Shaping Fitness Function for Evolutionary Learning of Game Strategies

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

ABSTRACT

In evolutionary learning of game-playing strategies, fitness evaluation is based on playing games with certain opponents. In this paper we investigate how the performance of these opponents and the way they are chosen influence the efficiency of learning. For this purpose we introduce a simple method for shaping the fitness function by sampling the opponents from a biased performance distribution. We compare the shaped function with existing fitness evaluation approaches that sample the opponents from an unbiased performance distribution or from a coevolving population. In an extensive computational experiment we employ these methods to learn Othello strategies and assess both the absolute and relative performance of the elaborated players. The results demonstrate the superiority of the shaping approach, and can be explained by means of performance profiles, an analytical tool that evaluate the evolved strategies using a range of variably skilled opponents.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Connectionism and neural nets, Parameter learning*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; J.m [**Miscellaneous**]:

General Terms

Algorithms

Keywords

Shaping, Fitness Evaluation, Coevolution, Othello

1. INTRODUCTION

The main motivation for this study originates from the question: what can be done to improve the results of an optimization algorithm on a given problem with respect to a specific performance measure? A typical approach is to modify the algorithm by tuning the parameters that affect its behavior. In particular, evolutionary algorithms involve numerous settings (including population size, variation operators, selection scheme) that need to be configured properly to attain satisfactory performance. Since devising such a configuration manually is usually nontrivial, various 'rule-of-thumb' recommendations, good practices and several techniques of automated parameter tuning have been proposed in the past [8]. Ultimately, if the configured algorithm still does not achieve the expected performance, it can be entirely replaced by a different one. Such a choice between a number of algorithms can also be done automatically by hyper-heuristics [3].

It may seem that, apart from searching the space of algorithms and/or their parameters, there are no alternative answers to the question posed in the beginning. Indeed, of the three abovementioned elements (algorithm, problem, performance measure), only the first one appears to be in experimenter's control. However, there is another option: instead of adjusting the algorithm for a particular problem, we can take a complementary approach and modify the problem to make it easier to solve by a particular algorithm. Such approach corresponds to the idea of *shaping* — a core concept of behavioral psychology [24] that already proved useful in, among others, reinforcement learning [21]. Shaping consists in a meaningful modification of the learning problem that brings the learner closer to reaching the behavior of ultimate interest [9]. We adopt shaping here as an umbrella term that characterizes our approach.

In case of evolutionary algorithms and optimization problems, shaping can be applied to two elements of problem definition. First, the search space, which can be shaped by, for instance, adjusting genetic encoding of solutions. Second, the fitness function, which may be purposefully different from the original objective performance measure in order to distort the evaluation of individuals and drive evolution through advantageous paths in the search space.

In this paper, we focus on shaping fitness function for evolutionary learning of game-playing strategies where the objective is to maximize the *expected utility*, i.e. the average outcome against all possible opponents. We investigate three different fitness assessment methods. Two of them are commonly used for such problems and employ random sampling of opponents [5] and coevolution [20]. The third method is a straightforward shaping approach, which we introduce in this paper. The idea is to shape the fitness function by sampling the opponents used for evaluation from a biased performance distribution. We show that using such

GECCO'13, July 6–10, 2013, Amsterdam, The Netherlands.

a function leads to more effective learning and results in stronger players both in absolute and relative terms. In order to verify whether it is actually shaping that drives the evolution towards better solutions, we also consider hybrid fitness evaluation methods which combine the three above approaches. This investigation is conducted with *performance profiles* [13], a novel tool that allows us to analyze the performance of a player with respect to opponents of different strength.

2. METHODS

Learning game-playing strategies is an example of a test-based problem [2, 11], in which candidate solutions are evaluated on a number of test cases such as environments, opponents or examples. In most real-world problems of this class, the large number of possible tests precludes computationally feasible calculation of objective performance measure. For this reason, solving test-based problems with evolutionary computation requires substituting the original performance measure with a computationally cheaper counterpart that drives the search process towards a possibly similar (and preferably the same) goal.

In the case of games, fitness is usually computed by averaging the results achieved against a limited number of tests, i.e., opponent strategies. The question that one needs to answer when designing such a fitness function is: how to choose the opponent strategies? The answer to this question is of utmost importance, as it is exactly the choice of these opponents that allows us to shape the fitness function and, accordingly, guide the learning process.

The set of evaluating opponents and the corresponding fitness function can be either static or dynamic. A static function employs a fixed pool of predefined players (e.g., expert strategies), and thus remains constant throughout evolution. A dynamic function, in contrast, uses a set of opponents that changes in every generation. By exposing players to different opponents each generation, learning algorithms that use dynamic evaluation function can be expected to produce players that win against a wider range of opponents. Here we confront three ways of dynamic opponent selection: random sampling, coevolution, and shaped sampling, the last being the main conceptual contribution of this study.

Before we discuss the algorithms that employ these methods, we need to present the game of Othello, which we use as our testbed, and the representation of strategies for this game. The latter is particularly important, since it directly determines the spaces of possible solutions and tests.

2.1 Othello and the WPC representation

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 (see Fig. 1). 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 position 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 own pieces on the board at the end of the game. If both players have the same number of disks, the game ends in a draw.

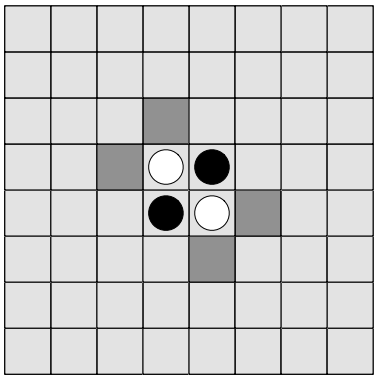

Figure 1: The initial board state of Othello

From the previously applied Othello-playing strategy representations such as multi-layer neural networks [1], spatial neural networks [4], and n-tuple networks [17, 16], we employ the simplest of them, the position-weighted piece counter (WPC) [18, 23]. 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 weight w_i to board location i and uses scalar product to calculate the value 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 case of an empty location, $+1$ if a black piece is present or -1 in case of a white piece. The players interpret $f(\mathbf{b})$ in a complementary manner: 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 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 valuable determines the move to be made. Ties are resolved at random. WPC is used to represent the candidate solutions to the learning problem as well as the opponent strategies employed by a fitness function.

2.2 Random Sampling Evolutionary Learning

The approach we refer here to as Random Sampling Evolutionary Learning (RSEL) was proposed by Chong *et al.* [5] under the name of Improved Coevolutionary Learning. This learning procedure is inspired by a $(\mu + \lambda)$ generational evolutionary strategy. The algorithm begins with a population of μ randomly generated individuals (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).

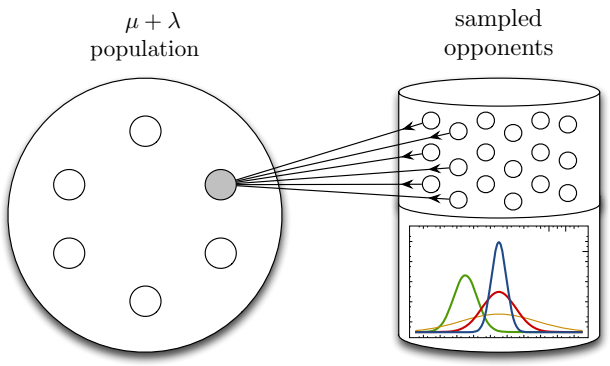

Figure 2: Fitness evaluation based on sampling opponents from a set with certain performance distribution.

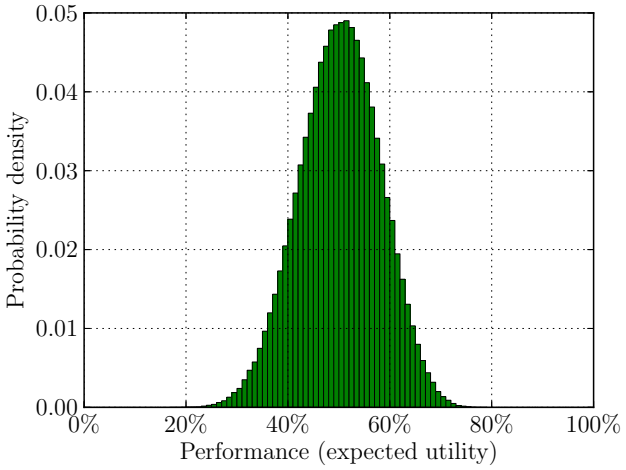

Figure 3: Performance distribution of randomly sampled WPC opponents

The fitness function used in RSEL is based on random sampling of opponents. A sample is drawn once per evaluation phase (i.e., once per generation for a generational evolutionary algorithm) by generating a set of WPC strategies. Afterwards, each individual in the population plays against all the opponents in the sample, and the average score determines its fitness. Consequently, the fitness calculated in RSEL is an unbiased estimator of the expected utility objective function.

Figure 2 schematically illustrates this type of fitness evaluation procedure. The figure highlights the fact that the performance in the population (statistical population), from which the sample is drawn, has certain distribution. This distribution affects the expected strength of evaluating opponents in the sample.

We conducted a preliminary experiment to estimate this distribution. Illustrated in Fig. 3, it was obtained by sampling 500,000 random strategies (in the same way as in RSEL) and calculating their performance as the average game outcome against another set of 1,000 random opponents. Clearly, the performance distribution resembles a normal one with a mean value of 50.2% and a standard deviation of 7.91%.

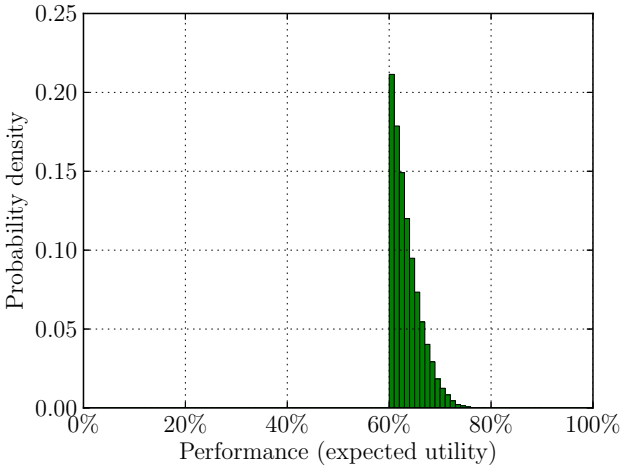

Figure 4: Performance distribution of shaped opponents

2.3 Shaped Sampling Evolutionary Learning

The performance distribution of evaluating opponents in RSEL (Fig. 3) indicates that the majority of WPC strategies achieve mediocre performance. Strong strategies are few and far between. As RSEL samples the opponents uniformly from this distribution, it is unlikely for an individual to face a strong opponent in the evaluation phase. For this reason, fitness function will be often unable to differentiate between two strategies that, e.g., play equally well against the average players but one of them is much better at beating the strong ones. This is unfortunate as, preferably, we would like to lead the evolution towards the strategies that are able to win with skilled players without losing the capability of winning with the opponents of average strength.

Therefore, we propose a method called Shaped Sampling Evolutionary Learning (SSEL), which operates largely like RSEL. The only exception is the fitness assessment phase (cf. Fig. 2), which uses a *biased* sample of opponents, i.e., a sample that is drawn from a different performance distribution. Here, such a shaped sample contains only the opponents that achieve the performance of more than 60%. We populate it in a similar manner in which Fig. 3 was obtained. Its performance distribution is shown in Fig. 4. In each fitness assignment phase, the set of evaluating opponents is randomly selected from the pool.

SSEL's rationale is to shape the fitness function in such a way that it promotes game-playing skills needed for defeating the strong opponents. In this context, RSEL can be considered as a reference, unshaped approach, where the set of evaluating opponents is an unbiased sample of the entire set of WPC strategies, and the performance distribution in the sample follows the the performance distribution of all WPC strategies.

2.4 Coevolutionary Learning

In both evolutionary approaches presented above, the fitness function is designed to reflect individual's *absolute* performance (whether using an unbiased or biased sample of opponents). Individual's absolute performance is, by definition, independent of other individuals in the population. In coevolutionary algorithms, by contrast, fitness function measures the *relative* performance of individuals with re-

spect to other evolving individuals. From many variants of this scheme proposed in past studies, we employ competitive coevolution, which is particularly useful when an objective evaluation function is unknown or difficult to compute. This is the case when learning game strategies: it is computationally infeasible to objectively assess the quality of a strategy, but it is easy to let the individuals interact by playing games. The outcomes of such games determine fitness, which in this context is often called *competitive fitness*.

An important aspect of fitness evaluation in coevolution is the interaction scheme that determines which individuals should interact. For symmetrical problems such as the game of Othello, a popular approach is *one-population coevolution* [19] which consists in evolving individuals in a single, homogeneous population, making them compete directly with each other. Although one-population coevolution has been praised for problem solving advantages and intensely exploited in the context of games, some results suggest that it can be useful to maintain simultaneously two populations of strategies [7, 14]. Each population contains the opponents used for evaluating strategies in the other population — the interactions occur only between individuals that belong to different populations.

In this paper, CEL employs the two-population interaction scheme illustrated in Fig. 5. In contrast to RSEL and SSEL, where fitness assessment involved sampling opponents from a fixed, predefined distribution, here the set of evaluating opponents is coevolving with the population of solutions. Thus, apart from the initial opponent population which is sampled randomly (i.e., from the performance distribution shown in Fig. 3), the distribution of opponents in CEL is not explicitly defined and may vary across generations. Because this influence is reciprocal, the resulting competitive fitness function is thus not only dynamic but also adaptive (details on fitness definition in both population will be provided in the experimental part).

From the shaping perspective, coevolution can be considered as a form of *autonomous shaping* performed on the basis of the current state of the candidate solution population. This stays in contrast to SSEL, where shaping is realized by manual and permanent modification of the opponent pool, and consequently, its performance distribution.

Since mixing multiple learning gradients can speed up learning and make the resulting solutions generalize better [15], we consider also a hybrid fitness evaluation method (see Fig. 5) that combines the competitive fitness employed in CEL with the fitness based on sampling opponents from the performance distributions used in RSEL or SSEL (cf. Figures 3 and 4). These approaches are referred to as CEL-RS and CEL-SS, respectively.

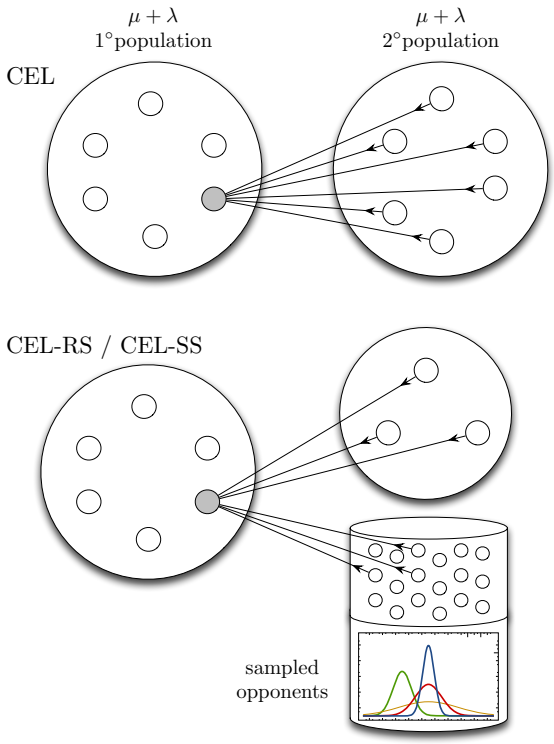

Figure 5: Fitness evaluation employed in coevolutionary learning and in the hybrid approach.

3. THE EXPERIMENT

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

In order to fairly compare the algorithms, we set them up so that their total computational effort as well as the computational effort per one generation are equal among methods. Following other studies [7, 6, 15], 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 to be divided between players: the winner gets one point and the loser zero points, or they get 0.5 point each in the case of draw. Each evolutionary run consists of 200 generations, in each of them $5,000$ games are played ($2,500$ 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 $[-0.2, 0.2]$. They also share the same simple mutation operator that perturbs all the weights using 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 $[-10, 10]$, effectively making the value equal to the respective bound. 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 generated in this way.

We emphasize that in all setups the evolutionary operators of selection and mutation and their parameters are the same. The differences between them lie only in the way fitness is assigned to individuals. In the following subsections we detail the setups of particular algorithms while Figs. 2 and 5 illustrate their interaction schemes.

3.1 RSEL setup

RSEL is a straightforward implementation of Random Sampling Evolutionary Learning algorithm described in Section 2.2, where $\mu = 25$ and $\lambda = 25$. During the evaluation phase, each individual is evaluated against a set of ρ random WPC opponents obtained through random sampling of the space of strategies. The outcomes of these double games are summed to determine individual's fitness. To fix the effort at $5,000$ games per generation, we set $\rho = 50$ ($50 \times 50 \times 2 = 5000$). Figure 2 illustrates this configuration. We emphasize that a collection of random opponents is drawn anew every generation.

3.2 SSEL setup

SSEL works in the same way as RSEL. The only difference between them lies in the fitness evaluation phase (cf. Fig. 2) which uses a different, biased distribution of opponent strategies. The modified player performance distribution contains exclusively the opponents that achieve performance of more than 60%. In order to acquire the strategies that meet this condition, we filtered a set of $500,000$ randomly sampled players, which resulted in a *shaping pool* of $54,454$ players, with performance distribution illustrated in Fig. 4. Individual's fitness is determined on the basis of outcomes of double games with the set of $\rho = 50$ opponents randomly selected from the pool. Note that the computational cost of creating the shaping pool is not included in our results.

3.3 CEL setup

CEL 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. Here, tests take the form of opponent strategies that challenge players from the candidate solutions population.

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 [10]. Test's fitness is the weighted number of points it receives for making distinctions. A test makes a distinction for a given pair of candidate solutions if the games it played against them gave different outcomes. To maintain diversity in the population, we employ a variant of *competitive fitness sharing* [22]; each point awarded for a distinction is weighted by the inverse of the number of tests that made that distinction. This extension promotes diversity in population, since a test that uniquely makes a distinction is rewarded more than the one that shares its distinction with many other tests.

To guarantee a fair comparison among all the studied algorithms, the selection scheme in the population of candidate solutions is the same as in RSEL and SSEL. The population of tests uses $(\mu+\lambda)$ evolutionary strategy, where $\mu = 25$ and $\lambda = 25$. Figure 5 illustrates how the fitness is assigned to a candidate solution in CEL.

3.4 Hybrid setups

The hybrid setups, referred to as **CEL-RS** and **CEL-SS**, combine the competitive fitness with the fitness based on sampling opponents from the performance distribution used in RSEL and SSEL respectively. Technically, each individual is evaluated on the basis of double games played against 25 individuals from the population of tests (as in CEL) and 25 random opponents drawn either from a uniform distribution (CEL-RS) or from the shaping pool (CEL-SS). This configuration is illustrated in Fig. 5. Note that while the population of candidate solutions still consists of 50 individuals, the population of tests is limited to 25 individuals.

4. RESULTS

We performed 30 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. We identify method's performance with the performance of its best-of-generation players.

4.1 Absolute Performance

To objectively assess the individuals we use the approximate measure of expected utility. This performance measure is the percentage of points (with 1.0 point for winning the game and 0.5 for a draw) obtained in $25,000$ double games ($50,000$ games in total) against the random WPC players, generated by drawing weights uniformly from the interval $[-10, 10]$. From now on, the term 'performance' refers to this absolute measure.

Figure 6 shows the performance of each method as a function of computational effort (which is here proportional to the number of generations). Each point in the plot is the performance of method's best-of-generation players averaged over 30 runs. Additionally, Table 1 compares the methods in terms of average performance of the best-of-run individuals accompanied by 95% confidence intervals.

The results demonstrate that CEL and RSEL obtain lower performance than the other methods. In particular, CEL is definitely the worst algorithm on this performance measure and barely reaches the level of 80%, roughly 5% less than any other method. Moreover, its best-of-run players' performance varies the most, which we attribute to its adaptive fitness function. As anticipated from the previous research [5], RSEL performs significantly better than CEL on the expected utility criterion because it employs an unbiased estimator of this measure as a fitness function. Intuitively, the hybrid of CEL and RSEL constructed by averaging their fitness functions (cf. Fig. 5) could be expected to result in averaging their performances as well. However, it turns out that combining these methods leads to synergy — CEL-RS learns faster and leads to a higher level of play than its constituents employed separately.

Both shaping-based approaches (SSEL and CEL-SS) stay very close to each other throughout the entire evolution, and are the best methods in the comparison.

Although the shaped fitness function employed by SSEL is designed to encourage winning against stronger opponents, it appears that such redefinition allowed evolution to find the strategies that are generally better with respect to the expected utility measure. This is somewhat surprising because, technically, it is RSEL that is tailored to maximize this measure by utilizing the precise estimate of expected

Table 1: Performance comparison of best-of-run players

Method	Performance [%]	t-value	p-value
CEL-SS	87.13 ± 0.46	-6.26	6.7×10^{-7}
SSEL	87.10 ± 0.61	-5.43	1.261×10^{-7}
CEL-RS	86.58 ± 0.44	-4.82	10.4×10^{-7}
RSEL	85.12 ± 0.45	-	-
CEL	81.11 ± 0.99	6.64	0.36×10^{-7}

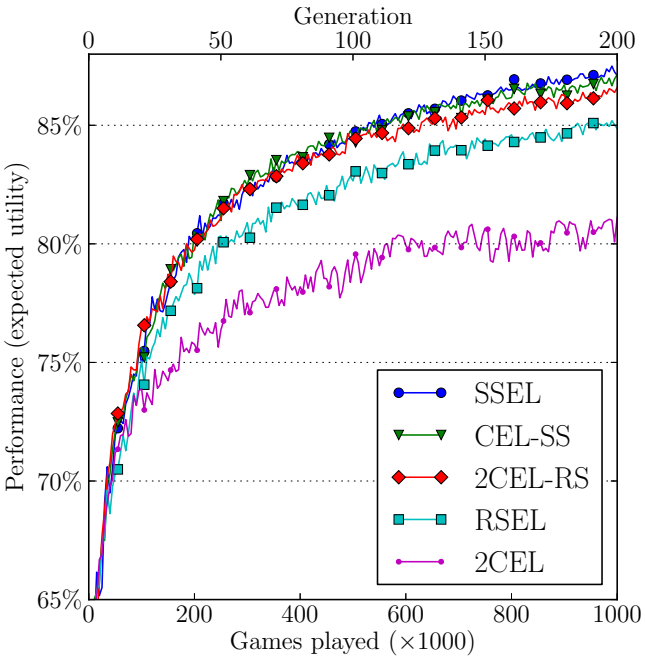

Figure 6: Performance of players over time.

utility to steer evolution — it learns from the same distribution of opponents on which it is later assessed. Yet it fares significantly worse than SSEL which employs a handcrafted, biased probability distribution of opponents during learning. Thereby, it may be beneficial to distort the fitness function of the learning problem, even if the objective function is known and can be precisely approximated. This observation forms major rationale for shaping.

4.2 Performance Profiles

To understand the characteristics of particular methods we use *performance profiles* [13] that break down the expected utility measure into more detailed information on how a strategy copes with the opponents of different strength. To prepare a performance profile, we randomly generate 500,000 players (opponents) using the same method as for generalization performance, i.e., 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 (generated on the fly). 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 according to its performance.

The ensemble of opponents partitioned into bins forms the basis for building the profile. The assessed strategy plays

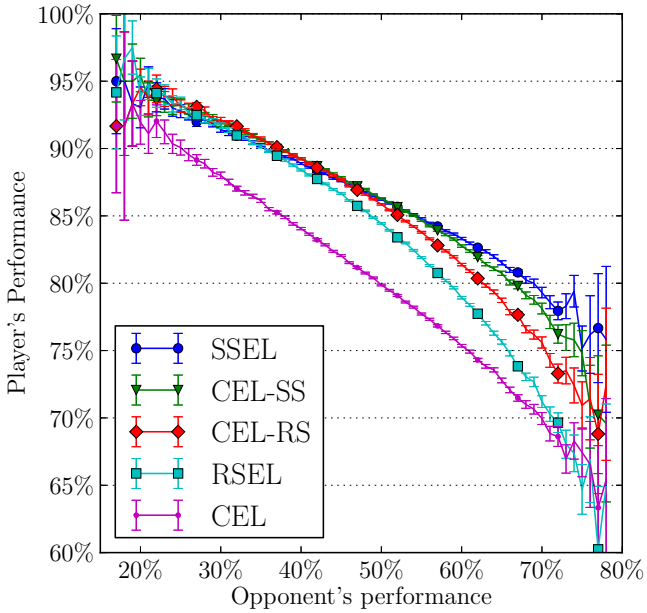

Figure 7: Performance profile plot illustrating the average percentage score against opponents of different performance.

double games against all opponents from each bin, and the average game outcome is plotted against the bins.

We apply the performance profiling to inspect the best-of-run individuals of all algorithms considered in this paper and present them in Fig. 7. Since we have 30 runs per method, we average the profiles over 30 best-of-run players. A point of coordinates (x, y) in a plot means that the best-of-run individuals have on average performance y when playing against the opponents of performance x. For example, the performance of most methods is about 90% for the opponents with performance of 35%.

In Fig. 7 the whiskers mark 95% confidence interval. Noticeably, they tend to widen towards the ends of plots. This is because it is hard to randomly generate the 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 the confidence intervals that were larger than 20% or were computed on a basis of less than 5 double games played. The decreasing trend in each data series demonstrates that it is generally harder to win with the stronger opponents than with the weaker ones.

The performance profile plots shed a new light on comparison between RSEL and CEL conducted by Chong et al. [5] and reproduced in Section 4.1. The plot explains the reason of the observed significant difference between these methods with respect to the expected utility measure. RSEL plays much better with the weak and mediocre opponents that occur frequently among the random WPC players. CEL, in contrast, learns to play against the strong opponents but at the same time underperforms against the weaker ones. The performance profile of CEL shows that it is the worst method in almost entire space of considered opponents. Only for the very strong ones (with performance of 70% or more) it is on a par with RSEL. We suppose that the performance distribution in a coevolving population of tests is biased even more than in SSEL — as solutions get better, their opponents (tests) must acquire more competence

to distinguish between them. This challenges the solutions to perform even better, and so the feedback loop closes. Investigating how the performance distribution of opponents, and consequently the fitness function, is adaptively shaped by coevolution is an interesting direction for the future work.

Apart from CEL, all other methods perform similarly when evaluated against the players of performance lower than 40% (Fig. 7). Above this value, the profiles start to diverge and RSEL is the method that loses the most when the opponents get more challenging. To explain this fact, let us recall that the performance distribution of random WPC players (see Fig. 3) contains mainly the mediocre players. For this reason, RSEL can be seen as specialized to play with the opponents of average skills. Given the shaped performance distribution of opponents used for fitness evaluation (see Fig. 4), SSEL is analogously focused at defeating the highly skilled players. However, this method manages to maintain also the capability to play with the weaker players. This is particularly interesting, knowing that SSEL's learners never meet players weaker than 60%. This may suggest that, at least for the game of Othello, it is enough to learn from the master opponents to obtain a well-playing strategy.

4.3 Relative Performance

In our final experiment we perform a round-robin tournament among all methods. This assessment determines a relative ranking of methods [12] by playing matches between the teams of players. Here, each team consists of 30 best-of-run players produced by an algorithm. Thus, a single match in the tournament involves $30 \times 30 = 900$ double games. Winning games against all opponent teams gives a method the round-robin performance of 100%.

In order to gain better insight into the results, let us first explain the nature of the round-robin tournament. In contrast to the absolute performance measure (Section 4.1) which evaluates an individual on a set of random opponents, in a round-robin tournament each individual faces only the individuals from the other teams. And, judging from the expected utility attained by particular methods (see Table 1), these players tend to be strong.

Table 2 presents the results of the tournament. SSEL is clearly the best algorithm in the relative comparison. Thanks to employing a handcrafted pool of evaluating opponents with shaped performance distribution, it is substantially biased towards competing with the highly skilled opponents. Considering the fact that the tournament is held between the relatively strong players, such a good result of SSEL could be anticipated. Furthermore, RSEL, which is specialized at beating mediocre opponents, inevitably loses to all other methods in head-to-head matches, and its aggregated round-robin performance is significantly lower than the performance of other methods.

The benefits from using a shaped performance distribution of opponents can be also observed when comparing CEL-RS with CEL-SS. According to Table 1, the difference in absolute performance between these methods is minor. However, in the round-robin tournament there is a 5% performance gap in favor of CEL-SS. This fact can be explained by analyzing Fig. 7 which demonstrates that the performance profile of CEL-SS is considerably more leveled than that of CEL-RS. Certainly, this results in better play against the strong tournament opponents.

Table 2: Round robin tournament

Method	SSEL	CEL-SS	CEL-RS	CEL	RSEL	**Final result**
SSEL	-	55.4%	58.5%	58.0%	64.8%	**59.2%**
CEL-SS	44.6%	-	55.8%	54.8%	59.6%	**53.7%**
CEL-RS	41.5%	44.3%	-	49.9%	59.4%	**48.8%**
CEL	42.0%	45.3%	50.1%	-	55.5%	**48.2%**
RSEL	35.2%	40.4%	40.6%	44.5%	-	**40.2%**

Another interesting observation is that CEL and CEL-RS play on a par (and much better than RSEL). This raises the question about the ratio between the random and population opponents that are sampled in the CEL-RS method. The presented results may suggest that sampling half of the evaluating opponents from the coevolving population is enough to learn to defeat strong players (RS-based opponents are on average much weaker than the population ones). However, whether a lower fraction would suffice as well, and what would happen if that fraction would be greater than 0.5, remains to be verified.

5. CONCLUSIONS

Let us rephrase here the original research question with which we began this paper: how can we improve the results of evolutionary algorithms on a problem of learning Othello-playing strategies with respect to the performance measure of expected utility? In this study we have attempted to answer this question by referring to the concept of shaping, borrowed from behavioral psychology. Specifically, we have modified the fitness function in the learning problem by changing the selection of the opponents used for strategy evaluation. By doing so we expected to shape the function and make it easier for an algorithm to explore the particularly desirable areas in the solution space.

The goal of improving the final learning results was attained. Both investigated methods of fitness function shaping — modifying the distribution of opponent performance a priori (SSEL) and autonomously by coevolution (CEL) — proved to influence the learning process and lead to significantly different results than the reference RSEL method.

An interesting direction for the future work is to investigate the performance distribution of opponents bred in coevolutionary learning. Knowing how this distribution evolves across generations could better explain the reported results and help to design a more effective method of adaptive fitness function shaping. Moreover, the analysis of opponent performance distributions provided by coevolution could help with understanding the coevolutionary pathologies [25]. Currently, our focus is on extending the algorithms considered in this paper with archives of historical opponents [22]. We anticipate that such long-time memory will sustain the learning progress and prevent forgetting the game-playing skills that allow defeating the weaker opponents.

6. ACKNOWLEDGMENT

This work has been supported by grant no. N N519 441939. M. Szubert has been supported by Polish Ministry of Science and Education, grant no. 2012/05/N/ST6/03152. W. Jaśkowski has been supported by Polish Ministry of Science and Education, grant no. 91–531/DS.

7. REFERENCES

[1] 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.

[2] A. Bucci, J. B. Pollack, and E. de Jong. Automated extraction of problem structure. In K. D. et al., editor, *Genetic and Evolutionary Computation – GECCO-2004, Part I*, volume 3102 of *Lecture Notes in Computer Science*, pages 501–512, Seattle, WA, USA, 26-30 June 2004. Springer-Verlag.

[3] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward. A classification of hyper-heuristics approaches. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, chapter 15, pages 449–468. Springer, 2nd edition, 2010.

[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 and J. B. Pollack. Ideal Evaluation from Coevolution. *Evolutionary Computation*, 12(2):159–192, Summer 2004.

[8] A. Eiben and S. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

[9] T. Erez and W. Smart. What does shaping mean for computational reinforcement learning? In *Development and Learning, 2008. ICDL 2008. 7th IEEE International Conference on*, pages 215 –219, aug. 2008.

[10] 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.

[11] W. Jaśkowski and K. Krawiec. Formal analysis, hardness and algorithms for extracting internal structure of test-based problems. *Evolutionary Computation*, 19(4):639–671, 2011.

[12] 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.

[13] W. Jaśkowski, P. Liskowski, M. Szubert, and K. Krawiec. Improving coevolution by random sampling. In *GECCO'13: Proceedings of the 15th annual conference on Genetic and Evolutionary Computation*, Amsterdam, The Netherlands, July 2013. ACM.

[14] 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.

[15] 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.

[16] K. Krawiec and M. Szubert. Learning n-tuple networks for othello by coevolutionary gradient search. In N. K. et al, editor, *GECCO 2011 Proceedings*, pages 355–362. ACM, ACM, 2011.

[17] 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.

[18] S. M. Lucas and T. P. Runarsson. Temporal difference learning versus co-evolution for acquiring othello position evaluation. In *CIG*, pages 52–59, 2006.

[19] 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.

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

[21] J. Randløv and P. Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 463–471. Morgan Kaufmann, San Francisco, CA, 1998.

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

[23] S. Samothrakis, S. Lucas, T. Runarsson, and D. Robles. Coevolving Game-Playing Agents: Measuring Performance and Intransitivities. *IEEE Transactions on Evolutionary Computation*, (99):1–15, 2012.

[24] B. Skinner. *The behavior of organisms: An experimental analysis.* Appleton-Century, 1938.

[25] R. A. Watson and J. B. Pollack. Coevolutionary dynamics in a minimal substrate. In L. S. et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 702–709, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.