Coevolutionary Temporal Difference Learning for Othello

Marcin Szubert, Wojciech Jaśkowski, and Krzysztof Krawiec

***Abstract*— This paper presents Coevolutionary Temporal Difference Learning (CTDL), a novel way of hybridizing coevolutionary search with reinforcement learning that works by interlacing one-population competitive coevolution with temporal difference learning. The coevolutionary part of the algorithm provides for exploration of the solution space, while the temporal difference learning performs its exploitation by local search. We apply CTDL to the board game of Othello, using weighted piece counter for representing players' strategies. The results of an extensive computational experiment demonstrate CTDL's superiority when compared to coevolution and reinforcement learning alone, particularly when coevolution maintains an archive to provide historical progress. The paper investigates the role of the relative intensity of coevolutionary search and temporal difference search, which turns out to be an essential parameter. The formulation of CTDL leads also to the introduction of Lamarckian form of coevolution, which we discuss in detail.**

I. 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 that this will change thanks to the methods that *learn* the strategies automatically with little *a priori* domain knowledge. Two intensely studied examples of such methods 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-based method. 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 [1], [2].

Here, we ask the question whether it is possible to combine the advantages of TDL and CEL in a single algorithm that would develop better strategies than any of these methods on its own. To this aim, we propose a hybrid method referred to as Coevolutionary Temporal Difference Learning (CTDL) and evaluate it on the game of Othello.

This paper is organized as follows. In Section II we describe rules, strategy representation, and previous research on learning Othello strategies. Section III reviews TDL and CEL and introduces CTDL. After describing the experimental setup in Section IV, we discuss the results in Sections V and VI, to conclude in Section VII.

M. Szubert, W. Jaśkowski, and K. Krawiec are with the Institute of Computing Science, Poznan University of Technology, Piotrowo 2, 60965 Poznań, Poland; email: {wjaskowski,kkrawiec}@cs.put.poznan.pl, marcin.g.szubert@gmail.com.

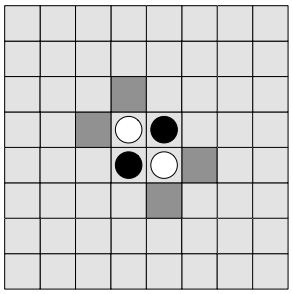


Fig. 1. The initial board configuration in Othello

II. 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.

A. 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. Figure 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 of 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

978-1-4244-4815-9/09/$25.00 ©2009 IEEE

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.

B. Strategy Representation

One of the main issues to consider when learning game strategy is the architecture of the learner, which is mainly determined by a strategy representation. There is a multitude of reasonable strategy representations; here, 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) representation, 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 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. Alternatively, 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 resulting in a very fast board evaluation. Moreover, a WPC strategy can be often easily interpreted just by inspecting the weight values. Table I presents the weight matrix of an exemplary player that clearly focuses at taking possession of the corners because they are given the highest values.

TABLE I
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

C. Previous Research

The game of Othello has been a subject of computational intelligence research for more than 20 years. The significant interest in this game may be explained by its large 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 [3]. 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 [4], 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 [5].

Recently, the mainstream research on Othello has moved towards better understanding of which types of learning algorithms and player architectures work the best. The CEC Othello Competitions [6] pursued this direction by limiting the ply depth to one, effectively disqualifying the algorithms that employ a brute-force game tree search. Although the WPC representation 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 multilayer perceptron (MLP), and a spatial MLP.

The most challenging scenario of elaborating game strategy is learning without any reference to human knowledge or game strategy given a priori. This task formulation is addressed by, among others, Temporal Difference Learning (TDL) and Coevolutionary Learning (CEL), which were investigated in the context of Othello by Lucas and Runarsson [2]. That study inspired our research and will be also referred to in the following section.

III. METHODS

A. Coevolutionary Learning

Coevolutionary algorithms are variants of evolutionary computation where individual's fitness depends on other individuals. The 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 [7], [8]).

Coevolutionary Learning (CEL) follows the competitive evaluation 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. Though this general scheme seems straightforward, it misses many details that need to be filled in, some of which relate to evolutionary computation (population size, variation operators, selection scheme, etc.), while some others pertain 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 [9] used one of the simplest evolutionary algorithm, a random hill-climber to successfully address the problem of learning backgammon strategy. In the referred work [2], Lucas and Runarsson used (1 + λ) and (1, λ) 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 it was used only for evolving small-board Go players [1].

Various forms of CEL have been successfully applied to many two-person games, including Backgammon [10], Chess [11], Checkers [12], NERO [13], Blackjack [14], Pong [15], AntWars [16], [17] and a small version of Go [18].

B. Coevolutionary Archives

The central characteristic of CEL is that it refrains from the use of the *objective fitness* of individuals. This feature makes it appealing for applications where objective fitness cannot be unarguably defined or is costly 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 [19].

In order to deal with coevolutionary pathologies, *coevolutionary archives* that try to sustain progress 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, who are replaced occasionally, typically when they prove inferior to some population members. Of course, an archive still 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* [20].

In this study we use Hall of Fame (HoF, [21]), one of the simplest forms of archive. 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 homogenous population of players, a setup called *one-population coevolution* [22] or *competitive fitness environment* [7], [23]. It is worth to point out that the recent work on coevolution indicates that, even if the game itself is symmetric, it is worth to maintain in parallel two types of strategies: *solutions*, which are expected to improve as evolution proceeds, and *tests*, whose main purpose is to differentiate solutions by defeating some of them. Recent contributions [24], [25], [26] 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.

C. Temporal Difference Learning

Temporal Difference (TD), a method proposed by Sutton in 1988 [27], has become a popular approach for solving reinforcement learning tasks. Some suggest [28] that the famous chess playing program by Samuel [29] in 1959 was in fact taught by a simple version of temporal difference learning (however others [30] treat it rather as a first example of coevolution). One of the most spectacular successes of temporal difference learning in game playing is undoubtedly Tesauro's TD-Gammon [31]. This influential work has triggered off a lot of research in reinforcement learning and TD methods, including their applications to Othello [32], [33].

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's 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, 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:

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

where λ is the learning rate, 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. 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:

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

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:

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

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:

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

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. This is a popular technique whose major advantage is that it does not need anything besides the learning system itself. 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 certain probability. After such a random move, no weight update occurs.

D. Coevolutionary Temporal Difference Learning

The past results of learning WPC strategies for Othello [2] and small-board Go [1] 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 it fails to produce a well-performing strategy. CEL progresses slower, but, if properly tuned, eventually outperforms TDL. Therefore, 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 TD learning and coevolutionary learning. In the TD 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 standard selection and variation operators and the cycle repeats.

Other hybrids of TDL and CEL have been occasionally considered in the past. Kim *et al.* [34] 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. In [35], Singer has shown that reinforcement learning may be superior to 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 learning 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.

IV. Experiments

We conducted several experiments comparing CTDL, CEL, TDL, and their extensions with the Hall of Fame (HoF) archive [21], all implemented using *Evolutionary Computation in Java* (ECJ) library [36]. To provide fair comparison, all runs used the same settings (taken from [2] when possible) and stopped when the number of games played reached $4,500,000$. For statistical significance, each experiment was repeated 30 times.

A. Algorithms and setup

1) TDL: TDL is an implementation of a gradient-descent temporal difference algorithm $TD(0)$ described in Section III-C and parametrized as in [2]. 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 $\alpha = 0.01$.

2) CEL: CEL uses a generational coevolutionary algorithm with population of 50 individuals initialized with all weights set to 0[1]. 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 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 1800 generations to get the total of $4,500,000$ games.

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 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

[1]Unintuitively, we have found for Othello that initializing the weights with 0 for all individuals leads to better solutions than when weights were draw at random from $[-1, 1]$.

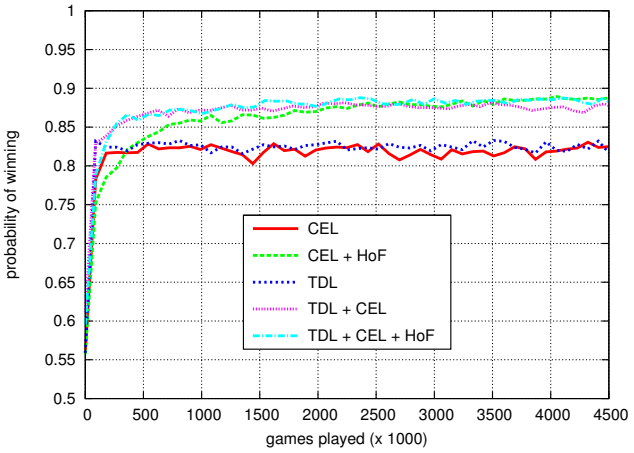


Fig. 2. Average performance of the best-of-generation individuals measured as a probability of winning against a **random** player, plotted against the number of training games played.

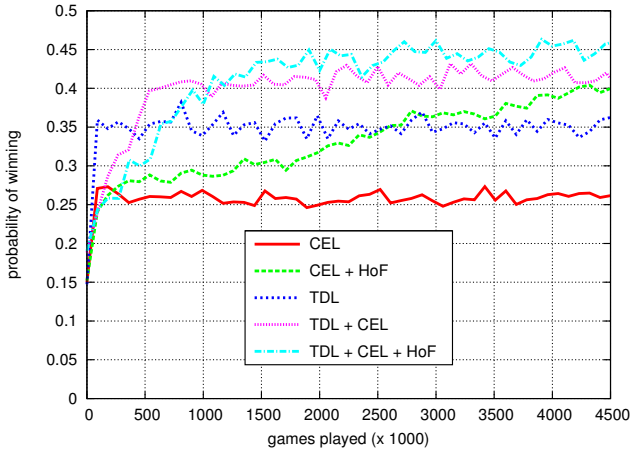


Fig. 3. Average performance of the best-of-generation individuals measured as a probability of winning against a standard **heuristic** player (both players randomized with probability 0.1), plotted against the number of training games played.

attain $4,500,000$ training games, the number of generations was set to 900.

4) CTDL = TDL + CEL: CTDL combines TDL and CEL as described in Section III-D, with the TDL phase parameters described in *1)* and CEL phase parameters described in *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 $4,500,000$. 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 1765 generations.

5) CTDL+HoF = TDL + CEL + HoF: This setup combines *3)* and *4)* and does not involve any extra parameters.

B. Measuring strategy quality

In order to monitor progress in an objective way, 50 times per run (approximately every $90,000$ of games) we assess the quality of the best-of-generation individual. As learning 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 [19]. This approach is impossible to implement in practice due to the inconceivably high number of possible strategies for Othello. Thus, following [2], 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.

1) Playing against a 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* [19].

2) Playing against the standard heuristic player: This method tests how well the player copes with a moderately competent opponent using WPC shown in Table I. 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 [2], we assume that the ability of playing such a randomized game is highly correlated with playing the original Othello.

V. Results

In the first experiment, we compared five methods described in the previous section. Figure 2 illustrates how the strategies produced by these algorithms perform on average against a random player. Note that for the population-based methods (i.e., all except pure TDL), each graph point represents the probability of winning of a best-of-generation individual averaged over 30 experimental runs. The best-of-generation individual was selected basing on the individuals' fitness. For TDL, the graphs show the average performance of the single solution 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 non-hybrid methods (CEL and TDL) are in the long run significantly worse than the hybrid ones. As expected, in the beginning the quality of individuals produced by the TDL-based algorithms is higher than of those produced by methods that do not involve TDL. In particular, TDL+CEL or TDL+CEL+HoF look superior, as they quickly achieve good performance and are best in the long run. Interestingly, CEL+HoF achieves eventually similar level of play as TDL+CEL(+HoF), but it learns significantly slower.

Figure 3 illustrates progress measured as the quality of best-of-generation individuals versus the standard heuristic

TABLE II
THE RESULTS OF THE ROUND-ROBIN TOURNAMENT OF BEST-OF-RUN INDIVIDUALS.

Team	Games	Wins	Draws	Defeats	Points
TDL+CEL+HoF	7200	4918	203	2079	14957
TDL+CEL	7200	4171	209	2820	12722
CEL+HoF	7200	3726	225	3249	11403
TDL	7200	2873	206	4121	8825
CEL	7200	1787	207	5206	5568

TABLE III
WEIGHTED PIECE COUNTER VECTOR OF THE BEST EVOLVED PLAYER.

1.02	-0.27	0.55	-0.10	0.08	0.47	-0.38	1.00
-0.13	-0.52	-0.18	-0.07	-0.18	-0.29	-0.68	-0.44
0.55	-0.24	0.02	-0.01	-0.01	0.10	-0.13	0.77
-0.10	-0.10	0.01	-0.01	0.00	-0.01	-0.09	-0.05
0.05	-0.17	0.02	-0.04	-0.03	0.03	-0.09	-0.05
0.56	-0.25	0.05	0.02	-0.02	0.17	-0.35	0.42
-0.25	-0.71	-0.24	-0.23	-0.08	-0.29	-0.63	-0.24
0.93	-0.44	0.55	0.22	-0.15	0.74	-0.57	0.97

player. The results are clear: TDL+CEL+HoF is the best, then TDL+CEL, TDL, CEL+HoF and CEL, which is the worst here. It can be also observed that the HoF archive helps both CEL and TDL+CEL to achieve a higher level of play. Nevertheless, regarding CEL+HoF setup, it needs approximately 10 times more games than the simplest hybrid approach (TDL+CEL), to reach comparable performance. Once again TDL starts learning rapidly, but it stagnates after several thousands of games.

A. Round-robin tournament of best-of-run individuals

In order to confirm our results, we performed a round-robin tournament between all best-of-run individuals, i.e., the best-of-generation individuals from the last generation. We created five teams, one for each method, each one composed of 30 best-of-run individuals (one per run). Next, a round-robin tournament was played, where each strategy played against $4 \times 30 = 120$ strategies from the opponent teams for a total of 240 games (120 as white and 120 as black). The final score of a team was determined as the sum of points obtained by its players in overall $7,200$ games.

The results of this competition, presented in Table II, confirm the former observations: the best method in direct comparison is TDL+CEL+HoF. Moreover, the ranking of methods is consistent with the ranking obtained from measuring the quality against a standard heuristic player (Fig. 3). This may be explained by the fact that in this tournament, an individual faces exclusively well-performing strategies, so what matters here is the ability to play against a competent opponent and not against a random one.

The WPC vector of the best scoring player, who is also a member of the winner team TDL+CEL+HoF, is shown in Table III and presented graphically by means of a weight-proportional grayscale in Fig. 4b (darker squares denote larger weights, i.e., more desirable locations on the board). An important observation is that the WPC matrix is quite symmetric. Similarly to the heuristic player's strategy, which is shown graphically in Fig. 4a, the corner locations are the most desirable, while their immediate neighbors have very low weights. However, in contrast to the heuristic player, the edge locations at distance 2 from the corners get very high weights.

B. 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 the best algorithm (TDL+CEL+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. 5, proves that CTDL performance indeed depends on the TDL-CEL ratio and that the ratio greater than 1 is detrimental. This figure demonstrates also the tradeoff between the learning speed and the ultimate player quality.

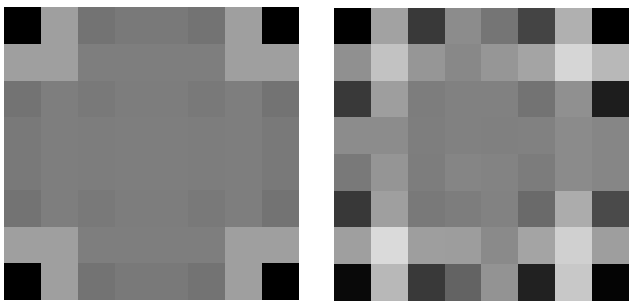
In Fig. 6, which illustrates the performance against the standard heuristic player, different TDL-CEL ratios cause only slight differences in the long run. This was confirmed by playing another best-of-run tournament between setups with different TDL-CEL ratios, which ended with each team scoring a very similar number of points.

C. Negative learning rate

As we have seen above, 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 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 TDL+CEL+HoF to TDL+CEL+HoF with learning rate $\alpha = -0.01$, which implies that TDL in the latter method deteriorates the strategies found by CEL. The results, shown in Fig. 7, prove that a purposeful TDL is one of the key factors explaining the success of the hybrid approach.

VI. LAMARCKIAN COEVOLUTION PERSPECTIVE

The algorithm proposed in this paper can be considered as a form of *Coevolutionary Memetic Algorithm*. Memetic Al-



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

Fig. 4. Weighted Piece Counter vectors illustrated as Othello boards with locations shaded accordingly to corresponding weights.

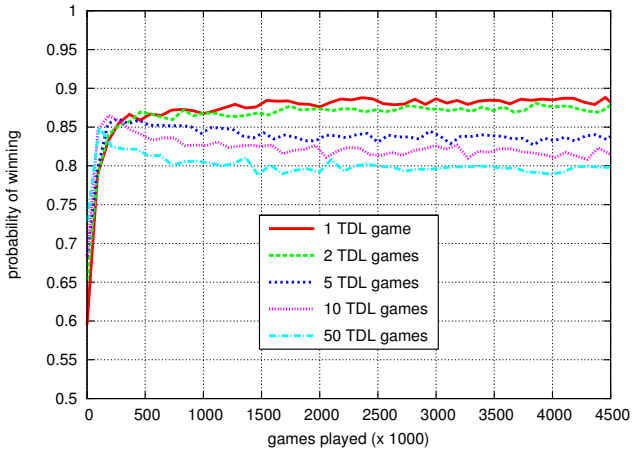


Fig. 5. Average probability of winning of the best-of-generation individuals found by TDL+CEL+HoF against a **random** player for different TDL-CEL ratios, plotted against the number of training games played.

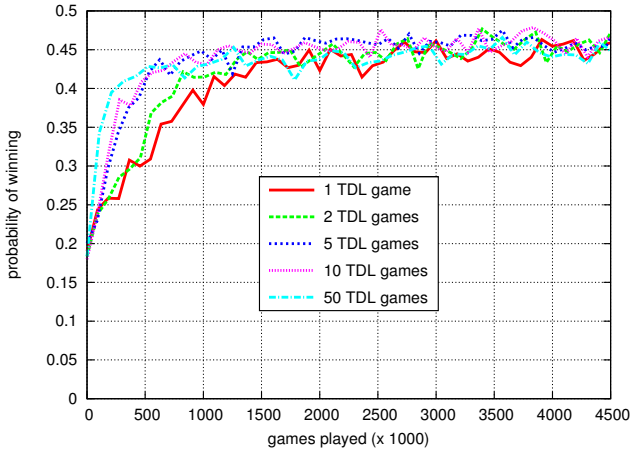


Fig. 6. Average probability of winning of the best-of-generation individuals found by TDL+CEL+HoF against a standard **heuristic** player for different TDL-CEL ratios, plotted against the number of training games played.

gorithms are hybrid approaches coupling a population-based global search method with some form of local improvement. Since these algorithms usually employ evolutionary search, they are often referred to as *Lamarckian Evolution* or *Genetic Local Search.* Technically, they 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 solution performance. This sounds both puzzling and appealing, as normally an objective quality measure is an indispensable prerequisite

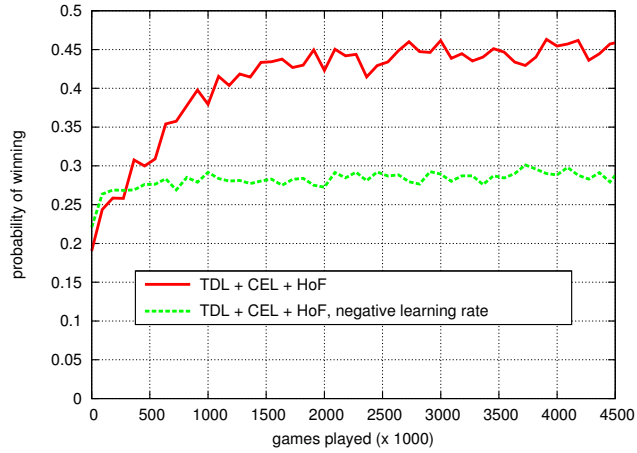


Fig. 7. Average probability of winning of the best-of-generation individuals against a standard **heuristic** player found by TDL+CEL+HoF for different learning rates of TDL, plotted against the number of training games played.

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 that control the relative intensity of local learning and population learning (called here TDL-CEL ratio) are essential for effective learning. They define the inherent tradeoff between exploration and exploitation. Too intensive reinforcement learning stage (exploitation) can lead to premature convergence to a local optimum, making it difficult for the coevolutionary stage (exploration) to move towards better solutions. Too intense coevolution phase, 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.

VII. Summary

This study presented CTDL, a novel way of hybridizing coevolutionary search with reinforcement learning designed to benefit from mutually complementary characteristics of both approaches. 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 we tested CTDL on Othello only, we hypothesize that CTDL could be also beneficial for other games where both TDL and CEL can be applied separately.

Apart from the trade-off between exploration and exploitation discussed in Section VI, 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 open the possibility of using more advanced coevolutionary archive

methods such as LAPCA and IPCA [37] and potentially obtaining better results.

ACKNOWLEDGMENTS

This work was supported in part by Ministry of Science and Higher Education grant # N N519 3505 33 and grant POIG.01.01.02-00-014/08-00. Wojciech Jaśkowski gratefully acknowledges a scholarship from Sectoral Operational Programme 'Human Resources Development', Activity 8.2 co-financed by the European Social Fund European Union and the Government of Poland.

REFERENCES

[1] T. P. Runarsson and S. Lucas, "Co-evolution versus self-play temporal difference learning for acquiring position evaluation in small-board Go," *IEEE Transactions on Evolutionary Computation*, vol. 9, 2005.

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

[3] K.-F. Lee and S. Mahajan, "The development of a world class othello program," *Artif. Intell.*, vol. 43, no. 1, pp. 21–36, 1990.

[4] M. Buro, "Logistello: A strong learning othello program," in *19th Annual Conference Gesellschaft für Klassifikation e.V.*, 1995.

[5] ——, "Takeshi Murakami vs. Logistello," 1997.

[6] [Online]. Available: http://algoval.essex.ac.uk:8080/othello/html/Othello.html

[7] P. J. Angeline and J. B. Pollack, "Competitive environments evolve better solutions for complex tasks," in *Proceedings of the 5th International Conference on Genetic Algorithms*, S. Forrest, Ed., 1993, pp. 264–270.

[8] Y. Azaria and M. Sipper, "GP-gammon: Genetically programming backgammon players," *Genetic Programming and Evolvable Machines*, vol. 6, no. 3, pp. 283–300, 2005.

[9] J. B. Pollack and A. D. Blair, "Co-evolution in the successful learning of backgammon strategy," *Machine Learning*, vol. 32, no. 3, pp. 225–240, 1998.

[10] ——, "Co-evolution in the successful learning of backgammon strategy," *Machine Learning*, vol. 32, no. 3, pp. 225–240, 1998.

[11] A. Hauptman and M. Sipper, "Evolution of an efficient search algorithm for the mate-in-N problem in chess," in *Proceedings of the 10th European Conference on Genetic Programming*, ser. LNCS, M. E. et. al, Ed., vol. 4445, 2007, pp. 78–89.

[12] D. B. Fogel, *Blondie24: playing at the edge of AI*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.

[13] K. Stanley, B. Bryant, and R. Miikkulainen, "Real-time neuroevolution in the nero video game," *Evolutionary Computation, IEEE Transactions on*, vol. 9, no. 6, pp. 653–668, 2005.

[14] J. B. Caverlee, "A genetic algorithm approach to discovering an optimal blackjack strategy," in *Genetic Algorithms and Genetic Programming at Stanford 2000*, J. R. Koza, Ed. Stanford, California, 94305-3079 USA: Stanford Bookstore, June 2000, pp. 70–79.

[15] G. A. Monroy, K. O. Stanley, and R. Miikkulainen, "Coevolution of neural networks using a layered pareto archive," in *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 2006, pp. 329–336.

[16] W. Jaśkowski, K. Krawiec, and B. Wieloch, "Evolving strategy for a probabilistic game of imperfect information using genetic programming," *Genetic Programming and Evolvable Machiness*, vol. 9, no. 4, pp. 281–294, 2008.

[17] W. Jaśkowski, K. Krawiec, and B. Wieloch, "Winning ant wars: Evolving a human-competitive game strategy using fitnessless selection," in *Genetic Programming*, ser. LNCS, M. O'Neill, L. Vanneschi, S. Gustafson, A. I. E. Alcázar, I. D. Falco, A. D. Cioppa, and E. Tarantino, Eds., vol. 4971. Springer, 2008, pp. 13–24, lNCS49710013.

[18] A. Lubberts and R. Miikkulainen, "Co-evolving a go-playing neural network," in *Coevolution: Turning Adaptive Algorithms upon Themselves*, R. K. Belew and H. Juillè, Eds., San Francisco, California, USA, 7 July 2001, pp. 14–19.

[19] S. G. Ficici, "Solution concepts in coevolutionary algorithms," Ph.D. dissertation, Waltham, MA, USA, 2004.

[20] T. Miconi, "Why coevolution doesn't "work": Superiority and progress in coevolution," in *EuroGP 2009*, 2009.

[21] C. D. Rosin and R. K. Belew, "New methods for competitive coevolution," *Evolutionary Computation*, vol. 5, no. 1, pp. 1–29, 1997.

[22] S. Luke and R. Wiegand, "When coevolutionary algorithms exhibit evolutionary dynamics," in *Workshop on Understanding Coevolution: Theory and Analysis of Coevolutionary Algorithms (at GECCO 2002)*, A. Barry, Ed. New York: AAAI Press, 2002, pp. 236–241.

[23] S. Luke, "Genetic programming produced competitive soccer softbot teams for robocup97," in *Genetic Programming 1998: Proceedings of the 3rd Annual Conference*, J. R. K. et. al, Ed., Madison, Wisconsin, USA, 1998, pp. 214–222.

[24] S. Ficici and J. Pollack, "A game-theoretic memory mechanism for coevolution," in *Genetic and Evolutionary Computation*, ser. Lecture Notes in Computer Science, vol. 2723, 2003, pp. 286–297.

[25] E. D. de Jong, "The MaxSolve algorithm for coevolution," in *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, 2005, pp. 483–489.

[26] E. D. de Jong, "A Monotonic Archive for Pareto-Coevolution," *Evolutionary Computation*, vol. 15, no. 1, pp. 61–93, Spring 2007.

[27] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, 1988.

[28] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[29] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 44, no. 1, pp. 206–227, 1959.

[30] A. Bucci, "Emergent geometric organization and informative dimensions in coevolutionary algorithms," Ph.D. dissertation, Waltham, MA, USA, 2007.

[31] G. Tesauro, "Temporal difference learning and td-gammon," *Commun. ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[32] A. Leouski, "Learning of position evaluation in the game of othello," Tech. Rep., 1995.

[33] E. P. Manning, "Temporal difference learning of an othello evaluation function for a small neural network with shared weights," 2007.

[34] K.-J. Kim, H. Choi, and S.-B. Cho, "Hybrid of evolution and reinforcement learning for othello players," *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, pp. 203–209, 2007.

[35] J. A. Singer, "Co-evolving a neural-net evaluation function for othello by combining genetic algorithms and reinforcement learning," in *International Conference on Computational Science (2)*, 2001, pp. 377–389.

[36] S. Luke, "Ecj 18 - a java-based evolutionary computation research system," *http://cs.gmu.edu/ eclab/projects/ecj/*, 2008.

[37] E. D. de Jong, "A monotonic archive for pareto-coevolution," *Evolutionary Computation*, vol. 15, no. 1, pp. 61–93, 2007.