Proceedings of the 2007 IEEE Symposium on
Computational Intelligence and Games (CIG 2007)

Coevolving Strategies for General Game Playing

Joseph Reisinger, Erkin Bahçeci, Igor Karpov and Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin
{joeraii,erkin,ikarpov,risto}@cs.utexas.edu

Abstract— **The General Game Playing Competition [1] poses a unique challenge for Artificial Intelligence. To be successful, a player must learn to play well in a limited number of example games encoded in first-order logic and then generalize its game play to previously unseen games with entirely different rules. Because good opponents are usually not available, learning algorithms must come up with plausible opponent strategies in order to benchmark performance. One approach to simultaneously learning all player strategies is coevolution. This paper presents a coevolutionary approach using NeuroEvolution of Augmenting Topologies to evolve populations of game state evaluators. This approach is tested on a sample of games from the General Game Playing Competition and shown to be effective: It allows the algorithm designer to minimize the amount of domain knowledge built into the system, which leads to more general game play and allows modeling opponent strategies efficiently. Furthermore, the General Game Playing domain proves to be a powerful tool for developing and testing coevolutionary methods.**

Keywords: Coevolution, General Game Playing, Artificial Neural Networks

I. INTRODUCTION

The General Game Playing (GGP) Competition has been held since 2005 at the National Conference for Artificial Intelligence [1], [2]. It is a multi-round competition employing dozens of games, both familiar and new, and poses a unique challenge for Artificial Intelligence (AI) methods by requiring a successful player to perform approximate heuristic search and generalize from a limited number of example games to a large class of previously unseen games. GGP is an interesting problem because it requires the algorithm designer to take into account the kinds of structures and commonalities that may exist *across* games. This emphasis in turn advances the cause of general AI.

The most successful machine game players, such as Deep Blue and Chinook, use game tree evaluation techniques supplemented by carefully crafted and game-specific databases of human knowledge [3], [4]. While very successful, methods that rely on a custom board evaluation function do not work in GGP because the expert domain knowledge is not available for previously unseen games. Other game players employ a more learning-oriented approach to game playing. For example, TD-gammon learned backgammon by playing games against a handicapped copy of itself [5]. Although much of the gameplay is learned, such implementations still require internal game representations to be hand-coded by the designer and thus do no apply to GGP.

In contrast, coevolution is potentially a highly effective approach to general game playing. In coevolution, the space of possible policies and multiple opponent strategies are explored with little a priori knowledge. This paper introduces `nnrg.hazel`, a first implementation of the coevolutionary approach to GGP. There are four main benefits of such an approach: 1) it minimizes the amount of domain knowledge required, 2) it is applicable to a broad range of games, 3) it generates competent solutions and 4) it uses fewer evaluations than standard evolutionary approaches. The `nnrg.hazel` agent uses NeuroEvolution of Augmenting Topologies (NEAT) [6] to evolve a population of complexifying artificial neural networks to serve as heuristic state evaluators in minimax partial game tree search. Each evolved neural network represents a policy for playing as a particular player in a particular game. The fitness of each policy is determined by playing against opponent strategies that are evolved as well.

The `nnrg.hazel` agent competed in the AAAI 2006 General Game Playing Competition [1] It came in 5th place in the preliminary rounds and 6th place out of 12 overall, which is significant considering that the competition was targeted towards symbolic approaches and did not allow much time for learning. This paper evaluates `nnrg.hazel`'s ability to learn general behavior on a sample of games from the GGP corpus; in the long-term this work serves as a first step towards establishing GGP as a tool for testing coevolutionary methods and in general establishing coevolution as a robust method for learning general strategies in multiplayer games.

This paper is divided into seven sections: Section 2 gives an overview of the GGP domain, section 3 discusses the coevolutionary approach for evolving neural network heuristic evaluators and section 4 gives the results on six different games. Finally, sections 5–7 evaluate the contributions and discuss future work opportunities of this study.

II. GENERAL GAME PLAYING

The General Game Playing Competition consists of perfect-information, deterministic games with any number of cooperating and competing players. The games can involve simultaneous moves or can be turn based. Game rules in GGP are specified in a Game Description Language (GDL), which originates from KIF, a language to describe knowledge in first-order logic [7], [1]. The GDL specification for a game contains the initial state, legal move rules, state transition rules, goal score rules, and terminal state rules. Each game state consists of some number of clauses which are updated in response to player moves according to a set of state transition rules.

```
 1. (role xplayer)
 2. (role oplayer)
 3. (init (cell 1 1 b))
 4. (init (cell 1 2 b))
 ...
 5. (init (cell 3 3 b))
 6. (init (control xplayer))
 7. (<= (next (cell ?m ?n x))
 8.     (does xplayer (mark ?m ?n))
 9.     (true (cell ?m ?n b)))
10. (<= (next (control xplayer))
11.     (true (control oplayer)))
 ...
13. (<= (row ?m ?x)
14.     (true (cell ?m 1 ?x))
15.     (true (cell ?m 2 ?x))
16.     (true (cell ?m 3 ?x)))
17. (<= (line ?x)
18.     (row ?m ?x))
 ...
19. (<= (legal ?w (mark ?x ?y))
20.     (true (cell ?x ?y b))
21.     (true (control ?w)))
22. (<= (legal xplayer noop)
23.     (true (control oplayer)))
 ...
24. (<= (goal xplayer 0)   (line o))
25. (<= (goal oplayer 100) (line o))
 ...
26. (<= terminal (line x))
```

Fig. 1. **Condensed description of Tic Tac Toe in the GDL.**

A digest of the GDL representation for Tic Tac Toe is given in figure 1. Lines 1 and 2 specify the player roles, `xplayer` and `oplayer`. Lines 3–6 define which clauses are true in the initial state of the game (`init` clauses); in this case the initial state consists of a blank board with control given to `xplayer`). The `next` rules specify which clauses will become true in the next state given the current state and the moves chosen by each player. For example, line 7 specifies that a blank board cell will become marked as `x` if `xplayer` marks that cell, and line 10 states that after `xplayer` makes a move, control will pass to `oplayer` (simulating turn-taking). Rules at lines 19 and 22 give examples of moves that are legal for each player. In Tic Tac Toe, the player who does not have control has only a single move choice: `noop`. Lines 24–25 specify some of the goal conditions, which are rules describing how many points each player receives at the end of the game. Note that the `line` relation is a function defined specifically for Tic Tac Toe (line 13). Finally, `terminal` rules (e.g. line 26) determine when a game will end.

During the GGP competition, game playing agents are connected to a game server that conducts the matches. At the beginning of each game, the game server sends players the rules. The players then have a fixed amount of time to process the game rules before play begins. During gameplay, each player has a limited amount of time (*play clock*) to make its move. If a player does not send its move within the required time or if it attempts an illegal move, the game server picks a random move for that player. A game ends when a terminal state is reached. Since the competition is designed mainly for symbolic relational approaches, the time constraints proved to be restrictive for approaches relying on game simulation rather than rule-knowledge. However the domain itself is interesting as an incubator for learning general behavior and for analyzing coevolutionary learning approaches.

Since general game playing is a relatively young research area, relatively few studies have been conducted. Pell employed a logic-programming approach to generalize play to the category of chess-like games [2]. Games were encoded in a logical game description language and the rules of a specific game were processed using logic-programming methods, such as partial evaluation and abstract interpretation. The goal was to automatically convert an inefficient general player into a more efficient specialized player for that particular game. The GGP Competition can be viewed as an extension of Pell's work to a much larger class of games.

Expanding on Pell's work, Epstein et al. considered two-person, perfect information board games more generally [8]. Their player, Hoyle, used a multi-tiered architecture. It first tries to decide on a move using a few basic heuristics to avoid obvious mistakes; if a decision cannot be made by the first tier, the second tier is consulted. The second tier consists of *advisors* (domain-specific heuristics) that *comment* on the current state and a decision is made based on these comments. Hoyle learns to use these advisors for the particular game by playing against an external expert model. Inspired by human players' use of spatial pattern recognition abilities, Epstein et al. later applied their architecture to spatial heuristic learning as well [9]. Although generally applicable, Hoyle relies on preexisting expert players; in GGP, such experts do not exist in general. Players must learn competitive strategies on their own, given only the game rules.

Kuhlmann et al. extend the concept of general heuristic learning to the GGP competition [10]. Through internal game simulations their agent automatically discovered features such as piece count and distances between piece pairs from the game rules. These features were then used to derive heuristics, which in turn were ranked based on their applicability to the current game. Although this approach makes several assumptions about the kinds of heuristics that are likely to work well across a broad spectrum of games it performs very well in practice: The agent based on Kuhlmann's work placed 3rd in the 2006 GGP Competition.

In contrast, the basic `nnrg.hazel` agent framework, strives to be truly general. It avoids using domain knowledge and instead relies solely on internal simulation of game dynamics to refine its gameplay, and will be described next.

III. Monotonic Coevolution with NEAT

Using a neuroevolution (NE) method called NEAT, heuristic game state evaluators for states without explicit goal conditions are generated; opponent strategies are coevolved in order to focus search in promising areas of strategy space. Section III-A describes the NEAT algorithm, III-B, its application to minimax search and III-C, the general coevolutionary framework.

A. The NEAT Method

NeuroEvolution of Augmenting Topologies (NEAT) [6] is a policy-search reinforcement learning method that uses a

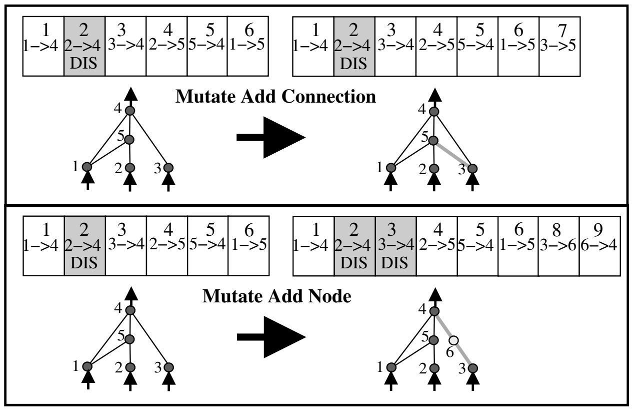

Fig. 2. **The two types of structural mutation in NEAT.** The genetic encoding of each network is shown as a series of genes. In each gene, the innovation number is shown on top, the two nodes connected by the gene in the middle, and a possible "disabled" symbol at the bottom; the weight value is not shown. A new connection or node is added to the network by adding connection genes to the genome. Assuming the connection (left) is added before the node, the genes would be assigned innovation numbers 7, 8, and 9, as the figure illustrates. In this manner, the origin of every gene in the population is known, allowing matching genes to be identified in different genome structures.

genetic algorithm to find optimal neural network policies. NEAT automatically evolves network topology to fit the complexity of the problem while simultaneously optimizing network weights. NEAT employs three key ideas: 1) incremental complexification using a variable-length genome, 2) protecting innovation through speciation, and 3) keeping dimensionality small by starting with minimally connected networks. By starting with simple networks and expanding the search space only when beneficial, NEAT is able to find significantly more complex controllers than other fixed-topology learning algorithms. This approach is highly effective: NEAT outperforms other NE methods on complex control tasks like double pole balancing [6] and robotic strategy-learning [11]. These properties make NEAT an attractive method for evolving neural networks in complex tasks.

Each genome in NEAT includes a list of *connection genes*, each of which refers to two *node genes* being connected. Each connection gene specifies the in-node, the out-node, the weight of the connection, whether or not the connection gene is expressed (an enable bit), and an *innovation number*, which allows finding corresponding genes during crossover (figure 2). Innovation numbers are inherited and allow NEAT to perform crossover without the need for expensive topological analysis. Genomes of different organizations and sizes stay compatible throughout evolution, and the problem of matching different topologies [12] is essentially avoided. NEAT speciates the population so that individuals compete primarily within their own niches instead of with the population at large. This way, topological innovations are protected and have time to optimize their structure before they have to compete with other niches in the population. The reproduction mechanism for NEAT is *explicit fitness sharing* [13], where organisms in the same species must share the fitness of their niche, preventing any one species from taking over the population.

The principled complexification exhibited by NEAT is a desirable property in competitive coevolution: As the antagonistic populations refine their strategies and counter-strategies, complexification becomes necessary in order to generate novel strategies without "forgetting" past strategies [14]. The next section describes how NEAT is applied in the context of GGP.

B. Heuristic Game State Evaluators

The neural networks generated by NEAT are evaluated as heuristic functions estimating the value of game states when explicit goal information is not available. These heuristic functions are combined with standard lookahead using α-β-pruned minimax [15]. During the competition as well as the experiments presented in this paper, lookahead search is restricted to a single ply in order to reduce evaluation time, allowing longer evolutionary runs.

Initially, the neural networks start out as a fully connected mapping of 40 input neurons to a single output neuron. The inputs represent the current state and the output specifies the heuristic value of the state. How to represent the game state effectively is a central challenge for coevolving general game-players. The first order logic of the GDL is fundamentally more expressive than the finite set of propositional features that can be encoded in the input layer of an artificial neural network. While it is possible to determine the input layer structure for each game a priori, such an approach would make generalization difficult. Therefore, `nnrg.hazel` utilizes a random projection of input features onto the 40 input neurons, which requires only knowledge of possible state clauses. An input layer size of 40 was chosen to reduce the overall number of parameters while avoiding state aliasing. It may be possible to do mapping more intelligently, for instance using game feature detectors [10]; such methods are an important future research direction.

In games with simultaneous moves or more than two players, minimax search is inefficient because each player move must be evaluated against all possible combinations of opponent moves. With more than two players such combinatorial evaluation is necessary when it is necessary to be robust against coalitions; if moves are made simultaneously it ensures robust play given any contingency. In order to make fitness evaluation efficient, heuristic move values in simultaneous games are computed against a random sampling of opponent moves. Such an optimization ensures that the move ultimately selected is a good response to *some* subset of opponent moves without having to evaluate all possible simultaneous moves. Furthermore, the risk of coalition-formation during the competition is negligible as player identities are not made available.

C. Coevolutionary Setup

To ensure that the evolved heuristic evaluators work well against a range of opponents, the opponent strategies themselves are evolved simultaneously through coevolution. In coevolution, an individual's fitness is evaluated against some combination of opponents drawn from the evolving populations, rather than against a fixed fitness metric. This approach yields several major benefits over traditional evolution: 1) Coevolution allows the opponent strategies to be learned by the algorithm, reducing the amount of information the algorithm designer must provide *a priori*, 2) Under certain conditions, coevolution may facilitate *arms races*, where individuals in both populations strategically complexify in order to learn more robust behaviors [16], 3) Coevolution may reduce the total number of evaluations necessary to learn such robust strategies, leading to more efficient search [17].

In order to facilitate arms races and make coevolution efficient, the algorithm needs to ensure monotonic progress. Without such a guarantee, as evolution progresses populations can "forget" past strategies, resulting in cycling behavior [17], [18]. Several methods have been proposed to ensure monotonic progress, implementing several game-theoretic solution concepts: The Pareto-Optimal Equivalence Set (IPCA) [19], Nash Equilibria [20], and Maximization of Expected Utility (MaxSolve) [17].

The `nnrg.hazel` agent employs Rosin's *covering competitive algorithm* (CCA), which ensures monotonic progress towards some Pareto optimal solution [21]. Unlike more complex algorithms which generate the entire Pareto-Optimal equivalence set (e.g. IPCA [19]), the CCA focuses on finding a single Pareto optimal solution, making it ideal for use during competitions. CCA is similar to the Dominance Tournament algorithm described in previous NE literature (e.g. [11]).

The basic CCA algorithm for the two-population case is as follows. Each population maintains a ranked set of teachers T that are the previous dominating strategies for that player (in the order which they were developed). A population is said to be *currently dominated* if it has not yet developed a solution that can beat all teachers in the opponent population. At each generation, each individual a from the currently dominated population A plays each teacher t_B from the ordered set of opponent teachers T_B in rank order. If a loses a game against some t_B, no further games are performed. If a is capable of beating all $t_B \in T_B$, then a is inducted into T_A and B is set to be the currently dominated population. At the end of the evolutionary run, the highest ranking member in T for each population is taken to be the final solution.

In addition to CCA, each member of both populations also plays a number of bouts against random combinations of opponents from the currently evolving populations. Fitness is calculated as the weighted sum of the scores obtained during the CCA evaluations and during N normal evaluations,

$$F(o) = \alpha \sum_{d \in D_o} f_d(o) + (1-\alpha) \sum_{i=0}^{N} f_{X_i}(o),$$

where o is the organism being evaluated, D_o is all combinations of opponent strategies in the dominance ranking that o was able to beat, and X_i is a random variable mapping i to some combination of opponent strategies in the current population. In all reported experiments $\alpha = \frac{10}{11}$ and $N = 50$. This evaluation strategy focuses coevolutionary search on individuals capable of achieving high CCA rankings (i.e. individuals capable of outperforming many opponent strategies). Although other parameter variants are not considered in this paper, previous experiments with NEAT have shown these settings to be robust. CCA was chosen based on its previous empirical successes: It has been applied successfully to several complex domains including 9×9 Go. In the next section CCA is evaluated on a five game sample from the GGP corpus, elucidating several of its strengths and weaknesses.

IV. Results

This section presents learning curve, scalability and disengagement results obtained in applying coevolutionary NEAT to GGP. The `nnrg.hazel` agent was evaluated on the following sample of two-player games from the GGP corpus:

1) *Connect Four* - The standard Connect Four rules; players score 100 points for a win, zero points for a loss, and 50 for games with no winner.
2) *Chinese Checkers* - A small two-player variant of Chinese Checkers using standard rules. Three pegs in the goal yields 100 points, two yields 50, one yields 25 and zero yields 0.
3) *Criss Cross* - Same rules as Chinese Checkers but with a smaller board.
4) *Blocker* - Two players on a 4×4 board. The crosser tries to build a bridge to the other side of the board one square at a time. The blocker impedes the progress of the crosser by placing walls. Play ends when the board is full or the crosser has made it across. Crosser wins 100 points for making a bridge (blocker wins 0), and the blocker wins 100 points for keeping the crosser from making it across.
5) *Two-Board Tic-Tac-Toe* - Tic-tac-toe on two boards at once. Players alternate turns playing on each board. The first player to get three in a row wins 100 points. Tie games yield 50 points apiece.

For each game, evolution was run for 300 generations (longer than the constraints of the competition) and the learning curves, average teacher set size, scalability and disengagement results are presented here. Confidence scores are calculated using Student's t-test with $\alpha = 0.05$ and all error bars depict the 95% confidence interval.

A. Learning Curves

Standard evolutionary plots of fitness are not meaningful in coevolution because fitness scores are inherently relative. However, since CCA guarantees monotonic progress, the teacher set size for each role can be taken as a rough measure of the best fitness over time. A plot of the average number of teachers in each role per generation is given in figure 3. Blocker is omitted as the `crosser` role never exceeds a teacher set size of one. Based on these plots, the six games can be divided roughly into three groups: Chinese Checkers and Melee Chess both reach approximately 30 teachers, Connect

Four and Criss Cross both approximately 10, and Blocker and Double Tic Tac Toe have fewer than 5 teachers. Blocker in particular shows strong evidence of disengagement: the `crosser` role is not able to beat the `blocker` teachers.

Learning curves are calculated by playing each teacher as a 1-ply heuristic against a random heuristic searching to the same depth. Results are plotted as the teacher's rank vs. the average match score (averaged over 100 trials; figure 4). In Connect Four and Criss Cross, the evolved solutions perform significantly better than random 1-ply players for both roles. In Connect Four, both roles outperform random (`white`: $p < 0.02$, `black`: $p < 0.03$); in Criss Cross, both roles outperform random (`red`: $p < 10^{-3}$, `teal` $p < 10^{-4}$). In Blocker, `blocker` outperforms random ($p = 0.1$) but `crosser` does not. In Double Tic Tac Toe, `xplayer` outperforms random ($p < 0.03$) but the `oplayer` is not significantly better than random. Finally, in Chinese Checkers, the evolved strategies significantly *underperformed* against the random player; this result is analyzed in more detail in the discussion section. Overall the evolved players perform significantly better than the random ones, indicating that coevolution is generating robust strategies rather than strategies targeted to beat only the other population's teachers.

B. Scalability

Heuristic evaluators in `nnrg.hazel` are evolved using only 1-ply search in order to minimize the amount of time required for each evaluation. This approach leads to a possible problem: The evolved heuristic evaluators might be tuned to the structure of the 1-ply problem and not scale well when used as n-ply evaluators. Scalability can be tested simply by using the evolved 1-ply heuristics to perform deeper search. However this approach does not control for the performance benefit of simply looking ahead farther into the game tree. In order to control for this benefit, the random player is given knowledge of the goal states and the same lookahead depth as the evolved player. Thus the benefit from deeper lookahead is equal for both players and any performance difference must be due to the better heuristic.

A summary of the scalability results is given in figure 5. Except in the case of Chinese Checkers, performance decreases slightly as the search depth is increased. The `red` player in Chinese Checkers plays significantly better with 2-ply search than with 1-ply search (4135 to 4656 points; $p < 10^{-8}$) though there is no significant difference between 2-ply and 3-ply search ($p > 0.1$). The `teal` role, on the other hand, plays slightly worse with 1-ply search than with 3-ply search (5382 to 5126 points; $p = 0.03$). All other games exhibit statistically significant decreases in performance for both roles: Criss Cross `teal` decreases from 7799 points to 7493 points, `red` decreases from 8360 points to 7224 points, Connect Four `white` decreases from 7586 points to 6746 points, `black` decreases from 6729 points to 5690 points, Double Tic Tac Toe `xplayer` decreases from 4798 points to 4336 points and `oplayer` decreases from 6420 points to 5836 points. Taken together, these results indicate that there is

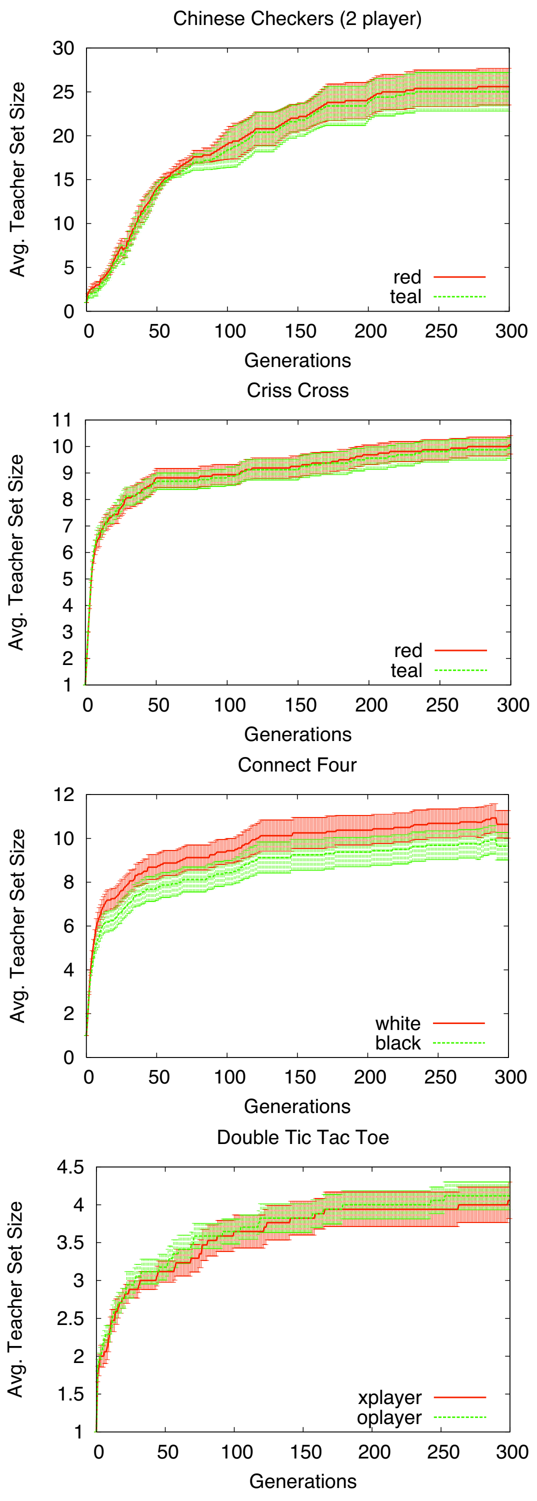

Fig. 3. **Average teacher set size achieved at each generation.** The number of teachers indicates how disengaged the coevolutionary system is. Games that achieve large teacher sets are more likely to exhibit arms-race dynamics. Except for in Blocker (omitted), `nnrg.hazel` found coevolutionary gradients as evinced by the increase in teacher set size.

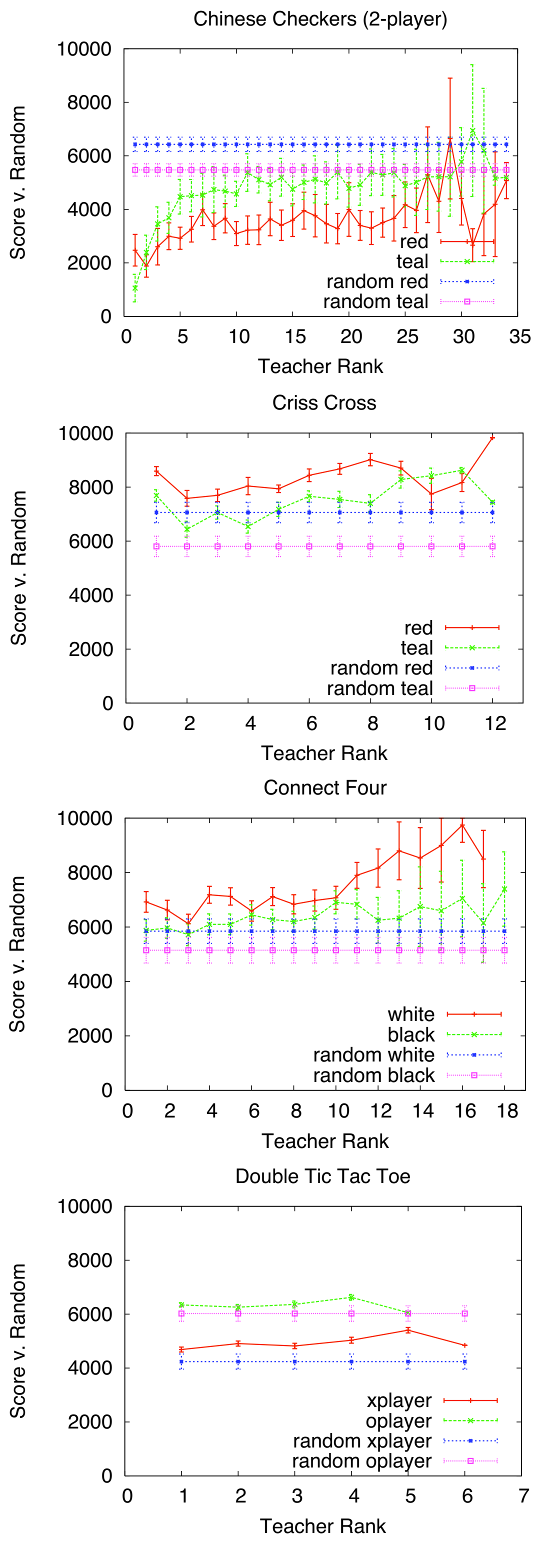

Fig. 4. **Teacher rank vs. average performance against a 1-ply random heuristic.** Except for Chinese Checkers, all evolved players outperform 1-ply random starting with the first teacher rank, indicating that most of evolutionary progress is concerned with refining strategies that comparison to random cannot elucidate. Although it is outperformed by random, Chinese Checkers exhibits strong evidence of learning.

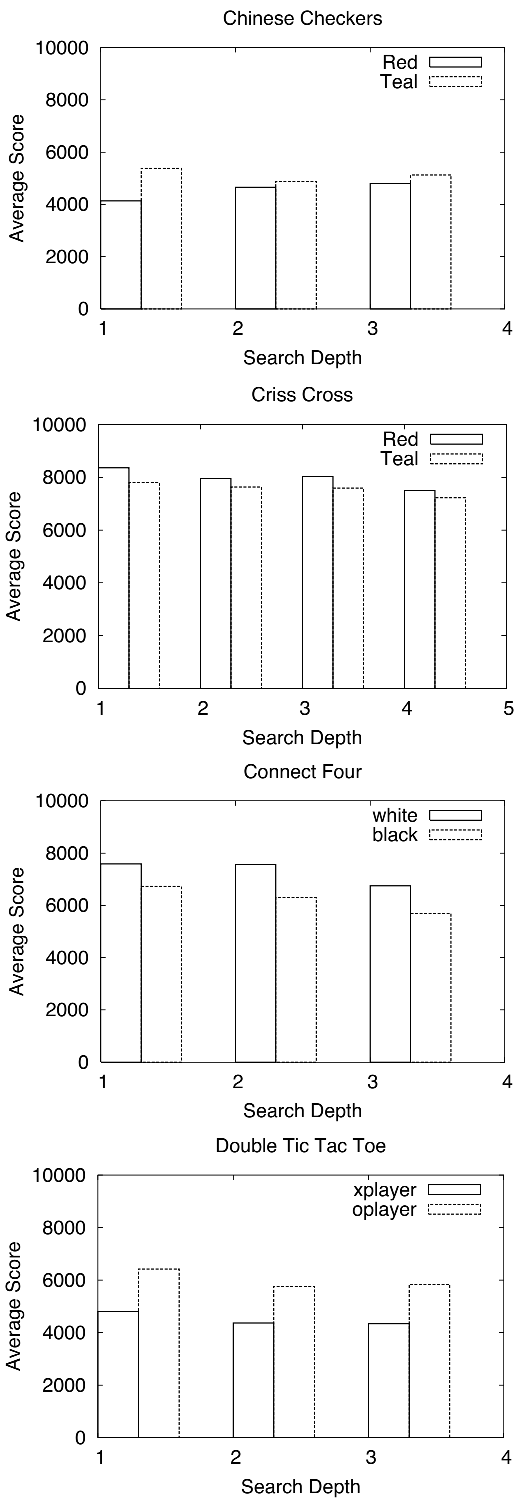

Fig. 5. **Performance as search depth increases.** This plot depicts the average performance of the highest ranked teachers for each role as the search depth for both players is increased. In almost all cases, performance decreases as depth is increased, indicating that training on 1-ply search does not necessarily scale to deeper search.

	Tied	Role 0	Role 1	p
blocker	0.0	19.0	281.0	$< 10^{-32}$
chinese checkers	0.86	189.86	109.29	> 0.15
criss cross	4.56	188.13	107.31	< 0.05
connect 4	4.1	288.0	7.9	$< 10^{-42}$
tictactoe	0.7	152.0	147.29	> 0.9

Fig. 6. **Average number of generations each player is undominated.** The higher the disparity between the two means, the more significant the disengagement.

a trade-off during learning: Evaluations should be performed using deeper search in order to ensure better scalability, but doing so would reduce the total number of generations that would fit in the allotted time during the competition.

C. Disengagement

Even with monotonic progress guarantees, coevolution can stall due to *disengagement*, i.e. when one population outperforms the other to the point that there is no longer a clear gradient that selection can follow. Although disengagement did not impact `nnrg.hazel`'s performance during the competition, in the extended evolutionary runs presented here, it turns out to be an issue. In two of the games, the coevolving populations became significantly disengaged (figure 6). In order to measure disengagement quantitatively, first the number of generations where each teacher set was not dominated by the other is computed. This value is averaged per role, yielding the average amount of time that each role dominates the other. Disengagement is exhibited when these means differ significantly, according to a Student's t-test, and the degree of disengagement is reflected by the confidence score.

In Blocker, the `crosser` role dominates on average 19 out of 300 generations, while `blocker` dominates 281 out of 300 generations. In Connect Four, `white` dominates 288 out 300 generations while `black` dominates 7.9 out of 300. Criss Cross exhibits significant disengagement: The `red` role dominates 188.13 out of 300 generations while `teal` dominates 109.29 out of 300. However, the disengagement in Criss Cross is not as severe as in the other two games. In all other games, the two means do not differ significantly, indicating that disengagement is less common. However, since disengagement impacts the performance of a non-trivial proportion of games sampled, it should be addressed in future work.

V. Discussion

The `nnrg.hazel` agent performs well against the random player in most cases suggesting that coevolution is able to extract useful game playing behavior in general. This is a positive result, especially given a simple random projection is used to map he state space to input layer, rendering it difficult to detect structure in the domain. Also, the coevolutionary fitness function actually results in few games against truly random players. If beating the random player were the true evolutionary goal, then a better approach would be to perform hundreds of evaluations against such a player until a statistically significant average score is produced. Since the coevolutionary teacher set size rarely exceeds 30, such an approach would require an order of magnitude more evaluations.

Since coevolution allows learning all players' strategies simultaneously, it is difficult to measure absolute progress without having another game player to benchmark against. In the GGP competition, competitors are expected to play as optimally as possible given the time constraints. Thus performance against a random opponent may not be the best benchmark for predicting a player's actual performance during the competition. One interesting application of the GGP competition is to form a relative performance metric for completely new games.

Similarly, the suboptimal performance in the Blocker, Double Tic Tac Toe and Chinese Checkers domains leads to interesting insights about the pitfalls and limitations of this approach. First, the Blocker domain presents a unique challenge to learning as the random play dynamics with opponent move sampling favor the `blocker` role to win approximately 95% of the time. Since the game is inherently symmetric and simultaneous, this phenomenon is due to a single rule in the game: If both players play on the same location, `blocker`'s play takes precedence. This advantage turns out to be so strong that in the one-ply case, evolution cannot find more effective `crosser` strategies. Second, the performance in Double Tic Tac Toe can also be traced to the opponent move sampling heuristic, which causes gameplay to become stochastic, yielding less effective solutions. Third, the performance of Chinese Checkers is due to backtracking: The player is able to return to previous states by simply doing the reverse of the last move. Coupled with heuristic evaluation, this ability causes states with a particularly high score to act as basins of attraction, forcing the play into a cyclic behavior. This limitation can only be addressed by adding domain knowledge from the rules. In general there is no state aliasing in GGP: most games keep a move counter which is part of the state. Thus some kind of analysis would have to be performed to remove such state counters and identify states which are the same. This is precisely the approach taken by Kuhlmann et al. [10].

In light of these results, several areas of future work become readily apparent; the next section is devoted to their discussion.

VI. Future Work

The results elucidate several areas for future work that can largely be divided into 1) improvements addressing weaknesses in the coevolutionary approach, 2) improvements that can be achieved by including domain knowledge and 3) general improvements to make `nnrg.hazel` more competitive.

Based on the result in section IV-C it is clear that coevolutionary disengagement is a problem in several of the games tested, most likely due to inherent imbalances under random play. Disengagement can be mitigated by providing evolution with a smooth gradient; in CCA for example a *managed challenge* can be used where solutions that are difficult to beat are stored in a test bank for future comparisons [22].

Since CCA was developed, a number of coevolutionary algorithms have been proposed implementing various other solution concepts, such as Nash equilibrium [20], the Pareto-Optimal Equivalence Set [19], and Maximization of Expected Utility [17]. An interesting direction for future work is determining what solution concept (or mixture of concepts) is best for the most general class of games. In cooperative games, for example, the desired solution concept is to maximize social welfare. Whereas in antagonistic games, Pareto optimality may be a more useful goal.

Mapping the state space to the neural network input layer randomly is a source of inefficiency. Even in simple games states are likely to be aliased thus the neural network has no way of distinguishing the fine details of the game. However, one reason the random project was chosen over other, more intelligent, methods is that it makes minimal assumptions about the game. Generalized heuristics do exist for identifying optimal input mapping structures (e.g. RAAM [23]) and should be evaluated in this domain.

Currently, heuristics are only evaluated in a one-ply search scenario but later used to control iterative deepened search during the actual competition. From the results in section IV-B it is clear that this approach does not necessarily scale, at least against random opponents. Better results could be achieved by performing deeper search during evolution itself; however, before doing so it is imperative to study the inherent trade-off due to longer learning times.

Finally, an interesting direction for research in GGP is evolving a general *basis* population for seeding evolutionary search. A basis population capable of outperforming random initialization in finding good solutions early during evolution would be particularly beneficial during the competition. Such a population must, by nature, capture some essence of game playing that is common across most games. Such work would be closely related to the field of transfer learning.

VII. Conclusion

This study introduced a coevolutionary system, `nnrg.hazel`, for learning effective strategies in the General Game Playing domain. In doing so, this work represents (to our knowledge) the first comparison of generic coevolutionary performance across several games. The `nnrg.hazel` agent was designed from the beginning to require as little domain knowledge as possible from the game implementation, allowing it to be truly general. In experiments testing learning and search scalability it was found that in general the coevolved heuristics outperformed random, but the learned heuristics did not necessarily scale well to arbitrary search depths. Some domains also exhibited evidence of coevolutionary disengagement, indicating that more efficient search may be possible. Overall General Game Playing is an important tool for improving coevolutionary algorithms as well as for further exploration in AI focusing on broadly applicable problem solving strategies.

Acknowledgments

The authors would like to thank the Stanford Logic Group, particularly Nat Love and Tim Hinrichs, for developing and running the General Game Playing competition. This research was supported in part by the National Science Foundation under CISE Research Infrastructure Grant EIA-0303609 and through a Graduate Research Fellowship to the first author.

References

[1] M. R. Genesereth, N. Love, and B. Pell, "General game playing: Overview of the aaai competition," *AI Magazine*, vol. 26, no. 2, pp. 62–72, 2005.

[2] B. D. Pell, "Strategy generation and evaluation for meta-game playing," Ph.D. dissertation, University of Cambridge, August 1993.

[3] J. Schaeffer, R. Lake, P. Lu, and M. Bryant, "CHINOOK: The world man-machine checkers champion," *The AI Magazine*, vol. 16, no. 1, pp. 21–29, 1996.

[4] M. Campbell, A. J. H. Jr., and F. hsiung Hsu, "Deep blue," *Artificial Intelligence*, vol. 134, no. 1-2, pp. 57–83, 2002.

[5] G. Tesauro, "TD-gammon, a self-teaching backgammon program achieves master-level play," *Neural Computation*, vol. 6, pp. 215–219, 1994.

[6] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[7] M. R. Genesereth, "Knowledge interchange format," in *Principles of Knowledge Representation and Reasoning: Proceedings of the Second Intl. Conference (KR '91)*, 1991, pp. 599–600.

[8] S. L. Epstein, "For the right reasons: The forr architecture for learning in a skill domain." *Cognitive Science*, vol. 18, no. 3, pp. 479–511, 1994.

[9] S. L. Epstein, J. Gelfand, and E. Lock, "Learning game-specific spatially-oriented heuristics," *Constraints*, vol. 3, no. 2-3, pp. 239–253, 1998.

[10] G. Kuhlmann, K. Dresner, and P. Stone, "Automatic heuristic contruction in a complete general game player," in *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, July 2006.

[11] K. O. Stanley and R. Miikkulainen, "Competitive coevolution through evolutionary complexification," *Journal of Artificial Intelligence Research*, vol. 21, pp. 63–100, 2004.

[12] N. J. Radcliffe, "Genetic set recombination and its application to neural network topology optimization," *Neural computing and applications*, vol. 1, no. 1, pp. 67–90, 1993.

[13] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proceedings of the Second International Conference on Genetic Algorithms*, J. J. Grefenstette, Ed. San Francisco: Kaufmann, 1987, pp. 148–154.

[14] K. O. Stanley and R. Miikkulainen, "Continual coevolution through complexification," in *Proceedings of the Genetic and Evolutionary Computation Conference*. San Francisco: Kaufmann, 2002.

[15] D. E. Knuth and R. W. Moore, "An analysis of alpha-beta pruning," *Artificial Intelligence*, vol. 6, pp. 293–326, 1975.

[16] L. Van Valin, "A new evolutionary law," *Evolution Theory*, vol. 1, pp. 1–30, 1973.

[17] E. de Jong, "The maxsolve algorithm for coevolution," in *GECCO '05: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM Press, 2005, pp. 483–489.

[18] S. G. Ficici, "Monotonic solution concepts in coevolution," in *GECCO '05: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM Press, 2005, pp. 499–506.

[19] E. D. de Jong, "The incremental pareto-coevolution archive," in *GECCO '04: Proceedings of the 2004 Conference on Genetic and Evolutionary Computation*. Springer, 2004, pp. 525–536.

[20] S. G. Ficici and J. B. Pollack, "A game-theoretic memory mechanism for coevolution," in *GECCO '03: Proceedings of the 2003 Conference on Genetic and Evolutionary Computation*, 2003, pp. 286–297.

[21] C. Rosin, "Coevolutionary search among adversaries," Ph.D. dissertation, University of California, San Diego, 1997.

[22] J. C. Bongard and H. Lipson, "'managed challenge' alleviates disengagement in co-evolutionary system identification," in *GECCO '05: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM Press, 2005, pp. 531–538.

[23] J. B. Pollack, "Recursive auto-associative memory: Devising compositional distributed representations," in *Proceedings of the 10th Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum, 1988, pp. 33–39.