Coevolutionary Temporal Difference Learning for Small-Board Go

Krzysztof Krawiec, *Member, IEEE* and Marcin Szubert

***Abstract*—In this paper we apply Coevolutionary Temporal Difference Learning (CTDL), a hybrid of coevolutionary search and reinforcement learning proposed in our former study, to evolve strategies for playing the game of Go on small boards (5×5). CTDL works by interlacing exploration of the search space provided by one-population competitive coevolution and exploitation by means of temporal difference learning. Despite using simple representation of strategies (weighted piece counter), CTDL proves able to evolve players that defeat solutions found by its constituent methods. The results of the conducted experiments indicate that our algorithm turns out to be superior to pure coevolution and pure temporal difference learning, both in terms of performance of the elaborated strategies and the computational cost. This demonstrates the existence of synergistic interplay between components of CTDL, which we also briefly discuss in this study.**

I. Introduction

Local and global search techniques feature complementarily appealing characteristics. The former ones investigate the search space in immediate proximity of the current solution and use the gathered information to determine the search direction that seems to be most promising for the nearest feature. The latter ones, particularly evolutionary algorithms that are of interest here, explore in parallel distant areas of the search space and typically do not care much about the local structure of the fitness landscape. Thus, these approaches assume radically different perspectives on exploration and exploitation of the search space, and both can turn out useful depending on the characteristic of the problem and the stage of the search process.

It is not surprising then, that hybridization of different varieties of evolutionary algorithms with local search has been subject to intense research in past, resulting in such approaches like memetic algorithms or genetic local search [1]. However, there are only a few results on hybridizing local search with coevolution [2], [3]. Because there are no obvious reasons for which coevolution could not benefit from local search too, in our research we intend to fill that gap. In [4] we investigated the interplay between temporal difference learning (TDL), a canonical reinforcement learning method typically trained by some form of gradient descent, and simple competitive coevolution, in which individuals compete with each other and propagate their features using the principles of simulated evolution. This fusion, termed Coevolutionary Temporal Difference Learning (CTDL), proved beneficial for learning to play Othello. Here, we conduct an analogous investigation for the game of small-board Go.

K. Krawiec and M. Szubert are with the Institute of Computing Science, Poznan University of Technology, Piotrowo 2, 60965 Poznań, Poland; email: {kkrawiec,mszubert}@cs.put.poznan.pl.

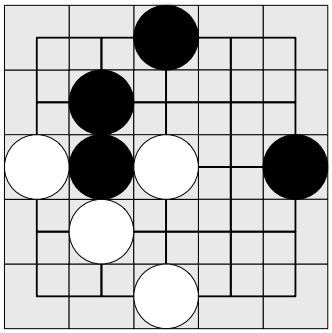

Fig 1. The game of small-board Go

This paper is organized as follows. In Section II we describe the rules of the game, the way in which they are simplified to study algorithms that autonomously learn to play Go, and a brief history of research in this area. Section III introduces the representation of strategy adopted in this study, reviews key concepts of temporal difference learning and coevolutionary learning, and introduces CTDL. In Section IV we describe the experiment and discuss its results, finally concluding in Section V.

II. The 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* [5].

A. Original Game Rules

Go is played by two players, black and white, typically on an 19×19 board, though the rules can be easily applied to any board size. Figure 1 shows the board state of an examplary 5×5 Go game. Players make moves alternately, blacks first, by placing their stones on unoccupied intersections of the grid formed by the board. At any time the player who is about to move may pass his turn. The game ends if both players pass consecutively.

The objective of the game is, roughly speaking, to control more territory than the opponent at the end of the game. This can be achieved by positioning stones to form connected groups enclosing as many vacant points and opponent's stones as possible. A stone *group* is a set of same-colour stones adjacent to each other; empty intersections adjacent to

a group are called its *liberties*. If a group has no liberties, *i.e.*, it is completely surrounded by opponent's stones or edges of the board, then it is captured and removed.

A legal move consists of 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 so-called *ko rule*. A suicide is a potential move that would deprive player's own group of liberties. Moves leading to suicides are illegal. Ko rule, in turn, forbids moves that recreate a previous board state (*i.e.,* arrangement of stones on the board).

The winner is the player who at the end scores more points calculated according to agreed scoring system. There are two most popular systems known as *area counting* (Chinese) and *territory counting* (Japanese). Both systems take into consideration the number of empty intersections surrounded by a player (player's *territory*). In area counting, the number of stones a player has on the board is added to this value, whereas in territory counting the number of captured stones (*prisoners*) is added instead. The reader interested in more detailed description of Go rules is referred to [6].

B. Simplified Game Rules

There are a few noteworthy issues about the original rules of Go that make developing computer players difficult. First of all, human players stop 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.* Computer Go players based on concrete evaluation functions usually continue playing the game until all intersections are *explicitly controlled.* As a result, games played by computers can be much longer than those played by humans.

In some rule sets (including Chinese rules), ko rule appears in form of *super-ko*, which prohibits repetition of a board state during a single game. Recurrently appearing states may lead to cycles and, theoretically, an infinite game. However, a literal implementation of this rule would require storing all previous board configurations and comparing them to the current state. Since most of possible state cycles are not longer than 3, a reasonable approach is to remember just two previous board configurations. To ensure that the game ends, an additional upper limit on the total number of moves can be placed. Exceeding this limit results in declaring game's result as a draw [7].

Finally, according to the original 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 as *dead* and is removed at the end of the game, when both players agree which groups would inevitably be captured. Because determining which stones are dead is not a trivial problem, it is much easier to assume that all groups on the board are alive. Thus, explicit capture is always required to remove opponent's group.

In this study we decided to use the above simplifications and the Japanese scoring system.

C. Previous Research on Computer Go

The game of Go has been a subject of computational intelligence research for more than 40 years and it is increasingly recognized as a great challenge because best computer players can still be beaten by professional human players in full-board games [8], [9]. This is a result of a huge combinatorial complexity of this game which is much higher than for other popular two-player deterministic board games – its state space cardinality is around 10^{170} and the game tree has an average branching factor of around 200. For this reason a lot of research on computer Go focuses on versions with smaller boards, like 9×9 or even 5×5. The rules of the game are flexible enough to be easily adapted to such boards.

Conventional Go-playing programs are precisely tuned expert systems based on a thorough human analysis of the game. In such programs, knowledge of professional Go players formulated as a multitude of rules and guidelines is applied to game-playing algorithm 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 possibility of its articulation in the source code of the playing program.

An appealing alternative to using hand-coded expert rules is an approach based on Monte Carlo techniques [10], which by contrast, requires very little domain knowledge. This method chooses a move on the basis of statistics collected during thousands of pseudorandom playouts starting from the current board state. One of the strongest Go programs using this approach is *Fuego* [11], 2009 Computer Olympiad champion in 9×9 Go [12], which has recently defeated 9-dan professional player on the same board size. Another major program, *Many Faces of Go* [13], champion from 2008, uses an interesting combination of Monte Carlo Tree Search with an older knowledge-based approach.

Nevertheless, 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 and compared in the context of small-board Go by Runarsson and Lucas [14]. There are more examples of using self-learning approaches for Go including [7] and [15]. A comprehensive review of all AI methods applied to computer Go can be found in [16].

III. Methods

A. Strategy Representation

Representation of strategy is the primary determinant of behavior and quality of playing of a computer-based player. However, in this study we are mainly interested in analyzing the *relative* improvements that the hybridized CTDL method can bring when compared to its constituents, so the absolute player's performance is of secondary importance. Therefore, we employ here probably the least sophisticated strategy representation for board games — weighted piece counter (WPC). WPC assesses the utility of a particular board state

TABLE I
A STRATEGY REPRESENTED BY WPC

-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

by *independently* considering the occupancy of all board intersections. Technically, WPC is a matrix that assigns a weight w_i to each board intersection i and uses 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 board's size and b_i is +1, -1, or 0 if, respectively, intersection i belongs to the black player, the white player, or is empty. In the course of the game, player's WPC is used as a board evaluation function within 1-ply minimax search. 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 intersections.

The main advantage of WPC is its simplicity resulting in a very fast board evaluation. Moreover, WPC strategies can be often easily interpreted and compared just by inspecting the weight values. Table I presents the weight matrix of an exemplary player for 5×5 Go that clearly focuses at taking possession of the middle of the board while avoiding corners.

B. Coevolutionary Learning

Coevolutionary algorithms are variants of evolutionary computation where individual's fitness depends on other individuals. Evaluation of an individual takes place in the context of at least one other individual, and may be of cooperative or competitive nature. In the former case, individuals share the fitness they have jointly elaborated, whereas in the latter one, a gain for one individual means a loss for the other. Past research has shown that this scheme may be beneficial for some types of tasks, allowing task decomposition (in the cooperative variant) or solving tasks for which the objective fitness function is not known *a priori* or is hard to compute, with games being the most representatives examples of such problems [17], [18].

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 lacks many details that need to be filled in, some of which relate to evolutionary computation (population size, variation operators, selection scheme, etc.), while some 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 [19] used one of the simplest evolutionary algorithm, a random hill-climber to successfully address the problem of learning backgammon strategy. Runarsson and Lucas [14] used (1 + λ) and (1, λ) Evolution Strategies to learn a strategy for the game of small-board Go. 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. Moreover, self-adapting mutation strength was also employed, but for larger populations its effects were unnoticeable.

Various forms of CEL have been successfully applied to many two-person games, including Backgammon [20], Chess [21], Checkers [22], Othello [23], NERO [24], Blackjack [25], Pong [26], and AntWars [27], [28].

C. Coevolutionary Archives

The central characteristic of CEL is that it refrains from using 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. This and other undesirable phenomena, jointly termed *coevolutionary pathologies*, have been identified and studied in past [29].

In order to deal with coevolutionary pathologies, *coevolutionary archives* were introduced that try to sustain progress. 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* [30].

In this study we use Hall of Fame (HoF, [31]), one of the simplest archives. HoF stores all the best-of-generation individuals encountered so far. The population members, 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 studies quoted above involve a single population of players, a setup called *one-population coevolution* [32] or *competitive fitness environment* [17], [33]. Although such design seems most natural for games, recent work on coevolution indicates that, even if the game 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 and yielding to others. It has been demonstrated [34], [35] 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. Here, however, we remain within the one-population framework to keep our setup as simple as possible and limit the number of factors that could interfere with hybridization of CEL and TDL.

D. Temporal Difference Learning

Temporal Difference (TD), a method proposed by Sutton in 1988 [36], has become a popular approach for solving reinforcement learning tasks. Some suggest [37] that the famous chess playing program by Samuel [38] in 1959 was in fact taught by a simple version of temporal difference learning (however others [39] 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 [40]. This influential work has triggered off a lot of research in reinforcement learning and TD methods, including their applications to Go [7], [15], [41].

The $TD(\lambda)$ procedures solve prediction learning problems that consist in estimating the future behavior of an incompletely known system from the past experience. TD learning occurs whenever systems 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 α 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 game-playing 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:

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

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. During game play, moves are selected on the basis of the most recent evaluation function.

Go is deterministic, thus the course and the outcome of a 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, it is typical in TDL to force a random move in each turn, with certain probability. After such a random move, no weight update occurs.

E. Coevolutionary Temporal Difference Learning

The past results of learning WPC strategies for small-board Go [14] and Othello [23] demonstrate that TDL and CEL exhibit complementary features. TDL typically learns much faster and converges within several hundreds of games, but then stucks and fails to produce a well-performing strategy, no matter how many games it plays. CEL progresses slower, but, if properly tuned, eventually outperforms TDL. Therefore, it sounds reasonable to combine these approaches into a hybrid algorithm that would possibly exploit the advantages of both methods.

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 bred 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.* [42] 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. Recent work by Manning [43] demonstrate a bounded-size Nash Memory archive for coevolution [44] and employed TDL as a weight mutation

operator. In [2], Singer has shown that reinforcement learning can be superior to random mutation as an exploration mechanism. His Othello-playing strategies were three-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 and network weights were modified after every move using back-propagation algorithm. The evolutionary phase consisted of a round-robin tournament that determined players' fitnesses, followed by crossover at feature-level and mutation. The experiment yielded a strategy that was 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 [31], all implemented using *Evolutionary Computation in Java* (ECJ) library [45]. To provide fair comparison, all runs used the same settings and stopped when the number of games played reached 2 millions. For statistical significance, each experiment was repeated 25 times. Wherever it was possible we used parameters taken directly from our previous comparison of the same set of methods [4]. The only exception was so called TDL-CEL ratio (see section IV-A4) which we increased to 10.

A. Algorithms and setup

1) TDL: TDL is an implementation of a gradient-descent temporal difference algorithm $TD(0)$ described in Section III-D. 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 randomly. 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 800 generations to get the total of $2,000,000$ 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. The number of generations was set to 400.

4) CTDL = TDL + CEL : CTDL combines TDL and CEL as described in Section III-E, with the TDL phase parameters described in *1)* and CEL phase parameters described in *2)*. It alternately repeats the TDL phase and the CEL phase until the total number of games attains $2,000,000$. The exact number of generations depends on the TDL-CEL ratio, which we define as the number of self-played TDL games per one generation of CEL. For example, if the TDL-CEL ratio is 10 (default), there are $3,000$ games per generation (including the round-robin tournament of CEL).

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

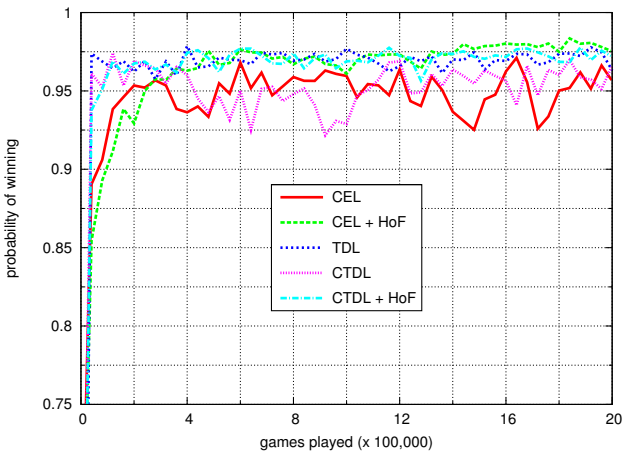

Fig 2. Probability of the best-of-generation individual winning against a random player, plotted against the number of training games played.

B. Results

To monitor the progress of evolution, 50 times per run (approximately every $40,000$ games) we appoint the individual with the highest fitness (i.e., the subjectively best strategy) as the best-of-generation individual and assess its performance. For TDL, we take the only solution maintained by the method. A fully objective assessment requires playing against all possible opponents, but the sheer number of them makes this option impossible. Thus, we rely on two approximate quality measures: playing against a random player and against a predefined, human-designed WPC strategy whose weights were based on the best players presented in [14]. Both of them estimate individual's quality by playing $1,000$ games (500 as black and 500 as white) and calculating the probability of winning.

1) Performance against random player: Figure 2 illustrates how the best-of-generation strategies perform on average against the random player, by which we mean a strategy that makes a (legal) random move in each turn. Data points represent the probability of winning of a best-

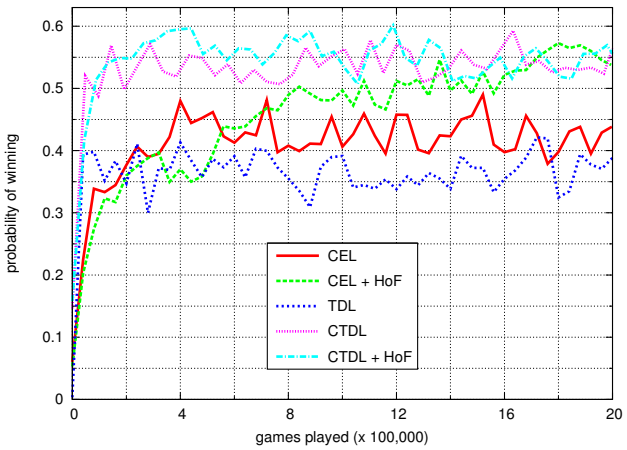

Fig 3. Probability of the best-of-generation individual winning against the heuristic WPC player (both players randomized with probability 0.1), plotted against the number of training games played.

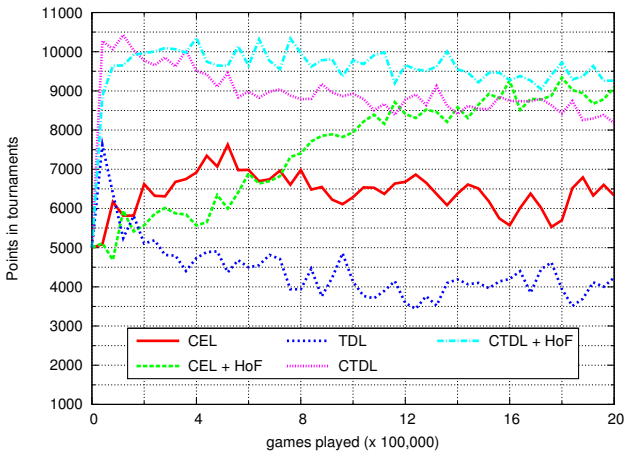

Fig 5. Scores obtained by teams of best-of-generation individuals in the round-robin tournament.

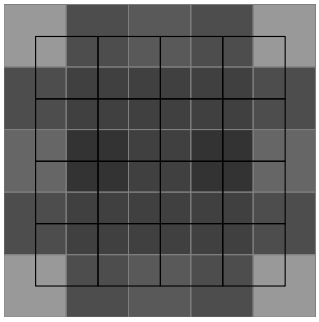

Fig 4. Graphical representation of WPC of the handcrafted heuristic strategy used for performance assessment.

of-generation individual averaged over 25 runs. The graphs clearly indicate that all methods are almost certain to defeat such an opponent. This is not surprising given that, even in case of small 5×5 board, branching factor of Go is relatively high, especially at the beginning of the game, thus there are many ways in which a bad move can be made. Random players for games with smaller branching factor, like Othello, can be more challenging.

It seems that among all compared methods CEL achieves the worst performance on this quality measure. A possible explanation is that other methods can encounter random opponents, or beginners that behave like random players, during the entire learning process, and thus have to remember how to deal with them. In particular, TDL-based methods are taught partially by randomized self-play, whereas HoF-based methods remember the beginner players from the early generations of evolution. In CEL, on the contrary, very bad strategies go extinct in the early stages of the run and infrequently emerge later. As a result, simple CEL can forget how to respond to such player's behavior.

2) Performance against heuristic WPC: Figure 3 presents the probability of winning of best-of-generation individuals when confronted with a handcrafted WPC strategy presented in Table I and graphically in Fig. 4 (the darker the shade, the lower the weight). To obtain this graph, following [23], we forced both players to make random moves with probability $\epsilon = 0.1$; this allowed us to take into account a richer repertoire of players' behaviors and make the resulting estimates more continuous and robust. The plots clearly demonstrate that the fusion of local search with coevolution can be beneficial: methods that incorporate both TDL and CEL quickly pass the 0.5 mark and maintain their performance for the rest of the run, while the remaining ones struggle to make progress. Only CEL+HoF joins the leaders at the end of the run, at much greater computational expense. This is probably possible thanks to the archive (HoF), as pure coevolution (CEL) stucks at a significantly lower performance level.

Though the performance of all methods in absolute terms is rather moderate, this can be attributed to simplicity of WPC representation, which is not the best choice for the highly non-positional game of Go. Let us also notice that we do not know the performance of the optimal WPC-represented Go strategy (if such a strategy exists), so judging the above probabilities as objectively good or bad would be inconsiderate.

3) Round-robin tournament between teams of best individuals: The above results let us conclude that the handcrafted WPC strategy is more challenging than the random player. Unfortunately, even if randomized, such an opponent cannot be expected to represent the full spectrum of possible behaviors of Go strategies represented as WPCs. In order to get a more realistic performance estimate, we recruit a more diverse set of opponents by organizing tournaments between the teams of best-of-generation individuals representing particular methods. The same best-of-generation strategies that were subject to individual performance assessment in previous sections are now combined into five teams, each representing one method and having 25 members (one per run). Next, we play a round-robin tournament between the teams, where each strategy plays against $4 \times 25 = 100$ strategies from the opponent teams for a total of 200 games

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

Team	Games	Wins	Draws	Defeats	Score
CTDL+HoF	5000	3039	147	1814	9264
CEL+HoF	5000	2979	151	1870	9088
CTDL	5000	2666	180	2154	8178
CEL	5000	2050	179	2771	6329
TDL	5000	1361	153	3486	4236

TABLE III
THE RESULTS OF DIRECT MATCHES BETWEEN TEAMS OF BEST-OF-RUN INDIVIDUALS FOUND BY PARTICULAR METHODS.

	CTDL+HoF	CEL+HoF	CTDL	CEL	TDL
CTDL+HoF	—	1974	2052	2385	2853
CEL+HoF	1740	—	2149	2352	2874
CTDL	1653	1558	—	2255	2712
CEL	1326	1359	1445	—	2199
TDL	870	870	996	1500	—

(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 $5,000$ games, using the scoring scheme presented in Section IV-A2. The results of the tournaments, presented in Fig. 5, are even more evident than the former charts: the hybrid methods combining CEL with TDL take the lead extremely quickly, and later on yield only a little to the other algorithms, in particular to CEL+HoF.

The results of the last round-robin tournament, which may be viewed as the relative performance of best-of-run individuals, are presented in detail in Tables II and III. It can be observed that CEL+HoF loses against CTDL+HoF in direct comparison but against other teams it achieves similar or even slightly better performance (against CTDL).

4) Changes observed in genotypic traits: The above aggregate results of multiple runs let us draw sound conclusions about the superiority of some approaches to others, but say little or none about the actual dynamics of the learning process. Figure 6 presents the timeline of snapshots of genotypes of best-of-generation individuals taken from a single CEL+HoF run. It is interesting to note that qualitative genotypic changes occur in bursts, over relatively short time spans, with rather lengthy periods of stagnation in between, which resonates with the theory of punctuated equilibria [46]. It is also striking how qualitatively different are the best genotypes discovered by evolution in particular stages. For instance, the central field is for quite a long time considered as undesired, and then suddenly, after approximately a half million of games (over 100 generations), its utility dramatically changes. The gradual discovery of different forms of symmetry of the board is also remarkable: the central symmetry, evidently present in generations 96 and 112, is later abandoned, and, after a period of exploring asymmetric WPCs, a new form of reflection-like symmetry emerges in the final stages of the run. Notably, this final WPC bears significant resemblance to our heuristic WPC player presented in Table I and Fig. 4, even though the evolving individuals never played against the heuristic WPC, which was used only for external assessment. This convergence is surely not coincidental.

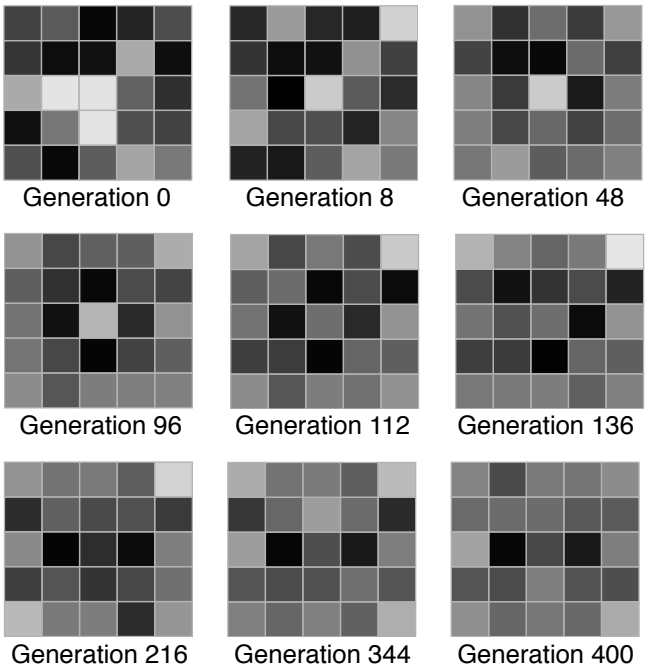

Fig 6. Genotypic changes in the best-of-generation individuals of an exemplary CEL+HoF run. Individuals are illustrated as 5×5 Go boards colored accordingly to correspondingweights (white = -1, , black = 1).

V. CONCLUSION AND DISCUSSION

In [4], we demonstrated that hybridizing coevolution with gradient-based local search proves beneficial when learning strategy of the game of Othello. Here, we come to similar conclusions for the game of Go. Based on that, we hypothesize that CTDL could be also beneficial for other games where both TDL and CEL can be applied separately. Also, there is no reason to claim that this conclusion holds only for strategies represented by WPCs – any representation to which gradient-based search is applicable is likely to benefit from this hybridization.

Though this result is encouraging, we are far from complete understanding of the underlying causes of this synergy. TDL, as a gradient-based technique, is able to link the outcome P of its actions to each parameter of solution (element of WPC matrix) independently and calculate the desired correction vector (cf. Eq. 3). With this skill, it complements evolution, which is devoid of such ability. However, the changes introduced by TDL to individual's genotype result exclusively from the characteristics of that particular individual and abstract from the global characteristics of the population. Thus, such modifications can be incompatible with the current trend of changes resulting from evolution, and potentially deteriorate effectiveness of the search process (e.g., by leading to premature convergence).

This becomes even more intriguing when we realize that in our method TDL actually ignores the fitness assigned to the individual by evolution and trains it by self-play. Therefore,

TDL operates in abstraction from the actual fitness landscape and has no access to player's performance as measured in the context of population. Thus, what we witness here is a quirky variant of local search where the performances of neighboring solutions directly depend on the current (optimized) solution. Yet, as the results prove, TDL is able to substantially improve the convergence of CEL.

The above observations call for in-depth explanation and theoretical support, and point out the future directions we envision for this research.

ACKNOWLEDGMENTS

This work was supported in part by Ministry of Science and Higher Education grant # N N519 3505 33.

REFERENCES

[1] P. Moscato, "Memetic algorithms: A short introduction," *Mcgraw-Hill'S Advanced Topics In Computer Science Series*, pp. 219–234, 1999.

[2] 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.

[3] M. Bardeen, "Td-learning and coevolution: Hiding or guiding?" 2002.

[4] M. Szubert, W. Jaśkowski, and K. Krawiec, "Coevolutionary temporal difference learning for othello," in *IEEE Symposium on Computational Intelligence and Games*, 2009.

[5] E. Lasker, *Go and go-moku: the oriental board games*. Dover Publications, 1960.

[6] R. Bozulich, *The Go Player's Almanac*. Ishi Press, Tokyo, 1992.

[7] A. Lubberts and R. Miikkulainen, "Co-evolving a Go-playing neural network," Workshop, Genetic and Evolutionary Computation Conference, Gecco-2001, San Francisco, 2001.

[8] D. Mechner, "All systems go," *Sciences-New York*, vol. 38, no. 1, pp. 32–37, 1998.

[9] G. Johnson, "To test a powerful computer, play an ancient game," *The New York Times*, 1997.

[10] B. Bouzy, B. Helmstetter, and T. Hsu, "Monte-carlo go developments," in *Advances in Computer Games: Many Games, Many Challenges: Proceedings of the 10th Advances in Computer Games Conference (ACG 10)*. Kluwer Academic Pub, 2003, p. 159.

[11] M. Enzenberger and M. Müller, "Fuego - an open-source framework for board games and go engine based on monte-carlo tree search," 2009.

[12] M. Müller, "Fuego at the computer olympiad in pamplona 2009: a tournament report," 2009.

[13] D. Fotland, "Knowledge Representation in the Many Faces of Go," 1993.

[14] 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.

[15] N. N. Schraudolph, P. Dayan, and T. J. Sejnowski, "Learning to evaluate go positions via temporal difference methods," Tech. Rep., 2000.

[16] B. Bouzy and T. Cazenave, "Computer go: an ai oriented survey," *Artificial Intelligence*, vol. 132, no. 1, pp. 39–103, 2001.

[17] 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.

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

[19] 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.

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

[21] 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.

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

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

[24] 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.

[25] 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, Jun. 2000, pp. 70–79.

[26] 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.

[27] 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.

[28] W. Jaśkowski, K. Krawiec, and B. Wieloch, "Winning ant wars: Evolving a human-competitive game strategy using fitnessless selection," in *Genetic Programming*, vol. 4971. Springer, 2008, pp. 13–24.

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

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

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

[32] 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.

[33] 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.

[34] 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.

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

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

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

[38] 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.

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

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

[41] H. A. Mayer, "Board representations for neural Go players learning by temporal difference," in *IEEE Symposium on Computational Intelligence and Games CIG 2007*, 2007.

[42] 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.

[43] E. P. Manning, "Using resource-limited nash memory to improve an othello evaluation function," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 1, pp. 40–53, 2010.

[44] 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.

[45] S. Luke, "Ecj 19 - a java-based evolutionary computation research system," *http://cs.gmu.edu/~eclab/projects/ecj/*, 2009.

[46] N. Eldredge and S. Gould, "Punctuated equilibria: an alternative to phyletic gradualism," *Models in paleobiology*, vol. 82, p. 115, 1972.