Proceedings, The Thirteenth AAAI Conference on
Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17)

Algorithm Selection in Zero-Sum Computer Games

Anderson Rocha Tavares
Advisor: Luiz Chaimowicz
Laboratory of Multidisciplinary Research in Games
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
anderson@dcc.ufmg.br

Abstract

Competitive computer games are challenging domains for artificial intelligence techniques. In such games, human players often resort to strategies, or game-playing policies, to guide their low-level actions. In this research, we propose a computational version of this behavior, by modeling game playing as an algorithm selection problem: agents must map game states to algorithms to maximize their performance.

By reasoning over algorithms instead of low-level actions, we reduce the complexity of decision making in computer games. With further simplifications on the state space of a game, we were able to discuss game-theoretic concepts over aspects of real-time strategy games, as well as generating a game-playing agent that successfully learns how to select algorithms in AI tournaments.

We plan to further extend the approach to handle incomplete-information settings, where we do not know the possible behaviors of the opponent.

Introduction

To deal with the complexity of competitive computer games, humans usually resort to strategies to guide their actions. Strategies can specify a multitude of game-playing behaviors: aggressive attacks, conservative defenses, and everything in-between. In practical terms, strategies are policies that determine the course of low-level game actions to execute. A player attempts to map the current game state to the best possible strategy in its repertoire, taking into account that his choice will affect the future state (sequential reasoning) and that other players are likewise attempting to maximize their performance (multiagent reasoning).

In this research, we formulate this situation as a computational problem: if we consider software-controlled players instead of humans and a portfolio of algorithms instead of a repertoire of strategies, we have a problem of algorithm selection (Rice 1976) in a multiagent, sequential setting imposed by the computer game. For our purposes, an algorithm is any procedure that maps a game state to a low-level action: it can be as simple as a lookup table, or as sophisticated as a full game-playing agent with heuristics, planning and/or learning mechanisms.

By reasoning over algorithms instead of low-level game actions, the complexity of the decision problem for playing agents is reduced at the potential expense of optimality. This is a reasonable trade-off in computer games, which are mostly real-time domains: players do not have time to reason over huge state-action spaces to find optimal actions.

In this research we focus on two-player zero-sum computer games. We adopt Markov games[1] as a formal model of a computer game. In this model, players act simultaneously in each state. A transition function, which might be unknown to the players, determine the next state, based on their joint actions. Reward functions determine the rewards for players at each transition. In our Markov game model, players select algorithms instead of low-level game actions. This reduces the game's branching factor on the players perspective.

Algorithm selection has been studied in complex problems, such as SAT (Xu et al. 2008) and General Video Game Playing (Bontrager et al. 2016). Our framework extends such previous efforts by accounting for multiple agents and by allowing shifting between algorithms regularly, instead of a one-shot selection to solve the problem at hand. However, our current results, detailed in the next section, were obtained by testing the case with one-shot selection as well. Studies on multi-stage algorithm selection are in progress.

Current results

In a first experiment, we instantiated a special case of our approach upon real-time strategy game StarCraft. The special case has a single state, i.e., each player maps only the initial game state to an algorithm. This reduces the Markov game to a normal-form game, whose payoff matrix indicates expected relative performance among algorithms. In this setting, players are algorithm selectors and their policies are probability distributions over algorithms. Nash Equilibrium over the payoff matrix specifies a safe policy for algorithm selection in this setting.

Our experiments confirmed a trend in game-theoretical research (McCracken and Bowling 2004; Ganzfried and Sandholm 2015): principled approaches to exploit sub-optimal opponents can be useful, although they are not as safe as the equilibrium policy. Such exploitation policies are able to

[1] Also known as Stochastic Games (Shapley 1953).

achieve higher payoffs than that of the equilibrium policy and they have bounded losses in case their exploitation attempts fail. These results are in (Tavares et al. 2016), where the algorithms were called strategies.

We also built a functional StarCraft bot, named MegaBot[2], based on the framework of Tavares et al. (2016), to participate in AI tournaments. MegaBot has a portfolio of three algorithms, which are non-dominant bots of AIIDE 2015 tournament[3]. We selected non-dominant bots for the portfolio to test whether MegaBot performs well by learning how to properly select algorithms rather than due to a powerful portfolio.

In StarCraft AI tournaments, we do not know the normal-form game's payoff matrix beforehand, thus we learn its values via minimax-Q's update rule (Littman 1994). Besides, in StarCraft, we can identify the adversary's bot name. In the terms of our model, MegaBot knows against which algorithm it is playing. This reduces the normal-form game matrix to a single column, corresponding to the payoffs against that specific algorithm. This can be seen as a multi-armed bandit, so that MegaBot employs an ϵ-greedy mechanism for algorithm selection.

MegaBot placed 7th out of 16 competitors in CIG[4] and 9th out of 22 competitors in AIIDE[5] 2016 StarCraft AI competitions. It has outperformed each of its portfolio components and received an honorable mention for its learning curve (measured in rate of victories per round). This indicates that algorithm selection can be a useful approach for real-time strategy games. MegaBot did not score better because no component of its portfolio could defeat the strongest competitors.

Currently, we are investigating the multi-state setting for algorithm selection, in which players start with an algorithm and can switch it during the match, being able to select the best algorithm for the current context. This corresponds to the general Markov game formalism. To simplify the state space, as a first approach we cluster similar states to form the Markov game's decision points, allowing us to employ tabular reinforcement learning methods.

In the multi-state setting, preliminary experiments in microRTS (Ontañón 2013) suggest that Q-learning (Watkins and Dayan 1992) and MinimaxQ (Littman 1994), which are traditional tabular reinforcement learning methods, are competitive against PuppetSearch (Barriga, Stanescu, and Buro 2015) and Adversarial Hierarchical Task Network planning (Ontañón and Buro 2015), which are state-of-the-art search methods.

Next steps

The proposed model assumes that players know each other's algorithm portfolio. Formally, this is a complete-information Markov game, which is addressed by methods such as MinimaxQ (Littman 1994). However, in a realistic setting, the agent is aware of the opponent's presence but does not know his possible behaviors. Formally, this is an incomplete-information Markov game. A special case of the incomplete-information Markov game is the single-state setting, which corresponds to an adversarial multi-armed bandit (Auer et al. 1995). The Exp3 method of Auer et al. (1995) exhibits theoretical performance guarantees for this problem, dismissing complete information.

On the one hand, MinimaxQ handles the multi-stage setting by accounting for future states when updating the action-values, but requires complete information to calculate a policy to map the current state to a probability distribution over actions. On the other hand, Exp3 does not account for future states, but dismisses complete information to calculate a policy. Our proposed approach combines the strengths of the two methods: we account for future states when updating the current action-values, by using MinimaxQ's update rule, and dismiss complete information, by using Exp3 method to calculate the policy. To the best of our knowledge, this method to handle incomplete-information Markov games is novel.

We want to investigate whether our method bounds agent's losses, extending the guarantees of Exp3 from single to multi-state settings and/or MinimaxQ's guarantees from complete- to incomplete-information settings. Experimental results may be useful in this sense: a robust performance of our method may indicate that further investigation of its theoretical properties can be fruitful. In such experiments, we could also test other bandit methods (e.g. ϵ-greedy or UCB (Auer, Cesa-Bianchi, and Fischer 2002)) instead of Exp3, to verify their performance in practice.

An interesting direction for future research comes from the fact that so far we worked with out-of-the-box algorithms, i.e. game-playing bots. It would be interesting to merge our approach with the discovery of options - temporally extended actions - in Markov decision processes (Machado, Bellemare, and Bowling 2017), as our concept of algorithms seems connected to the concept of options. This way, we would be automatically learning how to select algorithms as well as creating novel, and potentially better, game-playing algorithms to select.

Future studies could also address the issue of parameterized algorithms. As each parameter setting can specify a different behavior, the number of possible behaviors to select can become intractable. It might be possible to determine bounds in the expected performance of an algorithm within certain parameter ranges, so that the problem become tractable again.

Acknowledgments

The author acknowledges CNPq and CAPES for the PhD scholarship and FAPEMIG for support in this research, as well as the reviewers for useful suggestions and remarks.

References

Auer, P.; Cesa-Bianchi, N.; Freund, Y.; and Schapire, R. E. 1995. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Foundations of Computer Science. Proceedings, 36th Annual Symposium on*, 322–331. IEEE.

[2]https://github.com/andertavares/MegaBot

[3]https://www.cs.mun.ca/~dchurchill/starcraftaicomp/2015/

[4]https://sites.google.com/site/starcraftaic/result

[5]http://www.cs.mun.ca/~dchurchill/starcraftaicomp/2016/

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47(2-3):235–256.

Barriga, N. A.; Stanescu, M.; and Buro, M. 2015. Puppet Search: Enhancing Scripted Behavior by Look-Ahead Search with Applications to Real-Time Strategy Games. In *11th Artificial Intelligence in Interactive Digital Entertainment Conference (AIIDE)*.

Bontrager, P.; Khalifa, A.; Mendes, A.; and Togelius, J. 2016. Matching Games and Algorithms for General Video Game Playing. In *12th Artificial Intelligence in Interactive Digital Entertainment Conference (AIIDE)*, 122–128.

Ganzfried, S., and Sandholm, T. 2015. Safe Opponent Exploitation. *ACM Transactions on Economics and Compuation* 3(2).

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, 157–163. New Brunswick, NJ: Morgan Kaufmann.

Machado, M. C.; Bellemare, M. G.; and Bowling, M. 2017. A Laplacian Framework for Option Discovery in Reinforcement Learning. *CoRR* abs/1703.00956.

McCracken, P., and Bowling, M. 2004. Safe strategies for agent modelling in games. *AAAI Fall Symposium on Artificial Multi-agent Learning* 103–110.

Ontañón, S., and Buro, M. 2015. Adversarial hierarchical-task network planning for complex real-time games. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Ontañón, S. 2013. The combinatorial multi-armed bandit problem and its application to real-time strategy games. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Rice, J. R. 1976. The algorithm selection problem. *Advances in computers* 15:65–118.

Shapley, L. S. 1953. Stochastic games. *Proceedings of the National Academy of Sciences* 39(10):1095–1100.

Tavares, A.; Azpúrua, H.; Santos, A.; and Chaimowicz, L. 2016. Rock, Paper, StarCraft: Strategy Selection in Real-Time Strategy Games. In *12th Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, 93–99.

Watkins, C. J. C. H., and Dayan, P. 1992. Q-learning. *Machine Learning* 8(3):279–292.

Xu, L.; Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2008. SATzilla: portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research* 32:565–606.