Application of MCTS in Atari Black-box Planning

Alexander Shleyfman
Technion, Haifa, Israel
alesh@campus.technion.ac.il

Alexander Tuisov
Technion, Haifa, Israel
alexandt@campus.technion.ac.il

Carmel Domshlak
Technion, Haifa, Israel
dcarmel@technion.ac.il

Abstract

Action selection in environments where the problem structure is hidden by an action simulator presents a challenge for domain-independent action planning. Using the Arcade Learning Environment (ALE) that supports Atari 2600 games, recent research on the subject led to several planning algorithms suitable for this challenging settings. The most competitive of this set of algorithms are variants of best-first search with action pruning based on the properties of the states already generated by the simulator. Pushing the envelope of domain-independent planning with simulators, we show that a different family of algorithms, one that plans for a bounded-length trajectories and not only for the next action to apply, allows solving problems that so far were out of our reach. In particular, we present a family of such Monte-Carlo Tree Search algorithms that favorably compete with its state-of-the-art counterparts. Likewise, noticing that the two algorithmic approaches are rather complementary, we examine both a pre-sampling based selection among the two, as well as an alternating composition of the algorithms, and show that they favorably compete with both of their individual components.

Introduction

Popular in the 80s, recently Atari 2600 games once again became a center of attention, but now for an entirely different reason. In 2013, Bellemare *et al.* (2013) introduced the Arcade Learning Environment (ALE) – a convenient platform for domain-independent planners and learners with an accessible interface to numerous Atari video games. In these black-box planning domains, the set of actions, the vector state, and the objective function are fully observable, and all actions have a deterministic effect. At the same time, the action and reward dynamics are inaccessible, and given only via a simulator.

While making the problems closer to many challenges of the real-world applications, these traits of the ALE setup also prevent one from using techniques that have been developed over the years for search in declaratively represented domains, such as planning as heuristic search or planning as satisfiability (Russell and Norvig 2010; Geffner and Bonet 2013). At the high level, this leaves us only with variants of the brute-force search methods such as breadth-first search (BRFS), as well as with sampling-based Monte-Carlo Tree Search (MCTS) algorithms.

The first evaluation of such techniques on ALE was done by Bellemare *et al.* (2013), and it showed that UCT (Kocsis and Szepesvári 2006), one of the most popular MCTS algorithms, substantially outperforms BRFS on a wide set of Atari 2600 games. Recently, however, Lipovetzky *et al.* (2015) and Shleyfman *et al.* (2016) showed that some more sophisticated variants of breadth-first search, namely Iterative Width (IW(i)) and Prioritized Iterative Width (P-IW(i)), respectively, exhibit significantly better performance than UCT. Both these algorithms exploit state pruning that focuses the search only on states which are "novel" with respect to the previously discovered states, with the difference between the two being primarily in the way the state novelty is defined.

The results of Lipovetzky *et al.* (2015) and Shleyfman *et al.* (2016) positioned the breadth-first search algorithms as the tools of choice for problem setups like ALE, and somewhat pushed MCTS into the shadow here. However, a closer look at the empirical results reveal that the superior performance of IW(i) and P-IW(i) over UCT was not uniform across the different Atari 2600 games, and in fact, the two sets of tools could have been seen as complementary in terms of games coverage. Furthermore, several works have shown, both formally and empirically, that UCT is not necessarily the most effective MCTS algorithm available (Coquelin and Munos 2007; Bubeck *et al.* 2009; Feldman and Domshlak 2014; 2013).

Given the above, in this work we examine whether the effectiveness of MCTS techniques in the ALE environment can be pushed substantially further, and provide an affirmative answer to this question, even using relatively straightforward techniques. Specifically, following the path taken by Shleyfman *et al.* (2016), we consider action selection in ALE as a multiarmed bandit style competition between the actions available at the current state. However, in contrast to the work of Shleyfman *et al.*, the competition here is done at the level of action sequences, and thus the planning is done not for the next "best" state, but for some time epoch of a preset length. We show that this approach both dominates UCT, as well as favorably competes with P-IW(1). Furthermore, we show that this approach is complementary

to the breadth-first variations, and explore some techniques that successfully combine the two, either by selecting the more appropriate method to the task at hand, or alternating between the two approaches.

Background

The ALE problem, as it was formalized by Jinnai and Fukunaga (2017), is a tuple $\langle V, A, f, s_0, r\rangle$, where:

- $V = \{v_1, \ldots, v_n\}$ is a finite set of state variables with finite domains $\mathcal{D}(v_i)$, and each state is represented by a complete assignment to these variables (the variable/value pairs are written as $v_i = d$, and sometimes referred to as *facts*);
- A is a finite set of actions, with all actions being applicable in all states;
- f is a deterministic transition function represented by a simulator, with $f(a, s)$ being the state of the game that follows the application of an action a in the state s;
- s_0 is a starting state; and
- r is a real-valued reward function, with $r(s)$ being the reward obtained by applying (any) action in state s.

This setting is somewhat similar to the classical planning in the sense that the current state of the game starting with s_0 and actions A are known to the agent. However, both the transition and reward functions are initially hidden, and are gradually revealed with the search progress through interactions with the simulator: upon simulation of applying action a in state s, the resulting state $f(a, s)$ and the reward $r(f(a, s))$ are being revealed to the planner. Following Shleyfman *et al.* (2016), a cumulative reward $R(s)$ of a state s is recursively defined as $R(s') = R(s) + \gamma^d r(s)$, where s' is the unique parent state of s, γ is a discount factor, and d is the depth of s in the search tree.

P-IW(1)

One of the more prominent algorithms for black-box planning is P-IW(1). It is a regular breadth-first search with a following modifications: When a state s is generated it is assigned a novelty value. A state is declared novel if it has at least one fact $v_i = d$ in s was not previously observed as an element of some state s'' which had a higher cumulative reward than s. If a state is not novel, it is pruned from the search tree. Ties in the BRFS queue are broken by the cumulative reward of the states belonging to the nodes in question. The action a chosen to be applied in s_0 is the action on a path to the leaf node with a state that yields highest cumulative reward. Novelty-based algorithms however often explore the search space in a highly imbalanced manner, often leading to "single tunnel" phenomenon, where the search tends to prune all but one path to the deeper parts of the search tree, and the comparison between different actions close to the root becomes skewed. P-IW(1), although slightly alleviates the problem, is no stranger to this drawback either.

UCT

Another algorithm that had been put to the test in aforementioned setting is UCT, a MCTS-based algorithm originally created for MDP planning by Kocsis and Szepesvári (2006) and adapted for ALE by Bellemare *et al.* (2013). UCT explores the search space by growing a search tree in a manner that treats every search tree node as an independent multi-armed bandit problem (MAB). The algorithm makes use of the UCB1, constructing a statistical confidence intervals for each of the arm in the search node. The UCT algorithm, however, treats each arm optimistically, i.e., evaluating each arm j only by the left part of the interval, resulting in the formula:

$$x_j + \sqrt{2(\ln n)/n_j},$$

where x_j is the the mean payout for arm j, n_j is the number of plays of arm j, and n is the total number of plays from the current node. The strategy of UCT is to pick the arm with the highest upper bound each time. In the *selection* phase, the algorithm moves down on the tree nodes, using the statistics necessary to treat each position as a MAB. This phase lasts until the algorithm reaches a tree leaf. The phase of *expansion* occurs when UCB1 no longer applies. An unvisited child position is randomly chosen, and a new record node is added to the search tree. And after that come the *simulation* and *back-propagation* phases. These are typically done by Monte Carlo simulation (until the algorithm reaches some preset horizon), than averaging on the result of this simulation, correspondingly.

Algorithm stops at a leaf node, and expands it. It then applies a random simulation (rollout) of a certain length to obtain a score estimate for all its successors, and back-propagates the result. It is important to note that the UCT procedure saves in the memory only the search tree built so far, and does not save any simulated rollouts. ALE problems, however, differ from the MDP and game tree problems UCT was originally developed for. ALE environment is deterministic, thus any reward observed is deterministically achievable. This fact shifts our interest from finding best average reward to finding a maximal reward. Moreover, UCB1 formula minimizes the cumulative regret, where ALE design and dynamics suggest one should try and minimize simple regret (the difference between best trajectory found, and best trajectory there is), closer to the setting presented by Schulte and Keller (2014).

Tree Search Algorithms

In this section, we describe the MCTS family of algorithms in general, and approaches we used in particular. Since the MCTS algorithms were not originally developed for the ALE setting, description of adaptations made to fit MCTS to the problem at hand also follows.

MCTS family of algorithms for game trees as presented by Chaslot *et al.* (2008) has the following structure: it consists of four steps, namely selection – traversing the search tree from the root until leaf node using *selection strategy*, expansion – storing some descendants of the chosen node using *expansion strategy*, simulation – evaluation of the leaf node using *simulation strategy*, and back-propagation – updates the value of the nodes based on the results of the simulation using *back-propagation strategy*. These steps are re-

peated as long as the resources allowed per decision aren't exhausted.

Each of these steps has its own strategy, and to describe an MCTS algorithm it is sufficient to describe these four strategies. It is worth noting that the application of the generalization of the MCTS methods – Trial-Based Heuristic Tree Search (THTS) (Keller and Helmert 2013) had been explored in the context of classical planning (Schulte and Keller 2014), which also features deterministic actions, but differs from our setting.

1. In the deterministic setting back-propagation strategy is pretty straightforward. We propagate back the reward of the best trajectory in every algorithm, where by "trajectory" we mean a sequence of consequently applied actions, and by "best" we mean leading to an end state (determined by a preset horizon) with highest cumulative reward. Since the ALE setting is deterministic, every trajectory can be followed through, and this cumulative reward is achievable at the execution. Therefore, there is no need to consider sub-optimal trajectories after the exploration is finished. This method of back-propagation was discussed in the work of Schulte and Keller (2014) (albeit in a cost rather than a reward setting). Back-propagation strategies will not be discussed further.

2. Selection strategy – deviating from MCTS scheme, we permit selecting non-leaf nodes. Moreover, following the work of Tolpin and Shimony (2012) on MCTS algorithms, in some of the following methods we decouple selection in the root node from selection in the rest of the tree. The intuition for this approach is derived from the game setting, where we apply actions step-by-step, and not fully execute the best trajectory found by the algorithm.

3. Simulation and Expansion strategies – in the deterministic setting of ALE, there is no much point in generating (via simulator) already existing states, thus we combine both these strategies into one. This will result in some nodes in the search tree, where not all the immediate successors had been generated. This approach is not implemented in the UCT algorithm, which "spends" most of the allocated budget on simulating rollouts that will add one tree node each, resulting in a much more shallow search tree.

   Simulating a result here means sampling a trajectory from the trajectory space via Monte-Carlo procedure. Since we would like to compare trajectories of an equal length (otherwise we might create a bias towards longer, but not necessarily better trajectories, or, as in games with negative rewards, shorter but less informed trajectories), we opt to simulate the result only until a certain planning horizon, which is also a maximal trajectory length. In what follows, simulation of trajectory or a trajectory itself may be referred as rollout.

Using this formulation we describe the family of algorithms presented below in terms of the selection and simulation strategies they employ.

UNIFORM

The first, and the most trivial approach, UNIFORM is based upon a uniform selection of the immediate successor of the root. The simulating strategy of applying uniformly selected random actions until it reaches maximal trajectory length l[1]. Given the fact that in the ALE setting, there is a fixed predetermined number of actions in each state (18, to be exact), it is equivalent to picking uniformly at random a trajectory of length l. This approach, however, suffers from some drawbacks. For example, it seems to be "wasting" numerous simulations on creating trajectories from less rewarding (at least so far) children of the root. So, it would seems natural to employ some gradual "candidate rejection" technique.

SEQHALVING

The next approach, SEQHALVING, tries to tackle this problem. SEQHALVING has the same simulation strategy as UNIFORM, but it employs more sophisticated selection strategy. It approaches the selection problem as multi-armed bandit, where the children of the root node are considered bandit arms, and the act of creating a rollout from a node is parallel to sampling a bandit arm. Now one could use an apparatus created for solving MAB problem with fixed budget (since our budget is known in advance to the decision making process), namely SEQUENTIALHALVING technique suggested by Karnin *et al.* (2013). It operates as described in Algorithm 1.

Algorithm 1 SEQHALVING

initialize $T \leftarrow$ total budget
initialize $S_0 \leftarrow$ set of all children of the root node
initialize $n \leftarrow |S_0|$
for $r = 0$ **to** $\lceil log_2 n \rceil - 1$ **do**
    make rollout from every node $i \in S_r$ for

$$\left\lfloor \frac{T}{|S_r| \lceil log_2 n \rceil} \right\rfloor$$

    times
    let S_{r+1} be the set of $\lceil |S_r|/2 \rceil$ successors of the root with the highest maximal rewards
return the subtree rooted at the first action in the best trajectory

On one hand, this technique alleviates the aforementioned problem. On the other hand, the exploration dynamics of the nodes beyond the immediate successors of the root are still unbiased, and contain no elements of exploitation of any previously found high-reward nodes.

ϵ-GREEDY

As discussed before, it may be beneficial to try and shift the dynamics towards exploitation. In the following approach,

[1]Note that here there is no difference was the selected node previously was previously generated or not. Previously generated node, however, will not spend the allocated budget, since the state is already part of the search tree.

ϵ-GREEDY selection strategy chooses a node leading to the best trajectory with some probability $0 < \epsilon < 1$, and selects a node uniformly otherwise. Simulation strategy follows a similar principle. The first node of the rollout (if non-leaf) and encountering a non-leaf node yielding a positive reward are decision points. At any decision point, the simulation follows previously simulated trajectory with probability ϵ until the next decision point (or until the planning horizon if no more decision points are present). With probability $1 - \epsilon$, or when in a leaf node, the algorithm chooses the next action uniformly. All children of a node that have not been previously generated are counted as if they were yielding a reward of $-\infty$.

Reasoning

The family of the aforementioned algorithms have at least two reassuring properties. First, given a fixed amount of always applicable actions (as is the case in ALE setting), each trajectory of a preset length can be generated uniformly. Thus, the expectation of the value of the recommended trajectory is monotonic in budget, which is a property novelty-based algorithms such as P-IW(1) don't possess. Second, the probability of recommending the best trajectory for a given planning horizon converges to 1, given a sufficiently large budget. This is also not true for novelty-based algorithms, because of the pruning procedure involved. Note, however, that the optimality of the trajectory is judged with regard to the fixed horizon l, and not for the whole game.

Experimental Results

Out of the presented family of MCTS algorithms we choose to evaluate UNIFORM, SEQHALVING, and ϵ-GREEDY. We are evaluating them against two of the most prominent state-of-the-art algorithms P-IW(1) and UCT. As mentioned before, the testbed for these evaluations will be the games of the ALE setting by Bellemare *et al*. (2013). As in previous works, we exclude two games: SKIING game was already left out in the experiments of Lipovetzky *et al.* (2015) due to certain issues with the reward structure of this game[2], and BOXING since by Shleyfman *et al.* (2016), the game boils down to striking in arbitrary directions since the second player is inactive, thus every algorithm trivially scores the possible maximum. This leaves us with 53 of the 55 different games. We also contemplated to leave out the SPACE INVADERS games, since it may be flawed, and terminates due to some inner bug. However, we decided to leave it, since all the algorithms compete in the same conditions.

We have implemented our algorithms on top of the implementation of Lipovetzky *et al.* (2015), with the addition of P-IW(1) by Shleyfman *et al.* (2016). The implementation of the UCT algorithm was provided by Bellemare *et al.* (2013). Its exploration constant here is set to 0.1. In each decision point, the algorithm normalizes its rewards according to the first reward it has found.

In our experiments, we use the setting of frames reuse proposed by Bellemare *et al.* (2013). In this setting the frames in the sub-tree of the previous lookahead provides the algorithms with "additional" simulations, since there is no point in re-generation of already existing states. In their works Lipovetzky *et al.* (2015) and Shleyfman *et al.* (2016) used the a lookahead budget of 150000 simulated frames (or, equivalently, 30000 search nodes), with time limit of 18000 frames (5 minutes) for each game. This, however, seems unpractical to us, since the duration of full simulation process of a "five-minute-real-time" game may, in practice, take more than three days of computation time. Therefore, we limit our simulation budget to 50000 frames (or, equivalently, 10000 search nodes). It is worth noting that even after this limitation, the evaluation process is still extremely demanding to computational resources, which severely limits our ability to thoroughly check many configurations without compromising on the variance of the results.

The lookahead depth was limited to 1500 frames (300 search nodes, or, equivalently, 25 seconds of game time), and the accumulated rewards were discounted as $R(s') = R(s) + \gamma^d r(s)$ where s a unique parent of s', and d is the depth of s in the search tree. The discount factor was set to $\gamma = 0.95$. To reduce the variance of the result each game was executed 30 times, with seeds $0, \ldots, 29$.

Table 2 shows that UNIFORM, SEQHALVING, and ϵ-GREEDY rather consistently outperform UCT. For example, on the 53 games, UNIFORM achieved higher average scores in 44 games, 1 game ended up with a draw, and UCT achieved higher average scores in 8 games. The situation is almost the same for SEQHALVING and ϵ-GREEDY, with the only difference that both of them draw in one more game, rather than winning it. It's also important to note that this family of MCTS algorithms also outperforms P-IW(1) in a majority of the games, as Table 2 shows us (albeit with a bit less significant margin).

Composite methods

As can be seen from Table 1, the relative performance of the novelty-based and MCTS approaches varies highly depending on the game (take the scores for FROSTBITE and JAMESBOND as an example). These approaches seem to be somewhat complementary indeed, so it seems natural to try and combine them in some way, such that we could reap the benefits of both. In this section we cover a few ways in which a composition between them could be achieved. It is worth mentioning that there was some earlier attempts to use some novelty properties in MCTS by Soemers *et al.* (2016), however, the methods introduced in their work depend heavily on the GVG-AI environment, and cannot be generalized to ALE straight forward. Also, it should be noted that we attribute the effect on the relative scores of P-IW(1) and MCST methods on different games to some innate property of the games themselves (rather than pure chance).

Naïve approach

One can assume that the aforementioned property might be discovered relatively early in the game (the correctness of this assumption will be discussed later). If this is true, one also can invest a relatively negligible amount of simulations

[2]The rewards in this game are time based, and it is challenging to extract these rewards in the black-box ALE setting.

Game	P-IW(1)	UCT	UNIFORM	SEQHALVING	$\epsilon = 0.67$
ALIEN	13264	6382	**17240**	14696	16881
AMIDAR	**2041**	47	910	1005	1031
ASSAULT	1552	1625	1831	1805	**1867**
ASTERIX	**347800**	303333	255875	264283	271267
ASTEROIDS	5548	4122	9987	10541	**12490**
ATLANTIS	197450	178753	197547	197133	**200223**
BANK HEIST	592	518	1793	2117	**2498**
BATTLE ZONE	3667	41500	130400	**232200**	103967
BEAM RIDER	4868	5024	14208	15340	**15745**
BERZERK	485	555	**739**	723	684
BOWLING	64	22	**83**	**83**	80
BREAKOUT	856	805	849	849	**864**
CARNIVAL	5605	4787	**6759**	6446	6323
CENTIPEDE	**186964**	106260	136776	138767	144143
CHOPPER COMMAND	2140	18243	36480	42500	**52240**
CRAZY CLIMBER	**140833**	135563	97134	84525	91753
DEMON ATTACK	**38898**	24128	30512	30700	31641
DOUBLE DUNK	-16	**24**	**24**	**24**	**24**
ELEVATOR ACTION	23077	14427	26137	25790	**26390**
ENDURO	0	279	405	**428**	426
FISHING DERBY	18	34	36	**39**	26
FREEWAY	**33**	0	8	8	7
FROSTBITE	**7667**	272	293	295	293
GOPHER	28618	8215	29285	29544	**30099**
GRAVITAR	1177	2888	5767	**5862**	5235
HERO	5702	10100	13933	**14478**	13946
ICE HOCKEY	15	39	**56**	**56**	**56**
JAMESBOND	40	385	13813	**15068**	14927
JOURNEY ESCAPE	2507	1320	**86120**	84420	77830
KANGAROO	**4337**	2048	2027	2047	2107
KRULL	**11443**	8742	5241	5744	6023
KUNG FU MASTER	**79717**	50347	63000	63690	63667
MONTEZUMA REVENGE	0	0	50	**293**	193
MS PACMAN	23584	17502	30893	30958	**33548**
NAME THIS GAME	**16713**	14927	13992	13779	14062
PONG	**21**	**21**	**21**	**21**	**21**
POOYAN	18943	14655	20358	**21182**	20802
PRIVATE EYE	920	100	1040	9	**1076**
QBERT	21727	17598	38274	**43233**	36553
RIVERRAID	**11702**	6316	9607	9915	9994
ROAD RUNNER	**62650**	39043	59650	60343	55283
ROBOTANK	6	45	**58**	**58**	**58**
SEAQUEST	590	543	499	490	**609**
SPACE INVADERS	2448	2482	2603	**2665**	2417
STAR GUNNER	1373	**1467**	1233	1217	1200
TENNIS	**24**	3	**24**	**24**	22
TIME PILOT	54137	52640	53060	53052	**63386**
TUTANKHAM	135	229	**271**	259	257
UP N DOWN	73325	63272	99037	**100593**	99639
VENTURE	**33**	0	10	17	0
VIDEO PINBALL	**592102**	323700	340109	328939	334709
WIZARD OF WOR	115347	98327	127667	129713	**133525**
ZAXXON	21553	24540	45047	**52217**	46123
Best in	16	3	9	15	16

Table 1: Performance results over the 53 Atari 2600 games. The algorithms P-IW(1), UCT, UNIFORM, SEQHALVING, $\epsilon = 0.67$, (ϵ-GREEDY) are evaluated over 30 episodes for each game. The look ahead of every algorithm is limited to a budget of 50000 simulated frames. The maximum episode duration is 18000 frames. Numbers in bold show best performer in terms of average score. The **Best in** row shows on how many games an algorithm scored the maximum.

	P-IW(1)	UCT	UNIFORM	SEQHALVING	$\epsilon = 0.67$
$p - IW(1)$	0	34	20	21	19
UCT	17	0	8	8	8
$Uniform$	33	44	0	17	21
$SeqHalving$	31	43	34	0	25
$\epsilon = 0.67$	34	43	31	27	0

Table 2: presents on how many instances the algorithm in row X outperformed the algorithm in column Y. For example, UNIFORM was strictly better than UCT on 44 out of 53 instances.

in order to discover which algorithm is better suited for this particular game. We propose to do it in the following way:

1. in the first decision point of the game, use some budget of simulations B to run P-IW(1);
2. delete the search tree;
3. use budget B again to run an UNIFORM algorithm described in Section ;
4. delete the search tree;
5. for the rest of the decision points in the game, use the algorithm that yielded the highest cumulative reward.

It is important to emphasize that this decision is being made only once per game, thus B can be relatively large compared to the budget allocated per decision point.

ALTERNATION

One of the techniques for combining different approaches for the same problem is alternating between them. In classical planning this approach was introduced by Röger and Helmert (2010).

The version of alternation presented here builds a search tree as dictated by the UNIFORM algorithm at every even decision point, and as dictated by P-IW(1) at any odd decision point (thus alternating between the two). This type of combination, however, presents an ambiguity when we reuse already generated frames. In order to get rid of this ambiguity, we apply these rules:

1. odd step: run the P-IW(1) algorithm, do not prune the nodes that are already present in the search tree (including those added in UNIFORM step), and ignore them in the novelty calculation, so more nodes can escape pruning;
2. after recommending the "best" action chosen by the algorithm, add the nodes rooted in the node created by the recommended action to the search tree as they are;
3. even step: run the UNIFORM algorithm;
4. once again, pass the subtree of the winning action to the next iteration;
5. in each step, recommend the action, that leads to the trajectory with the highest commutative reward.

In contrast to the MCTS family of algorithms described in Section , this algorithm often picks actions that lead to the rooted subtrees of different sizes and depths. This may result both in positive and negative outcomes, as will be demonstrated in the next Section.

Empirical Evaluation of the Composite Methods

We evaluate both composition methods mentioned in the previous Section on the setting described in Section [3]. We compare both amalgamation methods to the base line presented by the algorithms P-IW(1) and UNIFORM (which was chosen as a representative of the MCTS family as the most basic method). Two numbers in the *Naïve* approach represent the score that the algorithm that is based upon running the algorithm chosen by the classification, and the accuracy of the algorithm (the percentage where the classification was correct). If the score on both P-IW(1) and UNIFORM pre-test is equal (this mostly happens when none of

[3] Some games were evaluated over 15 episodes because of the lack of time

the algorithms could find any reward from the initial state), we choose randomly between the two.

The budget B allocated to each of the two iterations is 150000 (30000 nodes), this is tree times more than the budget allocated to each decision point.

Table 3 shows that the *Naïve* approach is a weak binary classifier (its accuracy is 62.8%), which means that the underlying assumption mentioned earlier is not entirely correct. The results of the classifier may be boosted however, by applying the tests to the game in question with different random seeds , and pick the algorithm that achieved most points in the majority of the runs (accuracy of this approach is 75.8%).

The experiments show that the ALTERNATION technique typically results in a score closer to the maximum between P-IW(1) and UNIFORM. In some games, however, it fails to achieve even a minimum score between them. These games can be divided into two groups. First – adversarial games, e.g.,PONG, TENNIS, or FISHING DERBY. The common denominator of this games is the presence of the negative rewards (in adversarial games negative rewards appear if the opponent is getting some positive rewards). In this setting, the disadvantage comes from the frame reuse of the algorithm. The subtrees produced by the P-IW(1) part and the UNIFORM part are mixed in one search tree. Abundance of the negative rewards makes longer trajectories more likely to yield negative score overall, which makes the algorithm choose an action starting the shorter trajectories, thus gravitating towards myopic decisions (and those aren't likely to be optimal). On the other hand, there are games like CRAZY CLIMBER and VIDEO PINBALL, in which the rewards are very sparse. That once again leads to a bias, but now towards longer rollouts of the UNIFORM algorithm. This leads us to conclusion, that in order to get a better version of this algorithm, there is a need in more balancing between the P-IW(1) and UNIFORM parts of the algorithm.

To summarize the experimental result on the ALTERNATION presented in Table 3. In most games the scores of the algorithm lie in-between the scores of P-IW(1) and UNIFORM with a slight bias towards the UNIFORM algorithm. However, on 15 games the algorithm scores more than the maximum between the two components, and on 44 games it scores more than minimum. An important finding is that in 6 games the algorithm scores more than 150% of the maximum of its components.

Summary

Black-box planning still presents a challenging task, since unlike in the classical planning domains, a black-box planner cannot rely on the off-the-shelf techniques that employ any kind of reasoning over propositional encoding of actions and goals. Previous works show that BRFS-like algorithms with pruning, such as IW(1) and P-IW(1) are setting the state-of-the-art performance, with the key to success being structural, similarity-based approximation of duplicate pruning (Lipovetzky *et al.* 2015). However, we have demonstrated that methods based on the Monte-Carlo simulations can be competitive in the majority of the Atari games, given

Game	P-IW(1)	UNIFORM	*Naïve*		ALTERNATION
ALIEN	13264	17240	13264	0.37	**31939**
AMIDAR	2041	932	2041	0.90	**2216**
ASSAULT	1551	**1831**	1774	0.73	1825
ASTERIX	**349041**	255875	349041	1.00	273283
ASTEROIDS	5548	**9986**	9053	0.77	8222
ATLANTIS	197450	**197546**	197163	0.40	196870
BANK HEIST	591	1792	1247	0.77	**2782**
BATTLE ZONE	3666	130400	130400	1.00	**259333**
BEAM RIDER	4868	14208	14208	0.97	**14922**
BERZERK	485	739	707	0.80	**1044**
BOWLING	63	**83**	81	0.93	78
BREAKOUT	**855**	832	832	0.79	849
CARNIVAL	5605	**6759**	6598	0.80	6682
CENTIPEDE	**186964**	136775	155046	0.37	149865
CHOPPER COMMAND	2140	**36480**	6283	0.03	26986
CRAZY CLIMBER	**140388**	90418	91614	0.04	76948
DEMON ATTACK	**38897**	30511	35572	0.60	33252
DOUBLE DUNK	-16	**24**	24	1.00	**24**
ELEVATOR ACTION	22503	**26136**	25918	0.97	25556
ENDURO	0	**405**	254	0.63	358
FISHING DERBY	18	**36**	29	0.63	-1
FREEWAY	**32**	7	32	1.00	31
FROSTBITE	**7667**	293	3572	0.50	994
GOPHER	28618	**29284**	29284	0.67	27812
GRAVITAR	1176	**5766**	5766	1.00	5448
HERO	5702	**13932**	9765	0.47	13858
ICE HOCKEY	14	**55**	39	0.60	**55**
JAMESBOND	40	13813	12149	0.87	**14883**
JOURNEY ESCAPE	2506	**86120**	5161	0.07	69900
KANGAROO	**4336**	2026	2026	0.00	3300
KRULL	**11443**	5240	8200	0.50	10193
KUNG FU MASTER	**79716**	63000	65735	0.17	64446
MONTEZUMA REVENGE	0	50	25	1.00	**536**
MS PACMAN	23583	30893	26245	0.47	**33229**
NAME THIS GAME	**16713**	13991	14391	0.23	14086
PONG	**21**	**21**	21	0.77	2
POOYAN	18926	**20358**	20235	0.93	20283
PRIVATE EYE	919	**1040**	1040	0.50	731
QBERT	21726	**38274**	29782	0.73	36189
RIVERRAID	**11702**	9607	11702	0.77	10045
ROAD RUNNER	62655	58982	62655	0.69	**127715**
ROBOTANK	6	**58**	54	0.93	**58**
SEAQUEST	**589**	498	621	0.63	350
SPACE INVADERS	2447	2602	2535	0.60	**2620**
STAR GUNNER	**1373**	1233	1320	0.33	1300
TENNIS	**24**	**24**	24	0.57	19
TIME PILOT	**54136**	53060	50800	0.43	53606
TUTANKHAM	134	**271**	157	0.23	252
UP N DOWN	73325	99037	76000	0.13	**99730**
VENTURE	33	10	21	0.73	**96**
VIDEO PINBALL	**592101**	340108	439210	0.33	323869
WIZARD OF WOR	115346	127666	124746	0.47	**132540**
ZAXXON	21553	**45046**	39385	0.77	38800
Best in	18	22		0.63	17

Table 3: Performance results over the 53 Atari 2600 games. The algorithms P-IW(1), UNIFORM, *Naïve*, and ALTERNATION are evaluated over 30 episodes for each game. The look ahead of every algorithm is limited to a budget of 50000 simulated frames. The maximum episode duration is 18000 frames. Numbers in bold show best performer in terms of average score. Average scores were rounded to a nearest integer

some reasonable adaptations to the setting. The empirical results show that all 4 algorithms proposed in this work significantly outperform UCT and are competitive with the state-of-the-art P-IW(1).

Empirical evaluation also shows us that the blind search and MCTS approaches excel in different games, and a composition of the two may yield a more consistent result on a previously unseen task. Further experiments with different types of composition tend to support this claim. We explore two types of composition: pre-sampling based selection among the complementary methods, and an alternation between them. Both composite methods perform with mixed success. It is possible that a smarter classification of games can give an insight what algorithm should be used on the problem at hand.

References

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The Arcade Learning Environment: An evaluation platform for general agents. *JAIR*, 47:253–279, 2013.

Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *Algorithmic Learning Theory, 20th International Conference, ALT 2009, Porto, Portugal, October 3-5, 2009. Proceedings*, pages 23–37, 2009.

Guillaume M JB Chaslot, Mark HM Winands, H JAAP VAN DEN Herik, Jos WHM Uiterwijk, and Bruno Bouzy. Progressive strategies for monte-carlo tree search. *New Mathematics and Natural Computation*, 4(03):343–357, 2008.

Pierre-Arnaud Coquelin and Rémi Munos. Bandit algorithms for tree search. In *UAI 2007, Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, Vancouver, BC, Canada, July 19-22, 2007*, pages 67–74, 2007.

Zohar Feldman and Carmel Domshlak. Monte-carlo planning: Theoretically fast convergence meets practical efficiency. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, UAI 2013, Bellevue, WA, USA, August 11-15, 2013*, 2013.

Zohar Feldman and Carmel Domshlak. On mabs and separation of concerns in monte-carlo planning for mdps. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*, 2014.

Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2013.

Yuu Jinnai and Alex S. Fukunaga. Learning to prune dominated action sequences in online black-box planning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 839–845, 2017.

Zohar Shay Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1238–1246, 2013.

Thomas Keller and Malte Helmert. Trial-based heuristic tree search for finite horizon mdps. In *ICAPS*, 2013.

L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *ECML*, pages 282–293, 2006.

N. Lipovetzky, M. Ramírez, and H. Geffner. Classical planning with simulators: Results on the Atari video games. pages 1610–1616, 2015.

Gabriele Röger and Malte Helmert. The more, the merrier: Combining heuristic estimators for satisficing planning. *Alternation*, 10(100s):1000s, 2010.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.

Tim Schulte and Thomas Keller. Balancing exploration and exploitation in classical planning. In *Proceedings of the Seventh Annual Symposium on Combinatorial Search, SOCS 2014, Prague, Czech Republic, 15-17 August 2014.*, 2014.

Alexander Shleyfman, Alexander Tuisov, and Carmel Domshlak. Blind search for atari-like online planning revisited. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 3251–3257, 2016.

Dennis JNJ Soemers, Chiara F Sironi, Torsten Schuster, and Mark HM Winands. Enhancements for real-time monte-carlo tree search in general video game playing. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pages 1–8. IEEE, 2016.

David Tolpin and Solomon Eyal Shimony. MCTS based on simple regret. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.