Dissertation Review of "Multi-Agent Reinforcement Learning in Markov Games"

Sina O. Azartash SAZARTA1@JH.EDU
Johns Hopkins University

Abstract

This paper summarizes and critiques the PhD dissertation of Dr. John Wilbur Sheppard, "Multi-Agent Reinforcement Learning in Markov Games", published 1997. Specifically, we present an overview of his research, review related literature, analyze his major contributions, describe his significant algorithms, compare his work to modern advancements, and discuss some areas of applications. Sheppard uniquely focuses his research on two competing agents learning together applied to the problem of challenging differential games. He begins by evaluating a genetic algorithm, k-NN classification, and Q-learning applied to two differential games of pursuit. The performance of these algorithms inspire him to construct a co-learning method which utilizes the strengths and weaknesses of each algorithm in a complimentary way. The result was a genetic algorithm that acts as a teacher to 1-NN which demonstrated superior performance to either single learner. This sets the stage for developing two novel algorithms: Memory Based Co-Learning (MCBL) and then later Tree-Based Co-Learning (TBCL). MCBL provided a strategy to learn solutions to differential games from examples stored in a memory base. TCBL functions similarly except that instead of storing examples, it constructs a decision tree that partitions the state space of the game. Both MCBL and TCBL were run and analyzed on the simple game of force, the pursuit game, and two more challenging variants of the pursuit game. Overall, Sheppard's strong demonstrations of competitive co-learning contributed to the inspiration for many more complex multi-agent reinforcement methods 25 years later.

1. Scope

Multi-Agent Reinforcement Learning (MARL) in Markov Games is a relatively broad topic with many different specializations. Sheppard specifically focuses on discrete simulations of two-player differential games in which opposing agents compete to learn together. The background, context, and motivation for this refined focus is explained below.

Generally, systems in which multiple agents or a population interact to achieve some goal or accomplish some task is called Distributed AI [Distributed AI Ref]. The research in this paper was on multi-agent systems which is different in that it focuses on the behavior development of individual agents Sheppard (1997). Furthermore, the focus of this research was agents learning through competition. For this reason, experimentation was applied to zero-sum games in which the payoff received by one player is at the equivalent expense of the other player.

Research stems from the foundation of reinforcement learning applied to the Markov Decision Process(MDP). In this model, a single agent receives feedback from an environment in which the current state depends only on the previous state and the state transitions are modeled by a probability function Littman (1994). The agent objective is to find a policy that maximizes expected sum of discounted rewards. This is done by mapping a history

of states, actions, and rewards. The discount factor controls how much future rewards vs near-term rewards have on the optimal policy. Markov games extends this to allow multiple agents to adapt to each other with competing goalsSheppard (1997). This causes the optimal policy for each agent to become dynamic and contextual because agents have to account for the reactive policy changes of other agents. As the number of agents increases in a game, the game becomes exponentially more complex. Handling this complexity was not relevant to the goals of Sheppard's research and therefore two player games were chosen.

During the time of this dissertation, there was relatively little work on co-learning applied to game theory and most of it was limited to relatively simple games.One similar work of co-learning on differential games was available, but it focused on temporal difference learning with a connectionist approach, while Sheppard focused on symbolic reasoning systems. Furthermore, this other work produced pure-strategies while Sheppard also allowed for mixed strategies.Sheppard (1997)

Sheppard chose more challenging games. They were challenging mainly for the following complications. First, the games selected were multi-step with delayed feedback meaning that an agent has to take multiple steps for each game and feedback is not received immediately. This complicates the reinforcement learning process in that approximating the value of an individual action in a sequence of actions becomes much more difficult. A successful victory in a game does not necessarily indicate that a single action taken was beneficial. Likewise, a failed game does not necessitate that an action taken in that game was harmful. Sheppard chose games that payoff was not received until after the game was over. Second, Sheppard focused on learning differential games on-line during play. This is much more challenging because the moves are simultaneous, the state and action space are continuous, and the state transitions are modeled using differential equations Issacs (1976). Handling simultaneous turns in a game is a more challenging problem than having players take turns sequentially. The differential equations model how actions taken by the player changes the states of the game over time. The objective of a differential game is to approximate the value of the game based on the game's payoff and to determine the optimal strategy for each agent that yields this value. Lastly, Sheppard focused on the problem of handling Markov games with a large state-action space. This causes many traditional reinforcement algorithms to become intractable. Model based reinforcement algorithms such as value iteration and policy iteration become impractical because they require full sweeps of the state action space in order to find the optimal policies. Model-free based models such as Q-Learning become difficult to use in that the lookup table grows to intractable size with a large action-state space. In summary, one significance of Sheppard's work is creating algorithms that can handle these complications.

2. Problem

As mentioned, the main problems were not receiving feedback until the end of the game and a state-action space that is both continuous and large. A variety of alternative methods are discussed in the dissertation, but only the main ideas relevant to the design of the algorithms section are covered. In short, to handle a continuous space, games are discretized into specific time stamps with a controlled resolution. Temporal difference learning is used to address delayed feedback multi-step prediction. Next, function approximation, memory

based methods, and genetic algorithms were methods for addressing the large space. Finally, some additional challenges arose from the selected approach. These problems were resolved using example selection and memory editing approaches.

Temporal difference learning focuses on predicting discounted payoff from a given state. The discount is similar to the discount factor used in Q-learning in that a low discount factor means past updates have less influence and a high discount factor indicates that past predictions receive more of an equal weightSutton (1988). In fact Q-learning is a special form of temporal difference learning where there is one admissible action for each state Dayan and Sejnowski (1994). Essentially, this is an extended form of supervised learning based on gradient descent. The main idea is to construct a weight vector that minimizes the error between predicted value and observed value as the weight vector gets updated. The weight vector is updated by a set of state vectors paired with the associated outcomes. Likewise, in multistep prediction, the observed state-actions and associated rewards are stored in a batch where one batch represents one game. The batch is applied to the weights only after the end of each game.

The main idea behind handling the large state action space is to reduce time complexity by avoiding extra calculation. Function approximation techniques lower time complexity by estimating value rather than calculating it. Instead of determining the value of each state, a set of features associated with each state are generalized to states with similar features Li et al. (2021). In other words, function approximation seeks to find the most similar state and then uses that similar state to estimate value. Examples of function approximators include linear combination of features, kernel functions, nearest neighbor approaches, decision trees and deep reinforcement learning. In this paper, Sheppard decides to use k-Nearest Neighbor classification (kNN).

Memory Based Learning is an approach to function approximation involving external memory. Analogous to episodic memory in humans, successful policies are remembered as examples in memory as a database[Ramni A SHORT SURVEY ON MEMORY BASED REINFORCEMENT LEARNING 2019]. The examples represent regions of the state-action space. When evaluating a state in a game, the most similar example(s) in the memory database are searched for using classification. The action mapped to the selected similar example(s) is then associated with the current state being evaluated. The process follows that each time an agent gains a larger reward than the expected a priori, the state-action pairs sequences that lead up to the reward are stored in the database. Ramani (2019). Typically, when multiple sequences are available for a given state, then the sequence with the maximal reward is chosen.

Using Memory based learning introduces new additional problems. The quality of the examples stored in memory can significantly influence performance. Without a priori information, the examples in memory are initialized randomly and then improved upon through trial and error. As mentioned in the scope section, a significant challenge of these experiments was that a winning outcome in a game does not guarantee an action was beneficial. Therefore, the examples stored in the database are not guaranteed to be optimal. Several non-optimal examples in the database may limit the performance of the agent. In addition, the concept of storing successively similar examples to explore the state-action space incurs a type of bias in which the full-state action space may not be explored.

To address these issues associated with memory based learning, Sheppard included a method for selecting pertinent examples and a method for sufficient representative exploration in his memory based algorithms. To encourage exploring different regions of the state-space, a probability value was assigned to sometimes select a random action instead of the action with the highest expected reward. To select better examples, a secondary algorithm was selected to serve as an oracle to guide the memory based algorithm. The memory based algorithm used kNN and the secondary algorithm was a genetic algorithm.

An alternative to memory based algorithms for learning sequential rules in Markov games is Genetic Algorithms(GAs). A precursor to GA was classifier systems applied to MDP[Shep citation 50 pg 81]. Briefly summarized, the classifier generates a series of rules that are associated with an expected payoff. A rule is binary condition-action pair. A higher expected payoff indicates a higher fitness value for that rule. The rule conditions are activated by a message list which triggers a bid in a procedure referred to as the bucket brigade to see if that strength of the rule should be updated. Similar to the classifiers system, in GAs classifiers are modified in a series of reproducing generations where the probability of reproducing is proportional to the fitness value. The classifiers selected for reproduction are modified through mutation and crossover which promotes exploration of the space of classifiers. In short, mutation randomly modifies a classifier and crossover swaps attributes of a classifier.

Finally, a problem that occurred during experimentation was that the algorithms had a high computation time. Tens of thousands of games up to 100,000 games were required to learn a game which took from several hours to a few days. Sheppard used memory editing methods to modify the algorithms to require less memory, but retain the majority of performance. Specifically, Wilson's algorithm Wilson (1972) was combined with the Ritler method Ritter et al. (1975). Wilson's Algorithm classifies training examples in a database using k-NN. Examples that are incorrectly classified are deleted from the example set, reasoning that such points are likely noise. Riter et. al discards correctly classified points so that only points near the classification boundary exist.

3. Approach

An overview of the contributions surrounding the experimental design and the results are provided. Their rationale of the approach and significance of the results are also further discussed.

Sheppard chose the pursuit game to evaluate his algorithms. A pursuit game is a special type of differential game with 2 players: (P) the pursuer and (E) the evader. The evader attempts to escape while the pursuer's attempts to prevent the evader's objective. The game is based on the Homicidal Chauffeur Game [citation] , but more difficult because the pursuer has a speed variable and the evader has a radius of curvature. Also, the evasive task on the Homicidal Chauffeur Game has a highly nonlinear solution surface which makes the pursuit game difficult to characterize.

Sheppard extends three variations of pursuit to be more difficult. First, an immovable boundary or wall is placed at x=0 to block movement. The results showed that optimal strategies away from the wall stayed the same, but this complicated the optimal strategies when an agent came near the wall which was evident through a slower learning conver-

gence. Player strategies already depended on the position of the other player, but now also depended on the wall. Furthermore, the average performance for optimal play was more in favor of P because the wall forced E to move sub-optimally. In the second variation, player mobility was limited. P could only turn plus or minus 45 degrees while E could only turn 90. This complication actually made it so no optimal solution was available. To compare the learned policy against the optimal policy, a heuristic was substituted as the optimal policy. The heuristic defined P to aim as closely as possible at E and for E to turn sharply. In the third variation, two pursuers chased E and E was given extra capabilities. Specifically, E can change its speed, turn angle, introduce noise in P, and is supplied with 13 features describing the state of the game. The results showed this made learning much more difficult as sometimes there is no possibility of escape, and sometimes, there is no known optimal strategy.

4. Contributions

Sheppard demonstrated the application of co-learning to differential games. The main contributions are as follows. First, Sheppard provided an extensive literature review of machine learning, multi-agent reinforcement learning, and co-learning techniques on Markov games. Second, he evaluates the strengths and weaknesses of three algorithms applied to difficult pursuit games described in experiment one. Third, he shows that using a second learning algorithm to generate examples for a memory based learning algorithm in co-learning increases performance farther than what either algorithm achieves on their own in experiment described in experiment two. Fourth, Sheppard introduces and demonstrates a novel memory based co-learning algorithm in experiment three. In his fifth contribution, he introduces and demonstrates an alternative algorithm that uses decision trees instead of memory in experiment three.

5. Results

This section described the experiment that supported the conclusions of the contributions. Then, the algorithm associated with that experiment is briefly described.

5.1 Experiment One: Evaluating Learning Strategies

3 learning strategies were compared and contrasted on the single pursuit game and the double pursuit game. The algorithms were a memory-based version of standard Q-Learning and a kNN, and GA. For kNN, one neighbor was used (1-NN). For both the single and double pursuer games, Q-Learning performed well initially, but then plateaued. For 1-NN, the results were polar. The evasion success was overall very high with 1 pursuer, but very low with 2 pursuers. The poor performance of 1-NN for the 2-pursuer problem was attributed to the presence of irrelevant attributes, a larger search space, and most likely, generating bad examples at early phases in the game. GA performed excellently on both problems, but started more slowly.

The Q-Learning algorithm does not use a complete look up table. Instead, it stores examples in memory. In contrast, in 1-NN state variables are treated as features and actions are treated as class. The outcome of each game is used to label the class. Both

the memory based Q-Learning and 1-NN searched a database to find the nearest examples on the assumption that the most similar state is assumed to require similar actions. The matching is done using a similarity threshold based on the normalized euclidean distance of action and state. For Q-Learning, if there does not exist a match of the current state with an example in the database, then an action is chosen at random. This fosters exploration. Otherwise, the candidate in the database is combined with all available actions for the candidate state, and the state-action with the highest expected reward is chosen. Also, there is a 30 percent probability that a random action is chosen regardless of the best action which also facilitates exploration. Newly encountered state-action pairs are stored in memory and current ones are updated using Q-learning. For 1-NN, a state-action sequence is searched for that leads to E evading P for a complete game. If the resulting sequence does not result in E winning, then random actions are selected and the game is run again until E wins.

For the GA, a population of plans compete based on the highest fitness. A single plan is a set of 20 rules with each rule having a rule strength. Initially, all rules are maximally generally which means all rules match all states. Following training, rules that are fired are generalized or specialized using hill climbing by modifying upper and lower bounds. If there is a match, then both bounds of the rule shift closer to the state. If there is no match, only the closer bounds shift. The strength of the rules are updated based on the payoff received from the game using a profit sharing plan. Plan fitness is determined by running each plan against a set of games and computing the mean payoff for the set of tests. Rules within a plan are selected for mutation and crossover using fitness proportional selection.

5.2 Experiment Two: GAME and GAMB

Sheppard introduced the bootstrapping algorithm named Genetic Algorithm and Memory Based Learning (GAMB) in which GA serves as a teacher for 1-NN. The inspiration was that each algorithm brings a different set of skills to the learning task. Indeed, results showed both learned better than they would have on their own reaching a performance of nearly 100% on the two-pursuer problem. Next, Sheppard reduced the memory requirements of GAMB without significantly affecting performance. The resulting algorithm was called GAME (Genetic Algorithm Memory based learning with Editing). Results of editing showed significant success. Evasion accuracy decreased logarithmically until almost all examples were edited. As little as 11 examples achieved 80% evasion, 21 examples produced 86% evasion and 66 examples produced 90%. In GAMB, GA is run until a performance threshold is reached and then communicates its findings to 1-NN. As 1-NN learns, GA adds some additional examples until the combined system reaches an asymptotic limit. A performance threshold of 0, 50, and 90 percent was tested before GA could teach 1-NN. 90 percent was found to yield the best performance and contained a smoother line indicating 1-NN was supplied with fewer bad examples. Interestingly, the performance of GAME was even higher than GA. This is believed to occur because only the best examples of a given generation were passed to 1-NN.

Next, Sheppard reduced the memory requirements of GAMB without significantly affecting performance. The resulting algorithm was called GAME (Genetic Algorithm Memory based learning with Editing). The editing procedure used was classifying examples based

on matching associated actions. A k-NN with K=5 classified each example based on all the other examples in the database set. Classification was based on the normalized euclidean distance between neighboring actions being less than a pre-specified threshold. Correctly classified points were then deleted with a 25 % probability.

5.3 Experiment Three: MBCL

Sheppard introduces the novel algorithm Memory-based Co-Learning (MCBL) in which k-NN identifies the most similar example in the database and Q-learning updates the expected payoff of the examples in the database. Three types of games were generated in equal partitions using a uniform probability distribution: random actions for both players, random actions for E while fixed for P, and random actions for P while fixed for E. Briefly, for each state in the game, k-NN provides similar examples. The total moves available for the similar examples form a candidate move set for each player. This candidate set determines the most similar moves using k-NN for the other player. An actual action is generated for each player by a uniform probability distribution applied on the candidate move sets. This promotes exploration. An expected payoff matrix is constructed by calculating the Q-value of the current state with an action from each player. The current Q-value is updated by adding the old Q-value to the discounted maximum Q-value for the next state that results from applying the suggested action pair.

5.4 Experiment Four: TBCL

Sheppard introduces Tree Based Co-Learning (TBCL) as an alternative to MBCL to reduce space complexity. TBCL is not a memory based method, but instead iteratively partitions the state space using a Kd-tree. Kd-trees reduced the computational complexity of kNN to logarithmic expected time. The leaves of the tree contain a game matrix which represents behavioral strategies. Learning consists of updating the game matrix based on actual play and resolving the linear program. The decision tree is split if the performance of a node is lower than a threshold.

The TBCL algorithm is summarized as follows. Learning begins with the root node representing the entire state space. Since the root node is initially also a leaf node, the root contains a game matrix which contains uniformly distributed random values. Using linear programming (LP) on the game matrix, mixed strategies are calculated for both players. Pure strategies with a probability of zero are dropped. These strategies are run on the game which Q-Learning uses to update the game matrix. Then, new mixed strategies are calculated again using LP. A loop of game playing, Q-Learning, and LP is continued for a fixed number of iterations. If the performance of the resulting strategies does not meet the performance threshold, the node is split. Splitting a node is performed by selecting the node's state variable and then splitting the state space along the midpoint defined by that variable. The state variable selected is the variable that maximizes the euclidean distance between the two resulting game matricies. After splitting, the learning loop is restarted on each node until a partition is found that meets the threshold performance. In order to play the game, the mixed strategies for both players are found by traversing the tree to find the partition that covers the current state.

6. Applications

The implications of Sheppard's work are significant because it demonstrated many possible applications and inspiration for future research. Sheppard showed that competing agents can be used to learn from each other in differential games using a shared memory database or a decision tree. Some interesting modifications would extend into partially observable Markov games or hidden markov models. Several ideas of application inspired from his work are discussed.

The first idea comes from the fact that different agents can use the same data structure or some shared data configuration to exchange learned knowledge. One of the most pressing problems in Sheppard's research was the computational burden. A variety of more efficient data structures, preprocessing, and data pipelines can make his algorithms much more practical. Also, methods analogous to transfer learning and partial transferring learning could be used to exchange information between trained agents and untrained agents. In other words, the presence of trained agents could speed up untrained agents faster than what a priori information in data could accomplish.

Next, using a shared information medium to store the experiences of agents inspires testing several additional algorithms. In this paper, Sheppard compared an eager approach and a lazy approach to differential games. There are several different classes of reinforcement learning algorithms and other machine learning methods including neural networks that all have their own strengths and weaknesses. Different combinations of algorithms can produce more complimentary results on Markov games.

In addition, more than two agents or synthetic unique agents can be included in the helpful teacher scenario. There could exist several configurations of multiple teachers and/or multiple students that optimizes a complimentary approach. Furthermore, there could exist a configuration in which training starts with two agents and then scales up to three agents after a certain performance threshold. More agents could be added gradually to handle more complexity and be more general. On the other side, specific algorithms could be used to train an optimum helpful teacher prior to the game. The optimum helpful teacher would maximize the learning rate of the primary learning algorithm.

In the most interesting case, one of the agents could be a human. The first agent could find the most succinct example to present to humans using active learning. An algorithm would minimize the number of examples needed and maximize the learning effect of an example to make human involvement possible. Alternatively, two agents could compete against each other in a game and humans could intervene only when the two agents get stuck.

7. Related Work and Advancements

Since 1997, finding specific advancement in memory based co-learning has been difficult, but there have been a variety of advancements to Multi-Agent Reinforcement Learning.

Much research has focused on combining multi-agent reinforcement learning with deep neural networks Ramani (2019). Deep Q-Networks is such an example using neural networks to better approximate the Q-function . Meng et. al introduces a memory based deep learning on POMDP's by using Long-Short Term Recurrent Neural Networks that are memory

based Meng et al. (2021). Wang et. discusses an improvement to the evaluation function in least-squared variant of value iteration. The action-value function was represented by a kernel function or an overparameterized neural network.Yang et al. (2020)

Similar to co-Learning with competitive agents is Actor-Critic algorithms. The actor method learns the policy and critic methods rely on value function approximation. Konda and Gao (2000). The algorithm combines the strengths of these methods in which the critic uses an approximation architecture to learn the value function which updates the actor's policy parameters.

Multi Agent Reinforcement Learning in which the human participates as an agent with AI agents is referred to as Collaborative Reinforcement Learning (CRL) Li et al. (2021) This is not to be confused with Collaborative Multi Agent Reinforcement Learning which uses global joint action rewards to model individual rewards Kok and Vlassis (2006). Wang et. al introduced active learning assisted by a helpful teacher. The teaching algorithm provided a sequence of adaptive contrastive examples to the learner to speed up the learning process. Wang et al. (2021)

References

Peter Dayan and Terrence Sejnowski. Td() converges with probability 1. *Machine Learning - ML*, 14:295–301, 03 1994. doi: 10.1007/BF00993978.

R. Issacs. Differential games, 1976.

Jelle R. Kok and Nikos Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *J. Mach. Learn. Res.*, 7:1789–1828, dec 2006. ISSN 1532-4435.

Vijay Konda and Vijaymohan Gao. Actor-critic algorithms. 01 2000.

Zhaoxing Li, Lei Shi, Alexandra I. Cristea, and Yunzhan Zhou. A survey of collaborative reinforcement learning: Interactive methods and design patterns. In *Designing Interactive Systems Conference 2021*, DIS '21, page 1579–1590, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384766. doi: 10.1145/3461778.3462135. URL `https://doi.org/10.1145/3461778.3462135`.

Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In William W. Cohen and Haym Hirsh, editors, *Machine Learning Proceedings 1994*, pages 157–163. Morgan Kaufmann, San Francisco (CA), 1994. ISBN 978-1-55860-335-6. doi: https://doi.org/10.1016/B978-1-55860-335-6.50027-1. URL `https://www.sciencedirect.com/science/article/pii/B9781558603356500271`.

Lingheng Meng, Rob Gorbet, and Dana Kulić. Memory-based deep reinforcement learning for pomdps, 2021. URL `https://arxiv.org/abs/2102.12344`.

Dhruv Ramani. A short survey on memory based reinforcement learning, 2019. URL `https://arxiv.org/abs/1904.06736`.

G. Ritter, H. Woodruff, S. Lowry, and T. Isenhour. An algorithm for a selective nearest neighbor decision rule (corresp.). *IEEE Transactions on Information Theory*, 21(6):665–669, 1975. doi: 10.1109/TIT.1975.1055464.

John Wilbur Sheppard. *Multi-Agent Reinforcement Learning in Markov Games*. PhD thesis, USA, 1997. UMI Order No. GAX97-30786.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Mach. Learn.*, 3(1):9–44, aug 1988. ISSN 0885-6125. doi: 10.1023/A:1022633531479. URL https://doi.org/10.1023/A:1022633531479.

Chaoqi Wang, Adish Singla, and Yuxin Chen. Teaching an active learner with contrastive examples, 2021. URL https://arxiv.org/abs/2110.14888.

Dennis L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-2(3):408–421, 1972. doi: 10.1109/TSMC.1972.4309137.

Zhuoran Yang, Chi Jin, Zhaoran Wang, Mengdi Wang, and Michael I. Jordan. On function approximation in reinforcement learning: Optimism in the face of large state spaces, 2020. URL https://arxiv.org/abs/2011.04622.