Analyzing Competitive Coevolution across Families of N-Player Games through Tree Search

Sean N. Harris
snh0037@auburn.edu
BONSAI Lab, Auburn University
Auburn, AL, USA

Daniel R. Tauritz
dtauritz@acm.org
BONSAI Lab, Auburn University
Auburn, AL, USA

Samuel Mulder
szm0211@auburn.edu
BONSAI Lab, Auburn University
Auburn, AL, USA

Abstract

In this work, we examine Monte Carlo tree search (MCTS) as an adaptive benchmark for competitive coevolution, both as a method for characterizing individual solutions, and as a measure of global progress throughout a coevolutionary run. We find that MCTS provides a much more practical measurement of solution quality than existing methods of comparing to randomly-sampled solutions. Additionally, we introduce a class of n-player games for which MCTS can be shown to have comparable performance for different n, allowing for the characterization of how coevolution performs in games with different numbers of players. We demonstrate these techniques for n-player Nim and an n-player variant of Othello.

CCS Concepts

• **Theory of computation** → **Evolutionary algorithms**; • **Computing methodologies** → *Adversarial learning*; **Genetic algorithms**.

Keywords

competitive coevolution, n-player games, Monte Carlo tree search

ACM Reference Format:

Sean N. Harris, Daniel R. Tauritz, and Samuel Mulder. 2025. Analyzing Competitive Coevolution across Families of N-Player Games through Tree Search. In *Foundations of Genetic Algorithms XVIII (FOGA '25), August 27–29, 2025, Leiden, Netherlands.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3729878.3746621

1 Introduction

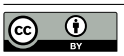
The field of competitive coevolution studies the application of evolutionary algorithms (EAs) to adversarial problems, from playing competitive games like Chess or Soccer, to real-world defense and security scenarios between attackers and defenders. A wide variety of complex problems can be modeled and studied as adversarial games, which makes competitive coevolutionary algorithms (CoEAs) a very appealing tool.

Unfortunately, analyzing the performance of a competitive CoEA raises numerous difficulties. By definition, the fitness of an individual solution or strategy in a competitive CoEA is only measurable in relation to one or more other individuals, and there is rarely an objective benchmark to compare to, making it difficult to quantitatively measure performance, or observe how the performance of solutions changes over the course of evolution. To further complicate things, the choice of opponents to compare against matters when evaluating performance, and the wrong choice can present a misleading picture of coevolutionary progress. Many existing methods of analyzing coevolutionary progress, such as CIAO plots [2] or master tournament matrices [14], are frequently criticized through this lens, as they use individuals from the same coevolutionary run as a point of comparison [12]. This is seen as "using the training set as a test set", which can lead to measurements which are not representative of performance against typical opponents. Effective analysis methods should use opponents which are entirely independent from the coevolutionary runs being examined, in order to avoid this issue.

A frequent scenario for practitioners is the need to compare the performance of multiple competitive CoEAs with different configurations or features, either to select the best-performing configuration, or just to study differences in their behavior. One of the most common ways to carry this out is to play the evolved solutions from different coevolutionary runs against each other directly, such as in a round-robin tournament. Alternatively, solutions can be tested against a common set of benchmark opponents, though the choice of opponents will heavily influence the resulting performance measure. One advantage of comparing against a benchmark is that it provides a measure of the population's performance over time during evolution (albeit one relative to the chosen benchmark), making it easier to identify cases where the quality of solutions is stagnating, decreasing, or suddenly improving, and allowing these events to be cross-referenced with other recorded data.

However, when studying the behavior of competitive CoEAs, another scenario often emerges which is much more difficult to analyze. When considering a competitive CoEA, we often want to know how its performance is affected by the properties of the problem it's applied to. Identifying specific problem features which result in poor performance can provide insights of where a given competitive CoEA is applicable, but can also lead to an understanding of why those algorithms fail there and the development of more specialized competitive CoEAs that perform better in that domain.

We are particularly interested in the impact that the number of players has on coevolution, but this is an especially difficult feature to examine: when varying the features of a problem, such as by changing the rules or the number of players in a game, these established methods of comparison no longer work. Even a small change to the rules of a game might drastically shift the landscape of strategies. Adding a third player to a 2-player game often alters the character of the game entirely. Because of this, we can not directly test solutions from different games against each other. Which game's standard of performance could we apply? Chess and



FOGA '25, Leiden, Netherlands

ACM ISBN 979-8-4007-1859-5/2025/08
https://doi.org/10.1145/3729878.3746621

Shogi are very similar board games, but a Chess player losing to a Shogi player in a game of Shogi says little about their skill at Chess, nor the reverse. There may also be technical reasons why the genotype evolved for one game might not be compatible with the variant game at all. What about playing against a common set of benchmark opponents? The same issue arises: it is difficult to claim that a given benchmark provides an equivalent challenge in both game environments.

In this work, we apply Monte Carlo tree search (MCTS) as an adaptive benchmark opponent in several games with a particular property: the number of players in the game can be varied without changing the shape of the game tree, which we name "n-player equivalence". The purpose of this property is to minimize differences in MCTS performance caused by differing tree depths and branching factors per node, resulting in very similar search characteristics regardless of the number of players, as we show for 2, 3, and 4-player games. We discuss this property further in Section 3.1. In particular, we examine Nim and a variant of Othello.

Additionally, we demonstrate the use of MCTS to measure a population's performance over time during competitive coevolution, in situations where an appropriate problem-specific benchmark may not be known, and where randomly-generated opponents are too weak to meaningfully distinguish the performances of evolved solutions. This use case is applicable to any problems where MCTS is appropriate, and is not limited to the specially-designed problems described in this work.

2 Related Work

Coevolutionary performance has been measured in a variety of different ways since competitive CoEAs were first developed. The related techniques of CIAO plots (for "Current Individual against Ancestral Opponents") [2] and master tournament matrices [14] both aim to visualize coevolutionary progress over time by displaying the performance of evolved individuals against those in different generations of the same coevolutionary run. These plots capture the idea that, if competitive coevolution is successful, individuals from later generations should outperform individuals from earlier generations, and coevolutionary pathologies may be identified if the resulting plots are not monotonically increasing with successive generations.

Rosin and Belew [16] use a number of different analysis techniques while investigating coevolutionary techniques. One metric they measure is win rates over time per population, which can only be qualitatively analyzed. They also make use of master tournament matrices to study coevolutionary progress during individual runs, in order to visualize the impact of coevolutionary methodologies aimed at producing more consistent progress. Additionally, for the game of Nim in particular, they examine metrics based on specific domain knowledge, like plotting the rate at which individuals make sub-optimal moves per generation, and the time required per run to reach the optimal Nim genotype.

Examining the concept of coevolutionary progress, Miconi [12] distinguishes between measures of "local progress", improvement compared to individuals in the same generation; "historical progress", improvement compared to individuals in past generations; and "global progress", improvement compared to the entire search space. It is argued that these distinct concepts have been incorrectly conflated by existing coevolution research, and that analysis techniques like CIAO plots and master tournament matrices are misleading, because they only measure historical progress but are treated by researchers as if they were a measure of global progress. The author also identifies a specific example where methods which successfully promoted historical progress were actually harmful to global progress.

Miconi proposes a specific rule for coevolutionary analysis: "When applying these techniques, do not pit individuals against the opponents with which they have coevolved." This is equated with the need to avoid contaminating the training set with examples from the test set in machine learning. One methodology which obeys this rule is to perform several coevolutionary runs, and cross-test the best solutions from these separate runs against each other.

A recent example of methodology commonly used for comparing differently-configured competitive CoEAs is given by Nolfi and Pagliuca [15], who compare several methods for promoting consistent coevolutionary progress. They use master tournament matrices to visualize historical progress in their coevolutionary runs, which they suggest also provides a measurement of global progress. They also compare the best solutions evolved from each competitive CoEA against the other configurations in pairwise games, in order to test for statistically significant differences in performance.

Jaśkowski et al. [5] studied randomly-generated genotypes as opponents both during coevolution and as a benchmark to measure the quality of evolved strategies for Othello. By comparing against randomly-generates strategies, they are able to measure performance over time of several competitive CoEA methodologies, and statistically compare the resulting solutions from these methodologies. By performing a large round-robin tournament of randomly-generated strategies and measuring their relative performance, they are also able to produce "performance profiles" for their evolved solutions to examine how they fared against opponents of varying strength. Randomly-generated opponents are very appealing as a point of comparison, since the idea of global progress is typically defined as increasing performance against the set of all possible opponents (which random genotypes are sampled from).

However, their results show a substantial disparity between performance against these randomly-generated strategies and performance against other coevolved strategies. In fact, the methodology that performed the strongest against random opponents was actually one of the worst-performing against strategies produced by the other competitive CoEAs! The authors argue that this disparity was caused by the low average quality of randomly-generated opponent strategies, meaning that high performance against these strategies does not correspond well to performance against particularly strong opponents, which are rarely found at random. This problem also impacts the usefulness of their performance profiles, as they are unable to compare against strategies at very high (or very low) performance levels, even after spending extreme computational effort to test different random strategies in a round-robin tournament. As an additional problem in our use case, since randomly-generated strategies are uniformly sampled from the solution space, different variants of a given problem may result in a significantly different distribution of performance in the sampled strategies.

In a later work, Jaśkowski et al. [6] examine both randomly-generated strategies and strategies evolved specifically to be tests as part of a performance profile. Unlike in the previous implementation where the tests in a performance profile are generated in a large round-robin tournament of random genotypes, here strategies can be evolved specifically to occupy "bins" in the set of tests corresponding to a specific difficulty (defined as performance against random strategies). This is effective in many situations where randomly-generated strategies are not workable, but is described as being a very computationally intensive process which still struggles at the extreme ends of the difficulty scale. These methods allow the authors to more clearly demonstrate the previously observed behavior where some competitive CoEA configurations performed weaker against random strategies but stronger against high-difficulty opponents. Using evolved strategies as test cases poses a risk of introducing bias into the performance metric, at least in theory. Despite this, the authors found that the impact was very minor for the problems they observed, though also problem-dependent.

We believe that, for problems where it is applicable, tree search should be able to fulfill a similar use case, while overcoming some of the shortcomings of randomly-generated strategies or evolved test cases. Tree search methods should generally outperform randomly-generated strategies, but they are also parameterized by the budget given to them in terms of how many nodes to search, which makes it possible to model arbitrarily weak or arbitrarily strong opponents, even approaching the optimal strategy, given enough computational resources [1]. This parameter also allows tree search to be used to construct similar performance profiles by comparing evolved solutions to tree search with a higher or lower search budget.

3 Methodology

Our methodology consists of two separate contributions: first, we introduce the use of tree search-based opponents (and particularly MCTS-based opponents) as a technique for analyzing global progress in competitive CoEAs, by providing opponents of adjustable strength which can respond intelligently to individual evolved strategies. We emphasize that this methodology is applicable to competitive CoEAs in all settings where tree search can be effectively performed, and is not specific to the analysis of n-player games. Second, we identify a class of problem domains which allow the number of players in a game to be changed without changing the shape of the game tree, which we hypothesize will encourage consistent and comparable tree search behavior across these different problem domains. We test this hypothesis for our specific problem domains in Section 4.1.

In order for MCTS to serve as a consistent benchmark across games with different numbers of players, we need to establish that MCTS has comparable search behavior between these game variants. One of the primary factors in the performance of tree search is the branching factor of the game, the number of distinct actions that can be taken in any state. The size of a search tree as search depth increases grows exponentially with the branching factor, meaning that in games with a high branching factor it is far more difficult to search deeper into future game turns.

To minimize the impact this has on the performance of MCTS, we look at a class of games chosen such that the number of players in the game can be varied without changing the shape of the game tree, which we call "n-player equivalence" and define more formally in Section 3.1. This means that in each state, the branching factor and the depth to reach a terminal state will be identical. We predict that games sharing this property will encourage consistent performance for MCTS even when the number of players is changed. Note that this property alone does not guarantee consistent performance, as changing the number of players will change the structure of payoffs reachable from each state. We examine the impact of this in Section 4.1. For games where this property does not hold, MCTS-based analysis is still useful for analyzing coevolution within a single game, but will not produce directly comparable results across different games.

3.1 n-Player Equivalence

A *game tree* is a tree that represents the structure of a game, where vertices correspond to game states, and an edge from a state G to its child H indicates that G has an action which moves to state H. Tree search algorithms like MCTS search this space directly, and the shape of the game tree impacts the performance of the search. If we can find games which have identically-shaped game trees (that is, their game trees are isomorphic to one another), it eliminates a major source of variability in the performance of MCTS.

In our case, the goal is to identify a class of games which can be parameterized by the number of players, but where the n-player and m-player games are *game tree isomorphic* for any n and $m > 0$. We call this property *n-player equivalence*. An easy way to prove this property is to compare against the (often trivial) 1-player game in that class[1]. For every game state, we can strip out all information in the game relating to player identity (such as turn order or any "ownership" of resources) and imagine that there's only one player. If this never affects the set of available moves (including termination conditions for the game), then it demonstrates that the n-player game is game tree isomorphic to the 1-player game. Since isomorphism is transitive and symmetric, by showing that this is true for all n, it shows that the shape of the game tree is identical for any n and m. We demonstrate this for our chosen games in Section 3.3 (Nim) and Section 3.4 (Othello).

Note that identifying game tree isomorphism can sometimes fail in subtle ways when the "game tree" is not actually a tree. *Transpositions* in games occur when an identical state can be reached through two different sequences of moves. For many purposes, it makes sense to treat these identical states as a single state, turning the game tree into a graph. This is commonly done in tree search, as it prevents redundant exploration [1]. However, choosing which states to equate in this way often involves player identity! Two states which are identical in the 1-player game might be distinct in the 2-player game because they differ in turn order or resource ownership. When using an MCTS implementation that merges transpositions, the 1-player game "tree" may be significantly smaller and easier to search.

[1]For Nim and Othello, the 1-player version is primarily a tool for establishing n-player equivalence, and isn't examined in its own right.

Games with n-player equivalence typically involve players monotonically either adding or removing pieces on a board, with the game ending when the board is full or empty respectively. All impartial games (which include Nim) can be extended to an n-player equivalent class, since by definition the set of available moves in an impartial game is independent of the current player.

3.2 Monte Carlo Tree Search

Monte Carlo tree search (MCTS) is a well-studied decision-making algorithm which uses a search tree to explore the space of possible states and actions [1]. In contrast to other common tree search methods like the (depth-limited) minimax algorithm, MCTS does not need a state evaluation heuristic, and instead performs stochastic *simulations* to sample possible terminal states. This makes MCTS valuable for applications like general game-playing, where expert knowledge about the problem domain is not necessarily available [19]. However, note that for problem domains where minimax is particularly effective, it could likely be substituted into the methodology presented in this work with few other changes.

The algorithm consists of a cycle of steps: *Selection* – selecting a leaf node of the current search tree to explore from, according to a given *tree policy*, *Expansion* – expanding the search tree by picking an action to explore from the selected leaf node, *Simulation* – simulating the remainder of the game to sample a terminal state from the newly added node, according to a given *simulation policy*, and *Backpropagation* – propagating the outcome from that terminal state up through the tree to update the statistics stored in each parent node.

The commonly-used tree policy of *Upper Confidence bounds for Trees* (UCT) is designed to focus search around lines of play where each player takes actions with a high expected payoff, balancing this with some exploration of under-examined actions. In a given node, the UCT policy selects from child nodes j following:

$$\underset{j}{\operatorname{argmax}}\ \bar{X}_j + 2C_p\sqrt{\frac{2\ln n}{n_j}} \tag{1}$$

where $\bar{X}_j$ is the average payout observed from previous visits of j, n is the number of times the parent node has been visited during search, n_j is the number of times j has been visited, and C_p is an exploration parameter. UCT selects an unvisited node when one exists (treated as a score of infinity) [1].

We use a very standard implementation of MCTS with UCT tree policy and a uniform random simulation policy. For UCT, we use a value for C_p of $1/\sqrt{2}$, which is recommended as a default when payoffs are in the range $[0, 1]$ [1]. We do not use transposition tables for the reasons described in Section 3.1.

3.3 Nim

Nim is a simple game in which players take turns removing stones from one or more heaps, with the goal of being the player to remove the final stone. An action in Nim consists of selecting a heap, and a (nonzero) number of stones to remove from that heap. Some versions also reverse the win condition, making it a loss to remove the final stone (called "misère play", as opposed to "normal play"). For most of our experiments, we use normal play with 4 starting heaps of 3, 4, 5, and 4 stones respectively. Nim is an *impartial game*, a well-studied class of games in combinatorial game theory characterized by the property that the set of available actions in any state is independent of the player.

On its face, extending Nim to more than two players is simple. Since the set of available moves is independent of the current player, and since the game is won by making the last move, it would seem sufficient to have n players taking turns with no further alterations. However, with three or more players, the game can reach positions where the current player has no chance to win, but their action will still decide the game's winner. The losing player's choices become arbitrary, which makes it hard for the remaining players to predict the result of their actions, as the eventual outcome of their choices will be determined by the losing player's whims, essentially a coin flip. Worse yet, since this will be the case in any endgame for some losing player, it becomes difficult for any player to reason about the game at all – this uncertainty propagates all the way to the root of the game tree. One solution given by Li [10] is to define rankings for the players based on the turn order when the game terminates, with more recent players receiving higher ranks: if player m takes the last stone, then player $m+1$ is assigned last place, player $m+2$ is assigned second to last place, and so on wrapping around, with player $m-1$ receiving second place, and player m receiving first place. Krawec [9] uses a similar approach, defining an *alliance matrix* to describe each player's ordered preference for who should win the game. The *Standard Alliance Matrix* (SAM) is defined to produce the same ordering as Li uses, which simplifies analysis of the game. In our implementation, we convert these ranks into payoffs with a linear mapping between 0 and 1.

Krawec [9] additionally defines the *game value function* $g(G)$ for impartial games, where the game value of a given state is determined by which player can guarantee a win, numbered relative to the current player. If the game value is 0, the current player has a winning strategy. If the game value is 1, then the next player has a winning strategy, regardless of the current player's actions. This is related to the Sprague Grundy Function in two-player impartial games [4]. Under the SAM, g can be computed recursively from the game values of successor states with:

$$g(G) = \min(g(Y) + 1, \forall Y \in G) \tag{2}$$

where the game value for the end state in normal play Nim is $n-1$, as the last-moving player wins. The optimal strategy for the current player is to select an action leading to a state G that minimizes

$$g(G) + 1 \pmod{n} \tag{3}$$

To coevolve strategies for Nim, we modify the genotype representation introduced by Rosin and Belew [16]. Their representation takes advantage of the small state space of Nim, with a binary gene for each possible state encoding whether that state is desirable or not. When selecting a move, an evolved strategy picks the first move that would lead to a desirable state (or just the first move if none are marked desirable). For n-player Nim, a binary gene is insufficient to encode an optimal strategy: a state that guarantees a win for player 2 is undesirable for player 1 if player 1 has a winning strategy, but is desirable if all other actions allow player 3 to win (under the SAM). Instead, our genotype representation uses an integer in $[0, n)$ per state, where the evolved strategy picks a move that would lead to the lowest-valued state (picking the first in case

of a tie). A genotype using real values in $[0, 1]$ would also suffice for any n.

Since the state space of Nim can be searched exhaustively, an optimal genotype can be constructed by assigning the gene for every state G following Equation 3. Using this, an evolved genotype can be measured by its Manhattan distance to the optimal genotype, which provides an alternative performance measure to validate our analysis methodology against.

It is easy to show that the class of n-player Nim games are n-player equivalent. Because the set of available moves is entirely determined by the size of the heaps, and is independent of the current player or the history of moves, reducing any state to its 1-player equivalent does not change the set of moves available. The added SAM does not affect the available moves, only the payoff.

3.4 Othello

Othello is a classic board game in which players take turns placing pieces of their color on a grid. When a piece is placed, if there exists a continuous line of opponents' pieces in between the placed piece and another piece of the player's color, all those opponents' pieces are flipped to the current player's color. The game ends when the board is filled, and the player with the most pieces of their color wins. The standard version of the game is played with two players, and with the movement restriction that valid moves must capture pieces.

In order to modify this game to allow n-player equivalence, this restriction on valid moves is lifted, so that each player may play in any of the remaining spaces. Note that this does distort the gameplay somewhat, as players can now immediately play on the valuable corner and edge spaces instead of needing to build out from the center. This also significantly increases the branching factor of each state, which has computational impacts. However, since this game is only being used as an environment to study the behavior of coevolution in general, it is not necessary that it share its behavior with normal Othello.

To extend this game to more than two players, we need only specify that capture applies to lines of any opponent pieces, even if they have mixed colors. During evolution, players receive a fitness based on the fraction of the board they control at the end of the game. However, during analysis, we instead report on a payoff metric, where the winning player receives a payoff of 1, the player with the fewest pieces receives a payoff of 0, and the remaining players receive a linearly scaled payoff based on their rank order. This makes the resulting payoffs more comparable to those from Nim, and also makes the relative performance of different strategies more visible, as the fraction of the board each player controls by the end of the game tends to be fairly even. Our implementation starts with an empty board, instead of the standard starting position in Othello, in order to more easily generalize to n players. Due to runtime constraints, we use a 6×6 board instead of the standard 8×8 board.

This variant of Othello can be shown to have n-player equivalence by considering the distinctly boring game of 1-player Othello. In 1-player Othello, with our relaxed move validity, the game consists entirely of choosing an unoccupied space each turn to add a piece, and repeating until the board is full. For any n-player Othello, we can map each state to 1-player Othello by stripping away information about the owners of the pieces and the turn order. The same open spaces will be available for actions in either game, and the only impact of an action up to player-equivalence is to fill that tile as in 1-player Othello, as flipping pieces only changes their ownership and can not add or remove pieces from the board. So for any n, n-player Othello is game tree isomorphic to 1-player Othello, and therefore any two n- and m-player Othello games will have identically-shaped game trees.

We use a genotype representation for Othello introduced by Lucas [11] and improved by Jaśkowski and Szubert [7], which uses n-tuple networks in a state evaluation function to conduct a 1-ply search. Each n-tuple represents a list of board spaces, and references a lookup table based on the state of those board spaces. For example, in a 2-player game where each space can be empty, black, or white (3 options), each 4-tuple will index a lookup table with $3^4 = 81$ entries corresponding to the possible combinations of pieces in those 4 spaces. The values in these lookup tables are evolved as a linear genotype of weights. In order to select a move, an agent looks through the list of valid actions and their successor states. Each successor state is evaluated by applying the full set of n-tuples, and summing the resulting weights. Then, the action leading to the state with the highest evaluation score is chosen. n-tuple networks can also make use of the board's symmetries to reuse the same n-tuple and weights in 8 equivalent positions (the D_8 group of symmetries on the square), which reduces the parameter space. The n-tuple network we use for this work is the set of all straight 3-tuples, plus the set of all 2×2 squares. For technical reasons, we use integers in [-100, 100] to represent the weights.

The ability to use a specialized genotype taking into account the geometry of the board makes Othello a valuable contrast to Nim, where our chosen genotype encodes a policy on the game tree directly. In practice for other games, the genotypes used for most competitive CoEAs will be more similar to Othello than Nim, as it is rarely feasible to enumerate an entire game tree.

3.5 Competitive Coevolution

In order to evolve agents to play n-player Nim and Othello, we employ a 1-population competitive CoEA. Implementation-wise, this type of CoEA is similar to a standard EA, except that the evaluation function needs to sample n players from the population, and the returned fitness values are scored relative to that individual's opponents. In general, an individual needs to participate in several evaluations against a variety of opponents to receive a sufficiently representative fitness score. Also note that for coevolution, the entire population needs to be re-evaluated after every generation, as the set of opponents has changed. The parameters used in this work for coevolution are given in Table 1.

The parameters chosen were tuned by hand such that coevolution is typically successful, and has enough time to plateau in performance within the evaluation limit for most problem configurations. The choice to only compare each individual against 20 opponents may seem small given population sizes of 200 to 1,000, but we found that increasing this value much higher had diminishing returns in terms of producing more accurate fitness

	Nim	Othello
μ	500	100
λ	500	100
Generations	1,001	1,001
Evals per individual	20	20
Total evaluations	20,010,000	4,004,000
Genotype length	600	5250
Gene range	$[0, n)$	$[-100, 100]$
Mutation rate	30%	30%
Gene mutation chance	33%	33%
Mutation std. dev.	50%	10%
Evolution strategy	$(\mu + \lambda)$	$(\mu + \lambda)$
Parent selection	2-tournament	2-tournament
Survival selection	Truncation	Truncation

Table 1: Competitive CoEA parameters for Nim and Othello, used for all player counts.

values. However, this choice may be responsible for some of the noise observed in our results, and it may be valuable to sample a larger quantity of opponents per individual when analyzing global progress of coevolution. For mutation, we report the mutation rate, the chance per gene of mutating, and the standard deviation of the normally-distributed mutation size as a percentage of the gene's range.

3.6 MCTS Benchmarks

While this work is primarily focused on evaluating competitive CoEAs across different game parameters, we want to emphasize that MCTS is a useful tool for analyzing competitive CoEAs in general. Unless a problem domain has been well-studied, there are not often clear benchmarks to use when evaluating the performance of the solutions generated through competitive coevolution. Random opponents often don't make a good benchmark, as they have difficulty distinguishing between strong strategies. This can be seen in Figure 8 and Figure 10, where performance against random opponents plateaus very early in evolution, and remains horizontal even for populations which are still improving against the MCTS benchmark. Evolved strategies from other runs can be used to construct a benchmark, but during the initial implementation of a competitive CoEA, the quality of the evolved strategies may be unknown; they may be no better than random. On problems where it's applicable, MCTS can provide a useful benchmark which is adaptable to the specifics of the environment or the behavior of the opponents it's facing. MCTS has been well-studied, which makes it easier to identify problems where it will likely be effective. The applicability of MCTS as a general-purpose benchmark for competitive CoEAs does not require any of the restrictions we impose in this work regarding n-player equivalence or the behavior of MCTS over different game trees.

Fitness-over-time plots are a standard tool for analyzing the behavior of regular EAs, as they allow the identification of many useful pieces of information that might not be obvious from the final results of evolution. Many of these details concern how quickly and consistently the population is improving: the presence of a plateau where improvements stagnate, whether improvement is smooth or exhibiting punctuated equilibrium, and points in time where the quality of solutions may have decreased. These plots can also be correlated with other data collected per-generation to better characterize events on the graph, like a sudden increase or decrease in solution quality. For competitive CoEAs, fitness itself is a relative measure and can't be used to show coevolutionary progress, but performance against a benchmark like MCTS can be an effective substitute. For the purposes of comparing individuals across different games, these plots allow comparison of the behavior of competitive CoEAs without requiring that MCTS have comparable performance in these games.

An interesting feature of MCTS for this purpose is the ease with which its search quality can be tuned, by adjusting the number of iterations allowed for search, often reported by the number of *simulations*. It is known that MCTS converges to the optimal policy as the number of simulations increases, and does so relatively quickly [8]. When selecting MCTS benchmarks, we can examine a range of different simulation counts to get different perspectives on the solutions being tested. When the simulation count is too low or too high, solutions of different qualities will be hard to distinguish from one another because they will strongly outperform or be outperformed by the MCTS opponent. In our experiments, we tested powers of two for simulation counts until we found values where MCTS performed similarly to the evolved solutions, to highlight differences in performance between individuals. It may also be valuable to use multiple simulation counts for this comparison, as there are cases where two different strategies may be ranked differently in their performance against benchmarks of different skill levels [5].

4 Experiments

We describe three sets of experiments performed to investigate the behavior of MCTS as a methodology for measuring global progress in coevolution. First, we investigate our hypothesis that our choice of n-player equivalent games should result in MCTS search trees with similar characteristics, allowing our results to be compared across different numbers of players. Then, we apply our technique to Nim and Othello for 2-, 3-, and 4-player games, and compare to the payoffs observed against randomly-generated opponent genotypes. For Nim, we additionally visualize the distance in configuration space of the evolved genotypes to the theoretically optimal strategy.

4.1 Analysis of Monte Carlo Tree Search

In order for MCTS to be usable as a point of comparison between games with different numbers of players, we need MCTS to have similar characteristics when searching the game trees for each of those game variants. While our choice of games ensures that the full game tree has an identical shape regardless of the player count, it could be the case that different circumstances cause MCTS to perform a wider or deeper search with its simulation budget. The UCT tree policy specifically balances exploration of under-visited branches, and exploitation of branches which show promising payoffs.

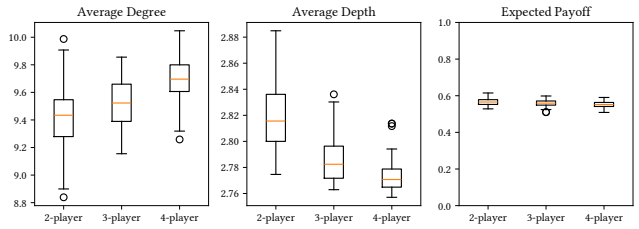


Figure 1: MCTS tree properties for 2, 3, and 4-player Nim, averaged over 100 search trees from the initial state. Left: the average degree of nodes in the search trees. Middle: the average depth of nodes in the search trees. Right: the payoff of the selected action as observed during search.

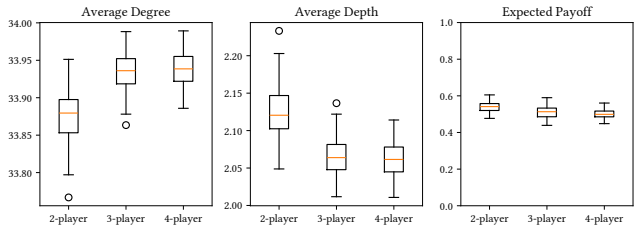


Figure 2: MCTS tree properties for 2, 3, and 4-player Othello, averaged over 100 search trees from the initial state. Left: the average degree of nodes in the search trees. Middle: the average depth of nodes in the search trees. Right: the payoff of the selected action as observed during search.

To explore this, we ran MCTS from the initial position of each game 100 times, with a budget of 1024 simulations. From the resulting trees, we measured three properties:

(1) The average depth of nodes in the tree
(2) The average degree of nodes in the tree
(3) The average payoff of the action chosen by MCTS

If these games are equally difficult for MCTS for different numbers of players, we would expect the search trees to have a similar shape, and to find moves which predict a similar payoff.

The box plots in Figure 1 and Figure 2 show remarkably consistent values on all three metrics between different numbers of players, on both Nim and Othello. The payoffs found during search were particularly consistent, suggesting that MCTS is having comparable difficulty regardless of the number of players. As we will discuss in Section 4.1.1, these metrics can vary radically when comparing games with substantially different search difficulty.

4.1.1 Examining a Counterexample. To strengthen this finding, we also examine a scenario where two variants of a game with identically-shaped search trees *do* have significantly different search difficulty. For this purpose, we compare against misère Nim. For a given starting position, the state space for misère Nim is identical to normal play Nim, only altering the payoffs. Because of this, we can construct a starting state where the first player has a winning strategy in one variant, but a significant disadvantage in the other. As a result, we can construct an example of what we should expect to see in a case where two variants of a game do not have comparable difficulty.

As it turns out, it is surprisingly difficult to find such a nontrivial set of starting heaps with these qualities. By exhaustive search, the most complex examples we found for n players, with $n > 2$, were of the form $[1, 1, ..., 1, x]$, containing a number of 1-heaps equal to an integer multiple of n, and one heap with any nonzero number of stones. In normal play Nim, the winning strategy for the starting player is to take the entire x-heap, so that the remaining 1-heaps must be removed one at a time, and that they will always remove the last heap of every group of n. In misère Nim, the first player has three choices:

(1) If they take one of the 1-heaps, player 2 can take the entire x-heap, and the number of remaining 1-heaps ensures that player 1 will take the final stone, resulting in player 2 winning.
(2) If they take the entire x-heap, the number of remaining 1-heaps ensures that they will take the final stone, resulting in player 2 winning.
(3) If they take part of the x-heap, player 2 has an identical choice. Player 2 either has a winning strategy, or they can force player 3 to win using one of the above two actions, which they would prefer over any third option by the SAM. Player 1 would prefer for player 2 to win over player 3, so this option is strictly worse than the other two.

Therefore, the starting player does not have a winning strategy.

We repeated the methodology described above, running MCTS on 3-player Nim with the initial heap state of $[1, 1, 1, 1, 1, 1, 8]$, under both normal and misère rules. This specific initial state was chosen because it has a similar number of possible states to the one used during our coevolutionary experiments (576 vs. 600, respectively). Unlike in the previous experiments, the resulting search trees between these two games had radically different results for all three metrics, as shown in Figure 3. For normal play Nim, MCTS searched deeper and with a lower branching factor, as it had an easier time identifying a good strategy and focusing its search in that direction. Since we do not observe this behavior between Nim or Othello variants with different player numbers, this supports our claim that MCTS performs comparably on these problems regardless of player count.

4.2 Nim

Figure 4 plots, for 2, 3, and 4 players respectively, the mean performance of the top 3 individuals per generation against an MCTS opponent, averaged over 10 runs. The MCTS opponent is given a budget of 1024 simulations; this value is chosen because it provides good contrast between high and low-performing individuals for this game. Due to the expense of running these games, we sample every 10th generation. We use the top 3 individuals instead of the single top individual because "top" here is measured by local fitness during evolution, and it is often the case that this measurement misidentifies the globally best individuals during that generation. This manifests in the plot as a sharply-oscillating line.

2-player Nim tended to produce steadier progress throughout coevolution, while progress in 3-player Nim typically stagnated

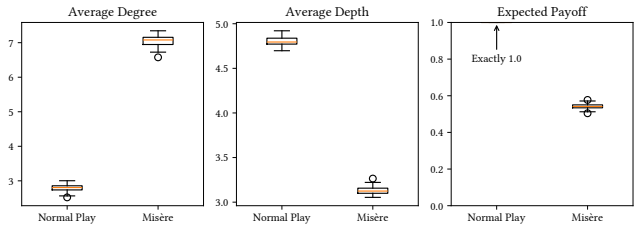


Figure 3: MCTS tree properties for normal play and misère Nim with a specially-chosen starting state, averaged over 100 search trees from the initial state. Left: the average degree of nodes in the search trees. Middle: the average depth of nodes in the search trees. Right: the payoff of the selected action as observed during search. Payoffs for normal play are not visible, as trees found a payoff of 1.0 in every run.

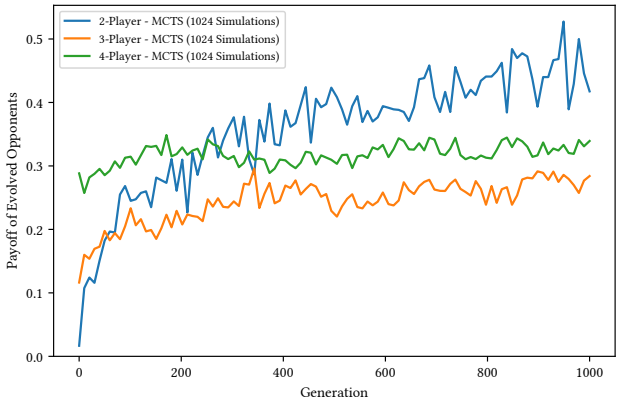


Figure 4: 2-player, 3-player, and 4-player Nim – Payoff against MCTS for the top 3 individuals per generation, averaged over 10 runs.

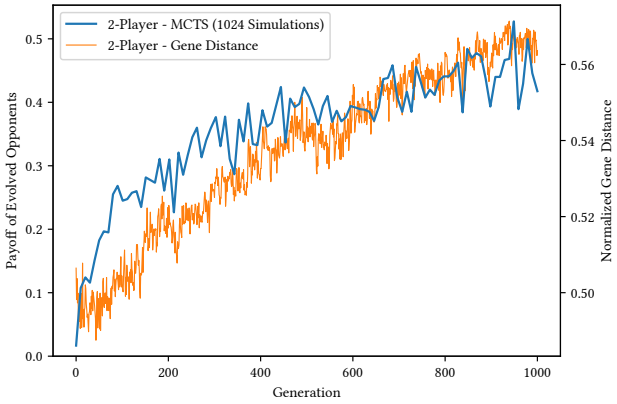


Figure 5: 2-player Nim – Payoff against MCTS and distance to optimal genes for the top 3 individuals per generation, averaged over 10 runs.

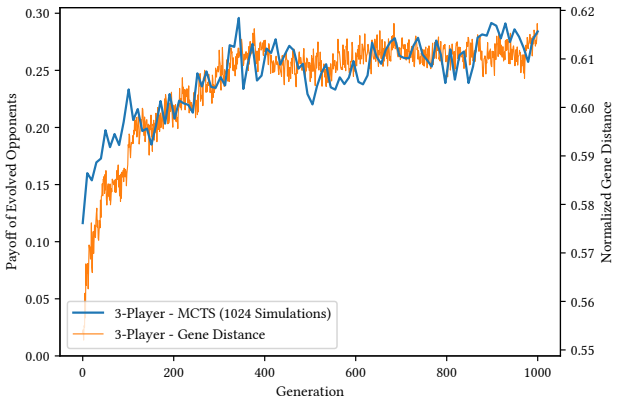


Figure 6: 3-player Nim – Payoff against MCTS and distance to optimal genes for the top 3 individuals per generation, averaged over 10 runs.

part-way into coevolution. This effect was even more pronounced for 4-player Nim, which struggled to make progress at all. On individual coevolutionary runs for 4-player Nim, fitness often declined substantially at times, suggesting that coevolutionary pressures were pushing the population away from global improvements. This may indicate that coevolutionary pathologies like cycling become more common here as the player count increases.

Several of the final strategies in 2-player Nim were able to achieve fitness values of 0.5 or higher against the MCTS opponent, indicating that they were able to win more than half of their games against MCTS with 1024 simulations. None of the strategies generated in 3-player or 4-player Nim were able to reach a similar level of performance.

Note that the fitness function used here gives partial wins to players based on their "preference" for the winner, following the SAM. This means that in 3- and 4-player Nim, the largely random strategies at the start of coevolution can achieve a nonzero fitness against the MCTS opponents.

4.2.1 Comparison to Optimal Genotypes. As discussed in Section 3.3, Nim is a simple enough game that we can construct an optimal genotype by assigning genes following Equation 3. This gives us an alternative method of measuring the global performance of individuals in the population; we can use this to validate the results above.

We use the Manhattan distance between an evolved genotype and this optimal genotype as our metric, as it measures how far away each individual gene is from its optimal value. Each suboptimal gene corresponds to a game state in which the player has incorrectly predicted the winner for optimal play, which can lead to suboptimal moves that decrease their final payoff. The largest possible deviations in a gene correspond to game-losing moves rated as game-winning, and vice-versa. Note that this is not a perfect measure of optimality, as some states occur more frequently, or are more critical than others to score accurately. We normalize this

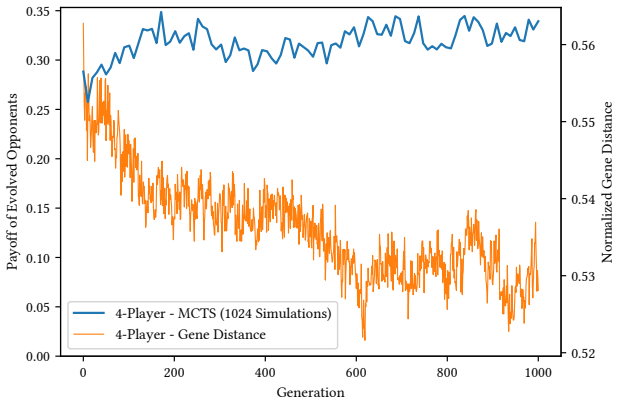


Figure 7: 4-player Nim – Payoff against MCTS and distance to optimal genes for the top 3 individuals per generation, averaged over 10 runs.

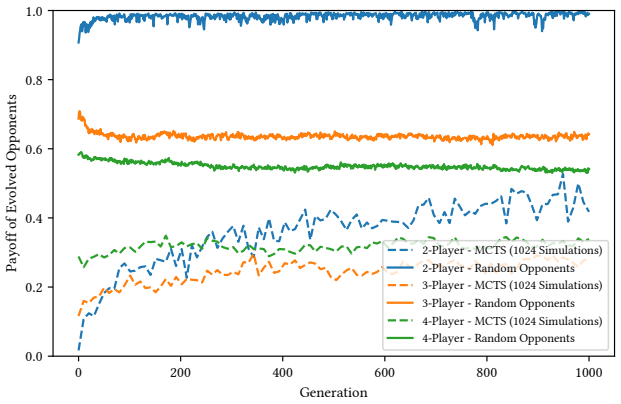


Figure 8: 2-player, 3-player, and 4-player Nim – Payoff against random genotypes for the top 3 individuals per generation, averaged over 10 runs (solid lines). Payoff against MCTS in corresponding dashed lines.

distance value by dividing it by the number of genes and the range of gene values, to produce a final value between 0 and 1.

Figure 5, Figure 6, and Figure 7, plot (in orange) this gene distance metric over time, again averaged over the top 3 individuals for the same 10 runs of coevolution. The payoff against MCTS is reproduced from Figure 4 for comparison. For individual experiments, and particularly for 2- and 3-player Nim, the results match up fairly closely with the results produced through comparison to MCTS agents. This suggests that, at least for Nim, MCTS-based analysis does provide a good estimate of global performance.

For 4-player Nim, the overall trend unexpectedly showed that strategies were moving *away* from the optimal strategy. Despite this, these two metrics remain correlated in terms of their shape in some places. This is likely a result of our choice to weight all genes evenly for this metric. Some changes to the genotype correlate very strongly with changes to fitness, while others have little impact even if they are moving away from the optimal strategy.

4.2.2 Random Opponents. Comparison against randomly-sampled opponents is an existing analysis method aimed at obtaining an unbiased representation of global progress over time [5]. In order to compare this methodology with our MCTS-based approach, we compare the top 3 individuals from each generation with a set of 10,000 randomly-generated Nim genotypes. Recall that our genotype for n-player Nim is a list of integers in $[0, n)$ representing the desirability of each state. We sample these uniform-randomly to create our opponents. We measure the performance of an individual as their average payoff in each of the n positions in the turn order against 10,000 sets of random opponents – these sets are created by randomly permuting the list of random genotypes for each position in the turn order, guaranteeing that each individual will face each random genotype once in every position. The same permutations are used for all individuals being measured. We do this instead of a round-robin tournament so that the number of games can be controlled; it would otherwise grow exponentially as n increases.

Results analyzing the same coevolutionary runs as above against randomly-generated opponents are presented in Figure 8, with the corresponding MCTS results shown as dashed lines. Note that for 2- and 3-player Nim, the evolved agents strongly outperform random genotypes, making it difficult to see a change in performance over time, even when progress is visible against MCTS. For 2-player Nim, payoff against the random genotypes is seen to briefly increase at the start of each run, but progress is no longer visible once the evolved strategies become too strong. The results for 4-player Nim are inconclusive, since both analysis methods suggest that there was little to no improvement in performance in these runs.

For 3-player Nim, an unusual phenomenon occurs: even though both MCTS and the distance to the optimal genotype indicate that the performance of evolved solutions is increasing over time (Figure 6), performance against random solutions actually decreases from the start of evolution, and then remains stagnant even while the other two metrics continue to improve. It is possible that this is caused by intransitive relationships in the strategy space for n-player Nim, where the population can improve its payoff against high-skill strategies, while simultaneously becoming easier for low-skill strategies to exploit. A similar behavior was seen by Jaśkowski et al. [5], who found that Othello strategies from one set of coevolutionary runs which performed the best against randomly-generated opponents actually performed relatively poorly when playing directly against strategies from other runs of coevolution.

4.3 Othello

Figure 9 plots the mean performance of the top 3 individuals per generation against an MCTS opponent, averaged over 5 runs. The MCTS opponent is given a budget of 1024 simulations. Again, we sample every 10th generation for performance reasons.

For 2- and 3-player Othello, we see similar results to Nim: coevolution for 2-player Othello produces global progress for most of the run length, while in 3-player Othello it tends to stagnate early into its runs. 4-player Othello gets similar results to 3-player Othello, with much more consistent coevolutionary progress than

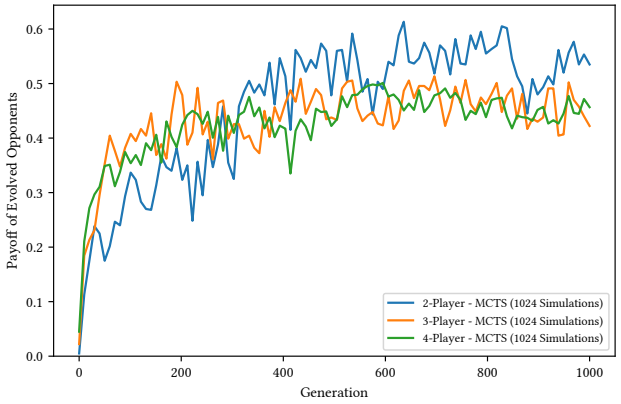


Figure 9: 2-player, 3-player, and 4-player Othello – Payoff against MCTS for the top 3 individuals per generation, averaged over 10 runs.

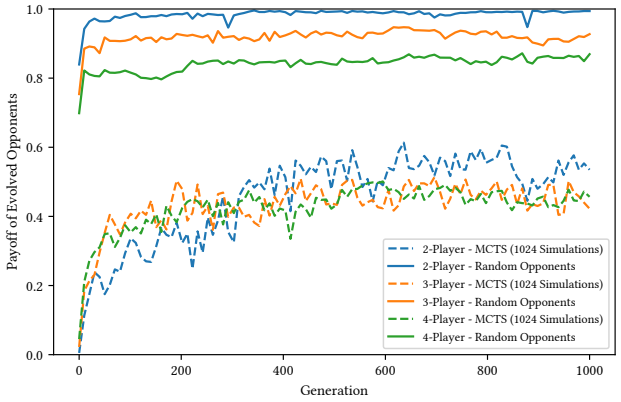


Figure 10: 2-player, 3-player, and 4-player Othello – Payoff against random genotypes for the top 3 individuals per generation, averaged over 10 runs (solid lines). Payoff against MCTS in corresponding dashed lines.

for 4-player Nim. As with Nim, a fitness of 0.5 indicates that a strategy is achieving an identical average payoff to its opponents. The resulting strategies for 2-player Othello outperform MCTS with 1024 simulations, while for 3- and 4-player Othello they exhibit weaker performance than MCTS with this configuration.

4.3.1 Random Opponents. As with Nim, we compare these Othello agents to randomly-generated genotypes, using weights randomly sampled in [-100, 100]. As Othello runs slower than Nim, we compare against a collection of only 1,000 randomly generated individuals, and only test every tenth generation. Note that, while we are basing our comparison to random opponents on previous work by Jaśkowski et al. [5], which also used Othello as a test case, that work deliberately chose a very simple strategy representation which evaluates faster but produces weaker strategies, so we can not compare our results directly against theirs. We initially attempted to use the same representation, but found that it performed very poorly against MCTS under our modified Othello rules.

Like with Nim, the randomly-generated genotypes perform too poorly against evolved genotypes to provide much useful feedback compared to MCTS. The very beginning of each run does show a brief improvement over time, as would be expected since the initial population is itself a collection of random genotypes. Keep in mind that for both Nim and Othello, the relative payoffs observed against random genotypes between different player counts is probably not indicative of the relative performance of coevolution on these games, since modifying the behavior of the game affects the distribution of strategy quality across the space of random genotypes.

5 Discussion

The n-player equivalent variant of Othello we investigated for this work was very successful in its compatibility with coevolution and MCTS with different player counts. Coevolution was able to develop effective strategies in all of our experiments, and MCTS had very consistent performance regardless of the number of players. Some of the changes necessary for this variant did have undesirable side effects, however. Ensuring that the game was game tree isomorphic for any number of players required us to greatly increase the branching factor, which made MCTS less effective overall and also slowed down evaluations for evolved strategies, as they need to evaluate their n-tuple network against every action.

Further, by relaxing restrictions on move location, the overall game strategy is heavily altered from normal Othello. Normal Othello play starts in the center of the board and moves outward, as pieces can normally only be placed adjacent to existing pieces (and only when it would enable a capture). A major part of the game's strategy involves trying to claim the edge and corner spaces, which are respectively more difficult or impossible to capture. Instead, with these restrictions removed, the evolved strategies invariably play their first pieces in the corners, and then primarily play on the edges until those spaces are filled. The result is that large parts of the board quickly become completely untouchable, resulting in a simpler game overall. Even so, the game in this form appears to accommodate enough strategy for interesting coevolutionary behavior, and the n-player equivalence property makes it appealing for future research on n-player games.

In retrospect, it may be worth examining an alternate implementation of n-player equivalent Othello which is more measured in its relaxing of move restrictions. If we require that moves must take place adjacent to an already-occupied space, the set of available actions is still not dependent on the player identity of any pieces, but behaves closer to standard Othello and makes it more difficult to claim the edge and corner spaces. The first move would need to be restricted to one of the spaces in the center of the board, similar to the rules for Reversi (which Othello is a variant of).

2-player Nim is a classic game both in game theory and competitive coevolution for a reason, as the game is very easy to analyze and results in very consistent coevolution. However, while n-player Nim remains conveniently easy to analyze, it seems to become much more difficult to play with higher numbers of players. Strategy in 2-player Nim revolves around identifying which states will

allow you or your opponent to guarantee a win. However, when $n > 2$, it becomes necessary to rely on your opponents making rational moves in order for any strategy to be reliable. MCTS seems to have no additional difficulty with the search space as the number of players increases, but it searches with the built-in assumption that other players will behave the same way that it does. In theory, imposing a fixed alliance matrix makes the game predictable, but that is only if it provides a strong enough evolutionary pressure to push the population towards optimal strategies. It is very possible that this never successfully occurred in our experiments: the results for 4-player Nim (Figure 7) showed that the population instead moved away from the optimal strategy over time, and it is not clear whether this is just due to random genetic drift among genes that do not significantly contribute to the strategy, or if there was actual evolutionary pressure in the wrong direction. The exact reasons for this would be worth studying in more detail. Perhaps it would be feasible to determine, in any generation, what the optimal strategy would be *relative to the current population.*

5.1 Future Work

We only tested a very standard implementation of MCTS, but there is a wide field of improvements and variations to the technique that can improve its performance or make it applicable to a wider variety of scenarios [17], like imperfect-information games [3] or games with a large or continuous action space [18]. Using more advanced techniques would allow this style of analysis to be extended to a much wider range of problems. Additionally, while we chose in this work to use MCTS in both problem domains for consistency, minimax search would also be a good choice for Othello and a number of similar games. For problems where a good state evaluation heuristic exists for intermediate states, minimax can sometimes outperform MCTS [1].

Performance profiles [5, 6] are an analysis technique which we did not explore but which would be a very good fit for tree search. The strength of MCTS can be adjusted by varying the number of simulations allowed, and minimax can be controlled with an equivalent metric. Performance profiles with MCTS would be a particularly useful tool for examining the effect of different player counts and the impact that has on coevolution, since the conditions that made MCTS comparable between player counts would still apply. We observed for some player counts on Nim that performance against weaker random search players could decrease over the course of a run while performance was increasing against MCTS. This would suggest that Nim for 2, 3, and 4 players might produce different curves on a performance profile.

Our results support our hypothesis that ensuring n-player equivalence – that the shape of the game tree remains the same across changes in player count – encourages consistent performance of MCTS opponents across different n. However, testing this for two games is not enough to show that it holds in general. Player-equivalence between games is certainly not a sufficient condition to guarantee consistent performance, as Section 4.1.1 demonstrates. It would be valuable to develop a larger collection of games with different characteristics to ensure that our analysis is not as heavily affected by the idiosyncrasies of individual games. It would also be useful to develop more robust metrics to determine when tree search is behaving comparably in different scenarios, rather than just comparing general tree metrics. A broad analysis of the relevant behavior of MCTS on random, synthetic game trees might be a good approach for this analysis, as the study of synthetic game trees has had success in exploring other behavioral characteristics of MCTS [13].

6 Conclusion

In this work, we examined the use of MCTS as a benchmark for solutions in competitive coevolution, and demonstrated its use to analyze the performance of competitive CoEA solutions. One of the primary motivations of this technique is to analyze how the behavior of competitive CoEAs differs when changing the properties of the evaluation function, particularly the number of players in a competitive game. Typically, this analysis is very difficult, as solutions for two different problems are typically not comparable. To this end, we introduced the concept of "n-player equivalent" classes of games, games which can be extended to an arbitrary number of players while maintaining identically-shaped game trees. We introduced an n-player Othello variant and extended a standard genotype for Nim to n-players, and showed that these classes of variant games were n-player equivalent. Since differences in the game tree structure are a major contributor to differences in the performance of tree search, we hypothesized that by ensuring isomorphism in game trees across a class, we could minimize differences in the behavior of MCTS.

In order to evaluate this hypothesis, we examined the search trees generated by MCTS for different n on both Nim and Othello. By comparing the average degrees and average depths of nodes in these search trees, along with the expected payoffs from MCTS, we found that MCTS exhibits extremely similar behavior across different n for Nim and especially Othello. To contrast this, we constructed an example where we would expect MCTS to perform very differently across two different games with isomorphic search trees, comparing normal-play and misère Nim with a starting condition that makes normal-play Nim much easier to search than the misère version. Our results show that in this case where a major difference in MCTS behavior exists, these same metrics on the resulting search trees produce an obvious difference between the games, suggesting that the observed similarity in metrics across n-player equivalent games supports our hypothesis of actual behavioral similarity.

As a benchmark for analyzing coevolution, MCTS performs very well overall, providing a useful measurement of global progress that can be more flexible than comparing to randomly-generated opponents. We were able to observe some configurations of coevolution which consistently produced strong coevolutionary progress versus MCTS, and others which performed poorly, while comparison to random genotypes did not clearly distinguish them. If a practitioner were in the same situation without access to this methodology, it might be very difficult for them to ever notice that, for example, their competitive CoEA for 4-player Nim was only producing marginal improvements by the end of coevolution. In the case of Nim they may be able to analyze the resulting strategies from existing game theoretic knowledge, but that would be much harder for Othello, and might be completely impractical for an entirely novel problem domain. We subjectively found it extremely helpful to have

this tool available while tuning the competitive CoEAs and game scenarios described throughout this work. Not every problem is amenable to tree search, but for those that are, tree search provides a very straightforward benchmark.

This does raise the question: if your problem domain is suitable for using tree search, why use competitive coevolution at all? If tree search was not competitive with evolved strategies from the end of a coevolutionary run, it would not be usable as a benchmark. Perhaps it would be better to let tree search solve the problem on its own! There are three primary reasons why it is still useful to study coevolution in domains where tree search can outperform it:

First, tree search is very computationally intensive. For Othello, in producing our plots we found that MCTS took around a thousand times longer to run per-game than our evolved strategies (about 5/second versus 5000/second), despite both players getting relatively even payoffs. The evolved strategy does take many hours to create, but once it is done, it can be reused indefinitely for very little cost. For problems where a solution needs to be used repeatedly, it can be worth spending the time to fine-tune a competitive CoEA in order to get the best quality of output.

Second, when studying a simplified problem, it may be the case that design choices for a competitive CoEA which perform well in that simple case also perform well in a more complicated version of the problem. Perhaps the larger problem has a high branching factor, or a continuous state space, or imperfect information, all of which can be a hindrance to tree search. If you have reason to believe that a competitive CoEA that performs well on the simple problem would also perform well on the complex problem, it makes sense to evaluate your design decisions with tree search so you have a robust understanding of what is effective for the broader class of problems.

Finally, for researchers studying competitive coevolution itself, having a clear method for measuring global progress in coevolution is invaluable, even if it is limited to relatively simple classes of problems. In this case, it does not matter that another method solves the problem better, because the goal is really to develop generalizable knowledge about how competitive coevolution works.

References

[1] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 1 (March 2012), 1–43. doi:10.1109/TCIAIG.2012.2186810

[2] Dave Cliff and Geoffrey F. Miller. 1995. Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In *Advances in Artificial Life*, Federico Morán, Alvaro Moreno, Juan Julián Merelo, and Pablo Chacón (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 200–218.

[3] Peter I. Cowling, Edward J. Powley, and Daniel Whitehouse. 2012. Information Set Monte Carlo Tree Search. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 2 (June 2012), 120–143. doi:10.1109/TCIAIG.2012.2200894

[4] Patrick M Grundy. 1939. Mathematics and games. *Eureka* 2 (1939), 6–8.

[5] Wojciech Jaśkowski, Paweł Liskowski, Marcin Szubert, and Krzysztof Krawiec. 2013. Improving coevolution by random sampling. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation* (Amsterdam, The Netherlands) *(GECCO '13)*. Association for Computing Machinery, New York, NY, USA, 1141–1148. doi:10.1145/2463372.2463512

[6] Wojciech Jaśkowski, Paweł Liskowski, Marcin Szubert, and Krzysztof Krawiec. 2016. The performance profile: A multi–criteria performance evaluation method for test–based problems. *International Journal of Applied Mathematics and Computer Science* 26, 1 (March 2016), 215–229. doi:10.1515/amcs-2016-0015

[7] Wojciech Jaśkowski and Marcin Szubert. 2016. Coevolutionary CMA-ES for Knowledge-Free Learning of Game Position Evaluation. *IEEE Transactions on Computational Intelligence and AI in Games* 8, 4 (Dec 2016), 389–401. doi:10.1109/TCIAIG.2015.2464711

[8] Levente Kocsis, Csaba Szepesvári, and Jan Willemson. 2006. Improved monte-carlo search. *Univ. Tartu, Estonia, Tech. Rep* 1 (2006), 1–22.

[9] Walter O. Krawec. 2012. Analyzing n-player impartial games. *International Journal of Game Theory* 41, 2 (01 May 2012), 345–367. doi:10.1007/s00182-011-0289-3

[10] Shuo-Yen Robert Li. 1978. N-person Nim and n-person Moore's Games. *International Journal of Game Theory* 7, 1 (01 Mar 1978), 31–36. doi:10.1007/BF01763118

[11] Simon M. Lucas. 2008. Learning to Play Othello with N-Tuple Systems. *Australian Journal of Intelligent Information Processing* 4 (2008), 1–20. https://repository.essex.ac.uk/3820/

[12] Thomas Miconi. 2009. Why Coevolution Doesn't "Work": Superiority and Progress in Coevolution. In *Proceedings of the 12th European Conference on Genetic Programming* (Tübingen, Germany) *(EuroGP '09)*. Springer-Verlag, Berlin, Heidelberg, 49–60. doi:10.1007/978-3-642-01181-8_5

[13] Khoi P. N. Nguyen and Raghuram Ramanujan. 2024. Lookahead Pathology in Monte-Carlo Tree Search. *Proceedings of the International Conference on Automated Planning and Scheduling* 34, 1 (May 2024), 414–422. doi:10.1609/icaps.v34i1.31501

[14] Stefano Nolfi and Dario Floreano. 1998. Coevolving Predator and Prey Robots: Do "Arms Races" Arise in Artificial Evolution? *Artificial Life* 4, 4 (10 1998), 311–335. doi:10.1162/106454698568620

[15] Stefano Nolfi and Paolo Pagliuca. 2025. Global progress in competitive co-evolution: a systematic comparison of alternative methods. *Frontiers in Robotics and AI* 11 (Jan. 2025). doi:10.3389/frobt.2024.1470886

[16] Christopher D. Rosin and Richard K. Belew. 1997. New Methods for Competitive Coevolution. *Evolutionary Computation* 5, 1 (03 1997), 1–29. doi:10.1162/evco.1997.5.1.1

[17] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. 2023. Monte Carlo Tree Search: a review of recent modifications and applications. *Artificial Intelligence Review* 56, 3 (01 Mar 2023), 2497–2562. doi:10.1007/s10462-022-10228-y

[18] Timothy Yee, Viliam Lisy, and Michael Bowling. 2016. Monte Carlo tree search in continuous action spaces with execution uncertainty. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (New York, New York, USA) *(IJCAI'16)*. AAAI Press, New York, New York, USA, 690–696.

[19] Maciej Świechowski, HyunSoo Park, Jacek Mańdziuk, and Kyung-Joong Kim. 2015. Recent Advances in General Game Playing. *The Scientific World Journal* 2015, 1 (2015), 986262. doi:10.1155/2015/986262