Neuroevolution of Multimodal Ms. Pac-Man Controllers Under Partially Observable Conditions

William Price[1] and Jacob Schrum[2]

***Abstract*— Ms. Pac-Man is a well-known video game used extensively in AI research. Past research has focused on the standard, fully observable version of Ms. Pac-Man. Recently, a partially observable variant of the game has been used in the MS. PAC-MAN VS. GHOST TEAM COMPETITION at the Computational Intelligence and Games (CIG) conference. Restricting Ms. Pac-Man's view makes the game more challenging. Ms. Pac-Man can only see down halls within her direct line of sight. The approach to this domain presented in this paper extends an earlier approach using MM-NEAT, an algorithm for evolving modular neural networks. Experiments using several forms of evolved and human-specified modularity are presented. The best evolved agent uses a human-specified task division with output modules for different situations: no ghosts, edible ghosts, and threat ghosts. This approach placed first at the MS. PAC-MAN VS. GHOST TEAM COMPETITION at CIG 2018 against seven other competitors with an average score of 7736.63.**

I. INTRODUCTION

Ms. Pac-Man is a challenging domain for several reasons. One is that the game is non-deterministic. Another is that an agent must demonstrate a number of intelligent behaviors to succeed. It must be able to navigate the maze and collect all the pills. In addition, the agent must be able to distinguish between threatening ghosts and edible ghosts and act appropriately. Despite these challenges, many successful approaches to the game have been developed [1], [2], [3].

Adding partial observability makes this problem even harder [4] (Fig. 1). This constraint limits the amount of state information an agent can observe. Therefore, Ms. Pac-Man must reason about the locations of both pills and ghosts that she cannot see. This lack of information makes simple actions such as turning a corner much riskier, since an unseen ghost could suddenly appear and catch Ms. Pac-Man.

To address the challenge of partial observability, the agents described in this paper use models of the pills and ghosts. The pill model tracks which pills the agent has eaten and which remain. Since initial pill locations are known, this pill model gives the agent perfect information about the state of pills in the maze despite partially observable conditions. In contrast, the ghost model can only provide probabilistic information. This model keeps track of sightings of ghosts and infers the possible locations of each previously observed ghost based on the movement rules the ghosts follow. The model also tracks whether or not the ghosts are edible.

[1]W. Price is a recent graduate of Southwestern University, Georgetown, TX 78626, USA pricew@alumni.southwestern.edu

[2]J. Schrum is an Assistant Professor of Computer Science at Southwestern University, Georgetown, TX 78626, USA schrum2@southwestern.edu

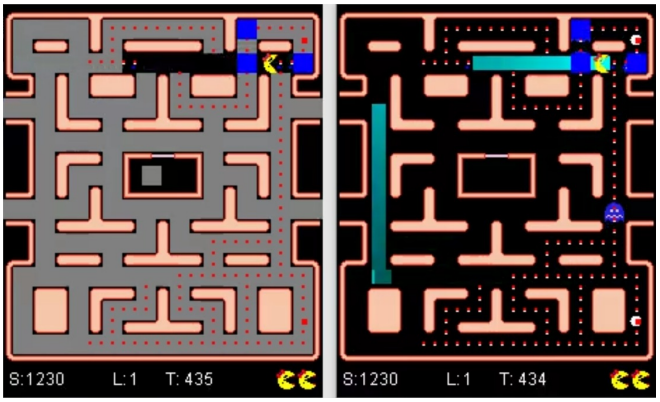


Fig. 1. **Champion Ms. Pac-Man Behavior from Partially Observable and Fully Observable Perspectives.** Left: the maze as perceived by Ms. Pac-Man. Areas she cannot see are gray, but the blue boxes represent locations where she expects an edible ghost might be. Right: same state with hidden components revealed. There is one ghost in the lower right that Ms. Pac-Man cannot see, and which her ghost model is not aware of. This champion uses distinct control modules, and the light blue paths indicate spaces in the maze where she was recently using a multitask module associated with seeing edible ghosts. This agent clears all four mazes.

These models feed sensor information to neural network controllers evolved with Modular Multiobjective NeuroEvolution of Augmenting Topologies (MM-NEAT [1]), which has been successful in fully observable Ms. Pac-Man. Networks use different output modules to handle different situations, encouraging multimodal behavior. These modules can either be hand designed or discovered via evolution. This paper presents results using several modularity schemes supported by MM-NEAT, but drops the use of multiple objectives. This paper demonstrates that the addition of models for unseen phenomena can help an established method for developing multimodal behavior, MM-NEAT, succeed in a partially observable domain. An agent using this approach placed first in the Ms. Pac-Man track of the 2018 MS. PAC-MAN VS. GHOST TEAM COMPETITION[1].

II. PREVIOUS MS. PAC-MAN RESEARCH

Pac-Man and Ms. Pac-Man have been the focus of much research [5], the most relevant of which is presented below.

A. Full Observability

Early approaches to (Ms.) Pac-Man used genetic programming, including work done by Koza [6] and Lucas [3]. These approaches used high-level actions that were hand designed, which requires domain knowledge about Ms. Pac-Man.

[1]http://www.pacmanvghosts.co.uk/results.html

Alhejali and Lucas combined genetic programming with training camps to produce better agents for Ms. Pac-Man [7]. Training camps pit agents against sub-problems of a domain, which are easier to learn solutions to. Once solutions to each sub-problem have been learned, the solutions can be aggregated into a policy for playing Ms. Pac-Man. These training camps improve performance, but require an expert to design the camps in the first place.

Brandstetter and Ahmadi used genetic programming, but with directional sensors [2], as done in this paper. A directional sensor is one sensor that is evaluated for each available movement direction, leading to different sensor readings. A preference is generated for each direction, resulting in the agent moving in the direction with the highest preference.

Schrum and Miikkulainen also used directional sensors, but with modular neural networks [1]. Each module represents a different policy for the agent to follow. These policies may be discovered through evolution or handcrafted. This approach is modified to work in the partially observable version of the game within the current paper.

Ghost controllers can also be evolved. For example, Cardona et. al. [8] evolved both Ms. Pac-Man and ghost controllers using competitive coevolution. Agents used minimax tree search with evolved weights for a linear combination utility function. They found better performance when evaluating the top three controllers from each competing population as opposed to evaluating only the champion.

Also relevant is Deep Reinforcement Learning, which is famous for succeeding in many Atari games using raw pixel inputs [9]. The Atari version of Ms. Pac-Man initially proved challenging for these methods, but Seijan et. al. [10] mastered the game using a combination of deep learning, modular network structure, and collaborative learning. The modular network structure used is similar to the approach of Schrum and Miikkulainen [1] used in this paper. Learning is collaborative because separate agents focus on highly granular in-game goals, and their individual preferences are aggregated to control Ms. Pac-Man. Though learning from raw pixel input is impressive, this version of the game lacks the challenge of partial observability faced in this paper.

B. Partial Observability

The MS. PAC-MAN VS. GHOST TEAM COMPETITION[4] is an international competition hosted at the Computational Intelligence and Games (CIG) conference. Partially observable conditions add a layer of complexity to the task of creating a successful agent. One can submit both Ms. Pac-Man and ghost agents. Tournament play is round-robin style. Ms. Pac-Man agents compete against all ghost agents multiple times. Scores are averaged to produce a final score for ranking. Because the competition began in 2016, relatively little research has been done in this variant of the game.

The competition code provides several starter agents for a baseline level of performance. The starter Ms. Pac-Man agent follows a simple rule set and scores below 3000 on average against the starter ghost agents, which are also rule based. Ghost teams consist of four copies of one agent controller.

Coevolution has previously been used to create Ms. Pac-Man and ghost controllers under partial observability conditions [11]. Specifically, genetic programming was used with high-level actions, as in early Pac-Man research [6], [3]. It would be interesting to see how the coevolution of controllers using low-level actions performs, though the current paper is purely restricted to the evolution of Ms. Pac-Man controllers.

III. MS. PAC-MAN DOMAIN

The fully observable and partially observable Ms. Pac-Man domains are both described in detail.

A. Full Observability

The objective of Ms. Pac-Man is to maximize score by eating pills scattered about four different mazes. Each pill is worth 10 points. Ms. Pac-Man navigates mazes, eating pills as she comes into contact with them. Upon eating all pills in a given maze, Ms. Pac-Man advances to the next maze.

Hostile ghosts wander each maze, chasing Ms. Pac-Man. Collision with a ghost results in a lost life for Ms. Pac-Man, ending the game if she has no more lives. She starts with three lives, and can gain a fourth by earning 10000 points. There are four special power pills located in the corners of each maze that allow Ms. Pac-Man to eat ghosts for a limited time. Eating ghosts earns points: 200, 400, 800, and 1600 points for the first, second, third, and fourth consecutively eaten ghost respectively. Therefore, it is advantageous to hunt as many ghosts as possible during the duration of a power pill. While ghosts are edible, they move at half speed. In each new maze, the time that ghosts remain edible from a power pill decreases. Power pills themselves yield 50 points.

Ghosts only make decisions at junctions, with two exceptions. Each ghost's direction is reversed when Ms. Pac-Man eats a power pill, or with a small probability on any time step. In contrast, Ms. Pac-Man can always move in any direction.

Eaten ghosts are returned to the lair in the center of the maze where they are out of play for a short amount of time. However, if a ghost should emerge from the lair during the duration of a power pill, Ms. Pac-Man will have to deal with both threatening and edible ghosts at the same time.

In the original Ms. Pac-Man game, there are fruits that randomly spawn in the center of the maze. These fruits give points when consumed like a pill. However, the simulator used in the competition does not contain fruit.

B. Partial Observability

In traditional Ms. Pac-Man, agents have knowledge about the entire game state. Players can see ghosts in the lair as well as the position and movement direction of ghosts outside the lair. Players can also see the locations of all uneaten pills.

In the partially observable game, agents have a reduced view of the game state. Multiple types of partial observability are supported. Vision can be restricted to a radius around Ms. Pac-Man using either Manhattan distance or Euclidean distance. Alternatively, vision can be restricted to line of sight, meaning Ms. Pac-Man can see for a limited distance in straight lines down hallways, and walls block her vision.

Nothing around a corner can be seen. The line of sight constraint is used in the competition and assumed for the rest of this paper. Ms. Pac-Man has no information about the state of the lair, or about ghosts or pills she cannot see.

This partially observable version poses new challenges to Ms. Pac-Man. It requires her to have a memory of previous states of the game, including pills already eaten and the locations of ghosts previously seen. From this memory, she must decide the proper actions to take. Even with such memory, bad luck can cause Ms. Pac-Man to be caught off guard when rounding a corner. She could also be surrounded and trapped by ghosts she cannot see, especially if the ghosts are allowed to have full access to the game state. The simulator can be configured to impose partial observability on Ms. Pac-Man, the ghosts, or both. The competition restricts all agents to partial observability, though ghosts with full observability are also used in the experiments of this paper.

IV. EVOLUTIONARY METHODS

Controllers were evolved in a manner similar to previous research in the fully observable game using MM-NEAT[2] [1].

A. Evolutionary Algorithm

Ms. Pac-Man is treated as a single objective problem of maximizing game score. This approach contrasts to previous work using MM-NEAT in the fully observable game [1], which used separate pill and ghost score objectives with the multiobjective evolutionary algorithm NSGA-II [12]. This paper shows that one objective is sufficient to produce skilled agents in the partially observable version of the game.

In this paper, simple $(\mu + \lambda)$ selection is used. First, μ parent networks are evaluated. Then tournament selection is used with crossover and mutation to produce λ child networks, which are also evaluated. From the combined $(\mu + \lambda)$ size population, the μ best performing networks form the next parent generation. This pure elitist approach pits parents against their children, potentially saving valuable network structures that were not passed to child networks.

B. NeuroEvolution of Augmenting Topologies

Because neural networks control the Ms. Pac-Man agents, a way of encoding these networks is needed. This paper uses the genome representation from NEAT (NeuroEvolution of Augmenting Topologies [13]). NEAT is an algorithm for evolving neural networks with arbitrary topologies. It has been successful in a variety of domains [14], including the fully observable version of Ms. Pac-Man [1].

The crux of NEAT is the insight that specific nodes and edges tend to serve the same purpose in different networks with a common ancestor. Therefore, NEAT assigns identification numbers to every node and edge in a network when it appears. These IDs allow for networks to be aligned in a sensible way during crossover (sexual reproduction).

In addition to crossover, NEAT employs three types of mutation. First, weights of edges can be slightly perturbed. Second, an edge can be added between two nodes. Finally, nodes can be added by splitting an existing edge in two. These mutations allow networks to gradually complexify from a simple starting point of networks with no hidden nodes, resulting in sparse but effective networks.

[2] http://southwestern.edu/~schrum2/re/mm-neat.php

C. Modular Networks

Effective Ms. Pac-Man agents display multiple modes of behavior. These modes address different situations she must face: hunting edible ghosts, eating pills, and fleeing threats, among others. MM-NEAT encourages multiple modes of behavior with modules in the output layer of the network.

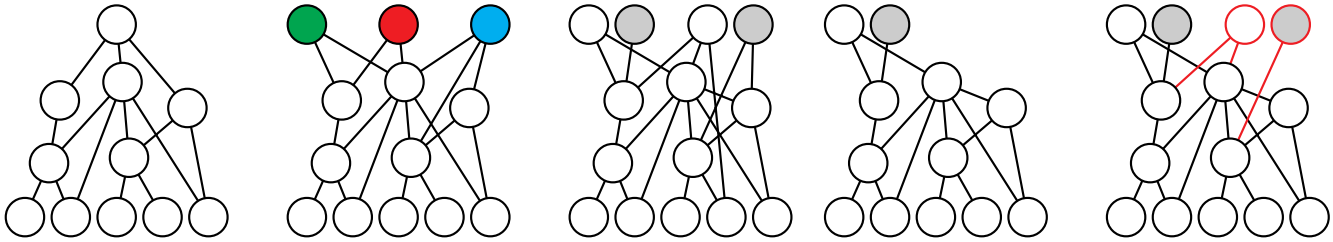
The modular network architectures supported by MM-NEAT are summarized in Fig. 2. These architectures are the same as presented in [1]. A module is simply a group of outputs that can define the behavior of an agent. Each module reacts differently to the inputs fed into the network, allowing for distinct behaviors to be represented by each module.

One module networks (1M) are standard neural networks (Fig. 2a). In Ms. Pac-Man, these networks have one output: preference for the currently considered direction (Section V-B). Multitask networks (Fig. 2b) have a fixed number of modules/outputs that are used according to a human-designed task division. One output is used on each time step and the others are ignored. The task division in this paper has three output modules (3MT, Section V-E). Networks can also have a fixed number of modules, each combining a *policy* neuron with a *preference* neuron. Policy neurons are the usual output neurons that define agent behavior (direction preference). Each policy neuron is combined into a module with a preference neuron, and the module whose preference neuron has the greatest activation for a given set of inputs is the module whose policy neuron defines the behavior of the network. Networks with two (2M, Fig. 2c) and three (3M) preference modules are used in this paper.

Another approach to developing network modularity is to let evolution decide on the number of modules. Networks start with one preference module (Fig. 2d), but can gain additional preference modules via module mutation. There are several forms of module mutation [1], but the one used in this paper is Module Mutation Duplicate (MM(D), Fig. 2e). MM(D) duplicates an existing network module. The new module has a policy neuron with the same incoming edges as the policy neuron of a randomly chosen module being duplicated. However, the preference neuron within that module creates an edge from a random node in the network. Therefore, the module will be used at different times than the original module it was modeled after. At first, both modules behave the same, but are used at different times. Across generations, the behaviors of the two modules can diverge in ways advantageous to the network.

V. EXPERIMENTS

A single 3MT Ms. Pac-Man controller was victorious in the MS. PAC-MAN VS. GHOST TEAM COMPETITION at CIG 2018, but several experiments were conducted after submitting the final entry in order to gain a deeper understanding of how various related methods perform in the domain.



(a) One module (b) Three Multitask Modules (c) Two Preference Modules (d) Before Module Mutation (e) After MM(D)

Fig. 2. **Modular Networks:** These example networks require one policy neuron to define agent behavior, as in Ms. Pac-Man. Inputs are at the bottom, and outputs are at the top. (a) Standard neural network with just one module. (b) Multitask network with three modules, which are color coded. A human-specified task division indicates when to use the green, red, or blue policy. (c) A network with two modules that uses preference neurons (colored gray) to determine which module to use. (d) Starting network in a population where Module Mutation is enabled. It has one module with an irrelevant preference neuron. (e) After MM(D), the network gains a new module that duplicates the behavior of another module. The behavior is the same because its policy neuron linked to the same neuron sources with the same link weights as the policy neuron in the module that was duplicated. However, the new preference neuron is linked to a random source with a random weight so that the new module is used in different situations. This process can be repeated to create more modules. After Module Mutation, all preference neurons become relevant. Extra modules allow networks to learn multimodal behavior more easily by making it possible to associate a different module with each behavioral mode.

A. Pill and Ghost Models

To deal with limited visibility, ghost and pill models are used. These models extend code provided by the competition organizers [4]. The pill model tracks unconsumed pills. Since initial pill locations are known, and Ms. Pac-Man is the only agent that removes pills from the environment, this model provides perfect information about pill locations.

The ghost model tracks potential locations of ghosts based on observations. Initially, no ghosts are known to the model, so for each ghost the probability associated with each location is 0. If a ghost is visible, the probability for that ghost at that location is 1, and other locations become 0. When a ghost leaves view, the model relies on the fact that ghosts can only change directions at junctions. Ghost movement speed is known, so the location of the unseen ghost progresses through the hallway maintaining it's probability. There is a random chance that ghosts will flip direction at any time, but the model does not take this into account, since it would complicate the model with many low-probability predictions.

The probability of a ghost being at a location is split at junctions. Since Ms. Pac-Man cannot see which direction ghosts pick at junctions, the probability that the ghost took any available direction is split evenly from the original probability. Therefore, a ghost prediction with probability 0.5 entering a T-junction will split into two 0.25 predictions in the two *available* directions. When a prediction drops below a threshold of 0.125, it is removed from the model.

The description thus far is for the pill and ghost models provided with the competition code. However, the ghost model did not account for edible ghosts. The enhanced model used in this paper does track whether or not ghosts are edible. Ghosts are marked as edible when a power pill is eaten. The model also accounts for the fact that edible ghosts move at half speed, and resets edible ghosts to threats once they are eaten, or the duration of the power pill effect expires. Tracking this extra information is vital for being able to track and eat edible ghosts, and thus maximizing score.

B. Direction-Evaluating Policy

Ms. Pac-Man agents make movement decisions as in Brandstetter and Ahmadi [2], and previous MM-NEAT research [1]. Each possible direction Ms. Pac-Man could travel is evaluated using the agent's network to produce a preference score. The direction with the highest preference score in a given time-step is the direction that the agent will move in. Because sensors are directional, re-applying the same sensor in each direction will often yield different values. By having the agent choose between primitive actions instead of complex ones, agents are less constrained, and complex behavior that does emerge is all the more impressive.

C. Sensor Configuration

There are 43 sensors, mostly based on those of Schrum and Miikkulainen [1]. This research made a distinction between *conflict* and *split* sensors. Split sensors allow a game entity to be interpreted in different ways by different sensors under different circumstances. Specifically for Ms. Pac-Man, the ghosts are sensed using split sensors, meaning that there is a subset of sensors that apply only to threat ghosts, and another subset of similar sensors that apply only to edible ghosts. In contrast, conflict sensors would only have one set of sensors for ghosts in general, and then some additional sensors indicating whether or not each ghost is edible. Since the previous work showed that split sensors were superior, only split sensors are used in this paper.

Nine sensors are undirected (Table I), meaning that they return the same result for each available movement direction. As a result, these sensors can only differentiate direction preferences in combination with directional sensors. They are either binary sensors, or measure some proportion. They are self-explanatory, with the exceptions of the Edible Time and Power Pill Time. The Edible Time refers to the highest edible time of any currently observed ghost, so it will be 0 when ghosts are not visible, even if the ghost model is aware of edible ghosts. The Power Pill Time tracks the time remaining until the benefit of eating a power pill wears off.

TABLE I
Undirected Sensors in Ms. Pac-Man

Sensor Name	Description
Bias	Constant value of 1
Proportion Pills	Number of remaining regular pills
Proportion Power Pills	Number of remaining power pills
Proportion Edible Ghosts	Number of possible edible ghosts
Proportion Edible Time	Remaining known time ghosts are edible
Proportion Power Pill Time	Remaining possible edible time since eating power pill
Proportion Game Time	Remaining evaluation time
Any Ghosts Edible?	1 if any ghost is suspected of being edible, 0 otherwise
Close to Power Pill?	1 if Ms. Pac-Man is within 10 steps of a power pill, 0 otherwise

This sensor does not depend on any awareness of ghosts, which means it can be positive when edible ghosts are out of view, but also when threat ghosts who have returned from the lair after being eaten are in plain sight. Without the Edible Time sensor, the agent might assume that any visible ghost was edible if it had recently eaten a power pill. This is not always the case as ghosts leave the lair in a non-edible state.

The directional sensors (Table II) are calculated with respect to particular directions. Typical examples are directional distances to the first entity of a given type, e.g. pill, that would be encountered along the shortest path starting in the given direction and never reversing. Sensors can also gather other information about entities, such as the probability that a ghost is present according to the ghost model.

The most complicated directional sensor is Options From Next Junction, which looks at the next junction in a given direction, and counts the number of subsequent junctions that can be safely reached from the first junction without reversing. The safety of a route is determined by taking all agent distances into account and assuming threat ghosts will follow the shortest path to the target junction.

The main difference between the sensors in this paper and previous work [1] is the use of the pill and ghost models. The pill model provides perfect information, so all pill sensors rely on the pill model for sensor readings. However, the ghost model is probabilistic, so sensors about the nearest ghost in a given direction actually provide information about the nearest *potential* ghost location. There are several sensor groups corresponding to the first, second, third, and fourth closest ghosts in a given direction (entries in Table II that mention the n^{th} nearest potential ghost of some type are actually sets of four sensors), but because the ghost model splits predictions at junctions, all four sensors in a set could actually provide information about different possible locations of the same ghost.

It is also possible that a particular entity is not present, or at least not able to be sensed. For example, all power pills may have already been eaten, or the ghost model may not yet be aware of any ghosts. In these cases, sensors return a special value of -1. Otherwise, sensor values are scaled to the range $[0, 1]$. The maximum distance is 200 steps.

D. Ghost Opponents

Ms. Pac-Man agents were evolved against the starter ghost team provided by the competition organizers [4], whose behavior is summarized here. When a ghost reaches a junction, it follows a few logical rules. If the ghost is edible and can see Ms. Pac-Man, or Ms. Pac-Man is close to a power pill, the ghost flees. If the ghost can see Ms. Pac-Man and neither of the previous conditions are true, the ghost pursues her. If the ghost cannot see Ms. Pac-Man, it makes a random move. Random moves only happen under partial observability conditions, but ghost teams can be set to operate with full or partial observability.

E. Multitask Division

The task division used by the multitask approach (3MT) relies on the model of ghost locations rather than any actual observed ghosts. The task division is as follows: One module is used when the ghost model is not aware of any ghosts. The second module is used when the model is only aware of edible ghosts. The third module is used when the model is aware of any threatening ghost. If the model is aware of both edible and threat ghosts, then the third module is used, since the presence of threat ghosts overrides the module for edible ghosts. This conservative approach prioritizes fleeing threats over chasing edible ghosts.

F. Evolving Networks

Five types of networks were evolved: One Module (1M), Two Module (2M), Three Module (3M), Module Mutation Duplicate (MM(D)), and Three Module Multitask (3MT). Population size $\mu = \lambda = 100$. The mutation rates were: 5% chance of weight perturbation per link, 40% chance of a new link, and 20% chance of a new node. In `MM(D)` runs, Module Mutation has a 10% chance of being applied to each offspring. The crossover rate is 50%. These settings were used in previous work [1].

Each modularity type was evolved against ghost teams with both partial and full observability. The time limit was 8000 ticks, which gives agents enough time to visit each maze. If an agent clears the fourth maze, evaluation ends. Each network was evaluated 10 times to account for the noisiness of the domain, and average scores were used as fitness. Ideally, more evaluations would be conducted, but 10 is a trade off between accuracy and overall run time. Each population was evolved for 200 generations. For each set of conditions, there were 20 distinct evolutionary runs.

VI. RESULTS

This section describes the results of evolution, the behaviors of champion networks, and results of the MS. PAC-MAN VS. GHOST TEAM COMPETITION at CIG 2018.

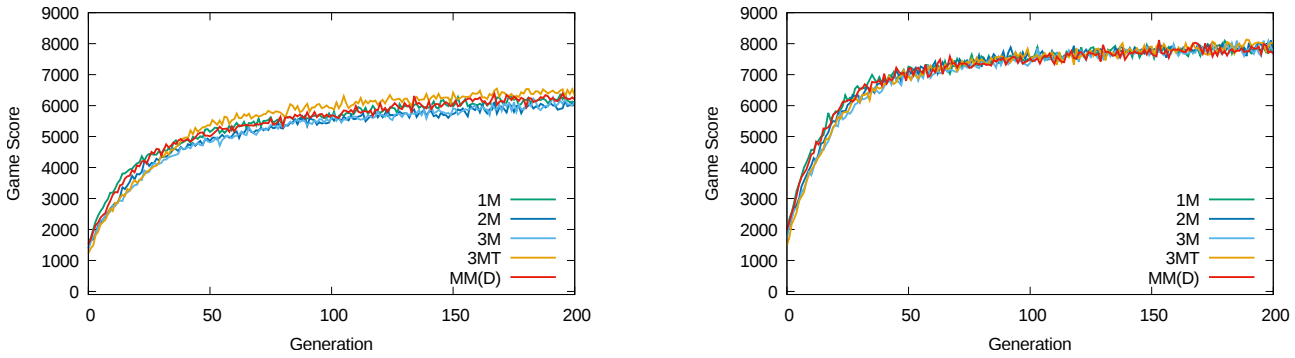
A. Evolution

Average scores across 20 runs of each modularity type are in Fig. 3. Against ghosts with full observability (Fig. 3a), performance growth slows after around 50 generations, and all approaches produced agents with average scores around 6000. This uniformity across different types of modularity is likely due to the use of split sensors instead of conflict sensors, which is consistent with previous work [1].

TABLE II

Directional Sensors in Ms. Pac-Man

Sensor Name	**Description**
Nearest Pill Distance	Distance to nearest regular pill in given direction
Nearest Power Pill Distance	Distance to nearest power pill in given direction
Nearest Junction Distance	Distance to nearest junction in given direction
Max Pills in 30 Steps	Number of pills on the 30-step path in the given direction that has the most pills
Max Junctions in 30 Steps	Number of junctions on the 30-step path in the given direction that has the most junctions
Options From Next Junction	Number of junctions reachable from the next nearest junction that Ms. Pac-Man is closer to than a threat ghost
n^{th} Nearest Potential Edible Ghost Distance	Distance to n^{th} nearest possible edible ghost in given direction
n^{th} Nearest Potential Threat Ghost Distance	Distance to n^{th} nearest possible threat ghost in given direction
n^{th} Nearest Potential Edible Ghost Probability	Likelihood that n^{th} nearest possible edible ghost in given direction is actually present
n^{th} Nearest Potential Threat Ghost Probability	Likelihood that n^{th} nearest possible threat ghost in given direction is actually present
n^{th} Nearest Potential Edible Ghost Approaching?	1 if n^{th} nearest possible edible ghost in given direction is approaching, 0 otherwise
n^{th} Nearest Potential Threat Ghost Approaching?	1 if n^{th} nearest possible threat ghost in given direction is approaching, 0 otherwise
n^{th} Nearest Potential Threat Ghost Path Has Junctions?	1 if directional path to n^{th} nearest possible threat ghost contains any junctions, 0 otherwise



(a) Ghosts with Full Observability

(b) Ghosts with Partial Observability

Fig. 3. **Average Ms. Pac-Man Champion Scores During Evolution Across 20 Runs:** These are the average scores across champion agents evolving for 200 generations. All methods perform similarly due to the use of split sensors. (a) Evolving agents against ghosts with full observability leads to lower average scores around 6000. This is because ghosts have an unfair advantage over Ms. Pac-Man agents. (b) Evolving agents against ghosts with partial observability leads to higher average scores around 8000.

Against ghosts with partial observability (Fig. 3b), performance growth slows around 30 generations. Against the partially observable opponents, all modularity approaches have average scores around 8000. The better scores against the partial observability ghosts make sense, given that these ghosts are handicapped in a way similar to Ms. Pac-Man.

B. Final Behavior

Qualitatively, the agents demonstrated behaviors necessary for high performance in Ms. Pac-Man. All evolved agents know to hunt for pills and flee threat ghosts. When Ms. Pac-Man sees edible ghosts, she generally pursues them. However, despite uniform performance during evolution, each modularity approach exhibits different performance in post-evolution evaluations. Specifically, each of the 20 champions for each modularity type evolved against each ghost type (full or partial observability) was evaluated an additional 100 times each against both types of ghost. For each champion, an average, minimum, and maximum score across the 100 evaluations was produced, and each of these 20 values were averaged to summarize results. The standard error of the average of the average scores is also provided.

Post evaluations are a more reliable measure of performance than 10 evaluations during evolution. In fact, since plots of champion scores (Fig. 3) show the score of the best agent out of 100, they often overestimate actual champion performance. Tables III and IV show post evaluation results against partial observability (PO) and full observability (FO) ghosts respectively, using the same settings from evolution: Evaluations end after 8000 ticks or four levels, whichever comes first. In practice, agents that do not run out of lives will clear the fourth level before reaching the time limit. Post evaluations were also conducted with the same settings as the MS. PAC-MAN VS. GHOST TEAM COMPETITION: No level limit, but a time limit of 4000 ticks. This short time limit means evolved agents often run out of time in the third maze, and never clear all four. These results are in Tables V and VI against partial and full observability ghosts respectively.

In post evaluations, whether agents were evolved against partial or full observability ghosts, 3MT champions have the highest average, maximum, and minimum scores. The difference in average scores between 3MT and other methods is statistically significant in each case according to pairwise Student's t-tests (adjusted $p < 0.001$ with Bonferroni error correction). This result is surprising given that 3MT had average scores comparable to other methods during evolution. In fact, the same statistical tests show that none of the other methods are significantly different from each other. However, 3MT more consistently pursues edible ghosts (Fig. 1) and flees threats, due to specific modules for each case.

TABLE III

Champion Evaluations Against Partial Observability Ghosts With Level Limit of Four, Time Limit of 8000

Opp	Method	Avg ± Std Err	Min	Max
PO	1M	5093.8 ± 514.47	300.0	17193.5
PO	2M	5068.2 ± 420.89	231.5	16518.5
PO	3M	4766.0 ± 517.86	181.4	16214.5
PO	**3MT**	$\mathbf{10347.0 \pm 394.80}$	**929.5**	**19545.0**
PO	MM(D)	5019.1 ± 642.95	222.0	16077.5
FO	1M	2552.4 ± 582.46	306.5	9682.0
FO	2M	2765.9 ± 564.71	174.0	9981.5
FO	3M	2839.8 ± 584.42	249.5	11556.5
FO	**3MT**	$\mathbf{6480.1 \pm 568.09}$	**464.5**	**17054.5**
FO	MM(D)	2752.1 ± 441.27	178.0	11317.5

TABLE IV

Champion Evaluations Against Full Observability Ghosts With Level Limit of Four, Time Limit of 8000

Opp	Method	Avg ± Std Err	Min	Max
PO	1M	1147.1 ± 121.94	100.0	6349.5
PO	2M	1145.6 ± 111.22	99.0	6523.5
PO	3M	1231.5 ± 157.55	115.0	7284.5
PO	**3MT**	$\mathbf{3219.1 \pm 160.06}$	**204.0**	**11937.5**
PO	MM(D)	1097.6 ± 134.48	102.0	6936.0
FO	1M	937.6 ± 167.63	96.0	6432.5
FO	2M	949.0 ± 187.81	95.0	5554.0
FO	3M	1035.1 ± 240.71	102.5	6666.5
FO	**3MT**	$\mathbf{2344.3 \pm 274.08}$	**173.5**	**10121.5**
FO	MM(D)	911.3 ± 134.34	75.0	7223.5

TABLE V

Champion Evaluations Against Partial Observability Ghosts With Competition Time Limit of 4000 Steps

Opp	Method	Avg ± Std Err	Min	Max
PO	1M	4085.3 ± 401.60	216.5	13568.5
PO	2M	3720.1 ± 310.01	232.5	13651.0
PO	3M	4213.9 ± 405.31	208.0	13600.0
PO	**3MT**	$\mathbf{7912.8 \pm 252.76}$	**830.5**	**15127.0**
PO	MM(D)	3808.9 ± 505.02	202.0	12746.0
FO	1M	2063.0 ± 426.69	228.0	9236.5
FO	2M	2244.2 ± 455.23	197.5	8868.0
FO	3M	2346.0 ± 468.13	272.0	8807.0
FO	**3MT**	$\mathbf{4900.5 \pm 425.45}$	**393.0**	**13412.0**
FO	MM(D)	2351.6 ± 342.45	176.5	10534.0

TABLE VI

Champion Evaluations Against Full Observability Ghosts With Competition Time Limit of 4000 Steps

Opp	Method	Avg ± Std Err	Min	Max
PO	1M	917.3 ± 97.92	99.5	5814.5
PO	2M	947.1 ± 97.62	84.0	6028.5
PO	3M	998.3 ± 113.75	82.0	6366.0
PO	**3MT**	$\mathbf{2765.3 \pm 151.10}$	**145.0**	**10602.0**
PO	MM(D)	924.3 ± 96.98	97.5	6243.0
FO	1M	851.1 ± 171.06	89.0	5656.5
FO	2M	821.4 ± 163.47	83.0	5499.0
FO	3M	842.3 ± 147.32	103.0	5895.5
FO	**3MT**	$\mathbf{2147.4 \pm 250.80}$	**131.5**	**9921.5**
FO	MM(D)	680.5 ± 129.98	88.5	5778.0

One behavior not exhibited by any champion was luring ghosts to power pills, as witnessed in the fully observable game [1]. Luring is the clustering of ghosts near a power pill so that Ms. Pac-Man can easily eat the ghosts in succession, leading to higher scores. This behavior is hard to develop without full observability, so its lack is not surprising.

Preference neuron approaches tend to focus on one module and ignore others. Even MM(D) champions tend to have only one module, though lesser members of each population evolve additional modules. However, because of the use of split sensors, agents using one module still develop distinct reactions to threat and edible ghosts. For example, a sensor for proximity to a threat ghost can decrease an agent's preference for a direction, but proximity to an edible ghost can increase the preference for that same direction. However, the interplay of sensors is complicated, and ghosts of multiple types can be present, so these approaches seem to make errors in judgment more often. These agents sometimes jitter in uncertainty and ultimately get captured when near a threat ghost. This outcome demonstrates a failure of evolved preference-based task divisions in this particular domain.

In contrast, 3MT must use different modules in different situations, and forcing this task division seems to cause more risk-averse behavior and better response to threat ghosts in general. Being risk-averse seems to be especially useful in this very unpredictable partially observable domain. Videos of agent behavior of each type are available online[3]. Since 3MT behavior is more robust it was entered into the competition at CIG 2018.

TABLE VII

Results From Ms. Pac-Man Track of Ms. Pac-Man Vs. Ghost Team Competition at CIG 2018

Entry (3MT agent)	`Squillyprice01`	7736.63
Entry	`GiangCao`	7516.6
Entry	`thunder`	6733.1
Entry	`PacMaas`	6275.0
Included with code	`Starter PacMan`	5865.5
Included with code	`StarterPacManOneJunction`	1134.3
Included with code	`StarterNNPacMan`	535.0
Entry	`user76`	120.0

C. CIG 2018 Competition Results

In the MS. PAC-MAN VS. GHOST TEAM COMPETITION at CIG 2018, each Ms. Pac-Man entry played 20 games against each of the four ghost entries submitted to the competition, for a total of 80 games per entrant in the Ms. Pac-Man track. Games ended after 4000 time steps, or when Ms. Pac-Man ran out of lives. The average score across all 80 games formed the agent's overall score for the competition.

Of the four ghost team agents competing, two were provided by the competition organizers. There were eight Ms. Pac-Man agents, three of which were provided by the competition. A 3MT agent, as described in this paper, won the Ms. Pac-Man track with an average score of 7736.63 points across 80 games (Table VII). The source code[4] includes evolved champion networks for all modularity types, and an R script for the statistical analysis reported earlier.

The agent was entered under the name `Squillyprice01`. Scores were close between `GiangCao` and `Squillyprice01`. Entrants `thunder` and `PacMaas` were close in performance. `Starter PacMan` was the last entrant to produce scores that were even slightly comparable to the head of the pack.

[3] `southwestern.edu/~schrum2/SCOPE/popacman.php`
[4] `https://github.com/schrum2/MsPacManEntry`

VII. DISCUSSION AND FUTURE WORK

Different modular approaches all achieve similar levels of performance during evolution. This result is consistent with previous research [1], in which split sensors were shown to provide parallel pathways through the network, making it easy to associate distinct modes of behavior with different sensors, even when using only one module. However, for preference-based approaches to focus on only one module may be a local optimum, made all the more appealing by the limited information gleaned from just 10 noisy evaluations.

However, the high performance of 3MT in post-evolution evaluations proves that distinct output modules are useful. This multitask approach outperforms all other forms of modularity, which contrasts with previous champion evaluations in the fully observable game [1], where some champions using preference neurons earned exceptionally high scores. This seemingly contradictory result seems to depend on the robustness of agent behavior in the highly unpredictable partially observable domain. The 10 evaluations that agents experience during evolution is not much, so over a population of 100 there will be some individuals that get lucky 10 times in a row, which inflates champion fitness scores. The more thorough post-evolution scores tend to be lower because being lucky 100 times in a row is improbable.

Unpredictability may also explain why agents evolved against full observability ghosts perform worse than agents evolved against partial observability ghosts, even when evaluated against full observability ghosts. One would expect agents evolved against full observability ghosts to learn to handle them better. However, it seems that due to the unpredictability of partial observability ghosts, agents evolved against them learn more conservative and robust policies. In fact, observation of the videos of successful 3MT behavior shows that the agent is timid in the presence of ghosts, and avoids pills that seem safe to eat from the perspective of a human observer, which further explains why its post-evolution scores are higher than those of other methods.

The 3MT agent that won the competition was evolved against partial observability ghosts. Table VII shows its average competition score, but results from individual games or between specific pairings were not released. The source code of all ghost teams is also not available. Therefore, a detailed analysis of 3MT's strengths and weaknesses against specific opponents is not available at this time.

Little attention has been given to creating ghost teams, with a few exceptions [8], [11]. Multitask networks could produce high performing ghost agents. The pill model would have to be updated as the ghosts are not immediately aware of what pills Ms. Pac-Man has eaten. The techniques used to create the ghost model could be adapted to keep track of Ms. Pac-Man, though her movements are not as restricted, and thus harder to predict. With a working ghost team, it would be possible to competitively coevolve Ms. Pac-Man and ghost team agents. Ghost teams could also evolve against pre-built Ms. Pac-Man agents provided by Williams et. al. [4], or against champion agents evolved in this paper.

VIII. CONCLUSIONS

This paper compared five different modular neural network architectures in the partial observability version of Ms. Pac-Man: standard one module networks (1M), two and three module networks with preference neurons (2M, 3M), three module multitask networks (3MT), and networks subject to Module Mutation Duplicate (MM(D)). Evolution performance was almost identical across all methods due to the use of split sensors. However, 3MT proved more robust across large numbers of evaluations. Thus, the 3MT architecture was used to produce an agent that won first place in the MS. PAC-MAN VS. GHOST TEAM COMPETITION at CIG 2018. These methods dealt with partial observability using models of the ghosts and pills. This victory shows that a neuroevolution approach that was successful in the fully observable version of the game also works under partially observable conditions, if provided with improved sensors that take advantage of models of where unseen entities might be.

ACKNOWLEDGMENTS

This research is supported in part by the Summer Collaborative Opportunities and Experiences (SCOPE) program, funded by various donors to Southwestern University.

REFERENCES

[1] J. Schrum and R. Miikkulainen, "Discovering Multimodal Behavior in Ms. Pac-Man through Evolution of Modular Neural Networks," *IEEE Transactions on Computational Intelligence and AI in Games*, 2016.

[2] M. F. Brandstetter and S. Ahmadi, "Reactive Control of Ms. Pac Man Using Information Retrieval Based on Genetic Programming," in *Computational Intelligence and Games*. IEEE, 2012.

[3] S. M. Lucas, "Evolving a Neural Network Location Evaluator to Play Ms. Pac-Man," in *Computational Intelligence and Games*. IEEE, 2005.

[4] P. R. Williams, D. Perez-Liebana, and S. M. Lucas, "Ms. Pac-Man Versus Ghost Team CIG 2016 Competition," in *Computational Intelligence and Games*. IEEE, 2016.

[5] P. Rohlfshagen, J. Liu, D. Perez-Liebana, and S. M. Lucas, "Pac-Man Conquers Academia: Two Decades of Research Using a Classic Arcade Game," *IEEE Transactions on Games*, 2018.

[6] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[7] A. M. Alhejali and S. M. Lucas, "Using a Training Camp with Genetic Programming to Evolve Ms Pac-Man Agents," in *Computational Intelligence and Games*. IEEE, 2011.

[8] A. B. Cardona, J. Togelius, and M. J. Nelson, "Competitive Coevolution in Ms. Pac-Man," in *Congress on Evolutionary Computation*. IEEE, 2013.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing Atari with Deep Reinforcement Learning," in *NIPS Deep Learning Workshop*, 2013.

[10] H. van Seijen, M. Fatemi, R. Laroche, J. Romoff, T. Barnes, and J. Tsang, "Hybrid Reward Architecture for Reinforcement Learning," in *Neural Information Processing Systems*, 2017.

[11] A. Dockhorn and R. Kruse, "Combining Cooperative and Adversarial Coevolution in the Context of Pac-Man," in *Computational Intelligence and Games*. IEEE, 2017.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, 2002.

[13] K. O. Stanley and R. Miikkulainen, "Evolving Neural Networks Through Augmenting Topologies," *Evolutionary Computation*, 2002.

[14] S. Risi and J. Togelius, "Neuroevolution in Games: State of the Art and Open Challenges," *IEEE Transactions on Computational Intelligence and AI in Games*, 2017.