Int. J. Appl. Math. Comput. Sci., 2011, Vol. 21, No. 4, 717–731
DOI: 10.2478/v10006-011-0057-3

EVOLVING SMALL–BOARD GO PLAYERS USING COEVOLUTIONARY TEMPORAL DIFFERENCE LEARNING WITH ARCHIVES

KRZYSZTOF KRAWIEC, WOJCIECH JAŚKOWSKI, MARCIN SZUBERT

Institute of Computing Science
Poznań University of Technology, ul. Piotrowo 2, 60–965 Poznań, Poland
e-mail: {kkrawiec,wjaskowski,mszubert}@cs.put.poznan.pl

We apply Coevolutionary Temporal Difference Learning (CTDL) to learn small-board Go strategies represented as weighted piece counters. CTDL is a randomized learning technique which interweaves two search processes that operate in the intra-game and inter-game mode. Intra-game learning is driven by gradient-descent Temporal Difference Learning (TDL), a reinforcement learning method that updates the board evaluation function according to differences observed between its values for consecutively visited game states. For the inter-game learning component, we provide a coevolutionary algorithm that maintains a sample of strategies and uses the outcomes of games played between them to iteratively modify the probability distribution, according to which new strategies are generated and added to the sample. We analyze CTDL's sensitivity to all important parameters, including the trace decay constant that controls the lookahead horizon of TDL, and the relative intensity of intra-game and inter-game learning. We also investigate how the presence of memory (an archive) affects the search performance, and find out that the archived approach is superior to other techniques considered here and produces strategies that outperform a handcrafted weighted piece counter strategy and simple liberty-based heuristics. This encouraging result can be potentially generalized not only to other strategy representations used for small-board Go, but also to various games and a broader class of problems, because CTDL is generic and does not rely on any problem-specific knowledge.

Keywords: temporal difference learning, coevolution, small-board Go, exploration vs. exploitation, games.

1. Introduction

Despite having been a subject of artificial intelligence research for more than 40 years, the game of Go remains a great challenge as the best computer players continue to yield to professionals. This is a result of the huge combinatorial complexity, which is much higher for this game than for other popular two-player deterministic board games—there are about 10^{170} board states and the game tree has an average branching factor of around 200. These figures, together with other specific features of Go, make it impossible to directly adopt techniques that proved successful in other board games, like chess or checkers (Mechner, 1998; Johnson, 1997).

Many of canonical Go-playing programs are precisely tuned expert systems founded on a thorough human analysis of the game. Such programs typically employ a multitude of rules elicited from professional Go players in order to recognize particular board patterns and react to them. However, this knowledge-based approach is constrained by the extent and quality of the available knowledge, and by the designer's ability to articulate it in a playing program. These shortcomings, which manifest themselves when applying the knowledge-based methodology to most games, turn out to be particularly painful for Go.

No wonder that some computer Go researchers abandon this cognitive perspective, which aims at mimicking expert reasoning, in favor of a behavioral perspective that does not care much whether the player's perception of the game state and internal strategy representation exhibit analogies to human players. Approaches that belong to the latter group typically assume that a player has no initial knowledge about game specifics (apart from the game definition) and involve some form of *learning* to automatically harvest it from the outcomes of games played against opponents. The major differences between the representatives of this trend consist in *when* (in the course of learning) the outcomes of such *interactions* are transformed into knowledge and *how* it is done.

Such a learning task can be formalized as maximization of the expected outcome of the game (or the probability of winning for binary-outcome games), which is some function of strategy parameters. The form of that function is unknown to the learner and, for nontrivial games like Go, very complex, which precludes any attempts at solving this problem analytically. Also, the sheer size of the domain of that function (the search space) is immense even for simple representations of the game strategy. Thus, some form of random sampling of the search space becomes inevitable, which explains the popularity of Monte Carlo (MC) techniques in this context (see, e.g., Müller, 2009).

The primary contribution of this paper is a method that assumes the aforementioned behavioral perspective and employs a randomized search *simultaneously* in two different modes: local (intra-strategy) and global (inter-strategy). To conduct the search in the former mode, we employ gradient-based temporal difference learning that works with a single strategy at a time and trains it by a randomized self-play. For the inter-strategy mode, our method relies on coevolutionary learning which maintains a population of strategies, makes them play against each other, and uses the outcomes of games to guide the process of random sampling of the search space in subsequent iterations. The proposed approach, termed Coevolutionary Temporal Difference Learning (CTDL), hybridizes thus two radically different techniques that complement each other in terms of exploration and exploitation of the search space.

This paper is organized as follows. In Section 2, we shortly present the game of Go and its customization adopted for this study. In Section 3, we detail the CTDL approach, starting from describing its constituents: temporal difference learning and coevolutionary learning. Section 5 presents the results of an extensive computational experiment and their analysis. In Sections 6 and 7, we discuss the results and conclude this contribution. Where appropriate, we refer to and review the related work; a comprehensive review of all AI methods applied to computer Go can be found in the work of Bouzy and Cazenave (2001).

2. Game of Go

The game of Go is believed to have originated about 4000 years ago in Central Asia, which makes it one of the oldest known board games. Although the game itself is very difficult to master, its rules are relatively simple and comprehensible. For this reason the famous chess player, Edward Lasker summarized Go in the following way: *The rules of Go are so elegant, organic and rigorously logical that if intelligent life forms exist elsewhere in the universe they almost certainly play Go* (Lasker, 1960).

2.1. Original game rules. Go is played by two players, black and white, typically on a 19×19 board. Players make moves alternately, blacks first, by placing their stones on unoccupied intersections of the grid formed by the board. The player who is to move may pass his/her turn. The game ends if both players pass consecutively.

In a broad sense, the objective of the game is to control more territory than the opponent at the end of the game. This can be achieved by forming connected stone *groups* enclosing as many vacant points and the opponent's stones as possible. A stone group is a set of stones of the same color adjacent to each other; empty intersections adjacent to a group constitute its *liberties*. When a group loses its last liberty, i.e., becomes completely surrounded by the opponent's stones or edges of the board, then it is captured and removed.

A legal move consists in placing a piece on an empty intersection and capturing enemy groups which are left without liberties. Additional restrictions on making moves concern *suicides* and the *ko rule*. A suicide is a potential move that would reduce the number of liberties of the player's own group to zero. Moves leading to suicides are illegal. The ko rule states that a move that recreates a previous board state (i.e., the arrangement of stones on the board) is not allowed either.

The winner is the player who scores more points at the end of the game. The scores are determined using a scoring system agreed upon before the game. The two popular systems include *area counting* (Chinese) and *territory counting* (Japanese). Both ways of calculating the score of a player take into consideration the number of empty intersections surrounded by the player (the player's *territory*). This figure is augmented by the number of the player's stones on the board in the area counting system, or by the number of captured stones (*prisoners*) in the territory counting system. Throughout this study we assume using the Chinese scoring scheme with no *komi* (i.e., points given in advance to one of the players). The reader interested in a more detailed description of Go rules is referred to the book by Bozulich (1992).

2.2. Adopted computer Go rules. There are a few noteworthy issues about the rules of Go that make developing computer players particularly difficult. For this reason, we restrict our research to the following, simplified version of Go.

First of all, the immense cardinality of the state space and the large branching factor mentioned in Introduction render the 19×19 board Go intractable for many algorithms. Fortunately, the game rules are flexible enough to be easily adapted to smaller boards without loss of the underlying 'spirit' of the game, so in a great part of studies on computer Go the board is downgraded to 9×9 or 5×5. Following Lucas and Runarsson (2006) as well as Lubberts and Miikkulainen (2001), we consider playing Go on a 5×5 board (see Fig. 1). Although the small-board version is significantly different from the original, it can the still be used for educational purposes and to demonstrate basic concepts of the game.

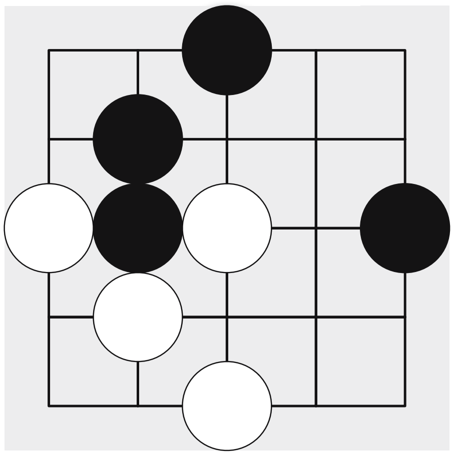


Fig. 1. Small-board version of Go.

A more subtle difficulty in adopting Go rules to computer programs concerns the fact that human players end a game after they agree that they can gain no further advantages. In such situations they use a substantial amount of knowledge to recognize particular intersections as *implicitly controlled.* According to the game rules, if it is impossible to prevent a group from being captured, it is not necessary to capture it explicitly in order to gain its territory. Such a group is considered *dead* and it is removed at the end of the game when both players decide which groups would inevitably be captured. Because determining which stones are dead is nontrivial for computer Go players, we assume that all groups on the board are alive and that capturing is the only way to remove the opponent's group. As a consequence, games are continued until all intersections are *explicitly controlled* and, thus, are much longer than those played by humans.

Finally, in some rule sets (including Chinese rules that we employ), the ko rule is superseded by *super-ko* that forbids repetition of board states across a single game. Recurrently appearing states imply cycling and, theoretically, an infinite game. However, strict implementation of super-ko requires storing all previous board configurations and comparing each of them to the current state. Since most of possible cycles are not longer than 3, we use a reasonable approach in which we remember just two previous board configurations. However, longer cycles can still occur, so to ensure that the game ends, an upper limit of 125 on the total number of moves is additionally imposed. Exceeding this limit results in declaring the game's result as a draw (Lubberts and Miikkulainen, 2001).

3. Methods

All players considered in the remaining part of this paper rely on the Weighted Piece Counter (WPC) to represent their strategies. The WPC is a matrix (unrolled here to a vector for brevity) that assigns a weight w_i to each board intersection i and uses a scalar product to calculate the utility f of a board state $\mathbf{b}$:

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

where s is the board size and b_i is $+1$, -1, or 0 if, respectively, intersection i is occupied by the black player or the white player, or remains empty. This means that we employed the simplest form of the direct coding of the board state termed *Koten* board representation by Mayer (2007), in which exactly one input for each intersection is provided for the evaluation function f. Although such an input signal does not carry information about the states of neighboring intersections, which seems to be essential in Go, the direct board encoding is frequently used in related studies (Runarsson and Lucas, 2005; Schraudolph *et al*., 2001). While the Go board has eight axes of symmetry and the WPC could be simplified to cover just $1/8$ of the board, we do not reveal this fact to the learners and let them learn WPC for the entire board, following the behavioral perspective mentioned in Introduction. 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, the WPC may be viewed as an artificial neural network comprising a single linear neuron with inputs connected to board intersections.

The fact that the WPC weighs the occupancy of each board intersection independently makes it probably the least sophisticated strategy representation for board games. One can also argue whether the WPC is appropriate for the rather weakly-positional game of Go (as compared to, e.g., Othello). Therefore, in objective terms, we do not anticipate the strategies elaborated in the following experiment to beat the top-ranked computer players. However, in the context of this study, this should not be perceived as a hindrance, as our primary goal is to investigate the interplay between coevolutionary learning and TDL in the hybridized approach, and the potential impact it has on the player's performance as measured against the constituent strategies, with a hope that at least some of the conclusions can be generalized to more sophisticated strategy representations.

Still, the WPC's simplicity and its positional character bring substantial advantages, fast board evaluation being the most prominent one. WPC strategies can also be easily interpreted and compared by inspecting the weight values. For instance, Table 1 presents the weight matrix

Table 1. WPC strategy of the heuristic player.

-0.10	0.20	0.15	0.20	-0.10
0.20	0.25	0.25	0.25	0.20
0.10	0.30	0.25	0.30	0.10
0.20	0.25	0.25	0.25	0.20
-0.10	0.20	0.15	0.20	-0.10

of a sample player for 5×5 Go that clearly aims at occupying the center of the board while avoiding the corners.

3.1. Temporal difference learning. Since the influential work of Tesauro (1995) and the success of his *TD-Gammon* player learned through a self-play, *Temporal Difference Learning* (TDL) has become a well-known approach for elaborating game strategies with little or no help from human knowledge or expert strategies given *a priori*. TDL is a method proposed by Sutton (1988), but its origins go back to the famous checkers playing program by Samuel (1959) (although Bucci (2007) suggests that it was rather the first example of coevolution). Impressive results obtained by Temporal difference (TD) methods applied to Reinforcement Learning (RL) problems have triggered off a lot of research including their applications to computer Go (Schraudolph *et al.*, 2001; Lubberts and Miikkulainen, 2001; Lucas and Runarsson, 2006; Mayer, 2007).

The use of RL techniques for learning game strategies stems from modeling a game as a sequential decision problem, where the task of the learner is to maximize the expected reward in the long run (game outcome). The essential feature of this scenario is that the actual (true) reward is not known before the end of the game, so some means are necessary to propagate that information backwards through the series of states, assign credit to particular decisions, and guide intra-game learning.

Though the two most popular RL approaches, Monte Carlo and temporal difference methods (particularly the TD(0) algorithm) represent two extremities in implementing this process (Sutton and Barto, 1998), they share the underlying idea of estimating chances of winning for particular states i.e., finding the state value function) using the sample of experience. In MC-based methods it is indispensable to wait until the end of the game when its exact outcome is known and can be back-propagated to contribute to the predictions made for encountered states. TD(0), on the contrary, looks only one step ahead, analyzes the differential information about the values of both states (the error between temporally successive predictions), and uses it to update the estimate of the current state's value.

TD(λ) is an elegant umbrella that embraces the above special cases of TD(0) and MC (which is equivalent to TD(1)). It makes it possible to smoothly adjust the lookahead 'horizon' by tuning the parameter λ, which refers to the so-called *eligibility trace* and can be interpreted as a *trace decay* factor. Since the acquired knowledge should be generalized across the space of possible states, the function approximation in a *gradient-descent* TD(λ) algorithm is used to predict state values. Technically, the prediction of the game outcome P_t at a certain time step t can be considered a function of two arguments: the current state of the game and the vector of modifiable weights $\mathbf{w}$, which are arbitrary parameters modified in the process of learning. In each step, the weights are updated using the following 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 the gradient $\nabla_w P_t$ is the vector of the 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 sequence of states and the corresponding predictions that appeared in the single game. In the case of TD(0), the weight update is determined only by its effect on the most recent prediction P_t:

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

In this case, P_t takes the form of a board evaluation function f computed as a dot product of the board state vector $\mathbf{b}_t$ and the WPC weights vector $\mathbf{w}$ (see Eqn. (1)). The value returned by f is subsequently squeezed to the interval $[-1, 1]$ using the hyperbolic tangent. This mapping is necessary to obtain the same binary outcome of the game for multiple, linearly independent final states of the board. Eventually, P_t is calculated using the following equation:

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

By applying (4) to the $TD(0)$ update rule (3) and calculating the gradient, we obtain the desired correction of weight w_i at 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 used instead of the prediction P_{t+1}. The outcome is $+1$ if the winner is black, -1 if it white, and 0 when the game ends in a draw.

The process of learning consists in applying the above formula to the WPC vector after each move. The training data for that process, i.e., a collection of games, each of them being a sequence of states $b_1, b_2, \ldots$, is acquired via a self-play, as such a technique does not require anything besides the learning system itself.

Go is a deterministic game and therefore 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. Thanks to random moves, players are confronted with a wide spectrum of possible behaviour of their opponents, including quite unexpected ones, which makes them more robust and versatile.

3.2. Coevolutionary learning. The temporal difference learning approach presented above is a gradient-based local search method that maintains a single model of the learned phenomenon and, as such, has no built-in mechanisms for escaping from local minima. Evolutionary computation, a global search neo-Darwinian methodology of solving learning and optimization problems, has completely opposite characteristics: it lessens the problem of local minima by maintaining a population of candidate solutions (individuals), but has no means for calculating individually adjusted corrections for each solution parameter. Therefore, it seems an attractive complementary alternative for TD for learning game strategies.

However, one faces substantial difficulty when designing a fitness function, an indispensable component of an evolutionary algorithm that drives the search process, for the task of learning game strategies. To properly guide the search, the fitness function should *objectively* assess the utility of the evaluated individual, which, in the case of games, can be done only by playing against *all* possible opponents strategies. For most games such an approach is computationally intractable. Considering instead only a limited sample of opponents lessens the computational burden but biases the search. For this reason, a much more appealing alternative from the viewpoint of game learning is *coevolution*, where the individual's fitness depends on the results of interactions with other individuals from the population. In learning game strategies, such an interaction consists in playing a single game, and its outcome increases the fitness of the winner while decreasing the fitness of the loser. This evaluation scheme is typically referred to as competitive coevolution (Angeline and Pollack, 1993; Azaria and Sipper, 2005).

CoEvolutionary Learning (CEL) of game strategies 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. In practice, this generic scheme is supplemented with various details, some of which relate to evolutionary computation (population size, variation operators, selection scheme, etc.), while others pertain specifically to coevolution (the way the players are confronted, the method of fitness estimation, etc.). CEL embraces a broad class of algorithms that have been successfully applied to many two-person games, including backgammon (Pollack and Blair, 1998), chess (Hauptman and Sipper, 2007), checkers (Fogel, 2002), Othello (Lucas and Runarsson, 2006), NERO (Stanley *et al.*, 2005), blackjack (Caverlee, 2000), PONG (Monroy *et al.*, 2006), and Ant Wars (Jaśkowski *et al.*, 2008a; 2008b). In particular, Runarsson and Lucas (2005) used $(1 + \lambda)$ and $(1, \lambda)$ evolution strategies to learn a strategy for the game of small-board Go.

3.3. Coevolutionary learning with archives. As the set of opponent strategies that an individual faces is limited by the population size, the evaluation scheme used in pure CEL is still only a substitute for the objective fitness function. The advantage of this approach when compared with evolution with a fitness function based on a fixed sample of strategies is that the set of opponents changes with time (from one generation to another), so that individuals belonging to a particular lineage can together face more opponents. In this way, the risk of biasing the search towards an arbitrary direction is expected to be reduced. However, without some extra mechanisms, there is no guarantee that the population will change in the desired direction(s) or change at all. The latter scenario, lack of progress, can occur when, for instance, the player's opponents are not challenging enough or much too difficult to beat. These and other undesirable phenomena, jointly termed *coevolutionary pathologies*, were identified and studied in the past (Watson and Pollack, 2001; Ficici, 2004).

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* (Miconi, 2009).

In this study we use Hall of Fame (HoF, cf. Rosin and Belew, 1997), one of the simplest archive forms. HoF stores all the best-of-generation individuals encountered so far. The individuals in the population, apart from playing against their peers, are also forced to play against randomly selected players from the archive. In this way, an individual's fitness is partially determined by confrontation with past 'champions.' Additionally, we use the archive as a source of genetic material: parent solutions used to breed a new generation come either from the population or from the archive, with equal probability.

Most of the work referred to above involves a single homogenous 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 pointing out that the latest work on coevolution indicates that, even if the game itself is symmetric, it can be beneficial to maintain in parallel two types of strategies: *candidate 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 (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.

3.4. Coevolutionary temporal difference learning. The past results of learning WPC strategies for small-board Go (Runarsson and Lucas, 2005) and Othello (Lucas and Runarsson, 2006) demonstrate that TDL and CEL exhibit complementary features. TDL learns much faster and converges within several hundreds of games, but then it stucks, and, no matter how many games it plays, eventually fails to produce a well-performing strategy. CEL progresses more slowly, but, if properly tuned, outperforms TDL in the long run. 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*. It maintains a population of players and alternately performs TD learning and coevolutionary learning. In the TD phase, each player is subject to a 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.

We reported our first results with CTDL in the work by Szubert *et al.* (2009), where it was applied to learn strategies of the game of Othello. The overall conclusion was positive for CDTL, which produced strategies that, on average, defeated those learned by TDL and CEL. Encouraged by those results, we wondered whether CTDL would prove beneficial also for other purposes, and decided to apply it to the more challenging game of small-board Go. Preliminary results of these efforts were presented in the paper by Krawiec and Szubert (2010), which provides a complete account of this endeavor.

Other hybrids of TDL and CEL were 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 a typical genetic algorithm with mutation as the only variation operator. Singer (2001) showed 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 a 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.

4. Experimental setup

In order to evaluate the idea of hybridizing coevolution with temporal difference learning, several experiments comparing CTDL, CEL, TDL, and their extensions with HoF were conducted. All algorithms were implemented using our coevolutionary algorithms library called cECJ (Szubert, 2010) built upon the Evolutionary Computation in Java (ECJ) framework (Luke, 2010). It was assumed that the uttermost element influencing the time of training is the time required to play a game, so the time consumed by such operations as selection, mutation, evaluation was neglected. In other words, our unit of computational effort is a single game. To provide fair comparison, all runs were stopped when the number of games played reached 2000000. For statistical confidence, each experiment was repeated 25 times.

4.1. Algorithms and setup. For experiments, five methods were prepared, each being a combination of techniques described in the previous section: CEL, TDL and HoF. Wherever possible, parameters taken directly from our previous comparison of the same set of methods (Szubert *et al.*, 2009) were used. Detailed settings follow.

4.1.1. Basic coevolution (CEL). CEL uses a generational coevolutionary algorithm with a population of 50 individuals, each being a 5×5 WPC matrix initialized randomly from the $[-1, 1]$ range. In the evaluation phase, a round-robin tournament is played between all individuals (including self-plays), with wins, draws, and losses rewarded by 3, 1, and 0 points, respectively. For each pair of individuals, two games are played, with players swapping the roles of the black and the white player. The evaluated individuals are subject to tournament selection with tournament size 5, and then, with probability 0.03, their weights undergo Gaussian mutation ($\sigma = 0.25$). Next, individuals mate using one-point crossover, and the resulting offspring form the subsequent generation. As each generation requires 50×50 games, each run lasts for 800 generations to get the total of 2000000 games.

4.1.2. Coevolution with an archive (CEL + HoF). This setup extends the previous one with the HoF archive. Each individual plays games with all 50 individuals from the population 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. After each generation, the individual with the highest fitness joins 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. For this algorithm, 2000000 games translates into 400 generations.

4.1.3. Temporal difference learning (TDL). TDL is an implementation of the gradient-descent temporal difference algorithm TD(λ) described in Section 3.1. The weights are initially set to 0 and the learner is trained solely through a self-play, with random moves occurring with probability 0.1. The learning rate was set to $\alpha = 0.01$, which is a standard value for this method; the value of trace decay λ will be determined in Section 5.

4.1.4. Coevolutionary temporal difference learning (CTDL = TDL + CEL). CTDL combines CEL and TDL as described in Section 3.4, with the CEL phase parameters described in 4.1.1 and the TDL phase parameters described in 4.1.3. It alternates the TDL phase and the CEL phase until the total number of games reaches 2000000. The individuals are initialized randomly like in CEL. Note that TD(λ) is executed for all individuals in the population. 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 8 (default), there are $50 \times 50 + 8 \times 50 = 2900$ games per generation, which leads to 690 generations.

4.1.5. Coevolutionary temporal difference learning with an archive (CTDL + HoF = TDL + CEL + HoF). This setup combines 4.1.2 and 4.1.4 and does not involve any extra parameters.

4.2. Performance measures. It is widely known that monitoring the progress of learning in the interactive domain is hard since, generally (and for Go in particular), there is no precise or easily computable objective performance measure. A fully objective assessment requires playing against all possible opponents, but their sheer number of makes this option impossible. Previous researches used mainly external players as the reference strategies (Silver *et al.*, 2007; Runarsson and Lucas, 2005).

In this study, to monitor the progress, 50 times per run (approximately every 40000 games), we appoint an individual with the highest fitness (i.e., the subjectively best strategy) as the best-of-generation individual and assess its performance (for TDL, the single strategy maintained by the method is the best-of-generation by definition). This individual plays then against two opponents: a predefined, human-designed WPC strategy, and a simple non-WPC strategy. In both cases, the best-of-generation plays 1000 games against the opponent strategy (500 as black and 500 as white), and the resulting probability of winning becomes our estimate of its *absolute* performance. The third performance measure introduced below gauges the *relative* progress of particular methods via a round-robin tournament of representative individuals.

It should be emphasized that the interactions taking place in all assessment methods do not influence the learning individuals.

4.2.1. Performance against the WPC heuristic. This performance measure is the probability of winning with the WPC heuristic, a fixed player encoded as a WPC vector shown in Table 1. This strategy was loosely based on a player found by Runarsson and Lucas (2005).

All WPC-based players are deterministic. Thus, in order to estimate the probability of winning of a given trained player against the WPC heuristic, we forced both players to make random moves with probability $\epsilon = 0.1$; this allowed us to take into account a reacher repertoire of the players' behaviors and make the resulting estimates more continuous and robust. The same technique was applied by Lucas and Runarsson (2006).

4.2.2. Performance against the Liberty Player. To provide another, qualitatively different from WPC benchmark for the developed methods we created a simple game-specific heuristic strategy based on the concept of liberties (c.f. Section 2). This strategy, called here the Liberty Player, focuses on maximizing the number of its own liberties and minimizing the number of the opponent's liberties at the same time. It looks 1-play ahead and evaluates a position by subtracting the number of opponent liberties from the number of its own liberties. Ties are resolved randomly. As with the WPC heuristic, both players are forced to make random moves with probability $\epsilon = 0.1$.

4.2.3. Round-robin tournament between teams of the best individuals. A handcrafted heuristic strategy, even if randomized, cannot be expected to represent in full the richness of possible behavior of Go strategies. In order to get a more realistic performance estimate, we recruit sets of diverse opponents composed of best-of-generation individuals representing particular methods. Technically, each team embraces all best-of-generation strategies found by 25 runs of a particular method. Next, we play a round-robin tournament between the five teams representing particular methods, where each team member plays against all $4 \times 25 = 100$ members from the opponent teams for a total of 200 games (100 as white and 100 as black). The final score of a team is determined as the sum of points obtained by its players in overall 5000 games, using the scoring scheme presented in Section 4.1.1, thus the maximum number of points possible to get by a team in a single tournament is $5000 \times 3 = 15000$.

Let us notice that the round robin tournament offers yet another advantage: there is no need to randomize moves (as was the case when playing against a single external player), since the presence of multiple strategies in the opponent team provides enough behavioral variability.

5. Results

5.1. Finding the best trace decay λ for TD-based methods. In order to assure a fair comparison with other methods, the best value of trace decay λ was first determined by running TDL with various settings of this parameter and testing the resulting strategies using the absolute performance measures introduced in Section 4.2. Technically, because the randomized self-play makes the performance of the TDL learner vary substantially with time, we decided not to rely on the *final* outcome of the method alone. To make the estimates more robust, we sampled each run every 40000 games for the last 800000 games and averaged the performances of strategies (thus, the performance of each run was estimated using 20 individuals).

Table 2 shows the results averaged over 25 runs (this holds for all experiments, unless stated otherwise). For the WPC heuristic, the winning rate is maximized for $\lambda = 0.98$, while for the Liberty Player, this happens for $\lambda = 0.95$. Because the influence of λ for the Liberty Player is much smaller than for the WPC heuristic and the differences for $\lambda \in [0.6, 0.99]$ are very small for the Liberty Player, we chose 0.98 as the optimal value for λ to be used in all further experiments.

5.2. Comparison of coevolutionary temporal difference learning with other methods. In this experiment, our CTDL and CTDL+HoF were compared with their

Table 2. Probability of winning against the WPC heuristic and the Liberty Player for a player found by TD(λ) for different trace decays λ. Means and standard deviations calculated over last 20 sample points of all 25 runs.

λ	Against the WPC heuristic	Against the Liberty Player
0	0.420 ± 0.129	0.496 ± 0.116
0.2	0.444 ± 0.123	0.532 ± 0.102
0.4	0.465 ± 0.125	0.547 ± 0.104
0.6	0.483 ± 0.130	0.559 ± 0.102
0.8	0.497 ± 0.134	0.560 ± 0.098
0.9	0.543 ± 0.131	0.564 ± 0.093
0.95	0.599 ± 0.140	$\mathbf{0.567} \pm 0.089$
0.96	0.597 ± 0.143	0.557 ± 0.089
0.97	0.613 ± 0.133	0.563 ± 0.086
0.98	$\mathbf{0.630} \pm 0.148$	0.557 ± 0.084
0.99	0.617 ± 0.157	0.554 ± 0.094
1.0	0.545 ± 0.195	0.548 ± 0.109

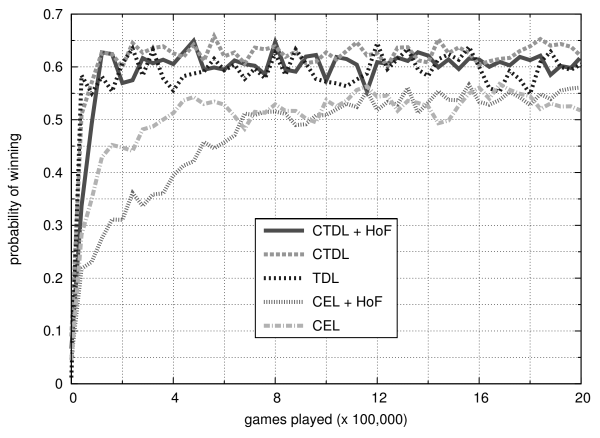
constituent methods: CEL, CEL+HoF and TDL. Figure 2 shows the progress of the methods measured by the performance against the fixed external players: the WPC heuristic and the Liberty Player. It can be observed that the relative courses of the methods' performance are similar for both plots. Both measures agree that, in the long run, pure coevolution is worst, producing players that win only about 50% of games. Moreover, CEL learns much slower than TDL-based methods. Adding the Hall of Fame archive to CEL makes it learn even slower, and, surprisingly, does not lead to better results.

As expected, the quality of individuals produced by the TDL-based algorithms in early stages of the run is higher than that of those produced by methods that do not involve TDL. In particular, CTDL or CTDL+HoF look superior, as they quickly achieve good performance and are best in the long run. Interestingly, pure TDL seems as good as other TDL-based methods when playing with the WPC heuristic, but it is significantly worse than CTDL or CTDL+HoF when crossing swords with the Liberty Player. The best of CTDL and CTDL+HoF players attained around a winning rate of 65% with both players.

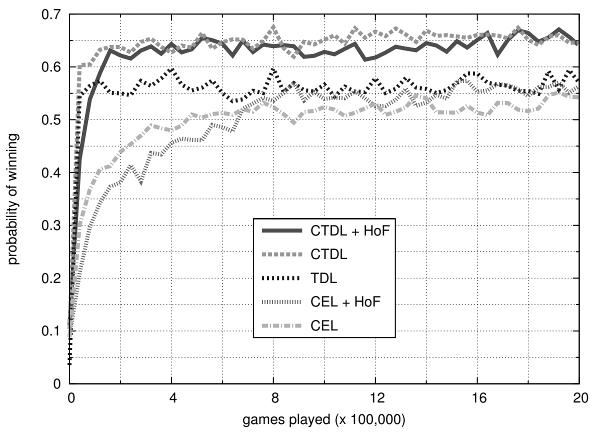
Though the performance of all methods in absolute terms is rather moderate, this should be attributed, in the first place, to the simplicity of WPC representation, which is not suited for the non-positional game of Go. Note also that the performance of the optimal[1] WPC-represented Go strategy is unknown, so judging the above probabilities as objectively good or bad would be inconsiderate.

The comparison with external players demonstrates that the fusion of coevolution with local search can be beneficial. However, the results presented in Fig. 2 do not allow stating whether there is any advantage of adding

[1]The 'optimality' may be defined in many ways, but here the maximal expected utility solution is a reasonable choice (c.f. Ficici, 2004).



(a)



(b)

Fig. 2. Comparison of learning methods. Average performance of the best-of-generation individuals measured as the probability of winning against the WPC heuristic (a) and the Liberty Player (b).

the coevolutionary archive technique to CTDL, but the comparison with just two external strategies may not be enough. In order to gain additional insight in the course of learning of the methods, every 40000 games we run the round-robin tournaments between best-of-generation representatives of each run, as described in Section 4.2.3. The points earned in the tournaments are plotted in Fig. 3.

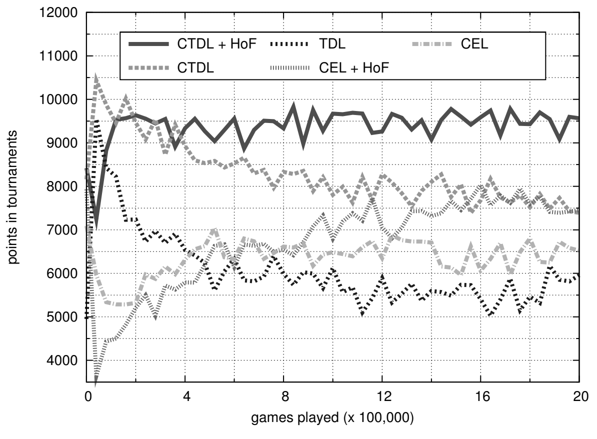


Fig. 3. Relative comparison of learning methods. Points obtained by teams of the best-of-generation players in the round-robin tournaments that were played 50 times during the run. The maximum possible number of points to obtain by a team in a single tournament was 15000.

This time, CTDL alone is also good in comparison to other methods, but CTDL armed with the Hall of Fame archive performs clearly better, and its superiority over any other method is undeniable. The TDL compound of CTLD+HoF makes it very fast, and though it learns slower than TDL or CTDL, it needs only about 100000 games to outperform TDL and another 300000 to gain advantage over CTDL.

Not all our previous conclusions were confirmed in the relative performance assessment. Most notably, though TDL was found to be clearly better than CEL when gauged using the absolute performance measures, it is now the worst method in the long run, worse even than basic coevolution. It may be also observed that adding the archive to CEL is still profitable.

Generally, all three performance measures used in this study are just estimations of the true performance. Computing the value of this function is computationally infeasible for even such a small game as 5×5 Go, since it requires playing with all possible strategies. Despite the conceptual advantages of using the round-robin tournament as a performance measure (c.f. Section 4.2.3), it is hard to absolutely state which estimate of the true performance measure is the best one. Therefore, we do not claim any statistical difference between CEL and TDL. On the other hand, in the light of all three evaluation measures, CTDL+HoF is superior to other methods.

Table 3 presents detailed results of a round-robin tournament of teams the best-of-run individuals, that is, the best-of-generation individuals found in the last generation, after 2000000 games. We interpret the pairwise match between individuals from different teams as the result of direct comparison between two methods; thus obtaining more points in a direct match means that one method is more likely to be better than the other one. Using this interpretation, the round-robin tournament orders the methods linearly: CTDL+HoF $>$ CEL+HoF $>$ CTDL $>$ TDL $>$ CEL, and there are no cycles between them. Note that, although CEL+HoF got more points than CTDL in total, CTDL was slightly better in the direct match (49.2% to 48.8%).

We also analyzed in detail the results of the round-robin tournament and determined the best individual strategy evolved in CTDL+HoF, i.e., the one that obtained the highest number of points when playing with other strate-

Table 3. Results of the round-robin tournament for the teams of individuals from the last generations. Each number is the percentage of points obtained in the tournament; the values may not total 100% since there were 3 points for win and 1 for draw.

	CTDL+HoF	CTDL	CEL+HoF	CEL	TDL	Total
CTDL+HoF	–	64.3%	60.4%	64.3%	66.0%	63.8%
CTDL	34.3%	–	49.2%	56.5%	57.0%	49.3%
CEL+HoF	37.9%	48.8%	–	54.4%	57.5%	49.7%
CEL	34.5%	42.5%	44.0%	–	53.0%	43.5%
TDL	31.5%	41.0%	40.7%	45.5%	–	39.7%

Table 4. Weighted piece counter vector of the best player evolved by CTDL+HoF.

0.46	−0.05	0.66	1.15	−1.42
0.4	1.54	2.29	1.06	1.67
0.11	1.16	1.44	0.85	0.02
2.02	0.69	1.39	0.54	0.89
−0.51	1.02	−0.22	0.66	−0.22

gies during the tournament. This strategy, presented in Table 4, achieved 478 points out of 600 possible (79.7%). Although the center of the board is generally preferred to corners, it is surprising that this strategy exhibits no clear axes of symmetry.

5.2.1. Determining the best TDL–CEL ratio. The number of TDL games per each evolutionary generation (the TDL–CEL ratio) seems to be potentially an important parameter of CTDL and CTLD+HoF methods (we used eight TDL games per generation in the experiments reported hitherto). We investigated this issue by running our best algorithm, CTDL+HoF, for different TDL–CEL ratios. The probability of the best-of-generation individual winning against the external players for different TDL–CEL ratios is presented in Fig. 5, whereas the relative performance measured by the round-robin tournament is given in Fig. 4. Notice that '0 games' is equivalent to CEL+HoF. Apart from this extreme setting, the plots do not reveal any substantial differences as far as the final performance is concerned. Despite the fact that having more TDL games (see '8 games' and '16 games' in Fig. 4) speeds up the learning, the difference, initially substantial, becomes rather negligible after several hundreds of thousands of training games. Based on these results, we conclude that CTDL+HoF is moderately sensitive to the TDL–CEL ratio and recommend values greater than 8 for this parameter, which confirms our earlier findings for Othello (Szubert *et al.*, 2009).

5.2.2. Changes observed in genotypic traits. The aggregate results of multiple runs let us draw sound conclusions about the superiority of some approaches to others, but say little about the actual dynamics of the learning process. Figure 6 presents genotypes of best-of-generation individuals taken every 40000 games from a single CTDL+HoF run. The sequence of genotypes starts in the top-left corner of the figure and should be read row-wise. Colors correspond to WPC weights (white = −1, black = 1).

Despite the fact that CTDL+HoF does not seem to qualitatively improve against our two external players after 200000 training games (c.f. Fig. 2), the explorative forces of coevolution apparently continue to substantially change the genotype. It is striking how qualitatively different the best genotypes discovered by evolution in particular stages are. It is also surprising that the central symmetry, which was found beneficial in various stages of evolution, was eventually abandoned in favor of traces of axial symmetry or no clear symmetry at all. Although the final WPC bears some resemblance to our heuristic WPC player presented in Table 1, the asymmetry makes its genotype much more different than intuitively presumed.

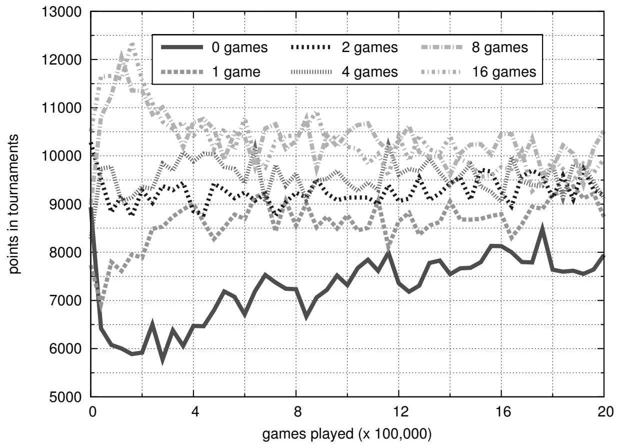


Fig. 4. Relative comparison of CTDL+HoF using different TDL–CEL ratios. Points obtained by teams of the best-of-generation players in the round-robin tournaments that were played 50 times during the run (approximately every 40000 games).

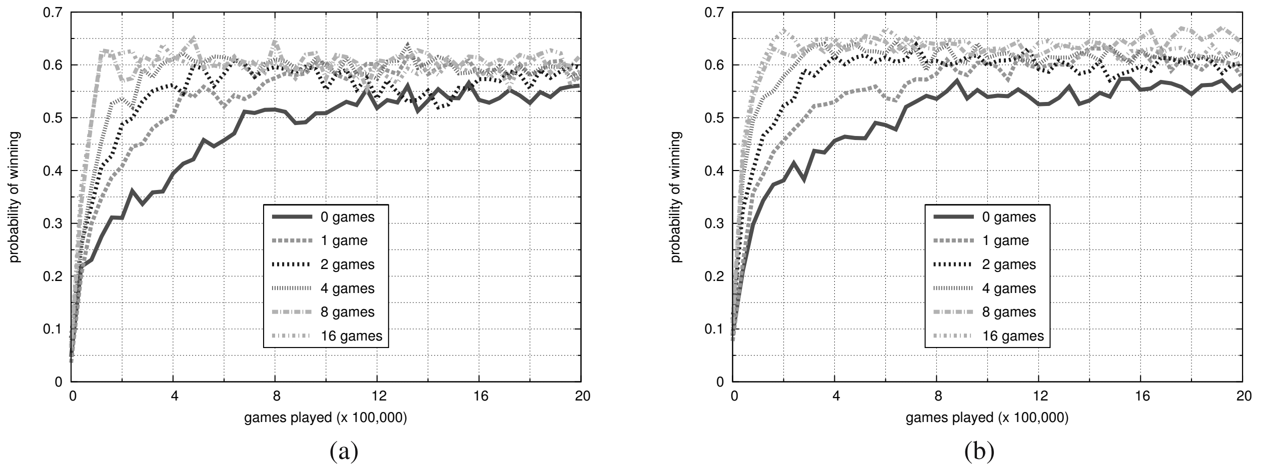


Fig. 5. Comparison of CTDL+HoF using different TDL–CEL ratios. Plots present average performance of the best-of-generation individuals measured as the probability of winning against the WPC heuristic (a), the Liberty Player (b).

6. Discussion

An in-depth conceptual analysis of the interplay between intra-game learning (temporal difference) and inter-game learning (coevolution) in CTDL reveals that it is more intricate and sophisticated than it may appear at a first glance.

Though TDL and CEL employ the WPC in exactly the same way when playing a game, they attach a fundamentally different interpretation to WPC weights during learning. TDL attempts to modify the weights so as to faithfully model the true value function describing winning chances for each game state. It does so because its training formula (3), when applied to the terminal state, substitutes the actual game outcome ($+1$, 0, or -1) in place of the learner's estimate P_{t+1}. CEL, on the contrary, does not refer to any such absolute values, so only the *ordering* of state values is relevant for it. In effect, TDL is trying to solve a bit different, more constrained (and thus presumably more difficult) problem. Technically, for each local minimum $\mathbf{w}_{\min}$ of the error function that the gradient-descent TDL aims at (including the global minima), there are infinitely many other WPCs that produce identical behavior of the player against any strategy.[2] Each of such strategies is equally desirable from the viewpoint of CEL, but many of them would be considered completely worthless by TDL (e.g., because of overestimating the true state values).

With TDL and CEL guiding the search process in different directions, their efforts can happen to cancel each other and render CTDL ineffective. However, the experimental results clearly demonstrate that this is not the case, most probably because, in a highly dimensional search space (25 elements of the WPC), it is very unlikely for TDL and CEL to adopt strictly opposite search directions. TDL, at least on average, benefits from disturbances introduced by CEL, which force it to consider solutions that it would not come upon otherwise, and probably helps it escape local minima. This is parallel to observations in the field of optimization, where the search algorithm can benefit from being allowed to consider infeasible solutions that do not meet some of the assumed constraints, as this opens new 'shortcuts' in the search space (Michalewicz, 1996).

It might also be the case that the loss of performance resulting from the incompatibility discussed here is compensated by synergy of other features of both methods. For that instance, TDL, as a gradient-based technique, is able to link the outcome P of its actions *independently* to each parameter of the solution (element of the WPC matrix) and calculate the desired correction vector (cf. Eq. 3). It is also capable to simultaneously update all strategy parameters (weights). With this skill, it complements evolution, which is devoid of such an ability.

7. Conclusion

There are at least two general lessons that can be learned from this study. Firstly, we can conclude that different modes of adopting the Monte Carlo methodology in a learning algorithm may lead to fundamentally different dynamics of the learning process and final outcomes. Secondly, using qualitatively different modes of randomization can be synergetic, leading to substantially better performance when compared with the constituent methods (and randomization modes).

Although our evolved WPC players would most probably yield to other contemporary strategies that use more sophisticated representations, we need to emphasize that our primary objective was to hybridize two al-

[2]Such strategies can be generated by, e.g., scaling all elements of $\mathbf{w}_{\min}$ by the same factor.

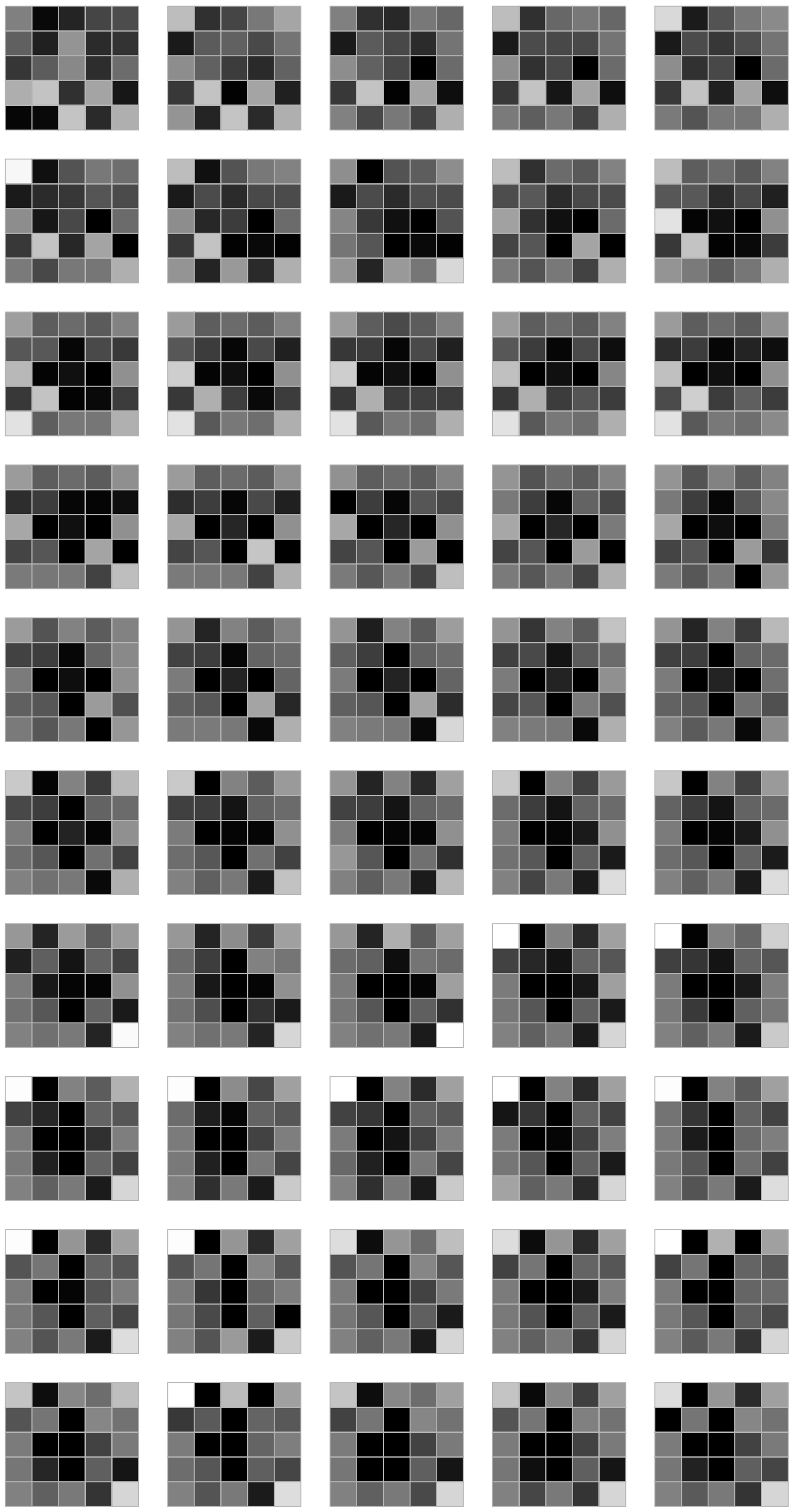


Fig. 6. Genotypic changes observed in the best-of-generation individuals of an example CTDL+HoF run sampled every 40000 games (the first generation in the upper-left corner, arranged row-wise). Individuals are illustrated as 5×5 Go boards colored accordingly to the corresponding WPC weights (white = -1, black = 1). WPCs were scaled using the maximum absolute weight.

gorithms that learn fully autonomously and study the *relative* gains that result from their synergy. To quote Arthur Lee Samuel's declaration, *The temptation to improve the machine's game by giving it standard openings or other man-generated knowledge of playing techniques has been consistently resisted* (Samuel, 1959, p. 215).

This result confirms our former observations (Szubert *et al*., 2009), when we demonstrated that hybridizing coevolution with TD(0) proves beneficial when learning the strategy of the game of Othello. Here, we come to similar conclusions for the game of small-board Go, and additionally note that extending the lookahead horizon by using TD(λ) with λ close to 1 can boost the performance of CTDL even further. Adding a simple coevolutionary archive to this mixture makes it even better. Thus, there is growing evidence to support our claim that hybridizing coevolution with temporal difference learning can be beneficial.

In the context of games, this result is still preliminary due to the small board size, and claiming that CTDL scales well with the board size would be premature. Evolving an effective strategy for larger boards is much more challenging, in particular for the game of Go, where absolute positions are of minor importance compared to the 'topology' of the board state. Approaching such a problem using the WPC for strategy representation would not make sense. Note, however, that CTDL is a generic coevolutionary metaheuristic: none of its constituent methods is aware of strategy encoding, as they simply process vectors of numbers (strategy parameters). Thus, CTDL can be applied to other strategy representations that can be encoded by vectors, including representations that already proved successful on selected games (like *n*-tuples). We look forward to investigate such scenarios, presuming that for problems and representations on which TDL and CEL alone perform well, CTDL could act in a synergetic way.

Apart from being encouraging from the practical viewpoint, CTDL seems to rise also interesting theoretical issues that deserve further research. One of them pertains to the way CTDL performs the local search. In essence, by relying on a randomly perturbed self-play, TD serves only as a *substitute* for local search, as it has no access to the objective fitness function. Nevertheless, it positively contributes to our hybrid. Thus, it 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 for local search. By analogy to the terms *memetic algorithms* and *Lamarckian evolution* that are usually used to refer to various hybrids of evolution and local search, which typically alternate genetic search for the population and local search for individual solutions, this paradigm can be termed *coevolutionary memetic algorithm* or *Lamarckian coevolution*. 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.

Acknowledgment

This work has been supported by the Polish Ministry of Science and Higher Education under the grant no. N N519 441939.

References

Angeline, P.J. and Pollack, J.B. (1993). Competitive environments evolve better solutions for complex tasks, *Proceedings of the 5th International Conference on Genetic Algorithms, Urbana-Champaign, IL, USA*, Vol. 270, pp. 264–270.

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

Bouzy, B. and Cazenave, T. (2001). Computer Go: An AI oriented survey, *Artificial Intelligence* **132**(1): 39–103.

Bozulich, R. (1992). *The Go Player's Almanac*, Ishi Press, Tokyo.

Bucci, A. (2007). *Emergent Geometric Organization and Informative Dimensions in Coevolutionary Algorithms*, Ph.D. thesis, Brandeis University, Waltham, MA.

Caverlee, J.B. (2000). A genetic algorithm approach to discovering an optimal blackjack strategy, *Genetic Algorithms and Genetic Programming at Stanford*, Stanford Bookstore, Stanford, CA, pp. 70–79.

de Jong, E.D. (2005). The MaxSolve algorithm for coevolution, *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, GECCO 2005, Washington, DC, USA*, pp. 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, Brandeis University, Waltham, MA.

Ficici, S. and Pollack, J. (2003). A game-theoretic memory mechanism for coevolution, *Proceedings of the 2003 International Conference on Genetic and Evolutionary Computation, GECCO'03, Chicago, IL, USA*, pp. 286–297.

Fogel, D.B. (2002). *Blondie24: Playing at the Edge of AI*, Morgan Kaufmann Publishers, San Francisco, CA.

Hauptman, A. and Sipper, M. (2007). Evolution of an efficient search algorithm for the mate-in-n problem in chess, *Proceedings of the 10th European Conference on Genetic Programming, EuroGP'07, Valencia, Spain*, pp. 78–89.

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

Jaśkowski, W., Krawiec, K. and Wieloch, B. (2008b). Winning Ant Wars: Evolving a human-competitive game strategy using fitnessless selection, *11th European Conference on Genetic Programming, EuroGP 2008, Naples, Italy*, pp. 13–24.

Johnson, G. (1997). To test a powerful computer, play an ancient game, *The New York Times*, July 29.

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, CIG 2007, Honolulu, HI, USA,* pp. 203–209.

Krawiec, K. and Szubert, M. (2010). Coevolutionary temporal difference learning for small-board Go, *IEEE Congress on Evolutionary Computation, Barcelona, Spain*, pp. 1–8.

Lasker, E. (1960). *Go and Go-Moku: The Oriental Board Games*, Dover Publications, New York, NY.

Lubberts, A. and Miikkulainen, R. (2001). Co-evolving a Go-playing neural network, *Coevolution: Turning Adaptive Algorithms Upon Themselves, Birds-of-a-Feather Workshop, Genetic and Evolutionary Computation Conference, GECCO 2001, San Francisco, CA, USA*, pp. 14–19.

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, CIG 2006, Reno/Lake Tahoe, NV, USA*, pp. 52–59.

Luke, S. (1998). Genetic programming produced competitive soccer softbot teams for RoboCup97, *Genetic Programming 1998: Proceedings of the 3rd Annual Conference, Madison, WI, USA*, pp. 214–222.

Luke, S. (2010). ECJ 20—A Java-based Evolutionary Computation Research System, `http://cs.gmu.edu/~eclab/projects/ecj/`.

Luke, S. and Wiegand, R. (2002). When coevolutionary algorithms exhibit evolutionary dynamics, *Workshop on Understanding Coevolution: Theory and Analysis of Coevolutionary Algorithms (at GECCO 2002), New York, NY, USA*, pp. 236–241.

Mayer, H.A. (2007). Board representations for neural Go players learning by temporal difference, *IEEE Symposium on Computational Intelligence and Games, CIG 2007, Honolulu, HI, USA*, pp. 183–188.

Mechner, D.A. (1998). All systems Go, *The Sciences* **38**(1): 32–37.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, London.

Miconi, T. (2009). Why coevolution doesn't "work": Superiority and progress in coevolution, *Proceedings of the 12th European Conference on Genetic Programming, EuroGP'09, Tübingen, Germany*, pp. 49–60.

Monroy, G.A., Stanley, K.O. and Miikkulainen, R. (2006). Coevolution of neural networks using a layered Pareto archive, *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO 2006, Seattle, WA, USA*, pp. 329–336.

Müller, M. (2009). Fuego at the Computer Olympiad in Pamplona 2009: A tournament report, *Technical report*, University of Alberta, Alberta.

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. (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): 210–229.

Schraudolph, N.N., Dayan, P. and Sejnowski, T.J. (2001). Learning to evaluate Go positions via temporal difference methods, *in* N. Baba and L.C. Jain (Eds.) *Computational Intelligence in Games*, Studies in Fuzziness and Soft Computing, Vol. 62, Springer-Verlag, Berlin, Chapter 4, pp. 77–98.

Silver, D., Sutton, R. and Müller, M. (2007). Reinforcement learning of local shape in the game of Go, *Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India*, pp. 1053–1058.

Singer, J.A. (2001). Co-evolving a neural-net evaluation function for Othello by combining genetic algorithms and reinforcement learning, *International Conference on Computational Science, San Francisco, CA, USA*, pp. 377–389.

Stanley, K., 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**(1): 9–44.

Sutton, R.S. and Barto, A.G. (1998). *Reinforcement Learning: An Introduction*, The MIT Press, Cambridge, MA.

Szubert, M. (2010). cECJ—Coevolutionary Computation in Java, `http://www.cs.put.poznan.pl/mszubert/projects/cecj.html`.

Szubert, M., Jaśkowski, W. and Krawiec, K. (2009). Coevolutionary temporal difference learning for Othello, *IEEE Symposium on Computational Intelligence and Games, CIG 2009, Milan, Italy*, pp. 104–111 .

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

Watson, R.A. and Pollack, J.B. (2001). Coevolutionary dynamics in a minimal substrate, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2001, San Francisco, CA, USA*, pp. 702–709.

Krzysztof Krawiec (Ph.D.: 2000, D.Sc.: 2005) is an associate professor at the Poznań University of Technology, Poland, working mainly on topics related to evolutionary computation and pattern recognition. His recent work includes evolutionary computation for machine learning, primarily for learning game strategies and for evolutionary synthesis of pattern recognition systems; the role of program semantics in genetic programming, particularly in problem decomposition and operator design; coevolutionary algorithms, mainly extraction of coordinate systems from test-based problems. More details are available at `www.cs.put.poznan.pl/kkrawiec`.

Wojciech Jaśkowski is a Ph.D. student and an assistant researcher at the Laboratory of Intelligent Decision Support Systems, Institute of Computing Science, Poznań University of Technology, Poland. He is an author of more than 20 publications in computational intelligence. His main research addresses co-optimization, coevolution, genetic programming, and learning strategies for interactive domains and games.

Marcin Szubert received the M.Sc. degree in computer science from the Poznań University of Technology in 2009. Since then he has been pursuing his Ph.D. in the Laboratory of Intelligent Decision Support Systems at the Institute of Computing Science of the same university. His research interests primarily cover the area of computational intelligence and machine learning with applications to game playing. Particularly, he works on hybridizing reinforcement learning methods and coevolutionary algorithms for developing artificial neural networks.

Received: 28 September 2010
Revised: 18 January 2011