Improving Coevolution by Random Sampling

Wojciech Jaśkowski, Paweł Liskowski, Marcin Szubert and Krzysztof Krawiec
Institute of Computing Science, Poznan University of Technology
Piotrowo 2, 60965 Poznań, Poland
{wjaskowski,pliskowski,mszubert,kkrawiec}@cs.put.poznan.pl

ABSTRACT

Recent developments cast doubts on the effectiveness of coevolutionary learning in interactive domains. A simple evolution with fitness evaluation based on games with random strategies has been found to generalize better than competitive coevolution. In an attempt to investigate this phenomenon, we analyze the utility of random opponents for one- and two-population competitive coevolution applied to learning strategies for the game of Othello. We show that if coevolution uses two-population setup and engages also random opponents, it is capable of producing equally good strategies as evolution with random sampling for the expected utility performance measure.

To investigate the differences between analyzed methods, we introduce *performance profile,* a tool that measures the player's performance against opponents of various strength. The profiles reveal that evolution with random sampling produces players coping well with mediocre opponents, but playing relatively poorly against stronger ones. This finding explains why in the round-robin tournament, evolution with random sampling is one of the worst methods from all those considered in this study.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Algorithms, Experimentation

Keywords

Competitive Coevolution, Solution Concepts, Othello, Maximization of Expected Utility, Strategy Learning, Performance Profile

1. INTRODUCTION

For two decades coevolutionary learning (CEL) has been perceived as a method-of-choice for learning in interactive domains such as game playing. It has attracted substantial interest and has been successfully applied to many games [1, 13, 12] mainly because it does not require handcrafted opponents or game databases to learn from. Coevolution is a knowledge-free method [21]: the process of strategy learning is driven solely by playing games between individuals and by the survival of the fittest.

Recently, Chong *et al.* proposed Improved Coevolutionary Learning [5] and showed that it significantly surpasses coevolutionary learning on the game of Othello. Their method is a straightforward knowledge-free variant of evolution strategy, in which individuals are evaluated through playing games against random players. Since the individuals do not confront each other, the use of the term 'coevolution' for this method is disputable, so within this study we refer to it as Random Sampling Evolutionary Learning (RSEL).

CEL has been known for its self-adapting capabilities [1], thus the experimental evidence that such a simple algorithm as RSEL performs better might have surprised not only the authors of this paper. In response, this study attempts to understand this phenomenon and show in which respects coevolution can outrank RSEL.

Our main contribution is demonstrating that CEL can be as effective as RSEL at finding good Othello players with respect to the performance measure of expected utility, but it is better at playing against the strong opponents. We show that this is possible for a two-population variant of CEL that additionally uses random players. We do not limit our attention to players' overall performance, but also scrutinize how they fare against opponents of different strength. To illustrate this, we propose *performance profile* plots, a new tool that gives insight into this characteristics.

2. BACKGROUND

An essential prerequisite for comparing algorithms that learn to play games is deciding what the goal of learning is. Simply saying that the goal is to learn the 'best strategy' is vague. The best strategy could be the one that maximizes the expected game outcome or the one that minimizes the worst possible loss. In order to be unambiguous in this respect, one has to choose a formal *solution concept* [9], which precisely defines which candidate solutions are solutions to the problem and which are not.

Ficici [9] postulated that solution concept is a part of a problem, and that in order to solve the problem while avoiding coevolutionary pathologies, an algorithm must *implement* the assumed solution concept. This holds when an algorithm monotonically approximates the concept, i.e., candidate solutions do not get worse with time with respect to an *performance measure* related to the assumed solution concept. Due to the nature of interactive domains and solution concepts, providing such a guarantee usually requires an unbounded archive of candidate solutions. Thus, the practically useful algorithms formally do not implement any solution concept, but are *designed for* one of them, yet without any formal guarantees of monotonic approximation. In this spirit, de Jong proposed the MaxSolve algorithm [6], which was designed for the Maximization of Expected Utility (MEU) solution concept [24], but does not implement it[1].

MEU, as opposed to other solution concepts such as Pareto-optimal set [7], has a natural real-valued performance measure: the player's expected utility, which is the expected game outcome against all possible opponents from a given search space. This measure can be easily estimated: one generates a number of random strategies from the search space and calculates the average result of games played against them. Clearly, this performance measure is consistent with the way the individuals are evaluated in RSEL, since RSEL was designed for MEU.

In contrast, CEL, a straightforward coevolutionary learning used recently as a control approach for RSEL [5], does not implement any known solution concept, nor it was designed for any, in particular not for MEU. From this perspective, the RSEL supremacy over CEL should not astound, because the two algorithms have been compared using a measure known as generalization performance in machine learning, which is merely a different name for expected utility.

However, the fact RSEL is better than CEL against an average opponent does not imply it can beat every opponent defeated by CEL and vice versa. Opponents vary in strength, and some methods may be more apt to produce players that win against, e.g., strong opponents than the weak ones. In MEU, such differences can cancel each other or become neglected because of the very uneven distribution of opponent strength, causing the methods to be apparently similar. One of the objectives of this paper is to investigate these deficiencies of MEU and obtain a more adequate picture of method performance.

3. RELATED WORK

The game of Othello is a simple but challenging testbed that has been widely used to analyze and compare different learning methods and player representations. Most approaches to develop Othello-playing programs were based on external domain knowledge (e.g., in the form of expert-labeled game databases) and efficient game tree search algorithms [3]. However, recent research has seen the advent of methods that learn autonomously, without any help of external domain knowledge. Typical examples of such methods are CEL and Temporal Difference Learning, which were applied to Othello separately [4, 19, 28] as well as in combination [27, 22]. Our work follows this direction of knowledge-free methods and focuses on CEL.

Apart from the learning algorithm, an important aspect that influences the strength of Othello strategies is player representation. A good overview of different representations and their performance is provided by the Othello Position Evaluation Function League [17]. Besides the simplest weighted piece counter (WPC) representation, more complex ones include: a symmetric n-tuple network [18], a multi-layer perceptron (MLP) [2], and a spatial MLP [4]. League rankings indicate that of these representations, n-tuple networks can achieve the highest performance against the predefined heuristic player. Nonetheless, in this work we decided to use the WPC encoding, because our goal is to consistently confront a few learning methods in fair conditions rather than to develop a player that is strong in absolute terms.

The players trained by particular methods have to be objectively compared. Most related studies use a score against a single benchmark opponent as a performance measure. Typical examples of such opponents are the heuristic player introduced by Yoshioka *et al.* [29], the positional and mobility strategies from Iago program [25], and the random player which makes a random legal move in each turn. An alternative approach was employed by Chong *et al.* [5] who compared CEL and RSEL with respect to their *generalization performance* measured as an average score against a very large number of random WPC-encoded players. Yet another method is direct confrontation of the players produced by compared methods in, e.g., a round-robin tournament [14]. In this paper, we use expected utility, head-to-head tournament, and a novel tool of performance profiles to assess the quality of players.

4. METHODS

The conceptual framework of this study consists of three elements: the definition of the game, the representation of strategies, and the algorithms that learn to play the game. We detail them in the following.

4.1 Othello and WPC encoding

The game of Othello is a deterministic, perfect information, zero-sum board game played by two players on an 8×8 board. There are 64 identical pieces which are white on one side and black on the other, with the colors representing players. The game starts with the four central squares of the board occupied with two black and two white pieces. Players make moves alternately by placing their pieces on the board until it is completely filled or until neither of them is able to make a legal move. The location to place a piece on has to fulfill two conditions. Firstly, it must be adjacent to an opponent's piece. Secondly, the new piece and some other piece of the current player must form a vertical, horizontal, or diagonal line with a continuous sequence of opponent's pieces inbetween. After placing the piece, all such opponent's pieces are flipped. A legal move requires flipping at least one of the opponent's pieces. The objective of the game is to have the majority of pieces on the board at the end of the game. If both players have the same number of pieces on the board, the game ends in a draw.

When designing an algorithm that learns to play Othello, one of several strategy representations that vary in complexity may be adopted (see Section 3). Here we focus on the

[1]To be precise, de Jong uses the same name MaxSolve for two algorithms: one with formal guarantees of monotonicity and its bounded version without them [6].

arguably simplest of them, position-weighted piece counter (WPC). WPC is a linear weighted board evaluation function which implements the state evaluator concept, i.e., it is explicitly used to evaluate how desirable a given board state is. It assigns a weight w_i to a board location i and uses scalar product to calculate the utility f of a board state $\mathbf{b}$:

$$f(\mathbf{b}) = \sum_{i=1}^{8\times 8} w_i b_i,$$

where b_i is 0 in the case of an empty location, +1 if a black piece is present or -1 in case of a white piece. The players interpret $f(\mathbf{b})$ conversely: the black player prefers moves leading towards states with a higher value, whereas lower values are favored by the white player.

All methods considered in this paper employ WPC as a state evaluator in a 1-ply setup: given the current state of the board, the player generates all legal moves and applies f to the resulting states. The state gauged as the most desirable determines the move to be made. Ties are resolved at random.

4.2 Random Sampling Evolutionary Learning (RSEL)

The approach we refer to here as Random Sampling Evolutionary Learning (RSEL), proposed by Chong *et al.* [5] under the name of Improved Coevolutionary Learning, is a $(\mu+\lambda)$ generational evolution strategy. The algorithm starts with a population of μ randomly generated players (vectors of 64 real-valued weights of WPC strategies in our case). In every generation, each of the μ fittest individuals produces λ/μ offspring through a mutation operator (thus, all populations except for the initial one consist of μ parents and λ offspring of those parents).

Fitness calculation in RSEL is straightforward. A sample of random WPC strategies is generated, the individual plays against each of them, and the average game result determines its fitness. Thus, individual's fitness in RSEL is an estimate of the expected utility of the strategy it encodes.

4.3 Coevolutionary Learning (CEL)

While coevolutionary algorithms borrow the overall mechanics from evolutionary algorithms, their distinctive feature is an explicit use of interactions between individuals for fitness assessment. By interacting, individuals in population directly influence each other's fitness, as opposed to evolutionary algorithms, where their evaluation is independent (and in this sense external and objective). From a few variants of this scheme proposed in past studies, we employ *competitive coevolution*, which is particularly useful when an objective evaluation function is difficult to compute. This is the case when learning game strategies: exact calculation of individual's performance requires playing a large or an infinite number of games, thus it is infeasible. However, it can be conveniently substituted by a rough assessment based on interactions within a population. The outcomes of such interactions determine fitness, called *competitive fitness* in this context. By engaging players in the mutual pressure to outperform each other, coevolutionary learning intends to provide an adaptive learning gradient that might otherwise be hard to obtain [23].

An important choice in designing coevolutionary algorithms is the interaction scheme. For symmetrical problems such as the game of Othello, the typical approach is *one-population coevolution* [20] which consists in evolving individuals in a single population, making them compete directly with each other. Although one-population coevolution has been intensely exploited in the context of games, some research [15, 8] suggests that it is worth to separately maintain two types of individuals: *candidate solutions* or *learners*, which are expected to improve over time as evolution proceeds, and *tests* or *trainers*, which main purpose is to differentiate the candidate solutions by defeating some of them and losing against the others. Thereby, coevolution is allowed to adaptively select the tests used for evaluation.

To ensure comparability with RSEL, coevolutionary learning (CEL) in this study is a $(\mu+\lambda)$ evolutionary strategy equipped with competitive fitness, calculated either via a round-robin tournament among individuals for one-population coevolution, or by confronting all candidate solutions with all tests in case of two-population coevolution. Therefore, the only difference between CEL and RSEL lies in the method of fitness assessment.

5. THE EXPERIMENT

The objectives of the experiment are twofold. Firstly, we want to determine which of the considered methods (RSEL, various configurations of CEL, and hybrids thereof) yields better Othello players given a fixed computational budget. To this aim, we employ the expected utility performance measure and round-robin tournament. Our second goal is to explain the anticipated differences by profiling the strategies using opponents of varying difficulty.

In order to fairly compare the algorithms, we set them up so that the total computational effort as well as the computational effort per one generation are equal between methods. Following other studies [16, 6], we identify the computational effort with the number of games played in interactions among individuals.

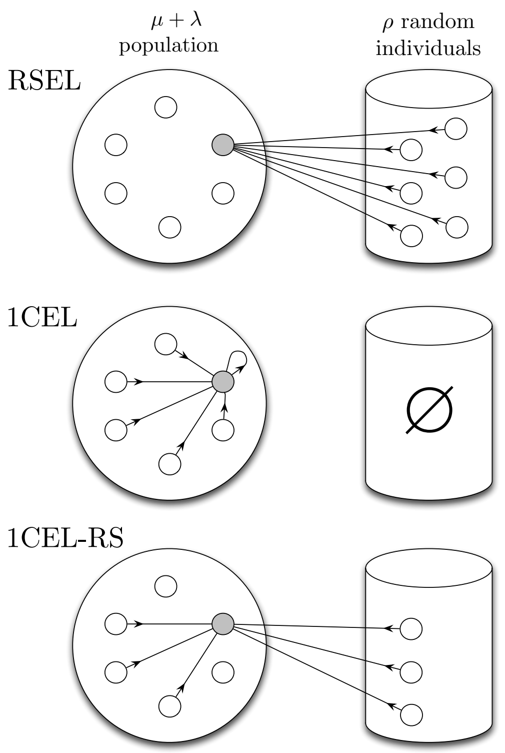
A single interaction is a double game, where both individuals play one game as black and one game as white player. In each game, one point is divided between players: the winner gets 1 point and the loser 0 points, or they get 0.5 points each in case of a draw. Each evolutionary run consists of 200 generations, and 5,000 games are played in each of them (2,500 double games), which adds up to the total effort of 1,000,000 games per run.

All methods start with an initial population filled with individuals whose weights are randomly drawn from $[-0.2, 0.2]$. The only search operator used by all algorithms is a simple mutation that perturbs all the weights with additive noise. The WPC weight $w_i^{'}$ of the offspring is obtained by adding a small random value to the corresponding WPC weight of the parent:

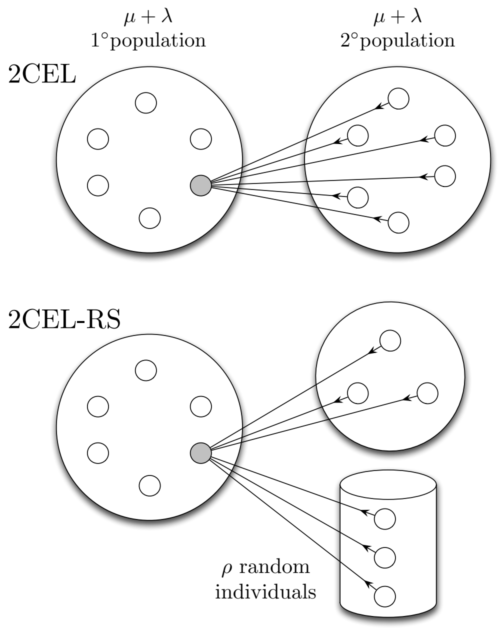
$$w_i^{'} = w_i + 0.1 \cdot U[-1, 1],$$

where $U[-1,1]$ is a real number drawn uniformly from the interval $[-1,1]$. Weights resulting from mutation are clamped to the interval $[-10,10]$. Consequently, the space of strategies we consider is a $[-10,10]^{64}$ hypercube.

Some of the setups and performance assessment methods employ *random WPC players*. Each such player is obtained independently by drawing weights uniformly from the interval $[-10,10]$. In the following, by 'random player/opponent' we mean a WPC player obtained in this way.



(a) RSEL and 1CEL



$\mu+\lambda$ 1°population
$\mu+\lambda$ 2°population
2CEL
2CEL-RS
ρ random individuals

(b) 2CEL

Figure 1: The visualization emphasizes differences in fitness assignment among methods considered in the paper. An arrow means that a game is played between two players.

We emphasize that all setups use the same evolutionary operators of selection and mutation and differ only in the way fitness is assigned to individuals. In the following subsections we detail the setups of particular algorithms, and Figs. 1a and 1b illustrate their interaction schemes.

5.1 RSEL Setup

RSEL is a straightforward implementation of Random Sampling Evolutionary Learning algorithm described in Section 4.2, where $\mu = 25$ and $\lambda = 25$. During the evaluation phase, each individual is evaluated against a set of ρ random opponents. To fix the effort at 5,000 games per generation, we set $\rho = 50$ ($50 \times 50 \times 2 = 5000$). Figure 1a illustrates this configuration. We emphasize that a collection of random opponents is drawn anew in every generation.

5.2 1CEL Setups

1CEL is a one-population coevolutionary algorithm which implements the competitive $(\mu + \lambda)$ evolutionary strategy scheme outlined in Section 4.3. All individuals in population play double games with each other (round-robin tournament). In each generation, $\mu = 25$ best performing strategies produce via mutation $\lambda = 25$ children. The nature of this competitive fitness assessment is illustrated in Fig. 1a.

The second method, **1CEL-RS**, is a hybrid of 1CEL and RSEL that combines the competitive fitness with random sampling. Technically, each individual is evaluated on the basis of double games with 25 randomly selected individuals from the population (as in 1CEL) and $\rho = 25$ random opponents (as in RSEL). This configuration is illustrated in Fig. 1a.

5.3 2CEL Setups

2CEL is a two-population competitive coevolutionary algorithm in which individuals are bred in two separate populations. The first population contains candidate solutions, while the second one maintains tests, which here take the form of opponent strategies that challenge candidate solutions. The fitness of a candidate solution is the sum of points it obtains in double games with all tests.

Tests, in turn, are rewarded for making *distinctions* between candidate solutions [11]. The test's fitness is the weighted sum of points it receives for making distinctions. A test makes a distinction for a given pair of candidate solutions if the games played with it give different outcomes. To maintain diversity in the population, we employ *competitive fitness sharing* [26]; each point for distinction is weighted by the inverse of the number of tests that make this distinction. Consequently, genetic novelty is preferred, since tests that make unique distinctions are rewarded more than the ones that make the same distinctions as other tests in the population.

To guarantee fair comparison among the studied algorithms, the selection scheme in the population of candidate solutions remains the same as in 1CEL. The population of tests uses $(\mu + \lambda)$ evolutionary strategy, where $\mu = 25$ and $\lambda = 25$. Figure 1b illustrates how the fitness is assigned to a candidate solution in 2CEL.

Table 1: Performance comparison of best-of-run players. All t-Student tests have been performed against RSEL.

Algorithm	Performance [%]	t-value	p-value
RSEL	.86.46 ± 0.25	-	-
2CEL-RS	.86.44 ± 0.26	-20.30	3.73×10^{-54}
1CEL-RS	.83.63 ± 0.37	-12.63	1.29×10^{-28}
1CEL	.80.34 ± 0.54	-20.53	6.62×10^{-55}
2CEL	.79.97 ± 0.58	-0.09	0.46

The last method, **2CEL-RS**, is 2CEL enhanced by random sampling. While the population of candidate solutions still consists of 50 individuals, the population of tests is limited to 25 individuals. The fitness of a candidate solution is the sum of points obtained in 25 double games with the tests and 25 double games with the random opponents (cf. Fig. 1b).

6. RESULTS

We performed 120 runs for each method. In the following, the best-of-generation individual is the individual with the highest fitness in the population. By the best-of-run player we mean the best-of-generation player of the last generation.

6.1 Performance Comparison

To assess the individuals we use the approximated measure of expected utility. To calculate it, we let every individual play 25,000 double games (50,000 games in total) against random WPC players, obtained by drawing weights uniformly from the interval $[-10, 10]$. With one point for winning the game, zero for losing, and 0.5 for a draw, the expected utility of a player is in the range of $[0, 1]$, but for clearer presentation we report it in percent points. From now on, the term 'performance' refers to this measure.

Figure 2 shows how the performance of each method changes as a function of computational effort (which is here proportional to the number of generations). Each point on the plot is the performance of method's best-of-generation player averaged over 120 runs. Table 1 compares the methods in terms of the average performance of the best-of-run individuals accompanied by 95% confidence intervals, t-Student test values and p-values. The two-tail t-test compares RSEL with other methods.

Our results confirm Chong's *et al.* conclusion that evolution with fitness purely based on random sampling (RSEL) is significantly better than one-population coevolution [5]. The new result is that it is also much better than two-population coevolution. Although the difference between 2CEL and 1CEL is not statistically significant and some parts of the curves in Fig. 2 overlap, the comparison of setups that mix coevolution with random sampling, i.e. 2CEL-RS and 1CEL-RS, shows that separating learners from trainers in two-population setup pays off with respect to the expected utility.

The 1CEL-RS setup can be viewed as a hybrid of RSEL and 1CEL, thus it is not surprising to see its performance falling in between those two [5]. What is not obvious, however, is that a similar effect is cannot be observed in the case of 2CEL: RSEL and 2CEL-RS produce equally good strategies on average.

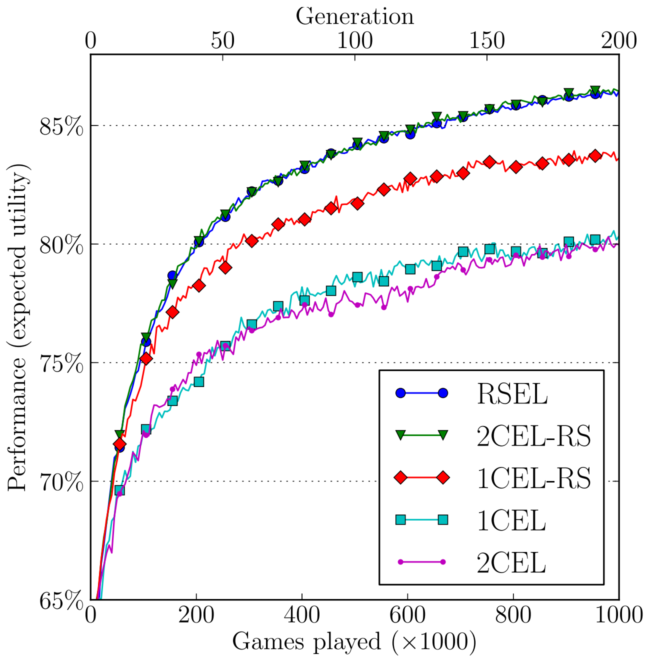


Figure 2: Comparison of methods over time in terms of expected utility performance measure.

However, while having roughly the same expected performance against opponents, do 2CEL-RS and RSEL differ in their capability of winning against opponents of different strength? In an attempt to answer this question, we will break down the performance value with performance profiles.

6.2 Analysis with Performance Profiles

To better understand characteristics of particular methods we devise *performance profile*. This tool gives us better insight into how a strategy copes with opponents of different strength.

To prepare a performance profile, we randomly generate 500,000 players (opponents) by sampling WPC weights uniformly and independently from the $[-10, 10]$ interval. Next, the performance of each opponent is estimated by playing 1,000 double games with random WPC strategies. The range of possible performance values, i.e., $[0, 1]$, is then divided into 100 bins of equal width, and each opponent is assigned to one of these bins based on its performance.

Building the opponents database is computationally expensive: here it required playing $500{,}000 \times 1{,}000 \times 2 = 1{,}000{,}000{,}000$ games. However, once created, it can be reused[2].

The ensemble of opponents partitioned into bins forms the basis for building the profile. The assessed player plays double games with all the opponents from each bin, and the average game outcome is plotted against the bins.

We apply performance profiling to inspect the best-of-run individuals of all algorithms considered in this paper and present them in Fig. 3. Since we have 120 runs per method, we average the profiles over 120 best-of-run players. A point of coordinates (x, y) in a plot means that the

[2]The data and Java code for creating performance profiles for Othello are available at `http://www.cs.put.poznan.pl/wjaskowski/projects/performance-profiles`.

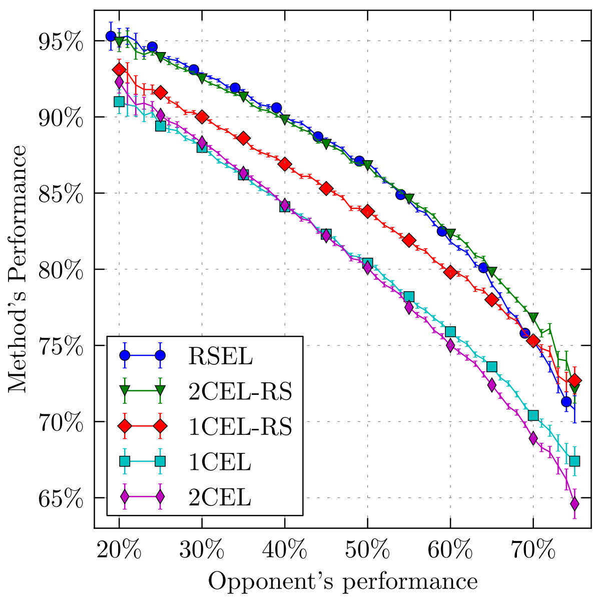


Figure 3: Performance profiles of five analyzed methods. Each point (x, y) means performance y over opponents of performance x. The whiskers mark 95% confidence interval.

best-of-run individuals have on average performance y when playing against opponents of performance x. For example, the performance of RSEL is about 90% for opponents with performance of 40%.

The whiskers in the plots mark 95% confidence intervals. Notably, they tend to widen towards the ends of plots. This is because it is hard to randomly generate opponents that are very strong or very weak, so the extreme bins contain relatively few opponents. For the same reason we removed the points with confidence intervals that were larger than 20% or were computed on a basis of 120 or fewer double games.

The decreasing trend in each data series confirms an obvious fact that it is harder to win with stronger opponents than with the weaker ones. We can also see that some methods are dominated by others, e.g., 1CEL and 2CEL are dominated by 2CEL-RS. Although 2CEL-RS is most of the time significantly better than 1CEL-RS (which is why it has a much higher performance), the difference between them decreases with the increasing opponent strength and it is hard to tell which one fares better against the strongest players.

What is however clearly visible in the plots is that while RSEL copes well with weak opponents (performance similar to 2CEL-RS), it fares worse when confronted with stronger ones (performance worse than 1CEL-RS and 2CEL-RS). Interestingly, 1CEL-RS has a complementary characteristics, so its plot and RSEL's plot cross at roughly 68% opponent strength. However, since the strong opponents are infrequent, the advantage of 1CEL-RS over RSEL on them cannot compensate its inferior position when it comes to weaker opponents, and its overall performance is still worse (cf. Table 1).

Table 2: The results of the round-robin tournament (percents).

Algorithm	1CEL-RS	2CEL-RS	1CEL	RSEL	2CEL	**Overall**
1CEL-RS	-	53.4%	53.3%	56.3%	56.7%	**54.9%**
2CEL-RS	46.6%	-	51.2%	53.2%	53.9%	**51.3%**
1CEL	46.7%	48.8%	-	51.8%	52.7%	**50.0%**
RSEL	43.7%	46.8%	48.2%	-	52.0%	**47.7%**
2CEL	43.3%	46.1%	47.3%	48.0%	-	**46.2%**

6.3 Round-Robin Tournament Comparison

Our final experiment is a round-robin tournament among all methods. This assessment determines a relative ranking of methods [13] by playing matches between teams of players. Every team consists of 120 best-of-run players produced by an algorithm. Thus, a single match in the tournament involves $120 \times 120 = 14400$ double games. Winning all games gives the round-robin-performance of 100%.

Table 2 presents the results of the tournament. Evolutionary learning with random sampling loses to all other methods except 2CEL in head-to-head matches and its aggregated round-robin-performance is significantly lower than 1CEL-RS, 2CEL-RS, and 1CEL. However, the difference between RSEL and 2CEL is too small to conclude anything for sure. Adding a random sampling component improves both one- and two-population coevolution (1CEL-RS vs. 1CEL and 2CEL-RS vs. 2CEL). Also, one-population variants are consistently better than two-population ones (2CEL vs. 1CEL and 2CEL-RS vs. 1CEL-RS). 1CEL-RS wins against all other algorithms and is clearly the best in terms of round-robin performance.

The ranking in terms of expected utility presented in Table 1 is different than the one resulting from the round-robin tournament. Do the rankings contradict each other?

In order to explain this discrepancy, notice that, contrary to the previous assessments which involved random opponents, here each individual in a team plays only with the players from the opponent teams, and those players (co)evolved to be strong. According to Table 1, the performance of team members is in the range of 79–87% and, as we have seen in Section 6.2, the methods differ in performance profiles. Figure 3 does not provide reliable data on how methods behave for very strong opponents because they are too scarce, but the trends are evident and can be extrapolated. As we already pointed out, RSEL, for instance, matches the performance of 1CEL-RS at opponent strength of 68%, where their curves cross, and then keeps on heading downwards. This explains why it ranks low in the round-robin tournament.

Similarly, performance profile plot shows that 1CEL-RS's curve is the most leveled, matching 2CEL-RS's plot at around 75% opponent performance and, presumably, surpassing it for the stronger opponents. We hypothesize that this may account for the former one scoring 3.6% more in the final tournament ranking.

7. DISCUSSION

Most of our findings confirm the results reported by Chong *et al.* [5]. When gauged with the performance measure of expected utility, evolutionary learning with random sampling indeed provides significantly better results than simple one-

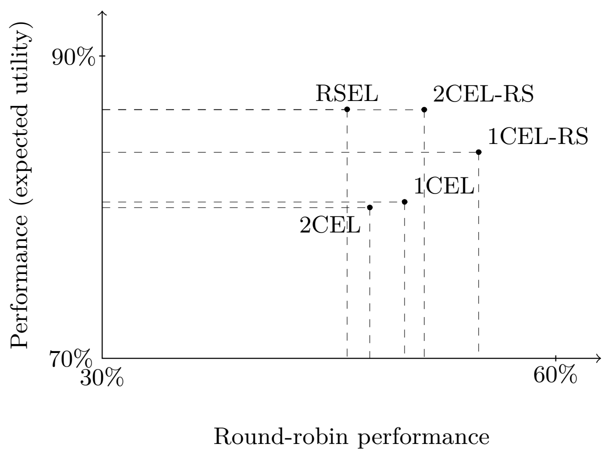


Figure 4: Methods compared in a two-objective manner. The objectives are: i) expected utility performance, and ii) round-robin tournament result.

population coevolution. However, as we have shown, this comes at a price: performance profiles reveal that RSEL, while being able to play well with easy opponents, has troubles when facing the stronger ones. This was further confirmed in the round-robin tournament, where RSEL, faced only with the highly-skilled opponents evolved by competitive methods, finished in the last but one place.

The beneficial role of hybridization with random sampling is puzzling. As we found out when preparing the performance profiles, drawing a strong opponent at random is very unlikely. From the 500,000 random opponents drawn, all had performance lower than 80%. Moreover, most of them are mediocre, so they cannot be challenging for the individuals that were evolving in the population for many generations. Thus, it is not obvious how they could form a useful training gradient. Yet coevolution obtains better results when coupled with random sampling. This effect is especially interesting for 1CEL-RS, which has indisputably won the round-robin tournament. Explaining this effect in detail requires further investigation, but we speculate that random sampling might play a role in preventing forgetting [10], a coevolutionary pathology that hampers learning.

Despite the fact that 1CEL-RS and 2CEL-RS rank among the top three algorithms both for the expected utility criterion and the round-robin tournament, we observe a trade-off of their performance against weak and strong players. This trade-off manifests in the performance profile plot (Fig. 2), but we can alternatively visualize it by plotting all methods against two objectives: the (estimated) expected utility and the round-robin-performance. Figure 4 shows that no method dominates all others on both criteria, however 2CEL-RS is close to dominate RSEL, 1CEL dominates 2CEL, and remarkably, coevolution with random sampling dominates coevolution without it.

This raises a question: which performance measure (and which solution concept in general) is more *practically useful*? Are strategies that play well on average preferable to those which can cope with strong opponents, but sometimes lose with the weak ones? Unfortunately, this question cannot be answered without assuming certain probability distribution of strategies as they occur in the real world, which is in general difficult.

8. CONCLUSIONS

This study presented an evidence that coevolution can offer advantage over evolution with random sampling for learning game strategies. In the particular case of the game of Othello and WPC strategy representation, this advantage was observable in head-to-head confrontation of generated solutions. An algorithm that autonomously and dynamically redefines its own fitness function (CEL) can produce strategies which are better in such confrontation than those produced by an algorithm that relies on (approximately) stationary fitness function (RSEL). However, pure coevolution is not sufficient to attain that, and additional means are required (here: two-population setup and random opponents). It remains to be seen whether analogous conclusions may be drawn for interactive domains other than Othello.

To provide detailed insight into the performance of players considered in this study, we proposed performance profiles. We used this straightforward tool to explain, among others, the differences between the results of the round-robin tournament and the comparison in terms of expected utility. However, it can be potentially useful also for explaining other phenomena, the dynamics of the search process, behavior of other algorithms, and helping design the new ones. Nevertheless, we must admit that in the preliminary variant presented here this tool suffers from certain limitations. First and foremost, the very strong (or very weak) opponents are hard to generate randomly. As a result, performance profile plots show only a partial picture and their reliability on the extremes of performance is low. This could be improved by adding players of certain performance, possibly generated in a non-random fashion (e.g., the strategies learned by CEL or RSEL), at the price of introducing a bias of unknown characteristic. Secondly, more powerful strategy representations, like n-tuples [18], may provide players with performance near 100% against all random opponents. In such cases, performance profiles would cease to differentiate the methods, so it is hard to anticipate how useful they would be in such contexts.

9. ACKNOWLEDGMENT

This work has been supported by grant no. N N519 441939. W. Jaśkowski has been supported by Polish Ministry of Science and Education, grant no. 91–550/DS. M. Szubert has been supported by Polish Ministry of Science and Education, grant no. 2012/05/N/ST6/03152.

10. REFERENCES

[1] R. Axelrod. The evolution of strategies in the iterated prisoner's dilemma. In L. Davis, editor, *Genetic Algorithms in Simulated Annealing*, pages 32–41. Pitman, London, 1987.

[2] K. J. Binkley, K. Seehart, and M. Hagiwara. A Study of Artificial Neural Network Architectures for Othello Evaluation Functions. *Transactions of the Japanese Society for Artificial Intelligence*, 22(5):461–471, 2007.

[3] M. Buro. The evolution of strong othello programs. *Entertainment Computing-Technology and Applications*, pages 81–88, 2003.

[4] S. Y. Chong, M. K. Tan, and J. D. White. Observing the evolution of neural networks learning to play the game of othello. *Evolutionary Computation, IEEE Transactions on*, 9(3):240–251, 2005.

[5] S. Y. Chong, P. Tino, D. C. Ku, and Y. Xin. Improving Generalization Performance in Co-Evolutionary Learning. *IEEE Transactions on Evolutionary Computation*, 16(1):70–85, 2012.

[6] E. D. de Jong. The maxsolve algorithm for coevolution. In H.-G. B. et al., editor, *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, volume 1, pages 483–489, Washington DC, USA, 25-29 June 2005. ACM Press.

[7] E. D. de Jong. A Monotonic Archive for Pareto-Coevolution. *Evolutionary Computation*, 15(1):61–93, Spring 2007.

[8] E. D. de Jong and A. Bucci. DECA: dimension extracting coevolutionary algorithm. In M. C. et al., editor, *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 313–320, Seattle, Washington, USA, 2006. ACM Press.

[9] S. G. Ficici. *Solution concepts in coevolutionary algorithms*. PhD thesis, Waltham, MA, USA, 2004. Adviser-Pollack, Jordan B.

[10] S. G. Ficici. Multiobjective Optimization and Coevolution. In J. Knowles, D. Corne, and K. Deb, editors, *Multi-Objective Problem Solving from Nature: From Concepts to Applications*, pages 31–52. Springer, Berlin, 2008.

[11] S. G. Ficici and J. B. Pollack. Pareto optimality in coevolutionary learning. In J. Kelemen and P. Sosík, editors, *Advances in Artificial Life, 6th European Conference, ECAL 2001*, volume 2159 of *Lecture Notes in Computer Science*, pages 316–325, Prague, Czech Republic, 2001. Springer.

[12] D. B. Fogel. *Blondie24: Playing at the Edge of AI*. Morgan Kaufmann Publishers, Sept. 2001.

[13] W. Jaśkowski, K. Krawiec, and B. Wieloch. Evolving strategy for a probabilistic game of imperfect information using genetic programming. *Genetic Programming and Evolvable Machines*, 9(4):281–294, 2008.

[14] W. Jaśkowski, K. Krawiec, and B. Wieloch. Winning ant wars: Evolving a human-competitive game strategy using fitnessless selection. In M. O. et al., editor, *Genetic Programming 11th European Conference, EuroGP 2008, Proceedings*, volume 4971 of *Lecture Notes in Computer Science*, pages 13–24. Springer-Verlag, mar 2008.

[15] H. Juillé and J. B. Pollack. Coevolving the "ideal" trainer: Application to the discovery of cellular automata rules. In *University of Wisconsin*, pages 519–527. Morgan Kaufmann, 1998.

[16] K. Krawiec, W. Jaśkowski, and M. Szubert. Evolving small-board go players using coevolutionary temporal difference learning with archive. *International Journal of Applied Mathematics and Computer Science*, 21(4):717–731, 2011.

[17] S. Lucas and T. P. Runarsson. Othello Competition; http://algoval.essex.ac.uk:8080/othello/League.jsp.

[18] S. M. Lucas. Learning to play Othello with N-tuple systems. *Australian Journal of Intelligent Information Processing Systems, Special Issue on Game Technology*, 9(4):1–20, 2007.

[19] S. M. Lucas and T. P. Runarsson. Temporal difference learning versus co-evolution for acquiring othello position evaluation. In *IEEE Symposium on Computational Intelligence and Games*, pages 52–59. IEEE, 2006.

[20] S. Luke and R. P. Wiegand. When coevolutionary algorithms exhibit evolutionary dynamics. In A. M. Barry, editor, *GECCO 2002: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, pages 236–241, New York, 2002.

[21] J. Mandziuk. *Knowledge-Free and Learning-Based Methods in Intelligent Game Playing*, volume 276. Springer, 2010.

[22] E. P. Manning. Using resource-limited nash memory to improve an othello evaluation function. *Computational Intelligence and AI in Games, IEEE Transactions on*, 2(1):40–53, 2010.

[23] J. B. Pollack and A. D. Blair. Co-evolution in the successful learning of backgammon strategy. *Machine Learning*, 32(3):225–240, 1998.

[24] E. Popovici, A. Bucci, R. P. Wiegand, and E. D. de Jong. *Handbook of Natural Computing*, chapter Coevolutionary Principles. Springer-Verlag, 2011.

[25] P. Rosenbloom. A world-championship-level othello program. *Artificial Intelligence*, 19(3):279–320, 1982.

[26] C. D. Rosin and R. K. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.

[27] M. Szubert, W. Jaśkowski, and K. Krawiec. Coevolutionary temporal difference learning for othello. In *IEEE Symposium on Computational Intelligence and Games*, pages 104–111, Milano, Italy, 2009.

[28] S. van den Dries and M. A. Wiering. Neural-Fitted TD-Leaf Learning for Playing Othello With Structured Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 23(11):1701–1713, Nov. 2012.

[29] T. Yoshioka, S. Ishii, and M. Ito. Strategy acquisition for the game "othello" based on reinforcement learning. In *ICONIP*, pages 841–844, 1998.