Control and Cybernetics

vol. **40** (2011) No. 3

Learning board evaluation function for Othello by hybridizing coevolution with temporal difference learning*†

by

Marcin Szubert, Wojciech Jaśkowski and Krzysztof Krawiec

Institute of Computing Science, Poznań University of Technology
Piotrowo 2, 60965 Poznań, Poland
email: {mszubert,wjaskowski,kkrawiec}@cs.put.poznan.pl

Abstract: Hybridization of global and local search techniques has already produced promising results in the fields of optimization and machine learning. It is commonly presumed that approaches employing this idea, like memetic algorithms combining evolutionary algorithms and local search, benefit from complementarity of constituent methods and maintain the right balance between exploration and exploitation of the search space. While such extensions of evolutionary algorithms have been intensively studied, hybrids of local search with coevolutionary algorithms have not received much attention. In this paper we attempt to fill this gap by presenting Coevolutionary Temporal Difference Learning (CTDL) that works by interlacing global search provided by competitive coevolution and local search by means of temporal difference learning. We verify CTDL by applying it to the board game of Othello, where it learns board evaluation functions represented by a linear architecture of weighted piece counter. The results of a computational experiment show CTDL superiority compared to coevolutionary algorithm and temporal difference learning alone, both in terms of performance of elaborated strategies and computational cost. To further exploit CTDL potential, we extend it by an archive that keeps track of selected well-performing solutions found so far and uses them to improve search convergence. The overall conclusion is that the fusion of various forms of coevolution with a gradient-based local search can be highly beneficial and deserves further study.

Keywords: evolutionary computation, coevolutionary algorithms, reinforcement learning, memetic computing, game strategy learning.

*Submitted: September 2010; Accepted: August 2011.

†This is an extended and amended version of the paper, presented at the 5th Congress of Young IT Scientists (Międzyzdroje, 23-25.IX.2010).

1. Introduction

The past half century of AI research on games demonstrated that handcrafting well-performing strategies, though feasible, is challenging and expensive in terms of human and computer effort. There is growing hope for a change due to methods that *learn* the strategies automatically with little *a priori* domain knowledge. Two intensely studied examples are Temporal Difference Learning (TDL) and Coevolutionary Learning (CEL).

TDL is a canonical variant of reinforcement learning, where the playing agent aims at maximizing a delayed reward, and is typically trained by some form of gradient descent. CEL breeds a population of strategies that compete with each other and propagate their features using the principles of simulated evolution. The essential difference between TDL and CEL is that TDL guides the learning using the whole course of the game while CEL uses only the final game outcome. As a result, TDL in general learns faster than CEL. However, for some domains a properly tuned CEL can eventually find strategies that outperform those generated by TDL (Runarsson and Lucas, 2005; Lucas and Runarsson, 2006).

From another perspective, evolutionary algorithms (and CEL in particular) have widely recognized explorative ability, while TDL is a local-search technique that greedily and quickly makes its way towards a nearby local optimum, thus having an exploitative potential. Hybridization of explorative and exploitative techniques has already produced promising results in the fields of optimization and machine learning. It is commonly presumed that approaches employing this idea, like memetic algorithms that combine evolutionary algorithms with local search, benefit from complementary characteristics of constituent methods and maintain the right balance between exploration and exploitation of the search space. While such extensions of evolutionary algorithms have been intensively studied, hybrids of local search with coevolution have not received much attention yet.

In this paper, we ask whether it is possible to combine the advantages of TDL and CEL in a single algorithm that would develop better solutions than any of these methods on its own. To this aim, we propose a hybrid method referred to as Coevolutionary Temporal Difference Learning (CTDL) that works by interlacing global search provided by competitive coevolution and local search by means of temporal difference learning. In our previous research we have already evaluated this method on the games of Othello (Szubert et al., 2009) and small-board Go (Krawiec et al., 2011). Here, we perform further investigation on CTDL in the context of Othello, for which both CEL and TDL were independently tested and compared (Lacas and Runarsson, 2006).

This paper is organized as follows. In Section 2 we describe the rules, strategy representation, and previous research on learning Othello strategies. Section 3 reviews TDL and CEL and introduces CTDL. After describing the experimental setup in Section 4, we discuss the results in Sections 5 and 7, to conclude in Section 8.

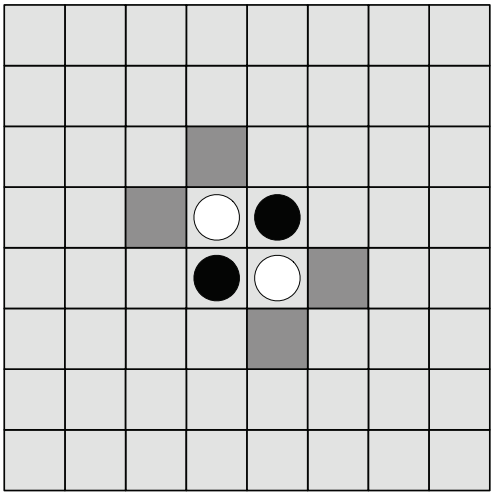


Figure 1. The initial board configuration in Othello

2. The game of Othello

A minute to learn... a lifetime to master is the motto of the game of Othello. Indeed, despite its apparent simplicity, Othello is one of the most challenging board games with numerous tournaments and regular world championship matches. The name of the game stems from Shakespeare's drama *"Othello, the Moor of Venice"*, and is meant to illustrate that the game is full of dramatic reversals caused by rapid changes in dominance on the board.

2.1. Game rules

Othello is played by two players on an 8×8 board. Typically, pieces are disks with a white and black face, each face representing one player. Fig. 1 shows the initial state of the board; each player starts with two stones in the middle of the grid. The black player moves first, placing a piece, black face up, on one of four shaded locations. Players make moves alternately until no legal moves are possible.

A legal move consists of placing a piece on an empty square and flipping appropriate pieces. To place a new piece, two conditions must be fulfilled. Firstly, the position of the piece 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 contiguous sequence of opponent's pieces in between. After placing the piece, all such opponent's pieces are flipped; if multiple lines exist, flipping affects all of them. This feature makes the game particularly dramatic: a single move may gain the player a large number of pieces and swap players' chances of winning.

A legal move requires flipping at least one of the opponent's pieces. Making a move in each turn is obligatory, unless there are no legal moves. The game ends when both players have no legal moves. The winner is the player who at the end has more disks; the game can also end with a draw.

2.2. Strategy representation

One of the main issues to consider when learning a game strategy is the architecture of the learner, which is mainly determined by the representation adopted to store its strategy. There are two common ways in which strategies can be represented, namely, as *a move selector* or as a *state evaluator* (also known as *position evaluator*). A move selector takes the current state of the game as an input and returns a move to be made. A state evaluator, on the other hand, is used to estimate how beneficial a given state is for the player. With a help of a game tree search algorithm, this allows for selecting the move that will lead to the most favorable future state.

Most recent works on learning Othello strategies have focused on creating board evaluation functions and we decided to follow that trend in this study. During the game, in order to select a move, we evaluate all states at 1-ply. Regarding the choice of the architecture of the state evaluator, we rely on a heuristic assumption that to judge the utility of a particular board state it is enough to *independently* consider the occupancy of all board locations. This principle is implemented by the *weighted piece counter* (WPC), which assigns a weight w_i to each board location i and uses scalar product to calculate the utility f of a board state $\mathbf{b}$:

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

where b_i is $+1$, -1, or 0 if, respectively, location i is occupied by a black piece, white piece, or remains empty. The players interpret the values of f in a complementary manner: the black player prefers moves leading to states with larger values while smaller values are favored by the white player. WPC may be viewed as an artificial neural network comprising a single linear neuron with inputs connected to board locations.

The main advantage of WPC is its simplicity leading to very fast board evaluation. Moreover, a strategy represented by a WPC can be easily interpreted just by inspecting the weight values. Table 1 presents the weight matrix of an exemplary player that clearly focuses at the corners because they are given the highest values.

2.3. Previous research

The game of Othello has been a subject of artificial intelligence research for more than 20 years. The significant interest in this game may be explained by its large

Table 1. The heuristic player's strategy represented by WPC

1.00	-0.25	0.10	0.05	0.05	0.10	-0.25	1.00
-0.25	-0.25	0.01	0.01	0.01	0.01	-0.25	-0.25
0.10	0.01	0.05	0.02	0.02	0.05	0.01	0.10
0.05	0.01	0.02	0.01	0.01	0.02	0.01	0.05
0.05	0.01	0.02	0.01	0.01	0.02	0.01	0.05
0.10	0.01	0.05	0.02	0.02	0.05	0.01	0.10
-0.25	-0.25	0.01	0.01	0.01	0.01	-0.25	-0.25
1.00	-0.25	0.10	0.05	0.05	0.10	-0.25	1.00

state space cardinality (around 10^{28}) and high divergence rate, causing that it remains unsolved, that is — a perfect Othello player has not been developed yet.

Conventional programs playing Othello are based on a thorough human analysis of the game leading to sophisticated handcrafted evaluation functions. They often incorporate supervised learning techniques that use large expert-labeled game databases and efficient look-ahead game tree search. One of the first examples representing such approach was BILL (Lee and Mahajan, 1990). Besides using pre-computed tables of board patterns, it employed Bayesian learning to build in so-called *features* into evaluation function. Today, one of the strongest Othello programs is Logistello (Buro, 2002), which also makes use of advanced search techniques and applies several methods to construct evaluation features and learn from previous games. Nevertheless, it still relies on powerful hardware, which is one of the main factors that allowed Logistello to beat the world champion Takeshi Murakami in 1997.

Recently, the mainstream research on Othello has moved towards better understanding of which learning algorithms and player architectures work the best. The CEC Othello Competitions[1] pursued this direction by limiting the ply depth to one, effectively disqualifying the algorithms employing a brute-force game tree search. Although WPC is among strategy representations accepted by the competition rules, all the best players submitted so far to the competition were based on more complex architectures involving numerous parameters. Examples of such architectures are: a symmetric n-tuple network, a multi-layer perceptron (MLP), and a spatial MLP.

The most challenging scenario of elaborating a game strategy is learning without any support of human knowledge and opponent strategies given a priori. This formulation is addressed by, among others, Temporal Difference Learning (TDL) and Coevolutionary Learning (CEL), which were applied to Othello by Lucas and Runarsson (2006). That study inspired our research and will be also referred to in the following section.

[1] http://algoval.essex.ac.uk:8080/othello/html/Othello.html

3. Methods

3.1. Coevolutionary learning

Coevolutionary algorithms are variants of evolutionary computation where individual's fitness depends on other individuals. Evaluation of an individual takes place in the context of at least one other individual, and may be of cooperative or competitive nature. In the former case, individuals share the fitness they have jointly elaborated, whereas in the latter one, a gain for one individual means a loss for the other. Past research has shown that this scheme may be beneficial for some types of tasks, allowing task decomposition (in the cooperative variant) or solving tasks for which the objective fitness function is not known *a priori* or is hard to compute (the best example here are games, Angeline and Pollack, 1993; Azaria and Sipper, 2005).

Coevolutionary Learning (CEL) follows the competitive scheme and typically starts with generating a random initial population of player individuals. Individuals play games with each other, and the outcomes of these confrontations determine their fitness values. The best performing strategies are selected, undergo genetic modifications such as mutation or crossover, and their offspring replace some of (or all) former individuals. This scheme, of course, misses many details that need to be filled in, some of which relate to evolutionary computation (population size, variation operators, selection scheme, etc.), others pertaining specifically to coevolution (the way the players are confronted, the method of fitness estimation, etc.). No wonder CEL embraces a broad class of algorithms, some of which we shortly review in the following.

In their influential study, Pollack and Blair (1998) used one of the simplest evolutionary algorithms, a random hill-climber to successfully address the problem of learning backgammon strategy. Lucas and Runarsson (2006) used $(1 + \lambda)$ and $(1, \lambda)$ evolution strategies to learn a strategy for the game of Othello. An important design choice was the geometrical parent-child recombination: instead of replacing the parent by the best of the new offspring, the parent strategy was fused with the child strategy using linear combination. A self-adapting mutation strength was also considered, but eventually used only for evolving small-board Go players (Runarsson and Lucas, 2005).

Various forms of CEL have been successfully applied to many two-person games, including backgammon (Pollack and Blair, 1998), chess (Hauptman and Siper, 2007), checkers (Fogel, 2001), NERO (Stanley et al., 2005), blackjack (Caverlee, 2000), Pong (Monroy et al., 2006), ant wars (Jaśkowski et al., 2008), and a small version of Go (Lubberts and Miikkulainen, 2001).

3.2. Coevolutionary archives

The central characteristic of CEL is that it refrains from the use of *objective fitness* of individuals. This feature makes it appealing for applications where the objective fitness cannot be unarguably defined or is infeasible to compute.

Games, often involving huge numbers of possible strategies, are canonical representatives of such problems. However, inaccessibility of the objective fitness implies a serious impairment: there is no guarantee that an algorithm will progress at all. Lack of progress can occur when, for instance, player's opponents are not challenging enough or much too difficult to beat. These and other undesirable phenomena, jointly termed *coevolutionary pathologies*, have been identified and studied in the past (Ficici, 2004).

In order to deal with coevolutionary pathologies, *coevolutionary archives* were introduced. A typical archive is a (usually limited in size, yet diversified) sample of well-performing strategies found so far. Individuals in a population are forced to play against the archive members, replaced occasionally, typically when they prove inferior to some population members. Of course, an archive does not guarantee that the strategies found by evolution will be the best in the global, objective sense, but this form of long-term search memory enables at least some form of *historical progress* (Miconi, 2009).

In this study we use Hall of Fame (HoF, Rosin and Belew, 1997), one of the simplest archives. HoF stores all the best-of-generation individuals encountered so far. The individuals in population, apart from playing against their peers, are also forced to play against randomly selected players from the archive. In this way, individual's fitness is partially determined by confrontation with past 'champions'.

Most of the work quoted above involves a single homogeneous population of players, a setup called *one-population coevolution* (Luke and Wiegand, 2002) or *competitive fitness environment* (Angeline and Pollack, 1993; Luke, 1998). It is worth to point out that the recent work on coevolution indicates that, even if the game itself is symmetric, it can be useful to maintain in parallel two populations, each consisting of individuals encoding strategies of a particular type: *candidate solutions*, which are expected to improve as evolution proceeds, and *tests*, whose main purpose is to differentiate candidate solutions by defeating some of them. Recent contributions (Ficici and Pollack, 2003; de Jong, 2005, 2007) demonstrate that such design can improve search convergence, give better insight into the structure of the search space, and in some settings even guarantee monotonic progress towards the selected solution concept (Ficici, 2004).

3.3. Temporal difference learning

Temporal Difference (TD), a method proposed by Sutton (1988), has become a popular approach for solving reinforcement learning tasks. Some suggest (Sutton and Barto, 1998) that the famous checkers playing program by Samuel (1959) was in fact taught by a simple version of temporal difference learning (however others, Bucci, 2007, treat it rather as a first example of coevolutionary algorithm). One of the most spectacular successes of temporal difference learning in game playing is undoubtedly Tesauro's TD-Gammon (Tesauro, 1995).

This influential work has triggered off a lot of research in reinforcement learning and TD methods, including their applications to Othello (Manning, 2007).

The $TD(\lambda)$ learning procedures solve prediction learning problems that consist in estimating the future behavior of an incompletely known system using the past experience. TD learning occurs whenever a system state changes over time and is based on the error between the temporally successive predictions. Its goal is to make the preceding prediction to match more closely the current prediction (taking into account distinct system states observed in the corresponding time steps).

Technically, the prediction at a certain time step t can be considered as a function of two arguments: the outcome of system observation P and the vector of modifiable weights $\mathbf{w}$. A TD algorithm is expressed by the following weight update rule:

$$\Delta \mathbf{w}_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k, \tag{2}$$

where P_t is the prediction at time t and the gradient $\nabla_w P_t$ is the vector of partial derivatives of P_t with respect to each weight. The parameter α is the learning rate, while the trace decay $\lambda \in [0, 1]$ determines the rate of 'aging' of past gradients, i.e., the rate at which their impact on current update decays when reaching deeper into history. This general formulation of TD takes into account the entire history of the learning process; in case of $TD(0)$, the weight update is determined only by its effect on the most recent prediction P_t:

$$\Delta \mathbf{w}_t = \alpha(P_{t+1} - P_t) \nabla_w P_t. \tag{3}$$

When applied to the problem of learning Othello strategy represented by a WPC, P_t estimates the chances of winning given the game state $\mathbf{b}_t$ at time t. The WPC function f computes the dot product of the board state vector $\mathbf{b}_t$ and the weight vector $\mathbf{w}$ (see Eq. (1)), and the obtained value is subsequently mapped to a closed interval [-1, 1] using hyperbolic tangent, so that P_t has the form:

$$P_t = tanh(f(\mathbf{b}_t)) = \frac{2}{exp(-2f(\mathbf{b}_t)) + 1} - 1. \tag{4}$$

By applying (4) to the $TD(0)$ update rule (3) and calculating the gradient, we obtain the desired correction of weight w_i at the time step t:

$$\Delta w_{i,t} = \alpha(P_{t+1} - P_t)(1 - P_t^2) b_i. \tag{5}$$

If the state observed at time $t+1$ is terminal, the exact outcome of the game is known and may be used instead of the prediction P_{t+1}. The outcome value is $+1$ if the winner is black, -1 if white, and 0 when the game ends in a draw.

The process of learning consists of applying the above formula to the WPC vector after each move. The training data (i.e., collection of games) according to which the presented algorithm can proceed, may be obtained by self-play. The major advantage of this is that nothing besides the learning system is required. During game play, moves are selected on the basis of the most recent evaluation function.

Othello is a deterministic game, thus the course of the game between a particular pair of deterministic players is always the same. This feature reduces the number of game trees to be explored and makes learning ineffective. To remedy this situation, at each turn, a random move is forced with a certain probability. After such a random move, no weight update occurs.

3.4. Coevolutionary temporal difference learning

The past results of learning WPC strategies for Othello (Lucas and Runarsson, 2006) and small-board Go (Runarsson and Lucas, 2005) demonstrate that TDL and CEL exhibit complementary features. TDL learns much faster and converges within several hundreds of games, but then stucks, and, no matter how many games it plays, eventually fails to produce a well-performing strategy. CEL progresses slower, but, if properly tuned, eventually outperforms TDL. So, it sounds reasonable to combine these approaches into a hybrid algorithm exploiting advantages revealed by each method.

To benefit from the complementary advantages of TDL and CEL we propose a method termed *Coevolutionary Temporal Difference Learning* (CTDL). CTDL maintains a population of players and alternately performs temporal difference learning and coevolutionary learning. In the TDL phase, each player is subject to $TD(0)$ self-play. Then, in the CEL phase, individuals are evaluated on the basis of a round-robin tournament. Finally, a new generation of individuals is obtained using selection and variation operators and the cycle repeats.

Other hybrids of TDL and CEL have been occasionally considered in the past. Kim et al. (2007) trained a population of neural networks with $TD(0)$ and used the resulting strategies as an input for the standard genetic algorithm with mutation as the only variation operator. Singer (2001) has shown that reinforcement learning may be superior to a random mutation as an exploration mechanism. His Othello-playing strategies were 3-layer neural networks trained by interlacing reinforcement learning phases and evolutionary phases. In the reinforcement learning phase, a round robin tournament was played 200 times with network weights modified after every move using backpropagation algorithm. The evolutionary phase consisted of a round-robin tournament that determined each player's fitness, followed by recombining the strategies using feature-level crossover and mutating them slightly. The experiment yielded a strategy that was reported to be competitive with an intermediate-level handcrafted Othello player; however, no comparison with preexisting methods was presented. Also, given the proportions of reinforcement and evolutionary learning, it seems that

Singer's emphasis was mainly on reinforcement learning, whereas in our CTDL it is quite the reverse: reinforcement learning serves as a local improvement operator for evolution.

4. Experiments

We conducted several experiments comparing CTDL, CEL, TDL, and their extensions with the Hall of Fame (HoF) archive (Rosin and Belew, 1997), all implemented using *Evolutionary Computation in Java* (ECJ) library (Luke, 2008). To provide a fair comparison, all runs used the same settings (taken from Lucas and Runarsson, 2006, when possible) and stopped when the number of games played reached 10 million. For statistical significance, each run was repeated 50 times with different random number generator seeds.

4.1. Algorithms and setup

4.1.1. TDL

TDL is an implementation of a gradient-descent temporal difference algorithm $TD(0)$ described in Section 3.3 and parametrized as in Lucas and Runarsson (2006). The weights are initially set to 0 and the learner is trained solely through self-play, with random moves occurring with probability $p = 0.1$. The learning rate $\alpha = 0.01$.

4.1.2. CEL

CEL uses a generational coevolutionary algorithm with population of 50 individuals initialized with all weights set to 0[2]. During mutation, the weights are limited to the range $[-1, 1]$. In the evaluation phase, a round-robin tournament is played between all individuals, with wins, draws, and losses rewarded by 3, 1, and 0 points, respectively. The evaluated individuals are selected using standard tournament selection with tournament size 5, and then, with probability 0.03, their weights undergo a Gaussian mutation ($\sigma = 0.25$). Next, they mate using one-point crossover, and the resulting offspring is the only source of genetic material for the subsequent generation (there is no elitism). As each generation requires 50×50 games, each run lasts for 4000 generations to get the total of 10,000,000 games.

4.1.3. CEL + HoF

This setup extends the previous one with the HoF archive. Each individual plays games with all 50 individuals from the population (including itself) and with 50

[2]Unintuitively, we have found for Othello that initializing the weights with 0 for all individuals leads to better solutions than when weights were drawn at random from $[-1, 1]$ — see also Section 6.3.

randomly selected individuals from the archive, so that its fitness is determined by the outcomes of 100 games scored as in CEL. In each generation, the best performing individual is copied into the archive. The archive serves also as a source of genetic material, as the first parent for crossover is randomly drawn from it with probability 0.2. In order to attain 10,000,000 training games, the number of generations was set to 2000.

4.1.4. CTDL = TDL + CEL

CTDL combines TDL and CEL as described in Section 3.4, with the TDL phase parameters described in 4.1.1 and CEL phase parameters described in 4.1.2. It starts with players' weights initialized to 0 and alternately repeats the TDL phase and the CEL phase until the total number of games attains 10,000,000. The exact number of generations depends on the TDL-CEL ratio, which we define as the number of self-played TDL games per one generation of CEL. For example, if the TDL-CEL ratio is 1 (default), there are 2,550 games per generation (including the round-robin tournament of CEL) and the run lasts for 3,925 generations.

4.1.5. CTDL+HoF = TDL + CEL + HoF

This setup combines 4.1.3 and 4.1.4 and does not involve any extra parameters.

4.2. Measuring strategy quality

In order to monitor progress in an objective way, 100 times per run (approximately every 100,000 of games) we assess the quality of the best-of-generation individual. As learning a game strategy is an example of a *test-based problem*, an objective evaluation should take into account games with all possible players and be based on a particular solution concept (Ficici, 2004). This approach cannot be implemented in practice due to the number of possible strategies for Othello. Thus, following Lucas and Runarsson (2006), we rely on two approximate yet computationally feasible quality measures described below. Both of them estimate individual's quality by playing 1,000 games (500 as black and 500 as white) against certain opponent(s) and calculating the probability of winning.

4.2.1. Playing against the random player

This method tests how well the player fares against a wide variety of opponents. The opponents always choose moves at random, no matter what the board state is. Notice that this quality measure estimates the objective quality of an individual according to the the solution concept of *Maximization of Expected Utility* (Ficici, 2004).

4.2.2. Playing against the standard heuristic player

This method tests how well the player copes with a moderately strong opponent using WPC shown in Table 1. Since the game of Othello is deterministic, we force both players to make random moves with probability $\epsilon = 0.1$ to diversify their behaviors and make the estimated values more continuous. Though this essentially leads to a different game definition, following Lucas and Runarsson (2006), we assume that the ability of playing such a randomized game is highly correlated with the ability of playing the original Othello.

5. Results

5.1. Comparison of algorithms with an external opponent

In the first experiment, we compared five methods described in the previous section for performance against the random player and the standard heuristic player. Fig. 2 illustrates how the strategies produced by the algorithms perform on average against the random player. Each graph point represents the probability of winning of a best-of-generation individual, averaged over 50 runs. For the population-based methods (i.e., all except the pure TDL), the best-of-generation individual is the one with maximal fitness, while for TDL it is simply the only strategy maintained by the method. It is interesting to observe that the algorithms cluster into two groups with respect to the performance they eventually achieve. The pure methods (CEL and TDL) are significantly worse in the long run than the others. As expected, in the very beginning, the quality of individuals produced by the TDL-based algorithms is higher than of those produced by methods that do not involve TDL. Adding HoF archive improves both CEL and CTDL. Interestingly, CEL+HoF achieves eventually a higher level of play than CTDL(+HoF), but learns slightly slower.

Results of the second experiment are illustrated in Fig. 3, which shows the progress measured as the average performance of best-of-generation individuals when playing with the standard heuristic player. First, it can be observed that CTDL is clearly better than TDL, which initially learns rapidly, but stagnates after several thousands of games. Nevertheless, TDL beats CEL noticeably. Once again, the HoF archive helps both CEL and CTDL to achieve a higher level of play.

Second, while CTDL+HoF and CEL+HoF eventually produce players exhibiting a very similar level, CTDL+HoF does so after approximately 2,500,000 games, while CEL+HoF requires more than 7,500,000 games for this. In terms of the performance measure based on the standard heuristic player, CTDL+HoF achieves its peak performance three times faster than CEL+HoF.

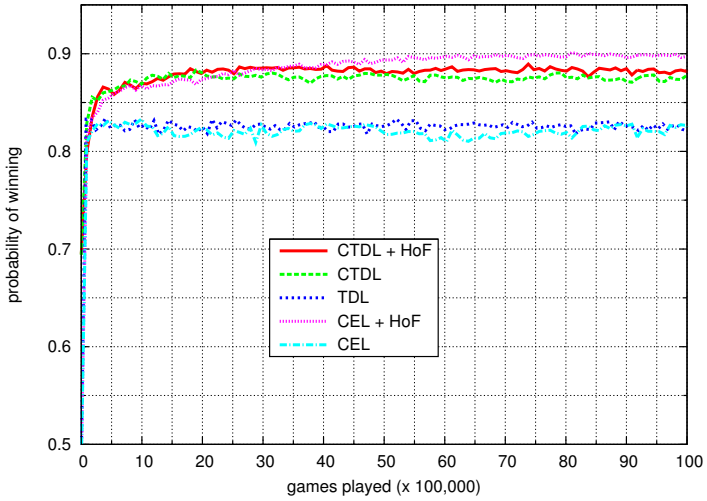


Figure 2. Average performance of the best-of-generation individuals measured as the probability of winning against the **random** player, plotted against the number of training games played

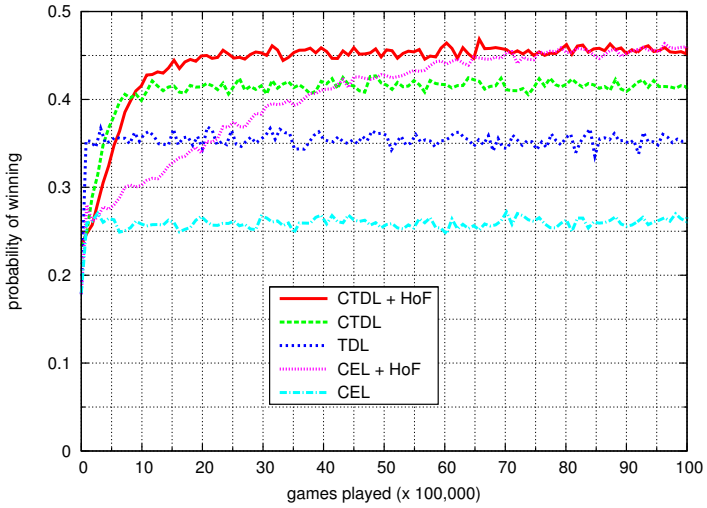


Figure 3. Average performance of the best-of-generation individuals measured as the probability of winning against the standard **heuristic** player (both players randomized with probability 0.1), plotted against the number of training games played

5.2. Comparison of algorithms in a round-robin tournament

The standard heuristic player used in the previous experiments, even if randomized, cannot be expected to represent fully the richness of strong Othello players. That is why we recruit a more diverse set of opponents through round-robin tournaments involving the teams of best-of-generation individuals representing particular methods. The best-of-generation strategies that were subject to individual performance assessment in the previous section are now combined into five teams, each representing one method. Since there are 50 runs of each method, each of teams consists of 50 members. Next, we play a round-robin tournament between the teams, each strategy playing against $4 \times 50 = 200$ strategies from all the opponent teams for a total of 400 games (200 as white and 200 as black). The final score of a team is the total score of its players in the total of 20,000 games, using the standard Othello scoring scheme.

This type of performance assessment allows us to track the relative progress of strategies produced by different algorithms. The results presented in Fig. 4 indicate that the methods combining CEL with TDL take the lead relatively quickly, and later on yield only a little to CEL+HoF, which is the only method exhibiting significant progress after 2,000,000 games. In particular, CTDL+HoF is able to score more than 40,000 out of possible 60,000 points, i.e. more than twice as much as CEL, which is clearly the worst in the field. Also, the HoF-based approaches learn slower because of an additional computational effort caused by the requirement to play games with archival opponents in each generation.

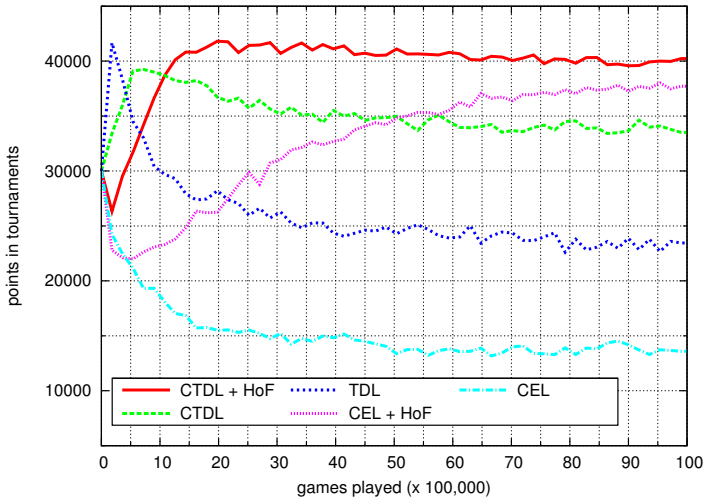


Figure 4. Scores obtained by teams of best-of-generation individuals in the round-robin tournament, plotted against the number of training games played

Table 2. The results of the round-robin tournament involving best-of-last-generation individuals

Team	Games	Wins	Draws	Defeats	Points
CTDL+HoF	20000	13232	538	6230	40234
CEL+HoF	20000	12361	633	7006	37716
CTDL	20000	10958	641	8401	33515
TDL	20000	7627	575	11798	23456
CEL	20000	4331	595	15074	13588

Table 3. Weighted Piece Counter vector of the best evolved player

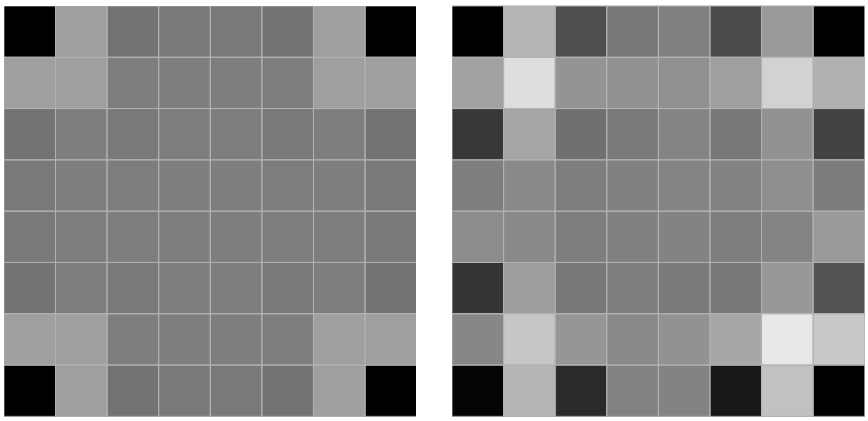
1.01	-0.43	0.38	0.07	0.00	0.42	-0.20	1.02
-0.27	-0.74	-0.16	-0.14	-0.13	-0.25	-0.65	-0.39
0.56	-0.30	0.12	0.05	-0.04	0.07	-0.15	0.48
0.01	-0.08	0.01	-0.01	-0.04	-0.02	-0.12	0.03
-0.10	-0.08	0.01	-0.01	-0.03	0.02	-0.04	-0.20
0.59	-0.23	0.06	0.01	0.04	0.06	-0.19	0.35
-0.06	-0.55	-0.18	-0.08	-0.15	-0.31	-0.82	-0.58
0.96	-0.42	0.67	-0.02	-0.03	0.81	-0.51	1.01

Although CEL+HoF approaches the performance of CTDL+HoF, even after 10,000,000 games it clearly falls behind CTDL+HoF. Notice that the relative performance curve for CEL+HoF seems to converge at the point of 9,000,000 games and there are no grounds to claim that it could actually catch up with CTDL+HoF. This experiment also confirms that CTDL+HoF is much quicker than CEL+HoF in terms of time to converge. Note, as well that the CTDL+HoF deterioration observed in the range of about 2,000,000 to 8,000,000 training games should be attributed not to loss of absolute performance, but to the fact that the other teams get better.

5.3. Best-of-last-generation individuals in round-robin tournament

For a better insight into the relative performance of individuals produced by different methods, in Table 2 we present the results of the round-robin tournament involving teams of *best-of-last-generation* individuals returned by the algorithms. The results confirm the former observations: the best method in direct comparison is CTDL+HoF.

The WPC of the globally best scoring player (i.e., over all algorithms and runs), produced by the winning CTDL+HoF team, is shown in Table 3 and presented graphically in weight-proportional grayscale in Fig. 5b (darker squares denote larger weights, i.e., more desirable locations on the board). An important



(a) The standard heuristic player (b) The best evolved player

Figure 5. Weighted Piece Counter vectors illustrated as Othello boards with locations shaded according to corresponding weights

observation is that the WPC matrix exhibits many symmetries. Similarly to the standard heuristic player's strategy, which is shown graphically in Fig. 5a, the corners are most desirable, while their immediate neighbors have very low weights. However, in contrast to the standard heuristic player, the edge locations at distance 2 from the corners get very high weights, which is consistent with typical preferences of human players.

5.4. Discussion

The three presented experiments, with the random player, the standard heuristic player, and the round-robin tournament lead to consistent results, which allow us to formulate two propositions:

1. CTDL, which consists of interlaced CEL and TDL phases, is significantly better than both its constituent methods alone.
2. The HoF archive improves the performance of both CEL and CTDL, allowing them to produce better individuals in the long run.

However, despite the fact that CTDL performs better than CEL, the direct comparison of CTDL+HoF and CEL+HoF requires a discussion. We can compare CTDL+HoF and CEL+HoF in terms of two metrics: i) the performance of players produced by a method in the long run, and ii) the speed of the method measured as the convergence time. The results of the three experiments show that CTDL+HoF is better than CEL+HoF with respect to the convergence time, because of the TDL phase, which quickly improves individuals maintained by the method to (at least) a moderate level of play.

Concerning the long-run performance metric, CTDL+HoF and CEL+HoF obtain a similar level of play against the standard heuristic player, and CTDL+HoF produces better players than CEL+HoF when compared in a round-robin tournament. However, CTDL+HoF is beaten by CEL+HoF when the random player is used as a benchmark. This somewhat surprising result can be explained by noticing that CEL(+HoF) and CTDL(+HoF) drive the learning process towards different goals[3], and the performance measure based on game results with the random opponent correlates better with the CEL+HoF goal. It seems that due to strong influence of self-play learning, CTDL(+HoF) produces players biased towards playing well against *strong* opponents, while players produced by CEL+HoF are geared for a wide range of opponents, including the *weak* ones. Thus, CTLD(+HoF) trades off its ability to play with weak players, which have their representatives in the wide spectrum of possible behaviour of the random player, for ability to play with strong opponents.

This explanation is supported by three facts. First, CTDL+HoF is equally good in the long run as CEL+HoF when faced with the standard heuristic player, which, on average, is stronger than the random one. Second, CTDL+HoF is better than CEL+HoF in the round-robin tournament, which involves only best-playing individuals at a certain stage of learning. Thus, clearly, the quality of opponents used to measure the performance of methods correlates with the relative difference between CTDL+HoF and CEL+HoF. Finally, notice that a similar correspondence emerges when we compare CEL with TDL on different performance measures. TDL produces players with the quality similar to that of CEL when compared with the random opponent. However, TDL is clearly a better choice, when quality is compared on the basis of performance with the standard heuristic player or in the round-robin tournament. TDL alone is, thus, also more biased towards playing with strong players than pure CEL.

6. Impact of parameter settings

6.1. TDL-CEL ratio

Preliminary experiments have shown that the TDL-CEL ratio is an important parameter of CTDL. We have investigated this issue by running CTDL+HoF for different TDL-CEL ratios. The probability of the best-of-generation individual winning against the random player for different TDL-CEL ratios, presented in Fig. 6, proves that CTDL performance depends on the TDL-CEL ratio and that the ratio above 1 is detrimental. Fig. 6 demonstrates also the tradeoff between the learning speed and the ultimate player quality.

In Fig. 7, which illustrates the performance against the standard heuristic player, different TDL-CEL ratios cause only slight differences in the long run. This was confirmed by playing another round-robin tournament involving teams

[3]The notion of such 'goals' has been formalized as *solution concepts* by Ficici (Ficici, 2004).

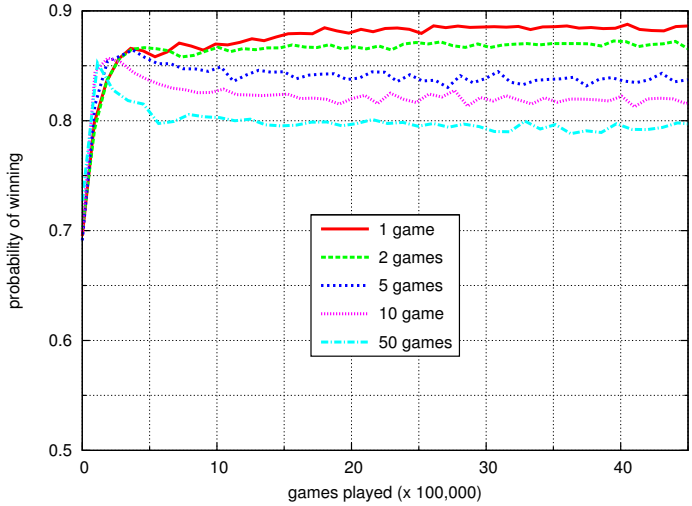


Figure 6. Average probability of winning of the best-of-generation individuals found by CTDL+HoF against the **random** player for different TDL-CEL ratios, plotted against the number of training games played

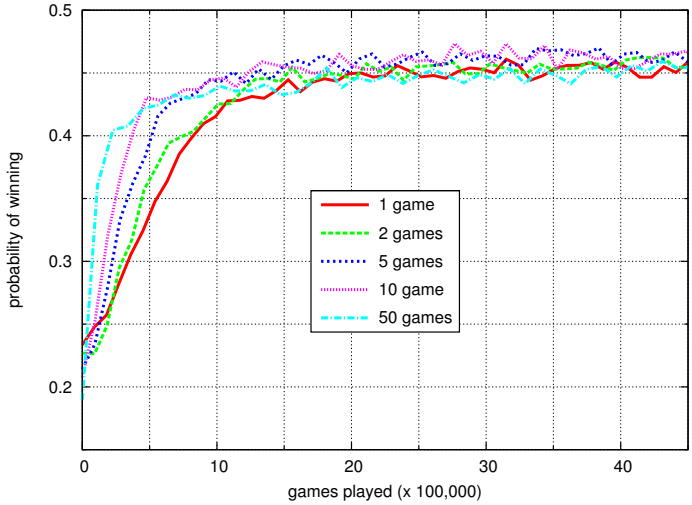


Figure 7. Average probability of winning of the best-of-generation individuals found by CTDL+HoF against the standard **heuristic** player for different TDL-CEL ratios, plotted against the number of training games played

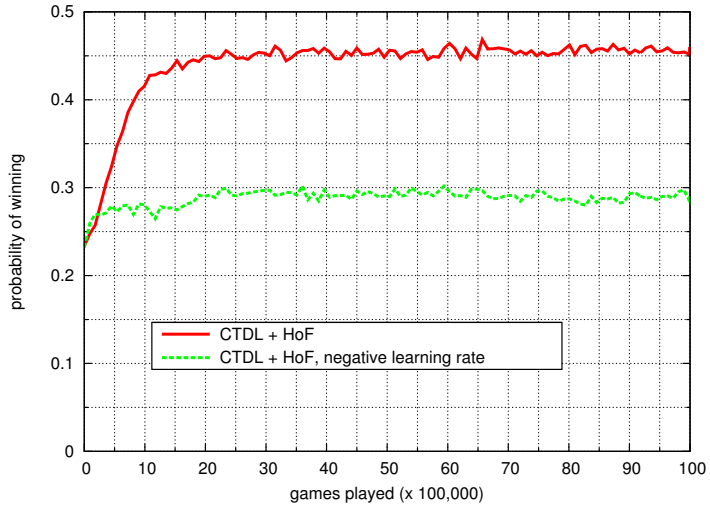


Figure 8. Average probability of winning of the best-of-generation individuals against the standard **heuristic** player found by CTDL+HoF for different learning rates of TDL, plotted against the number of training games played

of last-generation individuals obtained in setups with different TDL-CEL ratios, which ended with each team scoring a very similar number of points.

6.2. Negative learning rate

As shown in Section 5, the hybrid approach was able to learn remarkably better strategies than the non-hybrid methods. An interesting question is whether the purposeful (meaning: driven towards a greater probability of winning) character of changes brought in by TDL is essential, or TDL plays the role of a mere random mutation. To verify this, we compared regular CTDL+HoF to CTDL+HoF with learning rate $\alpha = -0.01$, meaning that TDL in the latter method disproves the strategies found by CEL. The results, shown in Fig. 8, show that a purposeful TDL is one of the key factors explaining the success of the hybrid approach.

6.3. Strategy initialization

A few additional experiments were motivated about by observations made during the preliminary tests. Particularly, it was noticed that the way strategies are initialized may have a substantial impact on the learning process. Recall from 4.1.1 that in case of TDL-based approaches initialization procedure sets all weights to zero. This setting was adapted from the previous work (Lucas and

Runarsson, 2006) where it is reported that such initialization leads to the best results. However, in coevolution, like in traditional evolutionary approaches, random initialization is most common for generating initial population as it distributes the initial population over different parts of the search space, helping to explore it. Evolutionary computation orthodoxy states that placing all initial individuals in the same point of the search space is a nonsense. Surprisingly, the following results imply that for this problem and a specific strategy representation, even population-based algorithms should start with zeroed candidate solutions.

Figs. 9 and 10 show the results of CTDL+HoF and CEL+HoF for three different initialization procedures. *Random-init* draws each weight uniformly from the given range ($[-1, 1]$ and $[-0.05, 0.05]$ were tested) while *zero-init* just sets all weights to zero for all the individuals. The primary observation is that proper initialization can largely affect the convergence speed of learning. Clearly, random-init results in slower convergence of both algorithms. Although narrowing the range from which the weights are drawn leads to improvement in learning speed, zero-init remains the best approach.

Finding the superiority of single point initialization quite unexpected, we tried to explain it by examining in detail the evolution of Othello strategies. Figs. 11 and 12 show how the board evaluation function of the best-of-generation individual changes throughout the learning by CTDL+HoF, for different initialization types. Although the final results in both figures are almost identical, zero-init finds a good approximation of the final solution to the last one much faster than random-init. This corresponds to differences in the convergence speed of these methods. Our analysis suggests also that another key characteristic of a well-playing WPC strategy is small magnitude of weights. The zero-initialized strategy naturally meets this requirement, in contrast to the random one which has larger weight variation. Fig. 12 illustrates that the evolutionary effort during the learning process is focused mostly on reducing weight values for the middle locations.

7. The interplay between TDL and CEL

In CTDL, the working set of solutions (population) is shared between two fundamentally different search algorithms. As the experimental results demonstrate, the interplay between them is beneficial. However, a deeper analysis that is beyond the scope of this paper leads to conclusion that both constituent methods, TDL and CEL, optimize towards substantially different goals. Thus, reaching a deeper understanding of this interplay is far from trivial and requires further studies. This section shortly discusses another observations that make it intriguing.

CTDL can be considered as a form of *Coevolutionary Memetic Algorithm*. Memetic Algorithms are hybrid approaches coupling a population-based global search method with some form of local improvement. Since these algorithms

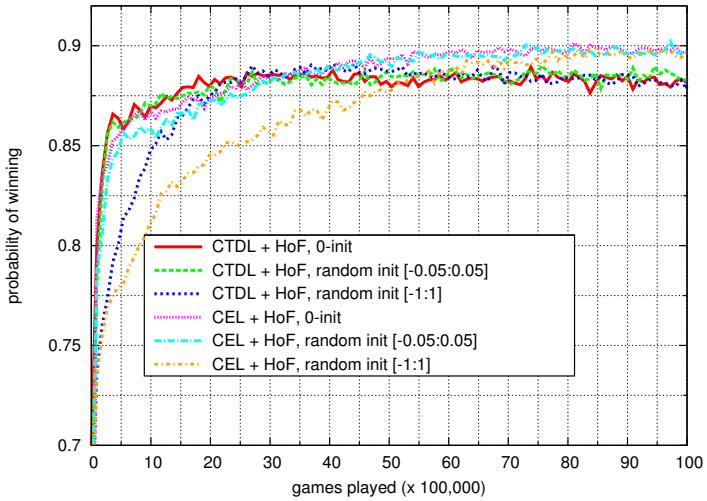


Figure 9. Average probability of winning of the best-of-generation individuals found by CTDL+HoF and CEL+HoF methods against the **random** player for different initialization procedures, plotted against the number of training games played

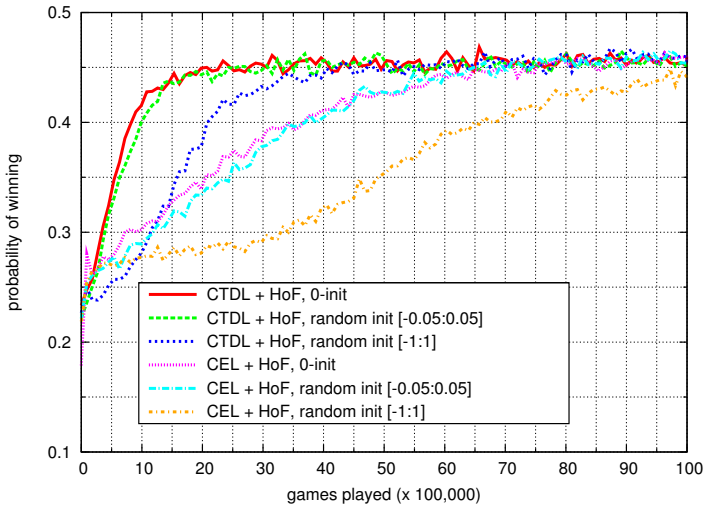


Figure 10. Average probability of winning of the best-of-generation individuals found by CTDL+HoF and CEL+HoF methods against the standard **heuristic** player for different initialization procedures, plotted against the number of training games played

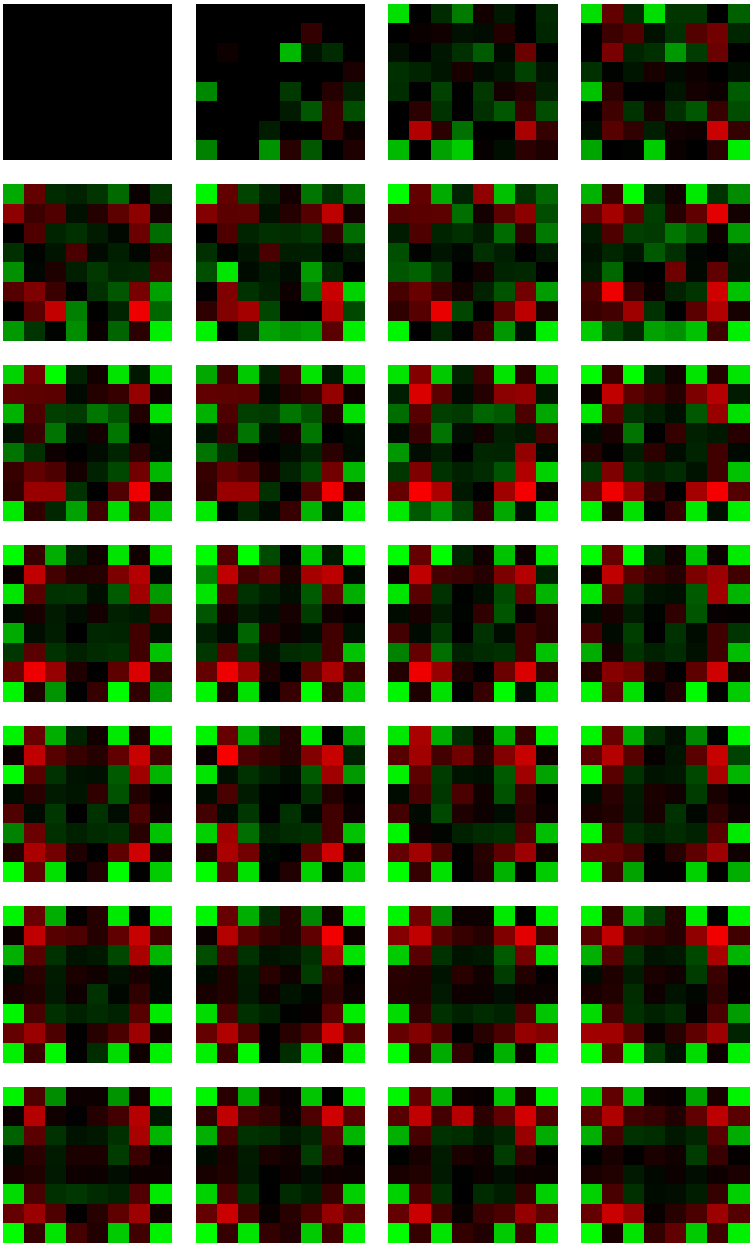


Figure 11. Evolution of **zero-initialized** population illustrated as a sequence of Othello boards (one drawing per 30 generations) colored accordingly to corresponding WPC weights (darker squares denote larger weights) of the best-of-generation individual. Read left to right, top to bottom

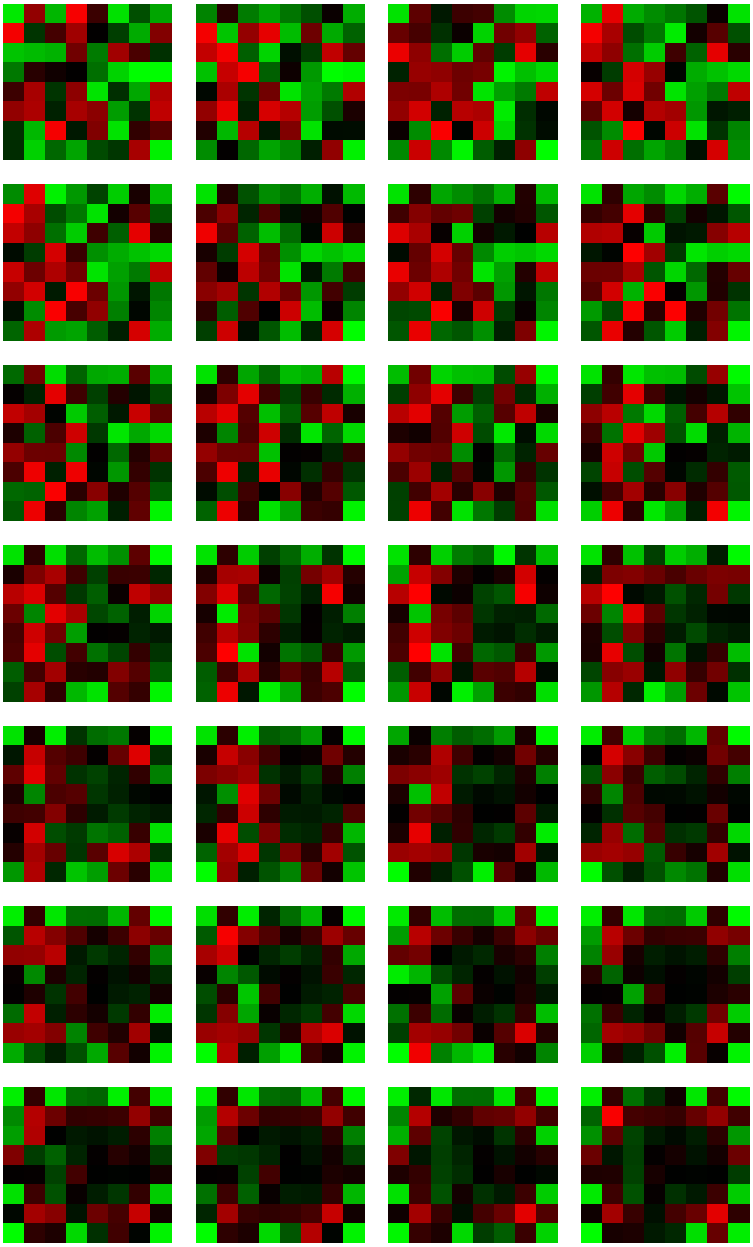


Figure 12. Evolution of **randomly initialized** population illustrated as a sequence of Othello boards (one drawing per 30 generations) colored accordingly to corresponding WPC weights (darker squares denote larger weights) of the best-of-generation individual. Read left to right, top to bottom

usually employ evolutionary search, they are often referred to as *Lamarckian Evolution*, to commemorate Jean-Baptiste Lamarck who hypothesized, incorrectly in view of today's neo-Darwinism, that the traits acquired by an individual during its lifetime can be passed on to its offspring. Technically, memetic algorithms typically alternate genetic search for the population and local search for individual solutions.

In this context, our Coevolutionary Temporal Difference Learning is an example of what might be termed *Lamarckian Coevolution* or *Lamarckian Coevolutionary Algorithm*. The reinforcement learning phase can be treated as a form of local search technique, especially as $TD(0)$ we use is a gradient descent method. In other words, $TD(0)$ combined with a randomly perturbed self-play serves as a substitute for a local search guided by the objective fitness function. Thus, it interestingly turns out that we can do a kind of local search without objective information about the solution performance. This sounds both puzzling and appealing, as normally an objective quality measure is an indispensable prerequisite for local search. We plan to elaborate on this observation in further research and hypothesize that some findings from the Memetic Algorithms literature are potentially applicable to our approach.

Another observation that supports the analogy between Lamarckian Evolutionary Algorithms and Lamarckian Coevolution is that, in both approaches, the parameters controlling the relative intensity of local and population learning (called here TDL-CEL ratio) are essential for effective learning. They define the tradeoff between exploration and exploitation. Too intensive reinforcement learning (exploitation) can lead to a premature convergence to a local optimum, making it difficult for the coevolutionary stage (exploration) to move towards better solutions. Too intense coevolution, on the other hand, does not give the reinforcement learning enough time to tune the strategies. The issue of an optimum exploration-exploitation balance requires more research.

8. Summary

CTDL and the results it attains for the nontrivial domain of the game of Othello constitute yet another argument in favour of the proposition that a smart hybrid of appropriately assorted global search and local search methods can result in a beneficial tradeoff between exploration and exploitation of the search space. Thanks to the coevolutionary component, CTDL is able to perform an effective search in absence of an objective performance measure (fitness function), which makes it significantly different from analogous non-coevolutionary hybrid methods such as memetic algorithms.

CTDL clearly benefits from the mutually complementary characteristics of both constituent approaches, CEL and TDL. The experimental results demonstrate that the fusion of these methods is indeed synergistic, leading to better performance of the co-evolved players for the game of Othello. The evolved learners reveal also basic 'understanding' of the game, such as the importance

of the board corners. Although in this paper we tested CTDL on Othello only, similar results obtained on small-board Go (Krawiec at al., 2011) makes us believe that CTDL could be beneficial for a wide spectrum of games where both TDL and CEL can be applied separately.

Apart from the tradeoff between exploration and exploitation discussed in Section 7, there are other aspects of this approach that are worth further investigation. In particular, using CTDL together with a two-population coevolution, with solutions and tests bred separately, would allow us to employ more advanced coevolutionary archive methods such as LAPCA and IPCA (de Jong, 2007) and potentially obtain better results.

Acknowledgments

This work was supported in part by Ministry of Science and Higher Education grant # N N519 4419 39. Wojciech Jaśkowski gratefully acknowledges a financial support from grant # N N516 1883 37.

References

Angeline, P.J. and Pollack, J.B. (1993) Competitive Environments Evolve Better Solutions for Complex Tasks. In: S. Forrest, ed., *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*. Morgan Kaufmann, University of Illinois at Urbana-Champaign, 264-270.

Archer, K.J. and Kimes, R.V. (2008) Empirical Characterization of Random Forest Variable Importance Measures. *Comp. Stat. & Data Anal.*, **52**(4), 2249-2260.

Azaria, Y. and Sipper, M. (2005) GP-Gammon: Genetically Programming Backgammon Players. *Genetic Programming and Evolvable Machines*, **6**(3), 283-300.

Bucci, A. (2007) *Emergent Geometric Organization and Informative Dimensions in Coevolutionary Algorithms*. Ph.D. thesis, Michtom School of Computer Science, Brandeis University.

Buro, M. (2002) Improving heuristic mini-max search by supervised learning. *Artificial Intelligence* **134** (1-2), 85–99.

Caverlee, J.B. (2000) A Genetic Algorithm Approach to Discovering an Optimal Blackjack Strategy. In: Koza, J.R., ed., *Genetic Algorithms and Genetic Programming at Stanford 2000*. Stanford Bookstore, Stanford, California, 70-79.

de Jong, E.D. (2005) The MaxSolve algorithm for coevolution. In: H.G. Beyer et al., eds., *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, vol. 1. ACM Press, Washington DC, USA, 483-489.

de Jong, E.D. (2007) A Monotonic Archive for Pareto-Coevolution. *Evolutionary Computation*, **15**(1), 61-93.

FICICI, S.G. (2004) *Solution concepts in coevolutionary algorithms.* Ph.D. thesis, Waltham, MA, USA.

FICICI, S.G. and POLLACK, J.B. (2003) A Game-Theoretic Memory Mechanism for Coevolution. In: E. Cantú-Paz et al., eds., *Genetic and Evolutionary Computation - GECCO 2003.* **LNCS 2723**. Springer, Chicago, IL, 286-297.

FOGEL, D.B. (2001) *Blondie24: Playing at the Edge of AI.* Morgan Kaufmann Publishers.

HAUPTMAN, A. and SIPPER, M. (2007) Evolution of an Efficient Search Algorithm for the Mate-In-N Problem in Chess. In: M. Ebner et al., eds., *Proceedings of the 10th European Conference on Genetic Programming.* **LNCS 4445**. Springer, Valencia, Spain, 78-89.

JAŚKOWSKI, W., KRAWIEC, K. and WIELOCH, B. (2008) Evolving Strategy for a Probabilistic Game of Imperfect Information using Genetic Programming. *Genetic Programming and Evolvable Machiness*, **9**(4), 281-294.

KIM, K.-J., CHOI, H. and CHO, S.-B. (2007) Hybrid of Evolution and Reinforcement Learning for Othello Players. *IEEE Symposium on Computational Intelligence and Games*, 203-209.

KRAWIEC, K., JAŚKOWSKI, W. and SZUBERT, M. (2011) Evolving Small-Board Go Players using Coevolutionary Temporal Difference Learning with Archive. *International Journal of Applied Mathematics and Computer Science*, **21**(4).

LEE, K.-F. and MAHAJAN, S. (1990) The development of a world class Othello program. *Artificial Intelligence Journal*, **43**(1), 21-36.

LUBBERTS, A. and MIIKKULAINEN, R. (2001) Co-Evolving a Go-Playing Neural Network. In: R.K. Belew & H. Juillé, eds., *Coevolution: Turning Adaptive Algorithms upon Themselves.* Birds-of-a-Feather Workshop. GECCO 2001, 14-19. http://nn.cs.utexas.edu/downloads/papers/lubberts.coevolution-gecco01.pdf

LUCAS, S.M. and RUNARSSON, T.P. (2006) Temporal difference learning versus co-evolution for acquiring othello position evaluation. *IEEE Symposium on Computational Intelligence and Games.* IEEE. 52-59.

LUKE, S. (1998) Genetic Programming Produced Competitive Soccer Softbot Teams for RoboCup97. In: J.R. Koza et al., eds., *Genetic Programming 1998: Proceedings of the Third Annual Conference.* Morgan Kaufmann, University of Wisconsin, Madison, 214-222.

LUKE, S. (2008) ECJ 18 - A Java-based Evolutionary Computation Research System. http://cs.gmu.edu/ eclab/projects/ecj/.

LUKE, S. and WIEGAND, R.P. (2002) When coevolutionary algorithms exhibit evolutionary dynamics. In: A.M. Barry, ed., *GECCO 2002: Proc. of the Bird of a Feather Workshops, Genetic and Evolutionary Computation conference.* AAAI, Menlo Park, CA, 236-241.

Manning, E.P. (2007) Temporal Difference Learning of an Othello Evaluation Function for a Small Neural Network with Shared Weights. *IEEE Symposium on Computational Intelligence and Games*, 216-223.

Miconi, T. (2009) Why Coevolution Doesn't "Work": Superiority and Progress in Coevolution. In: *Proc. of the 12th European Conference on Genetic Programming*. **LNCS 5481**, Springer, 49-60.

Monroy, G.A., Stanley, K.O. and Miikkulainen, R. (2006) Coevolution of neural networks using a layered Pareto archive. In: M. Keijzer et al., eds., *GECCO 2006: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, vol. 1. ACM Press, Seattle, Washington, USA, 329-336.

Pollack, J.B. and Blair, A.D. (1998) Co-Evolution in the Successful Learning of Backgammon Strategy. *Machine Learning*, **32**(3), 225-240.

Rosin, C.D. and Belew, R.K. (1997) New Methods for Competitive Coevolution. *Evolutionary Computation*, **5**(1), 1-29.

Runarsson, T.P. and Lucas, S.M. (2005) Coevolution versus self-play temporal difference learning for acquiring position evaluation in small-board go. *IEEE Transactions on Evolutionary Computation*, **9**(6), 628-640.

Samuel, A.L. (1959) Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, **3**(3), 211-229.

Singer, J.A. (2001) Co-evolving a Neural-Net Evaluation Function for Othello by Combining Genetic Algorithms and Reinforcement Learning. *Proc. of International Conference on Computational Science (2). Part II.* **ICCS 2001**, Springer, London, 377-389.

Stanley, K.O., Bryant, B, and Miikkulainen, R. (2005) Real-time neuroevolution in the NERO video game. *IEEE Transactions on Evolutionary Computation*, **9**(6), 653-668.

Sutton, R.S. (1988) Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, **3**, 9-44.

Sutton, R.S. and Barto, A.G. (1998) *Reinforcement Learning*. Vol. 9. MIT Press.

Szubert, M., Jaśkowski, W. and Krawiec, K. (2009) Coevolutionary Temporal Difference Learning for Othello. *IEEE Symposium on Computational Intelligence and Games*, 104-111.

Tesauro, G. (1995) Temporal difference learning and TD-Gammon. *Communications of the ACM*, **38**(3), 58-68.