Proceedings of the Fourteenth Artificial Intelligence and
Interactive Digital Entertainment Conference (AIIDE 2018)

Evolutionary MCTS with Flexible Search Horizon

Hendrik Baier, Peter I. Cowling
Digital Creativity Labs
University of York
hendrik.baier@gmail.com, peter.cowling@york.ac.uk

Abstract

In *turn-based multi-action adversarial games* each player turn consists of several atomic actions, resulting in an extremely high branching factor. Many strategy board, card, and video games fall into this category, which is currently best played by Evolutionary MCTS (EMCTS) – searching a tree with nodes representing action sequences as genomes, and edges representing mutations of those genomes. However, regular EMCTS is unable to search beyond the current player's turn, leading to strategic short-sightedness. In this paper, we extend EMCTS to search to any given search depth beyond the current turn, using simple models of its own and the opponent's behavior. Experiments on the game *Hero Academy* show that this *Flexible-Horizon EMCTS* (FH-EMCTS) convincingly outperforms several baselines including regular EMCTS, Online Evolutionary Planning (OEP), and vanilla MCTS, at all tested numbers of atomic actions per turn. Additionally, the separate contributions of the behavior models and the flexible search horizon are analyzed.

1 Introduction

In order to play adversarial games, computer programs typically use a search algorithm which aims at desirable future game states, such as those that maximize e.g. a heuristic evaluation function. *Monte Carlo Tree Search* (MCTS) (Kocsis and Szepesvári 2006; Coulom 2007) is a search framework which has been successfully applied to a variety of board games with branching factors of up to a few hundred (Silver et al. 2017), as well as many card games, video games, and non-game domains (Browne et al. 2012).

However, in *turn-based multi-action adversarial games* each turn consists of a sequence of atomic actions, instead of just a single action – which can lead to much higher branching factors. Board games such as Arimaa and Risk fall into this category, just like mobile games such as Battle of Polytopia (Midjiwan AB 2016), and PC games such as Civilization (Firaxis Games 2016a), XCOM (Firaxis Games 2016b), Might & Magic Heroes (Limbic Entertainment 2015), and Into the Breach (Subset Games 2018). If each player turn for example consists of moving nine units with ten available actions each, the resulting branching factor is 10^9. This complexity is too high for vanilla MCTS, even using various enhancements for reducing the effective branching factor. It can be a challenging search problem in such domains to just find a good action sequence for a single turn, even ignoring the future turns.

One possible approach is searching a tree in which each edge represents an atomic action instead of a complete turn. However, MCTS can often not search these trees deeply enough, and optimizes earlier actions too much compared to later actions. If MCTS searches action by action (Schadd et al. 2012), it still suffers from the problem that search decisions on earlier actions can influence later actions, but not vice versa. Justesen et al. therefore proposed a different, tree-less approach: *Online Evolutionary Planning* (OEP), an evolutionary algorithm that treats atomic actions as genes and complete turns as genomes (Justesen 2015; Justesen et al. 2017). OEP can optimize each action equally and simultaneously as it searches over the space of possible next turns. In recent work, we proposed a hybrid approach called *Evolutionary MCTS* (EMCTS), combining some of the features of MCTS and evolutionary algorithms (Baier and Cowling 2018). It searches a tree with nodes representing action sequences as genomes, and edges representing mutations of those genomes. EMCTS therefore explores the mutation space of evolutionary algorithms in a systematic, best-first manner, providing evolution with lookahead search.

EMCTS is the current state of the art in multi-action adversarial games. However, just like OEP it is still unable to search beyond the current player's turn, often leading to myopic and greedy behavior. In this paper, we propose a simple but effective way of extending the search of EMCTS to any desired depth beyond the current turn, naming the result *Flexible Horizon EMCTS* (FH-EMCTS).

As in prior work, we use the game *Hero Academy* as our testbed. We compare FH-EMCTS to vanilla EMCTS, OEP, and three other baseline search algorithms including two vanilla MCTS variants specifically designed for Hero Academy, at different numbers of actions per turn. We additionally analyze the separate effects of two integral parts of FH-EMCTS.

This paper begins with a brief review of relevant related work in Section 2. Section 3 describes our testbed, Hero Academy, outlines the baseline algorithms we are comparing to, and introduces FH-MCTS. Section 4 presents our experimental setup and results, and Section 5 gives our conclusions and suggests future work.

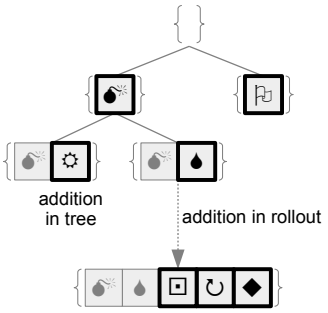

Figure 1: Tree structure of vanilla MCTS and its variants. Nodes represent partial action sequences, or the states resulting from them. Edges represent the addition of an atomic action to an action sequence, or the application of an atomic action to a state. After each node expansion, a rollout is performed for evaluation. (We use symbols to represent different atomic actions. Adapted from (Baier and Cowling 2018).)

2 Background and Related Work

This section reviews MCTS, and outlines previous approaches specifically for playing turn-based multi-action adversarial games: OEP and EMCTS.

2.1 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) (Kocsis and Szepesvári 2006; Coulom 2007) is a best-first tree search algorithm. The algorithm typically constructs a search tree with nodes representing game states, and edges representing actions leading from one state to another. In a deterministic game, this can also be seen as a tree in which nodes represent the list of actions that have been applied from the root state to reach their respective state. MCTS begins its search at a root node corresponding to the current game state. It then repeats the following four-phase loop until computation time runs out:

1. In the selection phase, a *selection policy* is used to traverse the tree until an unexpanded action is chosen.

2. In the expansion phase, the unexpanded action and a node representing its successor state are added to the tree.

3. In the rollout phase, a *rollout policy* is used to play out (part of) the remaining part of the simulated game, starting from the state represented by the newly added node.

4. In the backpropagation phase finally, the value estimates of all states traversed during the simulation are updated with the result of the finished game.

Several MCTS variants and enhancements have been proposed over time for increasingly complex games. We are using two specifically adapted variants of MCTS as baselines in our experiments, described in Subsection 3.2. They search a game tree as shown in Figure 1, in which each edge represents an additional action for the state under consideration.

2.2 Online Evolutionary Planning

Evolutionary algorithms (EAs) are a class of optimization algorithms inspired by natural selection that has been used extensively for evolving and training AI agents for games (Lucas and Kendall 2006; Risi and Togelius 2017). In the classic, *offline* evolutionary approach, an AI's parameters are evolved over many games, using its performance at playing the game as a fitness function (Cole, Louis, and Miles 2004; Chaslot et al. 2008; Alhejali and Lucas 2013). *Online* evolution is a newer approach, in which evolutionary algorithms are applied during gameplay – for example for evolving the next move(s) (Perez-Liebana et al. 2013; Gaina et al. 2017).

Online Evolutionary Planning (OEP) (Justesen 2015; Justesen et al. 2017) is a recent online evolutionary approach that can be applied to multi-action adversarial games. It optimizes the action sequence of the current turn, without looking ahead to future turns of the player or the opponent.

OEP begins its search by creating an initial population of genomes. Each genome represents a complete turn, a fixed-length sequence of actions. These actions can be sampled randomly, or another *initialization strategy* ϕ can be used. The population is then improved from generation to generation until computation time runs out. Each generation consists of the following four phases:

1. All genomes are translated to their phenotypes, the game states resulting from applying their action sequence to the current game state. Their fitness is then evaluated with the help of a static heuristic evaluation function.

2. A number of genomes with the lowest fitness is removed from the population.

3. The surviving genomes are each paired with a randomly chosen different genome, and create an offspring through uniform crossover (Syswerda 1989). If this leads to an illegal action in the offspring, it is repaired by a *repair strategy* ψ – in the simplest case by picking random legal actions.

4. A proportion of the offspring undergoes mutation. One randomly chosen action of the sequence is changed to another action randomly chosen from all legal actions. If this leads to illegal actions later in the sequence, they are replaced using the repair strategy ψ as well.

When the time budget is exhausted, OEP returns the action sequence represented by the current best genome, so it can be executed action by action. In the words of Wang et al. "the action selection problem is seen as an optimization problem rather than a planning problem" (Wang et al. 2016).

We use an improved variant of OEP as one baseline in our experiments, with the ϕ and ψ strategies described in Subsection 3.2.

2.3 Evolutionary MCTS

Evolutionary MCTS (EMCTS) (Baier and Cowling 2018) is the state of the art for playing multi-action turn-based adversarial games. It combines the tree search of MCTS with the genome-based approach of evolutionary algorithms such as OEP.

Instead of the vanilla MCTS tree seen in Figure 1, EMCTS builds a tree as shown in Figure 2. EMCTS does not start from an empty turn in the root, but from a complete sequence of actions – just like the genomes of OEP. EMCTS does not grow a tree that adds one action to the current sequence with every edge, but a tree that mutates the current sequence with every edge – using the same mutation operator as OEP. And EMCTS does not use rollouts to complete the game or the current turn and then evaluate it as our vanilla MCTS

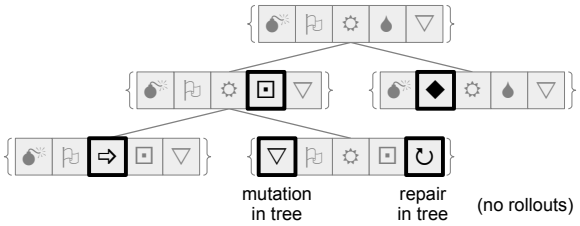

Figure 2: Tree structure of Evolutionary MCTS. Nodes represent complete action sequences (genomes), or the states resulting from them. Edges represent the mutation of an atomic action within a genome. Repairs can be necessary if those mutations can lead to illegal genomes. After each node expansion, the evaluation function is called instead of a rollout. (We use symbols to represent different atomic actions. Adapted from (Baier and Cowling 2018).)

Figure 3: The testbed game Hero Academy. The six symbols at the bottom represent the current player's hand. Adapted from (Baier and Cowling 2018).

baselines do (see Subsection 3.2), but it simply evaluates the sequences at the leaf nodes[1]. Backpropagation is unchanged.

EMCTS does not apply mutations randomly, but can pick precisely which action in the sequence to mutate and which other legal action to mutate it to[2]. While OEP turned the planning of the action sequence into an optimization problem, EMCTS thus takes the evolutionary optimization of the sequence and turns it back into a planning problem. It can be seen as tree search, but it can also be seen as a systematic exploration of the mutation landscape of OEP, giving evolution the benefit of lookahead.

EMCTS needs two more elements to be well-defined. First, it needs an *initialization strategy* ϕ for the root of its search tree, just like EAs such as OEP need a starting population of solutions. Second, EMCTS needs a method to handle mutations that lead to illegal action sequences – a *repair strategy* ψ just like in OEP. The ϕ and ψ strategies used in this paper, improved versions of those in (Baier and Cowling 2018), are described in Subsection 3.2.

3 Methods

This section briefly describes our testbed, lists the search algorithms we are comparing to, and finally presents our approach: Flexible-Horizon Evolutionary MCTS.

3.1 Test Domain: Hero Academy

Rules. Our test domain is a simplified Java clone (Niels Justesen 2015) of Hero Academy (Robot Entertainment 2012), a two-player turn-based tactics game. Players can use a variety of combat units, items, and spells by first drawing them from a card deck onto their hand, and then deploying, casting, or moving them on a battlefield of 9×5 squares. Special squares on this battlefield allow for unit deployment, boost the stats of individual units, or represent a player's two crystals. The game is won by the first player who either eliminates all enemy units, or destroys both enemy crystals. More details on implementation and rules can be found in (Justesen 2015).

A central mechanic of Hero Academy are the *action points* (APs). For each turn, the player to move receives a number of APs – five in the standard form of the game. Each AP can be used for any one atomic action such as deploying a unit from the player's hand onto the battlefield, moving a unit, attacking an enemy unit, healing a friendly unit, and others. The player can spend any number of APs on a single unit, for example by moving it several times. With an average of 30-60 actions available per game state, depending on the player, the full branching factor of a 5-AP turn can be roughly estimated to be $30^5 \approx 2.4 \times 10^6$ to $60^5 \approx 7.8 \times 10^8$. Especially at higher AP per turn, finding the best sequence of actions for any given turn can therefore be a challenging search problem in itself.

The order of cards in the deck as well as the opponent's cards are unknown to the Hero Academy player. However, this paper focuses on the challenge of multi-action turns, ignoring the aspects of hidden information and indeterminism like (Justesen et al. 2017) and (Baier and Cowling 2018).

In line with the prior work on Hero Academy, we use game knowledge for both state evaluation as well as action pruning and ordering:

State evaluation. All algorithms compared in this paper use the same heuristic evaluation function. This function is a linear combination of features such as the current health of individual units, whether they are equipped with certain items, and whether they are standing on special squares. Improving this hand-tuned function with machine learning could be interesting future work.

Action ordering. The two MCTS variants considered as baselines also make use of static action ordering, giving the more promising actions priority in their expansion and rollout phases. The heuristics used for this are simpler and faster than those of the evaluation function. Unlike vanilla EMCTS (Baier and Cowling 2018), FH-EMCTS also utilizes this fast ordering, as explained in Subsection 3.3.

The interested reader can refer to (Justesen 2015) for a full

[1]Evaluating at the leaf nodes is a well-known MCTS variant that was successfully employed for example in AlphaGo Zero and AlphaZero (Silver et al. 2017).

[2]No crossover operator is used.

definition of the heuristic evaluation function and the action ordering strategy.

3.2 Baseline Approaches

In order to make our results directly comparable to the literature, we test our approach against five of the algorithms described in (Justesen et al. 2017) and (Baier and Cowling 2018). One of them is a fast greedy search, two are vanilla MCTS variants, one is the evolutionary algorithm OEP, and one is EMCTS representing the state of the art for Hero Academy.

Greedy Action. The Greedy Action AI chooses the first action of its turn with a simple one-ply search of all legal actions, maximizing the heuristic evaluation of the immediately resulting state. This is repeated for each action of the turn.

Non-exploring MCTS. This AI is the first MCTS variant adapted for multi-action adversarial games in (Justesen et al. 2017). It searches a game tree as shown in Figure 1, in which each edge represents an additional action for the turn under consideration (or its application). The opponent's next turn can be reached if the tree grows deeper than the number of action points. The selection policy of this MCTS variant is UCB, and the rollout policy deterministically follows the action ordering heuristics. It was found to improve performance when rollouts are just long enough to complete the current turn of the player to act in the leaf node, calling the heuristic state evaluator at the end of the turn for a rollout result. The MCTS exploration factor is set to $C = 0$ in an attempt to grow a deep enough tree (pure exploitation).

Bridge-burning MCTS (BB-MCTS). This MCTS variant searches the same kind of tree shown in Figure 1. Instead of deterministic rollouts, it uses ϵ-greedy rollouts with $\epsilon = 0.5$, which also only reach to the end of the current turn of the leaf node. Its exploration factor is $C = 1/\sqrt{2}$. In order to grow a deep enough tree for multi-action turns however, it employs a technique called "bridge burning" in (Justesen et al. 2017).

The idea of "bridge burning" is to split the time budget for the current turn's search into several phases, equal to the number of actions per turn. During each phase, the MCTS search proceeds normally, but at the end of each phase, the most promising action at the root is executed, leading to the root state for the next phase. This can be implemented as the hard pruning strategy shown in Figure 4.

Greedy OEP. The Online Evolutionary Planning baseline is as described in Subsection 2.2. In our experiments, we use the same parameter settings as suggested in (Justesen et al. 2017): A population size of 100, a kill rate of 0.5, a mutation rate of 0.1, and uniform crossover and mutation operators. Additionally, we apply the Greedy Action AI as both initialization strategy ϕ and repair strategy ψ within OEP, leading to the improved *greedy OEP* as introduced by us in (Baier and Cowling 2018)[3].

Vanilla EMCTS. The Evolutionary MCTS baseline is as described in Subsection 2.3. As with OEP, we again follow

[3]To be precise, 20% of the starting population are filled with Greedy Action sequences, and 80% with random sequences. This kick-starts the search with higher-quality starting solutions.

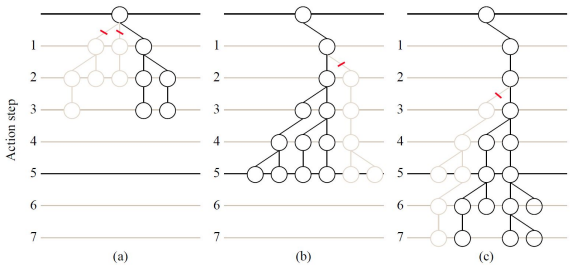

Figure 4: The "bridge burning" search strategy (adapted from (Justesen et al. 2017)). (a) After phase 1, all branches but the best one are pruned at the root. (b,c) After phases 2, 3, ... n, pruning is applied at depth 2, 3, ... n. The partial tree below the best branch is retained.

(Baier and Cowling 2018) in using the Greedy Action AI as both ϕ (for a quick and greedy root solution) and ψ (for mutations leading to illegal action sequences).

Finally, EMCTS searches trees with relatively large branching factors compared to vanilla MCTS. While the branching factor in Hero Academy games between the MCTS baselines is between 30 and 40, the branching factor of the EMCTS tree is about 30 *per action point* – so around 60-450 for the range of APs considered in this paper. In prior work (Baier and Cowling 2018), we dealt with this through "bridge burning", just as applied to the vanilla MCTS tree by BB-MCTS. Instead of executing the most promising action at the root after every search phase like BB-MCTS, EMCTS executes the most promising mutation at the root after each phase. The number of bridge burning phases, of successive searches and prunings/mutations, was the only EMCTS parameter tuned in (Baier and Cowling 2018), and we keep the value of 40 phases for 1 second searches unmodified at all numbers of action points per turn. The MCTS exploration factor was set to $C = 0$. The selection policy is UCB as in the other MCTS variants.

3.3 Flexible-Horizon Evolutionary MCTS

This subsection proposes our improved search algorithm, *Flexible-Horizon Evolutionary MCTS* or *FH-EMCTS*, as applied to playing multi-action turn-based adversarial games. It allows to extend the search horizon of EMCTS to any desired depth beyond the current turn, resulting in the strongest current AI for the test domain Hero Academy.

There are two major differences between vanilla EMCTS and FH-EMCTS: The first allows for deeper searches, and the second consists of faster behavior models that make these deeper searches effective at short time controls.

First, while the genomes in EMCTS encode only the actions of the current turn that is being searched, the genome length/search horizon h in FH-EMCTS is a parameter that can be set to any higher value as well. In the game of Hero Academy for example, the minimum length for its genomes is the number ap of action points per turn, but there is no upper limit (except for that imposed by diminishing performance on increasingly huge search spaces). Figure 5 illustrates this for the case of $ap = 3$: The EMCTS genome consists of the next

(a) Genome of regular EMCTS

(b) Genome of Flexible Horizon EMCTS

Figure 5: Example genomes of regular EMCTS and FH-EMCTS. Action points = 3, search horizon chosen for FH-EMCTS here = 7.

3 actions, but the FH-EMCTS genome is (an arbitrarily chosen) 4 actions longer, consisting of actions of the searching player shown on grey background, as well as actions of the opponent shown on white background. The basic idea is that all future actions of the searching player can be mutated and optimized during search, while opponent actions are always generated by an opponent model π. For every mutation in the tree, future actions of the searching player are only repaired by ψ when they become illegal, while opponent actions are always updated by policy π. This effectively turns the opponent into a part of the game's state transitions. As the fastest approximative model of reasonable behavior available to us in Hero Academy, we use the Greedy Action AI as opponent model π in this paper.

Second, since these longer genomes can lead to many more legal mutations per tree node, as well as many more calls to the Greedy Action AI (which is now used as a universal behavior model for ϕ, ψ, and π) per mutation, the computational cost needs to be reduced to make FH-EMCTS effective at equal search times. We have achieved this mainly by restricting each choice of the Greedy Action AI to the most promising 10 actions as ranked by same static action ordering also used by BB-MCTS and non-expl. MCTS. In addition, we employ lazy generation of the mutation actions in the tree search, and a number of other MCTS code optimizations. In Subsection 4.2, we analyze the separate effects of these improvements and the flexible search horizon on the performance of FH-EMCTS vs. vanilla EMCTS.

4 Experimental Results

This section describes our three sets of experiments for testing the proposed Flexible-Horizon Evolutionary MCTS in Hero Academy, as well as the results.

Unless specified otherwise, all comparisons were done with a search time limit of 1 second per turn, and each comparison consisted of 1000 games, with FH-EMCTS playing 500 games as the first player and 500 games as the second player. Games that had no winner after 200 turns were counted as draws, i.e. half a win for each player.

All algorithms used the parameter settings described in Section 3. Hero Academy was modified to allow for different numbers of action points per turn – from 2 to 15 APs – in order to vary the challenge for the search algorithms.

4.1 Tuning FH-EMCTS

In the first set of experiments, we tested FH-EMCTS against vanilla EMCTS at all action points per turn $ap \in \{2, \ldots, 15\}$ and all search horizons $h \in \{ap, \ldots, ap + 15\}$, with 400 games per comparison. The page limit does not allow a full presentation of the parameter landscape here, but FH-EMCTS performed best at the genome lengths given in Table 1. Despite some noise, searches with $h > ap$ were clearly most successful for relatively low ap, while at the highest tested ap the optimal setting was $h = ap$. These high ap lead to such large search spaces that it is not worth risking to miss a good current turn by spending limited search time on actions beyond the turn. Mainly due to the faster behavior models however, FH-EMCTS still consistently outperformed vanilla EMCTS significantly at all ap.

Table 1: The best-performing genome lengths/search horizons for FH-EMCTS vs. vanilla EMCTS. 1 second per move.

Actions per turn ap:														
2	3	4	5	6	7	8	9	10	11	12	13	14	15
Best genome length/search horizon h for FH-EMCTS:														
15	7	8	10	8	9	9	11	12	13	14	13	14	15
Win rates of FH-EMCTS vs. EMCTS in % (rounded):														
85	69	64	62	61	57	58	61	60	60	59	61	59	62

4.2 What makes FH-EMCTS strong?

If both flexible search horizons h and faster behavior models ϕ, ψ, and π improve EMCTS, how much improvement comes from either factor alone? In the second set of experiments, we tested this by letting FH-EMCTS play against vanilla EMCTS with $h = ap$, i.e. with the search horizon restricted to the current turn. This showed the impact of the faster behavior models alone. Additionally, we let FH-EMCTS play with the search horizons found to be optimal in Subsection 4.1, but against a version of vanilla EMCTS that was also using the faster behavior models. This showed the genuine improvement through deeper searches, given that faster behavior models are available.

Figure 6 compares these two conditions to the combined effect, i.e. the performance of full FH-EMCTS vs. unenhanced vanilla EMCTS. It shows that the improved behavior models are most useful with longer turns, as they require many more calls to ϕ, ψ, and π. The flexible search horizons however are most valuable at shorter turns, because those allow even a 1 second search to go far beyond the current turn. Long turns provide enough of a search challenge that trying to search them more deeply ($h > ap$) can lead to missing good immediate actions.

4.3 FH-EMCTS vs. all baselines

In the third set of experiments, we tested FH-EMCTS as tuned in Section 4.1 against a variety of baselines, in order to confirm its effectiveness. The opponents were the state-of-the-art vanilla EMCTS and the greedy OEP approach as proposed

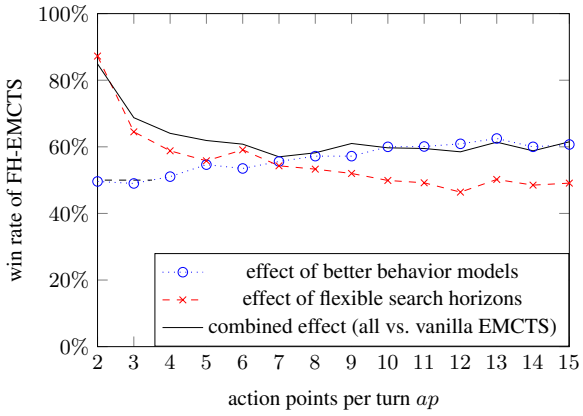

Figure 6: Performance of FH-EMCTS vs. vanilla EMCTS: the effects of the improved behavior models ϕ, ψ, and π, of searching beyond the current turn, and their combined effect. Search time is 1 second per turn. 1000 games per data point.

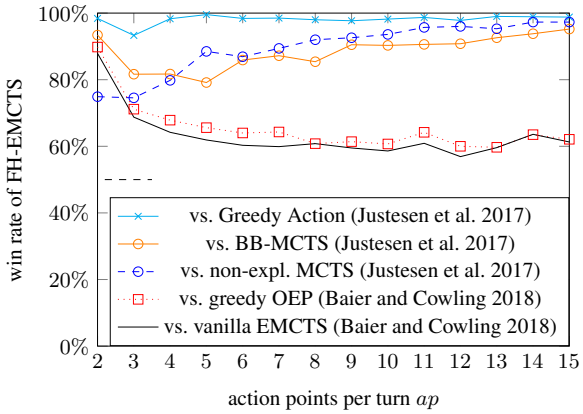

Figure 7: Performance of FH-EMCTS vs. all baselines. 1 second per turn. 1000 games per data point.

by us in (Baier and Cowling 2018), as well as the BB-MCTS, non-exploring MCTS, and Greedy Action techniques first used as baselines in (Justesen et al. 2017). Greedy Action is using action ordering and pruning here as well.

Figure 7 shows the results. FH-EMCTS consistently and significantly outperforms all baselines at all tested turn lengths. Notably, FH-EMCTS removes a weakness of vanilla EMCTS against non-expl. MCTS at very low APs per turn, where the ability to search beyond the current turn is crucial. Non-expl. MCTS and FH-EMCTS have this ability, while vanilla EMCTS does not. Furthermore, the weak performance of the Greedy Action AI demonstrates that the behavior models ϕ, ψ, and π used by FH-EMCTS do not have to be strong standalone players to make for adequate initialization/repair/opponent policies.

5 Conclusions and Future Work

This paper proposes a number of effective enhancements for EMCTS, the current state of the art search algorithm for playing turn-based multi-action adversarial games. In such games, each turn consists of multiple actions, which creates the challenge of extremely large branching factors per turn and has led previous approaches such as EMCTS and OEP to focus exclusively on optimizing the current turn. We identified this as a shortcoming of EMCTS in (Baier and Cowling 2018). With our enhancements, the proposed *Flexible Horizon EMCTS* (FH-EMCTS) can effectively search to greater depths (depending on turn length and search time), making it an even more promising approach for Hero Academy and similar turn-based multi-action adversarial games.

Several directions remain interesting for future work, some of which we already mentioned in (Baier and Cowling 2018) as well. First, the searching player's actions beyond the current turn are optimized by FH-EMCTS, but the opponent's actions are generated by an opponent model. Other variants could be imagined, in which the searching player's future actions are also coming from a fixed model, or in which the opponent's actions are also optimized online (possibly through an approach similar to (Hong, Huang, and Lin 2001)). Second, FH-EMCTS should be tested in other tactics and strategy games. In many Civilization-type games for example, the number of actions per turn varies over the course of the game, and flexible search horizons could therefore be most effective. Third, generalizing to larger classes of games will require dealing with randomness and partial observability. Fourth, generalization to commercial games will pose interesting challenges around replacing the game knowledge used by EMCTS (and OEP) with machine learning approaches. We have simply chosen to use the same policy as initialization strategy, repair strategy, and opponent model here—this could offer much room for improvement. Fourth, FH-EMCTS could just like OEP also be generalized to other problems such as micro battles (Wang et al. 2016) or online build order adaptation (Justesen and Risi 2017) in real-time strategy games. In these applications, the genomes would not represent a sequence of future actions for the player, but e.g. a list of scripts representing simple policies assigned to each unit, or a sequence of future units and buildings to be constructed, respectively. Fifth, it could be interesting to extend OEP to work with flexible search horizons as well, and check whether the resulting FH-OEP is still outperformed by FH-EMCTS as OEP is outperformed by vanilla EMCTS. And finally, the most interesting long-term task could be the exploration of algorithmic similarities between Evolutionary MCTS and certain local search algorithms and evolutionary algorithms, in order to further study the idea of EMCTS providing evolution with lookahead search.

References

Alhejali, A. M., and Lucas, S. M. 2013. Using genetic programming to evolve heuristics for a Monte Carlo Tree Search Ms Pac-Man agent. In *2013 IEEE Conference on Computational Intelligence and Games, CIG 2013*, 1–8.

Baier, H., and Cowling, P. I. 2018. Evolutionary MCTS for Multi-Action Adversarial Games. In *2018 IEEE Conference on Computational Intelligence and Games, CIG 2018*. Forthcoming.

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.;

Cowling, P.; Rohlfshagen, P.; Tavener, S.; Perez-Liebana, D.; Samothrakis, S.; and Colton, S. 2012. A survey of Monte Carlo Tree Search methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4(1):1–43.

Chaslot, G. M. J. B.; Winands, M. H. M.; Szita, I.; and van den Herik, H. J. 2008. Cross-entropy for Monte-Carlo Tree Search. *ICGA Journal* 31(3):145–156.

Cole, N.; Louis, S. J.; and Miles, C. 2004. Using a genetic algorithm to tune first-person shooter bots. In *2004 Congress on Evolutionary Computation (CEC 2004)*, 139–145.

Coulom, R. 2007. Efficient selectivity and backup operators in Monte-Carlo Tree Search. In *5th International Conference on Computers and Games, CG 2006. Revised Papers*, volume 4630 of *Lecture Notes in Computer Science*, 72–83.

Firaxis Games. 2016a. Civilization VI. https://civilization.com/.

Firaxis Games. 2016b. XCOM 2. https://xcom.com/.

Gaina, R. D.; Liu, J.; Lucas, S. M.; and Perez-Liebana, D. 2017. Analysis of Vanilla Rolling Horizon Evolution Parameters in General Video Game Playing. In Squillero, G., and Sim, K., eds., *20th European Conference on Applications of Evolutionary Computation, EvoApplications 2017*, volume 10199 of *Lecture Notes in Computer Science*, 418–434.

Hong, T.; Huang, K.; and Lin, W. 2001. Adversarial Search by Evolutionary Computation. *Evolutionary Computation* 9(3):371–385.

Justesen, N., and Risi, S. 2017. Continual Online Evolutionary Planning for In-game Build Order Adaptation in StarCraft. In Bosman, P. A. N., ed., *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017*, 187–194. ACM.

Justesen, N.; Mahlmann, T.; Risi, S.; and Togelius, J. 2017. Playing Multi-Action Adversarial Games: Online Evolution versus Tree Search. *IEEE Transactions on Computational Intelligence and AI in Games*. Forthcoming.

Justesen, N. 2015. Artificial Intelligence for Hero Academy. Master's thesis, IT University of Copenhagen.

Kocsis, L., and Szepesvári, C. 2006. Bandit based Monte-Carlo planning. In *17th European Conference on Machine Learning, ECML 2006*, volume 4212 of *Lecture Notes in Computer Science*, 282–293.

Limbic Entertainment. 2015. Might & Magic Heroes VII. https://www.ubisoft.com/en-gb/game/might-and-magic-heroes-7/.

Lucas, S. M., and Kendall, G. 2006. Evolutionary Computation and Games. *IEEE Computational Intelligence Magazine* 1(1):10–18.

Midjiwan AB. 2016. The Battle of Polytopia. http://www.midjiwan.com/polytopia.html.

Niels Justesen. 2015. Hero AIcademy. Available at https://github.com/njustesen/hero-aicademy.

Perez-Liebana, D.; Samothrakis, S.; Lucas, S. M.; and Rohlfshagen, P. 2013. Rolling Horizon Evolution versus Tree Search for Navigation in Single-Player Real-Time Games. In Blum, C., and Alba, E., eds., *2013 Genetic and Evolutionary Computation Conference, GECCO '13*, 351–358. ACM.

Risi, S., and Togelius, J. 2017. Neuroevolution in Games: State of the Art and Open Challenges. *IEEE Transactions on Computational Intelligence and AI in Games* 9(1):25–41.

Robot Entertainment. 2012. Hero Academy. Available at http://www.robotentertainment.com/games/heroacademy/.

Schadd, M. P. D.; Winands, M. H. M.; Tak, M. J. W.; and Uiterwijk, J. W. H. M. 2012. Single-Player Monte-Carlo Tree Search for SameGame. *Knowledge-Based Systems* 34:3–11.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T. P.; Simonyan, K.; and Hassabis, D. 2017. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *CoRR* abs/1712.01815.

Subset Games. 2018. Into the Breach. https://subsetgames.com/itb.html.

Syswerda, G. 1989. Uniform Crossover in Genetic Algorithms. In *3rd International Conference on Genetic Algorithms, ICGA 1989*, 2–9.

Wang, C.; Chen, P.; Li, Y.; Holmgard, C.; and Togelius, J. 2016. Portfolio Online Evolution in StarCraft. In *Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE-16*, 114–120.