Mei Kai Koh
mkkoh@uwaterloo.ca
20536826

Question 1:
The agent I implemented:
I was not able to discuss strategies beforehand or design agents that try to collude with my friends, as **I have no friends in this course**. 🥺 Please see the 2nd part of my answer for an explanation on how I would have implemented this if I was able to collude with friends.

My agent implements an adaptation of TfT. This variation is called Gradual. It is similar to TfT such that it cooperates on the first move, but if any of the opponents defects, the agent will defect as well. After *n* defections of any of the 2 opponents, the agent will also defect *n* times, and calms down with 2 cooperations to reset the opponent if the opponent is forgiving.

To explain how strategically powerful my agent is, I will be comparing it to TfT. TfT is a nice algorithm, such that it will try to cooperate and it does not hold grudges. In addition, it cannot be exploited since defections will be reciprocated. TfT can be implemented in a three-player IPD tournament by either ANDing or ORing the decision made by the 2 opponents. In *The Iterated Prisoners' Dilemma: 20 Years On*, it has been shown that two-player Gradual outperforms TfT in a noisy two-player IPD tournament with teams, though two-player Gradual did not outperform the colluding teams. However, the requirements of this questions are not for the agent to perform the best, but it is simply for the agent to perform well. Thus it suffices to create an agent that can perform worse than the properly colluding teams, at the level of TfT or higher (as we can use TfT as a benchmark for decent performance). Thus since two-player Gradual outperforms TfT in a noisy two-player IPD tournament with teams, three-player Gradual should be able to outperform some variation of three-player TfT (either ANDing or ORing the decision made by the 2 opponents) as well.

This technique is strategically powerful because it is nice, by cooperating by default, it reacts to the opponent with more severe punishments after each defect (which should disincentive the opponent to defect), and it forgives the opponent for defecting (by "calming down" with 2 cooperations to reset he opponent after retaliating), and because it has memory of how many times the opponent defected, it is better than TfT as it can look in the past and punish the opponent accordingly.

Dealing with noise:
In *The Iterated Prisoners' Dilemma: 20 Years On*, two-player Gradual was able to outperform TfT in a noisy two-player IPD tournament with teams, thus three-player Gradual should be able to outperform some variation of three-player TfT as well without any fancy noise-reduction algorithms.

The agent I would have implemented if I had friends in this course 😂:
Assuming I have *n* friends, we can define the first *k* iterations of IPD as a code for colluding. Let us define the code as **[Bool$_0$, …, Bool$_k$]**. Thus, if both opponents have the code **[Bool$_0$, …, Bool$_k$]** during the first k iterations (after checking for noisiness), we can collude and all play 0s for the remaining iterations.

If the opponents are not my friends, I will simply play the three-player Gradual agent as stated above.

Dealing with noise:
Since this is noisy channel IPD with 2% noisiness, we can define our agent to accept either a full match of **[Bool$_0$, …, Bool$_k$]** or some variation of **[Bool$_0$, …, Bool$_k$]**, where 1 and only 1 iteration may have a mismatch, ie **[!Bool$_0$, …, Bool$_k$]** is acceptable during the first k iterations.

In addition, there will be a check after every *x* number of iterations if my opponents are detected to be my friends on the *k+1*th iteration. This check will make sure that there is a maximum of one defection for every *x* iterations after we have detected that we can collude, to deal with noise. Else if there is more than 1 defection for *x* iterations, I will play the three-player Gradual agent as stated above, as I will not trust my friends anymore.

There is a possible loophole, where one of my friends finds out that my check is looking for 1 defection every *x* iterations. Thus my friend could sneak in a defection within the range of *x* iterations I am looking at, and I may consider that to be a result of the noisiness. And obviously, I could do the same and try to exploit my colluders, as they probably have a check similar to this.

References:
Kendall, G., Yao, X., & Chong, S. Y. (2007). The iterated prisoners dilemma: 20 years on. Singapore: World Scientific.

Mei Kai Koh
mkkoh@uwaterloo.ca
20536826

Question 2:

Paper: Superhuman AI for multiplayer poker

A) What are the motivations for this work?

Many important AI challenges involve multiple players and hidden information. AI advancement is often benchmarked through performance or solvability in games. However, most of the complex games which AI researchers have made incredible progress in (ie Go, chess, Dota 2, etc) have been two-player, zero-sum games. These games do not have much real-world significance as the real world deals with a plethora of hidden information and is not two-player zero-sum.

Six-player, no-limit Texas Hold'em is the most popular form of poker played by humans, and consists of dealing with incomplete information, exponential difficulty in achieving a Nash equilibrium (compared to a two-player poker game), and requires skills dependent on human psychology (such as bluffing). Creating an AI agent to beat professionals in six-player, no-limit Texas Hold'em advances AI research as it shows that AI agents could be successful in real-world multi-player scenarios involving hidden information, such as cybersecurity or taking down disinformation.

B) What is the proposed solution?

The proposed solution is to use the self-play, where the agent trains itself, using the **iterative Monte Carlo Counterfactual Regret Minimization (MCCFR) algorithm**, along with **action abstraction** and **information abstraction**.

During each iteration of the **MCCFR algorithm**, a player is designated as the "traverser" and a hand of poker is simulated based on the current strategy of players. Once the simulation hand is completed, the agent reviews the traverser's decisions and compares it with other possible decisions, as well as hypothetical decisions that would have been made following that decision. The difference between what the traverser receives for an action versus what the traverser actually received is called the **counterfactual regret**. Towards the end of the iteration, the agent updates the traverser's strategy to actions with higher counterfactual regret.

The agent uses **action abstraction**, which is an abstraction which reduces the number of actions that the agent must consider. For example, a $200 bet is similar enough to a $201 bet such that the agent does not need to consider both bet sizes, therefore reducing the number of decision states.

In addition, the agent uses **information abstraction**, which is a type of abstraction where the agent groups similar decision points based on revealed information, for example, a 10-high straight and a 9-high straight can be considered identical by the agent, further reducing the number of decision states.

The agent, using **MCCFR**, creates a **blueprint strategy** which is essentially the "strategy" of the agent. Pluribus plays the blueprint strategy during the first betting round, and conducts a real time search in subsequent rounds. The agent assumes that each player can play *k*-different strategies specific to each player, which is necessary in a complex game like poker, where strategies, deception, and bluffing can vary between rounds and players.

C) What is the evaluation of the proposed solution?

The **blueprint strategy** was able to be trained in just 8 days, using just $150 in cloud-computing costs. This agent was evaluated against elite poker players, many of which have earnings exceeding $10 million. In the first experiment, where the agent played 5 humans, Pluribus played 10,000 hands of poker against 5 elite poker players and had a win rate of 5 big blinds per 100 hands, which is a very strong victory against the human players (profitability p-value of 0.021).

In the second experiment, where a human played against 5 copies of Pluribus, each human played 5,000 hands of poker, Pluribus beat the humans by 32 milli big blinds per game (profitability p-value of 0.014).

The role of luck was also reduced in the evaluations, by using the AIVAT variance reduction algorithm, which is a technique for agent evaluation in imperfect information games. This allowed for achieving statistically significant results with roughly 10x fewer hands.

Overall, the 5 humans + 1 AI and 5 AI + 1 human experiments showed that Pluribus was able to master poker and the best poker players with statistical significance.

Mei Kai Koh
mkkoh@uwaterloo.ca
20536826

D) What are the contributions?

Six-player, no-limit Texas Hold'em is the most popular form of poker played by humans, and consists of dealing with incomplete information, exponential difficulty in achieving a Nash equilibrium (compared to a two-player poker game), and requires skills dependent on human psychology (such as bluffing). Creating an AI agent to master the game of six-player, no-limit Texas Hold'em has been the goal of researchers in game theory and AI for decades. Thus, solving such a complex game for the first time is a milestone in artificial intelligence, and was the remaining milestone in poker.

Pluribus has shown that AI agents outside of two-player zero-sum games do not need to have a theoretical guaranteed convergence to a Nash equilibrium in order to be successful. Pluribus was still capable of producing superhuman results. Pluribus has shown that AI agents can produce superhuman results in large scale imperfect information settings through self-play-with-search algorithms. The techniques used in this paper can apply to other complex multi-agent problems that we are facing.

Pluribus has shown that complex AI agents that operate outside of two-player zero-sum games can be trained at low costs ($150) and with techniques not specific to poker (**MCCFR**)

Pluribus has provided an outside perspective on what optimal play should look like in multiplayer Texas Hold'em. For example, Pluribus agreed that limping, or calling a "big blind" rather than folding/raising, is sub-optimal except for the "small blind" player. In addition, Pluribus disagreed that "donk betting", or starting a round betting when one ended the previous one with a call, is a mistake.

E) What are the future directions for this research?

Pluribus can train elite players to get even better strategies, similar to how AlphaZero has helped discover groundbreaking chess strategies. **MCCFR**, along with **action abstraction** and **information abstraction** can be applied to other challenging games with large state spaces, complex branching, and hidden information. Pluribus has shown that real world situations involving hidden information and multiple agents such as cybersecurity or fraud prevention can possibly be modelled and solved at superhuman levels. Pluribus has shown that AI agents can be created without expert domain knowledge (**MCCFR** is not specific to poker) and gives hope that other complex AI problems can be solved without huge amounts of compute.