**Understanding cooperation in AI part 1: The Bandit’s dilemma**

**Motivation**

As research into AI continues to grow, finding application in more and more areas of modern life, there has been an increasing urgency in the need to understand it. Simultaneously, popular culture and academia have taken an interest in finding differences and similarities between AI and organic life. My personal interest in this area has led me to consider famous tests of cooperation from other areas of research and use AI to recreate those experiments. By analogy to the study of mice as a simplified version of human psychology, I hope that the behaviour exhibited by the simple AI in these experiments can be used to inform the choices of those who create more powerful AI in the future.

**Introduction 1: The prisoner’s dilemma.**

In this classic example of interacting agents, two criminals are arrested and held without means of communicating. The prosecutors lack the evidence to convict on the principal charge, but they can convict both prisoners on a lesser charge. Each prisoner can betray the other by testifying that the other committed the principal crime or attempt to cooperate by staying silent. The payoff matrix below (Table 1) gives the number of years in prison that each prisoner would receive as punishment for each combination of actions.

It can be calculated that a prisoner with no control over the other’s actions will always be better off betraying than staying silent. We thus find a Nash equilibrium – the expected state if both prisoners play rationally – where both prisoners betray each other. However, many real people might suggest that if they had spent enough time bonding with the other prisoner during their criminal endeavours, there would be enough mutual trust to allow for both prisoners to stay silent and reduce their respective prison times compared to the Nash equilibrium. This possibility for mutual trust is what I wanted to explore in this project.

**Introduction 2: Multi-armed Bandits**

|  |  |  |
| --- | --- | --- |
| **B**  **A** | **B stays silent** | **B betrays** |
| **A stays silent** | -1  -1 | 0  -3 |
| **A betrays** | -3  0 | -2  -2 |

To replicate the multiple interactions that one imagines for the two prisoners building a mutual trust, I used a multi-armed bandit formulation. Originating from “One-armed bandit” slot machines, this class of algorithm picks one of *k* “arms” each round, receives a pay-out from that “arm”, and (if it is a learning algorithm) updates some metric for which of the *k* “arms” is most likely to give a large pay-out. Multi-armed bandit algorithms are commonly used for applications like AB-testing and action selection in reinforcement learning.

Table Payoff matrix for the Prisoner's Dilemma.

For this project, I used five types of multi-armed bandit algorithms to choose between the possible actions (stay silent or betray) for 200 rounds, learning – where applicable – from every round and updating accordingly. The random bandit picks to stay silent or to betray at random with an equal chance of picking either option. The rational bandit chooses to betray at every step, in accordance with the game theoretic Nash equilibrium. Although the rational bandit is the best choice for a single game, the fact that this experiment uses a repeating game actually leads the game theorist to an alternative optimal strategy: The optimal bandit will stay silent until the other prisoner betrays it, from which point it will always betray. The eGreedy bandit (using the eGreedy learning algorithm) has a hyperparameter value, , and will take a random action that proportion of the time. Otherwise, it will pick one of *k* actions, corresponding to the highest value of *,* . Each is calculated as

,

where r is the reward (number of years in prison), is a hyperparameter that varies how strongly the bandit weights the latest data compared to its existing knowledge and t is the current round. When , . Finally, the UCB bandit (using the UCB learning algorithm) has its own with its values updated in the same way but selects its next action by finding

,

where is a hyperparameter that varies how strongly the bandit favours exploring new options instead of exploiting the knowledge it has already gained and is the number of times that option has been chosen since the bandit started learning. The key difference between the eGreedy and UCB bandits from a behavioural perspective is that the eGreedy bandit has a random factor in its selection process whereas the UCB bandit is entirely deterministic.

**Results and Discussion**

Below are the results of a series of tests where all combinations of bandits were matched against each other, including two types of each learning bandit. All learned information and conditions were reset at the beginning of each matchup.

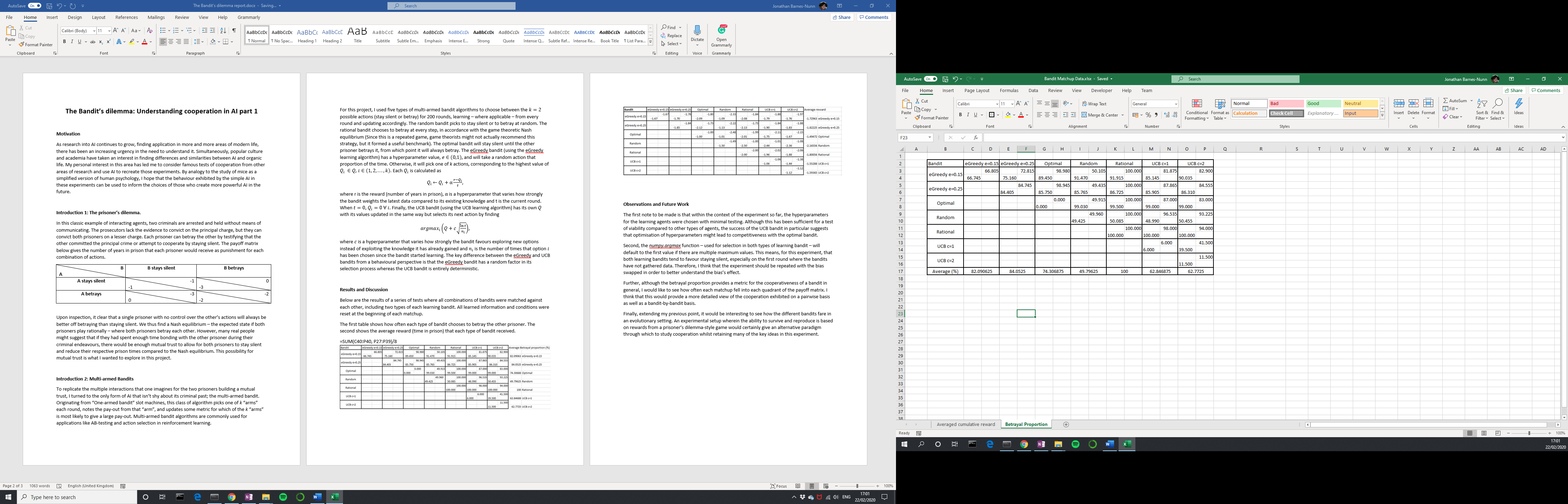


Table Average betrayal proportions for each type of bandit. Most values are averaged over 100 runs of the same matchup, the values in the bottom row are averaged over all runs and matchups.

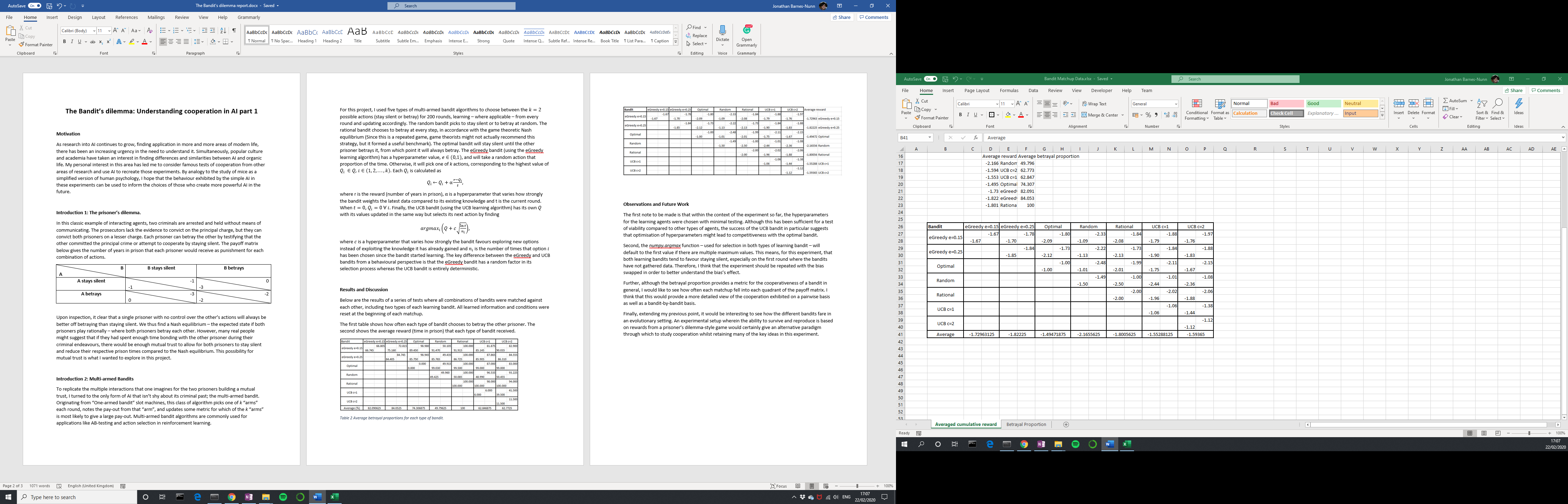


Table Average reward per round. Most values are averaged over 100 runs of the same matchup, the values in the bottom row are averaged over all runs and matchups.

Table 2 shows how often each type of bandit chooses to betray the other prisoner. Table 3 shows the average reward (time in prison) that each type of bandit received. From these it is clear that the benchmark (random, rational and optimal) bandits performed as expected. The random bandit chose both options equally and received the most negative average reward because of it. The rational bandit betrayed the other bandit every round and found its reward limited by the fact that most other bandits betrayed it in turn. The optimal bandit has a small average betrayal proportion, but this result is heavily biased by the pairing with itself. Considering only the matches between the optimal bandit and other types of bandit, the average betrayal proportion is 99.076%, the second lowest of all types of bandit. However, it outperforms all other bandits in average reward and in most pairings.

Comparing the results averaged over each type of bandit (Figure 1), there does not seem to be a strong linear correlation between betrayal proportion and average reward. The function does seem to have a maximum in the range though I suspect that the relationship is more complex than can be accounted for with this methodology.

A screenshot of a cell phone

Description automatically generated

Figure 1 Average reward and betrayal proportion for each type of bandit. From left to right: random, UCB c=2, UCB c=1, Optimal, eGreedy e=0.15, eGreedy e=0.25, Rational.

Looking again at Figure 1, the UCB bandit clearly outperforms the eGreedy bandit and the learning bandits that do less exploration (eGreedy e=0.15 and UCB c=1) outperform their more exploratory counterparts. This suggests that randomness and exploration are generally a hindrance to optimal performance. There are two possible reasons: the optimal bandit may truly be optimal and any improvements made to the learning bandits could functionally be the same as making them more similar to the optimal bandit or runs of 200 rounds may not allow for the more exploratory bandits to settle back onto cooperative behaviour as an equilibrium and receive consistently small rewards in that equilibrium.

**Observations and Future Work**

Although I have seen some interesting results already, this project is very much ongoing. I have listed below some observations and ideas that could allow me to extend this work further.

The first note to be made is that within the context of the experiment so far, the hyperparameters for the learning agents were chosen with minimal testing. Although this has been sufficient for a test of viability compared to the benchmark agents, the success of the UCB bandit in particular suggests that optimisation of hyperparameters might lead to competitiveness with the optimal bandit. As mentioned in the results section, the reduction of exploration seems to be the best way to achieve this optimisation.

Second, the *numpy.argmax* function – used for selection in both types of learning bandit – will default to the first value if there are multiple maximum values. This means, for this experiment, that both learning bandits tend to favour staying silent, especially on the first round where the bandits have not gathered data. Therefore, I think that the experiment should be repeated with the bias swapped in order to better understand the bias’s effect.

Further, although the betrayal proportion provides a metric for the cooperativeness of a bandit in general, I think that showing how often each matchup fell into each quadrant of the payoff matrix would provide a more detailed view of the cooperation exhibited on a pairwise basis as well as a bandit-by-bandit basis.

Finally, it would be interesting to see how the different bandits fare in an evolutionary setting. An experimental setup wherein the ability to survive and reproduce is based on rewards from a prisoner’s dilemma-style game would certainly give an alternative paradigm through which to study cooperation whilst retaining many of the key ideas in this experiment.

As a summary to those who wish to use this experiment to guide the principles of future AI design. The evidence gathered so far suggests that the optimal bandit is – as its name suggests – optimal for this formulation of the prisoner’s dilemma. Thus, I draw the conclusion that in a system that can be described simply and precisely, mathematical solutions are superior to statistical estimates and the only way to alter behaviour is to alter the system of rewards. This is, of course, an unoriginal claim but it is worth noting. However, if future work in this project adds layers of complexity, I am eager to see whether the optimal bandit still emerges as a unique solution.