Examination of Graphs in Multiple Agent Genetic Networks for Iterated Prisoner's Dilemma

Joseph Alexander Brown
School of Computer Science, University of Guelph
Guelph, Ontario, Canada, N1G 2W1
Email: jb03hf@gmail.com

***Abstract*—Multiple Agent Genetic Networks (MAGnet) are spatially structured evolutionary algorithms which move both evolving agents as well as instances of a problem about a combinatorial graph. Previous work has examined their use on the Iterated Prisoner's Dilemma, a well known non-zero sum game, in order for classification of agent types based on behaviours. Only a small complete graph was examined. In this study, a larger set of graphs with thirty-two nodes are examined. The graphs examined are: a cycle graph, two Peterson graphs with differing internal rings, a hypercube in five dimensions, and the complete graph. These graphs and properties are examined for a number of canonical agents, as well as a few interesting types which involve handshaking. It was found that the MAGnet system produces a similar classification as the smaller graph when the connectivity within the graph is high. Lower graph connectivity leads to a process by which disjoint subgraphs can be formed; this is based on the method of evolution causing a subpopulation collapse in which the number of problems on a node tends to zero and the node is removed.**

I. Introduction

The prisoner's dilemma is a commonly examined problem in game theory developed for the RAND corporation in the 1950s by Merrill Flood and Melvin Dresher. A motivating story is a police investigation of two suspects in a robbery case. The police have enough evidence to convict both of the suspects on a lesser charge of breaking and entering, and require at least one of the suspects to become a witness in order to convict the other in the robbery charge. Both are placed in separate interrogation rooms and are given the choice to either squeal on the other, which is a defection, or to keep silent, which is a cooperation. Both of the suspects are given this choice simultaneously and the jail time given to one will not cause an equal reduction in the jail time of the other, i.e. the game is non-zero sum. There are four outcomes: both squeal, the first suspect squeals on the second, the second squeals on the first, and neither squeal. These are associated with jail terms, or payoffs. This payoff matrix is found in Fig. 1. The temptation payoff (T) is received for defecting when there is a sucker (S) and cooperates. A cooperation payoff (C) is received when both cooperate, and a defection payout (D) when both defect. The preferences for the payouts are $S < D < C < T$, so being a sucker is worst outcome and suckering someone is the best result. For the iterated version, the iterated prisoner's dilemma (IPD), it is further required that alternatively being the sucker and suckering someone provides a payout of less than cooperating twice, i.e. $T + S < 2C$. A commonly used set of numerical utility scores associated which obey these properties is $T = 5, C = 3, D = 1$, and $S = 0$. Though changing these values has been shown to change the types of agents which are evolved [1].

This game is mathematically uninteresting when played only once; both players have a Nash equilibrium to defect if they play rationally. They both squeal to the police and receive the second to worst payoff. The dilemma comes from the obvious Pareto optimal solution of both staying quiet. The iterated game provides interesting behaviours, as agents are able to form cooperative networks. There becomes an incentive to work with another player so long as the game length is unknown[1]. The game has been studied extensively for its diverse uses in modeling problems in economics [2], biology [3], psychology [4], and political science [5][6].

In a previous paper [7] Multiple Agent Genetic Networks (MAGnet) were shown to provide a description of a set of agents which was close to fingerprinting [8]. Fingerprinting uses a continuum of agents drawn from Joss-Ann strategies, which range from Tit-for-Tat to its inverse Psycho, both defined in Section II-C, with various levels of noise. However, as MAGnet sorts based on the provided set of target agents, it has a divergence based on the agent's behaviours taking into account dominators. Dominators [9] are classifications of agents that play optimally against a defined set of other agent types taking into account the restrictions presented by a genetic algorithm's population compared to Evolutionary Stable Strategies (ESS).

ESS [10] have been shown not to be stable for a number of evolutionary algorithm populations [11][12][13]. This failure is most notably caused by the evolutionary algorithms having finite population, compared to the infinite expansion required for the determination of an ESS. Also ESS only allow for a organism to persist if it has a strictly higher fitness than the others in the population; however, the majority of EAs allow for those which are equally fit to persist in the population. Dominators are an alternative description of the properties which a ESS is trying to model, which can be used when the assumptions required to have an ESS fail. Dominators will present given a finite population of a set of agents, where the evolution tend towards.

ESS cannot be relied upon to give a definitive solution to the problem of evolution of IPD playing agents. The

[1]In the case that the game length is known, it can be found that both will defect for the entire game. The reasoning is that in the last move of the game, there is the Nash equilibrium to defect. Knowing that the last move will be defect from both players means that the game can be seen as having one less round. This line of reasoning then propagates back to the first round. Both players defect for the entire game.

978-1-4673-5311-3/13/$31.00 ©2013 IEEE

		Player A	
		Cooperate	Defect
Player B	Cooperate	C, C	T, S
	Defect	S, T	D, D

where $S < D < C < T$ and $T + S < 2C$.

Fig. 1. Prisoner's Dilemma. Scores for a pair of actions are in the form Player A, Player B.

evolutionary process in limited populations is more dynamic and networks of agents which play well against each other exist. Hence, discovery of new agents as well as classification becomes an interesting open problem — who am I playing against and how can I beat them. Agent Case Embeddings (ACEs)[14] present a method which allow for the classification of various evolutionary representations against a set of problem cases. However, the cases are presented as singletons and many interesting interactions in evolution of IPD players comes from finding an agent which can do better than a group in a tournament.

Li et al. [15] produce IPD agents which using a set of rule based determinations based on properties of the opposing agents allows for both identification of the other player and a method to defeat them. This system allowed for the construction of agents which generally outplayed Tit-for-Tat based on which opponent population is present.

This study examines how the sorting is affected by changes to the graphs which restrict gene flow in evolution. Evolution applied to IPD on different graphs [16] has been shown to produce different levels of cooperation, which can change from as much as 95% to zero dependant upon the graph chosen. Bryden et al. [17] examined graph based evolution on a number of graphs and problems and discovered that no one graph is superior for evolution, it is problem dependant. Chiong and Kirley [18] examined IPD agents using lattice and in small world networks, this change leads to less cooperative populations. The examination of which graph to use in the evolution of a Multiple Agent Genetic Network is of importance.

A number of biological creatures will manipulate the resources in order to better their chances for survival. Humans are perhaps the most successful at manipulating their environment in order to allow for survival, e.g. agriculture. *Homo sapiens* are not the only organism known for farming. Farmer ants, such as *Cyphomyrmex wheeleri*, have been known to become attached to specific species of fungi for millions of years, even when other food sources are available [19]. These species have experienced a form of 'lock-in'. When a new fungi is available to be cultivated a new species will emerge. The algorithm that is presented uses the idea of manipulation of environment as evolving agents move problem instances, its food sources, about a graph.

Traditionally Genetic Algorithms (GA) use a single population of data structures to produce a solution to a problem via a process of natural selection and production of other data structures. Spatially Structured GA [20] use multiple populations or graphs which allow for different pathways of evolution to be explored in parallel. However, only the genes of the agents are exchanged, the problem instances, are held the same for each population. The proposed method moves the problem instances about on a graph via a genetic agent. Hence, there is a sorting of agent types based on their behaviours.

Ensemble systems [21], while also having procedures in order to change the weight of problem instances, via *boosting* or *bagging* as will be explained below, differs in terms of the goals and implementation. Ensemble systems are used to provide a single general solution to a set of problem instances by providing a number of classifiers and then producing some method of assembling a meta-classification via a voting or weighting system. Evolutionary algorithms with multiple populations using a co-evolutionary approach have been created using this paradigm [22]. MAGnet has been known to, at times, produce a general solution. The goal of the MAGnet, however, is to sort problem instances via the creation of a locally optimal agent. In ensemble based systems, in each round a new random set of problem instances forms the fitness function, *bagging*, or the problem instances are weighted in terms of importance of their solution, *boosting*. MAGnet, on the other hand, allows the evolving agents to move a problem instance, but the set never changes.

The remainder of the paper is organized as follows. The IPD agents representation as finite state machines, evolutionary operations, and a selection of agent types used as the basis for experimentation, are presented in Section II. Section III examines Multiple Agent Genetic Networks in detail. Section IV provides the experimental settings for the system, the graphs used, and the agent classes used for the experiments. The results for these experiments can be found in Section V Finally, Section VI gives conclusions and directions for future study.

II. Iterated Prisoner's Dilemma Agents

A. Representation

Representation of the agents which play iterated prisoner's dilemmas in an evolutionary algorithm has been shown to also affect the types of agents which present themselves [23][24][25]. Finite state machines (FSM) have been found to be more cooperative than many other representations. Further, a number of previous studies, e.g. [26][27][13][28], have used evolution of FSM as a basis for the creation of agent types.

The representation used for the IPD players in this study is a Mealy FSM [29]. Mealy FSMs are transducers that operate over a finite alphabet of input symbols and responds from a finite alphabet of output symbols, as it traverses a finite number of states. It is defined as a five-tuple, $\langle Q, I, Z, \delta, \omega \rangle$, where Q is the set of states, I is the set of inputs, Z is the set of output symbols, δ is the state transition function, such that $\delta : I \times Q \rightarrow Q$, and ω is the output values, such that $\omega : I \times Q \rightarrow Z$. As the sizes of agents are fixed, the data structured used to encode this five-tuple is a state transition table. This table stores the action and transition for C and D inputs for each state, and an inital action. The machine begins in state one after the inital action is presented.

B. Evolutionary Operators

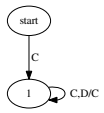
As the method with which we are constructing the agents is evolutionary in nature, the operators must be defined on the population. The evolution of FSM is controlled by a crossover

operation and a set of mutation operators. First is a crossover operation, which is two-point. The crossover exchanges entire states with their transitions and outputs intact between the two machines. There are three defined mutation operations: change a initial state (10%), change a transition (40%), and change an action (50%). The previous work only looked at the transition exchanges, which meant that state actions were held constant and could only be shifted by a crossover action between different agents. The newer division of mutations allows for a better search of the available space and is more in line with the Evolutionary Programming method of manipulation of FMS [30].

C. Agents

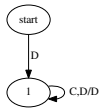
The agents used in this study are shown below as FSM:

Always Cooperate (*ALLC*):



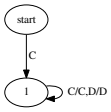
Initial:	C	
State	If C	If D
1	C→ 1	C→ 1

Always Defect (*ALLD*):



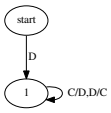
Initial:	D	
State	If C	If D
1	D→ 1	D→ 1

Tit-for-Tat (*TFT*):



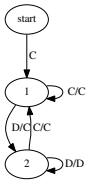
Initial:	C	
State	If C	If D
1	C→ 1	D→ 1

Psycho (*PSYCHO*):



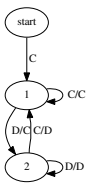
Initial:	C	
State	If C	If D
1	D→ 1	C→ 1

Tit-for-Two-Tats (*TF2T*):



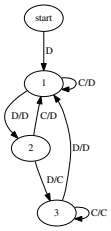
Initial:	C	
State	If C	If D
1	C→ 1	C→ 2
2	C→ 1	D→ 2

Two-Tits-for-Tat (*2TFT*):



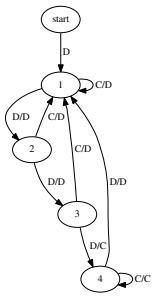
Initial:	C	
State	If C	If D
1	C→ 1	D→ 2
2	D→ 1	D→ 2

Fortress-3 (*FORT3*):



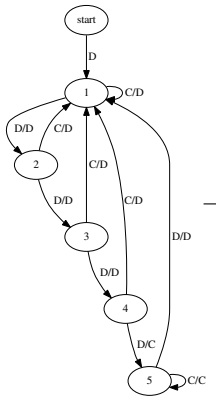
Initial:	D	
State	If C	If D
1	D→ 1	D→ 2
2	D→ 1	C→ 3
3	C→ 3	D→ 1

Fortress-4 (*FORT4*):



Initial:	D	
State	If C	If D
1	D→ 1	D→ 2
2	D→ 1	D→ 3
3	D→ 1	C→ 4
4	C→ 4	D→ 1

Fortress-5 (*FORT5*):



Initial:	D	
State	If C	If D
1	D→ 1	D→ 2
2	D→ 1	D→ 3
3	D→ 1	D→ 4
4	D→ 1	C→ 5
5	C→ 4	D→ 1

It has been recently brought to the author's attention that the naming convention used for the Fortress types differs from that of the originator [13]. The fortress number is based on the number of defections before the handshake, rather than the other convention of the number of states required to implement the minimal representation of the FSM. This first convention has been adopted, as it has interesting properties; the dominator of a set of coprime Fortresses is the product, e.g. Fortress-6 is the dominator of the set of Fortress-2 and Fortress-3. The conversion between the two conventions is simply to add one to the number of defections to get the number of states required for a minimal representation; one state to remember each of the defections and a cooperation state.

III. Multiple Agent Genetic Networks (MAGnet)

Multiple Agent Genetic Networks (MAGnet) are a spatially structured evolutionary algorithm that sorts a collection of problem instances into subsets through the use of evolving agents capable of moving problem instances from one node of a network to another, see Fig. 2.

The agents move about the network based on their ability to solve the problem instances. Each agent is evaluated to find

TABLE I. TABLE OF THE BEST AND WORST POSSIBLE SCORES FOR A MACHINE PLAYING A 100 MOVE IPD. CORRECTION TO SOME OF THE VALUES PUBLISHED IN [7] FOR THE FORTRESSES

Type	Best	Worst
ALLC	500	300
ALLD	100	0
TFT	300	104
2TFT	300	50
TF2T	400	108
PSYCHO	500	0
FORT3	392	0
FORT4	390	0
FORT5	388	0

the fitness on both the current and neighbouring nodes. The agent will then either make a movement or stay depending on its fitness on each of the nodes; it moves to the most fit. The fitness of an agent on a node with no problems, as they have been removed by the agents carrying them away, is defined to be the worst possible score. This is done to allow nodes to lose all their problem instances if the natural number of categories is smaller than the number of nodes. The term *subpopulation collapse* is used to describe the emptying of the collection of problem instances on a node. After an agent moves or stays, it will breed if there are other agents on its updated node. The agents on the node are re-evaluated, as the incoming agent could have brought a new problem instance which changes the fitness for all agents on the node, and a partner is selected. This breeding is done using normal genetic operators of crossover and mutation.

Using the raw fitness score is misleading in this application. For example, the best score when facing an ALLD is less than the worst score when facing an ALLC. The raw numbers would bias the subpopulations to want to contain ALLC, against which any agent can score well. Axelrod has a similar problem with using the 'raw' score as a fitness measure, as he states that a "better standard of comparison is how well you are doing relative to how well someone else could be doing in your shoes ... This is the proper test of successful performance" [5]. In order to remove this bias, we normalize the score on each of the types of machines.

Equation (1) provides the normalization function for a single type. It will produce a value in $[0, 1]$ where 0 is the worst score and 1 is the best.

for some number of generations **do**
  for all agents **do**
    find the node which the agent has the best fitness
    if we probabilistically select to move a problem **then**
      find the problem on the current node which the agent has its highest fitness
    end if
    move the agent and the problem to the node where the agent has the highest fitness
    if there are others upon that node **then**
      select a breeding partner and apply crossover/mutation
    end if
  end for
end for

Fig. 2. Pseudocode of the MAGnet system

$$fitness_{type} = \frac{score_{type} - worst_{type}}{best_{type} - worst_{type}} \quad (1)$$

Table I shows the best and worst possible scores over 100 runs of each of the testing agents which is used to normalize the agents. These scores do not assume that the agents playing have knowledge of the length of the game. If they did they might defect in the last round. The agents in this study do not posses enough states to count to 100, the number of rounds of play used in fitness evaluation. The normalization values for this study were created by exact methods (i.e. it is known that ALLD scores optimally on ALLC, etc.) on the well known-agent types as described in Section II-C. The problem of finding these values for an arbitrary problem instance might impede generalization of the results. The required numbers can be estimated in cases where an analytical method is not feasible by using a heuristic approach, e.g. by an evolutionary algorithm. The *total fitness* of an agent is the average of the fitness it obtains against all the agents on a node. This will produce a score from $[0, 1]$.

IV. EXPERIMENTAL SETTINGS

A. Agent Tests

1) Experiment 1 — Canonical Agents: The first set of agents is: ALLC, ALLD, TFT. These are the most commonly used in studies and are known from previous work [7] to have a dominator known as *Trifecta* which caused a total subpopulation collapse on the fully connected K-5 graph, see Fig. 4. It is likely that this machine would arise again, especially as the minimal implementation of Trifecta can be accomplished with three states. However, while Trifecta is known to dominate the set, it does not dominate two of the singletons, ALLD and TFT. Hence, different/larger graphs may allow for a more interesting breakdown in the associations.

2) Experiment 2 — Expanded Agent Set: This experimental set includes: ALLC, ALLD, TFT, 2TFT, TF2T, PSYCHO, FORT3, FORT4, FORT5. This is a more comprehensive test of the MAGnet's ability to sort agent types. FORT 3-5, are also all coprime making the implementation of a dominator without at least their product of states in the machine impossible. These machines should split well based on their behaviours.

B. Graphs

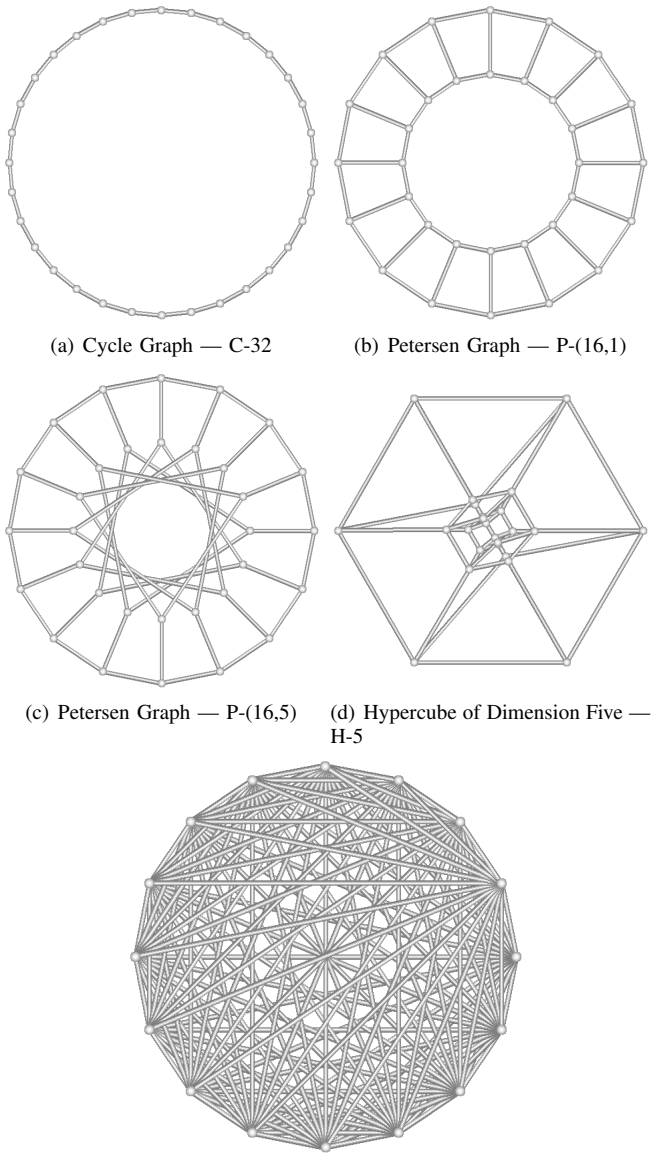
It is assumed the reader has some familiarity with graph theory. This section gives only a short review of the necessary properties to understand the results, see [31] for a good reference. A *simple graph* $G(V, E)$ is a non-empty set V of vertexes or nodes and a set E of unordered pairs of elements of V, called edges. Two distinct nodes, v_1 and v_2, are said to be *neighbours* if $(v_1, v_2) \in E$. The number of edges in E which contain a vertex is called the *degree*. A graph is *connected* if there is a pathway via the edges from any vertex to any other vertex. The graphs selected are all connected graphs with thirty-two vertexes, hence, the examination is on the type of connections, and not on the number of nodes. Two values are examined to measure the graphs' connections: *regularity* and *diameter*. If all vertexes in a graph have the same degree it is said to be *regular*. If the common degree of a graph is k, then

the graph is said to be k-*regular*. The *diameter* of a graph is the largest number of edges in any shortest path between any of the vertexes. It can be considered as the shortest path along the graph.

Fig. 3 presents the five graphs used for the experiments. The least connective is the Cycle graph C-32. This graph has a regularity of 2, the lowest, and a diameter of 32, the highest. Also selected are two Petersen graphs, P-(16,1) and (16,5). P-(16,1) is a two-cycle graph of size sixteen with a connection between each of the nodes in the outer cycle with the inner cycle. P-(16,5) similarly has two connected groups, the first is a cycle graph of sixteen nodes, the second inner graph is also a cycle but with each fifth vertex connected. Hence P-(16,1) and (16,5) both have the same regularity of 3, however, their diameters are 9 and 6 respectively. The hyper cube of dimension five, H-5, has a regularity of 5 and a diameter of 5. The final graph selected is the complete graph of thirty-two vertexes, K-32, which acts as the baseline as K-5 was used in the pervious work. K-32 has the highest regularity of the selected graphs, 31, and lowest diameter with 1.

C. MAGnet Settings

A population of three times the number of nodes, i.e. ninety-six, was created of finite state machines represented by their transition tables. These evolving agents are initialized at random, and distributed about the nodes at random. The agents have a space of 6 states, which is not enough memory to overcome the hundre rounds in the game; hence the action seen in footnote 1 cannot emerge. Evolution takes place when an agent moves to a new node. The chance to move an agent was set to 100% and the chance that an agent takes a problem instance to their new location was set to 50%. A tournament selection which compares two randomly selected agents on a node and returns the more fit of the pair was used as the selection mechanism. The operations of crossover and mutation are always applied if there is a breeding step. The MAGnet is run for one-hundred replicates. Thirty-two problem instances of each type of IPD player where randomly distributed around the graph. This is in order to show that the associations provide a true sorting and not variations caused by random chance.



(a) Cycle Graph — C-32

(b) Petersen Graph — P-(16,1)

(c) Petersen Graph — P-(16,5)

(d) Hypercube of Dimension Five — H-5

(e) Example Fully Connected Graph — K-16 (Used is K-32)

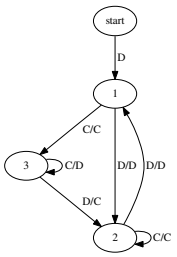
Fig. 3. Graphs used for the MAGnet experiments from least to most regularity

V. Results

In these results we look at the generality of solutions to problem instances, and therefore must have a method to describe over the course of multiple runs, if two problems have similar solutions. In each run of a MAGnet we call two problem instances A and B *associated at the* x*th level* if there exists a subpopulation node where the number of both problem types is greater or equal to x, i.e. $|A| \geq x$ and $|B| \geq x$. This can also be presented as a percentile of the number of problem instances which are in the populations, in this case thirty-two. When association value is examined over multiple runs it forms a diagram which can imply the existence of general solutions to a subset of problems. These diagrams use white to represent an association found between two problem instances in one-hundred runs of the MAGnet system and black means there is no association found. Note that these results are not corrected for subpopulation collapse. This means that the colour of all squares will lighten when the subpopulations collapse to one node. When a general solution exists it typically yields an image with an overall light shade like that of K-32 in Fig. 7.

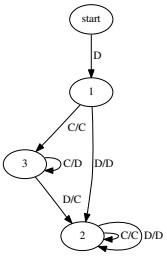
A. Experiment 1 — Canonical Agents

In almost all cases for the complete graph there is a collapse to a single node of all problem cases. There is no ability for MAGnet to separate this graph into disconnected sub-graphs; problems all can drop to the single node and Trifecta can emerge. This differs from the test made on the graph K-4 in [7] which found that it would always collapse onto a single node. This is in part due to the size of the graph of thirty-two nodes rather than five which allows for a greater chance of two or three nodes spilting the space. Note that the outcome on the main diagonal is as expected, a high relation of a node with itself. Looking at H-5 we see that it classifies each of the agent types into their own node. It allowed for a full separation



Initial: State	D If C	If D
1	C→ 3	D→ 2
2	C→ 2	D→ 1
3	D→ 3	C→ 2

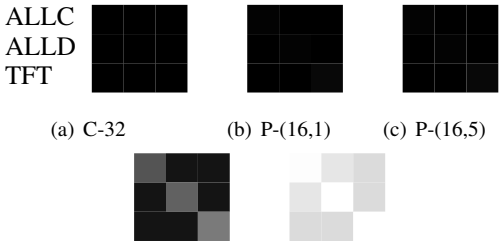
(a) Type A



Initial: State	D If C	If D
1	C→ 3	D→ 2
2	C→ 2	D→ 2
3	D→ 3	C→ 2

(b) Type B

Fig. 4. Trifecta agent types. Note the difference in the final state between returning and a TFT like state. The first can be exploited by a play of $(CDD)^*$, the second will drop into the TFT state and is more defensive.



(a) C-32 (b) P-(16,1) (c) P-(16,5)

(d) H-5 (e) K-32

Fig. 5. Association at the 50% level (16-problem instances on the same node out of 32) for the set of canonical agents: ALLC, ALLD, TFT.

(a) C-32 (b) P-(16,1) (c) P-(16,5)

Fig. 6. Association at the 25% level (16-problem instances on the same node out of 64) for the set of canonical agents with increased number of problem instances.

between the nodes. Looking at the individual runs for H-5, small families of just TFT, ALLC, and ALLD nodes occur. In a few cases nodes join TFT and ALLC, the nice players, with ALLD going off to its own nodes. The levels of TFT on such nodes are always slightly higher than the ALLC in order to prevent the ESS from being disrupted. As an evolutionary algorithm does not require a replacement if the score is strictly higher, both TFT and ALLC will score the same against a *nice* opponent. This forms a random walk on the levels of TFT and ALLC which can be exploited over the population by a ALLD. This explains the darker links between them and the darker main diagonal between K-32 and H-5.

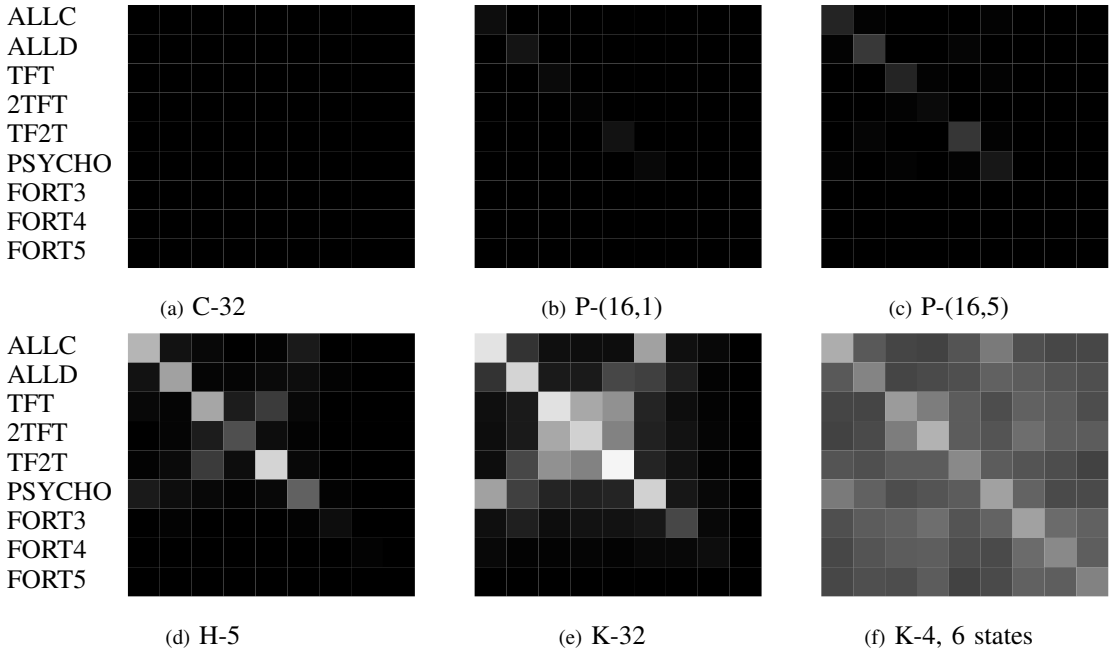
For the cycle graph and the two Peterson graphs there is no informative association produced via MAGnet. The reason for this is the properties of subpopulation collapse based on the regularity and degree of the graphs involved. All of these graphs have a low regularity. If a node loses all of its problem instances, which will probabilistically be likely to happen given the types of agents, then it is in effect removed from the graph; the node has the worst possible fitness score and no agent would choose to move to that location. A node undergoing such a collapse reduces the degree of all neighbouring nodes, and given a few of these nodes occurring will break down the graph into a number of disjoint subgraphs. These subgraphs will then independently sort the instances given the agents and instances in each subgraph. When a graph is fully connected or has a high regularity their is little chance of such a disjoint graph occurring, hence we see more association as the nodes removed do not change the graph's connectedness.

Looking at individual runs in the first experiment, however, tells an interesting story when it comes to why these associations are not available. Looking at the results for the Peterson and Cycle graphs, a large number of nodes still have problem instances, but these nodes are all of a distance of two or more away of any other node with problems. That confirms the idea of a breakdown in the graphs via the subpopulation collapse. On the individual remaining nodes there is a large number of a few problem instances of the same type, and usually one or two with a more sizable amount of all the types in a close to equal proportion. This seams to imply that Trifecta has been created on the large nodes and has drawn them together. However, other nodes have been 'stranded' as all of their neighbours flood into them. The diameter plays a smaller role in this than the degree of regularity. There are a few cases in which neighbours stay joined with problem instances. Those problem instances are for the most part the same type of player. This would develop an equilibrium in which the agents would either not move as both of the nodes are equally fit, or in the case of having a single difference, would mutate into each other creating an oscillation of agents and problem cases between the two nodes.

As an additional test of the system, the number of problem instances was doubled to sixty-four and the accepted level of association was halved to 25% before the Peterson graphs and the Cycle graph displayed the ability to classify the problem instances. This is demonstrated in Figure 6.

B. Experiment 2 — Expanded Agent Set

The second testing set shows very similar properties for each of the graphs as shown with the smaller canonical set. Looking at the levels of association, Fig 7, the complete graph K-32 presents an outcome which is close to K-4, though it highlights more of hard connections between the two, due to the greater number of nodes. It cleans out some of the noise seen in K-4, due to random chance fluctuations of being placed



(a) C-32 (b) P-(16,1) (c) P-(16,5)

(d) H-5 (e) K-32 (f) K-4, 6 states

Fig. 7. Association at the 50% level (16-problem instances on the same node out of 32) for the five graphs using the second larger set of agent types. Presented in 7(f) is K-4 with 6 states from the previous work to allow for comparison tool. Settings for K-4 can be found in [7], and differ from these tests.

on the same graph. H-5 shows a cleaner separation than K-32 on some of the agent types. For example, there is a very stark difference visible between 2TFT and TF2T/TFT via the H-5 graph not seen in K-32. This is interesting as their is a method of exploiting 2TFT by making a defection on every other move. This ability for exploitation is not seen in TF2T or TFT. PSYCHO is also separated more from ALLD, and ALLC and ALLD are more separated.

Again the fortresses stand alone, being not even associated with themselves. The handshaking strategies seem to be more resistant to being discovered by this behavioural technique of classification. Seeing as how they were originally discovered via a deep evolution, and they protect their behaviours closely, this is not surprising that MAGnet is having a hard time associating them to themselves. An agent would have to learn the hidden secret handshake in order to score. This was perhaps easier on a smaller graph than the thirty-two nodes presented to the agents in this case.

Looking at the higher diameter graphs, again the associations move to nothing. Again this is most likely due to the subpopulation collapse effect cutting the graph into disjoint subsets which will be unable to properly provide a classification. Looking at the individual runs confirms this as the nodes have cut off from their neighbours. The cycle graph seems to show the most chaos in how it is sorted after the run. The runs in general show that the fortresses are being classified with each other and themselves but they are very likely to be on small disjoint nodes with only a few of them. This could be again due to long deep evolution required, and the strategy for playing against them not easily emerging.

VI. Conclusions

The MAGnet system allows for the creation of an association between IPD play types via a behavioural and evolutionary approach. The system allows for the sorting of agents based on their evolutionary fitness from a set of moving IPD agents which are allowed to move instances of the problem about a graph. This study looked a number of graph types and examined the outcomes. It was found in general that graphs with a high degree of k-regularity provide the best basis for a classification. They allow for an agent to test its might against a wide variety of situations as well as provide more opportunities for problem instances to group together. Further, as the number of classification groups is less than the number of nodes provided, there is a property of subpopulation collapse which will in effect remove nodes from the graph. When the created graph's diameter can only increase due to this effect, and in some cases can cause disjoint subgraphs to form. As these disconnected components will not allow for the movement of a problem instance between them, this prevents a good classification from being made when looking at the levels of association between types.

For the larger graphs, the levels of association in general dropped when compared to the original tests on the fully connected graph of four nodes, K-4. However, the associative diagrams still show the family of strong connections which is presented by the fully connected graph. That is such results as ALLC and ALLD being close in terms of what defeats them. The TFT family are still seen as being close. Though 2TFT and TF2T are farther apart from each other. PSYCHO is shown being close to ALLD, as playing ALLD against both will produce an optimal score. The Fortresses are presented as not being close to anything in terms of their player behaviours. Hence, we have shown that MAGnet is able to scale up on the same graph type while still producing similar results.

The prototype for MAGnet was a distributed co-evolution of IPD players. The findings for this paper are mixed as to if MAGnet would be a good candidate for a distributed

approach. On the one hand each of the nodes, or a group of nodes, could be represented on the same processor element and requests for fitness and return of scores could be independently made. However, benefits would be seen primarily when the graph selected is sparse in terms of connectivity as this would require less messages. As we have seen MAGnet works best as a classification method when the graph is highly connected. However, the result for the H-5 graph in particular is interesting for an implementation of a distributed MAGnet as the hypercubes are a common parallel computing architecture. This implies a distributed MAGnet might do well on a hypercube based system with each processing element taking on the computation of a single node.

Further, there is a requirement to attempt to remove the need for normalization in order to allow for a more general MAGnet classification. This removal could be done by either making a preprocessed step to find the normalization values via a GA, or by letting the normalization value find themselves over the course of the evaluation by setting the high and low based on agents discovered in the classification process. There is little but speculation as to what such a change would do to the classifiers. On the one hand, by having the normalization values known beforehand there is perhaps too little of an evolutionary pressure to classify for those problem instances. Much like how fortresses seem to be resistant, even on the larger graphs, to be properly classified. On the other hand there might be the case that the populations and nodes will settle on being just marginally better than another on a problem and hence there is now too little pressure in general to sort agents properly. Which of these two hypothesis holds would have to be experimentally discovered.

ACKNOWLEDGEMENTS

The author would like to thank Daniel Ashlock for his helpful comments and the National Science and Engineering Research Council (NSERC) of Canada.

REFERENCES

[1] D. Ashlock, E.Y. Kim, and W. Ashlock, "A fingerprint comparison of different prisoner's dilemma payoff matrices", in *2010 IEEE Symposium on Computational Intelligence and Games (CIG)*, 2010, pp. 219–226.

[2] M. Hemesath, "Cooperate or defect? russian and american students in prisoner's dilemma", *Comparative Economics Studies*, vol. 176, pp. 83–93, 1994.

[3] K. Sigmund and M. A. Nowak, "Evolutionary game theory", *Current Biology*, vol. 9, no. 14, pp. 503–505, 1999.

[4] D. Roy, "Learning and the theory of games", *Journal of Technical Biology*, vol. 204, pp. 409–414, 2000.

[5] R. Axelrod, *The Evolution of Cooperation*, Basic Books, 1984.

[6] W. Poundstone, *Prisoner's Dilemma*, Anchor Books, 1993.

[7] J. A. Brown, "Multiple agent genetic networks for iterated prisoner's dilemma", in *2011 IEEE Symposium on Foundations of Computational Intelligence (FOCI)*, 2011, pp. 7–14.

[8] D. Ashlock and E.Y. Kim, "Fingerprinting: Visualization and analysis of prisoner's dilemma strategies", *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 5, pp. 647–659, 2008.

[9] J. A. Brown and D. A. Ashlock, "Domination in iterated prisoner's dilemma", in *24th Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2011, pp. 1125–1128.

[10] J. Maynard-Smith, *Evolution and the Theory of Games*, Cambridge University Press, Cambridge, UK, 1982.

[11] D. B. Fogel, G. B. Fogel, and P. C. Andrews, "On the instability of evolutionary stable strategies", *BioSystems*, , no. 44, pp. 135–152, 1997.

[12] S. G. Ficici and J. B. Pollack, "Effects of finite populations on evolutionary stable strategies", in *Proceedings of the 2000 Genetic and Evolutionary Computation Conference*, 2000, pp. 927–934.

[13] W. Ashlock and D. Ashlock, "Changes in prisoner's dilemma strategies over evolutionary time with different population sizes", in *Proceedings of the 2006 Congress On Evolutionary Computation*, 2006, pp. 1001–1008.

[14] D. Ashlock and C. Lee, "Agent-case embeddings for the analysis of evolved systems", *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 2, pp. 227–240, 2013.

[15] Jiawei Li, P. Hingston, and G. Kendall, "Engineering design of strategies for winning iterated prisoner's dilemma competitions", *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 4, pp. 348–360, 2011.

[16] D.A. Ashlock, "Cooperation in prisoner's dilemma on graphs", in *2007 IEEE Symposium on Computational Intelligence and Games (CIG)*, 2007, pp. 48–55.

[17] K.M. Bryden, D.A. Ashlock, S. Corns, and S.J. Willson, "Graph-based evolutionary algorithms", *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 550–567, 2006.

[18] R. Chiong and M. Kirley, "Effects of iterated interactions in multiplayer spatial evolutionary games", *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 4, pp. 537–555, 2012.

[19] N.J. Mehdiabadi, U. G. Mueller, S. G. Brady, A. G. Himler, and T. R. Schultz, "Symbiont fidelity and the origin of species in fungus-growing ants", *Nature Communications*, vol. 3, no. 840, 2012.

[20] E. Cant-Paz, "A survey of parallel genetic algorithms", *Calculateurs Paralleles*, vol. 10, 1998.

[21] R. Polikar, "Ensemble based systems in decision making", *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[22] D. A. Augusto, H. J. C. Barbosa, and N. F. F. Ebecken, "Coevolutionary multi-population genetic programming for data classification", in *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*, 2010, pp. 933–940.

[23] D. Ashlock, E-Y. Kim, and N. Leahy, "Understanding representational sensitivity in the iterated prisoner's dilemma with fingerprints", *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 36, no. 4, pp. 464 –475, 2006.

[24] W. Ashlock, "Why some representations are more cooperative than others for prisoner's dilemma", in *2007 IEEE Symposium on Foundations of Computational Intelligence (FOCI)*, 2007, pp. 314–321.

[25] H. Ishibuchi, H. Ohyanagi, and Yusuke Nojima, "Evolution of strategies with different representation schemes in a spatial iterated prisoner's dilemma game", *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 1, pp. 67–82, 2011.

[26] D. B. Fogel, "Evolving behaviors in the iterated prisoner's dilemma", *Evol. Comput.*, vol. 1, no. 1, pp. 77–97, 1993.

[27] D. B. Fogel, "On the relationship between the duration of an encounter and the evolution of cooperation in the iterated prisoner's dilemma", *Evol. Comput.*, vol. 3, no. 3, pp. 349–363, 1995.

[28] M. T. Tu, E. Wolff, and W. Lamersdorf, "Genetic algorithms for automated negotiations: a fsm-based application approach", in *11th International Workshop on Database and Expert Systems Applications*, 2000, pp. 1029–1033.

[29] G. H. Mealy, "A method of synthesizing sequential circuits", *Bell Systems Technical Journal*, vol. 34, pp. 1054–1079, 1955.

[30] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artifical Inteligence though Simulated Evolution*, John Wiley & Sons, New York, 1966.

[31] E. G. Goodaire and M.M. Parmenter, *Discrete Mathematics with Graph Theory (3rd Edition)*, Prentice-Hall, Inc., 2005.