UNDERSTANDING SINK STATE FORMATION IN FINITE STATE MACHINES USING THE PRISONER'S DILEMMA

by

CAMERON DAVID PIPES

(Under the Direction of Adam Goodie)

ABSTRACT

This paper investigates what behavior causes the formation of sink states in Finite State Machines (FSM) through the lens of the Iterated Prisoner's Dilemma (IPD) game. In the IPD, two players play multiple rounds where each round they can choose to cooperate or defect. A FSM playing the IPD will decide to cooperate or defect based on its trained arrangement of states and transitions. If all of the transitions from a state return back to that same state, that state is a sink state. This paper finds that the chance of training a sink state is relative to the method of training the FSM. Evolutionary learning techniques train sink states more often than reinforcement learning techniques in most scenarios. This investigation provides insight into the nature of finite state machines that could be applied in other domains.

INDEX WORDS: Finite State Machine, Sink state, Iterated Prisoner's Dilemma, Reinforcement learning, Evolutionary learning, Tragedy of the Commons

UNDERSTANDING SINK STATE FORMATION IN FINITE STATE MACHINES USING THE PRISONER'S DILEMMA

By

CAMERON DAVID PIPES

B.S., The University of Georgia, 2020

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2022

UNDERSTANDING SINK STATE FORMATION IN FINITE STATE MACHINES USING THE PRISONER'S DILEMMA

by

CAMERON DAVID PIPES

Major Professor: Adam Goodie

Committee: Prashant Doshi
Kimberly Van Orman

Electronic Version Approved:

Ron Walcott
Vice Provost for Graduate Education and Dean of the Graduate School
The University of Georgia
May 2022

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

Purpose of the Study

The purpose of this study is to develop a better understanding of sink states via finite state machines trained to play the iterated prisoner's dilemma. Existing scientific literature on the formation of sink states by machine learning agents is limited. This paper aims to provide insight on the positive and negative effects of sink states and investigate their impact on the function of a machine learning agent. The context of the iterated prisoner's dilemma is used to train the finite state machines due to the game's relevance to many real-world settings, such as economics and psychology.[19, 20] It is the belief of the author that the results of this paper will help inform future analysis of trained machine learning models that contain sink states.

Prisoner's Dilemma

The prisoner's dilemma is a well known game in the field of game theory.[2, 23] It presents a situation where two players choose to cooperate with each other or defect. A payoff (also called score or reward) is given to each of the players based on the actions that are chosen. The four possible outcomes of the game are given below.

If you play cooperate and the other player also plays cooperate, you both get a medium reward. If you play defect and the other player also plays defect, you both get a small reward. If you play defect and the other player plays cooperate, you get a large reward and they get no reward. Inversely, if you play cooperate and the other player plays defect, you get no reward and they get a large reward.

One may note that defection always results in a better payoff than cooperation, regardless of the other player's choice.[19, 21, 23, 24] Because defecting offers a greater reward than cooperating, purely rational self-interested players will defect, meaning the only possible outcome for two purely rational players is for them to both defect, even though mutual cooperation would give a greater reward.[19] Choosing to defect is each player's best response in all circumstances, thus it is the dominant strategy.[19] This situation entails that mutual defection is the only strong Nash equilibrium in the prisoner's dilemma, meaning neither player has anything to gain from changing their own strategy.[12, 23, 24] The Nash equilibrium is important because it informs what actions two rational players would take. It defines the optimal way to play against a rational opponent. Thus, the dilemma is that mutual cooperation gives a better reward than mutual defection but is not the rational outcome because the choice to cooperate, from a self-interested perspective, is irrational.[19, 20, 21, 25]

The prisoner's dilemma is studied for its ability to model many real world scenarios that involve trust and cooperation, particularly in economics, psychology, sociology, and politics.[10, 11, 20, 21, 22, 24, 25, 27] These perspectives on the prisoner's dilemma note that a strategy has to consider other players' actions and reactions to be successful.[20, 21]

Iterated Prisoner's Dilemma

The Iterated Prisoner's Dilemma is a turn-based game played between two players.[1, 3, 24] Each turn, each player chooses one of two possible actions: cooperate or defect. Each player then receives a payoff for that turn based on the combination of what they played and what their opponent played. Each player must lock in their action for the turn before their opponent's action is revealed and payoffs are distributed. Figure 1 gives a representation of what payoffs satisfy the conditions to meet the rules of IPD.[2] Reading from left to right, the row player in the matrix gets

the first value in a box, and the column player gets the second value in a box. For example, if the row player chooses cooperate and the column player chooses defect, the row player gets a payoff of D and the column player gets a payoff of A. If instead both the row player and the column player choose cooperate, both the row player and the column player get a payoff of B. To qualify as an Iterated Prisoner's Dilemma game, the value of the payoffs must be assigned such that $A > B > C > D$, as the variables correspond to the matrix in Figure 1. Additionally, $2B > (D + A)$. Figure 2 shows a common example IPD payoff matrix with values instead of variables. The total payoff for a player is found by taking the sum of their payoffs for each turn.

The significant difference between the regular prisoner's dilemma and the iterated prisoner's dilemma is that in the IPD both players remember previous actions and adjust their strategy accordingly.[2] Trust can be established or broken based on previous actions, allowing for mutual cooperation to exist. In the iterated prisoner's dilemma, optimal strategy involves attempting to achieve mutual cooperation with the other player and only defecting in retaliation.[1, 2, 24] This is a more nuanced strategy than always playing defect in the regular prisoner's dilemma. The specifics of when to attempt cooperation and when to retaliate are what differentiate the various advanced IPD strategies. One exception to this strategy is when the other player does not retaliate against your defection, such as an always cooperate strategy. In that situation, there is no threat, so the optimal strategy returns to always defect.[1, 2]

	Cooperate	Defect
Cooperate	B, B	D, A
Defect	A, D	C, C

Figure 1: The payoff matrix for Iterated Prisoner's Dilemma represented with variables, where A > B > C > D.

	Cooperate	Defect
Cooperate	3, 3	0, 5
Defect	5, 0	1, 1

Figure 2: A common example payoff matrix for Iterated Prisoner's Dilemma represented with numerical values.

Common Iterated Prisoner's Dilemma Strategy

Analysis of a single turn of the IPD (i.e. the regular non-iterated Prisoner's Dilemma) finds the Nash Equilibrium to occur when both players choose defect.[12, 19, 23, 24] This is because, when looking at a single turn, defect will give a greater payoff regardless of what the opponent plays. To illustrate, given the opponent plays cooperate and you play cooperate, you get a payoff of 3. Given the opponent plays cooperate and you play defect, you get a payoff of 5. If instead the opponent plays defect and you play cooperate, your payoff is 0. Given the opponent plays defect and you play defect, your payoff is 1. If we have the goal of maximizing our payoff in a single turn of the IPD, it is clear we should play defect.

However, this strategy does not scale well over more than one turn against rational opponents. Consider the case that you and your opponent play a 5 turn IPD and you and your opponent both choose to play defect each turn because it is the Nash Equilibrium. You each end with a total payoff of 5. If instead you and your opponent both chose to play cooperate every turn, both of your total payoffs would be 15. Given the goal is to get the greatest total payoff, we can see playing defect each turn is not necessarily optimal.[1]

Improving one step past the always defect strategy, we get reactionary strategies.[1] Reactionary strategies take in their opponent's last action(s) and use that information to decide their action. One popular and effective reactionary strategy is called tit-for-tat (TFT).[1, 2] TFT plays whatever the opponent played last round. On the first round, TFT plays cooperate. This strategy leads to mutual cooperation with opponents who are willing to cooperate, while also defensively defecting against opponents who play defect. Reactionary strategies like TFT score better than prescribed strategies, such as always defect, because reactionary strategies can adapt

to their opponent's playstyle.[2] A limitation of TFT is that it is unable to exploit opponents who always play cooperate.[2]

The most advanced strategies for the Iterated Prisoner's Dilemma come from reinforcement learning agents.[1] There are a myriad of different techniques to perform reinforcement learning. The underlying principle of reinforcement learning is to train the agent by having it repeatedly perform a task. The agent is rewarded for doing the task well, and not rewarded for doing the task poorly. The agent is built with the goal of maximizing its reward. The underlying algorithm that governs the agent's decision making process updates itself in response to the reward it was given. After many iterations of training, the agent learns to do the task well. The methods of reinforcement learning are somewhat intuitive because they mirror how humans learn. In the context of the Iterated Prisoner's Dilemma, the task the agent is performing is choosing cooperate or defect each round. Doing the task well would correspond to getting a high total payoff from the game (and thus a high payoff per turn as well).

However, an agent could be trained with a goal that is different from maximizing its own total payoff. An agent could be trained to instead maximize the difference between its score and its opponents score. The agent would then be rewarded based on how large that difference was, regardless of the value of the agent's score. This approach is similar to zero-determinant (ZD) strategies used in IPD.[26] ZD strategies lead to both players having lower total payoffs, which makes ZD strategies not as effective as other reinforcement learning strategies for the IPD.[26] Maximizing payoff difference is a strategy that is more applicable to zero-sum games.

Conversely, an agent could be trained to maximize the sum of its total payoff and its opponent's total payoff. The agent would then be rewarded based on the value of the sum, regardless of the value of the agent's score. This goal is less interesting mechanically because the

behavior of the agent converges to always play cooperate. However, the motivation is compelling as it mirrors why economic trade and teamwork lead to better overall results than economic isolation and strict competition.[19, 22]

Reinforcement Learning in the Iterated Prisoner's Dilemma

There are several behaviors that reinforcement learning agents exhibit in the IPD that provide insight into reinforcement learning as a technique. The first of which is behavior in the first round of the IPD. In the first round, an agent has no information about its opponent to use in deciding whether to cooperate or defect. The decision must be made based on the agent's prior training. This is why reactionary strategies such as TFT must prescribe their action on the first round as part of their definition.[19] When reinforcement learning agents are given the goal of maximizing their individual payoff and are trained against a collection of opponents, the agents strongly tend to cooperate in the first round.[1] This is because several of the training opponents would harshly retaliate against defection. Therefore, the agents' training determined that attempting to establish mutual cooperation on the first round was worth the risk of getting exploited by an opponent who plays defect on the first round. However, this trained behavior of cooperating on the first round would likely be different if the collection of training opponents was more aggressive and prone to defection. In the IPD, an agent's action in the first round plays a large role in the overall outcome of the game. A first round defection correlates with lower total payoffs on average for both the agent and the opponent if the opponent was willing to cooperate.[23]

It is well known that the quality and properties of training data will affect the function of an agent that trains with that data.[13, 15, 16] Furthermore, when an agent trains in a setting with other agents, the behavior of the other agents will affect the results of the training agent.[13, 15, 16]

For example, consider an agent *A* training to play chess. If agent *A* trains primarily against aggressive opponents, agent *A* will likely perform poorly against defensive opponents. However, if an agent *B* is trained against a wide range of aggressive and defensive opponents, agent *B* may perform better overall than agent *A*, but agent *B* could perform worse than agent *A* when facing aggressive opponents. Therefore, reinforcement learning agents should be trained with other agents that are as representative of real use case agents as possible. Consider another example where the reinforcement learning agent is piloting a self-driving car. If that agent is trained in simulation with other cars that perfectly obey traffic laws, the agent will be ill-prepared to handle more difficult situations caused by errors from other cars.

Taking from the IPD, reinforcement learning agents should also be trained with the notion that first impressions of another agent can greatly affect the outcome of the interaction. In a competitive setting such as the chess example, early moves can give away an agent's playstyle, similar to how defecting in the first round strongly correlates to further defection in the IPD.[1] Agents should be trained to take an initial action that signals their intent to other agents in cooperative settings, or hides their intent in competitive settings. In settings where agents interact with humans, care should be taken to avoid unwanted stereotyping of humans based on initial interactions.[20]

Finite State Machines

Finite state machines (FSMs) are a type of agent used for machine learning tasks. As an agent, finite state machines hold a policy that guides how the agent will interact with its environment. Finite state machines represent that policy through a collection of states and transitions.[15, 16, 17, 18]

A state in a finite state machine is a representation of the current position in the execution of its policy. A finite state machine can have many or few states, but must always have at least one state. A finite state machine can only exist in one state at a time. Finite state machines can be constructed in a way such that each state ascribes a particular action, and the finite state machine will perform that action when in that state.

A transition in a finite state machine assigns a change from one state to another based on an observation in the finite state machine's environment. The observation acts as a trigger for the finite state machine to take the transition from its current state to the transition's destination state. Transitions are one-directional and only between exactly two states. If a bi-directional behavior is wanted, it requires two transitions. Finite state machines can be defined such that a specific action is performed when a transition is taken, based on the observation in the environment.

Finite state machines are commonly used because they are easily human readable.[15, 18] Other machine learning agents can often be cumbersome to trace action sequences or decipher why an action was chosen, particularly if the agent was trained for a complex task. Finite state machines are comparably simple to trace action sequences and understand what observations and states led to those actions.[15] Finite state machines are also relatively easy to represent in a diagram form, even when trained for a complex policy.[16, 17] These characteristics make finite state machines a good choice when a clear and thorough understanding of an agent's trained behavior is required.

Finite state machines are the agent of choice for this paper because they give an unambiguous representation of if a state is or is not a sink state.

Sink States

With regards to a machine learning agent, a sink state is a state in which each possible transition from that state returns the agent back to the same state.[14] State refers to an agent's current collection of beliefs about itself and its environment. A state is deemed a sink state because the agent is stuck in that state once the agent transitions to the state. A sink state can exist if the physical characteristics of the agent's environment allow entry but not exit, such as navigating into a pit. Alternatively, a sink state can be a state in which the agent's decision making policy and reward structure tell the agent to stay in a state indefinitely, regardless of any possible environmental observations. This second interpretation of sink states is the most applicable to the iterated prisoner's dilemma.

A behavior reinforcement learning agents exhibit in the IPD is creating internal sink states.[1] In the context of the IPD, a sink state refers to a state that the agent will not leave, regardless of what actions the opponent and the agent take. For instance, a finite state machine (FSM) chooses its action based on its current state and the opponent's last action.[15] The FSM then also transitions to its next state (which could be back to the state it is already at) based on the opponent's action. A FSM agent can create a sink state if a state in the FSM transitions back to itself both when the opponent cooperates and when the opponent defects. Figure 3 shows a FSM that was trained using evolutionary algorithms to maximize its own score in standard IPD tournaments[1]. This FSM agent's training has resulted in state 4 becoming a sink state. Figure 4 shows state 4 of the FSM. If the agent is in state 4 and the opponent last played defect, the agent will play defect and transition back to state 4. If the agent is in state 4 and the opponent last played cooperate, the agent will play defect and transition back to state 4. Thus, once the agent reaches state 4, it will never leave state 4, regardless of how many rounds of the IPD are played

It is important to note that a FSM agent's actions are not relevant for defining a sink state. The FSM agent could play any combination of cooperate and defect, as long as every action transitions back to the previous state. Sink states can also be more loosely viewed as a small collection of states that transition back to each other and do not leave their collection.

Different reinforcement learning techniques will be more or less susceptible to training internal sink states. FSMs are highly susceptible to training internal sink states compared to other techniques.[15, 18] Some reinforcement learning techniques do not explicitly have internal states, but a pseudo-internal state could be defined. For example, standard neural networks do not keep an explicit internal state, but one could argue the value of the weights of the neuron-edge connections represent an internal state.[1] However, recurrent neural networks (RNNs) explicitly have internal states.[1] Hidden Markov Models (HMMs) also explicitly have internal states, but because of their probabilistic nature, HMMs are much less susceptible to training internal sink states.[1]

In the IPD, there exist many hand-crafted strategies that always defect after the opponent has defected some number of times.[1, 2, 23] This represents a kind of always defect sink state. Hand-crafted strategies are relevant because they are a deliberate, simple, and effective human designed approach to playing the IPD. The existence of these hand-crafted strategies shows that sink states can be desirable.

In general, when training reinforcement learning agents, sink states can be dangerous because an agent in a sink state is no longer updating based on its environment. Consider the previously discussed FSM agent. If this agent reaches state 4, it will never be able to reach mutual cooperation with its opponent. It is likely that the agent will be stuck in the undesirable mutual defection for the rest of the game. Sink states inhibit reinforcement learning agents from

reacting dynamically to their environment, which will likely reduce their performance.[14, 16] Researchers developing agents of any kind should be aware of sink states because they have the ability to drastically alter the behavior of the agent, for better or worse.

Evolutionary Learning

Evolutionary learning is a population based trial and error approach to training a machine learning agent that is inspired by evolutionary biology.[28, 29] Evolutionary learning involves repeating a cycle of evolution steps until a stopping criterion is reached.[29] First, an initial population of agents is created with randomly assigned specifications for their application, such as random weights for performing actions and taking transitions. The initial population of agents is evaluated and each agent is assigned a fitness score, which numerically represents the effectiveness of the agent.

The cycle begins with reproduction, involving either crossover, where the traits of two agents in the population are mixed to create a new child agent, or cloning, where one agent in the population is copied to create a new child agent. With either crossover or cloning, the new child agent has a chance of mutation, which randomly alters some of its traits. The child agents are then evaluated, assigned a fitness score, and added to the population with the parent agents. The population is then sorted by fitness score and a proportion of the agents with the lowest fitness scores are removed. The cycle then repeats with another step of reproduction. The cycle stops once a certain criterion is reached, such as a defined fitness score or number of iterations. The agent with the highest fitness score at the end of training is generally taken as the final result.

Evolutionary learning performs well at approximating a broad range of problems.[29] However, evolutionary learning can be prone to getting stuck in local optima.[28]

Tragedy of the Commons

The tragedy of the commons is a situation where one or a few individuals harm the collective through the over-use of a common resource.[6, 7, 8, 25, 30] A common example of the tragedy of the commons is a single farmer overgrazing livestock on common pasture, causing the pasture to be destroyed, depleting an otherwise renewable resource of fertile land.

In relation to environmental science, the tragedy of the commons is often mentioned in relation to sustainable development.[7] In relation to the prisoner's dilemma, the tragedy of the commons shows that decisions made under collective rationality may not necessarily be the same as those made under individual rationality.[7, 8] This mirrors how in the prisoner's dilemma mutual cooperation would yield better results than mutual defection, but defecting is the only rational choice for an individual.[19]

Modern analysis of the tragedy of the commons finds that regulation and societal norms work to mitigate the issue.[7] Despite this, the tragedy of the commons still finds applications in modern problems, such as climate change, waste pollution, antibiotic resistance, herd immunity, and digital pollution.[30]

Scientific Contribution

The scientific contribution of this paper involves producing data and experimentation towards answering questions regarding sink states: What causes machine learning agents to train sink states? Is the impact of sink states variable with different training methods? These questions are addressed through the lens of finite state machines trained to play the iterated prisoner's dilemma. Thus, scientific work in those domains is reinforced. A comparison of reinforcement learning and evolutionary learning in this domain is provided. Explicit analysis of these questions can be found in the discussion section of this paper.

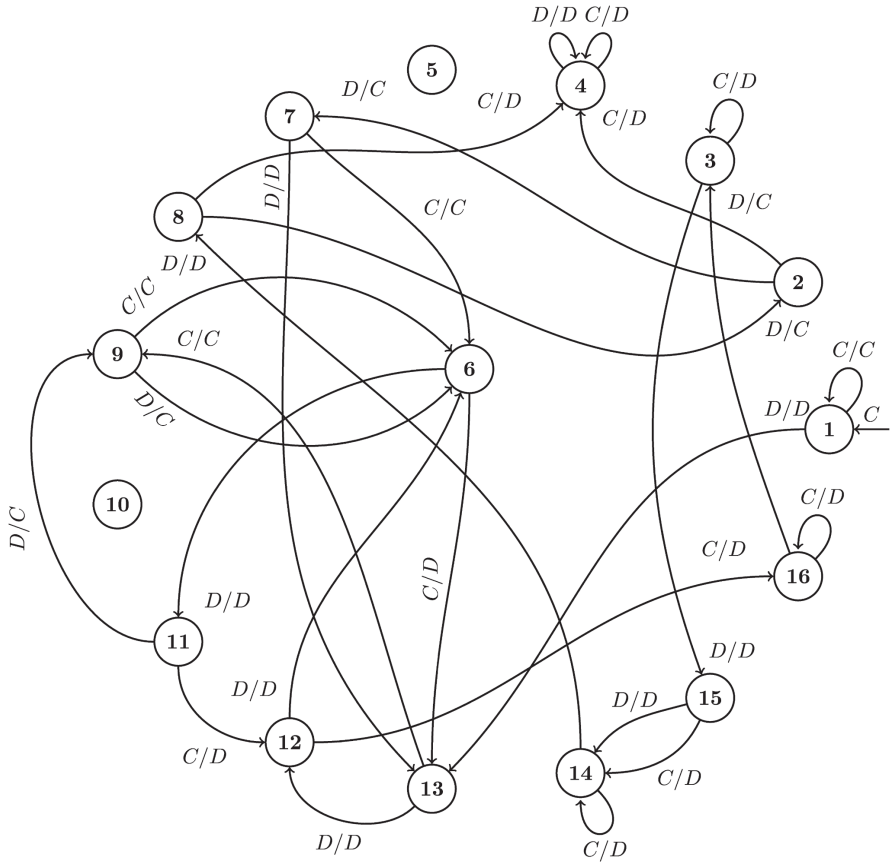

Figure 3: A finite state machine trained using evolutionary algorithms to maximize its own score in standard IPD tournaments[1].

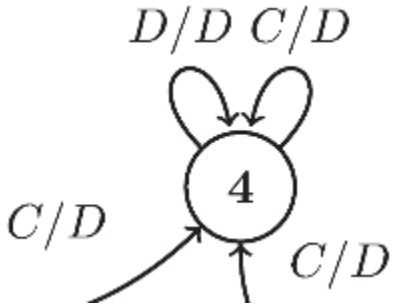

Figure 4: State 4 from the finite state machine shown in Figure 3[1].

CHAPTER 2

METHODS

Evolutionary Learning Implementation

The model being trained is a finite state machine (FSM). The FSM is a collection of states and transitions. The number of states has been set to 18, per evidence from related work[1] and experimental testing. DIfferent FSMs were tested with numbers of states ranging from 2 to 25. FSMs with a low number of states were found to perform less consistently, while any more states beyond 18 were found to have no positive impact on consistency or fitness score but increase training time. There are exactly two transitions from each state, but there can be as few as 0 or as many as every transition to a particular state. One transition from a state corresponds to the defect action in the prisoner's dilemma, the other corresponds to cooperate. Based on the action chosen by the opponent, the corresponding transition is taken by the FSM to reach the next state. A transition can go back to the same state it started from, in a loop pattern. If both transitions from a particular state return back to the same state, that is considered a sink state. Each transition also specifies what action the FSM will play next, either cooperate or defect.

A FSM will always start the prisoner's dilemma game from the same state, which could be considered the FSM's initial state. A FSM's action on the first turn is randomly chosen when the FSM is created, with equal probability between cooperate and defect. The FSMs are generated with random transitions, such that the destination state of the transition is assigned randomly and the action of the FSM when taking that transition is assigned randomly. However, the transitions do not change after generation. You may refer to Figure 3 for an example of a complete FSM, and you may refer to Figure 4 for an example of a single state with its transitions.

The FSMs are trained by an evolutionary learning approach, which may also be referred to as a genetic algorithm. The process begins by generating an initial population of 50 FSMs, which have random transitions. Those FSMs are then evaluated by playing a 50 round prisoner's dilemma game against an opponent who defects a predefined percentage of the time. The FSM's score in the prisoner's dilemma game is used to sort the FSMs in order of performance. This step will be relevant in subsequent iterations. Next, a child is created from each member in the population. Each child is a copy of their parent FSM, but there is a 25% chance per state that a mutation could occur in the child FSM. If a mutation occurs, there is an even chance that either a transition destination changes or a transition action changes. The child FSMs are evaluated in the same way as the parents, by playing a 50 round prisoner's dilemma game against an opponent who defects a predefined percentage of the time. The entire population (now 100 FSMs total) is sorted again based on their scores, but now the bottom 50 FSMs are removed from the population. The process of generating children, evaluating, sorting, and removing the worse performers is then repeated for 500 iterations (or stopped early if there have been 50 consecutive iterations where the top performing FSM has not changed). The entire 500 iteration process is repeated for different opponents (different percentage chance to defect).

Reinforcement Learning Implementation

The reinforcement learning approach also trains a finite state machine (FSM). The FSM is a collection of states and transitions. Testing for the reinforcement learning finite state machine (RLFSM) found no meaningful difference in consistency or fitness score for a number of states between 5 and 25. The RLFSM was chosen to also have 18 states to limit the number of differing factors between the RLFSM and the evolutionary learning FSM for the purpose of

comparison. The RLFSM also keeps a fitness score that is evaluated the same as the evolutionary learning FSMs for comparison.

The RLFSM assigns a weight to each possible action at a state (cooperate and defect), and assigns a weight to each possible transition. The weight value is calculated as the sum of reward earned by the RLFSM in the next two turns after the action or transition was selected, divided by the number of times that particular action or transition was selected. Each state has a weight for choosing cooperate and a weight for choosing defect. Each state has a weight for transitioning to each other state based on if the opponent played cooperate or defect last round. The total number of transition weights for a given state is 18 cooperate transition weights plus 18 defect transition weights for a total of 36 transition weights per state. All weights are initialized to 1.

The weights consider the next two turns as a way to account for the future impact of playing an action affecting the opponent's decision making. This formulation was found to significantly increase the fitness score of the RLFSM against a tit-for-tat opponent. During training, the actions and transitions played by the RLFSM are chosen stochastically, with each possible action or transition having a probability proportional to its weight. The weights are updated after each turn during training. Once training is finished, the RLFSM is made deterministic by selecting the action and transitions at each state with the highest weight. The deterministic RLFSM has one action and two transitions per state, one transition if the opponent plays cooperate, the other transition if the opponent plays defect. When training against a tit-for-tat opponent, the weight to play cooperate settled at 2.80 while the weight to play defect settled at 1.90 for all states. The RLFSM was trained for 500,000 iterations of 50 round IPD games, for a total of 25,000,000 rounds. This number of iterations was found to be sufficient for

the weights to converge to a settled value. After being made deterministic, the RLFSM was evaluated against the same opponent it trained against to gather the RLFSM's fitness score and sink state properties. The same fitness score and sink state definitions from evolutionary learning FSMs are used for RLFSM.

CHAPTER 3

RESULTS

Evolutionary Learning Implementation

Data was collected only from the best performing (highest fitness score) ELFSM in its population after training. 1000 unique populations of ELFSMs were trained for each opponent percent chance to defect, as shown on the graphs.

The average number of sink states trained by ELFSMs per the opponent's percent chance to defect is graphed in Figure 5. Training a sink state is most likely when the opponent's chance to defect is approximately 65%. There is a significant reduction in the amount of sink state training when the opponent is highly likely to defect or highly likely to cooperate.

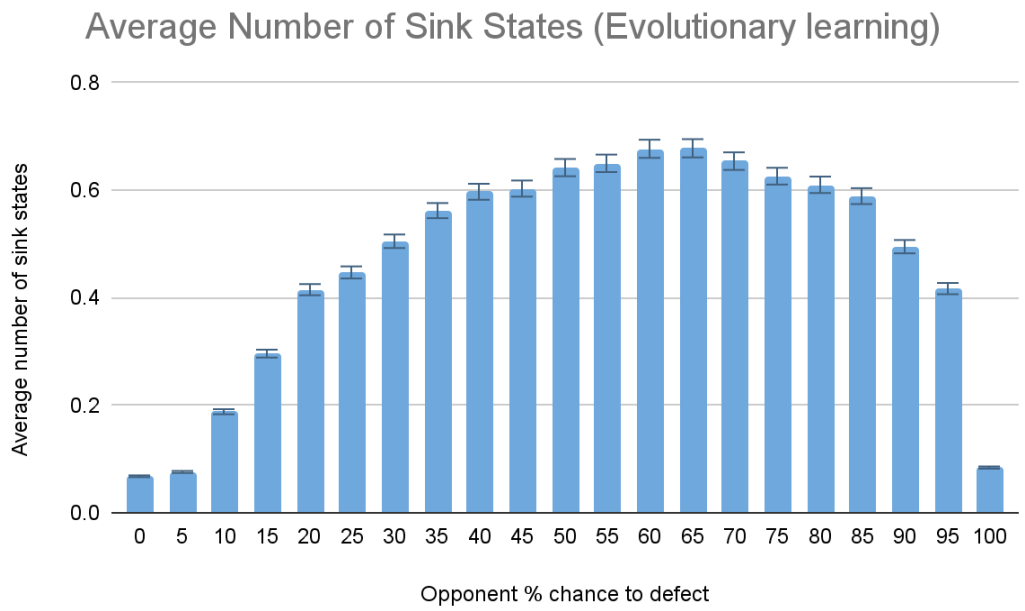

Figure 5: Average number of sink states trained by evolutionary learning FSMs per the opponent's percent chance to defect after 1000 iterations.

A comparison of the average fitness score between ELFSMs that trained sink states vs. ELFSMs that did not train sink states is graphed in Figure 6. The fitness scores for ELFSMs that trained sink states vs those that did not train sink states are nearly identical at all levels of opponent defection. However, there is a consistent but small increase in the fitness of ELFSMs with sink states over those without, which is about 1 fitness score point. At the extremes of 0 and 100 percent chance to defect, the ELFSMs were able to achieve a perfectly optimal score given the opponent's action. The trained ELFSMs achieved the optimal score by playing defect every round.

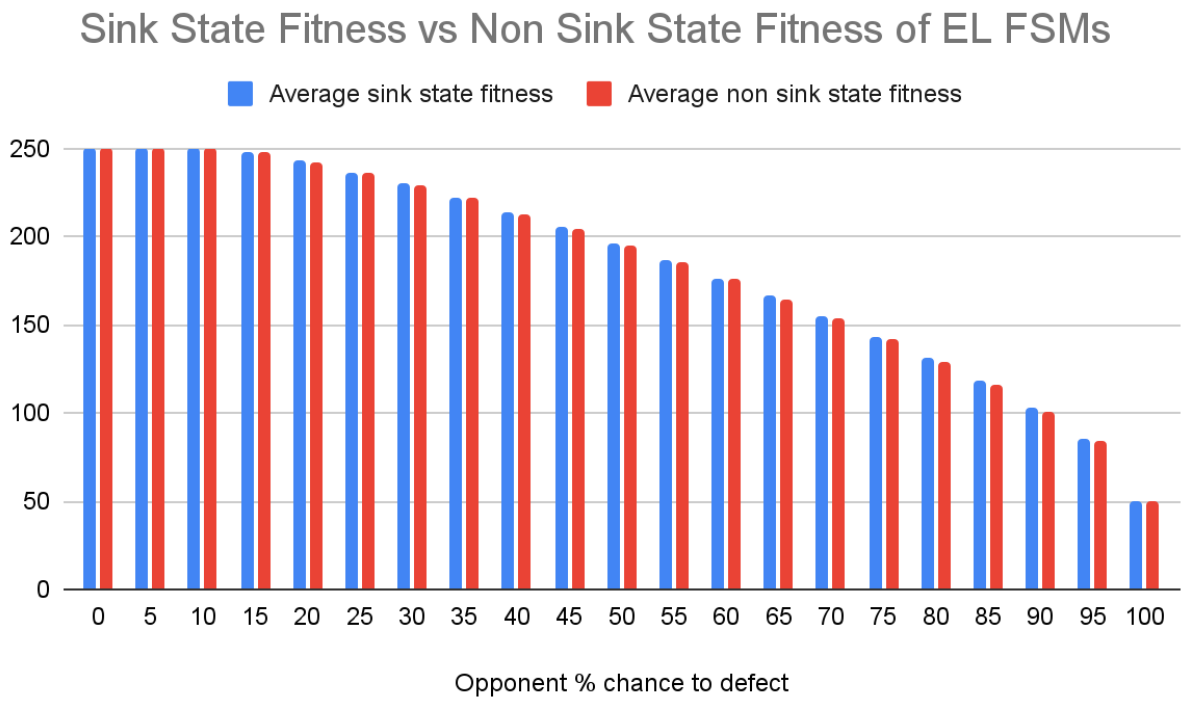

Figure 6: Average fitness score between evolutionary learning FSMs that trained sink states vs. evolutionary learning FSMs that did not train sink states.

The number of sink states trained by the highest fitness score ELFSMs of their population are graphed in Figure 7. It was by far most common for the ELFSMs to train either zero or one sink state. When the opponent's percent chance to defect was between 35% and 85%, it was most likely to train one sink state. Otherwise, it was most likely to train zero sink states. None of the trained ELFSMs had more than three sink states.

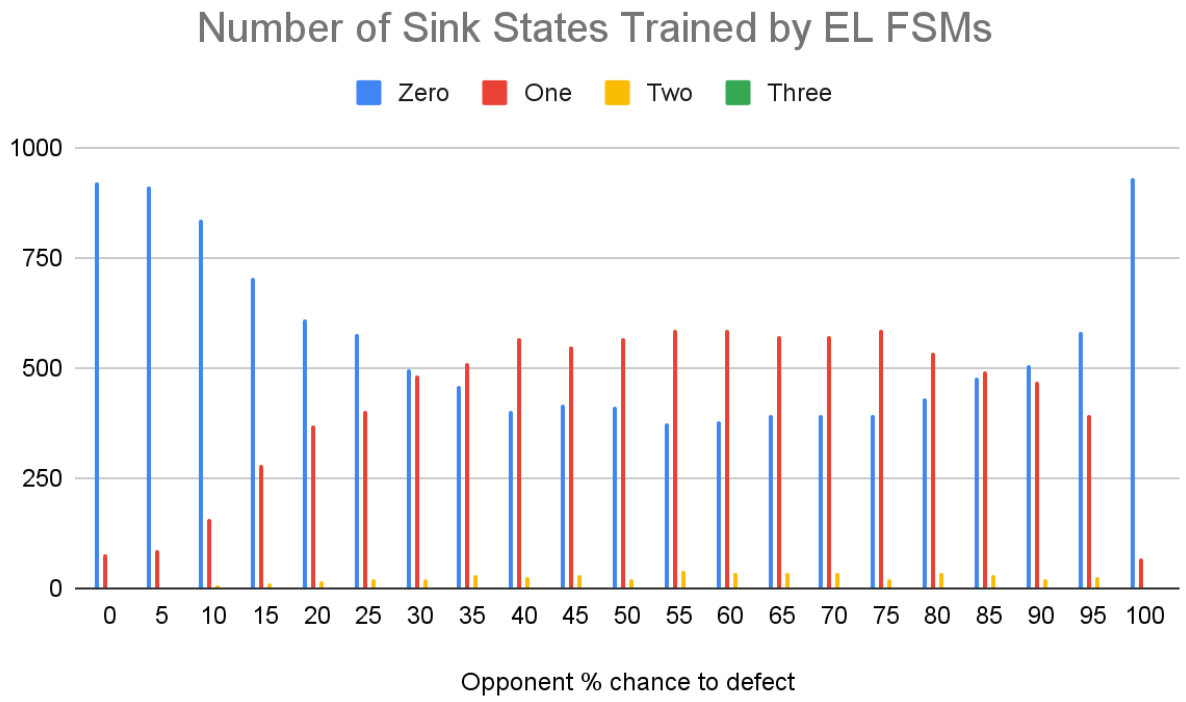

Figure 7: Number of sink states trained by evolutionary learning FSMs.

Data collected from training ELFSMs against a tit-for-tat strategy is listed in Table 1. When the ELFSMs were trained against an opponent playing the tit-for-tat strategy, the best ELFSM of the population always achieved a fitness score of 150, regardless of if one or more sink states were trained. This was due to the best ELFSM playing cooperate every round vs. the tit-for-tat opponent.

Table 1: Evolutionary learning FSMs trained against a tit-for-tat strategy.

Tit-for-Tat	
Average number of sink states	0.084
Standard Deviation	0.2982
Variance	0.0889
Standard Error of the mean	0.0094
Average sink state fitness	150
Average non sink state fitness	150
Number of FSMs without sink states	921
Number of FSMs with one sink state	75
Number of FSMs with two sink states	3
Number of FSMs with three sink states	1

The number of sink states trained by ELFSMs playing against a tit-for-tat strategy is graphed in Figure 8. Relatively few of the ELFSMs that played against tit-for-tat trained sink states. The sink state formation rate most closely resembled an opponent with a 0 percent chance to defect. This result correlates with the fact that the tit-for-tat opponent did play cooperate every round. Given that evolutionarily trained FSMs playing against tit-for-tat are adverse to training sink states, the results suggest training sink states is unfavorable when playing the IPD against cooperative opponents.

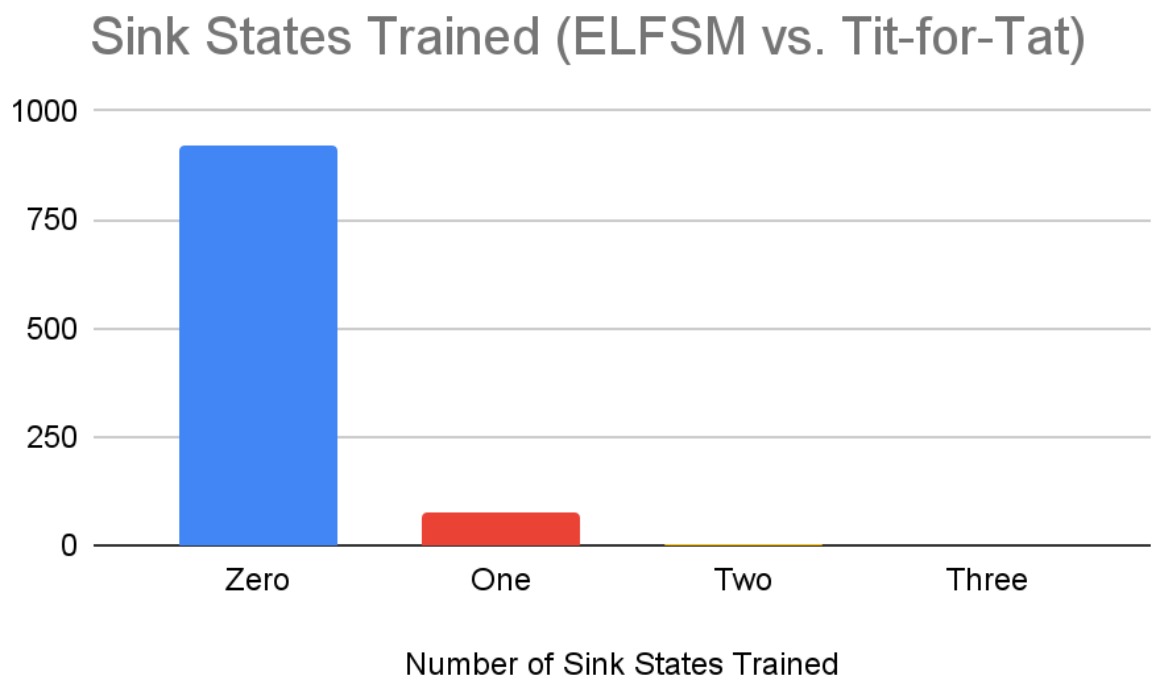

Figure 8: Number of sink states trained by evolutionary learning FSMs playing against a tit-for-tat strategy.

Reinforcement Learning Implementation

Data was collected after the RLFSMs were made deterministic. 1000 unique RLFSMs were trained for each opponent percent chance to defect, as shown on the graphs.

The average number of sink states trained by RLFSMs per the opponent's percent chance to defect is graphed in Figure 9. Training a sink state is most likely when the opponent's chance to defect is 100%. There is a slight increase in the amount of sink state training when the opponent is highly likely to cooperate. This data trend is nearly opposite of the ELFSM sink state formation. Overall, RLFSMs are less likely to form sink states than ELFSMs.

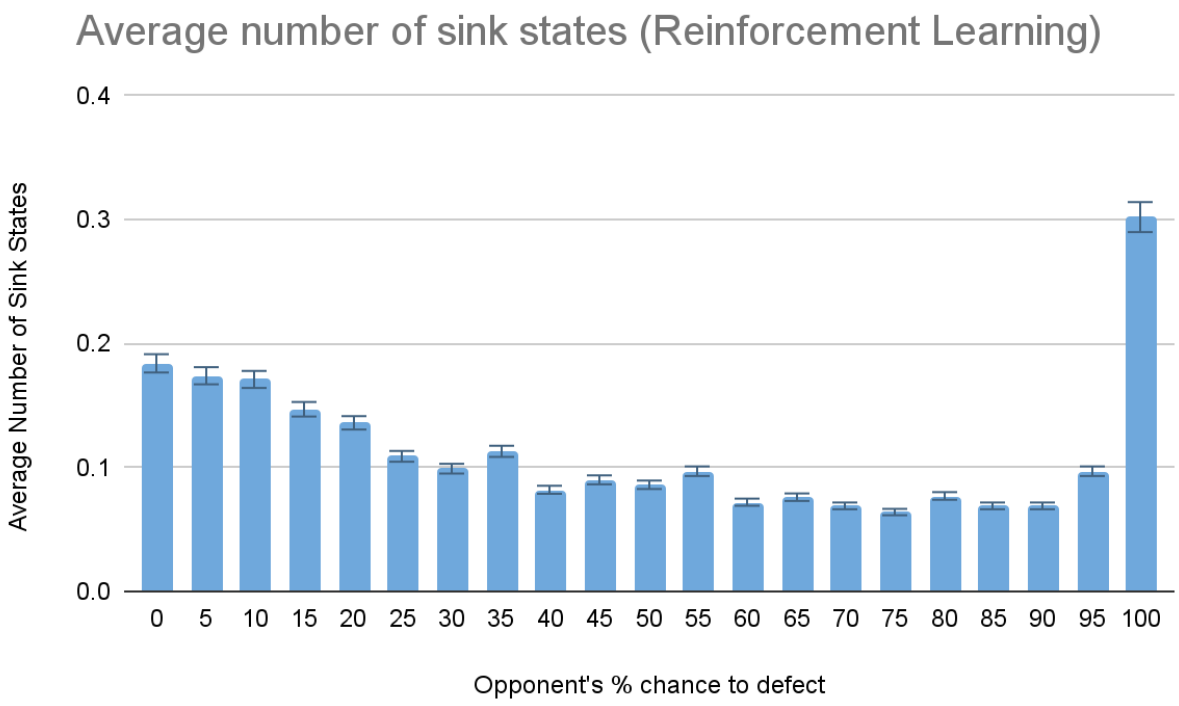

Figure 9: Average number of sink states trained by RLFSMs per the opponent's percent chance to defect after 1000 iterations.

A comparison of the average fitness score between RLFSMs that trained sink states vs. RLFSMs that did not train sink states is graphed in Figure 10. The fitness scores for FSMs that trained sink states vs those that did not train sink states are nearly identical at all levels of opponent defection. However, there is a consistent but small increase in the fitness of RLFSMs without sink states over those with sink states, which is an average of 1.3 fitness score points. This indicates the sink states on average detracted from the RLFSM's effectiveness. At the extremes of 0 and 100 percent chance to defect, most RLFSMs were able to achieve a perfectly optimal score given the opponent's action. The trained FSMs achieved the optimal score by playing defect every round.

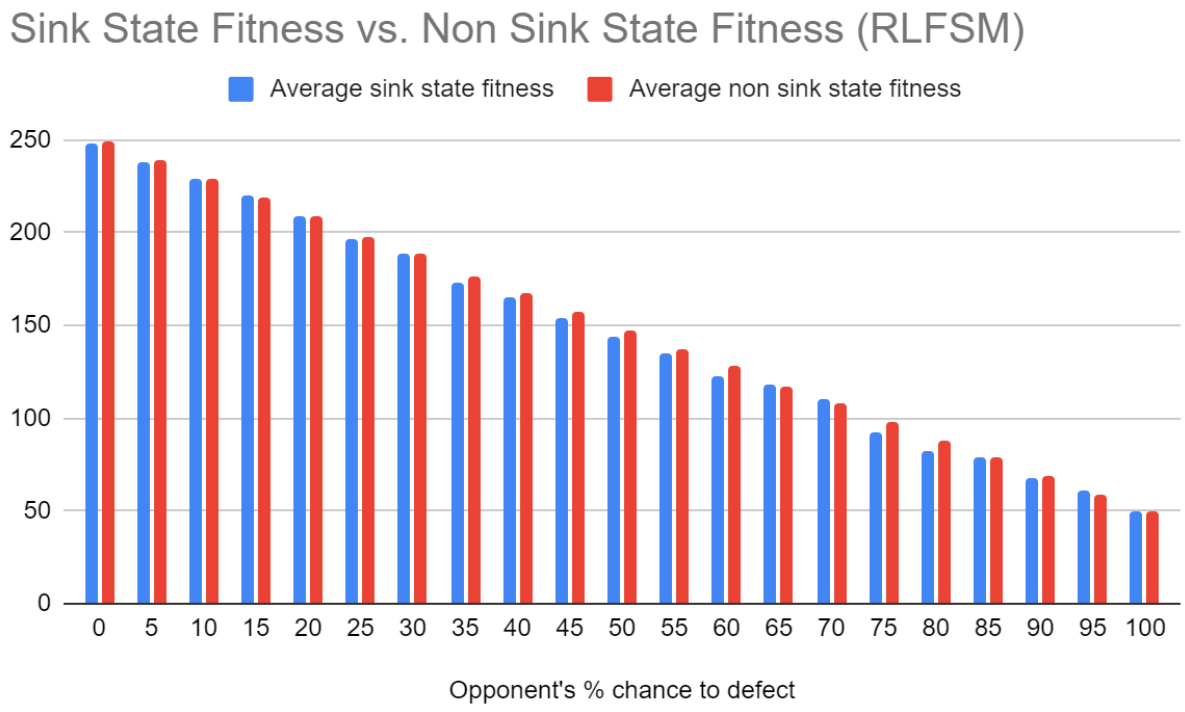

Figure 10: Average fitness score between RLFSMs that trained sink states vs. RLFSMs that did not train sink states.

The number of sink states trained by RLFSMs is graphed in Figure 11. It was by far most common for the FSMs to train zero sink states. When the opponent's percent chance to defect was 100% there was a dramatic rise in the chance to train one sink state. Otherwise, there is a slight increase in the chance to train one sink state close to 0% chance to defect. None of the trained RLFSMs had more than two sink states. RLFSMs almost never trained more than one sink state.

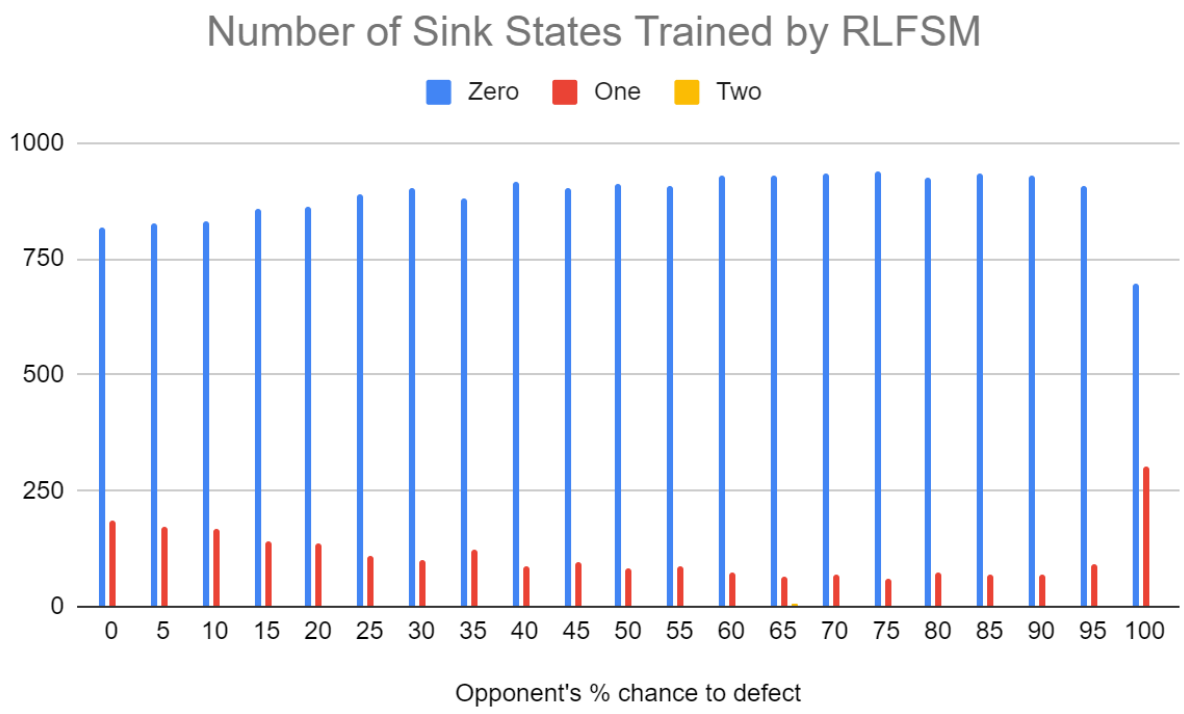

Figure 11: Number of sink states trained by RLFSMs.

Data collected from training RLFSMs against a tit-for-tat strategy is listed in Table 2. When the RLFSMs were trained against an opponent playing the tit-for-tat strategy, the trained deterministic RLFSM always achieved a fitness score of 150, regardless of if one or more sink states were trained. This was due to the RLFSM playing cooperate every round vs. the tit-for-tat opponent.

Table 2: RLFSMs trained against a tit-for-tat strategy.

Tit-for-Tat vs RL FSM	
Average number of sink states	0.056
Standard Deviation	0.2385
Variance	0.0569
Standard Error of the mean	0.0075
Average sink state fitness	147.81
Average non sink state fitness	149.66
Number of FSMs without sink states	946
Number of FSMs with one sink state	52
Number of FSMs with two sink states	2
Number of FSMs with three sink states	0

The number of sink states trained by RLFSMs playing against a tit-for-tat strategy is graphed in Figure 12. It was rare for the RLFSMs that played against tit-for-tat to train sink states. The sink state formation rate most closely resembled when RLFSMs played an opponent with a 65 percent chance to defect. Given that RLFSMs playing against tit-for-tat are adverse to training sink states, the results suggest training sink states is unfavorable when playing the IPD against cooperative opponents.

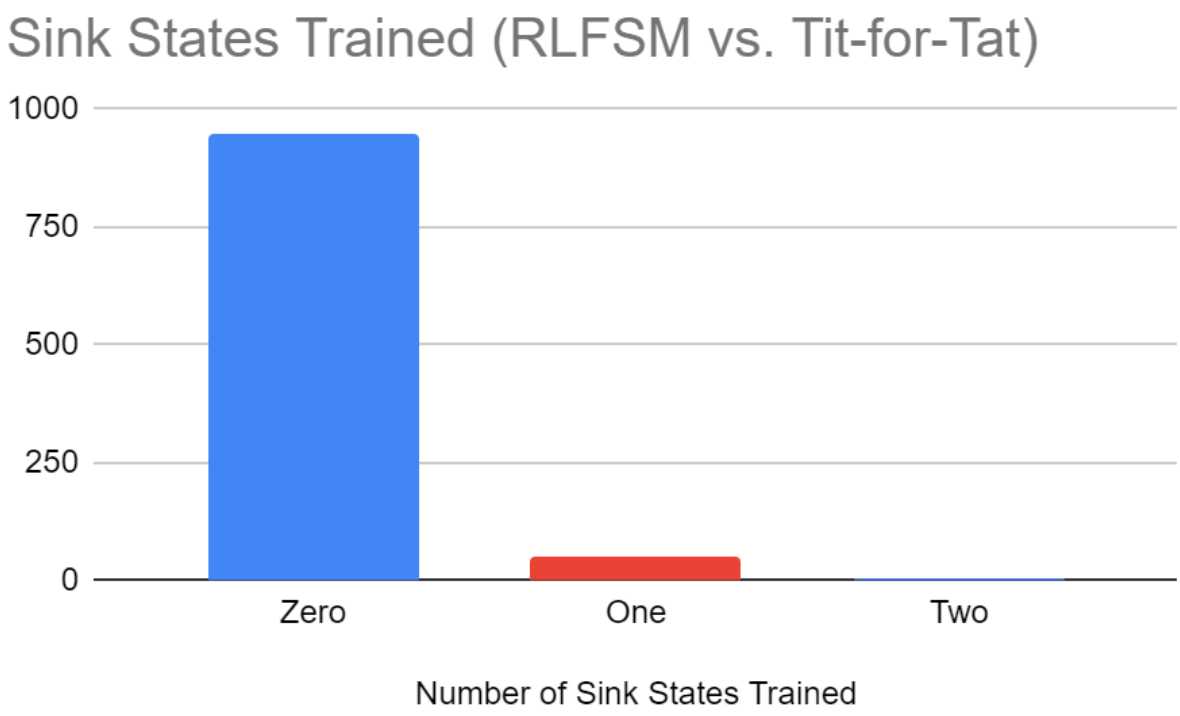

Figure 12: Number of sink states trained by RLFSMs playing against a tit-for-tat strategy.

CHAPTER 4

DISCUSSION

Game theory provides a useful tool for modeling real world phenomena in academic settings.[9, 10, 11, 12, 13] To that effect, the prisoner's dilemma is particularly applicable to economics and the tragedy of the commons. The design of the iterated prisoner's dilemma is such that the optimal strategy for maximum overall utility is mutual cooperation.[3, 5] This is clear from the utility payouts, with mutual cooperation being 3 + 3 = 6 utility while defection into cooperation yields 0 + 5 = 5 utility and mutual defection yields 1 + 1 = 2 utility. This formulation is supported by real world evidence that rational trade is mutually beneficial and leads to stronger economies by promoting specialization.[10, 11] Given this understanding, the behavior of the evolved FSMs gives a unique insight into other optimal behavior.[15, 16, 17, 18] When faced against an opponent who is willing to cooperate but will defensively defect, there is no individual incentive to ever defect. The lost utility from provoking the opponent to play defect outweighs any potential gain from defecting against cooperation. Mutual cooperation among rational opponents is an evolutionary stable strategy.[4] With rational opponents, the optimal strategy for maximum overall utility is also the optimal strategy for individual utility. However, this concept is disputed by the tragedy of the commons.

Korman and Vacus state the tragedy of the commons "aims to capture situations in public goods systems where self-interested individuals behave contrary to the common good by depleting or spoiling the shared resource."[6] In relation to the prisoner's dilemma, this reflects the relationship of defecting while others attempt cooperation. The tragedy of the commons provides a counter to the idea that rational agents will still achieve mutual cooperation when

motivated by self interest.[6, 7, 8] If the reward of mutual cooperation is too low or the punishment for mutual defection is too minor, the behavior of mutual cooperation will not arise. This can be seen by changing the utility payouts for the IPD. If maintaining the longevity of a shared public resource is the goal, regulations must be in place to ensure the utility of an individual's actions falls within the scope of the prisoner's dilemma definition.[2]

Harper et al. have shown that agents trained by reinforcement learning techniques exhibit behavior of mutual cooperation every round in the iterated prisoner's dilemma.[1] That behavior mirrors the behavior of evolutionary learning FSMs trained against tit-for-tat, and the results of this paper confirm that reinforcement learning techniques for FSMs in the IPD also train a low number of sink states. Sink states represent a lack of adaptation to opponent actions, which may be detrimental when playing the IPD against intelligent opponents.[14] However, intelligent IPD players trained by reinforcement learning or evolutionary learning always choose mutual cooperation when playing against other intelligent players, which is a static strategy. Thus, sink states appear to be most useful when the opponent's behavior is erratic. In evolutionarily trained FSMs, constant cooperation or constant defection leads to few sink states being trained. The evolutionary learning shows that sink states (and thus an inflexibility in thinking and response) are useful against chaotic and non-intentional opponents, but sink states are not useful against rational and intentional opponents. This is why evolutionary learning FSMs minimally trained sink states against 0% chance to defect and 100% chance to defect opponents; Those opponents are adhering to a non-chaotic behavior. Perhaps sink states in ELFSMs are a way to limit overanalyzing of an erratic opponent strategy, allowing a consistent response, which is likely to consistently play defect. In that case, training sink states is a reaction to erratic behavior that reintroduces some stability.

However, reinforcement learning FSMs do not follow the same patterns of evolutionary learning FSMs. The data gathered shows that RLFSMs are overall much less likely to train sink states than ELFSMs. Reinforcement learning FSMs do however train more sink states than evolutionary learning FSMs when the opponent always cooperates or always defects. The overall trends for RLFSM and ELFSM sink state formation are nearly inverse of each other. This study shows that patterns of sink state formation are highly dependent on the learning method used to train the agent. One common ground between RLFSMs and ELFSMs is that they both rarely trained sink states vs. a tit-for-tat opponent. This implies that training sink states vs. reactionary opponents is disadvantageous. Further study of sink state formation by other learning techniques would be insightful for better understanding these patterns.

To answer the questions posed by this paper about sink states: The method of sink state formation is variable dependent on the learning technique used. For evolutionary learning, a sink state is formed when initialization or mutation of an ELFSM randomly selects both transitions of a state to point back to itself. This could be done via two separate mutations, once for the opponent cooperate transition and once for the opponent defect transition. That ELFSM must also have a high enough fitness score to last in the population or at least pass the sink state on to a child ELFSM.

For reinforcement learning, a sink state is formed when the transition weights are greatest on the transition that returns a state to itself. This must be true for both the cooperate transitions and the defect transitions. For those weights to occur, the reward earned by the RLFSM after taking the transition must be greater than the reward earned after taking the other transitions on average.

By the nature of both of these learning methods, the sink states offer some utility or at least don't detract from the FSMs utility, otherwise they would be trained out. At the foundational level, both training methods rely on pseudo-randomness during training. Thus, although training is done via an intentionally defined method, some amount of sink state formation is left to chance.

For both evolutionary learning and reinforcement learning, the difference in fitness scores between FSMs with sink states and those without was relatively minor, approximately 1 to 2 fitness points on average. ELFSMs with sink states had higher fitness scores on average, while RLFSMs with sink states had lower fitness scores on average. These results indicate that the impact of sink states does vary based on the training method used, but not to a drastic degree. The same optimal end behavior of the trained FSMs could be generated with or without a sink state, limiting the effect of any sink states. For example, either reinforcement or evolutionary learning FSMs could train to always cooperate when playing against a tit-for-tat opponent, with or without a sink state. However, ELFSMs did train significantly more sink states on average than RLFSMs, showing that sink states were more beneficial to ELFSMs fitness during training. Altogether, evolutionary learning benefited more from sink states than reinforcement learning.

Future work studying sink state formation in another domain, especially one where it is hard to train optimal behavior, may yield more insight into the effects sink states have on machine learning agents. In an effort to combat the black box problem of artificial intelligence, all avenues of better understanding machine learning methods should be explored.[31, 32]

CHAPTER 5

CONCLUSION

This paper has investigated what behavior causes the formation of sink states in Finite State Machines (FSM) through the lens of the Iterated Prisoner's Dilemma (IPD) game. A FSM playing the IPD will decide to cooperate or defect based on its trained arrangement of states and transitions. If all of the transitions from a state return back to that same state, that state is a sink state. This paper finds that the likelihood of training a sink state is dependent on the method of training. Evolutionary learning methods were more prone to sink state formation than reinforcement learning methods. However, evolutionary learning and reinforcement learning produced finite state machines with similar scores in the iterated prisoner's dilemma. It is drastically unlikely that sink states will be formed when training against a rational opponent. Sink state formation is found to be correlated with erratic opponent behavior when using evolutionary learning techniques. Given the results gathered from these experiments, future work could seek to utilize sink states as a way of regulating responses to erratic situations, such as dynamically changing environments or unpredictable outside agents.

REFERENCES

[1] Harper M, Knight V, Jones M, Koutsovoulos G, Glynatsi NE, et al. (2017) Reinforcement learning produces dominant strategies for the Iterated Prisoner's Dilemma. PLOS ONE 12(12): e0188046. https://doi.org/10.1371/journal.pone.0188046

[2] Axelrod, Robert, and William D. Hamilton. "The Evolution of Cooperation." Science 211, no. 4489 (1981): 1390-396. Accessed March 1, 2021. http://www.jstor.org/stable/1685895.

[3] P. J. Darwen and Xin Yao, "Why more choices cause less cooperation in iterated prisoner's dilemma," *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, Seoul, Korea (South), 2001, pp. 987-994 vol. 2, doi: 10.1109/CEC.2001.934298. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=934298

[4] Alexander J. Stewart, Joshua B. Plotkin, "Prisoner's Dilemma: Extortion and cooperation", Proceedings of the National Academy of Sciences Jun 2012, 109 (26) 10134-10135; DOI: 10.1073/pnas.1208087109https://www.pnas.org/content/109/26/10134#F1

[5] Xiuqin Deng, Jiadi Deng, "A Study of Prisoner's Dilemma Game Model with Incomplete Information", *Mathematical Problems in Engineering*, vol. 2015, Article ID 452042, 10 pages, 2015. https://doi.org/10.1155/2015/452042

[6] Korman, A., Vacus, R. On the role of hypocrisy in escaping the tragedy of the commons. *Sci Rep* 11, 17585 (2021). https://doi.org/10.1038/s41598-021-97001-3

[7] Frischmann, Brett M., Alain Marciano, and Giovanni Battista Ramello. 2019. "Retrospectives: Tragedy of the Commons after 50 Years." *Journal of Economic Perspectives*, 33 (4): 211-28. https://doi.org/10.1257/jep.33.4.211

[8] Ostrom, Elinor. "tragedy of the commons." *The New Palgrave Dictionary of Economics. Second Edition.* Eds. Steven N. Durlauf and Lawrence E. Blume. Palgrave Macmillan, 2008. The New Palgrave Dictionary of Economics Online. Palgrave Macmillan. 23 June 2010 http://dx.doi.org/10.1057/9780230226203.1729

[9] Loginov, G. Cyclical behavior of evolutionary dynamics in coordination games with changing payoffs. *Int J Game Theory* 51, 1–27 (2022). https://doi.org/10.1007/s00182-021-00783-z

[10] David Carfì, Francesco Musolino, Game theory and speculation on government bonds, *Economic Modelling*, Volume 29, Issue 6, 2012, Pages 2417-2426, ISSN 0264-9993, https://doi.org/10.1016/j.econmod.2012.06.037.

[11] Friedman, D. On economic applications of evolutionary game theory. *J Evol Econ* 8, 15–43 (1998). https://doi.org/10.1007/s00191005005

[12] Chatterjee K., Doyen L. (2010) The Complexity of Partial-Observation Parity Games. In: Fermüller C.G., Voronkov A. (eds) *Logic for Programming, Artificial Intelligence, and Reasoning*. LPAR 2010. Lecture Notes in Computer Science, vol 6397. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-16242-8_1

[13] Shahaboddin Shamshirband, Ahmed Patel, Nor Badrul Anuar, Miss Laiha Mat Kiah, Ajith Abraham, Cooperative game theoretic approach using fuzzy Q-learning for detecting and preventing intrusions in wireless sensor networks, *Engineering Applications of Artificial Intelligence*, Volume 32, 2014, Pages 228-241, ISSN 0952-1976, https://doi.org/10.1016/j.engappai.2014.02.001.

[14] Wu A.S., Mathias H.D. (2020) Dynamic Response Thresholds: Heterogeneous Ranges Allow Specialization While Mitigating Convergence to Sink States. In: Dorigo M. et al. (eds) Swarm Intelligence. *ANTS* 2020. Lecture Notes in Computer Science, vol 12421. Springer, Cham. https://doi.org/10.1007/978-3-030-60376-2_9

[15] Carson Dunbar and Gang Qu. 2014. Designing Trusted Embedded Systems from Finite State Machines. *ACM Trans. Embed. Comput. Syst.* 13, 5s, Article 153 (November 2014), 20 pages. https://doi.org/10.1145/263855

[16] Oliva C., Lago-Fernández L.F. (2019) On the Interpretation of Recurrent Neural Networks as Finite State Machines. In: Tetko I., Kůrková V., Karpov P., Theis F. (eds) Artificial Neural Networks and Machine Learning – ICANN 2019: Theoretical Neural Computation. ICANN 2019. Lecture Notes in Computer Science, vol 11727. Springer, Cham. https://doi.org/10.1007/978-3-030-30487-4_25

[17] Walkinshaw, N., Taylor, R. & Derrick, J. Inferring extended finite state machine models from software executions. *Empir Software Eng* 21, 811–853 (2016). https://doi.org/10.1007/s10664-015-9367-7

[18] Spears W.M., Gordon D.F. (2000) Evolving Finite-State Machine Strategies for Protecting Resources. In: Raś Z.W., Ohsuga S. (eds) Foundations of Intelligent Systems. ISMIS 2000. Lecture Notes in Computer Science, vol 1932. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-39963-1_18

[19] Rapoport A. (1987) Prisoner's Dilemma. In: Palgrave Macmillan (eds) The New Palgrave Dictionary of Economics. Palgrave Macmillan, London. https://doi.org/10.1057/978-1-349-95121-5_1850-1

[20] Fehr, E., Fischbacher, U. The nature of human altruism. Nature 425, 785–791 (2003). https://doi.org/10.1038/nature02043

[21] Ahn, T., Ostrom, E. & Walker, J.M. Heterogeneous Preferences and Collective Action. Public Choice 117, 295–314 (2003). https://doi.org/10.1023/B:PUCH.0000003739.54365.fd

[22] Oosterbeek, H., Sloof, R. & van de Kuilen, G. Cultural Differences in Ultimatum Game Experiments: Evidence from a Meta-Analysis. Experimental Economics 7, 171–188 (2004). https://doi.org/10.1023/B:EXEC.0000026978.14316.74

[23] Axelrod, R. (1980). More Effective Choice in the Prisoner's Dilemma. Journal of Conflict Resolution, 24(3), 379–403. https://doi.org/10.1177/002200278002400301

[24] Dal Bó, Pedro, and Guillaume R. Fréchette. 2019. "Strategy Choice in the Infinitely Repeated Prisoner's Dilemma." American Economic Review, 109 (11): 3929-52. doi: https://doi.org/10.1257/aer.20181480

[25] Ostrom, E. (2015). Governing the Commons: The Evolution of Institutions for Collective Action (Canto Classics). Cambridge: Cambridge University Press. doi:https://doi.org/10.1017/CBO9781316423936

[26] Press, William H. and Dyson, Freeman J. 2012. Iterated Prisoner's Dilemma Contains Strategies that Dominate Any Evolutionary Opponent. Proceedings of the National Academy of Sciences. Vol. 109. No. 26. June 26, 2012. https://doi.org/10.1073/pnas.1206569109

[27] Deutsch, M. (1958). Trust and suspicion. Journal of Conflict Resolution, 2(4), 265–279. https://doi.org/10.1177/002200275800200401

[28] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), 2016, pp. 261-265 doi:https://doi.org/10.1109/ICGTSPICC.2016.7955308

[29] Harith Al-Sahaf, Ying Bi, Qi Chen, Andrew Lensen, Yi Mei, Yanan Sun, Binh Tran, Bing Xue & Mengjie Zhang (2019) A survey on evolutionary machine learning, Journal of the Royal Society of New Zealand, 49:2, 205-228, doi: https://doi.org/10.1080/03036758.2019.1609052

[30] V. Almeida, F. Filgueiras and F. Gaetani, "Digital Governance and the Tragedy of the Commons," in IEEE Internet Computing, vol. 24, no. 4, pp. 41-46, 1 July-Aug. 2020, doi: https://doi.org/10.1109/MIC.2020.2979639

[31] Alizadeh, F., Stevens, G. & Esau, M. (2021). I Don't Know, Is AI Also Used in Airbags?: An Empirical Study of Folk Concepts and People's Expectations of Current and Future Artificial Intelligence. i-com, 20(1), 3-17. https://doi.org/10.1515/icom-2021-0009

[32] Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S., & Yang, G. Z. (2019). XAI—Explainable artificial intelligence. Science Robotics, 4(37), eaay7120. doi: https://doi.org/10.1126/scirobotics.aay7120