Navigating Troubled Waters

Group 3
Lindgren Sven Pontus (A0276729L, T3, e1133035)
Hester James Douglas (A0217367Y, T1, e0543403)
Saravanan Yukesh Ragavendar (A0192340M, T2, e0360590)
Gigiolio, Blake Alexander (A0275276U, T2, e1125290)
November 24, 2023

Abstract

Piracy has been a significant issue in Southeast Asia, particularly due to the concentration of maritime commerce within the area. As such, we planned to model and solve the problem of pathing through AI planning and decision-making, utilizing a variety of algorithms to do so. We modeled the Straits of Malacca as a deterministic Markov Decision Problem (MDP) and ran Value Iteration (VI), Policy Iteration (PI), Q-Learning, and Deep Q-Learning (DQN) to obtain policies to path through the Straits of Malacca. The datasets we used were [1] and [2]. For this use case, it seems like Q-Learning is preferable over DQN while VI is likely preferable over PI. Our code is accessible [HERE]

1 Background

The United Nations recently called the waters of South East Asia the most dangerous in the world due to the presence of pirates operating in these waters[3]. This region accounts for 41% of all global piracy which means that Singapore is uniquely affected by the presence of pirates in the region, as it relies heavily on maritime trade in its economy. The issue closely aligns with the field of AI planning and decision-making, hence the reason to resolve it using relevant algorithms. The goal is to make a program that balances the cost of shipping goods long distances with the risk of travelling through potentially pirate-infested waters.

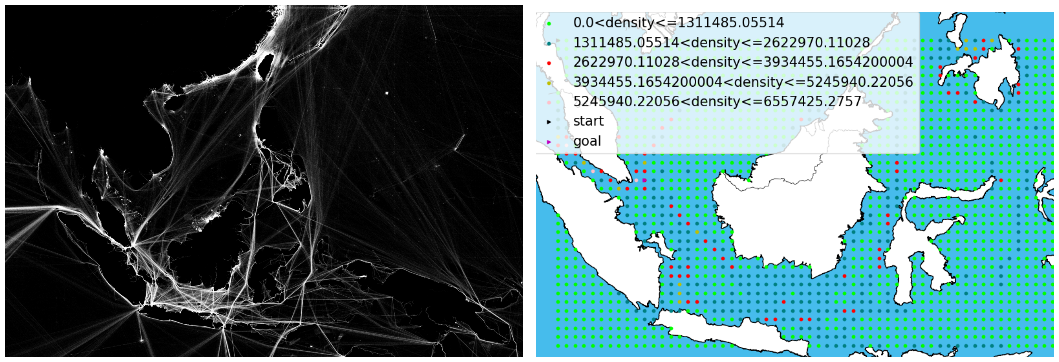
2 The Representation of States

Each state needs information about the local piracy risk. It was possible from the piracy dataset [1] to model the severity of individual pirate attacks. In order to accurately measure the risk the density of shipping is also needed. We used [2] which was one of very few free datasets. Figure 1 (a) shows a part of the raw density data. We rescaled the highly detailed density data and merged the data with the piracy data into a dictionary of states. Figure 1 (b) shows the density attribute of the states for a small portion of Southeast Asia for a dictionary of states. Every dot in the grid is a state.

The dictionary of states consists of longitude and latitude paired with a dictionary of attributes such as “danger” and “density”. Here is an example for the coordinate (100,-8):

```
"(100.0, -8.0)": {"neighbours": [[[100.5, -8.0], 0], [[100.0, -7.5], 1]],
    "n_attacks": 0, "danger": 0, "to_goal": 10.062305898749054, "density":
    110161.153}
```

Note that the dictionary of states has all the information about the transition matrix as the neighbours contain information about all possible actions in each state and all actions are deterministic.


0.0<density<=1311485.05514
1311485.05514<density<=2622970.11028
2622970.11028<density<=3934455.1654200004
3934455.1654200004<density<=5245940.22056
5245940.22056<density<=6557425.2757
start
goal

(a) Raw density data

(b) Density data for each state

Figure 1: The raw density data and the density data attribute of each state in our representation. Each dot in (b) represents a state.

3 Reward Function

The first hypothesis for the reward function is

$$\begin{aligned} R(s') = -\text{Danger}(s') - \text{Euclidean Distance}(s', \text{goal}) \\ R(\text{goal}) = 1000 \end{aligned} \tag{1}$$

The Danger function is split into 3 parts: mapping the significance of each pirate attack to a numerical danger value, propagating the danger value to its neighbours, and normalizing all danger values across the dataset.

The pirate attacks are categorized by their impact on the vessel and crew, and assigned a numerical value (this value is proportional to the impact) as shown in the table below:

Fired Upon	Hijacked	Explosion	Boarding	Boarded	Attempted	Suspicious	NA
15	10	7	5	5	1	0.5	1

All neighbours of states with danger associated receive the danger value as well, at a factor of $\gamma = 0.5$. All danger values across the dataset are normalised in accordance with the sigmoid function, modified to include tuned constants. Additionally, danger values are tuned based on the ratio of density to danger, giving an accurate proportion of risk based on how many ships travel in that region. $f(x)$ is the tuned danger, given x which is the original danger calculated (minus the density):

$$f(x) = \frac{\text{Danger}_{max}}{1 + e^{-x+1}} \cdot \frac{1}{Density} \tag{2}$$

The resultant danger value $\text{Danger}(s') = f(x)$ was added to the Euclidean distance of the state from the goal to penalize actions away from the goal, see equation 1. The reward for reaching the goal was set to be 1000, subject to changes in subsequent iterations. Shipping density is used to gauge the likelihood of a pirate attack occurring at a location 5.2.

4 Solving MDP

Before approaching to solve the MDP, it is important to define the target route to be planned, so there exists a real-world reference to judge the paths suggested by the algorithms. As such, for the Dynamic Programming methods and Q-Learning, the frequently used shipping lane between Singapore and Port Moresby, Papua New Guinea will be used. See Appendix C.1 for details about the reference route.

4.1 Value Iteration

Value Iteration was implemented as per the pseudo-code generally available, with minor changes to accommodate that our transition matrix only bears transition probabilities of 1. Considering the large number of states, full convergence would be computationally expensive, hence the epsilon termination condition was used to limit the number of iterations.

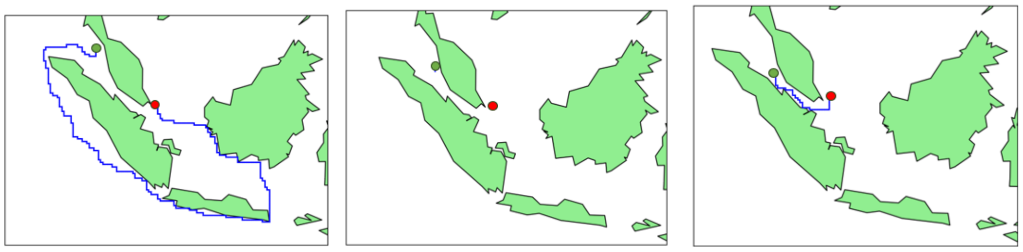
4.1.1 Parameter Tuning

To optimize the performance of Value Iteration, tuning of parameters such as epsilon and gamma was required, to balance between the number of iterations executed, the time taken for each iteration, and the correctness of the policy function. (covered in detail in Section 5.2). To obtain acceptable values for gamma and epsilon, VI was run for differing values of these parameters, and metrics such as the number of iterations required and time taken overall were recorded, and plotted in Appendix C.2.

Convergence characteristics are observed to be indistinguishably similar for ϵ values of 1×10^{-11} and lesser; this implies that it is reasonable to assume that VI has reached close to optimal policy and is not expected to converge further with any significance. The empirically determined $\epsilon = 1 \times 10^{-11}$ is chosen for the subsequent section, executing for a time of 195 seconds.

4.1.2 Preliminary Validation of Reward Function

The first rendition of the algorithm resulted in a largely unstable policy. An example of the policy acting in 3 different ways for 3 geometrically close points is shown as follows (green circle denotes the start, red denotes the end) in Figure 2.



(a) Start: (92.2, 6.2)
Policy chooses to take an abnormally long way around

(b) Start: (100, 4.4)
Policy gets caught in a loop at (100, 4.6)

(c) Start: (100.4, 3.2)
Policy takes an acceptable path to the goal

Figure 2: The path taken by the policy from 3 points in the Malacca Strait

It is to be noted that there are no pirate attacks, or negative rewards, that justify the unnecessarily long path in Figure 2a. Upon investigation, it was found that the use of Euclidean distance as a heuristic to the goal was one of the causes of instability. In Section 3, the use of Euclidean distance as a heuristic was promoted to push the algorithm towards the goal, to converge faster. However, it introduced the undesirable effect of favouring local optima without considering the global side effects of taking an action. In this context, the rewards function affects the "exploration" ability of Dynamic Programming methods.

As such, the rewards function was revised as shown in Equation 3:

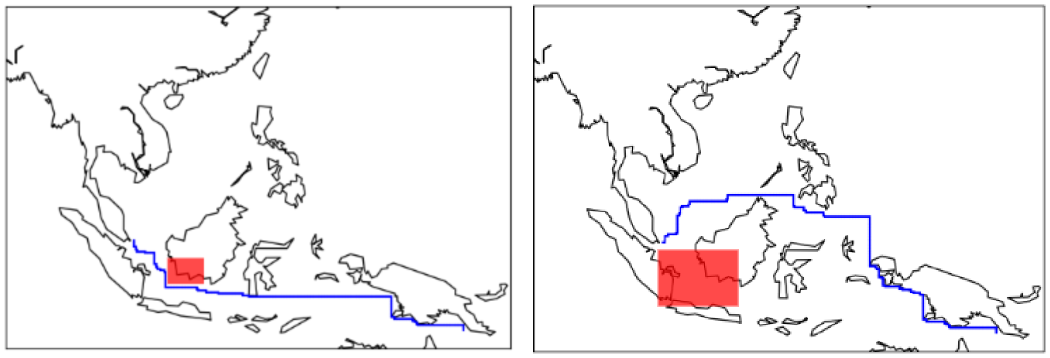
$$\begin{aligned} R(s') &= -\text{Danger}(s') - 1 \\ R(\text{goal}) &= 1000 \end{aligned} \tag{3}$$

With a constant for a penalty for moving, the distance to the goal is implicitly incorporated into the rewards function. A longer path is still penalized, but now, there is lesser convergence towards local optima when the danger is 0 for all neighbors; the penalty for entering any neighbor would be a constant and equivalent. Integration of this reward function solved the issues represented in Figure 2, and VI now tend to converge to a sufficient optimum that generates a close-to-optimal policy function. The following route is obtained from the generated policy, to model the scenario described in Section 4, as shown in Figure 4a.

4.1.3 Simulation of Piracy

Referring to the path generated in Figure 4a, it is to be noted how close the path is to the south of Borneo Island. This is not desirable in the maritime industry for several reasons, such as grounding, obstacle collision, unprecedented weather changes, and most importantly, in the context of this project, prone to piracy. From the perspective of the policy, however, one can understand why the closeness is being preferred; it represents the shortest possible path to the goal, and any danger associated with pirate attacks on the path might not be high enough to justify a diversion.

In the interest of research, suppose the planner views closeness to land as a major drawback, this would be represented in the MDP as a large negative reward on any state that has close proximity to land. Translating this to the reward function in Equation 3, $Danger_{max}$ should be increased to increase the impact of danger across the dataset. VI was then executed again for the modified reward function, and the path in Figure 3a is obtained.



(a) Minor modification (b) Major modification

Figure 3: The path taken according to the policy from Value and Policy Iteration

The red box in Figure 3a represents the relevant region of the map that has been affected by the modification of the reward function, and the addition of a few simulated pirate attacks within the region. Executing the updated policy appears to show a route that cleanly avoids the affected region while maintaining the shortest possible path.

There may be situations where not just closeness to land, but utilisation of an entire sea may be a drawback, due to weather or piracy reasons. Suppose the Java Sea is deemed as extremely unsafe, and the reward function is modified to penalise the entirety of the Java Sea, In this case, as depicted in Figure 3b. As expected, the policy chooses to go north of Borneo island, not finding a shorter option than this. In reality, the crew may decide that is might be worth risking a journey through a prescribed unsafe region, when other factors such as meeting timelines come into play. To achieve this notion of trade-off, the $Danger_{max}$ value can be adjusted accordingly to achieve either the path in Figure 4a or Figure 3b.

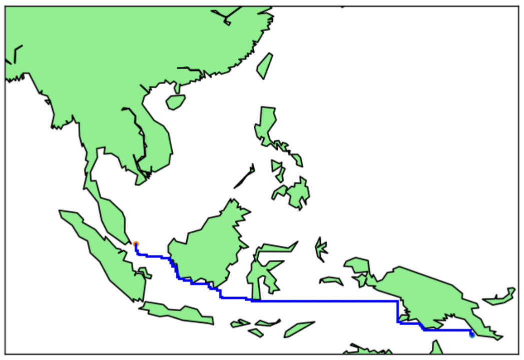
4.2 Policy Iteration

Policy Iteration was similarly implemented as per the pseudo-code available in the lecture notes, consisting of policy evaluation and policy improvement. Relevant parameters include the aforementioned gamma as well as theta, which has a similar role and value to epsilon for policy evaluation and had to be tuned for the same purpose as mentioned in Value Iteration.

The resulting policy generated was similar to that of Value iteration, as seen in Figure 4, converging in 365 iterations. However, the algorithm's runtime was significantly longer, lasting for approximately 14 hours and 35 minutes. The issue lay in the aforementioned 54107 states, whereby within the Policy Iteration algorithm per iteration, policy evaluation runs similarly to value iteration, resulting in a runtime similar to running Value Iteration and policy improvement 365 times.

4.3 Q-Learning

Q-Learning was implemented as per the pseudo-code available in the lecture notes with slight adjustments and focuses on two general choices, exploration (with the introduction of unknown rewards and transitions), exploitation (choosing the best-known action), and how to balance the two in accordance to the current state of the model. We also implemented an Upper Confidence Bound


(a) Start: (147.0, -10.0)
VI Policy takes the optimal path
to the goal

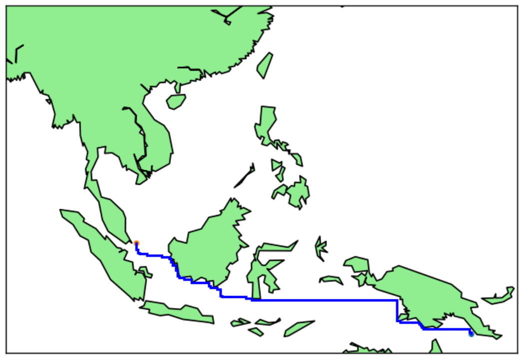
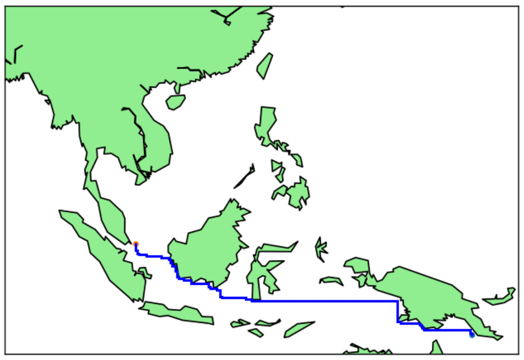

(b) Start: (147.0, -10.0)
PI Policy takes the optimal path
to the goal

Figure 4: The path taken according to the policy from Value and Policy Iteration

(UCB) in accordance with the one shown in the lecture to assist with the exploration step. The tuning parameters involved are the aforementioned gamma, alpha the learning rate for updating action in state utilities, c the constant for scaling the importance of exploration in the Upper Confidence Bound, the number of episodes to run through, epsilon which now dictates the probability of whether to take a random action instead of deciding via the UCB, and a reduction factor to reduce epsilon as the number of episodes increase.


(a) Start: (147.0, -10.0)
VI Policy takes the optimal path
to the goal

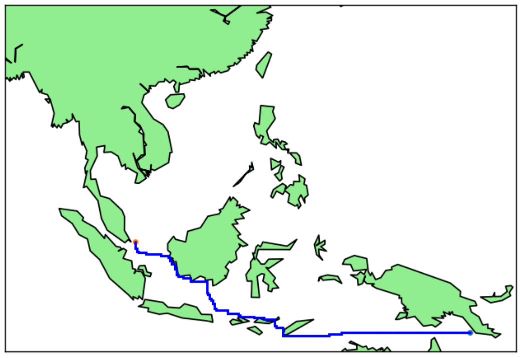

(b) Start: (147.0, -10.0)
Q-Learning Policy takes a sub-optimal path
to the goal

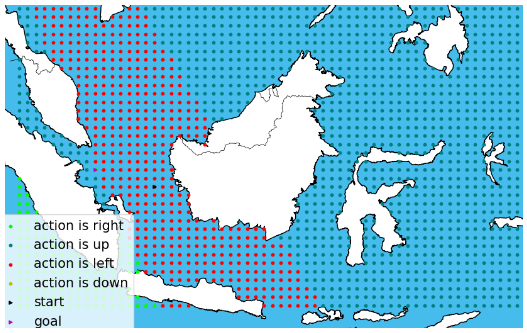
Figure 5: The path taken according to the policy from Value Iteration and Q-Learning

The resulting policy from 20000 episodes is as seen in Figure 5. It had a runtime of 1 hour and 22 minutes, comparatively faster than Policy Iteration as it does not have to traverse every state in the state space in every episode. However, due to this it is capable of being caught in loops and as seen in Figure 5 whereby the path goes down before going up, may not result in an optimal policy, potentially not reaching the goal if there are too few episodes.

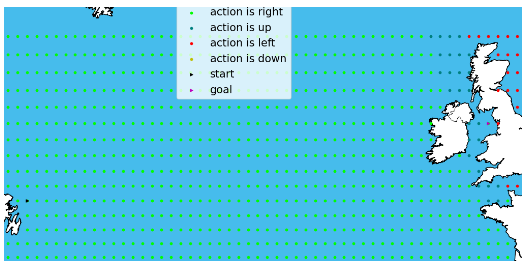
4.4 Deep Q-Learning

We implemented deep Q-Learning highly similar to the lectures (including experience replay, guiding function greedy in the limit of infinite exploration, target network) and spent time attempting to tune it. The input to the DQN was the location of the state relative to the goal. For details such as the reward function used see the GitHub. More tuning of the algorithm might improve the results.

In order to decrease the difficulty (needed because of time and resource constraints) we decided to simplify the problem to only consider the distance to the goal and not the pirate danger. All the results shown are for 1000 episodes and a maximum of 100 iterations per episode. This takes 1-4

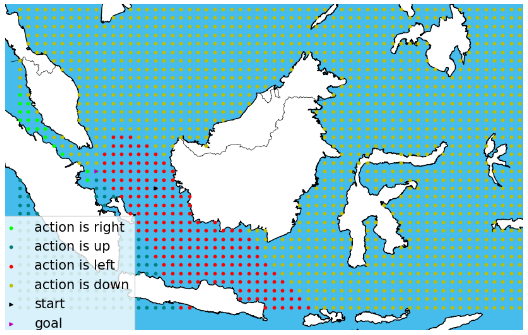


(a) Navigation between the west coast of Borneo and approximately Singapore.

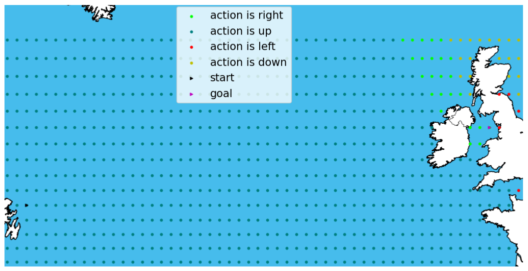


(b) Navigation between Canada and England.

Figure 6: Policy for two different scenarios calculated using a model trained on navigation between the west coast of Borneo and approximately Singapore (the scenario shown in (a)).



(a) Navigation between the west coast of Borneo and approximately Singapore.



(b) Navigation between Canada and England.

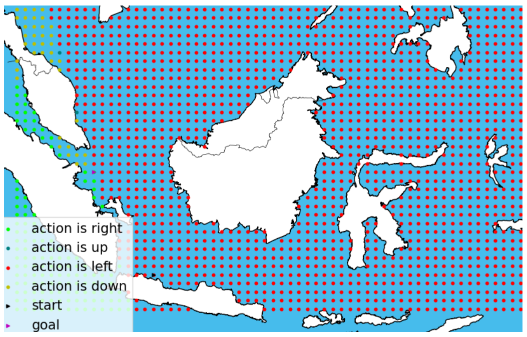
Figure 7: Policy for two different scenarios calculated using a model trained on navigation between the west coast of Borneo and approximately Singapore but with two different starting positions.

hours on a laptop depending on how often the agent reaches the terminal goal state. This is hence what is feasible to experiment with given the time constraint.

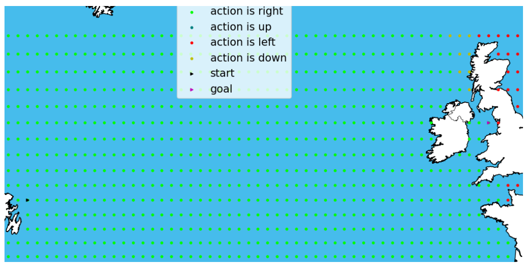
The results shown in figure 6 are for an agent trained in navigation between the west coast of Borneo and approximately Singapore. In figure 6a we can see that following the policy of the trained model from the starting position means that you end up where you want. When analyzing the policy of the model we start to see significant limitations in generalisability. Why would it suddenly be a good idea to go north when north of Singapore in order to reach Singapore? The model does however translate quite well to a different scenario. See figure 6b for the policy of the model trained in Asia used for a scenario about navigation between Canada and England.

It is reasonable that the agent learns the policy shown in 6a as it is quite optimal for the fixed start and goal. Due to the agent being unlikely to reach for example the coast of Brunei, it does not matter to it that the policy is rubbish there. To combat this we implemented a stochastic variation of the starting position for every episode, sampling among two different goals a bit east of Singapore, one above the goal and one below. This resulted in the policies of figure 7 for the Borneo-Singapore and Canada-England scenarios respectively. In the figure, only one of the starting positions is marked. Note that the move left region has been correctly shrunk compared with figure 6a due to the starting location north of the goal. The policy for the Canada-England case is worse far away from England but better close to England relative to figure 6. That the policy is not great far away from England is reasonable given that the model has only trained on short-distance navigation.

One thing we learned when working on this part of the project is that the difficulty of learning how to solve a problem varies with the problem formulation. This is very important when reasoning about whether or not deep Q learning can be a good choice for a specific problem. Figure 8 shows an



(a) The policy for navigation in Asia. Just one of the start positions is marked.



(b) The policy generated by the model for navigating between Canada and England.

Figure 8: Policy for two different scenarios calculated using a model trained on navigation between approximately Singapore and two states far away.

unsatisfactory outcome after the model trained on two different goals further away from the goal (approximately Jakarta and a state to the northeast of the goal similarly far away). This is the reason the previous results are for scenarios with the start position close to the goal, problems far away seem to be too difficult given the resource constraints.

It can be intuitively understood from Figure 8 why the problem is hard. Starting from the Jakarta case changing to moving up instead of left means you might end up in Thailand which is far away from the goal while if you just move left you are caught by Sumatra or Java, never risking very negative rewards. Due to the lack of anything catching the boat when moving up this becomes a quite difficult problem to learn as learning to move up and left while left in itself is much better than up likely is more challenging than first learning that up is good, left is good and then that up and left is better.

5 Policy Evaluation

For evaluating our policies, we chose to assess both the 'distance' cost of the path taken, as well as the 'danger' cost of the path taken. Both are assessed in terms of monetary cost. The path is run based on the policy, and the following values are collected:

5.1 Distance

To calculate distance costs, we use the following formula:

$$Total\ cost = (fuel\ per\ mile \cdot fuel\ cost\ per\ metric\ ton \cdot miles\ per\ latitude\ step) \cdot steps\ taken$$

$$Total\ cost = (0.435 \cdot 552 \cdot 69) \cdot steps\ taken$$

(In metric tons[4], USD[5], and miles respectively)

$$Total\ cost = 16568 \cdot steps\ taken$$

5.2 Danger

To calculate danger costs, we first assign coordinates near pirate attacks a danger value based on proximity. This is calculated using a danger gradient propagating from the site of the attack, assigning a danger score of 100 to the exact coordinate of the attack (or the closest depending on the scale used), and decreasing dangers up to 5 longitude/latitude steps from the attack. The details are in C.3.

The penalty for each step of the path chosen is then divided by the density of that region in order to simulate a realistic chance of encountering a pirate attack at that location. The total accumulated is divided by 100,000 and multiplied by 200 million, which is an example cost for the total cost of freight on a large shipping vessel.

5.3 Results

Through analysis of the policy results, we can compare the effectiveness of our different AI algorithms on this shipping problem. The evaluations are based on the scenario of Port Moresby to Singapore at a 0.2 scale. Due to the computationally heavy nature of DQN, calculating this scenario would take far too much time to evaluate in a reasonable time frame so it has been omitted.

Method Used	Distance (cost)	Piracy Penalty (cost)	Total cost
Value Iteration	53.40 (176946.24)	14.28 (28556.52)	913287.72 USD
Policy Iteration	53.40 (176946.24)	14.28 (28556.52)	913287.72 USD
Q-Learning	55.80 (184898.88)	14.11 (28218.91)	952713.31 USD

Value Iteration and Policy Iteration produced our best (and the same) results. Value Iteration took a much shorter time to run, giving it an edge over choosing policy iteration. Q-Learning took a slightly less risky path than value iteration, but took a longer path, leading to an overall higher penalty cost.

6 Conclusions and Future Work

Overall, our best results were seen by our Value Iteration and Policy Iteration, but seeing as Policy Iteration is very computationally intensive, Value Iteration seems preferable. Another conclusion from the study is that DQN seems to not always be the best choice for a reinforcement learning problem. Q-Learning was able to produce good results for much more difficult problems than deep Q-Learning given similar time for training. Hence, for a reinforcement learning formulation of the problem, Q-Learning is likely preferable over DQN. A conclusion from the DQN part is that it seems to be important to train on many different tasks in order to obtain the best generalisability.

There is a lot of potential future work. Other forms of data, such as real-time weather data and port regulations, could have been integrated with the existing data to from a more complex understanding of the environment. The action space could be expanded, such as having diagonal actions. Transition probabilities need not necessarily be 1: there may be cases where extreme weather or ship malfunctions may cause the intended action to result in another state. Multi-agent MDPs would be more relevant alongside shipping density to simulate the real world, while having multiple terminal states could enable paths to be planned to call at multiple ports in one voyage. A fully observable environment is an assumption made in this project; in reality, states may change with time (piracy may be reported en route, the piracy risk can change over time), or states may not be fully observable (e.g. miscommunication in live piracy reporting), hence POMDPs would be a better way to model the environment.

A couple of suggestions for what future studies can explore connected to DQN; running for more episodes and maximum iterations on more difficult problems using an HPC cluster, tuning a penalty for hitting islands in order to handle obstacles more efficiently, and looking into training on progressively more difficult starting positions.

References

[1] Paul Benden et al. “Crime at Sea: A Global Database of Maritime Pirate Attacks (1993–2020)”. In: *Journal of Open Humanities Data* 7 (2021).

[2] Diego A Cerdeiro et al. *World seaborne trade in real time: A proof of concept for building AIS-based nowcasts from scratch*. 20-57. International Monetary Fund, 2020.

[3] Adam McCauley. *Pirates in Southeast Asia: The World’s Most Dangerous Waters*. URL: `https://time.com/piracy-southeast-asia-malacca-strait/`.

[4] Dmitry Shafran. *Ship Fuel Consumption per Mile! OR How Much Fuel Does a Cargo Ship Use?* URL: `https://maritimepage.com/ship-fuel-consumption-per-mile-or-how-much-fuel-does-a-cargo-ship-use/`.

[5] Elizabeth Stratiotis. *Fuel Costs in Ocean Shipping*. URL: `https://www.morethanshipping.com/fuel-costs-ocean-shipping/`.

Appendix

A Division of labour

A rough sketch of the division of labour:

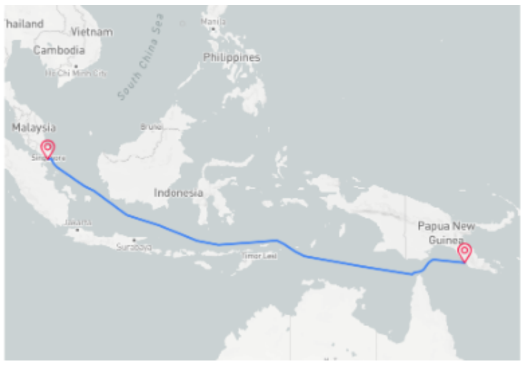
Name	Work Done
Pontus	Data Processing, Dictionary Representation of States, Deep Q-Learning
Blake	Data Processing, Risk Map, Reward Function, Policy Evaluation
James	Policy Iteration, Q-Learning, Data Visualisation (on Basemap)
Yukesh	Transition Matrix, Value Iteration, Q-Learning, Piracy Simulation

B Coding resources used

We have made use of coding resources during this project, for example, GitHub Copilot chat functionality. One example use case of Github Copilot was to generate a general architecture for Deep Q learning. This was later adopted and tuned to our specific use case. We for example created a function for how to encode the states as inputs to the neural network, changed the neural architecture, and devised a different guiding function for choosing between exploration and exploitation.

C Supplementary figures

C.1 Actual Reference Route



(a) Route from Singapore to Port Moresby, Papua New Guinea, as generated by Searoute

On a scale of 0.2, 54107 states were generated, that were either ocean or sea portions of the South East Asia region depicting the Java Sea.

C.2 Parameter Tuning

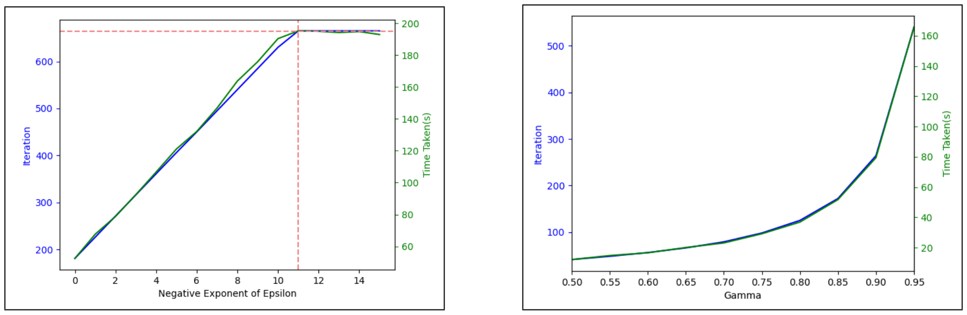


Figure 10: Number of iterations and time taken when varying VI parameters

C.3 Policy Evaluation

0	0	0	0	20	0	0	0	0
0	0	0	20	40	20	0	0	0
0	0	20	40	60	40	20	0	0
0	20	40	60	80	60	40	20	0
20	40	60	80	100	80	60	40	20
0	20	40	60	80	60	40	20	0
0	0	20	40	60	40	20	0	0
0	0	0	20	40	20	0	0	0
0	0	0	0	20	0	0	0	0

Table 1: Example of danger penalty in evaluation around a single pirate attack