuned as part of the model to generate a deceptive behavior. The discount factor plays an important role in this model of deceptive RL. Recall that a discount factor specifies the weight at which a future reward will contribute to a state. In this model, different discount factor is assigned to the agent depending on the goals' location and an additional description of the discount factor will be introduced.

```
n = 0 (initialization)

Perform q-learning to real goal using algorithm from Algorithm 1 to obtain policy R_optimal

Perform q-learning to fake goal using algorithm from Algorithm 1 to obtain policy F_n

Repeat:

      Perform q-learning to real goal using algorithm from Algorithm 1, initializing the Q (s, a) with (F_n - R_optimal) to obtain policy R_n

      Perform q-learning to fake goal using algorithm from Algorithm 1 initializing the Q (s, a) with R_n to obtain policy F_n

      n = n + 1

until F_n == F_n-1 && R_n == R_n-1
```

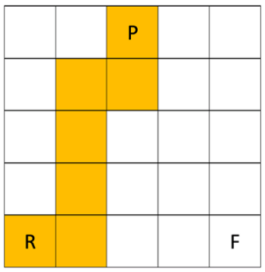
Algorithm 3 Algorithm to create a dissimulative policy.



(a) P⇒F: 0.9, P⇒R: 0.6



(b) P⇒F: 0.9, P⇒R: 0.9



(c) P⇒F: 0.6, P⇒R: 0.9

Figure 8 Effects on the policy for different discount factors.

In figure 8a, a discount factor of 0.9 is assigned to the agent (denoted as P) when training is carried out on the real goal (denoted as R) while a discount factor of 0.6 is assigned to the agent when training is done on the fake goal (denoted as

F). An uneven discount factor – with the agent being trained on the real goal having a larger discount factor than when being trained with on a fake goal – will result in a deceptive policy. The difference in the discount factor can be interpreted as allowing the agent to prioritize the importance of the policy being initialized at the start of training. It can also be thought of as the weight assigned to the initialized q-table at the start of training.

When training is carried out on the real goal, the agent having a higher discount factor, will adjust its path according to the fake goal because the value from the q-table that was initialized will decay at a lower rate. On the contrary, an agent training on the fake goal will adjust its path with respect to the real goal at a lesser extent as a result of the small discount factor. The policy of the real and the fake goal interacts by adjusting their policy at each stage until a balance path is obtained and convergence has been attained. The balance path is the orange colored path in figure 8. Similar reasoning can also be applied to figure 8b [P⇒F: 0.9, P⇒R: 0.9] and 8c [P⇒F: 0.9, P⇒R: 0.6].

However, if the setting of the agent, fake goal and the real goal is located symmetrically as shown in figure 8b, although an uneven discount factor will create a policy with the highest density, a discount factor that is equal will create a policy that has the highest density and also the most optimal. Therefore, to determine the deceptive policy in a model-free RL environment, two different settings of the discount factor typically 8a and 8b need to be inspected and the more dissimulative path will be chosen. The settings in figure 8c always generates a non-deceptive path and is for illustration purpose only.

Figure 9 shows an example of a dissimulative policy after convergence. From the figure, it can be seen that the dissimulative path is in the middle of the map. This result aligns with recent research made on the deceptive path planning. This path is also perfectly dissimulative in that an observer is unable to tell which goal the agent is moving towards when the agent is traveling from the start point to the LDP. A dissimulative policy is unpredictable and will be more efficient in the long run.

This model can be generalized to take into account *m* fake goals as follows,

1. Train the agent with the fake goal f_i, for i = 1, 2, 3 … m, initializing the q-table with the real goal plus fake goal f_j, for j = 1, 2, 3 … m/ {i} policy.

2. Train the agent with the real goal having the q-table initialized to the sum of m fake goal's policy minus the agent's optimal path's policy to the real goal (pre-trained, constant).

3. By treating step 1 followed by step 2 as a stage, the process (step 1 ⇒ step 2) is repeated until the policy created by the real and fake goal in stage (n - 1) converges to the policies created in stage (n).

Figure 9 Dissimulative policy created from Iterative Q-table Combination Approach.

Unlike the model-based method which generates its deceptive path based only one fake goal in a multiple fake-goal environment, the method proposed takes all the goal into account to generates its deceptive policy. However, the time needed for the path to converge is proportional to number of fake goals present in the environment. The higher the number of fake goals, the longer the time needed for the path to converge.

Occasionally, the agent will experience difficulty converging to a dissimulative policy and instead converges to a simulative policy. On top of that, the time taken for the policy to converge will take longer than average. This usually happens when the generated optimal policy to the real goal (used in step 2 of the method) happens to also be the dissimulative policy to the real goal, $R_{optimal} == R_{dissimulative}$. Furthermore, as a result of exploration, the deceptive learning agent will sometimes divert to a path that is less deceptive, contributing more truthful thus reducing the deceptive density of a policy or lengthen the time needed for the policy to converge. Although the probability of these occurrences occurring is below average, a measure is needed to take into account the difficulty for the algorithm to converge. Therefore, a cutoff time of 10 minutes is assigned to the algorithm. If the policy has yet to converge after 10 mins, this implies that one of the scenarios stated above has occurred and the algorithm should be restarted.

4 Experiments and Evaluation

The Pacman framework from the UC Berkeley CS188 [DeNero and Klein, 2014] was used to carry out the experiments in this research and the parameters used for the RL agent is listed in table 1.

Parameters	
Exploration probability	0.05
Learning rate	0.2
Discount rate	Even/uneven discount factor (as discussed in section 3)
Rewards for the goal	500

Table 1 Parameters used for the RL agent

This experiment is intended to test the effects of different planning methods adopted (independent variables – model-free/ model-based) by the agent through examining the performance and efficiency (dependent variables) of the generation of deceptive paths. The maps used in this experiment can be classified into two groups based on the number of fake goals – group 1: maps with a single fake goal and group 2: maps with two fake goals. Each group consists of 5 maps with different configurations that vary in terms of the position of the fake goal(s), the real goal, and the agent's starting position. These experiments can be summarized in table 2. Nevertheless, these experiments are designed to assess the hypothesis which states that the performance and efficiency of the deceptive path planning agent differ according to the types of model that was used to generate a path, and the efficiency as well as the performance of the model-based method is often more superior than the model-free method.

Methods (Independent Variable)	Model based (Control)	Naïve Q learning	Iterative Q-Table Combination
Number of trails	20	20	20

Table 2 The levels of independent variable and the number of repeated trials for each map (a total of 10 maps) that has a single fake goal (Group 1) and two fake goals (Group 2) .

In order to examine the dependent variables of the experiment, some evaluation metrics are needed. To measure the efficiency of the deceptive path, the deceptive density and optimality of a path are investigated. Recall that the density refers to the inverse number of truthful steps while optimality evaluates the number of steps with respect to the minimum steps needed to reach the real goal. Performance, on the other hand, measures the average time needed to generate a path for a map. A summary of the evaluation metrics is shown in table 3.

In reference to the maps chosen for each group, different map configurations were designed to test the algorithm's (model-based/ model-free) abilities to adapt to different environments. Since it is impossible to enumerate all the different map configurations, five configurations that cover broad ranges of scenarios were chosen for each group and is used to test the significance of the experiment's hypothesis. Furthermore, when conducting the experiment, the size of the map is fixed at 5 × 5. Other environment variables that are kept constant during the experiment includes the number of the real goal present in the map and for each group of tests (group 1 & 2), the number of fake goals.

Metric	Measure	Calculation
Optimality	Relative path length	$\frac{N_{total\ steps}}{N_{minimum\ steps}}$
Density	Deceptiveness	$\frac{1}{N_{truthful}}$
Average path generation time	Algorithm/method complexity	$\frac{\sum_1^n T_{run}}{n}$

Table 3 Evaluation metrics for the dependent variables (i.e. efficiency - density and optimality & performance).

5 Results and Analysis

5.1 Experiment results

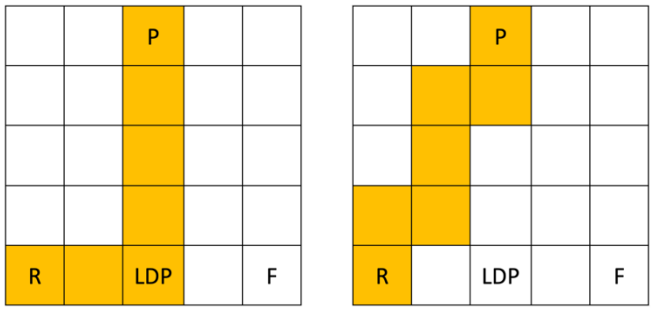
Before the experiment, the allocation of the discount factor needs to be determined for each map when using the iterative q-table combination approach. Therefore, five policies were generated for each map using the iterative q-table combination approach with an even and uneven discount factor (described in section 3). For each of the map, the average density for the policies was calculated and the set of discount factor (even or uneven) that generated the policy with a larger density will be used to carry out the experiment for that map.

A total of 600 trials were conducted over a span of 10 maps and the results were summarized and discussed. For each of the ten maps, a table (like the one shown in table 4) along with three figures (as shown in figure 10) will be used to examine the results. Each section in the table represents the different method used to generate the path. Furthermore, the outcomes are grouped by [optimality, density] pair and count represent the number of occurrences of the path in the trials that have the same optimality and density. Moreover, every policy generated in each map by the algorithm will have a unique pair of optimality and density. Consequently, the count value can be interpreted as the probability of a particular policy being generated. The higher the count value, the more probable that an agent will output that particular policy as its deceptive policy.

Table 4 represents the results from one of the five map configurations in the environment with a single fake goal (group 1). The path that is generated from each method using the map configuration in figure 10 (for 20 trials) was recorded. The iterative q-table combination method was carried out with an even discount factor.

Section 1	Iterative Q-table Combination			
Optimality	1	1.33	1	
Density	0.5	0.5	0.17	
Count	16	3	1	
Section 2	Model-Based approach			
Optimality	1			
Density	0.5			
Count	20			
Section 3	Naïve Q-learning			
Optimality	1	1	1	1
Density	0.2	0.17	0.25	0.5
Count	7	6	4	3

Table 4 Summary of the results for the experiment carried out in a single fake goal environment with the map configuration shown in (figure 10). 20 trials were carried out using each method and the results is grouped base on the optimality and density.



(a) *(b)*

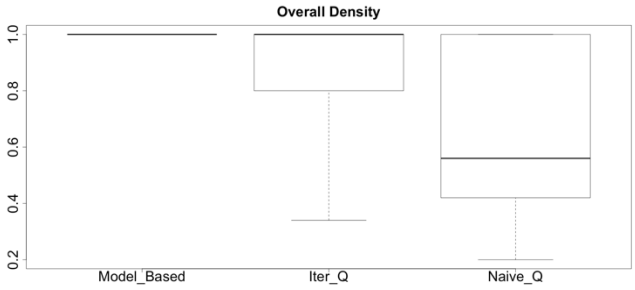
Figure 10 (a) corresponds to the path generated from the mode-based and the iterative q-table combination approaches while (b) represents the policy generated by the naïve q-learning approach.

Because of the randomness exhibited (due to explorations) in reinforcement learning, different policies were generated by the iterative q-table combination and the naïve q-learning method. Thus, each column in table 4 represents a path with its unique optimality and density. The column that is colored in blue (with the largest count number) represents the most probable deceptive path generated by that particular method. The most probable path is shown in figure 10, where figure 10a represents the path generated by the model-based/iterative q-table combination approach while figure 10b represents the policy generated by naïve q-learning.

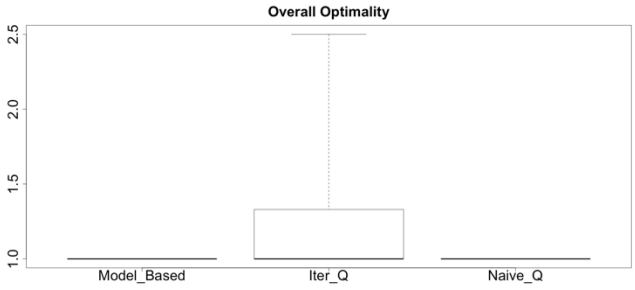
In this example, both the model-based and iterative q-table combination approach generate a more deceptive path when compared to the naïve q-learning approach. The proposed model-free method also generates a policy that behaves similarly as the model-based method 16 out of 20 trials. The results of other experiments can be interpreted using a similar fashion and can be found in the appendix section of the report. Unfortunately, the iterative q-table combination algorithm does not always output a policy that exhibits the same behavior as the path generated from the model-based method for some configuration of the map. In other words, the path's (and therefore [optimality, density] pair) from the iterative q-table combination and model-based approach differ with the iterative q-table combination approach often generating equally deceptive but slightly longer path (higher optimality value from evaluation metrics).

5.2 Overall

Boxplots were used to help visualize the distribution of the overall results. By normalizing the of the optimality and the density results (Iterative Q-table Combination and Naïve Q-learning) using the model-based optimality and density value and adding the counts that has the same normalize optimality and density value for each method over all of the maps (group 1 and 2), a box plot summarizing the optimality and density results can be plotted.



(a)



(b)

Figure 11 Boxplots is used to visualize the (a) density and (b) optimality of the overall results (single and multiple (two) fake goal) of the three methods used.

$$normalized\ density(optimality) = \frac{density(optimality)_{each\ method}}{density(optimality)_{model-based}} \quad [9]$$

As shown in figure 11, the policy generated from the iterative q-table combination approach achieves the same density/ optimality as a model-based method in about 50% of the trials and a density/optimality that is only slightly (25%) larger than the model-base results 25% of the trials.

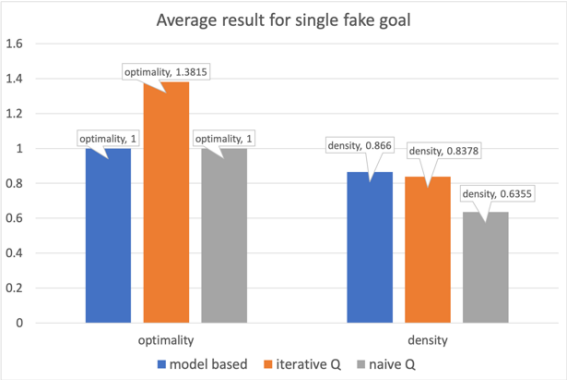


Figure 12 Average density and optimality for the single fake goal environment (group 1).

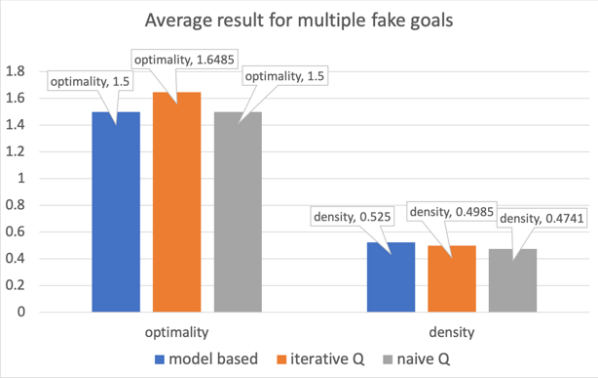


Figure 13 Average density and optimality for the multiple (2 in this case) fake goal environment (group 2).

	Model-Based approach	Naïve Q-learning	Iterative Q-table Combination
Single fake goal (group 1)	0.526	2.984	99.3152
Multiple fake goal (group 2)	0.498	3.012	126.6352

Table 5 The average path generation time for total trials of each group.

Furthermore, two bar charts for comparing both the average density and optimality of all the trials in each group of maps for the three methods are shown in figure 12 and 13. Evidently, the density of the model-based approach (assigned as the upper benchmark) has a higher average deceptive density compare to the iterative q-table combination method. It can also be observed that the optimality of the model-free method is higher than the optimality of the iterative q-learning approach (from the evaluation metrics, the higher the optimality value, the less optimal the policy). This implies that the iterative q-table combination approach will on average, generate a relatively deceptive but slightly longer path compared to the model-based method in both single and multiple fake goals scenarios. Nevertheless, the average density and optimality of the iterative q-table combination approach are reasonably close to the results obtained using the model-based method while higher than the results obtained using the naïve RL approach.

The average path generation time for total trials of each group (i.e. 300 trials for map one to five) regarding of the three methods, measured in seconds is shown in table 5, with the iterative q-table combination approach taking significantly longer to generate a deceptive policy when compared to the model-based method.

From the experiments results, there are some noticeable limitations that can be seen which has been touched upon in section 3. When the number of fake goal increases, the time needed for the policy to converge also increases thus deteriorating the performance. Furthermore, if the policy of the real goal is the same as the dissimulative path, the policy will have difficulty converging (if the agent is unable to converge to a policy that is less efficient). Moreover, since the generation of the dissimulative policy is dependent on the policy generated on the fake goal and the real goal, it is highly sensitive to the randomness due to the agent's exploration behavior, which explains the variability that can be seen in the results.

Therefore, the hypothesis presented in section 4, which claims that a model-based method is more superior than the model-free method (presented in this paper) in terms of efficiency and performance is supported.

6 Related Work

The concept of deceptive RL in this paper complements the research carried out by Master and Sardina's [2017] on deceptive path planning by incorporating a model-free method to generate a deceptive policy. In their research, a model-based method was used to find the deceptive path to the real goal. After the LDP has been identified, off-the-shelf path-planners was used to create different paths by incorporating various heuristics or through pruning away truthful nodes. On the contrary, a model-free method via RL was introduced in this paper to generate a deceptive policy. With this method, no explicit calculation is needed to identify the location of the LDP as this is done automatically during

the training process. Furthermore, heuristics on the map for identifying deceptive nodes are also automatically generated during the training process. Deceptive RL in this paper uses q-learning to practice deception. However, q-learning is not the only machine learning technique that has been used to perform deception.

General Adversarial Networks introduced by Goodfellow et al. [2014] is also a machine learning framework that has been used to implement deception. The two main components in GAN is known as the generator and the discriminator. The function of a generator is to generate a new data that mimics the properties of the input data supplied by the user, while the discriminator analyses the degree of similarity between two of the new data and output a Boolean result. GAN operates by simultaneously training a generative and a discriminative model, then feeding the output of each model back to each other. After several episodes of training, the generator will output a new dataset that has similar properties as the input datasets. Some of the well-known development in GAN in the realm of deception is the style-based generator architecture for GAN proposed by Karras et al. [2017] In this research, an application has been developed using a variation of GAN to generate highly realistic fake images of landscapes, human faces and animals.

Pérez-Rosas et al. [2015] on the other hand, researched deception detection of real-life trial data using supervised machine learning algorithm. By using decision tree and random forest and running leave-one-out cross-validation, they are able to classify truthful and deceptive statements from their witnesses based on their correct features of verbal and non-verbal behavior. Their experiments also achieve an accuracy of 60-75%. This process of machine learning classification can be generalized and be used in other applications based on the features correctly chosen for that application.

There is also extensive research on the subject of reward shaping in RL. For instance, Zou et al. [2019] proposed a new method of reward shaping in RL via Meta-Learning. Meta-Learning automatically learns efficient shaping reward for a RL task. Furthermore, rewards can also be shaped based on statistical inference of the environment. Marom and Rosman [2018] proposed a method that shape rewards based on a Bayesian reward shaping framework that incorporates prior beliefs using belief reward to alter the value of the reward distribution, thereby decreasing the amount of training needed to converge to the optimal policy. This method, however, is not suitable in the context of creating a deceptive policy as is treats the real and fake goal as homogeneous. Thus, the output policy of using this reward shaping method will always result in a simulative policy.

7 Conclusion

In this paper, it can be seen that the model-free RL method can be applied to solve a deceptive path planning problem. This paper also provides two model-free approaches of generating a deceptive policy using the RL algorithm which is the negated q-table approach and the iterative q-table combination approach. Both of these approaches use the idea of reward shaping via q-table initialization to allow the agent to guide its way to the real goal using a path that is deceptive in the eyes of an observer.

However, the iterative q-table combination approach can be seen as being a more superior approach as it provides a dissimulative policy instead of a simulative policy. It can also be seen in the experiment results that the average density and optimality of the iterative q-table combination approach is close to the results obtained when the model-based method proposed in Masters and Sardina's [2017] research was used.

Due to time constraints, only experimental proofs are able to be provided in this paper and any mathematical proofs will be left for future work. The iterative q-table approach also faces the issue of convergence for large maps. Furthermore, more advance q-learning method like the approximate q-learning is suspected to be able to improve the learning quality of the agent in generating a deceptive policy, nonetheless due to time constraints, it is not implemented.

References

DeNero, J. and Klein, D. (2014). Berkeley AI Materials. [online] Ai.berkeley.edu. Available at: http://ai.berkeley.edu/reinforcement.html [Accessed 26 Aug. 2014].

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).

Karras, T., Aila, T., Laine, S. and Lehtinen, J., 2017. Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196.

Koenig, S. and Simmons, R.G., 1996. The effect of representation and knowledge on goal-directed

exploration with reinforcement-learning algorithms. Machine Learning, 22(1-3), pp.227-250.

Marom, O. and Rosman, B., 2018, April. Belief reward shaping in reinforcement learning. In Thirty-Second AAAI Conference on Artificial Intelligence.

Masters, P. and Sardina, S., 2017, August. Deceptive Path-Planning. In IJCAI (pp. 4368-4375).

Masters, P. and Sardina, S., 2017, May. Cost-based goal recognition for path-planning. In Proceedings of the 16th Conference on Autonomous Agents and Multi-Agent Systems (pp. 750-758). International Foundation for Autonomous Agents and Multiagent Systems.

Ng, A.Y., Harada, D. and Russell, S., 1999, June. Policy invariance under reward transformations: Theory and application to reward shaping. In ICML (Vol. 99, pp. 278-287).

Pérez-Rosas, V., Abouelenien, M., Mihalcea, R. and Burzo, M., 2015, November. Deception detection using real-life trial data. In Proceedings of the 2015 ACM on International Conference on Multimodal Interaction (pp. 59-66). ACM.

Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Whaley, B., 1982. Toward a general theory of deception. The Journal of Strategic Studies, 5(1), pp.178-192.

Wiewiora, E., 2003. Potential-based shaping and Q-value initialization are equivalent. Journal of Artificial Intelligence Research, 19, pp.205-208.

Zou, H., Ren, T., Yan, D., Su, H. and Zhu, J., 2019. Reward Shaping via Meta-Learning. arXiv preprint arXiv:1901.09330.

Appendix

The following shows the figures for the most probable paths generated by each method for single and multiple (two) fake goals and the table of results (frequency of the [optimality, density] pair) for the experiments carried out in this research project. In each of the figures, (a) is the path generated from iterative Q-table combination approach, (b) is the path generated from the mode-based approach and (c) represents the policy generated by the naïve Q-learning approach. An uneven discount factor was found to give the largest deceptive density for all of the map configurations and was used proposed method to carry out the experiment when using the proposed method.

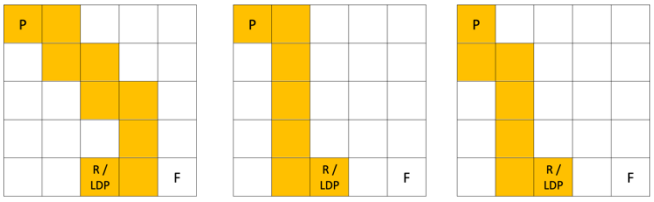


Figure 14 Most probable path generated by each method.

Section 1	Iterative Q-table Combination				
Optimality	1.33	1.67	1		
Density	Inf	Inf	Inf		
Count	15	4	1		
Section 2	Model-Based approach				
Optimality	1				
Density	Inf				
Count	20				
Section 3	Naïve Q-learning				
Optimality	1				
Density	Inf				
Count	20				

Table 6 Table of results.

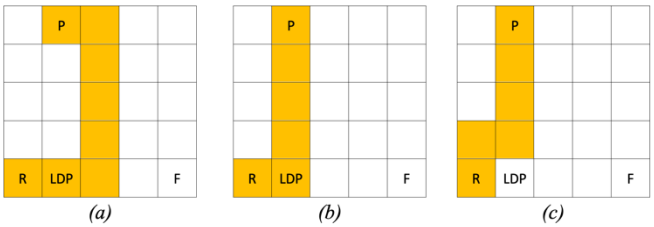


Figure 15 Most probable path generated by each method.

Section 1	Iterative Q-table Combination				
Optimality	1.4	1.8			
Density	1	1			
Count	16	4			
Section 2	Model-Based approach				
Optimality	1				
Density	1				
Count	20				
Section 3	Naïve Q-learning				
Optimality	1	1	1	1	1
Density	0.5	0.2	0.33	0.25	1
Count	6	5	3	3	3

Table 7 Table of results.

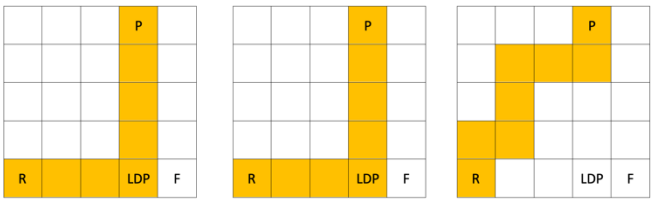


Figure 16 Most probable path generated by each method.

Section 1	Iterative Q-table Combination				
Optimality	1	1			
Density	0.33	0.25			
Count	15	5			
Section 2	Model-Based approach				
Optimality	1				
Density	0.33				
Count	20				
Section 3	Naïve Q-learning				
Optimality	1	1	1	1	1
Density	0.17	0.2	0.25	0.33	0.14
Count	10	3	1	1	5

Table 8 Table of results.

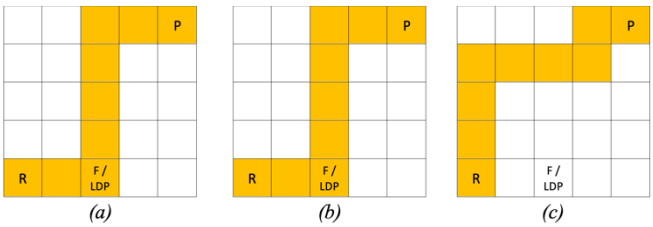


Figure 17 Most probable path generated by each method.

Section 1	Iterative Q-table Combination			
Optimality	1	1	1.25	
Density	0.5	0.33	0.2	
Count	11	7	2	
Section 2	Model-Based approach			
Optimality	1			
Density	0.5			
Count	20			
Section 3	Naïve Q-learning			
Optimality	1	1	1	1
Density	0.2	0.5	0.25	0.33
Count	7	6	5	2

Table 9 Table of results.

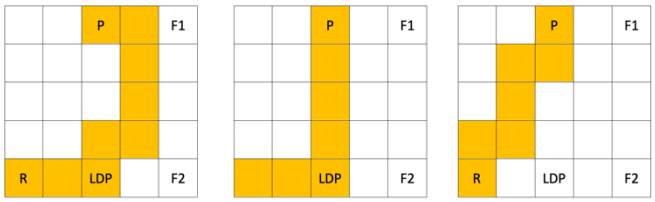


Figure 18 Most probable path generated by each method.

Section 1	Iterative Q-table Combination				
Optimality	1.33	1.33	1	1.33	1.33
Density	0.5	0.2	0.33	0.33	0.2
Count	12	2	2	2	2
Section 2	Model-Based approach				
Optimality	1				
Density	0.5				
Count	20				
Section 3	Naïve Q-learning				
Optimality	1	1	1	1	1
Density	0.2	0.17	0.25	0.33	0.5
Count	8	4	4	2	2

Table 10 Table of results.

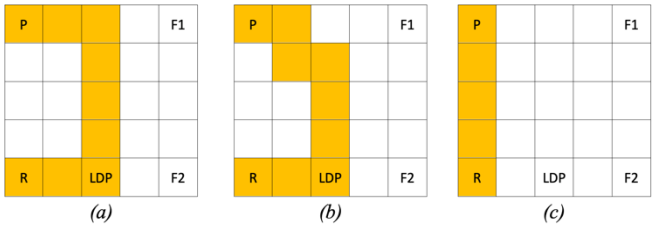


Figure 19 Most probable path generated by each method.

Section 1	Iterative Q-table Combination		
Optimality	2	1.5	2.5
Density	Inf	Inf	Inf
Count	14	2	4
Section 2	Model-Based approach		
Optimality	1		
Density	Inf		
Count	20		
Section 3	Naïve Q-learning		
Optimality	1		
Density	Inf		
Count	20		

Table 11 Table of result.

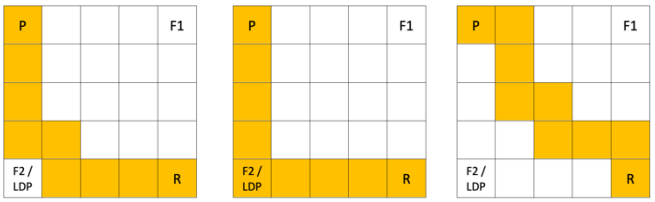


Figure 20 Most probable path generated by each method.

Section 1	Iterative Q-table Combination				
Optimality	1	1	1	1	1.5
Density	0.2	0.17	0.14	0.25	0.14
Count	6	6	2	4	2
Section 2	Model-Based approach				
Optimality	1				
Density	0.25				
Count	20				
Section 3	Naïve Q-learning				
Optimality	1	1			
Density	0.14	0.17			
Count	18	2			

Table 12 Table of results.

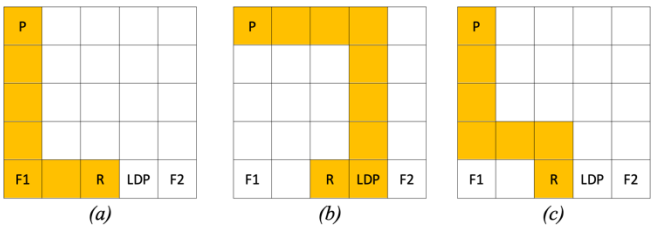
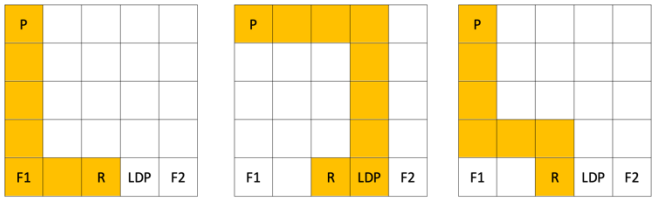


Figure 21 Most probable path generated by each method.

Section 1	Iterative Q-table Combination
Optimality	2
Density	Inf
Count	20
Section 2	Model-Based approach
Optimality	1
Density	Inf
Count	20
Section 3	Naïve Q-learning
Optimality	1
Density	Inf
Count	20

Table 13 Table of results.



(a) *(b)* *(c)*

Figure 22 Most probable path generated by each method.

Section 1	Iterative Q-table Combination			
Optimality	1	1	1	
Density	0.5	0.33	0.25	
Count	12	8	1	
Section 2	Model-Based approach			
Optimality	1			
Density	0.5			
Count	20			
Section 3	Naïve Q-learning			
Optimality	1	1	1	1
Density	0.5	0.33	0.25	0.2
Count	10	4	2	4

Table 14 Table of results.