Adversary Strategy Sampling for Effective Plan Generation

Lukáš Chrpa, Pavel Rytíř, Rostislav Horčík, Jan Čuhel, Anastasiia Livochka, Stefan Edelkamp
Faculty of Electrical Engineering, Czech Technical University in Prague
{chrpaluk,rytirpav,xhorcik,cuheljan,livocana,edelkste}@fel.cvut.cz

Abstract

Effective plan generation in adversarial environments has to take into account possible actions of adversary agents, i.e., the agent should know what the competitor will likely do.

In this paper we propose a novel approach for estimating strategies of the adversary, sampling actions that interfere with the agent's ones. The estimated competitor strategies are used in plan generation by considering that agent's actions have to be applied prior to the ones of the competitor, whose estimated times dictate the agent's deadlines. Missing these deadlines entails additional plan cost.

Introduction

The concept of plan generation in adversarial environment is not new (Applegate, Elsaesser, and Sanborn 1990). Succinct symbolic representations of state sets helped generating optimistic and strong cyclic adversarial plans (Jensen, Veloso, and Bowling 2001; Cimatti et al. 2003), a setting conceptually related to FOND planning (Kissmann and Edelkamp 2009). Such a setting, however, has to explore most if not all alternatives (in analogy to traditional game-tree methods such as minimax). Monte-Carlo Tree Search (MCTS) and Online Evolutionary Planning have been applied in adversarial environments such as the Hero Academy game (Justesen et al. 2018), or Starcraft (Justesen and Risi 2017). Deep Reinforcement Learning (DRL) has shown impressive results in Starcraft (Vinyals et al. 2019) and other (adversarial) domains such as the games of Chess or Go (Silver et al. 2018). The game-theoretical *Double-Oracle* algorithm (McMahan, Gordon, and Blum 2003) incrementally generates mixed strategies for each agent which are in a Nash Equilibrium if in each iteration each agent generates *best response* to the other agent's current strategy. Recent works (Rytíř, Chrpa, and Bošanský 2019; Chrpa, Rytíř, and Horčík 2020) incorporate automated planning into Double Oracle in a sub-class of *zero-sum games* in which two agents compete for the same set of (soft) goals, where each (soft) goal can be achieved by at most one of the agents.

In this paper, we present a heuristic method for estimating (mixed) strategies of an adversary in the aforementioned subclass of zero-sum games. Leveraging a heuristic for estimating earliest action application time, developed by Chrpa, Rytíř, and Horčík (2020), we propose a "sampling" method which provides potential application times of actions of the adversary that interfere with agent's actions. The sampling method, hence, provides *deadlines* for those agent's actions (later called *critical actions*) the agent has to take into consideration while it generates its plan.

Background

We assume a restricted form of *Temporal Planning* in a static, deterministic and fully observable environment. The environment is represented via state variables, where each variable has its own domain. We consider durative actions, where preconditions can be specified "at start", "at end" and "over all" and effects can be specified "at start" and "at end" such as in PDDL 2.1 (Fox and Long 2003).

Two-player *zero-sum games* consider sets of *pure strategies* (plans in our case) for each of the players and the sum of utility values of each player's strategy profile is zero. By a *mixed strategy* we mean a probability distribution over a set of pure strategies.

Planning in Zero-sum Games

With the presence of an adversary, the quality of plans depends on opponent actions that might interfere with agent's actions. Adversary's actions might invalidate preconditions of the agent's actions rendering them no longer applicable. Consequently, the agent might no longer achieve some (soft) goals which will be achieved by the adversary agent.

Inspired by the MA-STRIPS formalism (Brafman and Domshlak 2008), used in Multi-agent planning, we consider two (competing) agents, shared (soft) goals and durative actions. Let $\mathcal{ZP} = (V, A^1, A^2, I, G)$ be a ***zero-sum game planning task***, where V is a set of variables, A^1 and A^2 are disjoint sets of (durative) actions for the respective agent, I is an initial state and G is a set of soft goals.

We assume plans for both agents are executed simultaneously. We consider only problems where one action can invalidate a precondition of an action of the other agent which has not (yet) started. If conflicting actions are scheduled at the same time, then one action is randomly selected to be applied (with an equal chance) while the other become in-

applicable. Inapplicable actions are skipped during plan execution. After both plans are executed, the cost of the plan for each agent is determined as the sum of costs of soft goals the agent failed to achieve.

We adopt the following notions from Chrpa, Rytíř, and Horčík (2020). A ***critical fact*** (for the first agent) is a fact that is required by ***critical actions*** (from A^1) while it can be deleted by ***adversary actions*** (from A^2).

To achieve a given soft goal the agent might have to apply ***relevant*** critical actions which are part of a *disjunctive action landmark* for that goal (Hoffmann, Porteous, and Sebastia 2004). Adversary actions, however, set ***deadlines*** for corresponding critical actions as they delete critical facts required by these critical actions.

For a given mixed strategy S of the adversary we define ***response planning task*** $\mathcal{P}_S = (V, A^1, I, G)$ with a cost function assigning cost to critical actions (from A^1) as a sum of $p_a(S,t) * M_i$, where $p_a(S,t)$ is the probability of missing the deadline of the critical action a in time t and M_i is the cost for failing a soft goal G_i (a is relevant to G_i).

If for each soft goal at most one critical action has to be applied and the critical fact cannot be (re)achieved by any agent, the cost optimal plan for the response planning task accounts for the *best response* strategy.

Estimating Earliest Action/Fact Time

We adopt the algorithm for estimating lower bounds of action application time and fact occurrence time proposed by Chrpa, Rytíř, and Horčík (2020) which is inspired by the h^{max} heuristics from classical planning (Bonet and Geffner 2001). Each iteration involves selection of an action with the smallest application time that has not yet been processed. Determining action application time is done as maximum across the times of its "at start" and "over all" preconditions (while relaxing "at end" preconditions). Determining fact occurrence time is done as minimum of the current fact time and the time when the fact in the selected action effects takes place.

Note that occurrence times for some non-initial facts might be initially known as well as application times for some actions. If application times of some actions are known, then only the latest value of variables appearing in these actions is considered as an occurred fact.

Estimating Adversary Strategy

We propose a heuristic method that estimates times of adversary actions because such information is important for setting the deadlines for agent's critical actions and thus formulating the (best) response planning task. That said, we do not need to compute whole plans of the adversary.

Initially, the adversary actions are arranged into "clusters" such that each cluster corresponds to a critical fact (note that one adversary action can be in more clusters). The idea of estimating adversary action application times for a possible (single) plan of the adversary is based on iterative selection of adversary actions until all clusters are covered. Initially, the set of selected adversary actions is empty and the initial facts occurrence time is set to 0. Then, in the intermediate step, we estimate an action application/fact occurrence time while considering the selected adversary actions (see the previous subsection) and then we select an adversary action from a cluster that has not yet been considered.

We consider two variants of adversary action selection. The first variant considers all reachable adversary actions from not yet processed clusters. The rationale behind this variant is in assuming the adversary will tend to apply its adversary actions as early as possible. The second variant takes into account interference between selected and "to be selected" adversary actions such that we select an adversary action with the least interference with the already selected adversary actions. The rationale behind this variant is in assuming that the adversary will try to maximise parallel execution of its actions to minimise their application time. Probability of selecting a particular adversary action depends on their estimated application time such that those with lower application time are preferred analogously to the roulette wheel selection in genetic programming.

As adversary actions are selected randomly (according to the probability), different runs of the method produce different outcomes. The k outcomes of the method, or "samples", are then "arranged" into adversary strategy S.

Evaluation

For experiments, we used the Resource Hunting domain (Rytíř, Chrpa, and Bošanský 2019) and the Taxi domain, which represents a scenario of two taxi companies competing for passengers. We compared the sampling method to the naive one (generating make-span optimal plans without considering the adversary) and Nash Equilibria generated with the use of the Double Oracle method (Rytíř, Chrpa, and Bošanský 2019). As an optimal classical planner, we used the Fast Downward planner (Helmert 2006) with the potential heuristic (Pommerening et al. 2015) optimised by the diversification method proposed by Seipp, Pommerening, and Helmert (2015). As an optimal temporal planner we used CPT4 (Vidal 2011). We ran the experiments on Linux with 2.10GHz Intel Xeon CPU E5-2620 v4 with 32GB RAM.

The both variants of the action selection in the sampling method achieved very similar results. The utility value of the sampling method is 0.97, 0.997, 0.998 of the equilibrium value for the number of samples $k = 4, 256, 16384$, respectively. The utility value of the sampling method against the Naive method is 1.12, 1.15, 1.17 of the equilibrium value for $k = 4, 256, 16384$, respectively. The exploitability, normalised by the total game value, is 0.19 for the Naive approach. For a uniform strategy of 10 plans generated by the sampling method, the normalised exploitability is 0.086, 0.094, 0.124 for $k = 4, 256, 16384$, respectively. Since the sampling method is randomised it is harder to exploit.

Acknowledgements

This research was funded by AFOSR award FA9550-18-1-0097, by the OP VVV project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics" and by the Czech Science Foundation (18-07252S).

References

Applegate, C.; Elsaesser, C.; and Sanborn, J. C. 1990. An architecture for adversarial planning. *IEEE Trans. Systems, Man, and Cybernetics* 20(1): 186–194.

Bonet, B.; and Geffner, H. 2001. Planning as heuristic search. *Artif. Intell.* 129(1-2): 5–33.

Brafman, R. I.; and Domshlak, C. 2008. From One to Many: Planning for Loosely Coupled Multi-Agent Systems. In *The Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008*, 28–35.

Chrpa, L.; Rytíř, P.; and Horčík, R. 2020. Planning Against Adversary in Zero-Sum Games: Heuristics for Selecting and Ordering Critical Actions. In *The Thirteenth International Symposium on Combinatorial Search, SOCS 2020*, 20–28.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artif. Intell.* 147(1-2): 35–84.

Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *J. Artif. Intell. Res.* 20: 61–124.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research* 26: 191–246.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered Landmarks in Planning. *J. Artif. Intell. Res.* 22: 215–278.

Jensen, R.; Veloso, M.; and Bowling, M. 2001. OBDD-based optimistic and strong cyclic adversarial planning. In *ECP*, 265–276.

Justesen, N.; Mahlmann, T.; Risi, S.; and Togelius, J. 2018. Playing Multiaction Adversarial Games: Online Evolutionary Planning Versus Tree Search. *IEEE Trans. Games* 10(3): 281–291.

Justesen, N.; and Risi, S. 2017. Continual online evolutionary planning for in-game build order adaptation in StarCraft. In *The Genetic and Evolutionary Computation Conference, GECCO 2017*, 187–194.

Kissmann, P.; and Edelkamp, S. 2009. Solving Fully-Observable Non-deterministic Planning Problems via Translation into a General Game. In *KI*, volume 5803, 1–8. Springer.

McMahan, H. B.; Gordon, G. J.; and Blum, A. 2003. Planning in the Presence of Cost Functions Controlled by an Adversary. In *ICML*, 536–543.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015. From Non-Negative to General Operator Cost Partitioning. In *Proc. AAAI'15*, 3335–3341.

Rytíř, P.; Chrpa, L.; and Bošanský, B. 2019. Using Classical Planning in Adversarial Problems. In *Proceedings of the 31st IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 1327–1332.

Seipp, J.; Pommerening, F.; and Helmert, M. 2015. New Optimization Functions for Potential Heuristics. In *Proc. ICAPS'15*, 193–201.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play 362(6419): 1140–1144.

Vidal, V. 2011. CPT4: An Optimal Temporal Planner Lost in a Planning Competition without Optimal Temporal Track. In *Proceedings of the 7th International Planning Competition (IPC-2011)*, 25–28. Freiburg, Germany.

Vinyals, O.; Babuschkin, I.; Czarnecki, W.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.; Powell, R.; Ewalds, T.; Georgiev, P.; Oh, J.; Horgan, D.; Kroiss, M.; Danihelka, I.; Huang, A.; Sifre, L.; Cai, T.; Agapiou, J.; Jaderberg, M.; and Silver, D. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575. doi: 10.1038/s41586-019-1724-z.