Coevolving Game-Playing Agents: Measuring Performance and Intransitivities

Spyridon Samothrakis, Simon M. Lucas, *Senior Member, IEEE*,
Thomas Philip Runarsson, *Senior Member, IEEE*, and David Robles

***Abstract*—Coevolution is a natural choice for learning in problem domains where one agent's behaviour is directly related to the behaviour of other agents. However, there is a known tendency for coevolution to produce mediocre solutions. One of the main reasons for this is cycling, caused by intransitivities among a set of players.**

In this paper we explore the link between coevolution and games and revisit some of the coevolutionary literature in a games and measurement context. We propose a set of measurements to identify cycling in a population and a new algorithm that tries to minimise cycling in strictly competitive (zero sum) games. We experimentally verify our approach by evolving weighted piece counter value functions to play Othello, a classic two player perfect information board game. Our method was able to find extremely strong value functions of this type.

***Index Terms*—Games, Coevolution, Transitivity Measurements**

I. Introduction

Artificial Coevolution [1] is a process where one agent improves relative to another set of agents, hopefully in a never-ending arms-race [2]. There are a number of reasons why one might prefer using coevolution rather than regular artificial evolution. The major one is that the environment adapts to the agent, making any effort to measure fitness using a snapshot of that environment futile. This is typical in multi-agent settings. For example, if one wanted to find an optimal tennis agent (player), a flawed way to evolve a strong player using a standard evolutionary algorithm would be to measure performance against an existing optimal agent. An obvious flaw with this is that such a strong agent may not exist (and if it did there would be no need to evolve one). Furthermore, even if such an agent did exist, care would be needed to adapt its strength to be roughly equal to the current strength of the evolving population, thereby providing maximum uncertainty on the outcome of each game, and hence maximum information on knowing the result. In competitive coevolution, one starts with a population of randomly generated individuals which can be created with little effort and zero expert knowledge. These then compete against each other with the best individuals becoming the parents for the next generation. In the ideal case, performance improves during the course of evolution and, given enough generations, a strong set of players is produced.

Unfortunately, due to a set of issues, the situation is almost never ideal in coevolutionary domains. The evolved agents are usually stuck in suboptimal/mediocre solutions or cycle in the solution space, never actually reaching a fully optimum strategy (e.g. see [3], [4]) due to intransitivities. Intransitivities arise when there are cycles within the "A beats B" relation on a set of players. This problem has been identified in a series of publications (see Section IV), and it is directly related to optimality - in other words when do we consider an agent optimal in a coevolutionary setting, so that we get a gradient towards that solution. The major contribution of this paper is a systematisation of methods for coevolving agents all in one framework and experimentally verifying some of the approaches in the game of Othello. We provide a novel quantitative approach to measuring the intransitivities inherent within a population of coevolving players.

For the Othello experiments we study the coevolution of weighted piece counters (WPCs) to operate as value functions in a 1-ply minimax player, as described in Section VI. The aim here is not to find Othello players that are strong in absolute terms, or even especially strong value functions. The best performing value functions for Othello are tabular value functions [5] or N-tuple systems [6] [7] (which are essentially equivalent to each other), but these involve learning thousands or tens of thousands of parameters. We use Othello as an interesting domain of study in which to measure performance and intransitivities in coevolution. The existence of previous work using the same [8] or similar [9] representations also enables direct comparisons with alternative approaches. We show that intransitivities in this domain are indeed significant, and offer a simple but effective solution[1]. We use a standard evolutionary algorithm, but measure fitness against a co-evolving archive of players. Using this approach we have evolved the strongest weighted piece counters yet published, when measured against a set of existing strong players.

The rest of this paper is organised as follows: Sections II,III,IV provide the necessary background in games, player quality measurements and coevolution, respectively. A simple

Spyridon Samothrakis, Simon M. Lucas and David Robles are with the School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, United Kingdom (email: ssamot@essex.ac.uk, sml@essex.ac.uk, darobl@essex.ac.uk).

Thomas Philip Runarsson is with the School of Engineering and Natural Sciences, University of Iceland, Iceland (email: tpr@hi.is).

Manuscript received February 3, 2012; revised May 11, 2012; accepted June 25, 2012.

[1]Note that intransitivities are something that game designers frequently aim for to make a game balanced. However, this is not the focus of the current paper; instead we show how to measure the intransitivities in a coevolving population and how to at least partially overcome the problems presented by them.

novel algorithm, together with some proposals for measuring cycling in coevolution are shown in Section V. In Section VI, we perform a number of experiments where we showcase some of the aspects of the ideas presented in the previous section. Finally, we provide some insights and conclude this work in Section VII.

II. Games

The definition of a game is somewhat overloaded. On one hand it is used colloquially and widely to express situations where one has "fun", a subjective evaluation on the part of the player. For example, single player computer "games" (Tetris, Mario, Pacman) fall within the scope of this definition, as well as multi-player board games and children's games. On the other hand, games can be defined as every interaction between agents where one's decisions depend directly on the decisions of others, i.e. games are multi-agent settings. In this paper we will concern ourselves with the crosscut of these two sets: addressing games that are fun (like Chess, Poker and Othello), where one player's actions impact the other player. Thus, games like Tetris (where there is only one player) and games like the ones studied under game theory (e.g. oligopolies) are excluded from this study.

We will also only concern ourselves with what is known as zero-sum games, i.e the rewards at the end of the game sum to zero (these games are also known as "strictly competitive"). Most tabletop games are zero-sum. Although what we will be discussing applies, to a certain extent, to multi-player games, we will concentrate our discussion on two-player games.

A. Game Categories

In this section we will describe how games can be broken down into different categories based on their information content.

1) Games of Perfect and Complete Information: A game is called *perfect* if all players can observe all moves made by previous players and complete if there is no uncertainty in the rewards, or, equivalently, if there is no nature (chance) player playing. This category contains games like Go, Chess, Checkers and Othello. A characteristic of these games is that optimal strategies are deterministic. The most popular solution to such games is backwards induction (also known as minimax, and proven optimal in [10]). These games can be decomposed to simpler subgames, irrespective of the previous moves played, and these subgames can be solved independently. A chess puzzle is an example of such a subgame. The solver of the puzzle does not need to know any of the actions that led to the current state, since all information is displayed on the board. This is one of the most popular categories of games, and until the last decade monopolised the interest of games AI research, with Go replacing chess in recent years as the Drosophila of AI.

2) Games of Perfect and Incomplete Information: These games have a third player, usually termed *nature* or *chance*, that performs random moves. Thus they are perfect, as all moves are observable, but incomplete, as randomness from nature plays a part in the games. The most popular game in this category is backgammon, with nature providing the random rolls of the dice. The existence of nature causes uncertainty in the rewards, thus a game has to be played repeatedly before any conclusion on the quality of the players is reached. Backwards induction works here as well (this time known as expectimax [11])).

3) Games of Imperfect and Complete Information: In these games, players cannot assess the state they are in, as they cannot see other players' moves (partially or completely). Most Game-theoretic games fall in this category. One cannot simply use backwards induction to solve such games, as one does not know in which state one is. An aggregation of states in which one might be in a certain time is called an information set. Imperfection rarely comes in real life without some kind of randomness (i.e. incompleteness), so this category of games is sparsely populated. An example real life game in this category is *Battleship*.

4) Games of Imperfect and Incomplete Information: Most Card Games belong to this category. The players cannot observe all moves performed either by other players or by nature (e.g. in Poker, nature plays a hidden move either at the beginning of the game or at specified intervals, while all player moves are known). This is the hardest category of games to solve, and is the focus of much current research. The most popular solution algorithm for these games is Counterfactual Regret Minimisation (CFR) [12].

B. Nash Equilibria and Games

We mentioned algorithms that solve each kind of game (backwards induction and counterfactual regret minimisation). But what does "a solution" mean in a multi-agent setting? In order to give a coherent answer one has to define games a bit more precisely. Broadly speaking, a game[2] can be defined as the tuple $\langle N, H, P, f_c, (\mathcal{I}_i)_{i \in N} \rangle$, where N is the set of players, H are sequences of actions α_k of each player (Histories) drawn from the set of actions $\mathcal{A}$, P is a function that assigns a player to each history (including player 0, nature) $f_c(\alpha|h)$ gives the payoffs, and, finally, $(\mathcal{I}_i)$, is an information set, i.e. the histories between which an agent cannot distinguish. An agent is a strategy σ with a body. There are a number of ideas that we need to map from game theory to decision theory. The first one is the concept of a *strategy* $\sigma(h)$, a function that takes a history and outputs an action; one can intuitively think of agent policy and strategy interchangeably. The second idea is that of a solution concept. A solution concept is what one would deem as solution to the game. Unlike decision theory, where maximising expected reward is the obvious route to take, things are not that easy in game theory. Here, one has to define what is a good strategy in the context of other players. The most popular solution concept is *ϵ-Nash Equilibria*, equation 1,

$$\forall i, \sigma_i \in S_i, \sigma_i \neq \sigma_i^* : \epsilon + f_i(\sigma_i^*, \sigma_{-i}^*) \geq f_i(\sigma_i, \sigma_{-i}^*) \quad (1)$$

The above equation states that we consider a strategy profile to be in ϵ-Nash Equilibria if one of the players has, at

[2]For exact definitions of games and strategies, please see [13]

best, an incentive to deviate from its strategy unilaterally of ϵ (σ^*_{-i} is the strategy that is not part of the equilibrium). Intuitively, in two player symmetric zero-sum games Nash equilibria, a player, by being in equilibrium, limits its worst case reward/payoff by ϵ. Assuming now that the game is *symmetric*, i.e. that a strategy depends only on other strategies, not on who is playing them, then all strategies in a $\epsilon - Nash$ profile will deviate by the same ϵ. Note that ϵ being the worst case scenario does not apply to games of more than two players, as multiple opponents might at some point decide to attack you simultaneously.

Although this is an article about zero-sum games, it is worth mentioning that the concept of Nash Equilibria might not be as important in general-sum games. For example, in Iterated Prisoner's Dilemma, one might develop notions of "superrationality", where both players have an incentive to deviate from Nash and not defect, making everyone better off. Thus one might end up with populations of agents far away from equilibrium, because everyone benefits. This is not the case in zero-sum games, as there can be no cooperation.

III. Player Measurements

Irrespective of the procedure used to generate a player, one should be able to measure a player and acquire some meaningful information regarding its performance. In this section we will consider two ways to measure performance: implicit, based on some aspect of a player's decision function, and explicit, based on a player's observed game results.

A. Comparing Players Implicitly

Instead of comparing a player with other players, one might take a route where all players are assigned an objective value, and are measured based on that objective value. Such a value stems naturally from the definition of Nash Equilibrium. The amount a player can improve on his performance by changing strategies (or equivalently, the amount one can be beaten by his opponent, both being equivalent in zero-sum symmetric games) can now be treated as a measurement of how good a player is. Intuitively, in two player games, finding the distance from the Nash equilibrium (or the distance to the closest one when there are many equilibria) can be seen as asking the question "What is the worst score a player can get". This calculation is often termed "Best Response". This calculation is an extremely pessimistic one, and fails to capture the fact that many of your opponents might not be optimal, and thus it might be worth playing a different strategy than Nash so as to better exploit weak players. Nevertheless, it provides a good objective score for each player.

B. Comparing Players Explicitly

Players can be explicitly compared by playing games with each other. This statement of the obvious leads to two important issues: deciding which players should play each other, and interpreting the results of the game outcomes.

The most complete and informative approach is the full round robin league, where every player (or agent or team etc.) plays every other player. A prominent example of this is the English Premier Football League. In many games and sports a single game may involve an advantage for one side. Examples include the advantage of playing first in Hex, or the advantage of playing at home in a sports fixture. Hence it is normal for such leagues to include at least two fixtures between each pair of players. A full round robin league using this setup with n players involves $n(n-1)$ fixtures, making the computational cost quadratic in the number of players. Additionally, many games involve a stochastic element, and the results of one or two games between a pair of players may not be indicative of their true relative strength. One solution to this is to increase the number of games played between each pair in order to ameliorate the effects of noise.

Note that any deterministic value function such as the one used here means that there are only two outcomes when playing a game of perfect information such as Othello between a pair of players: when the first player plays as Black, and when it plays as White. A valid concern would be that the observed outcomes from two such games may not be indicative of true playing strength. There are two main approaches to overcoming this problem. One is to play games from a fixed or randomly chosen set of initial positions (similar to the two-move ballot used in Checkers tournaments). The other is to vary the nature of the game slightly by forcing random moves with a given probability (e.g. 0.1). Each of these changes the nature of the original game to some extent, though the forced random move approach is the more significant departure. There are two further points of interest to this.

The first is that human players are fundamentally very different in this respect [3]. Even when the game is noise-free, as is the case for classic board games such as Chess, Go, and Othello, if the optimal strategy is unknown (at least to the players), a given pair of players will play out a different game trajectory each time. This could be for several reasons, such as a winning player having imperfect memory of how he defeated this opponent previously, or a losing player trying alternative moves in an effort to avoid the blunders of previous games.

Secondly, perhaps surprisingly, good results can still be achieved while ignoring the problem, as will be shown in the results section of this paper. The proposed archive based algorithm is based on just two deterministic games between each member of the population and each archive player, yet we still observe strong players evolving.

C. Rating Systems

In human games we often have large numbers of players who wish to play games against players of a similar standard. Rating systems are used to assign such pairings, and also become an end in themselves. Players work hard to improve their standard of play, and their rating is used as a measure of success. Example rating systems include the Elo system originally developed for chess players [15]. Elo originally proposed modelling player skill variation with a normal distribution, but a logistic distribution is more widely used.

[3] And nearly all other respects as well [14].

γ	Elo	p(win)
0.05	1200	0.50
0.10	1320	0.67
0.20	1441	0.80
0.50	1600	0.91
0.95	1712	0.95

TABLE I: Gamma values, Elo ratings, and probability of win values of various rated players versus a player with $\gamma = 0.05$. The Elo level has been arbitrarily set such that an average player ($\gamma = 0.5$) has an Elo rating of 1600.

This is the basis of the logistic Bradley-Terry (BT) model of paired comparisons [16] and can be expressed in the following equation:

$$P(i \text{ beats } j) = \frac{\gamma_i}{\gamma_i + \gamma_j} \tag{2}$$

where $0 \leq \gamma_i \leq 1 \forall i$ and, for convenience, $\sum_i \gamma_i = 1$. The summation part is not strictly necessary but makes the estimation algorithm more convenient without any loss of generality. The above γ values can be transformed into the more familiar Elo style rating with the following transform:

$$e_i = \mu_{elo} + 400 * \log_{10} \gamma_i \tag{3}$$

where e_i is the Elo equivalent of γ_i and μ_{elo} is the entirely arbitrary desired average player rating. Table I shows the relationship between γ values, Elo ratings and win probabilities versus a player with a γ-value of 0.05. Understanding this relationship is important in order to interpret the values of the KL divergence measure of intransitivity developed below.

One key assumption of the model is that the expected game result depends only on the difference in rating between two players. Hunter [17] demonstrated how a Minorization-Maximization (MM) algorithm could be used to solve the straight Bradley-Terry, model plus many of its generalisations, with very little effort needed to tailor it to the situation at hand. The algorithm is similar in nature to the Expectation-Maximization (EM) algorithms used to train hidden Markov Models, for example. During the expectation phase the algorithm calculates the likelihood of the data given the current model parameters. During the maximization phase the parameters of the model are adjusted to increase (or maintain; never decrease) the likelihood of the data given the model.

The attraction of the Bradley-Terry model is that it enables a player to be rated on the basis of a relatively small number of games. For game leagues with a large number of players, *potentially* reliable ratings can be based on a tiny fraction of the number of games that would be involved in a full round robin league. Using Hunter's MM algorithm, the model can be fit to a set of game data in a small number of iterations (typically less than 20) from an arbitrary starting point, and will converge to the best possible fit. The caveat is whether the model is a good fit to reality. In many games played between human players, such as Chess for example, it has been found to be a reasonably good fit. As will be shown in this paper, it is a poorer though still useful model for fixed value function Othello players.

This type of modelling is not restricted to analysing player strengths given an existing set of games. Under certain assumptions similar techniques can be applied to tournament design in order to pick the best player in the minimum number of rounds [18] [19], or alternatively, with the minimum number of games played. Note that if a superior player always beats an inferior player, then a simple knock-out tournament would be the optimal way to find the strongest player. However, most games have an element of chance, and as the chance element increases so the likelihood increases of a knock-out tournament being won by some other player than the true best.

The Bradley-Terry model has potentially important implications for the coevolution of game playing agents, since picking the best player efficiently is important for the selection stage of the algorithm. However, note that the model does not allow for any intransitivities. Indeed, in the literature on paired comparison models, intransitivities are usually assumed to be unproblematic. For example, Smith [20] writes:

> "Experience shows that for human players or real life games, rating triangles are not a significant problem. Probably the reason for this is that the existence of triangles depends on the existence of specific player strengths and weaknesses. Human players tend to compensate for perceived weaknesses and to try for well rounded play, thus avoiding rating triangles."

Note that there are some more sophisticated rating schemes in use such as TrueSkill^{TM} [21]. TrueSkill^{TM} models each player with two numbers: one to estimate a player's skill level, and one for the uncertainty of that estimate. Like the Bradley-Terry model, however, it assumes that playing strength is transitive.

An interesting question that we aim to shed some light on in this paper is how much of a problem these intransitivities or "rating triangles" are when trying to coevolve value functions. The result of this analysis also has important practical implications. If the Bradley-Terry model could be reliably applied then the use of an archive and of full round-robin leagues would become redundant. It would be enough to retain the current best player and evolve against this. For our chosen study of evolving fixed value functions to play Othello we show conclusively that this is not sufficient.

There are two distinct cases in which the Bradley-Terry model may give poor results when applied to player measurement: they are essentially problems of under-prediction and of over-prediction. In one case there may be genuinely different but strongly intransitive relative player strengths (e.g. as in Rock-Paper-Scissors) that become hidden in the equal ratings assigned to such players. In such a case the model would incorrectly predict even outcomes between each pair of players, whereas a more expensive but complete pair-wise model would be able to make correct predictions.

Another case arises when the model fits well to most of the players, but the pool of players may contain some strong intransitivities. In such cases the model makes confident predictions that A will beat B, where A is the player with the higher rating, when in fact B will beat A most of the time.

D. Detecting Intransitivities

If there are intransitivities in your tournament, is there a way of detecting them? We propose the use of two different measurements: the transitivity index τ, and the KL divergence D_{KL} between the win rate predicted by the BT model and the observed win rate.

1) Computing a Transitivity Index: The first such measurement is the transitivity index [22]. There are many forms this index can take, but in each simple incarnation it measures the numbers of 3-dags (directed acyclic graphs with 3 nodes) that occur in the tournament divided by the total number of possible 3-dags. A 3-dag is what we would expect if there were no intransitivities. If player A beats player B and player B beats player C, we would expect player A to beat player C. This forms a dag. On the other hand, if player C beats player A then this forms a cyclic graph, which we also refer to as an intransitivity cycle.[4] Given n players a fully transitive relation would give $\binom{n}{3}$ 3-dags. If we symbolise with T_3 the number of times this actually happened, we get a transitivity measurement of:

$$\tau := {}^{T_3}/_{\binom{n}{3}} \tag{4}$$

The above equation measures the degree of transitivity in a population, with $\tau = 1$ when the population is fully transitive and $\tau = 0$ when the population is fully intransitive.

2) KL divergence: The problem with the above transitivity index is that it is somewhat brittle: a small change in the observed game outcomes can lead to a maximal change in the transitivity index, as will be seen in the examples below.

An alternative is to use a statistically motivated measure of intransitivity. One such approach is to fit a Bradley-Terry (BT) model (described above) to the observed game results, and then measure how well this predicts the results. In more detail, the BT model produces a rating for each player, and the difference in ratings between each pair of players can be used to estimate the probability that player A will beat player B. We can then measure how different the result is for each pair of players, and take an average over the set of players to see how well the model predicts the outcomes. Hence, we have the problem of measuring the difference between two probability distributions. A natural way to do this is to use the KL divergence[5].

Suppose the BT model predicts a win rate of p_{ij} for player i versus player j, but we observe a win rate of q_{ij}. In general the KL divergence D between two discrete probability distributions P and Q is defined as:

$$D(P||Q) = \sum_{i=1}^{n} P_i \log \frac{P_i}{Q_i} \tag{5}$$

We consider only two outcomes: when i wins and when j wins, hence $n = 2$; draws are modelled as half a win for each player. We map this to the predicted and observed win rates as p, q for $(i = 1)$ and $(1 - p), (1 - q)$ for $(i = 2)$.

Hence it can be shown that:

$$D_{ij}(p_{ij}||q_{ij}) = \ln \left[\frac{1-p_{ij}}{1-q_{ij}}\right] + p_{ij} \ln \left[\frac{p_{ij}(1-q_{ij})}{q_{ij}(1-p_{ij})}\right] \tag{6}$$

The KL divergence as defined above is an asymmetric measure of difference. We use the symmetric version by taking the average of $D(p, q)$ and $D(q, p)$.

One problem with our use of KL divergence lies in the assumption that the observed win rate is the actual win rate. When the number of games played is small, the empirical estimates can be highly unreliable (for example, the fact that player A beats player B by four games to zero does not imply that A's true win-rate against B is 1.0). The correct solution to this is to use Bayes' theorem to compute the posterior distribution of the win rate given the observed game outcomes. For the sake of simplicity we take an alternative approach in this paper: we add a small constant (for these experiments we used 0.001) to the count of the number of games won by each player, to avoid the empirical estimate taking on extreme values. The other weakness in our current method which stems directly from not using Bayes' theorem is that no account is taken of the number of games played. For example, if the BT model predicts a win rate of 0.4, then an observed game result of $2-2$ would be closely aligned with the prediction, whereas a result of $200 - 200$ would be rather unlikely.

A secondary problem with using the KL divergence is the problem of interpreting what a particular figure for the divergence means in practical terms. For example, a KL divergence of 0.0 means that the observations were exactly in line with the predictions of the BT model, but it is less obvious what a D_{KL} of 0.05 or 0.3 means. There is no bound on how different two distributions can be e.g., if $p_{ij} = 0.5$, then D_{KL} tends to infinity as q_{ij} tends to 0.0 or 1.0. In order to aid interpretation Table II shows the asymmetric KL divergence measure from equation 6 applied to a predicted win rate of p shown in the row when observing an actual win rate of q in the column.

3) Examples: In order to illustrate the differences between the transitivity index and the KL divergence we created some sample round-robin matrices with hypothetical games results from 10 games between each pair of players. Each matrix involves just three players and so illustrates aspects of each measure in a rather extreme way; nonetheless we believe these to be simple yet informative examples to study. The τ index can be calculated directly from the entries in each table. For the D_{KL} measure we fitted the BT model using our implementation of Hunter's MM algorithm to find the Elo scores, and use these to calculate the divergence between the actual and the expected win rates.

Each row shows the games won by that player against each opponent player. Table IIIa shows a *Rock-Paper-Scissors* style example. The τ index rates this correctly as having zero transitivity. The D_{KL} is high at 1.13, but would be even higher if we'd chosen a smaller value of the constant (recall that it would be infinite if we estimated the empirical win rates as zero or one).

[4]Note that [22] define use the term *transitivity cycle* but this seems like a contradiction in terms, since transitivities are not cyclic.

[5]This method was used in the tournament design algorithm developed by Glickman and Jensen [18], where they pair players in order to maximize the KL divergence between the current win rate estimates and the next ones (hence arriving at a maximal potential gain in information).

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.00	0.04	0.12	0.23	0.37	0.55	0.79	1.15	1.76
0.2	0.04	0.00	0.03	0.09	0.19	0.33	0.53	0.83	1.36
0.3	0.15	0.03	0.00	0.02	0.08	0.18	0.34	0.58	1.03
0.4	0.31	0.10	0.02	0.00	0.02	0.08	0.19	0.38	0.75
0.5	0.51	0.22	0.09	0.02	0.00	0.02	0.09	0.22	0.51
0.6	0.75	0.38	0.19	0.08	0.02	0.00	0.02	0.10	0.31
0.7	1.03	0.58	0.34	0.18	0.08	0.02	0.00	0.03	0.15
0.8	1.36	0.83	0.53	0.33	0.19	0.09	0.03	0.00	0.04
0.9	1.76	1.15	0.79	0.55	0.37	0.23	0.12	0.04	0.00

TABLE II: The asymmetric KL divergence between expected win rate p in the row versus observed win rate q in the column. For example, expecting 0.1 while observing 0.5 gives a divergence of 0.37, while the reverse of these gives a divergence of 0.51.

#	1	2	3	Elo
1	0	10	0	1600
2	0	0	10	1600
3	10	0	0	1600

TABLE IIIa: Sample results matrix for *Rock-Paper-Scissors* style players. Fitting the BT model gives each one an equal rating of 1600, yet the observed pair-wise results are very different. For this matrix, $\tau = 0.0$ and $D_{KL} = 1.13$

#	1	2	3	Elo
1	0	4	9	1676
2	6	0	6	1651
3	1	4	0	1473

TABLE IIIb: Sample results matrix for three players with some element of intransitivity. Based on all the results in this matrix Player 1 has a slightly higher rating than player 2, yet has lost 4-6 to that player. For this matrix, $\tau = 0.0$ and $D_{KL} = 0.05$. The τ measure rates this as totally intransitive, whereas the D_{KL} rates this as a mild departure from the predicted result.

Table IIIb shows a case with more mild intransitivity. The overall better player (the player with the higher rating having won more games in total) now loses to the second player 4-6. The τ index unfortunately rates this as being just as intransitive as the results in Table IIIa.

Table IIIc is the same as Table IIIb except that players 1 and 2 now win 5 games each. This is a difference in the result of just one game. The τ index undergoes a maximal change from 0.0 to 1.0, rating these results as being perfectly transitive. The D_{KL} scores in these cases are more reasonable. They show both results to be not too surprising given the predictions of the BT model, with Table IIIb diverging by 0.05 and Table IIIc diverging by 0.03. Note also how the Elo rating of player 3 has gone down from 1473 to 1471 despite the fact that its game results are identical to those in Table IIIb. The reason is that player 3 lost 4 games to player 2, and player 2 is weaker than before having only won 5 games against player 1. Player 3's single victory against the now higher rated player 1 is not sufficient to offset this.

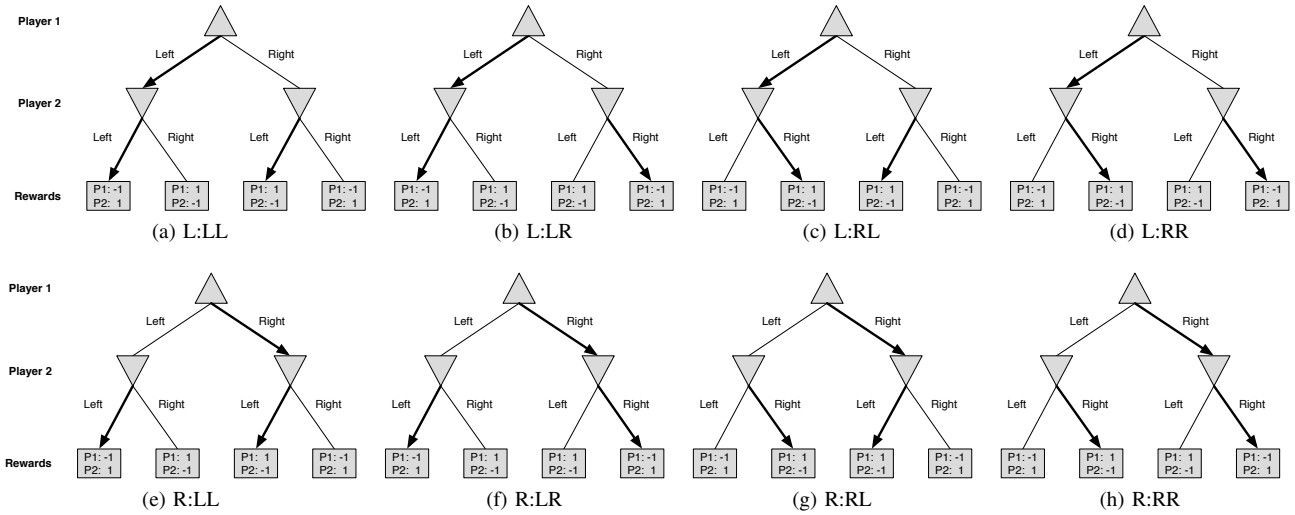
#	1	2	3	Elo
1	0	5	9	1703
2	5	0	6	1626
3	1	4	0	1471

TABLE IIIc: The same results as in Table IIIb except that players 1 and 2 now win 5 games against each other (instead of 4-6). This difference in outcome *of a single game* causes a maximal change in τ from 0.0 to 1.0. It has little effect on D_{KL}, changing it from 0.05 to 0.03.

IV. Coevolution & Evolution

In traditional artificial evolution each individual g_i in a population of individuals G is transformed to its corresponding phenotype and assigned a value $F : X \to R$, $G \subseteq X$ where function F is termed the *fitness function*, and X is the search space. A set of operators is applied on the population (ex. mutation, crossover), and we move to the next generation. The process continues until some stopping criterion is met. Coevolution does almost the same thing, with the exception that fitness is measured relative to other agents. Thus, at least in competitive coevolution, we are trying to evolve agents in multi-agent settings. This creates a situation where there is no *objective* function to measure overall progress, but rather a series of *subjective* tests [23]. In a significant number of cases it leads to mediocre performance from the evolved agents, mostly due to intransitivity [24].

We will examine these problems analytically below, and give a simple example of cycling behaviour. In order to showcase such cycling behaviour in a game of perfect information we introduce a very simple game called "matching sides". The rules are as follows: The first player chooses one of two moves, Left or Right. The second player then moves: Left or Right again. If the first player and the second player have made the same move, the second player wins. Otherwise, the first player wins. The reward of the winning player is 1, while the reward of the losing player is -1. All the relevant games and strategies of the game can be seen in Figure 1. Strategies are encoded in the format [Player 1 Move]:[Player 2 move in the Left Branch][Player 2 Move in the Right Branch]. Moves are coded as L for Left and R for Right. Note that a strategy coded in this way defines what to do when playing as player 1 and when playing as player 2. For example, consider the sample strategy L:RL — this plays L when playing first. When playing second it plays R if first player chose L, and plays L if first player chose R. Table IV shows the results of each possible strategy played against every other one, from the point of view of the



(a) L:LL (b) L:LR (c) L:RL (d) L:RR

(e) R:LL (f) R:LR (g) R:RL (h) R:RR

Fig. 1: Possible strategies in our "matching sides" game. Strategies are encoded in the format [Player 1 Move]:[Player 2 move in the Left Branch][Player 2 Move in the Right Branch]. L is used for Left and R for Right. Nash Equilibria are the following Strategies: R:LR (subfigure 1f), L:LR (subfigure 1b).

	L:LL	L:LR	L:RL	L:RR	R:LL	R:LR	R:RL	R:RR	Total
L:LL	0	0	2	2	-2	-2	0	0	0
L:LR	0	0	2	2	0	0	2	2	8
L:RL	-2	-2	0	0	-2	-2	0	0	-8
L:RR	-2	-2	0	0	0	0	2	2	0
R:LL	2	0	2	0	0	-2	0	-2	0
R:LR	2	0	2	0	2	0	2	0	8
R:RL	0	-2	0	-2	0	-2	0	-2	-8
R:RR	0	-2	0	-2	2	0	2	0	0

TABLE IV: Payoffs for each player versus every other player, from the point of view of the row player. Each table entry is the sum of two numbers, resulting from playing first and from playing second. For example, row two column three shows that L:LR versus L:RL leads to 2 points for strategy L:LR, 1 for when it plays first, and 1 for when it plays second.

row-player, together with the total score for each player. From this is can be seen that the optimal strategies are L:LR and R:LR, both with a total score of 8. Each pair-wise table entry is the sum of the score for the row player versus the column play when the row player plays first and when it plays second. Since there are only 8 possible strategies the optimal ones can be found trivially through enumeration, but the essential points still apply to much larger games that cannot be solved through enumeration.

We use "matching sides" here instead of the more popular "Rock-Paper-Scissors" (RPS) primarily because of the different information content of the game. RPS is a game of imperfect information, while analysis in this paper focuses on games of perfect information. For a complete up-to-date review of the principles behind co-evolution one can look at [25].

In Table IV the following cycle of intransitivity can be observed, with the greater-than sign indicating that one player directly beats the next player as follows:

$$\text{L:LL} > \text{L:RR} > \text{R:RR} > \text{R:LL} > \text{L:LL} \tag{7}$$

Next we illustrate how both round robin tournaments and limited-size archives can be trapped in such cycles.

A. Round Robin Tournaments

This is the original "coevolution" method. An initial population is generated at random, and players play against each other. The players are ranked according to some statistical method (or by taking exact measurements between the players playing). This approach generally does not produce strong agents (for example see [26]) or it requires heuristics specific to this method in order to work. Imagine a scenario involving a population of size 2 where the initial strategies are L:LL and L:RR. We have now two individuals, with L:RR being superior, and hence being selected as parent while L:LL is discarded. After a mutation, the population now becomes L:RR and R:RR. The stronger of these two is R:RR, which could then be beaten by an its offspring R:LL, which might then lead back to L:LL, and so on. Hence, it is possible for coevolution to follow the cycle shown in Equation 7. Now, because in our example we have a tiny game, it's possible to have larger populations, where all possible Nash Equlibria are part of the initial population. If that's the case, then the game will converge in round one. Ficici [27] has shown that the population will converge to the right mix of strategies if there is a need for mixed strategies (in the case where the game becomes imperfect). Alternatively, in this example the evolutionary algorithm could have broken out

of the intransitivity cycle at any point by mutating to either optimal strategy (L:LR or R:LR). In more complex games with larger strategy spaces this is harder to do. It is also possible to be very careful with what mutations can enter the population (as in [8] where parent-child weighted averaging was used to overcome the effects of noise) or with carefully balancing an action selector (as in [28]).

B. Archives

Another approach, that will be experimentally explored in this paper, is having archives; i.e., keeping snapshots at some intervals of the evolving population. A popular use of the archive has been to treat it as a Pareto front [29]. Thus, a player performs better than another if they perform better against all members of the archive.

In the "matching sides" game it is trivial to show that cycling can easily take place in this scenario. To see this, assume an archive of size one containing an initial Random Player L:LL. If the mutation cycle shown in Equation 7 is followed then obviously cycling will occur. In this simple example this can easily be solved by having a sufficiently large archive. For example, if the archive was of size 4 and consisted of all four players in the cycle, then only one of the optimal strategies would be superior, and if chanced upon would then be correctly identified as the optimal. However, as the state space increases, the probabilities of cycling become stronger, and need larger and larger archives to control the resulting effects.

C. Evolving Against a Fixed Player

A third approach, not belonging strictly to coevolutionary techniques, being closely aligned to evolution, but trying to solve the same problems, has been to evolve against an existing fixed player. However, it is easy to see why this might fail as well, especially if the comparison measurements are not done correctly. If we assume our fixed player is R:RR, it might be that one evolves L:RR as a winning player. This however is not an optimal strategy (although it is impossible to know it at the time). A way to escape this is to add some kind of noise and hope to explore different subtrees of the game as a result, but this can be problematic as well. Not all subtrees will be seen, and it will make the player extremely specialised (and maybe of limited value in the long run). The problem we observe with Othello is that we can evolve a player that successfully beats an existing heuristic player, but is not a good all-round performer. A natural extension of this idea is to evolve against a fixed *set* of players. This leads to the question of how to choose such a set. One approach is to create a large random set of players, as proposed by Chong et al. [9]. This is an interesting idea, but as we show later, does not lead to particularly strong players. In the "matching sides" example, evolving against the four players in the cycle (Equation 7) would provide an ideal set of fixed opponents.

D. Measuring the distance to Nash

This is a fourth way of doing coevolution as presented in [30]. We explicitly measure ϵ for each agent and we try to minimise it. This does not require having an optimal Nash player, it just requires a way for finding the worst possible performance of that agent (i.e. ϵ in two player settings) - or, equivalently - how much a player regrets playing a strategy. This is reasonably straightforward, but it might be the case that the game is too big and getting an accurate measurement might be impossible in practice.

V. Methodology

In this section we present a simple archival algorithm for achieving (or at least aiming towards) Nash equilibria in all four categories of games mentioned in Section II-A.

A. NashSolve

We propose a new simple algorithm for coevolving players. The algorithm is named *NashSolve*, in line with *MaxSolve* [31]. The pseudocode can be seen in Algorithm 1. Intuitively, we start with a random player in an archive of players. We compare all of our candidate solutions with this player, and record their performance. If the worst case performance of a new solution against the archive is above a *lowerThreshold*, we add it to the archive. We then iterate, assigning to each candidate solution its worst score against all the players in the archive. A fixed size archive is used. When the size limit is reached any new player added to the archive replaces the oldest player there. Note that we are trying to push up worst case performance, in line with the ideas presented in Section IV.

For games of incomplete information one has to play multiple games before a score can be recorded, but the same principle applies. The score of each player is its worst possible score in the archive.

Although not used in this paper, for games of imperfect information, the phenotypes of candidate solutions must express probabilistic policies, whereas it suffices to convert them to deterministic policies (by taking the action with the maximum probability as the only action one should perform at each information set) when inserting them in the archive.

For the kind of games we will test, it's easy to provide the evolutionary algorithm with additional information. A win in Othello gives you a score of 1, a draw 0.5 and a loss of 0. Because the worst case performance against a single player will always be 2, 1.5, 1, 0.5 or 0 (two wins, one win/one draw, two draws, one loss/one draw, two losses - as we need to play for both black and white to get symmetry), one can provide the algorithm with a weighted score against everyone in the archive, but add a new player only when we can beat every single player in the archive (or, equivalently not lose or draw to any player in the archive).

We are aware of certain similarities (and differences) with two other algorithms. The first of them is *MaxSolve* [31]. *MaxSolve* requires two populations, one generating tests and one generating solutions. Tests are then added to the archive, and used to test to the current set of solutions. The goal of *MaxSolve* is for the solution population to solve as many test cases as possible, i.e. win as much as it can in expectation (scores for each solution are linearly weighted averages against

Algorithm 1 NashSolve

```
define global input EA, GAME
define global input maxArchiveSize, lowerThreshold
define global input maxIterations
procedure NASHSOLVE
    for i ← 0 to maxIterations do
        Archive ← {RandomPlayer}
        Solutions ← EA.getCandidateSolutions()
        for each s in Solutions do
            minScore ← ∞
            for each a in Archive do
                score ← GAME.play(s, a)
                if score < minScore then
                    score ← minScore
            s.score ← minScore    ▷ Our main score is our worst case performance
        sort(Solutions)    ▷ Sort solutions, using another measurement (e.g. average) to break ties
        ▷ Check if player is suitable for the archive, if yes add it and remove any excess players
        if Solutions[0] < lowerThreshold then
            Archive.add(Solutions[0])
            if |Archive| > maxArchiveSize then
                Archive.remove(0)
        EA.pushSolutions(Solutions)    ▷ Give solutions back to the evolutionary algorithm
    return Archive[−1] ▷ Return last member of Archive
```

candidates from the test archive). Our algorithm can be seen as a simpler version of *MaxSolve* aimed towards games, where the ultimate goal is to improve your worst case performance and this requires no separate test generation procedure. Another algorithm one can draw parallels with is Incremental Pareto dominated Archive(IPCA)[29]. As with *MaxSolve*, IPCA also uses two populations, one for solutions and one for tests. Tests are kept if they are not dominated by any solution and no two solution members have the same score. A relevant mechanism exists for identifying useful tests. Scores for each solution are assigned by comparing the solution to each test and creating a Pareto ranking between different solutions. Again, our algorithm can be seen as a specialised version of IPCA (although the modifications required to IPCA are far greater than for *MaxSolve*), with a bounded archive, tailored towards zero-sum games, where there is some relationship between tests. As a final remark, note that our algorithm is more practical in nature, with no requirement to store and make comparisons with potentially enormous Pareto fronts. Furthermore, our algorithm can be implemented easily as a fitness function plug-in to some standard well designed evolutionary algorithm; in this case we use CMA-ES. This is a distinct advantage.

VI. Experiments with Othello

Othello is a challenging game, where the best computer players have played at superhuman levels ever since Buro's Logistello [32] defeated the Othello world champion in 1997. Nonetheless, it is still a game played by humans with active leagues in many countries. For studies in computational intelligence it has several attractive features: the rules are very simple and easy to implement efficiently, the game is unsolved and optimal strategies are not known, and the game can be very deceptive. The last point makes observing the game play rather interesting: the player that appears (at least to novice eyes) to be leading in the mid game frequently loses.

Othello is played on an 8×8 board between two players, black and white (black moves first). At each turn, a counter must be placed on the board if there are any legal places to play, else the player passes. Each move must be made on an empty square and must *pincer* one or more opponent counters on a continuous line between the new counter and an old counter. All opponent counters that are pincered in this way are flipped over to the color of the current player. The initial board has four counters (two of each color). The game terminates when there are no legal moves available for either player, which happens when the board is full (after 60 non-passing moves, since the opening board already has four counters on it), or when neither player can play. The winner is the player with the most pieces of their color at the end of the game. Counters placed in one of the four corners can never be flipped and therefore play a vital role in the game. Placing a high value on the corners tends to be the first thing learned, although interestingly our strongest evolved players do not place a high value on all the corners: perhaps they have learned to sacrifice a corner.

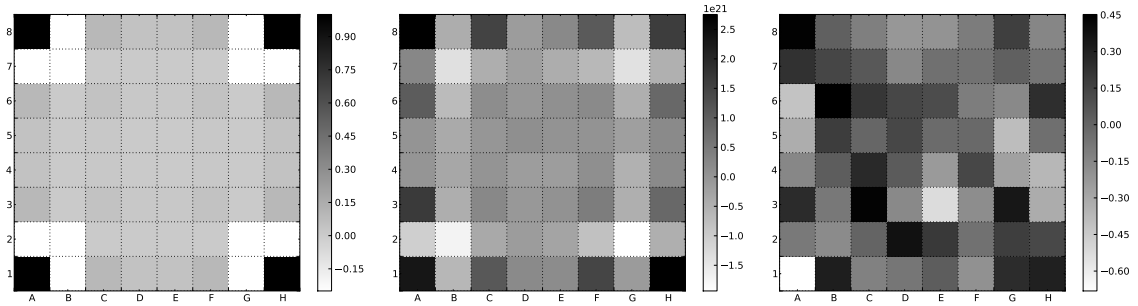
We perform a number of experiments in Othello using a weighted piece counter (WPC) as the form of value function to be evolved by each approach. The 8×8 board is unwound as a 64 element input vector $\boldsymbol{\phi}$. Each element of $\boldsymbol{\phi}$ is 0 for an empty square, $+1$ for a black counter and -1 for a white counter. Hence, black is the maximising player and white is the minimising player. The WPC has a vector $\mathbf{w}$ of 64 weights, one for each square on the board. The value of a board $v(\boldsymbol{\phi})$ is then calculated as the scalar product of the input vector and the weight vector:

$$v(\boldsymbol{\phi}) = \boldsymbol{\phi}\mathbf{w} \tag{8}$$

Figure 2 (left) shows the standard heuristic weights for Othello [33], where darker squares are ones estimated to be better to place a counter in. Note that these weights are symmetric, which seems reasonable given the natural symmetries of the game. Interestingly, the best weights we evolved are highly asymmetric, and significantly outperform the standard heuristic weights. The middle and right hand WPCs in Figure 2 illustrate this: both are asymmetric, and both beat the standard heuristic WPC.

When choosing a move a one-step-lookahead is performed. The resulting board (after- or post-decision-state) is evaluated and the move with the best corresponding evaluation is chosen. This is equivalent to one-ply minimax search[6]. The one-step look-ahead process is depicted in Figure 3 which shows a board with black to move. This also illustrates the limitations

[6]One-ply minimax is a degenerate case of the algorithm since for each move the algorithm considers only the player about to move.



(a) The heuristic WPC. (b) WPC found with an archive four. (c) WPC weights when evolving against the heuristic.

Fig. 2: WPC weights of some sample players: the darker the square, the more favourable it is for a player to play there.

of the WPCs in general. The weights indicate that playing next to a corner is bad, but cannot disregard this policy when the corner has already been offered to the other player (black has already offered it by occupying B2), or has already been filled.

We performed two experiments using NashSolve, one with an archive size of $|A| = 1$ and one with an archive size of $|A| = 4$. Since this is a perfect information game, we use NashSolve as follows: we start with a random player in the archive, and try to evolve a player that outperforms it (i.e. can beat it as both black and white). If we find one, we add it to the archive as well. We now need to find someone that can beat both players, and we continue this process until we get to the point where we have reached the maximum archive size. At this point we remove the oldest player from the archive and add the newest one.

We also performed three more experiments without using archives. One where we evolve against a standard heuristic player developed by [33] and also used in [8]. The strongest player that results from evolving directly against the heuristic we term the "Anti-Heuristic" player. We have also evolved two players against a large set of randomly generated players. This latter approach is the method proposed by Chong et al. [34], where they measured performance against a large population of randomly generated players, and used Chebyshev's inequality to place loose bounds on the true performance. In some follow up work Chong et al. [35] investigated the relationship between generalisation and diversity. More recently, Chong et al. [9] found that the distribution of performance measured in this way was well approximated by a Gaussian distribution, and used this observation to tighten the bounds. The algorithm was called Improved Coevolutionary Learning (ICL) by Chong et al. [9]. For these later two experiments (i.e. the ones using ICL), we generated 500 and $50,000$ random individuals respectively, with uniform random weights in the range $[-0.5, 0.5]$.

For all the methods under test we keep a permanent archive to store solutions during the course of evolution for later analysis and possible use. For NashSolve, every fixed number of generations $(300/|A|)$, we add one player to a permanent archive. In evolution against the heuristic, and for the ICL methods, we add to the permanent archive the best player of every generation. These permanent archives play no role in evolution, but are used to provide players for a round-robin tournament. This post-hoc tournament plays an important role in analysis, since learning in this domain is a noisy process, and there is no guarantee that the final generation will include the best player encountered during an evolutionary run.

We evolve using NashSolve for $30,000/|A|$ generations. To ensure the total number of games player remains similar for each method we evolve for 100 generations in the heuristic and the ICL experiments. At the end of the evolutionary run, we run a round-robin tournament with all participants for 100 games. In all experiments we use CMA-ES [36] with a population of size 100. Note that a "game" in the archival method consists of two games without noise (with each player playing as black and as white), whereas in the case of evolving against a heuristic we play 500 games (250 as black, 250 as white) with 0.1 probability of each player making a forced random move.

In the case of ICL we play $10,000$ games per phenotype (player) versus a set of randomly generated players in the case of ICL-500 (each player plays 10 games as black and 10 games as white against 500 fixed but randomly generated players). ICL-50,000 used a larger population but only two (noise-free) games against each of the 50,000 random players. The progress of the ICL-500 algorithm is shown in the graph and in the CIAO plot.

For all four experiments we have plotted the CIAO plot [37], by doing a round robin tournament between all players in the measurement archive. 100 games were played between each player with a forced random move value of 0.1. CIAO (which stands for " Current Individual against Ancestral Opponents") plots are heat maps that compare a player with all the players from a previous generation (or in our case, the measurement archive). A point in the map (x,y) signifies the relative strength of player x playing against player y taken from a previous generation. So on each row, one can see the strength of the player vs all the generations before it. Ideally the CIAO plot should have light colors in early generations and darker colours as we progress on the x axis for each player, signifying an increase in strength of the players as they progress from generation to generation.

Notice the lack of progress evident in the plots in Figures 4a, 4c and 4d. In the case of the archive of size 1, this is due to cycling. In the case of ICL, and also when evolving against

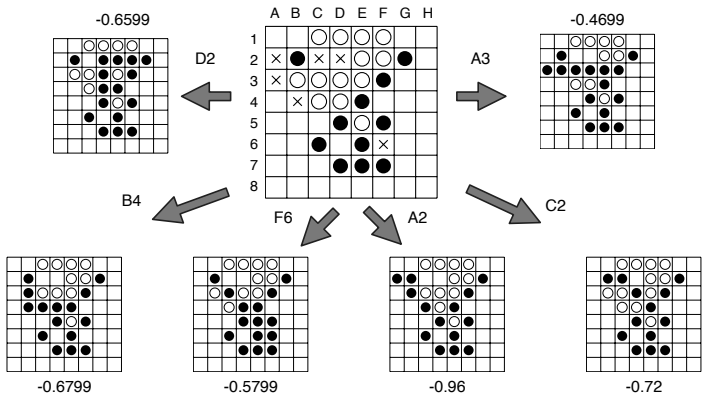
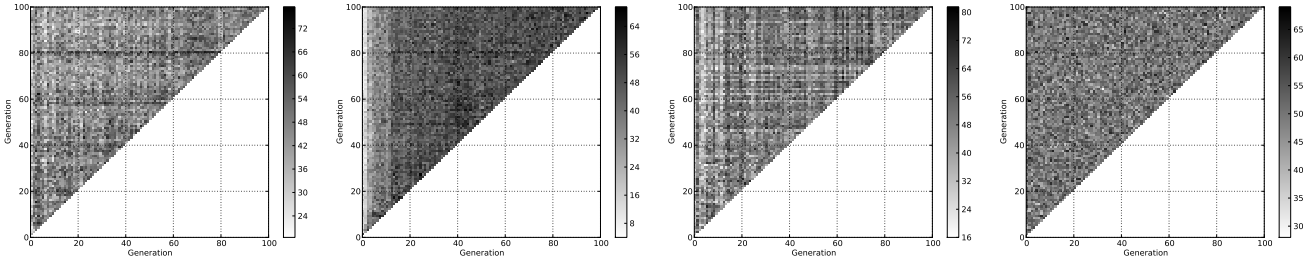


Fig. 3: One step look ahead move selection. The set of after-states is found by applying all possible legal moves to the current game state, for the player about to move. Each of the after-states is scored by the weighted piece counter (WPC) for that player, and the move leading to the most favourable state is selected. This is the state with the maximum value if it is black's move, or minimum value if it is white's move. In this example black will play A3 since this move has the maximum (least negative) value (-0.4699).



(a) CIAO plot with an Archive of size 1.

(b) CIAO plot with an Archive of size 4.

(c) CIAO plot when evolving against the heuristic.

(d) CIAO plot of ICL-500.

Fig. 4: CIAO plots for four experiments

a heuristic, it is because these do not provide ways to generate genuinely strong players. The players evolve to perform the task at hand well, but being good at the chosen task is not a strong indicator of being generally good against a set of strong players. In the case of evolving against a heuristic, evolution clearly over-fits to be very good at beating that specific player, but not good in general. In the case of ICL we see a slightly different type of over-fitting: it evolves players which are very good at beating a large number of randomly generated players, but this task is too easy to generate sufficient evolutionary pressure to produce strong players. These observations can be made from the data in Table VI.

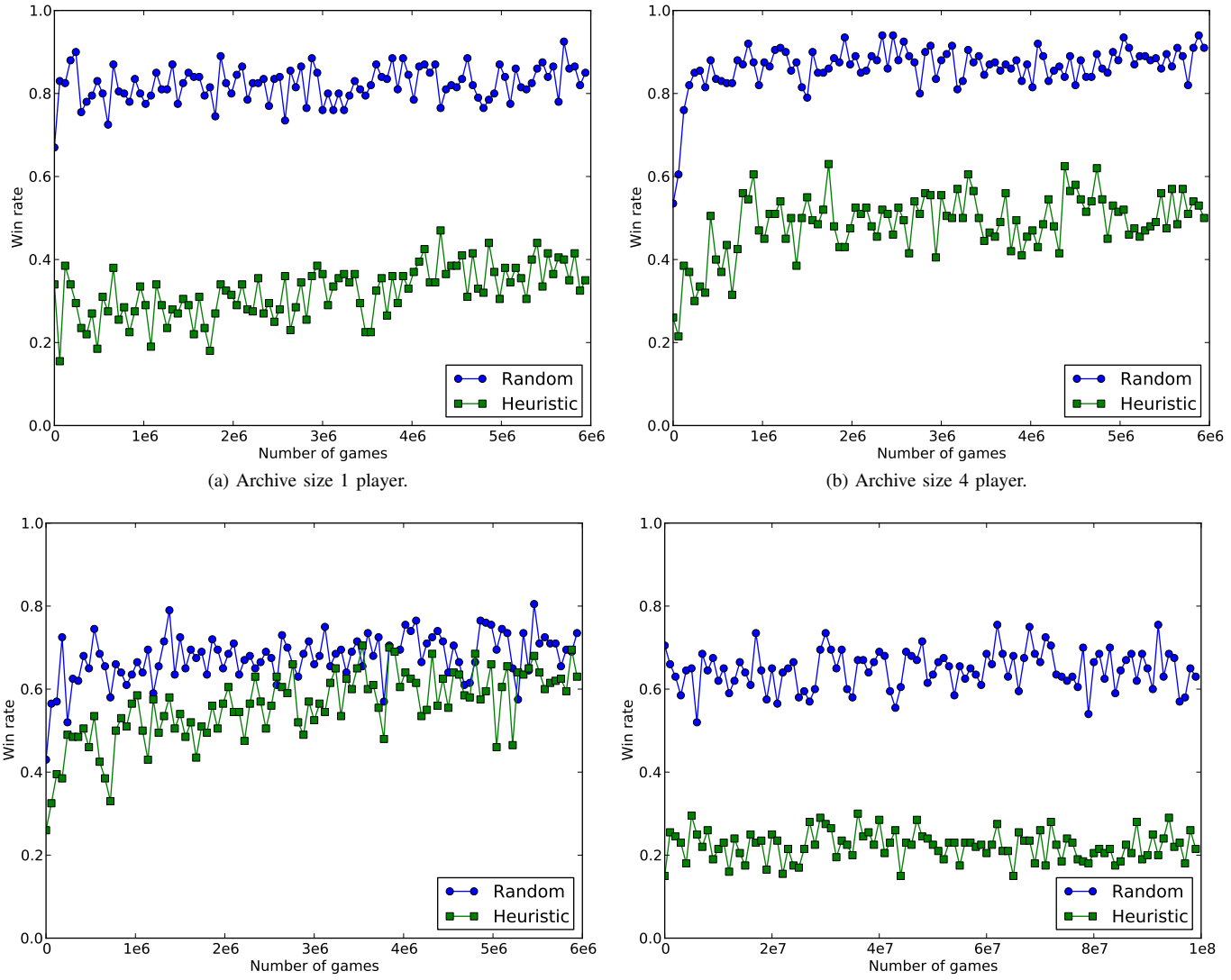
We also plot the performance of the players in the permanent archive (or the best of each generation, for the non-archival methods) versus random and the heuristic in Figure 5. Again, all comparisons use 100 games and a forced random move value of 0.1. One can see here how there is drift towards better and better strategies with an archive size of 4, although there is indeed some cycling (as is also evident from the CIAO plots).

The τ and D_{KL} score for each population can be seen in table V. This was measured using the same data set of players that formed the CIAO plots, and is used to indicate the transitivities between the sets of players saved during the run of each method. The higher the transitivity, the more evidence there is that evolution is working toward generally stronger players. The τ measure indicates this, with the archive-4 method having the highest value of τ and the ICL-50 method having the lowest value. Recall that D_{KL} does not measure intransitivity directly, but instead computes the divergence between the game results and the predictions of the Bradley-Terry model. Hence, the D_{KL} figures need a more careful interpretation, but can be explained with reference to the previous discussion in Section III-D2. In the case of the ICL-500 method we have a population of not especially strong players, so the observed results do not diverge much from the predictions of the BT model. In the archive-4 method we have a range of players from weak to strong with relatively low intransitivity. In the archive-1 and Against-Heuristic methods there is a fairly wide range of playing strength but significant intransitivity: hence we see the largest D_{KL} values.

	$\|A\| = 1$	$\|A\| = 4$	Against Heuristic	ICL-500
τ	0.6120	0.6793	0.6204	0.5013
D_{KL}	0.0087	0.0060	0.0099	0.00461

TABLE V: Transitivity measurements for each experiment.

Finally, we performed a round robin tournament with the best agents from a number of sample runs (we pick the best agent from each run by performing a round robin tournament



(a) Archive size 1 player.

(b) Archive size 4 player.

(c) Player evolved against the heuristic.

(d) ICL-500 Player.

Fig. 5: Performance (win rate) against uniform random and the Heuristic player for all evolved players, plotted against the number of games played during the evolutionary process.

using all the players in the permanent archive for that run). The results are shown in Table VI, with each row showing the number of points scored by each player versus the player in the column (1.0 for a win, 0.5 for a draw), given 20,000 games between each pair of players. We have also included an agent from a previous paper of Lucas and Runarsson [8], shown as "LR" in table VI. Lucas and Runarsson found that simple straight coevolution using a round robin league failed to evolve any good players, and they used parent-child weighted averaging to reduce the effects of noise and overcome this problem. The LR player evolved in this way is a strong player in our league, in joint second place with our typical archive-4 player. The strongest weighted piece counter we know of is the "SP $|A|$ = 4"; this was found by running our algorithm many times and picking the strongest of the resulting players.

The final two columns of the table show the percentage of points won from the total possible, and the BayesElo[7] score for each method. This is generated using a mean Elo score of 1600, and using the BayesElo software on all the league results to estimate the Elo ratings. The BayesElo software also gives posterior probabilities that each player is truly superior to each one below it. Note, given the intransitivities that we observe in this game, this does not say anything about whether one player will beat another one in a head-to-head match. Rather, it is a statement of whether the rank order is likely to be genuine, given this round-robin league of data[8]. Based on this, all rating orders are significant with a probability of 0.99 or higher except for players $|A|$ = 4 and LR, which had almost identical aggregate performance. Despite the fact that ICL-50,000 and $|A|$=1 both won 52% of their games, BayesElo estimates ICL-50,000 to have a higher rating with a probability of 0.99, and they differ by 6 Elo points which is deemed

[7]http://remi.coulom.free.fr/Bayesian-Elo/

[8]Note that the probability estimates produced by BayesElo offer a more natural statistical measure than the frequently used alternative of performing t-tests. BayesElo directly estimates the probability of superiority.

significant based on the 160,000 games played by each player. Not too much should be read into small differences though (e.g. less than 10 Elo points difference), because they apply to those particular players rather than the methods that generated them.

VII. Discussion and Conclusions

We have shown how cycling occurs naturally in coevolution and how such cycling can be measured. We have also explained the dynamics of cycling and explored how it affects different games: in one case a toy abstract game, in the other case the game of Othello played by 1-ply minimax players with weighted piece counter value functions. What should be noted here is the strength of intransitivities and cycling in coevolution and how this is different to humans. While human players retain some memory of previous events and attempt to avoid previous mistakes, thus creating more "cardinal" ladders, this is not the case with the simple players evolved in this study, and for that matter in most previous studies. This is because a player in our algorithm is completely defined by a vector of weights (in this case 64 weights) and has no adaptive element; any adaptation is the result of an evolutionary process rather than adapting to a particular opponent during a set of games.

Archives (or direct measurements against Nash Equlibria) help alleviate the problem of cycling, however since the size of the archive can be prohibitively large in order to guarantee progress, measuring the quality of coevolutionary progress is of paramount importance. It is worth noting that following the principles described in this paper, one should be able to tailor a particular algorithm to reduce the effects of intransitivities.

One of the main points of the paper is the difficulty of measuring player strength when comparing fixed value function game players, since the lack of adaptation can easily lead to strong intransitivities. While there are inherent dangers in measuring playing strength with respect to a particular set of opponents, our methods were shown to evolve among the strongest weighted piece counters (WPCs) yet discovered for playing Othello at 1-ply. They are superior to the standard heuristic weights, greatly superior to randomly generated WPCs, and those evolved using Chong et al.'s [9] ICL method. They are similar in strength to the WPC evolved by Lucas and Runarsson (LR). In direct competition, they outperform LR by a small but statistically significant margin. Players evolved using the ICL method were strongest against other random players, as might be expected, but ICL was only able to produce average rather than strong players when measured against non-random players, as shown in Table VI.

A particular problem arises when comparing the performance of a set of fixed value function players due to the inherent intransitivities that tend to arise as a result of their imperfect play, coupled with an inability to adapt. As already noted, performance of a given player is only defined with respect to its opponents. But a further point worth stressing is that filling a league of players with slight variations of a strong player's nemesis can make that strong player look extremely weak. At this stage we merely draw attention to this point. A solution we propose for future development is to evaluate the behavioural diversity of a set of players by comparing the moves they choose, given a large number of game positions. Then, a behavioural diversity constraint could be placed on the league whereby every player must differ from every other player by some threshold percentage of moves.

We used CMA-ES as the basis of our evolutionary algorithm and this gave good results, despite the fact that it typically created weights with very large numbers - we were able to amend this after the experiments were performed by fine tuning CMA-ES parameters. Following on from the previous paragraph that mentioned behavioural diversity, we believe there is also significant scope to develop behavioural variation operators for improved search in value function decision space. For example, it would be possible to define a mutation operator that generated mutants with a desired level of behavioural diversity from the parent. This could be done by analysing the statistical dependence of decision diversity on mutation distance.

Finally, the main points of this paper are that measuring fitness accurately and coping with the effects of intransitivities are of paramount importance when coevolving game players. This is especially true when evolution deals with value functions for use in non-adaptive players, which is the most widely used approach. In future it would also be interesting to evolve adaptive players that remembered past blunders against particular opponents; a small step toward more human-like play.

Acknowledgment

This research is partly funded by a postgraduate studentship from the Engineering and Physical Sciences Research Council.

References

[1] J. Paredis, "Coevolutionary computation," *Artificial Life*, vol. 2, no. 4, pp. 355–375, 1995.

[2] R. Dawkins and J. Krebs, "Arms races between and within species," *Proceedings of the Royal Society of London. Series B, Biological Sciences*, vol. 205, no. 1161, pp. 489–511, 1979.

[3] D. Cliff and G. Miller, "Co-evolution of pursuit and evasion II: Simulation methods and results," *From animals to animats*, vol. 4, pp. 506–515, 1996.

[4] R. Watson and J. Pollack, "Coevolutionary dynamics in a minimal substrate," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*. Morgan Kaufmann, 2001, pp. 702–709.

[5] M. Buro, "An Evaluation Function for Othello Based on Statistics," NEC, 4 Independence Way, Princeton, NJ, Tech. Rep. NECI Technical Report 31, 1997.

[6] S. M. Lucas, "Learning to Play Othello with N-Tuple Systems," *Australian Journal of Intelligent Information Processing*, vol. 4, pp. 1–20.

[7] E. Manning, "Using Resource-Limited Nash Memory to Improve an Othello Evaluation Function," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 2, no. 1, pp. 40 –53, 2010.

[8] S. Lucas and T. Runarsson, "Temporal difference learning versus co-evolution for acquiring othello position evaluation," in *Computational Intelligence and Games, 2006 IEEE Symposium on*. IEEE, 2006, pp. 52–59.

[9] S. Y. Chong, P. Tino, D. C. Ku, and X. Yao, "Improving generalization performance in co-evolutionary learning," *Evolutionary Computation, IEEE Transactions on*, vol. 16, no. 1, pp. 70 –85, feb. 2012.

[10] J. Pearl, "The solution for the branching factor of the alpha-beta pruning algorithm and its optimality," *Communications of the ACM*, vol. 25, no. 8, pp. 559–564, 1982.

[11] D. Michie, "Game-playing and game-learning automata," *Advances in programming and non-numerical computation*, pp. 183–200, 1966.

	SP, $\|A\|$ = 4	$\|A\|$ = 4	LR	Heuristic	ICL-50,000	$\|A\|$ = 1	ICL-500	Anti-Heuristic	Random	% won	BayesElo
SP, $\|A\|$ = 4		10519	10890.5	11122.5	14179.5	14039.5	15356	14982.5	17526.5	68	1756
$\|A\|$ = 4	9481		9507.5	11215	13339	13064	15018.5	15168.5	17367.5	65	1731
LR	9109.5	10492.5		10896	13824	12635	15266.5	14415	17416.5	65	1731
Heuristic	8877.5	8785	9104		11946.5	12868.5	14025	7394.5	16109.5	56	1653
ICL-50,000	5820.5	6661	6176	8053.5		9079.5	13042	16864.5	18297	52	1622
$\|A\|$ = 1	5960.5	6936	7365	7131.5	10920.5		12162.5	15360.5	16785	52	1616
ICL-500	4644	4981.5	4733.5	5975	6958	7837.5		12875.5	17529	41	1527
Anti-Heuristic	5017.5	4831.5	5585	12605.5	3135.5	4639.5	7124.5		14082.5	36	1481
Random	2473.5	2632.5	2583.5	3890.5	1703	3215	2471	5917.5		16	1282

TABLE VI: Full round robin tournament of 10 players for 20,000 games. Note how poorly the anti-heuristic player performs against everybody else (and how well it performs against the heuristic). The tournament measurements for transitivity are $\tau = 0.5535$ and $D_{KL} = 0.0067$, suggesting strong intransitivities (as is evident in the table).

[12] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret minimization in games with incomplete information," *Advances in Neural Information Processing Systems*, vol. 20, pp. 1729–1736, 2008.

[13] M. Osborne and A. Rubinstein, *A course in game theory*. The MIT press, 1994.

[14] W. Chase and H. Simon, "Perception in Chess," *Cognitive Pysychology*, vol. 4, pp. 55–81, 1973.

[15] A. E. Elo, *The rating of chess players: Past and present*. Arco Publishing, New York, 1978.

[16] R. Bradley and M. Terry, "Rank analysis of incomplete block designs. i. the method of paired comparisons." *Biometrika*, vol. 39, p. 324345, 1952.

[17] D. R. Hunter, "MM Algorithms for Generalized Bradley-Terry Models," *The Annals of Statistics*, vol. 32, p. 384406, 2004.

[18] M. E. Glickman and S. T. Jensen, "Adaptive Paired Comparison Design," *Journal of Statistical Planning and Inference*, vol. 127, pp. 279 – 293, 2005.

[19] M. Adler, P. Gemmell, M. Harchol, R. M. Karp, and C. Kenyon, "Selection in the presence of noise: The design of playoff systems," in *Proc. 5th Symposium on Discrete Algorithms*, 1994, pp. 564–573.

[20] W. D. Smith, "Rating systems for gameplayers, and learning," NEC, 4 Independence Way, Princeton, NJ, Tech. Rep. 93-104-3-0058-5, 1994.

[21] R. Herbrich, T. Minka, and T. Graepel, "TrueSkill(TM): A Bayesian Skill Rating System," vol. 20, 2007.

[22] O. Frank and F. Harary, "Cluster inference by using transitivity indices in empirical graphs," *Journal of the American Statistical Association*, pp. 835–840, 1982.

[23] S. Nolfi and D. Floreano, "Coevolving Predator and Prey Robots: Do "Arms Races" Arise in Artificial Evolution?" *Artificial Life*, vol. 4, no. 4, pp. 311–335, 1998.

[24] E. de Jong, "Intransitivity in coevolution," in *Parallel Problem Solving from Nature-PPSN VIII*. Springer, 2004, pp. 843–851.

[25] E. Popovici, A. Bucci, R. P. Wiegand, and E. D. de Jong, "Coevolutionary Principles," *Handbook of Natural Computing*, 2010.

[26] W. Konen and T. Bartz-Beielstein, "Reinforcement learning for games: failures and successes," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. ACM, 2009, pp. 2641–2648.

[27] S. Ficici and J. Pollack, "A game-theoretic approach to the simple coevolutionary algorithm," in *Parallel Problem Solving from Nature PPSN VI*. Springer, 2000, pp. 467–476.

[28] S. Chong, M. Tan, and J. White, "Observing the evolution of neural networks learning to play the game of othello," *Evolutionary Computation, IEEE Transactions on*, vol. 9, no. 3, pp. 240 – 251, june 2005.

[29] E. de Jong, "The incremental pareto-coevolution archive," in *Genetic and Evolutionary Computation–GECCO 2004*. Springer, 2004, pp. 525–536.

[30] S. Samothrakis and S. Lucas, "Approximating N-player behavioural strategy Nash equilibria using coevolution," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 2011, pp. 1107–1114.

[31] E. De Jong, "The MaxSolve algorithm for coevolution," in *Proceedings of the 2005 conference on Genetic and evolutionary computation*. ACM, 2005, pp. 483–489.

[32] M. Buro, "Takeshi murakami vs. logistello," *International Computer-Chess Association Journal*, vol. 20, no. 3, pp. 189–193, 1997.

[33] S. I. T. Yoshioka and M. Ito, "Strategy acquisition for the game Othello based on reinforcement learning," *IEICE Transactions on Information and Systems E82-D 12*, pp. 1618 – 1626, 1999.

[34] S. Y. Chong, P. Tino, and X. Yao, "Measuring generalization performance in coevolutionary learning," *Evolutionary Computation, IEEE Transactions on*, vol. 12, no. 4, pp. 479 –505, aug. 2008.

[35] ——, "Relationship between generalization and diversity in coevolutionary learning," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 1, no. 3, pp. 214 –232, sept. 2009.

[36] N. Hansen, S. Muller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[37] D. Cliff and G. Miller, "Visualizing coevolution with CIAO plots," *Artificial Life*, vol. 12, no. 2, pp. 199–202, 2006.

Spyridon Samothrakis is currently pursuing a PhD in Computational Intelligence and Games at the University of Essex. His interests include game theory, computational neuroscience, evolutionary algorithms and consciousness.

Simon Lucas (SMIEEE) is a professor of computer science at the University of Essex (UK) where he leads the Game Intelligence Group. His main research interests are games, evolutionary computation, and machine learning, and he has published widely in these fields with over 160 peer-reviewed papers, mostly in leading international conferences and journals. He was chair of IAPR Technical Committee 5 on Benchmarking and Software (2002 - 2006) and is the inventor of the scanning n-tuple classifier, a fast and accurate OCR method. Professor Lucas is the founding Editor-in-Chief of the IEEE Transactions on Computational Intelligence and AI in Games.

Thomas Philip Runarsson (SMIEEE) is a Professor of Operations Research with the School of Engineering and Natural Sciences, University of Iceland. His research interests include evolutionary computation, global optimization, games, and statistical learning. His work in constraint handling in evolutionary optimization has been well recognized and cited by the scientific community.

David Robles did his MSc in Computer Science at the University of Essex in 2008. He is now pursuing a PhD in Computer Science in the area of artificial intelligence in games.