I also realise, in the later stages of the project, that if such a system is proven to be useful and less exploitable than others, it would have meant not only breakthrough in the game of bridge, but also in artificial intelligence, for the following reasons.

Communication

Bidding in essence is a form of communication, where players selectively disclose important parts of their hands to their partners, and sadly their opponents too. Bidding systems can hence be regarded as a language, despite having difficulties only faced by its kind. For example, in everyday languages such as English, we have a wide range of vocabulary as well as limitless amount of space, granting it the capability to express any abstract and concrete part of our world, albeit occasionally inefficient.

This is contrasted by the limited number of tokens and amount of space in bidding. While the number of tokens, 36, is not exactly small, each token comes with two hidden costs when being bid. The first one is that each bid robs the partner of linearly increasing amount of bidding space. For example, a 6NT bid leaves only five biddings possible responses for the partner. This is why bidding sequences rarely starts at the 4-level, leaving effective starting bids to be 15. Secondly, each bid comes with the cost that the pair will end up playing at the named level, at the very least. For example, staring from 1C-1S-3C, the pair has committed themselves of playing at the 3-level.

These two constraints together limit the usual bidding sequences to a length of six. And in 3 bids, the pair has to convey information about their shape (length of each suit), strength (usually in the form of High Card Point, calculated from face cards) and controls (presence of Aces and Kings). At least this is the case in existing human bidding systems.

If our artificial agent is able to construct and use a language in this sandbox-like environment, a language that captures the essence of each hand, conveying in a manner and sequence such that most scenarios could be covered as partner branches off in each node, this would hopefully point to a new direction in knowledge representation and compression, easing the workload of devising a new communication paradigm as we introduce AI in other aspects of the world.

Cooperation

As more and more aspects of our lives suggest machine learning to be a solution for automation, the lack of cooperative elements in these agents becomes more and more apparent. We can see agents solving problems intelligently without explicit directions in the area of speech recognition, chess, Chinese chess, scheduling, but we rarely hear news, if any, about two agents cooperatively learn from scratch and achieve substantial feats.

The ability to cooperate, however, is crucial to push the boundary of artificial agents to the next level. Not only because this is the natural way of development, taking reference of various living beings, but also because the limitation of a huge, powerful entity has always proven to be less impactful than distributed competent components – the Internet would not be anywhere near as successful if it is based of centralized computing.

If the function, together with the values in the domain are all known, then the solution could be calculated by brute-force dynamic programming.

Pa (s, s’) = P (St+1 = s’ | st = s, at = a) is the probability that action a at time t would lead state s to state s’

Ra (s, s’) is the immediate reward received after transitioning from state s to state s’ due to action a

The core problem of the MDP is to specify a policy, or action to each possible state, that maximizes a function of the rewards attained in the process. Such a function may simply be cumulative sum, or in most cases, discounted sum.

If all the states, actions, rewards and probabilities are known, the solution can be found by brute force dynamic programming with a large enough memory. The value of each state the expected return of each action at each state could be gradually updated. This updates the optimal policy which ultimate converges as long as no states are permanently left unvisited.

This, however, assumes that all the defining features of a MDP is known prior, which is usually not true. In particular, the state transition probability is usually governed by chance or another decision maker’s policy, making it impossible to directly obtain.

Reinforcement learning algorithms perform particularly well in this area, by directly sampling states, making actions and interacting with the environment, it avoids the need of a transitioning probability.

Elimination of Luck

Among wide incomplete information games, bridge is in a special location where chance is involved but rarely is luck. This is due to the different formats of tournaments and the special cooperative property of bridge.

In team-based tournaments, bridge players are teamed by at least four players, or two pairs. When competing against other teams, the two pairs become N-S pair and E-W pair at different tables, facing the other team’s E-W pair and N-S pair respectively. The players in the same position of the two tables, which are of different teams, are then always dealt the same hand and placed under the same vulnerability in each deal. The dealer assigned in each deal is also equal. This duplicative property eliminates the luck involved in the strength of each dealt hand.

Bridge also discourages test-of-luck moves with its cooperative property and complete reveal obligation. If the game is non-cooperative, players would have gotten hold of most information needed at the start of the game, leaving test-of-luck moves with a higher expected return. In bridge, however, only half the information is revealed to each player and such move has a much higher risk. Even if players still continue to do so, due to complete reveal obligation, this move is either completely transparent to both opponents and partner or completely oblique to both. In the first scenario, opponents could then do an appropriate response after differentiating bluffing from real strength; And in the second scenario, players have to bear the risk of partner re-raising or changing contract, since he/she is provided with incorrect information.

If the function, together with the values in the domain are all known, then the solution could be calculated by brute-force dynamic programming.

Pa (s, s’) = P (St+1 = s’ | st = s, at = a) is the probability that action a at time t would lead state s to state s’

Ra (s, s’) is the immediate reward received after transitioning from state s to state s’ due to action a

The core problem of the MDP is to specify a policy, or action to each possible state, that maximizes a function of the rewards attained in the process. Such a function may simply be cumulative sum, or in most cases, discounted sum.

If all the states, actions, rewards and probabilities are known, the solution can be found by brute force dynamic programming with a large enough memory. The value of each state the expected return of each action at each state could be gradually updated. This updates the optimal policy which ultimate converges as long as no states are permanently left unvisited.

This, however, assumes that all the defining features of a MDP is known prior, which is usually not true. In particular, the state transition probability is usually governed by chance or another decision maker’s policy, making it impossible to directly obtain.

Reinforcement learning algorithms perform particularly well in this area, by directly sampling states, making actions and interacting with the environment, it avoids the need of a transitioning probability.

Bridge programs with sophisticated abstractions and bidding engines already exists, and we were hoping that the impact from a new bidding system, or a new use of bidding, could bring impact to the stale Bridge community.

While in the section system architecture, we mentioned the full approach and how game master would administer a self-play game of four agents. In reality, due to slow speed of convergence, we set the dealer to always be North and every player to be at vulnerable position. Eventually only the N-S pair is allowed to bid freely. This was not against our goal of observing whether agents could cooperate and communicate without designer’s intervention.

T

As it became more and more apparent that our network could not handle such a level of complexity, we simplified the problem in multiple stages. In the first stage, the dealer and the vulnerability of each deal is fixed. The dealer is always North and all seats are always at vulnerable position. The first choice was completely arbitrary and merely saves the coding nuisance since player North has a player code 0. The second choice was based on the fact that our agent at that time always overbids, and we hoped that a greater penalty would deter such behaviour.

In the second stage of simplifying the problem, we swapped out the agents at E-W positions and replace them by dummies, which fits into the API and only bids “Pass”. By only allowing N-S pair to bid, we were hoping they could develop a bidding system without interruption before considering the more complicated case of competitive bidding.

In the last stage of simplification, the deal generated is fixed and hence the hands N-S pair receive were also fixed. The statistics of the N-S hands were also transparent and given at the beginning, eliminating the need of using a bidding base. We hoped that such an approach could at least let the agents implicitly perceive each other’s action as meaningful and could exhibit cooperation in some ways.

The incapability to evaluate the situation and past bids is undoubtedly lethal to the learning of our agents, yet the tasks they truly face are even more difficult than that. Since our goal of the project is to build a bottom up bidding base, hence during the agents’ search for correlation between cards and bidding, it was impossible to do so. The means the bid of any contract would be equally good at first, since any bid could fit any holdings, and any bid could be good if partner “held that hand”.