Asymmetric Action Abstractions for Planning in Domains with Very Large Action Spaces and Real-Time Constraints

Rubens O. Moraes
Departamento de Informática
Universidade Federal de Viçosa - Campus Viçosa
Viçosa - Minas Gerais - Brazil
rubens.moraes@ufv.br

Levi H. S. Lelis
Departamento de Informática
Universidade Federal de Viçosa - Campus Viçosa
Viçosa - Minas Gerais - Brazil
levi.lelis@ufv.br

***Abstract*—Action abstractions restrict the number of legal actions available for real-time planning in multi-unit zero-sum extensive-form games, thus allowing algorithms to focus their search on a set of promising actions. Even though unabstracted game trees can lead to optimal policies, due to real-time constraints and the tree size, they are not a practical choice. In this context, we introduce an action abstraction scheme we call asymmetric abstraction. Similarly to unabstracted spaces, asymmetrically-abstracted spaces can have theoretical advantages over regularly abstracted spaces while still allowing search algorithms to derive effective strategies in practice, even in large-scale games. Further, asymmetric abstractions allow search algorithms to "pay more attention" to some aspects of the game by unevenly dividing the algorithm's search effort amongst different aspects of the game. We also introduce four algorithms that search in asymmetrically-abstracted game trees to evaluate the effectiveness of our abstraction schemes. An extensive set of experiments in a real-time strategy game developed for research purposes shows that search algorithms using asymmetric abstractions are able to outperform all other search algorithms tested.**

***Index Terms*—Asymmetric abstractions, real-time constraints, planning.**

I. Introduction

In real-time strategy (RTS) games the player controls a number of units to collect resources, build structures, and battle the opponent. RTS games are excellent testbeds for Artificial Intelligence methods because they offer fast-paced environments, where players act simultaneously, and the number of actions grows exponentially with the number of units the player controls. Also, the time allowed for planning is on the order of milliseconds. In this paper we assume two-player deterministic games in which all units are visible to both players.

A successful family of algorithms for planning in real time in RTS games uses what we call action abstractions to reduce the number of legal actions available to the player. Action abstractions reduce the number of legal actions a player can perform by reducing the number of legal actions each unit can perform. We use the word "action" if it is clear that we are referring to a player's or to a unit's action; we write "player action" or "unit-action" otherwise. For instance, Churchill and Buro [1] introduced a method for building action abstractions through what they called scripts. A script $\bar{\sigma}$ is a function mapping a game state s and a unit u to an action m for u. A set of scripts $\mathcal{P}$ induces an action abstraction by restricting the set of legal actions of all units to actions returned by the scripts in $\mathcal{P}$. We call an action abstraction generated with Churchill and Buro's scheme a uniform abstraction.

We introduce an action abstraction scheme we call asymmetric action abstractions (or asymmetric abstractions for short). In contrast with uniform abstractions that restrict the number of actions of all units, asymmetric abstractions restrict the number of actions of only a subset of units. We show that asymmetric abstractions can retain the unabstracted spaces' theoretical advantage over uniformly abstracted ones while still allowing algorithms to derive effective strategies in practice, even in large games.

Another contribution we offer is the introduction of four general-purpose algorithms that search in asymmetrically-abstracted trees: Greedy Alpha-Beta Search (GAB), Stratified Alpha-Beta Search (SAB), and two variants of Asymmetrically Action-Abstracted NaïveMCTS, denoted as A2N and A3N. GAB and SAB are based on Alpha-Beta pruning, Portfolio Greedy Search (PGS) [1], and Stratified Strategy Selection (SSS) [2]. PGS and SSS are algorithms designed for searching in uniformly-abstracted spaces. The other two algorithms, A2N and A3N, are based on NaïveMCTS, a search algorithm that uses combinatorial multi-armed bandits (CMAB) [3] to search in unabstracted spaces. In addition to the two variants of NaïveMCTS that search in asymmetrically-abstracted trees, we also introduce a NaïveMCTS baseline that, similarly to PGS and SSS, searches in uniformly-abstracted trees; we call this baseline A1N.

We evaluate our algorithms in μRTS [3], an RTS game developed for research purposes. 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.

Finally, μRTS codebase[1] contains most of the current state-of-the-art search-based methods, including the systems used in μRTS's competitions [4], thus facilitating our empirical evaluation. An extensive set of experiments show that our algorithms that search in asymmetrically-abstracted trees are able to outperform in terms of matches won their counterparts that search unabstracted and uniformly-abstracted trees.

Although we present our abstraction schemes and search algorithms in the context of RTS games, we believe our ideas and algorithms to also be applicable in other scenarios. For example, a robotic system, which controls several actuators simultaneously while trying to accomplish a task, can benefit from asymmetric abstractions. This is because some actuators might require a finer control than the others. To illustrate, the actuators controlling the arms of a robot planning a sequence of actions to open a door might need a "finer plan" than the actuators controlling the wheels of the robot, given that the robot is already in front of the door to be opened. Asymmetric action abstractions offer an approach that allows the planning system to focus on the arms of the robot rather than on its wheels. As another example, a search algorithm for uniformly-abstracted trees is the core of the artificial player of the commercial card game Prismata [5]. The idea we introduce in this work of performing search in asymmetric trees can also be used in such card games to enhance the strength of their artificial player. For example, the artificial player could benefit from an algorithm that discovers finer plans for the "more important" cards.

II. Zero-Sum Extensive-Form Games

RTS games can be described as zero-sum extensive-form games with simultaneous and durative actions, defined by a tuple $\nabla = (\mathcal{N}, \mathcal{S}, s_{init}, \mathcal{A}, \mathcal{B}, \mathcal{R}, \mathcal{T})$, where, $\mathcal{N} = \{i, -i\}$ is the set of **players** (i is the player we control and $-i$ is our opponent). $\mathcal{S} = \mathcal{D} \cup \mathcal{F}$ is the set of **states**. Every state $s \in \mathcal{S}$ defines a joint set of **units** $\mathcal{U}^s = \mathcal{U}^s_i \cup \mathcal{U}^s_{-i}$, for players i and $-i$. $\mathcal{A} = \mathcal{A}_i \times \mathcal{A}_{-i}$ is the set of **joint actions**. $\mathcal{A}_i(s)$ is the set of legal **actions** player i can perform at state s. Each action $a \in \mathcal{A}_i(s)$ is denoted by a vector of n **unit-actions** $(m_1, \cdots, m_n)$, where $m_k \in a$ is the action of the k-th **ready unit** of player i. A unit is not ready if it is already performing an action (unit-actions can have different durations). We denote the set of ready units of players i and $-i$ as $\mathcal{U}^s_{i,r}$ and $\mathcal{U}^s_{-i,r}$. We denote the set of unit-actions as $\mathcal{M}$. We write $\mathcal{M}(s, u)$ to denote the set of legal actions of unit u at s. $\mathcal{R}_i : \mathcal{F} \to \mathbb{R}$ is a **utility function** with $\mathcal{R}_i(s) = -\mathcal{R}_{-i}(s)$, for any $s \in \mathcal{F}$. The **transition function** $\mathcal{T} : \mathcal{S} \times \mathcal{A}_i \times \mathcal{A}_{-i} \to \mathcal{S}$ deterministically determines the sucessor state for a state s and the set of joint actions taken at s.

III. Uniform Abstractions

We define a **uniform action abstraction** (or uniform abstraction for short) for player i as a function mapping the set of legal actions $\mathcal{A}_i$ to a subset $\mathcal{A}'_i$ of $\mathcal{A}_i$. Action abstractions can be constructed from a set of scripts $\mathcal{P}$. Let the action-abstracted legal actions of unit u at state s be the actions for u that is returned by a script in $\mathcal{P}$, defined as,

$$\mathcal{M}(s, u, \mathcal{P}) = \{\bar{\sigma}(s, u) | \bar{\sigma} \in \mathcal{P}\}.$$

Definition 1: A uniform abstraction Φ is a function that receives a state s, a player i, and a set of scripts $\mathcal{P}$. Φ returns a subset of $\mathcal{A}_i(s)$ denoted $\mathcal{A}'_i(s)$. $\mathcal{A}'_i(s)$ is defined by the Cartesian product of actions in $\mathcal{M}(s, u, \mathcal{P})$ for all u in $\mathcal{U}^s_{i,r}$, where $\mathcal{U}^s_{i,r}$ is the set of ready units of i in s.

Algorithms using a uniform abstraction search in a game space for which player i's legal actions are limited to $\mathcal{A}'_i(s)$ for all s. This way, algorithms focus their search on actions deemed as promising by the scripts in $\mathcal{P}$, as the actions in $\mathcal{A}'_i(s)$ are composed of unit-actions returned by the scripts in $\mathcal{P}$.

IV. Asymmetric Action Abstractions

We introduce an abstraction scheme we call **asymmetric action abstractions** (or asymmetric abstractions for short) that is not as restrictive as uniform abstractions but still uses the guidance of the scripts for selecting a subset of promising actions. The key idea behind asymmetric abstractions is to reduce the number of legal actions of only a subset of the units controlled by player i; the sets of legal actions of the other units remain unchanged. We call the subset of units that do not have their set of legal actions reduced the **unrestricted units**; the complement of the unrestricted units are defined as the **restricted units**.

Definition 2: An asymmetric abstraction Ω is a function receiving as input a state s, a player i, a set of unrestricted units $\mathcal{U}'_i \subseteq \mathcal{U}^s_i$, and a set of scripts $\mathcal{P}$. Ω returns a subset of actions of $\mathcal{A}_i(s)$, denoted $\mathcal{A}''_i(s)$, defined by the Cartesian product of the unit-actions in $\mathcal{M}(s, u, \mathcal{P})$ for all u in $\mathcal{U}^s_i \setminus \mathcal{U}'_i$ and of unit-actions $\mathcal{M}(s, u')$ for all u' in $\mathcal{U}'_i$.

The following theorem shows that an optimal strategy derived from the space induced by an asymmetric abstraction is at least as good as the optimal strategy derived from the space induced by a uniform abstraction as long as both abstractions are defined by the same set of scripts.

Theorem 1: Let Φ be a uniform abstraction and Ω be an asymmetric abstraction, both defined with the same set of scripts $\mathcal{P}$. For a finite match with start state s, let $V^\Phi_i(s)$ be the optimal value of the game computed by considering the space induced by Φ; define $V^\Omega_i(s)$ analogously. We have that $V^\Omega_i(s) \geq V^\Phi_i(s)$.

The proof for Theorem 1 (provided in the Appendix A of the dissertation [6]) hinges on the fact that a player searching with Ω has access to more actions than a player searching with Φ.

A practical advantage of asymmetric abstractions is that they allow search algorithms to differently divide its "attention" among the units at a given state of the game. That is, depending on the game state, some units might be more important than others (e.g., units with low hit points), and asymmetric abstractions allow one to derive finer strategies to

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

these units by accounting for a larger set of unit-actions for them.

V. Searching in Asymmetrically-Abstracted Action Spaces

In this section we briefly describe the four algorithms we introduced for searching in asymmetrically-abstracted spaces (c.f. Chapter 5 of the dissertation [6]).

A. GAB and SAB

Greedy Alpha-Beta Search (GAB) and Stratified Alpha-Beta Search (SAB), two algorithms for searching in asymmetrically-abstracted trees, hinge on a property of PGS [1] and SSS [2] that has hitherto been overlooked. Namely, both PGS and SSS may come to an early termination if they encounter a local maximum. GAB and SAB take advantage of PGS's and SSS's early termination by operating in two steps. In the first step GAB and SAB search for an action in the uniformly-abstracted tree with PGS and SSS, respectively. The first step finishes either when (i) the time limit is reached or (ii) a local maximum is encountered. In the second step, which is run only if the first step finishes by encountering a local maximum, GAB and SAB fix the moves of all restricted units according to the moves found in the first step, and search in the asymmetrically-abstracted tree for unit-actions for all unrestricted units.

B. A2N and A3N

We call A2N the version of NaïveMCTS [7] that uses an action abstraction defined by two sets of scripts: $\mathcal{P}'$ and $\mathcal{P}''$. A2N divides the set of units into two subsets: the units related to $\mathcal{P}'$ and the units related to $\mathcal{P}''$. A2N can only sample unit actions m for the units u in the first group if m is returned by one of the scripts in $\mathcal{P}'$ for u. The unit actions A2N can sample for the second group of units is defined analogously. Note that the two subsets of units do not need to be disjoint as some units can have actions sampled from both sets of scripts. The action abstraction used by A2N is also asymmetric as the number of scripts in each set can be different.

We call Asymmetrically Action-Abstracted NaïveMCTS (A3N) the version of NaïveMCTS that accounts during search for all unit actions of the unrestricted units and only for the actions returned by the set of scripts $\mathcal{P}$ for the restricted units. The only different between NaïveMCTS and A3N is that in the latter, the select-sampling procedure can only sample combinations of unit-actions that are in the asymmetrically-abstracted tree.

VI. Experiments

We introduced nine strategies for selecting the unrestricted units. A selection strategy receives a state s and a set size N and returns a subset of size N of player i's units for asymmetric action abstractions. The selection of unrestricted units is dynamic as the strategies can choose different unrestricted units at different states. We defined empirically the best strategy and number of unrestricted units that are asymmetrically controlled by our search algorithms, see chapter 6.3 of the dissertation [6] for details.

To evaluate our algorithms, we tested GAB, SAB, and A3N against their baselines and the state-of-the-art systems for playing MicroRTS [7]. The baselines for GAB and SAB consist of the algorithms used for the first step of the search, PGS and SSS, respectively. The baseline for A3N, we used A1N, that consist of a A3N's variation without un-restricted units. Table I shows the average percentage of matches won by each approach tested in ten maps. It is notable how the algorithms using asymmetric action abstractions outperformed their baselines. SAB won 68% of the matches played against SSS. GAB and A3N achieve 87% and 88.5% of victories against the baselines PGS and A1N, respectively. Considering A3N against NS, the percentage is expressive, 93.5%. Overall, A3N wins more matches than any other approach tested.

Professional RTS players often control one unit at a time, focusing most of their attention to that single unit. GAB, SAB and A3N act similarly as their action abstractions provide a coarse set of unit-actions to all units but the units engaged in some critical activity in the game; for these units all actions are accounted for during search.

VII. Publications

The results of this dissertation were published in prestigious conferences in Artificial Intelligence. These papers are listed below:

- [8] **AAAI conference (Qualis A1)**. Asymmetric action abstractions for multi-unitcontrol in adversarial real-time scenarios.
- [9] **AIIDE conference (Qualis A3)**. Action abstractions for combinatorial multi-armed bandit tree search.

During the master's degree, we investigated and worked in other papers describing planning algorithms for zero-sum extensive-form games. Bellow, they are listed:

- [10] **AAAI conference (Qualis A1)**. Evolving action abstractions for real-time planning in extensive-form games.
- [11] **AIIDE conference (Qualis A3)**. Nested-greedy search for adversarial real-time games.
- [12] **IEEE Transactions on Games (Qualis A4)**. Strategy generation for multi-unit real-time games via voting.

The algorithms developed during my Masters allowed me to develop a system that won the **IEEE MicroRTS Competition in 2018**.We are currently working on a paper to be submitted to the Journal of Artificial Intelligence Research **(JAIR, Qualis A1)**, which extends our previous publications by presenting more experiments on MicroRTS and by discussing important implementation issues of algorithms that search in asymmetrically-abstracted spaces.

VIII. Contributions

The work presented by the dissertation advances the state-of-the-art in planning algorithms for domains with very large action spaces by introducing asymmetric action abstractions.

TABLE I
THE WINNING RATE OF THE ROW PLAYER AGAINST THE COLUMN PLAYER IN 100 MATCHES PLAYED IN 10 MAPS. 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 BEST WINNING RATE OF EACH COLUMN IS HIGHLIGHTED.

	HER	RAR	AHT	NS	WOR	A1N	SSS	PGS	PS	SCV+	LIR	STT	SAB	GAB	A3N	Avg.
HER	-	85.0	13.5	57.5	15.0	17.5	30.0	23.0	16.0	18.0	12.5	5.0	9.0	8.0	3.0	22.4
RAR	15.0	-	57.0	78.0	60.0	24.0	16.0	11.0	3.0	20.0	0.0	16.0	11.0	12.0	1.0	23.1
AHT	86.5	43.0	-	18.5	10.0	13.0	27.0	24.5	39.5	28.5	39.5	9.0	19.5	19.5	2.0	27.1
NS	42.5	22.0	81.5	-	41.0	35.0	22.0	20.0	28.0	26.5	20.0	20.0	27.0	23.0	6.5	29.6
WOR	85.0	40.0	90.0	59.0	-	29.0	49.0	43.0	40.0	50.0	35.0	38.5	27.5	31.5	26.5	46.0
A1N	82.5	76.0	87.0	65.0	71.0	-	36.5	34.0	49.0	46.0	34.0	28.0	24.0	26.0	11.5	47.9
SSS	70.0	84.0	73.0	78.0	51.0	63.5	-	54.0	42.0	37.0	28.0	27.0	32.0	17.0	16.0	48.0
PGS	77.0	89.0	75.5	80.0	57.0	66.0	46.0	-	45.5	42.5	46.0	27.5	32.0	13.0	18.5	51.1
PS	84.0	97.0	60.5	72.0	60.0	51.0	58.0	54.5	-	42.5	47.0	30.0	29.0	26.0	28.5	52.9
SCV+	82.0	80.0	71.5	73.5	50.0	54.0	63.0	57.5	57.5	-	61.0	40.5	44.0	34.5	23.5	56.6
LIR	87.5	100.0	60.5	80.0	65.0	66.0	72.0	54.0	53.0	39.0	-	53.0	35.5	17.0	41.0	58.8
STT	95.0	84.0	91.0	80.0	61.5	72.0	73.0	72.5	70.0	59.5	47.0	-	53.0	29.0	52.5	67.1
SAB	91.0	89.0	80.5	73.0	72.5	76.0	68.0	68.0	71.0	56.0	64.5	47.0	-	36.5	23.0	65.4
GAB	92.0	88.0	80.5	77.0	68.5	74.0	83.0	87.0	74.0	65.5	83.0	71.0	63.5	-	47.5	75.3
A3N	97.0	99.0	98.0	93.5	73.5	88.5	84.0	81.5	71.5	76.5	59.0	47.5	77.0	52.5	-	78.5

We also introduced asymmetric action abstractions for multi-unit zero-sum extensive-form games. We also introduced A2N, A3N, GAB, and SAB, four search algorithms for searching in asymmetrically-abstracted spaces. Similarly to uniformly-abstracted spaces, asymmetric abstractions also use domain-knowledge in the form of scripts. However, in contrast with uniform abstractions, which restrict all units to the unit-actions returned by the scripts, asymmetric action abstractions restrict only a subset of the units—the restricted units. Algorithms searching with asymmetric action abstractions account for all legal unit-actions of the remaining units—the unrestricted units. As a result, the strategy derived by search algorithms are focused on the unrestricted units, as the algorithms are able to derive finer plans for such units. Asymmetric action abstractions can be seen as an attention scheme, where the search "pays more attention" to a subset of units.

We evaluated our algorithms with an extensive set of experiments on μRTS. Our results suggest that A3N is the current state-of-the-art algorithm in this domain if one considers a large diversity of maps and opponents, similarly to the setting used in the μRTS annual competition [4]. If one considers large maps such as those used in commercial games, then GAB presented the strongest results.

Although we performed our experiments on μRTS, the ideas of this paper are general and could be applied to other games. For example, in collectible card games such as Hearthstone [13] and Magic: The Gathering [14] the player has to decide on the action of several cards. Algorithms could use asymmetric action abstractions to focus their search on a subset of the cards. The ideas introduced in this paper might also be applied in problems other than games. For example, a robotic system that controls several actuators while trying to accomplish a task can benefit from asymmetric action abstractions. This is because some actuators might require a finer control than the others.

REFERENCES

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

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

[3] S. Ontañón, "Combinatorial multi-armed bandits for real-time strategy games," *Journal of Artificial Intelligence Research*, vol. 58, pp. 665–702, 2017.

[4] S. Ontañón, N. A. Barriga, C. R. Silva, R. O. Moraes, and L. H. Lelis, "The first microrts artificial intelligence competition." *AI Magazine*, vol. 39, no. 1, 2018.

[5] D. Churchill and M. Buro, "Hierarchical portfolio search: Prismata's robust AI architecture for games with large search spaces," in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2015, pp. 16–22.

[6] R. O. Moraes, "Asymmetric action abstractions for real-time strategy games," 2019.

[7] S. Ontañón, "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*, 2013, pp. 58–64.

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

[9] R. O. Moraes, J. R. H. Mariño, L. H. S. Lelis, and M. A. Nascimento, "Action abstractions for combinatorial multi-armed bandit tree search," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. AAAI, 2018, pp. 74–80.

[10] J. R. Marino, R. O. Moraes, C. Toledo, and L. H. Lelis, "Evolving action abstractions for real-time planning in extensive-form games," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 2330–2337.

[11] R. O. Moraes, J. R. H. Mariño, and L. H. S. Lelis, "Nested-greedy search for adversarial real-time games," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2018, pp. 67–73.

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

[13] A. Dockhorn and S. Mostaghim, "Introducing the hearthstone-ai competition," 2019.

[14] C. D. Ward and P. I. Cowling, "Monte carlo search applied to card selection in magic: The gathering," in *Proceedings of the International Conference on Computational Intelligence and Games*. IEEE Press, 2009, pp. 9–16.