Action Abstractions for Combinatorial Multi-Armed Bandit Tree Search

Rubens O. Moraes
Departamento de Informática
Universidade Federal de Viçosa
Viçosa, Minas Gerais, Brazil

Julian R. H. Mariño
Instituto de Ciências Matemáticas e Computação
Universidade de São Paulo
São Carlos, São Paulo, Brazil

Levi H. S. Lelis
Departamento de Informática
Universidade Federal de Viçosa
Viçosa, Minas Gerais, Brazil

Mario A. Nascimento
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada

Abstract

Search algorithms based on combinatorial multi-armed bandits (CMABs) are promising for dealing with state-space sequential decision problems. However, current CMAB-based algorithms do not scale to problem domains with very large actions spaces, such as real-time strategy games played in large maps. In this paper we introduce CMAB-based search algorithms that use action abstraction schemes to reduce the action space considered during search. One of the approaches we introduce use regular action abstractions (A1N), while the other two use asymmetric action abstractions (A2N and A3N). Empirical results on μRTS show that A1N, A2N, and A3N are able to outperform an existing CMAB-based algorithm in matches played in large maps, and A3N is able to outperform all state-of-the-art search algorithms tested.

Introduction

Real-time strategy (RTS) games are challenging for planning systems because they present large action spaces and the time allowed for planning is very limited, typically in the order of milliseconds. Ontañón (2013; 2017a) proposed the use of combinatorial multi-armed bandits (CMAB) (Gai, Krishnamachari, and Jain 2010; Chen, Wang, and Yuan 2013; Combes et al. 2015) as a method for allowing Monte Carlo tree search (MCTS) algorithms (Browne et al. 2012) to focus their search on a set of promising actions; Ontañón's search algorithm was named NaïveMCTS. Although the CMAB approach helps the MCTS search to focus on a subset of actions, the number of actions to be evaluated during search grows quickly with the size of the map. As a result, NaïveMCTS does not scale to matches played in large maps.

In this paper we introduce three CMAB-based MCTS algorithms that leverage on the concept of action abstractions to further reduce the number of promising actions one has to consider during search. Action abstractions were originally introduced in the context of RTS combats by Churchill and Buro (2013), who used hard-coded scripts to induce action abstractions. A script is a function that, given a game state and a unit, returns which action the unit should perform in that state. Given a set of scripts, instead of searching over all possible actions, Churchill and Buro's search algorithm considers only the actions returned by the scripts. Moraes and Lelis (2018) introduced, also in the context of RTS combats, the concept of asymmetric actions abstractions. In contrast to Churchill and Buro's action abstractions, asymmetric abstractions restrict the number of actions available to a subset of units, while the search algorithm considers all available actions for the remaining units. Search algorithms using asymmetric abstractions can deliver a "finer control" to units deemed as more important at a given moment of the game by considering a greater number of actions for those units.

One of the CMAB-based algorithms we introduce uses regular actions abstractions. We call this algorithm A1N, where "A" stands for abstraction and "N" for naïve, since A1N is based on NaïveMCTS. We also introduce other two algorithms, which we call A2N and A3N, that use asymmetric action abstractions. A2N and A3N differ in the way that their abstractions are defined. A2N uses two sets of scripts to define its abstractions, one set is composed of economy-based scripts and is used to reduce the set of actions of economy-based units, and the other set is composed of combat-based scripts and is used to reduce the set of actions of military units. A3N uses Moraes and Lelis (2018)'s approach, i.e., it uses a single set of scripts to reduce the set of actions available during search for a subset of units; all actions of the remaining units are considered during search.

We evaluate our CMAB-based search algorithms that use action abstractions with an extensive set of experiments on μRTS. μRTS is a great testbed for our research because it offers an efficient forward model of the game, which is required by search-based approaches. Moreover, the game is much simpler than commercial video games, which allows us to quickly test different approaches. Finally, μRTS codebase[1] contains most of the current state-of-the-art search-based methods, thus facilitating our empirical evaluation.

The results of our experiments show that A1N, A2N, and A3N are all able to outperform NaïveMCTS. Our results also show that A3N is able to outperform all state-of-the-art algorithms tested, including those that competed in the latest μRTS competition (Ontañón et al. 2018).

Related Work

In an RTS combat each player controls a set of units that battles until one of the players is able to eliminate all the

[1] https://github.com/santiontanon/microrts

other player's units. Previous works proved action abstraction schemes to be effective in such scenarios (Churchill and Buro 2013; Justesen et al. 2014; Wang et al. 2016; Lelis 2017; Moraes and Lelis 2018). In this work we show that such abstractions can also be effective in the context of complete RTS matches. Also, previous works used simplified versions of the game in which features such as unit collision were disregarded. This distinction is important because scripts used to generate action abstractions are computationally more expensive in μRTS due to the need of running path-finding algorithms such as A* (Hart, Nilsson, and Raphael 1968). The time required to run the scripts subtracts from the very limited amount of time available for planning. We show that action abstractions can be valuable in the context of complete RTS matches despite their computational cost, as our experiments show that algorithms that use action abstractions are able to outperform un-abstracted ones.

Other methods for RTS games include non-CMAB MCTS approaches (Chung, Buro, and Schaeffer 2005; Sailer, Buro, and Lanctot 2007; Balla and Fern 2009) and Alpha-Beta pruning (Churchill, Saffidine, and Buro 2012). Similarly to NaïveMCTS, these methods do not scale to large maps due to the large number of actions available during search.

Scripts have also been used to guide the search by means other than action abstractions. Puppet Search (PS) (Barriga, Stanescu, and Buro 2017b) defines a search space over the parameter values of scripts. Strategy Tactics (STT) (Barriga, Stanescu, and Buro 2017a) combines PS's search in the script-parameter space with a NaïveMCTS search in the original state space for the combat units. Strategy Creation via Voting (SCV) generates scripts via voting (Silva et al. 2018), which are then used to play the game, showing the potential behind the scripts and how they can be combined to generated new and potentially powerful variations. In contrast with PS, STT, and SCV that generate novel scripts during the game, the algorithms we introduce in this paper use a set of scripts to generate action abstractions.

Although we tested our action abstraction schemes by introducing variants of NaïveMCTS, in principle, our schemes could be used to reduce the action space of other CMAB-based algorithms such as 2-Phase-CMAB approaches (Shleyfman, Komenda, and Domshlak 2014) and of search algorithms that use enhancements such as Hierarchical Expansion (Roelofs 2015).

Definitions and Notations

Following the notation of Ontañón (2017a), an RTS match can be described as a finite zero-sum two-player game with simultaneous-move and be denoted as $(P, S, A, \tau, L, W, s_{init})$, where:

- $P = \{max, min\}$ is the set of players.
- S is the match's set of states. A state $s \in S$ describes the match at a given moment. A state contains the position on the map of each unit controlled by the players as well as properties for these units, i.e., the current amount of hit points and if the units are currently performing an action.
- $A = A_{max} \times A_{min}$ is a set of joint player-actions, where A_{max} and A_{min} denote the player-actions of max and min, respectively. A player-action is represented by a vector of unit-actions, with one unit-action for each unit controlled by the player that is ready to act; a unit is not ready if it is already performing a unit-action.
- $\tau : S \times A_{max} \times A_{min} \rightarrow S$ is a deterministic transition function that, given a state in S and a set of joint player-actions, returns a resulting state in S.
- $L : S \times A \times P \rightarrow \{true, false\}$ is a function that given a state s, a player-action a and a player p, determines whether it is legal for p to execute a in s.
- $W : S \rightarrow P \cup \{draw, ongoing\}$ is a function that determines whether one of the players won the match, if the match finished in a draw, or if the match is still ongoing.
- s_{init} is the initial state of the match.

A decision point of player p is a state s in which p has at least one ready unit. A pure strategy is a function for player p mapping a state s to a player-action a. Although in general one might have to play a mixed strategy to optimize the player's payoffs in simultaneous move games (Gintis 2000), we follow the current state-of-the-art RTS methods (Barriga, Stanescu, and Buro 2017b; Lelis 2017; Churchill and Buro 2013; Churchill, Saffidine, and Buro 2012) and consider only pure strategies in this paper. In fact, we use state-space search algorithms to derive pure strategies in real time.

CMABs for RTS Games

Ontañón (2017a) modeled the search problem of deriving strategies for a player in an RTS match as a combinatorial multi-armed bandits (CMAB) problem. A CMAB problem can be defined by a tuple (X, μ). Here,

- $X = \{X_1, \cdots, X_n\}$, where each X_i is a variable that can assume K_i different values $\mathcal{X}_i = \{v_i^1, \cdots, v_i^{K_i}\}$, with $\mathcal{X} = \{(v_1, \cdots, v_n) \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_n\}$ being the possible combinations of value assignments for the variables in X; a value assignment $V \in \mathcal{X}$ is called a macro-arm.
- $\mu : \mathcal{X} \rightarrow \mathbb{R}$ is a reward function, that receives a macro-arm and returns a reward value for that macro-arm.

The goal in a CMAB problem is to find a macro-arm that maximizes the expected reward. This can be achieved by balancing exploration and exploitation until converging to an optimal macro-arm. In the context of RTS games, each decision point s can be cast as a CMAB problem in which X contains one variable for each ready unit of a player in s, thus a macro-arm $V \in \mathcal{X}$ represents a player-action and each value $v \in V$ represents a unit-action. The set $\mathcal{X}_i = \{v_i^1, \cdots, v_i^{K_i}\}$ represents the set of K_i legal actions for the i-th unit in X at s. Naturally, the goal is to find a macro-arm (player-action) that maximizes the player's reward. The reward is defined by a game-specific function.

Naïve Sampling

Since the number of macro-arms in $\mathcal{X}$ is often too large for decision points in RTS matches, Ontañón (2017a) derived a sampling procedure called Naïve Sampling (NS) to help deciding which macro-arms should be evaluated during search.

NS divides a CMAB problem with n variables in $n + 1$ multi-armed bandit (MAB) problems.

- n local MABs, one for each variable $X_i \in X$. That is, for variable X_i representing a unit, the arms of the MAB are the K_i values (unit-actions) in $\mathcal{X}_i$.
- 1 global MAB, denoted MAB_g, that treats each macro-arm V evaluated thus far in search as an arm in MAB_g. Naturally, MAB_g has no arms in the beginning of search.

At each iteration, NS uses a policy π_0 to determine whether it adds an arm to MAB_g through the local MABs (explore) or evaluates an existing arm in MAB_g (exploit).

1. If explore is chosen, then a macro-arm V is added to MAB_g by using a policy π_l to independently choose a value for each variable in X. Here, NS assumes that the reward of a macro-arm V can be approximated by the sum of the rewards of the individual values $v_i \in V$, denoted $\mu'(v_i)$. That is, $\mu(V) \approx \sum_{v_i \in V} \mu'(v_i)$.
2. If exploit is chosen, then a policy π_g is used to select an existing macro-arm in MAB_g.

In this paper we use ϵ-greedy for policies π_0, π_l, and π_g.

Ontañón (2017a) showed that NS can be used in the context of a MCTS algorithm to choose a subset of player-actions to be considered during search; the resulting algorithm was named NaïveMCTS. NaïveMCTS differs from other instantiations of MCTS algorithms in that it uses NS to decide which player-action to explore next. Instead of uniformly sampling which player-action to evaluate next as a vanilla MCTS algorithm would do, NaïveMCTS focuses through NS its search effort on player-actions composed of unit-actions that tend to yield good reward values. While NS allows NaïveMCTS to focus its search on a subset of promising player-actions, NaïveMCTS is still outperformed by search algorithms that rely on the guidance of scripts in matches played in large maps (Silva et al. 2018). Next, we describe how NaïveMCTS can leverage the domain knowledge encoded scripts through action abstractions.

Action-Abstracted NaïveMCTS (A1N)

An action abstraction for player p is a function mapping the set of legal actions A_p at a decision point s to a subset A'_p of A_p. Similarly to previous works (Churchill and Buro 2013; Wang et al. 2016; Lelis 2017; Moraes and Lelis 2018), we use a set of scripts $\mathcal{P}$ to define an abstraction. A script $\bar{\sigma}$ is a function that receives a decision point s and a unit u as input, and returns a legal unit-action m for u to perform at s. In an action abstraction, $M(s, u, \mathcal{P}) = \{\bar{\sigma}(s, u) | \bar{\sigma} \in \mathcal{P}\}$ is the set of unit-actions a unit can perform at s, and,

$$A'_p = \{(m_1, \cdots, m_n) \in M(s, u_1, \mathcal{P}) \times \cdots \times M(s, u_n, \mathcal{P})\}.$$

We call A1N a version of NaïveMCTS that uses an action abstraction induced by a set of scripts $\mathcal{P}$. The difference between NaïveMCTS and A1N is in the unit-actions sampled by NS while adding macro-arms to MAB_g. Instead of being able to sample from all legal unit-actions, A1N's NS is allowed to sample only from unit-actions in $M(s, u, \mathcal{P})$ for each unit u. As a consequence, the macro-arms (player-actions) added to MAB_g are restricted to those in A'_p.

Asymmetrically Action-Abstracted NaïveMCTS (A2N and A3N)

Regular action abstractions use the same set of scripts $\mathcal{P}$ to reduce the unit-actions available during search for all units controlled by the player. Moraes and Lelis (2018) introduced an abstraction scheme they called asymmetric action abstraction to allow a larger set of unit-actions to be available to a subset of key units (e.g., units that are engaged in combat). Moraes and Lelis's asymmetric abstractions are defined by selecting a subset of the player's units, called unrestricted units, for which all legal unit-actions would be accounted for during search, while all the other units (restricted ones) would have access only to the actions returned by the set $\mathcal{P}$.

In addition to Moraes and Lelis's asymmetrical scheme, we also consider the asymmetry produced by two sets of scripts $\mathcal{P}_e$ and $\mathcal{P}_b$, where $\mathcal{P}_e$ contains scripts for economy-related units (i.e., barracks, base, and workers) and $\mathcal{P}_b$ contains scripts for combat units (i.e., military units)—see Ontañón et al. (2018) for information about μRTS's units.

We call A2N the version of NaïveMCTS that searches in an asymmetrically-abstracted action space defined by two sets of scripts, one for economy units and one for combat units. We call A3N the version of NaïveMCTS that allows all legal unit-actions to the set of unrestricted units and only the actions returned by a set of scripts for all the other units.

A3N's Unrestricted Units

By changing the number of unrestricted units, which we denote by N, one can make A3N more similar or less similar to NaïveMCTS and A1N. That is, if all units are unrestricted, then A3N becomes NaïveMCTS; if no units are unrestricted, then A3C becomes A1C. In addition to the number of unrestricted units, one needs to define which units compose the unrestricted set. We test empirically several unrestricted set sizes and 7 strategies for selecting the unrestricted units, which are described below. The first four strategies are introduced in this paper (CE, FE, HP-, HP+), while the last three (AV+, AV-, R) were introduced by Moraes and Lelis (2018).

1. **Closest to an enemy (CE)**. CE selects the N units that are closest to an enemy unit at every decision point. The intuition behind CE is to allow a finer control for the units that are likely to be more threatened by enemy units.
2. **Farthest from an enemy (FE)**. FE selects the N units that are the farthest of an enemy unit at every decision point. The intuition is that by providing a finer control to units that are far of the enemy, one might allow a better overall positioning of the units.
3. **Less life (HP-)**. HP- selects the N units with the lowest hit points at every decision point. The intuition of this strategy is to provide a finer control to units that are about to be eliminated, hoping that a finer control will keep these units longer in the match.
4. **More life (HP+)**. HP+ selects the units with more hit points at a given decision point. We expect this strategy to be outperformed by HP- as we believe that providing a finer control to units with smaller rather than larger hit points yield better strategies, as explained above.

Str.	Unrestricted Set Size N									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
CE	76.9	71.5	65.6	57.6	53.2	40.1	36.9	30.7	27.8	24.4
FE	52.4	52.6	42.9	32.6	28.9	22.6	22.9	21.5	18.9	17.4
HP-	41.8	53.1	54.6	48.1	46.5	44.0	38.9	33.5	34.7	31.9
HP+	19.9	14.6	14.0	17.1	18.6	18.3	19.7	20.0	18.3	20.6
AV-	18.2	22.6	26.3	23.8	21.8	23.3	23.9	21.5	22.1	20.4
AV+	42.4	56.5	53.3	48.8	42.6	43.8	39.3	35.7	33.8	32.6
R	55.3	52.1	44.4	35.7	30.7	29.9	24.0	25.0	22.1	19.9

Table 1: Winning rate of variants of A3N against A1N in 40 matches played in 7 maps. The rows depict different strategies (Str.) and the columns different unrestricted set sizes (N). If a variant had won all 40 matches in all 7 maps, then it would have a winning rate of 100.0.

5. **More attack value (AV+)**. Let $av(u) = \frac{dpf(u)}{hp(u)}$, where $dpf(u)$ is the amount of damage per game cycle a unit can inflict to an enemy unit and $hp(u)$ is u's current amount of hit points. AV+ selects the N units with the largest av-values at every decision point. AV+ is similar than HP- as they both provide a finer control to units with low hp-value, however AV+ selects units with low hp-value and/or large dpf-value, while HP- only accounts for hp. Moraes and Lelis (2018) showed that AV+ yields the best results for combat scenarios that arise in RTS matches.
6. **Less attack value (AV-)**. AV- selects the units with smaller av-values. We expect this strategy to be outperformed by AV+, as we believe that is better to provide a finer control to units with smaller hp and larger dpf, instead of the opposite, as explained above.
7. **Random (R)**. R randomly selects N units in the beginning of the match to be the unrestricted units. R replaces an unrestricted unit that has its hp-value reduced to zero by a randomly selecting a restricted unit. The R strategy serves as a baseline.

Empirical Evaluation

Our empirical evaluation of A1N, A2N and A3N is divided into two parts. In the first set of experiments we test A3N with unrestricted sets of size N varying from 1 to 10 for each of the 7 strategies for selecting the unrestricted units described above. The goal of the first experiment is to find which combination of set sizes and selection strategies work well in the context of μRTS. We test A3N against A1N, which is a special case of A3N and thus a natural competitor. In this experiment we use 7 maps, including maps of size 8×8, 9×8, 16×16, 24×24, 32×32 and 64×64.

In the second set of experiments we test A1N, A2N and A3N against state-of-the-art search methods for RTS games. Namely, we test the following algorithms: Adversarial Hierarchical Task Network (AHT) (Ontañón and Buro 2015), an algorithm that uses Monte Carlo tree search and HTN planning; NaiveMCTS (Ontañón 2017b) (henceforth referred as NAV); the MCTS version of Puppet Search (PS) (Barriga, Stanescu, and Buro 2017b), Strategy Tactics (STT) (Barriga, Stanescu, and Buro 2017a), and two hard-coded scripts focused in rush, called Light rush (LR), and Worker Rush (WR) (Stanescu et al. 2016). In all experiments the algorithms are allowed 100 milliseconds of planning time for each decision point. In addition to the 7 maps used in the first experiment, in this experiment we added 11 other maps: one of size 24×24 and another of size 32×32, the remaining 9 maps are from Blizzard's StarCraft, with four being of size 64×64, one of size 96×128, and four of size 128×128.

Every match of both experiments is limited by a number of game cycles and the match is considered a draw once the limit is reached. The maximum number of game cycles is dependent on the map. We use the limits defined by Barriga et al. (Barriga, Stanescu, and Buro 2017b): 3000, 3000, 4000, 5000, 6000, 8000, 12000 and 12000 game cycles for maps of width 8, 9, 16, 24, 32, 64, 96 and 128 respectively. Each tested algorithm plays against every other algorithm 40 times in each map tested. To ensure fairness, the players switch their starting location on the map an even number of times. For example, if Algorithm 1 starts in location X with Algorithm 2 starting in location Y for 20 matches; we switch the starting positions for the remaining 20 matches.

The evaluation function used with A1N, A2N and A3N was a random play-out of 100 game cycles of length (approximately 10 actions for each player in the game). The play-out evaluates a state s by simulating the game forward from s for 100 game cycles with both players choosing actions randomly, until reaching a state s'. Then, the evaluation of s is given by $\Phi(s')$, where Φ is a function introduced by Ontañón (2017b). Φ computes a score for each player—$score(max)$ and $score(min)$—by summing up the cost in resources required to train each unit controlled by the player weighted by the square root of the unit's hit points. The Φ value of a state is given by player max's score minus player min's score. Φ is then normalized to a value in $[-1, 1]$ through the following formula $\frac{2 * score(max)}{score(min)+score(max)} - 1$.

The set of scripts used by A1N and A3N was composed by light rush (LR), heavy rush (HR), and ranged rush (RR) (Stanescu et al. 2016; Silva et al. 2018). These scripts train units which are immediately sent to attack the enemy. The difference among them is the type of unit trained, LR trains light units, HR trains heavy units, and RR trains ranged units. A2N uses LR, HR, and RR as its economy-based set of scripts, and NOKAV, Kiter (Churchill and Buro 2013), and Cluster (Lelis 2017), as its combat scripts. NOKAV is a strategy that chooses an attack action that will not cause more damage than that required to eliminate the enemy unit from the match; Kiter allows the units to move away from the enemy, and Cluster groups the units together. All experiments were run on 2.1 GHz CPUs.

First Experiment: A3N vs A1N

Table 1 shows the winning rate of each A3N variant against A1N in 40 matches played in each of the 7 maps. The winning rate is computed by summing the total number of victories and half of the number of draws of a variant of A3N and then dividing this sum by the total number of matches played; the result of the division is then multiplied by 100. For example, if a variant had won all 40 matches in all 7

	A3N	SCV	PS	STT	LR	A2N	WR	A1N	AHT	NAV
Total	1,125	1,044	931	860	729	686	680	661	311	198

Table 2: Total number of victories by each method in 18 maps (possible maximum of 1,620); draws are disregarded.

maps, then it would have a winning rate of 100.0. The rows show the strategies used for selecting the unrestricted set while the columns show the size of the set. We highlight the cells in which A3N had a winning rate larger than 50.0.

A3N tends to perform better with smaller unrestricted set sizes ($N \leq 5$). This is because the number of macro-arms that can be sampled by A3N grows quickly as one increases N, making it harder for the search procedure to encounter macro-actions that result in strong gameplay. As anticipated, the strategies HP+ and AV- were outperformed by their counterparts HP- and AV+. In contrast with the results of Moraes and Lelis (2018) for combat scenarios, the best performing strategy for playing full RTS matches is CE with $N = 1$, which had a winning rate of 76.9, and not AV+, which with its best N-value had a winning rate of 56.5. We hypothesize that this result is related to the characteristics of the game. Compared to commercial games such as StarCraft, all units in μRTS have low hit point values, thus not lasting long in the match. If preserving units longer in the match is a good strategy, then one should probably provide a fine control to the units closest to the enemy, rather than the units with largest AV+ value as the former are more likely to be eliminated than the latter.

Second Experiment: State-of-the-Art Comparisons

Table 2 shows the total number of matches won by each approach tested in all 18 maps; matches finishing in draws are not included in these results. The maximum achievable number of victories is $18 \times 9 \times 10 = 1,620$ (18 maps, 9 opponents, and 10 matches played against each opponent). Overall, A3N wins more matches than any approach tested, suggesting that if a large diversity of maps are considered, then A3N is the current state-of-the-art in μRTS matches. It is also interesting to observe how the NaïveMCTS approaches perform with respect to each other methods. NaïveMCTS (denoted as NAV) wins a total of 231 matches, being the method that wins the least amount of matches. A1N, which uses an action abstraction scheme, is already substantially better than NAV. The employment of an asymmetric abstraction through two sets of scripts further improves the results as A2N has more victories than A1N, but they are still behind some of the state-of-the-art approaches. The advantage of A2N over A1N is that the former allows a finer control over the combat units with the scripts NOKAV, Kiter, and Cluster. The best results are achieved when NaïveMCTS is combined with an asymmetric action abstraction that allows the unit closest to an enemy unit to consider all its legal moves in a given decision point. These results show that NaïveMCTS is able to leverage the knowledge encoded in scripts through abstraction schemes.

It is interesting to note how A3N's search procedure relates to how humans play RTS games. Professional players often control one unit at a time when optimizing for battles, and they often switch which unit will receive a finer control. A3N acts similarly as it provides a coarse abstraction to all units but the unit closest to an enemy unit, for which all actions are accounted for during search. Moreover, the strategy of selecting the unit that is closest to an enemy unit also allows A3N to quickly switch, depending on the game scenario, which unit will receive a finer control.

Table 3 shows the winning rate of the row method against the column method in maps of different sizes. The numbers shown in the table are truncated to one decimal place. We highlight the background of a cell of A1N, A2N, and A3N if they have a winning rate greater or equal to 50.0 against an opponent. We also highlight the cell with the highest entry in the "Avg" column, which shows the average winning rate of the row method in a given map size. This table allows ones to understand the circumstances in which a method is stronger than another. For example, NAV is one of the strongest methods in maps of size 8×8 because the number of macro-arms available to be sampled by NAV is much smaller in matches played in smaller maps, which allows NAV to effectively find good player-actions within the game's time constraint. However, as one increases the size of the maps, the performance of NAV drops quickly. For example, it drops from 77.2 in the smallest maps to 27.8 in maps of size 16×16.

In contrast with NAV, the action-abstracted approaches, specially A3N, present similar performance across different maps. The worst performances of A3N are in matches played on maps of size 8×8 and 32×32. The reason of A3N's lower performance on these maps is related to their "actual size". Although maps 32×32 are comparatively larger than maps of size 16, 24, one of the maps of size 32 has the players starting the game close to each other, similarly to what happens in a map of size 8, which favors the LR strategy. A3N wins almost all the other matches played on the other map of size 32, where the players start far apart. The difference between NAV and A3N suggests a direction of future research, which is to develop adaptive methods for creating asymmetric abstractions. Ideally, A3N would detect that it can play more similarly to NAV in smaller maps.

Another interesting point is the poor performance of A3N against STT and SCV in the largest maps. A1N, A2N, and A3N are restricted to the high-level strategies encoded in their set of scripts (e.g., the number of barracks to be built, whether or not to expand). By contrast, STT and SCV try to expand through search and voting, respectively, the high-level strategies encoded in the scripts. Note that one could also use the high-level strategies generated by STT and SCV to define the action abstractions used with A1N, A2N, and A3N, possibly further improving their results.

Although A3N is the best average performing method only in maps of size 9×8 and 24×24, it is close to the

Maps 8 × 8

	NAV	AHT	WR	LR	STT	SCV	PS	A1N	N2S	A3N	Avg.
NAV	-	100	20.0	100	65.0	30.0	100	100	100	80.0	77.2
AHT	0.0	-	10.0	100	20.0	0.0	100	100	100	30.0	51.1
WR	80.0	90.0	-	100	85.0	100	100	100	100	90.0	93.9
LR	0.0	0.0	0.0	-	10.0	0.0	50.0	60.0	90.0	10.0	24.4
STT	35.0	80.0	15.0	90.0	-	40.0	100	100	100	80.0	71.1
SCV	70.0	100	0.0	100	60.0	-	100	100	100	90.0	80.0
PS	0.0	0.0	0.0	50.0	0.0	0.0	-	100	70.0	0.0	24.4
A1N	0.0	0.0	0.0	40.0	0.0	0.0	0.0	-	60.0	0.0	11.1
A2N	0.0	0.0	0.0	10.0	0.0	0.0	30.0	40.0	-	0.0	8.9
A3N	20.0	70.0	10.0	90.0	20.0	10.0	100	100	100	-	57.8

Maps 9 × 8

	NAV	AHT	WR	LR	STT	SCV	PS	A1N	N2S	A3N	Avg.
NAV	-	100	100	40.0	50.0	100	0.0	10.0	0.0	0.0	44.4
AHT	0.0	-	50.0	25.0	40.0	50.0	0.0	0.0	0.0	0.0	18.3
WR	0.0	50.0	-	25.0	50.0	50.0	0.0	0.0	0.0	0.0	19.4
LR	60.0	75.0	75.0	-	35.0	50.0	65.0	10.0	0.0	0.0	41.1
STT	50.0	60.0	50.0	65.0	-	55.0	20.0	0.0	0.0	10.0	34.4
SCV	0.0	50.0	50.0	50.0	45.0	-	0.0	0.0	0.0	5.0	22.2
PS	100	100	100	35.0	80.0	100	-	10.0	20.0	10.0	61.7
A1N	90.0	100	100	90.0	100	100	90.0	-	60.0	35.0	85.0
A2N	100	100	100	100	100	100	80.0	40.0	-	40.0	84.4
A3N	100	100	100	100	90.0	95.0	90.0	65.0	60.0	-	88.9

Maps 16 × 16

	NAV	AHT	WR	LR	STT	SCV	PS	A1N	N2S	A3N	Avg.
NAV	-	100	40.0	0.0	7.5	52.5	45.0	5.0	0.0	0.0	27.8
AHT	0.0	-	22.5	50.0	25.0	75.0	75.0	55.0	50.0	30.0	42.5
WR	60.0	77.5	-	50.0	25.0	100	75.0	50.0	50.0	27.5	57.2
LR	100	50.0	50.0	-	72.5	50.0	95.0	75.0	90.0	65.0	71.9
STT	92.5	75.0	75.0	27.5	-	75.0	80.0	50.0	55.0	55.0	65.0
SCV	47.5	25.0	0.0	50.0	25.0	-	75.0	50.0	50.0	40.0	40.3
PS	55.0	25.0	25.0	5.0	20.0	25.0	-	60.0	42.5	35.0	32.5
A1N	70.0	50.0	40.0	25.0	25.0	40.0	42.5	-	15.0	10.0	35.3
A2N	100	45.0	50.0	5.0	35.0	50.0	55.0	35.0	-	40.0	46.1
A3N	100	70.0	72.5	40.0	70.0	60.0	72.5	55.0	50.0	-	65.6

Maps 24 × 24

	NAV	AHT	WR	LR	STT	SCV	PS	A1N	N2S	A3N	Avg.
NAV	-	100	45.0	0.0	10.0	2.5	0.0	7.5	5.0	0.0	18.9
AHT	0.0	-	0.0	35.0	0.0	25.0	47.5	55.0	37.5	2.5	22.5
WR	55.0	100	-	50.0	65.0	62.5	30.0	60.0	80.0	60.0	62.5
LR	100	65.0	50.0	-	70.0	50.0	35.0	50.0	40.0	27.5	54.2
STT	90.0	100	35.0	30.0	-	57.5	47.5	25.0	27.5	12.5	47.2
SCV	97.5	75.0	37.5	50.0	42.5	-	20.0	70.0	90.0	42.5	58.3
PS	100	52.5	70.0	65.0	52.5	80.0	-	52.5	30.0	10.0	56.9
A1N	92.5	45.0	40.0	50.0	75.0	30.0	47.5	-	30.0	12.5	46.9
A2N	95.0	62.5	20.0	60.0	72.5	10.0	70.0	70.0	-	15.0	52.8
A3N	100	97.5	40.0	72.5	87.5	57.5	90.0	87.5	85.0	-	79.7

Maps 32 × 32

	NAV	AHT	WR	LR	STT	SCV	PS	A1N	N2S	A3N	Avg.
NAV	-	50.0	22.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8.1
AHT	50.0	-	10.0	0.0	0.0	0.0	40.0	50.0	45.0	40.0	26.1
WR	77.5	90.0	-	25.0	32.5	25.0	50.0	50.0	50.0	40.0	48.9
LR	100	100	75.0	-	100	50.0	75.0	75.0	45.0	75.0	77.2
STT	100	100	67.5	0.0	-	10.0	80.0	30.0	35.0	60.0	53.6
SCV	100	100	75.0	50.0	90.0	-	95.0	57.5	45.0	60.0	74.7
PS	100	60.0	50.0	25.0	20.0	5.0	-	30.0	20.0	32.5	38.1
A1N	100	50.0	50.0	25.0	70.0	42.5	70.0	-	20.0	55.0	53.6
A2N	100	55.0	50.0	55.0	65.0	55.0	80.0	80.0	-	55.0	66.1
A3N	100	60.0	60.0	25.0	40.0	40.0	67.5	45.0	45.0	-	53.6

Maps 64 × 64

	NAV	AHT	WR	LR	STT	SCV	PS	A1N	N2S	A3N	Avg.
NAV	-	3.8	0.0	0.0	0.0	2.5	0.0	0.0	0.0	0.0	0.7
AHT	96.3	-	5.0	61.3	0.0	12.5	15.0	45.0	38.8	15.0	32.1
WR	100	95.0	-	62.5	32.5	12.5	32.5	50.0	41.3	48.8	52.8
LR	100	38.8	37.5	-	15.0	12.5	0.0	62.5	45.0	15.0	36.3
STT	100	100	67.5	85.0	-	56.3	40.0	86.3	67.5	36.3	71.0
SCV	97.5	87.5	87.5	87.5	43.8	-	13.8	90.0	81.3	28.8	68.6
PS	100	85.0	67.5	100	60.0	86.3	-	100	93.8	32.5	80.6
A1N	100	55.0	50.0	37.5	13.8	10.0	0.0	-	30.0	12.5	34.3
A2N	100	61.3	8.8	55.0	32.5	18.8	6.3	70.0	-	0.0	44.7
A3N	100	85.0	51.3	85.0	63.8	71.3	67.5	87.5	100	-	79.0

Maps 96 × 128

	NAV	AHT	WR	LR	STT	SCV	PS	A1N	N2S	A3N	Avg.
NAV	-	50.0	0.0	0.0	0.0	0.0	0.0	0.0	25.0	0.0	8.3
AHT	50.0	-	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.6
WR	100	100	-	0.0	50.0	0.0	0.0	0.0	0.0	0.0	27.8
LR	100	100	100	-	30.0	0.0	0.0	50.0	25.0	0.0	45.0
STT	100	100	50.0	70.0	-	0.0	80.0	100	100	80.0	75.6
SCV	100	100	100	100	100	-	100	100	100	70.0	96.7
PS	100	100	100	100	20.0	0.0	-	100	90.0	20.0	70.0
A1N	100	100	100	50.0	0.0	0.0	0.0	-	45.0	0.0	43.9
A2N	75.0	100	100	75.0	0.0	0.0	10.0	55.0	-	0.0	46.1
A3N	100	100	100	100	20.0	30.0	80.0	100	100	-	81.1

Maps 128 × 128

	NAV	AHT	WR	LR	STT	SCV	PS	A1N	N2S	A3N	Avg.
NAV	-	23.1	0.0	0.0	1.3	1.3	0.0	0.0	17.5	0.0	4.8
AHT	76.9	-	2.5	30.6	7.5	6.3	7.5	22.5	24.4	7.5	20.6
WR	100	97.5	-	31.3	37.5	6.3	16.3	25.0	20.6	24.4	39.9
LR	100	69.4	68.8	-	28.8	6.3	2.5	58.8	36.9	10.0	42.4
STT	98.8	92.5	62.5	71.3	-	29.4	22.5	81.9	76.3	33.1	63.1
SCV	98.8	93.8	93.8	93.8	70.6	-	40.6	88.8	90.6	61.9	81.4
PS	100	92.5	83.8	97.5	77.5	59.4	-	100	96.3	35.0	82.4
A1N	100	77.5	75.0	41.3	18.1	11.3	0.0	-	38.8	6.9	41.0
A2N	82.5	75.6	79.4	63.1	23.8	9.4	3.8	61.3	-	0.0	44.3
A3N	100	92.5	75.6	90.0	66.9	38.1	65.0	93.1	100	-	80.1

Table 3: Winning rate of the row player against the column player. The winning rate is computed by summing the total number of victories and half of the number of draws of the row player against a column player, and then dividing this sum by the total number of matches played; the result of the division is then multiplied by 100. If a method wins all matches, then it has a winning rate of 100. We highlight the cells in which A1N, A2N or A3N has a winning rate greater or equal to 50.0.

best performing method in most of the other maps. Thus, overall, A3N has the largest number of wins, as shown in Table 2.

Conclusions and Future Work

In this paper we have introduced three CMAB-based search algorithms that use action abstraction schemes for RTS games, A1N, A2N, and A3N. Our algorithms combine the guidance provided by scripts through action abstractions, the sampling procedure given by a CMAB approach, and MCTS search. An extensive set of experiments on μRTS, a simple and yet complete RTS game, shows that A1N, A2N, and A3N are able to outperform NaïveMCTS, the CMAB-based algorithm that inspired us to do our research. Our experiments also show that A3N, which uses an asymmetric action abstraction scheme, is able to outperform all state-of-the-art approaches tested in our experiment, which includes the algorithms that competed in the latest μRTS competition (Ontañón et al. 2018). Our algorithms and results suggest several directions of future research to further improve the current state of the art, which includes the development of stronger scripts for inducing better action abstractions as well as asymmetric action abstraction schemes that automatically adjust the size of the unrestricted set.

Acknowledgements

This research was partially supported by FAPEMIG, CNPq and CAPES, Brazil and NSERC, Canada, and done while R.O. Moraes and L.H.S. Lelis were visiting the University of Alberta. The authors thank the great suggestions provided by the anonymous reviewers.

References

Balla, R.-K., and Fern, A. 2009. UCT for tactical assault planning in real-time strategy games. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 40–45.

Barriga, N. A.; Stanescu, M.; and Buro, M. 2017a. Combining strategic learning and tactical search in real-time strategy games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 9–15. AAAI.

Barriga, N. A.; Stanescu, M.; and Buro, M. 2017b. Game tree search based on non-deterministic action scripts in real-time strategy games. *IEEE Transactions on Computational Intelligence and AI in Games*.

Browne, C.; Powley, E. J.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Liebana, D. P.; Samothrakis, S.; and Colton, S. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI* 4(1):1–43.

Chen, W.; Wang, Y.; and Yuan, Y. 2013. Combinatorial multi-armed bandit: General framework and applications. In *International Conference on Machine Learning*, 151–159.

Chung, M.; Buro, M.; and Schaeffer, J. 2005. Monte Carlo planning in RTS games. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*. IEEE.

Churchill, D., and Buro, M. 2013. Portfolio greedy search and simulation for large-scale combat in StarCraft. In *Proceedings of the Conference on Computational Intelligence in Games*, 1–8. IEEE.

Churchill, D.; Saffidine, A.; and Buro, M. 2012. Fast heuristic search for RTS game combat scenarios. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. AAAI.

Combes, R.; Shahi, M. S. T. M.; Proutiere, A.; et al. 2015. Combinatorial bandits revisited. In *Advances in Neural Information Processing Systems*, 2116–2124.

Gai, Y.; Krishnamachari, B.; and Jain, R. 2010. Learning multiuser channel allocations in cognitive radio networks: A combinatorial multi-armed bandit formulation. In *New Frontiers in Dynamic Spectrum, 2010 IEEE Symposium on*, 1–9. IEEE.

Gintis, H. 2000. *Game Theory Evolving: A Problem-centered Introduction to Modeling Strategic Behavior*. Economics / Princeton University Press. Princeton University Press.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2):100–107.

Justesen, N.; Tillman, B.; Togelius, J.; and Risi, S. 2014. Script- and cluster-based UCT for StarCraft. In *IEEE Conference on Computational Intelligence and Games*, 1–8. IEEE.

Lelis, L. H. S. 2017. Stratified strategy selection for unit control in real-time strategy games. In *International Joint Conference on Artificial Intelligence*, 3735–3741.

Moraes, R. O., and Lelis, L. H. S. 2018. Asymmetric action abstractions for multi-unit control in adversarial real-time games. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 876–883. AAAI.

Ontañón, S., and Buro, M. 2015. Adversarial hierarchical-task network planning for complex real-time games. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1652–1658.

Ontañón, S.; Barriga, N. A.; Silva, C. R.; Moraes, R. O.; and Lelis, L. H. S. 2018. The first microrts artificial intelligence competition. *AI Magazine* 39(1):75–83.

Ontañón, S. 2013. The combinatorial multi-armed bandit problem and its application to real-time strategy games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 58–64. AAAI.

Ontañón, S. 2017a. Combinatorial multi-armed bandits for real-time strategy games. *Journal of Artificial Intelligence Research* 58:665–702.

Ontañón, S. 2017b. Combinatorial multi-armed bandits for real-time strategy games. *Journal of Artificial Intelligence Research* 58:665–702.

Roelofs, G.-J. 2015. Action Space Representation in Combinatorial Multi-Armed Bandits. Master's thesis, Maastricht University, The Netherlands.

Sailer, F.; Buro, M.; and Lanctot, M. 2007. Adversarial planning through strategy simulation. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 80–87. IEEE.

Shleyfman, A.; Komenda, A.; and Domshlak, C. 2014. On combinatorial actions and cmabs with linear side information. In *European Conference on Artificial Intelligence*, 825–830.

Silva, C. R.; Moraes, R. O.; Lelis, L. H. S.; and Gal, Y. 2018. Strategy generation for multi-unit real-time games via voting. *IEEE Transactions on Games*.

Stanescu, M.; Barriga, N. A.; Hess, A.; and Buro, M. 2016. Evaluating real-time strategy game states using convolutional neural networks. In *Proceedings of the IEEE Conference on Computational Intelligence and Games*, 1–7. IEEE.

Wang, C.; Chen, P.; Li, Y.; Holmgård, C.; and Togelius, J. 2016. Portfolio online evolution in StarCraft. In *Proceedings of the Conference on Artificial Intelligence and Interactive Digital Entertainment*, 114–120. AAAI.