Competitive Coevolution in Ms. Pac-Man

Andrew Borg Cardona, Julian Togelius and Mark J. Nelson

***Abstract*—In this paper we investigate the suitability of the arcade game Ms. Pac-Man, as implemented in the recent Pac-Man versus Ghost Teams Competition, as a testbed for competitive coevolution. To that end, we explore competitive co-evolution techniques to co-evolve Pac-Man and Ghosts team controllers. We analyze in some detail the dynamics of evolution between the two classes and compare them with single-objective evolution and static controllers. We note differences between evolutions of the two classes, having observed higher fitness transitivity in Pac-Man than in the Ghosts. The problem of finding a well-performing general purpose Pac-Man is far different than that of finding a good and general Ghosts controller.**

Keywords: Coevolution, games

I. Introduction

Competitive coevolution is one of those methods within computational intelligence which seems to hold immense promise but has only a small number of practical successes, and where it's not quite clear what's ailing the approach. The basic idea in competitive coevolution is that you evolve not against a static fitness function, but against another population that uses the first population as a fitness function. The goal is to set off an *arms race*, where the two populations constantly compete to best one another, leading to truly open-ended evolution. The inspiration comes from biology, where e.g. foxes are locked in an arms race with rabbits: foxes develop better prey detection, sneakier prowling and faster running, and rabbits develop better predator detection, more confusing escape strategies and faster running. However, when trying to implement competitive coevolution in computer programs, researchers frequently run into a number of show-stopping problems, such as *cycling* and *loss of gradient*. Faced with these problems, most researchers turn to the algorithm and try to construct a better competitive coevolution algorithm that will not have the same weaknesses. However, one could also turn to the problem domain, and ask whether it has the qualities that would allow for competitive coevolution to work and an arms race to arise; and if not, one could investigate what modifications to the problem domain would increase the chances of seeing this happen. Perhaps due to the focus on algorithm invention in computational intelligence, this approach to the problems of competitive coevolution is currently underrepresented in the literature. In this paper we investigate the classic arcade game Ms. Pac-Man, as implemented in the popular Pac-Man versus Ghosts competition, as a testbed for competitive coevolution and perform experiments in coevolving Pac-Men and Ghosts.

The authors are with the Center for Computer Games Research at the IT University of Copenhagen, Rued Langgaards Vej 7, 2300 Copenhagen, Denmark. email: {aboc,juto,mjas}@itu.dk

A. Competitive coevolution

Competitive coevolution leading to arms races in biology was described by Dawkins and Krebs [1]. It is unclear where the idea to implement this kind of evolution *in silico* first originated, but probably the most prominent early success was Hillis' coevolution of sorting networks and testing problems, leading to much improved performance over simply evolving sorting networks against static fitness functions or even hand-crafting them [2].

An important paper by Rosin and Belew identified some pathologies of competitive coevolution and suggested several algorithmic enhancements [3] to overcome them. Of the pathologies, two stand out. *Loss of gradient* happens when one population becomes stuck in a minimum where all individuals in the population perform equally badly, possibly due to the other population developing individuals that convincingly beat all the individuals in the first population. *Cycling* refers to the phenomenon where the best individual of some generation beats its competitor of the same generation, but not its competitor of an earlier generation, due to intransitivity of the fitness function. (The phenomenon can be understood by reference to the game *rock, paper, scissors*: population A evolves rock, which makes population B evolve paper, so population A evolves scissors, and population B evolves rock...) The proposed solutions include various versions of the concept of an *archive* or *hall of fame*, where current individuals are compared against historical individuals; or *fitness sharing*, where an individual's fitness depends not only on how well it does against particular individuals of the opposing population but also on how "unique" these individuals are.

Several authors have applied competitive coevolution to predator–prey domains, coevolving one population of "predators" and one of "prey". Nolfi and Floreano coevolved such behaviors in physical robots, and found some evidence of arms races but also plenty of cycling [4]. Stanley and Miikkulainen evolved predators and prey in a simulated environment, using the NEAT method for evolving neural network-based controllers [5]. Results were encouraging but there seemed to be an upper limit to the complexity of strategies that were practically possible to evolve.

While the predator–prey paradigm involves asymmetric coevolution, with individuals in the two populations having markedly different abilities and affordances, others have investigated different paradigms. Togelius and Lucas investigated symmetric coevolution between racing car controllers on a continuum between relative and absolute fitness [6]. In further experiments using the car racing domain, Togelius et al. investigated the effect of using multiple populations [7].

As evolution can be an effective way of doing (phylogenetic) reinforcement learning [8], competitive coevolution is closely connected to the *self-play* training paradigm in ontogenetic RL, where agents are trained by playing against themselves. An early success was TD-Gammon [9], which learned to play backgammon by playing against itself, using a temporal-difference reinforcement-learning algorithm training a neural network through backpropagation, starting from a randomly initialized policy. Success in this domain appears to have depended strongly on features of the backgammon domain in particular, however. A few years later, Pollack and Blair [10] showed that a quite simple (perhaps even "naive") competitive coevolution approach, previously ignored due to its relatively weak performance on most problems, did surprisingly well at backgammon. Runarsson and Lucas [11] further clarified experimentally that the diversity of opponents in self-play is a key feature for the paradigm to be successful. They found that in games with less built-in stochasticity than backgammon, evolutionary computation methods that explicitly maintain opponent diversity via populations outperform agents trained via self-play using reinforcement-learning methods. Such results reinforce the importance of choosing the right opponent to evaluate against in competitive coevolution.

B. Competitive coevolution testbeds

As discussed above, the properties of the problem domain could well be at least as important as the properties of the algorithm for successful coevolution. What, then, makes for a good testbed for competitive coevolution? We suggest the following properties.

- *Depth of challenge*. The task should not be trivial. It is possible that the meager progress beyond the initial stage in some previous predator–prey experiments was due to there simply not being any substantially better strategies for simple tasks.
- *Smoothness of challenge*. It needs to be possible to develop from a trivial strategy to a very good strategy in small increments. The evolved 3D simulated creature representation used by Sims [12], and the similar representation later used by Miconi [13], for "box grabbing" seem problematic from this perspective. While it was apparently relatively easy to evolve creatures performing a simple locomotion behavior to lunge forward towards a box, the step towards more purposeful tactile manipulation seemed unattainable for evolution.
- *Fitness transitivity*. The better a strategy is against some other strategy, the better it should on average be against other strategies. In other words, performance of an agent against other agents should be correlated. This is exactly the feature that the game of rock-paper-scissors does not have.

C. Ms. Pac-Man

Ms. Pac-Man started out as an arcade game, developed by Midway and released by Namco in 1982[1]. It quickly became very successful, and was ported to numerous home computers and video game systems; modern adaptations are occasionally released, and it remains one of the most popular arcade games of all times.

The core game mechanics are simple. The player guides Ms. Pac-Man through a maze filled with pills and power pills. At any point, the player can choose to move up, down, right or left, but can never move through walls. A level is cleared when all pills have been eaten. Unfortunately, four ghosts are haunting the maze and will kill the protagonist upon contact. However, after eating a power pill, the tables are turned for a short time, during which Ms. Pac-Man will kill ghosts upon contact (they respawn at the centre of the level). As the game progresses the speed increases, the speed of the ghosts increases relative to Ms. Pac-Man's speed, and the length of the time a power pill works is diminished. The challenge of playing the game derives from the combination of quick reactions, route planning (optimising the route taken to eat all pills can be seen as an instance of the travelling salesperson problem) and of outsmarting the ghosts. In the arcade version of the game, the ghosts follow a simple yet somewhat unpredictable pattern, with the different ghosts having distinct "personalities".

The *Pac-Man versus Ghosts Competition* is an international game AI competition organized by Rohlfshagen and Lucas [2] [14]. The competition is based on a Java clone of Ms. Pac-Man, which features an API for controlling both Ms. Pac-Man and the ghost teams. The API provides a representation of the game state every 40 ms of simulated time (the game can be sped up to hundreds of times faster than real-time), and asks the controller for the next move to make. In 2011 and 2012, a large number of competitors submitted either Pac-Man controllers, ghost team controllers or both. Several of these submissions have been written up as academic papers. Notable controllers include Ruck Thawonmas' team's controller ICE Pambush, winner of the IEEE CEC 2009 Ms. Pac-Man Competition [15]. It was based on a set of pre-defined rules and on the idea of luring then ambushing ghosts. The controller has seen several updated and alternate versions later on, including the ICEP-feat-Spooks which incorporates and improves on the controller Spooks by Daryl Tose. The Spooks controller attempts to maximize the points from power pills and eating ghosts, ignoring the attempt to clear pills efficiently. The latest ICE Pambush version is the ICEP-IDDFS, which uses iterative deepening depth-first search in the ICEP-feat-Spooks. Pepels built a Monte-Carlo Tree Search Agent [16] with several enhancements to improve search and achieve good decisions within the time limit. Burrow and Lucas explored Temporal

[1]Ms. Pac-Man and its predecessor *Pac-Man* are similar, but there are several differences. For our purposes the most important difference is that Pac-Man is deterministic, whereas Ms. Pac-Man has random elements.

[2]http://pacman-vs-ghosts.net/

Difference Learning and evolution of MLPs (Multi Layer Perceptrons) [17] to play Ms. Pac-Man and achieved notably better results through evolution. Alhejali and Lucas [18] have also evolved Ms. Pac-Man Agents but using Genetic Programming; these controllers were shown to outperform hand-designed controllers.

D. Ms. Pac-Man as a competitive coevolution testbed

In [14] it is suggested that Ms. Pac-Man is an excellent domain for competitive coevolution. Indeed, the comparison to the predator-prey testbeds used by Nolfi and Floreano and by Stanley and Miikkulainen comes naturally, as Ms. Pac-Man is at its heart a predator-prey scenario. It seems obvious to coevolve a population of Pac-Man and a population of ghost teams. However, compared to the simple domains used by previous authors, it adds much complexity through restricting movement with a maze, adding the coordination challenge of the four ghosts, and the occasional role reversal through the power pills. Let's examine Ms. Pac-Man according to the criteria for a coevolution testbed discussed above.

- *Depth of challenge.* It seems clear that there is sufficient depth of challenge in Ms. Pac-Man, given that competitions are still taking place for the world championship among human players, and that no AI controller has been able to perform above intermediate human performance.
- *Smoothness of challenge.* For Ms. Pac-Man controllers, there would seem to be a very smooth transition from essentially random play (which still scores above 0) to world-class playing. The fitness metric, which is simply the score, has a fine granularity. However, for the ghost controllers there might be a risk of loss of gradient, as there might be little score differences between a bad ghost team controller and a less bad ghost team controller when playing against a high-performing Pac-Man controller.
- *Fitness transitivity.* It is very unclear to what extent good Pac-Man playing is tied to specific ghost strategies. The necessity of relatively domain-general skills such as path planning for good playing would seem to indicate a high degree of transitivity; however, it seems that some ghost team controllers submitted to the competition include highly specific behavior (such as "ambushing" Ms. Pac-Man) that works well against some Pac-Man controllers but not others.

E. Outline of the rest of the paper

In this paper we investigate the suitability of Ms. Pac-Man as a competitive coevolution testbed. First, we devise an evolvable controller architecture for both Pac-Man and Ghost teams with sufficient complexity to display somewhat sophisticated strategies. We then investigate the ability to evolve controllers using performance against two static controllers as a fitness function. The performance of those controllers are tested against both controllers they were evolved against (training set) and controllers they were not evolved against (testing set). In order to investigate fitness transitivity, performance against several static controllers are correlated. Next, we attempt competitive coevolution. Several different schemes are attempted in an exploratory study, and the most promising scheme is used for 10 evolutionary runs. The best evolved controllers from single- and coevolutionary runs are compared. At the conclusion of the paper, we discuss what we can learn from these results about both the problem and the method, and draw up some directions for future research.

II. Methods

A. Pac-Man controller architecture and representation

The Pac-Man controller implements a shallow MiniMax search in the utility function, searching possible moves until a junction in the maze. As an optimization to reduce branching factor, the search does not always consider the range of possible ghost moves that could take place in the interim. If the ghost is edible or far from Pac-Man, a move is assumed: towards Pac-Man if far away and the ghost is not edible, or away from Pac-Man if the ghost is edible. A full consideration of possible moves, however, is done when the ghost is near Pac-Man and not edible. The search is limited to a maximum of 60 game ticks, to avoid pathologically poor performance when Pac-Man is in an area far from any junction.

The utility function is a linear combination of the form

$$f(x) = \sum_{i=0}^{N} w_i d_i$$

where d represents a calculated distance or other value such as number of pills eaten, and w represents its weight. N is the total number of values taken from the current game state. There are a total number of 34 values used in the utility function, and thus 34 weights.

The genotype for Pac-Man directly represents the list of weights that are used in the utility function. Each weight can take a value from -1 to 1.

B. Ghost team controller architecture and representation

The controller for the team of ghosts is very similar to the Pac-Man controller; its utility function runs a MiniMax search that cuts off when either Pac-Man reaches a junction or the maximum of 60 game ticks is reached. There are two differences however. When a ghost must make a decision (in a junction), every possible move is considered, even if it is far from Pac-Man. This could potentially lead to large branching factors when all four ghosts require an action in the same tick, so the second difference is that in this special case (four ghost decisions simultaneously), all four ghosts are given random moves, adding a stochastic element to the controller.

The form of the utility function, and the genotype representation, are identical to those for the Pac-Man controller.

C. Static controllers

Various static controllers for both Pac-Man and the Ghosts team were used either for the single-objective evolutions or for testing and evaluating the evolution and co-evolution results. Below is a summary of these controllers:

- Starter Pac-Man / Starter Ghosts: These are 2 basic controllers included with the Ms. Pacman Vs Ghosts framework. They implement a very simple state machine to decide the next move.
- Aggressive Ghosts: A sample controller that simply hunts pacman, with occassional random moves.
- Legacy Ghosts: Another sample controller in which each ghost has its own behavior, and measures its next move towards Pac-Man based on different distance measures.
- Legacy2TheReckoning Ghosts: A sample controller that implements special conditions for dispersing or retreating from Pac-Man.
- Peterbb Ghosts: A simple yet effective Ghosts team controller that uses an evolve-able scoring system for guiding the moves. The scoring weights have been evolved through multi-objective evolution.
- Memetix Pac-Man / Memetix Ghosts: Two well-performing controllers by Daryl Tose. They are based on breadth-first search and use several pre-defined rules and scoring techniques to guide the moves.
- Mixmax Pac-Man / Starter_Ex: This is another league submission by Andrew Borg Cardona, implementing a variation of the Minimax algorithm. Starter_Ex pacman is a combination of the Starter Pac-Man and this Mix-Max Pac-Man, giving a challenge factor between the two.
- Tilman: This controller is a basic implementation of the MCTS within Pac-Man.
- ICEP_IDDFS Pac-Man: A very strong Pac-Man based on iterative deepening depth-first search, by Shirakawa, Nakamura and Thawonmas. As of this date, this controller ranks first in the Ms. Pac-Man vs Ghosts League.

D. Evolutionary algorithms employed

The weights for the utility function (defined on top of the MiniMax search) are evolved using a *(5+10)-ES* evolutionary algorithm, with Gaussian mutation applied to all weights (standard deviation of 0.2) and without any crossover operations. All algorithms are run for 50 generations.

For the single-objective evolutions, we use the cascading-elitism algorithm [19] to evolve a population of ghosts and a population of Pac-Man individually. A population of 15 ghosts is first evaluated against the Starter Pac-Man, and the best 10 are evaluated against the stronger MixMax Pac-Man. On the other hand, a population of 15 Pac-Man is evaluated against the Starter Ghosts, and the best 10 are evaluated against the Legacy Ghosts. The best 5 of each population are used for mutation and pass to the next generation.

For the competitive co-evolution experiments, we explore four different techniques and focus on one. The simplest method involves evaluating a population of one class against the elite of the competing class. Each evaluation consists of playing the full game 20 times, with 3 pacman lives and 16000 maximum game length. The individuals with the highest average are selected. The second method is a variation of this, but evaluates a population against the best three of the competing class (playing three times as many games). Selection is based on the sum of the average scores.

The remaining two methods make use of *fitness sharing* and *shared sampling*, as suggested by Rosin and Belew [3]. The first of these two methods encourages diversity by dividing the score of an individual against each competitor by the total score of all individuals against that competitor (three competitors are selected). We call this value the fitness ranking, denoted by R. The calculation is represented below:

$$R_x = \sum_j \frac{s_{xj}}{Sj}$$

where R_x is the ranking of inidividual x, j is one of the selected competitors, s_{xj} is the actual score of individual x against competitor j and S_j is the total score of all individuals against competitor j.

The last method takes the same concept of fitness sharing but excludes the individual's own score from the total S_j. The result is a further emphasis on diversity versus each individual's strength.

III. Results

For the sake of brevity, each approach is given a short name in the results. *Single* refers to the single-objective evolution; *Best* refers to the simple competitive co-evolution in which a population is tested against only one elite; *Top 3* involves evaluation against the top three elites; *Shared* uses fitness sharing and shared sampling; and *Shared Excl* uses the variation of fitness sharing which encourages more diversity.

A. Preliminary explorations

For our initial experiments, we ran the single-objective evolution once, and all the four co-evolution techniques once. The elite of each generation in each experiment was then evaluated 20 times against five static controllers (four in the case of Ghosts). The results are averaged to produce an overall performance index. Figure 1 displays the results for Pac-Man evolution, while Figure 2 displays the Ghosts team results.

In the case of the Pac-Man controllers, the single-objective evolution beats all other techniques, although in interpreting that result, it's worth noting that the controllers used as the target for single-objective evolution are also included in the performance index. The Top 3 co-evolution and the Shared Fitness Excl exhibit some erratic behavior, but result in fairly good scores, just outperforming the Best co-evolution. The Shared Fitness co-evolution displayed a much slower evolution progress, with the worst performance of the methods tested.

The results for the Ghosts team controllers are far less smooth. In this case, the Top 3 co-evolution outperforms

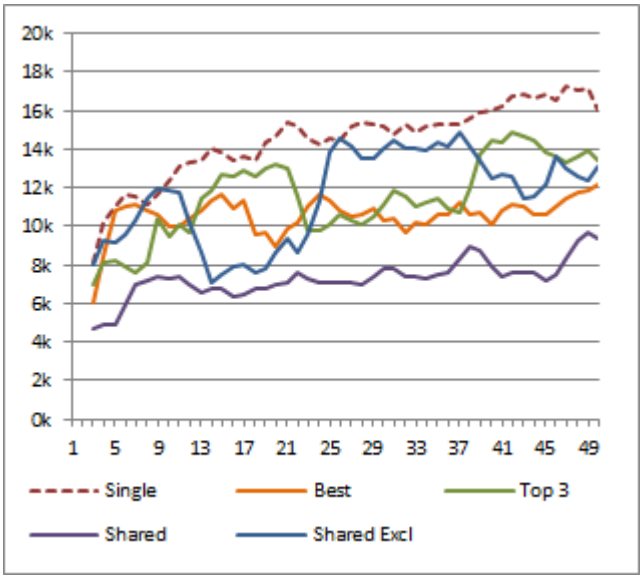

Fig. 1. The average score of the Pac-Man elite of each generation of all the five evolution techniques, played against five static Ghost team controllers. Data is smoothed over a rolling average of 3 points. The dotted line represents the single-objective evolution, while solid lines represent the competitive co-evolution. Higher scores represent better Pac-Man performance, i.e. better controllers.

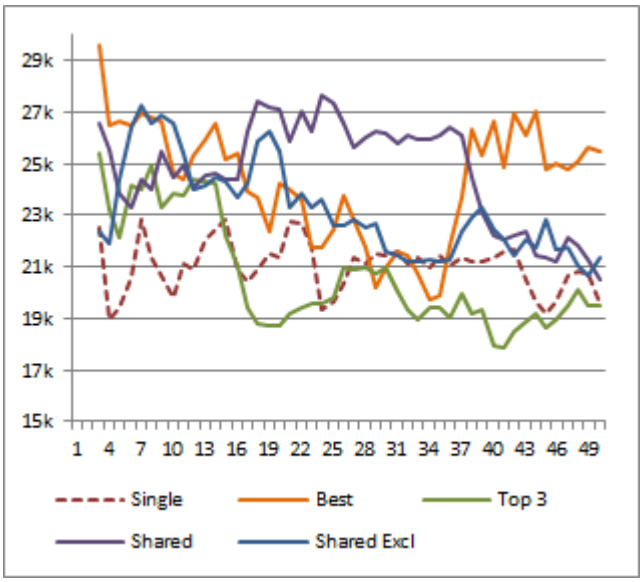

Fig. 2. The average score of the Ghosts elite of each generation of all the five evolution techniques played against four static Pac-Man controllers. Data is smoothed over a rolling average of 3 points. The dotted line represents the single-objective evolution, while solid lines represent the competitive co-evolution. Lower scores represent worse Pac-Man performance, i.e. better ghost controllers.

the single-objective evolution by a small margin towards the end. Unlike the case of Pac-Man controllers, the Shared co-evolution gave good results, along with the alternative Shared Excl evolution. The Best co-evolution progresses nicely, but suffers a large drop towards the last eight generations.

The preliminary explorations display various interesting behaviors that can be analyzed in further detail. We have first

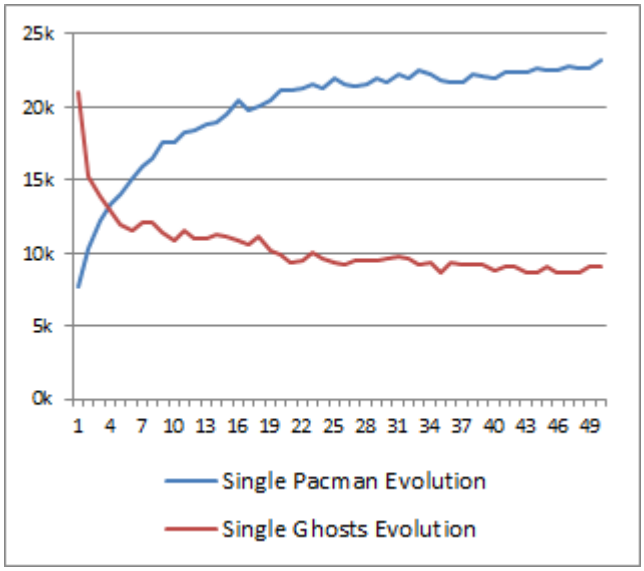

Fig. 3. Progress of the single-objective evolution for Pac-Man and for the Ghosts team. Pacman controller fights for a higher score, while the Ghosts team aim for decreasing the score.

taken the single-objective evolution and the simple Top 3 co-evolution techniques and run further detailed tests on these two.

B. *Single-objective evolution*

We ran the single-objective evolution 10 times for 50 generations and observed the evolution progress, illustrated in Figure 3. There is a very similar trend in both classes, with quick progress in the first 10 to 15 generations and stabilizing thereafter.

We then evaluated the elites of each generation against the five static Ghosts team controllers. The average scores against each competitor are depicted in Figure 4 for Pac-Man and in Figure 5 for the Ghosts team.

The Pac-Man controller produced in this experiment performs well against Starter and Legacy2, which were involved in the evolution process itself. It also performs well against Legacy, but its strategy does not vary greatly from that of Legacy2. Finally, Peterbb and Memetix controllers are very strong controllers, and although the scores remain quite low throughout, there is still steady, though limited, progress.

The Ghosts team easily kept the scores low for the Starter and Mixmax Pac-Man controllers, which the Ghosts were evolved against in this experiment. However the curve stabilizes at a very early stage. When faced with the tougher controllers, there was small but visible progress against ICEP_IDDFS, but hardly any against Memetix.

Table I shows a high correlation between the performance of the Starter, Legacy and Legacy2 controllers, as expected. There is a moderate correlation to the Peterbb and a very low correlation between any competitor and Memetix. In the case of the Ghosts, the correlations displayed in Table II are significantly smaller, except for the case between Memetix and ICEP_IDDFS competitors.

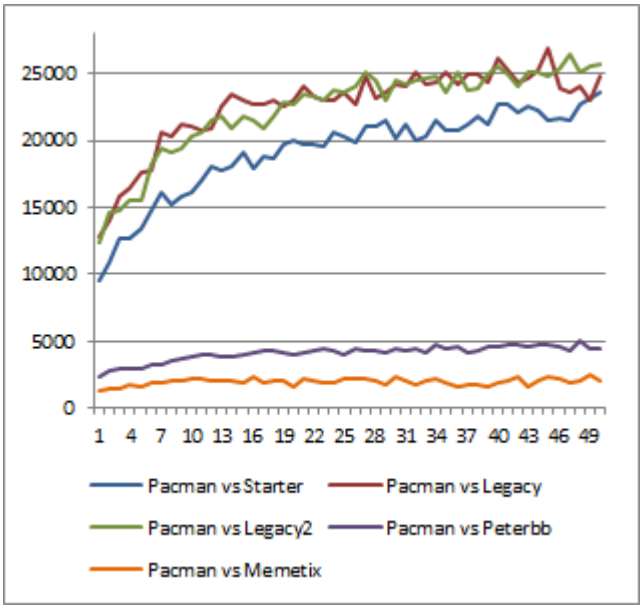

Fig. 4. The average scores of each Pac-Man elite in each generation of 10 single-objective evolution runs.

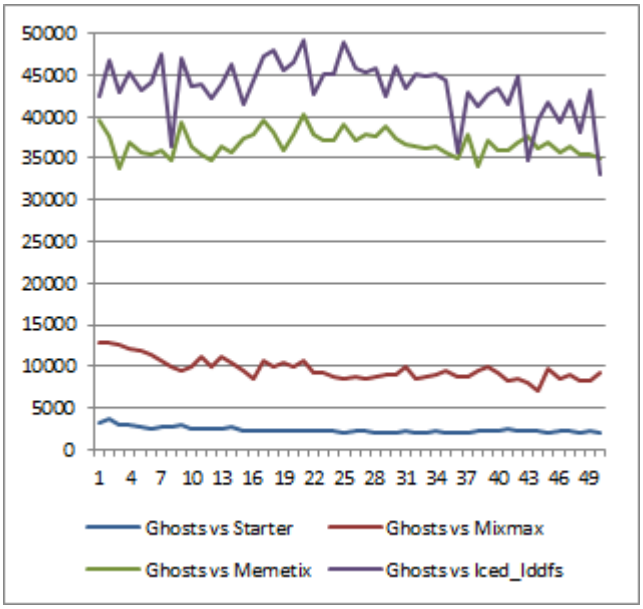

Fig. 5. The average scores of each Ghosts elite in each generation of 10 single-objective evolution runs.

	Starter	Legacy	Legacy2	Peterbb
Legacy	0.686	-	-	-
Legacy2	0.735	0.680	-	-
Peterbb	0.593	0.606	0.607	-
Memetix	0.2	0.2	0.197	0.23

TABLE I
THE POPULATION CORRELATION COEFFICIENTS OF THE SINGLE EVOLVED PACMAN VERSUS THE STATIC GHOSTS CONTROLLERS.

	Starter	Mixmax	Memetix
Mixmax	0.272	-	-
Memetix	-0.001	0.042	-
ICEP_IDDFS	0.0006	0.094	0.676

TABLE II
THE POPULATION CORRELATION COEFFICIENTS OF THE SINGLE EVOLVED GHOSTS VERSUS THE STATIC PAC-MAN CONTROLLERS.

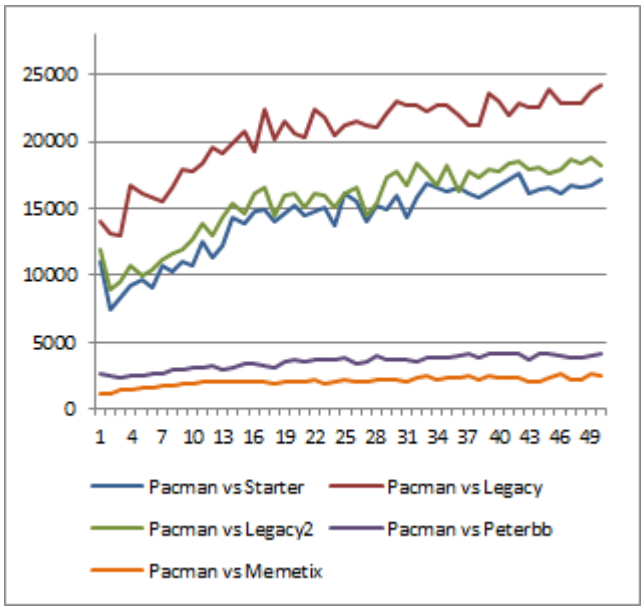

Fig. 6. The average scores of each Pac-Man elite in each generation of 10 *Top 3* competitive coevolution runs.

C. Competitive coevolution

In this second detailed experiment, we focused on the competitive co-evolution method that evaluates a population of one class against the top three elites of the competing class. We ran this method 10 times, and evaluated the elites of each generation against the static controllers. The results for Pac-Man are displayed in Figure 6, while those for the Ghosts team are displayed in Figure 7.

The overall performance against the Starter, Legacy, and Legacy2 controllers is quite good, although it fails to outperform the single-objective evolution. The same difficulties are however encountered against Peterbb and Memetix.

Interestingly, the co-evolved Ghosts progress smoothly against the Starter and Mixmax Pac-Man controllers, although they do not outperform the single-evolved Ghosts in these cases. However, co-evolution beats single-objective evolution against the Memetix and ICEP_IDDFS controllers, despite once again showing little progress in the case of Memetix.

The correlation coefficients for the performance of Pac-Man controllers in Table III are generally higher than those in Table I, particularly between any variable and Memetix. In Table IV we again see a moderately high correlation between the tough controllers Memetix and ICEP_IDDFS, as was also seen in the single-objective evolution. Interestingly we see more anti-correlations as well.

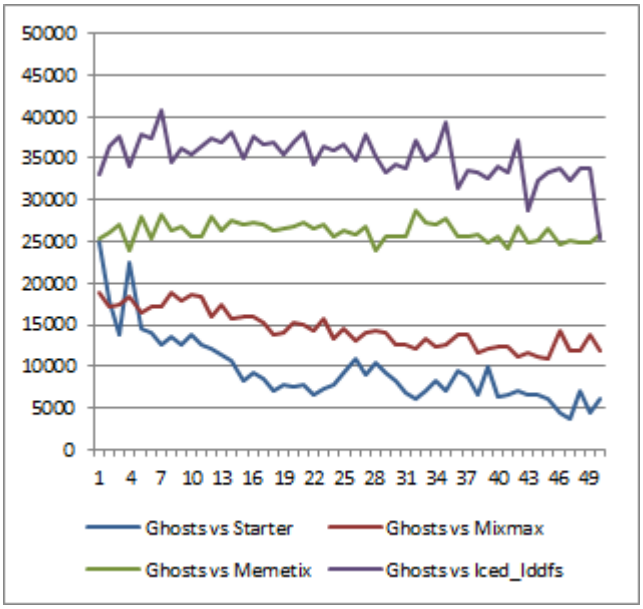

Fig. 7. The average scores of each Ghosts elite in each generation of 10 *Top 3* competitive coevolution runs.

	Starter	Legacy	Legacy2	Peterbb
Legacy	0.762	-	-	-
Legacy2	0.803	0.711	-	-
Peterbb	0.59	0.613	0.555	-
Memetix	0.339	0.403	0.32	0.475

TABLE III
THE POPULATION CORRELATION COEFFICIENTS OF THE COMPETITIVELY EVOLVED PACMAN VERSUS THE STATIC GHOSTS CONTROLLERS.

Finally, we tested the final elite of each evolution technique against each other. In Table V, S_p represents the single evolved Pac-Man, C_p represents the co-evolved Pac-Man, S_g is the single evolved Ghosts team and C_g is the co-evolved Ghosts team. The co-evolved controllers successfully outperform the single evolved controllers.

IV. DISCUSSION

A few high-level patterns in our results are clear. From our investigation, the Ms. Pac-Man problem has rather different characteristics for Pac-Man controllers and for ghost team controllers. Overall, it seemed much easier to evolve good controllers, both in a coevolutionary and a single-evolutionary mode, for Pac-Man than for ghost teams. Another somewhat puzzling difference is that the single-objective approach performed better than coevolution for Pac-Man, whereas the opposite was true for ghost teams.

	Starter	Mixmax	Memetix
Mixmax	0.268	-	-
Memetix	-0.312	-0.23	-
ICEP_IDDFS	-0.189	0.064	0.657

TABLE IV
THE POPULATION CORRELATION COEFFICIENTS OF THE COMPETITIVELY EVOLVED GHOSTS VERSUS THE STATIC PAC-MAN CONTROLLERS.

S_p vs S_g	C_p vs S_g	S_p vs C_g	C_p vs C_g
6228.7	10883.7	4775.5	6753.1

TABLE V
THE AVERAGE RESULTS OF RUNNING THE LAST ELITE IN EACH EVOLUTION TECHNIQUE AGAINST ITS COMPETING EVOLVED CONTROLLERS.

In order to assess fitness transitivity, we correlated the performance of controllers against several static controllers. Remarkably, these correlations were always strongly positive for Pac-Man, but smaller and occasionally negative for the ghost teams. This effect is clear regardless of whether controllers were trained with coevolution or single-objective evolution. A conclusion that can be inferred from these results is that fitness transitivity is higher for Pac-Man than for ghost teams. It is not clear why that would be the case. It could be that all ghost team strategies at this performance level are similar, whereas Pac-Man strategies at this level are different, with some clearly superior. Or, it could reflect the possibility that ghost team strategies are inherently more brittle. In any case, the discrepancy is a problem that needs to be investigated further, as it is likely to impede further coevolutionary experiments.

Overall, single-evolved controllers performed better than coevolved controllers. However, coevolved ghost teams performed better than single-evolved ghost teams when played against controllers not used in evolution, signifying a slight increase in generalisability.

All experiments in this paper used the same number of fitness evaluations. However, coevolution seems to have a slower evolution progress than single-objective evolution. That would mean that running evolution longer (for instance, 100 generations) might give additional benefit to co-evolution compared to using the same number of generations for all methods. Single-objective evolution sees little progress after around 30 generations (the curve stabilizes), and it is not likely to benefit much from running more evaluations in the same way.

One aspect that has not been discussed much is the role of the controller architecture and representation. It is possible that the current architecture is more suited to control Ms. Pac-Man than to control ghost teams, and that the fitness landscape it induces makes it hard to find good ghost-team controllers.

With this work, we hope to have laid groundwork for further, more systematic study of the possibilities of co-evolution in this domain, by investigating the properties of Ms. Pac-Man's Ghosts vs. Pac-Man domain as a testbed for competitive coevolution.

V. ONGOING AND FUTURE WORK

Looking at the somewhat surprising results, particularly for the ghosts, we created, evolved and tested a new controller architecture. Instead of using a linear combination in the utility function of the MiniMax, a Multi Layer Perceptron (MLP) was implemented to introduce non-linearity. Due to the resulting increase of dimensions, the distance measures

were simplified and the CMA-ES algorithm was used with all evolution and co-evolution types.

The elites of each evolution type was tested against static controllers in a tournament style against elites of the linear controller. In the case of Pac-Man, the MLP has performed consistently better than its linear counterpart. In the case of the ghosts however, the MLP seemed to deteriorate the performance for all evolutions.

In another experiment, we took the genotypes of the elites of each generation for the single-objective evolution of Pac-Man - one that showed a steady evolution progress - and inversed all the weights. These inversed weights were tested as Ghost controllers. Interestingly, the resulting performance was a very poor one, with no sign of evolution progress.

Even though the distance measures used for both Pac-Man and the Ghosts were identical, the problem of finding a set of weights to form a good strategy seems to difer significantly between the two. Several reasons might be behind this, which might be worth exploring further. One such reason is the game logic itself. Pac-Man's decisions happen after each game-tick to choose any new direction of movement. The Ghosts on the other hand, besides consisting of controlling four agents rather than one, cannot take decisions unless in a junction. Furthermore, the stochastic element in the game directly effects the ghosts by reversing their direction. These might cause higher variances in the scores which result in ambiguities during evolution and could direct evolution in the wrong direction.

It would be interesting to explore how co-operative co-evolution of the Ghosts might help or overcome these problems, such that each Ghost is a self-controlled agent whose fitness is based on its performance along a set of other Ghosts.

We also plan to conduct a more detailed analysis of cycling in the coevolutionary process. Further on, we plan to investigate the use of competitive coevolution to create ensembles of controllers with noncorrelated playing styles, and to introduce evolution of mazes as a third population.

VI. Conclusion

This paper investigated Ms. Pac-Man as a testbed for competitive coevolution. A controller architecture was devised, and experiments were undertaken with both single-objective evolution and coevolution. It was found that competitive Pac-Man controllers could be evolved using both approaches, but that results were much more mixed for ghost team controllers. Fitness transitivity, i.e. how much a solution's performance over several other solutions correlated, was found to be much higher for Pac-Man than for ghost teams, presenting a potential problem for competitive coevolution. Some evidence was found that coevolved controllers generalised better than single-evolved controllers, leading to a modest-confidence conclusion that this domain provides an interesting opportunity to study the emergence of arms races. Overall, Ms. Pac-Man appears to be a promising testbed for competitive coevolution, with a number of exciting future research projects possible.

Acknowledgements

The first author was supported by the Strategic Educational Pathways Scholarship Scheme (Malta). The scholarship is part-financed by the European Union European Social Fund.

References

[1] R. Dawkins and J. R. Krebs, "Arms races between and within species," *Proceedings of the Royal Society of London B*, vol. 205, pp. 489–511, 1979.

[2] W. D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure," in *Proceedings of the ninth annual international conference of the Center for Nonlinear Studies on Self-organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks on Emergent computation*, 1990, pp. 228–234.

[3] C. Rosin and R. Belew, "New methods for competitive coevolution," *Evolutionary Computation*, vol. 5, no. 1, 1996.

[4] S. Nolfi and D. Floreano, "Coevolving predator and prey robots: Do "arms races" arise in artificial evolution?" *Artificial Life*, vol. 4, pp. 311–335, 1998.

[5] K. O. Stanley and R. Mikkulainen, "Competitive coevolution through evolutionary complexification," *Journal of Artificial Intelligence Resarch*, vol. 21, pp. 63–100, 2004.

[6] J. Togelius and S. M. Lucas, "Arms races and car races," in *Proceedings of Parallel Problem Solving from Nature*. Springer, 2006.

[7] J. Togelius, P. Burrow, and S. M. Lucas, "Multi-population competitive co-evolution of car racing controllers," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, 2007.

[8] J. Togelius, T. Schaul, D. Wierstra, C. Igel, F. Gomez, and J. Schmidhuber, "Ontogenetic and phylogenetic reinforcement learning," *Kuenstliche Intelligenz*, 2009.

[9] G. Tesauro, "Temporal difference learning and TD-gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[10] J. B. Pollack and A. D. Blair, "Co-evolution in the successful learning of backgammon strategy," *Machine Learning*, vol. 32, pp. 225–240, 1998.

[11] T. P. Runarsson and S. M. Lucas, "Co-evolution versus self-play temporal difference learning for acquiring position evaluation in small-board go," *IEEE Transactions on Evolutionary Computation*, pp. 628–640, 2005.

[12] K. Sims, "Evolving 3d morphology and behavior by competition." in *Proceedings of Artificial Life IV*, 1994.

[13] T. Miconi and A. Channon, "The n-strikes-out algorithm: A steady-state algorithm for coevolution," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2006, pp. 1639–1646.

[14] P. Rohlfshagen and S. M. Lucas, "Ms pac-man versus ghost team cec 2011 competition," in *Proceedings of the 2011 IEEE Congress on Evolutionary Computation*. IEEE, 2011, pp. 70–77.

[15] R. Thawonmas and H. Matsumoto, *Proc. Asia Simulation Conference 2009 (JSST 2009)*, 2009. [Online]. Available: http://www.ice.ci.ritsumei.ac.jp/ ruck/PAP/jsst09-matsumoto.pdf

[16] T. Pepels and M. H. M. Winands, "Enhancements for monte-carlo tree search in ms pac-man," in *CIG*. IEEE, 2012, pp. 265–272.

[17] P. Burrow and S. Lucas, "Evolution versus temporal difference learning for learning to play ms. pac-man," in *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, Sept., pp. 53–60.

[18] A. Alhejali and S. Lucas, "Evolving diverse ms. pac-man playing agents using genetic programming," in *Computational Intelligence (UKCI), 2010 UK Workshop on*, Sept., pp. 1–6.

[19] J. Togelius, R. De Nardi, and S. M. Lucas, "Towards automatic personalised content creation in racing games," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2007.