Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Learning to Avoid Dominated Action Sequences in Planning for Black-Box Domains (Extended Abstract)

Yuu Jinnai, Alex Fukunaga
Graduate School of Arts and Sciences
The University of Tokyo

Abstract

Black-box domains where the successor states generated by applying an action are generated by a completely opaque simulator pose a challenge for domain-independent planning. The main computational bottleneck in search-based planning for such domains is the number of calls to the black-box simulation. We propose a method for significantly reducing the number of calls to the simulator by the search algorithm by detecting and pruning sequences of actions which are dominated by others. We apply our pruning method to Iterated Width and breadth-first search in domain-independent black-box planning for Atari 2600 games, adding our pruning method significantly improves upon the baseline algorithms.

1 Introduction

Recently, planning in *black-box* domains with much more *opaque* domain models has attracted attention, spurred by interest in developing a game-independent AI playing algorithm for video games (Bellemare et al. 2013). In black-box planning, a state vector and a set of actions are available, as well as an objective function for evaluating states. However, the only way to compute the successor state s' resulting from applying an action a to state s is to execute $s' = Simulate(s, a)$, a black-box simulation function for which the internal dynamics are inaccessible.

Such black-box domains present a challenge for search-based planning, because the pruning techniques which enabled effective search in domains with transparent domain models are not applicable, leaving us only with brute-force methods such as breadth-first search. However, it was recently shown that Iterative Width (IW) (Lipovetzky and Geffner 2012), a search strategy which prunes the search space by focusing only on states which are "novel" compared to previously expanded nodes, can be used as the basis for a successful, search-based planner for black-box planning (Lipovetzky, Ramirez, and Geffner 2015).

In this paper, we investigate duplicate avoidance in *on-line planning* settings for black-box domains, as exemplified by the on-line planning for the ALE (Bellemare et al. 2013), where an agent plays video games by repeating the loop: (1) solving a planning problem with a very limited resource budget, and (2) execute an action. Because this setting poses a series of related planning episodes, there is an opportunity to improve planner performance over time by learning a duplicate avoidance strategy.

Specifically, we seek to eliminate actions (and sequences of actions) which are dominated by others (and lead to duplicate states). For example, in the ALE, 18 actions are always available (the joystick has 9 states – up/down/left/right/4 diagonals/"neutral", and the "fire" button has 2 states, 9x2=18). Previous work in search-based planning for the ALE treats all 18 actions as applicable at every state (Lipovetzky, Ramirez, and Geffner 2015; Shleyfman, Tuisov, and Domshlak 2016). However, in any particular game, many of these 18 actions are dominated ("useless"): First, some actions are trivially dominated because they are completely ignored, or the program always treats them as being equivalent to other actions (e.g., in some games, the state of the "fire" button is irrelevant). Second, some actions are *conditionally* dominated because, in a given context, the action results in the same state as another action (e.g., in a maze-based game, if the agent is stuck against a wall to the left, then the "left" action is useless because (in some games) it results in the same state as "no action"). More generally, *sequences of actions* can be useless. For example, some actions can have *cooldown periods*, i.e, after action a is used, executing a again has no effect for the next t seconds (e.g. firing missiles in shooting games).

In this paper, we propose two methods to detect dominated action sequences and reduce the amount of applying such sequences. Unlike the standard method proposed by Taylor and Korf (1993), our method is applicable to black-box planning.

2 Dominated Action Sequence Pruning (DASP)

To improve the performance of planning episodes (Step 3) of the online black-box planning, we propose Dominated Action Sequence Pruning (DASP), a method to eliminate dominated action sequences in black-box planning.

DASP finds a set of action sequences A^L using Algorithm 1, based on the search tree explored in previous planning episodes.

Algorithm 1. [Find minimal action sequence set]

1. Initialize A^L_{min} to the set of all action sequences which

generate one or more non-duplicate nodes.

2. Let $G = (V, E)$ be a hypergraph where $v_i \in V$ represents an action sequence $\mathbf{a}_i$ with no non-duplicate search nodes, and hyperedge $e(v_0, v_1, .., v_n) \in E$ if there exist one or more duplicate search nodes generated by all of $\mathbf{a}_0, \mathbf{a}_1, ..., \mathbf{a}_n$ but not by any other actions sequences.
3. Add the minimal vertex cover of G to A^L_{min}.

DASP only uses A^L for the rest of the planning episodes, except for the first node expansion in each episode. That is, when expanding a state s, which is reached by a trajectory $(a_0, a_1, ..., a_n)$, an action a is applied only if all of $(a), (a_n, a), (a_{n-1}, a_n, a), ...$ are in A^L. For the first node expansion in each episode, we apply all the available actions including dominated action sequences. If a pruned action sequence generates a non-duplicate node, then the action sequence will be put in A^L for the next planning episode by Algorithm 1. See (Jinnai and Fukunaga 2017) for details.

3 Dominated Action Sequence Avoidance (DASA)

DASP classifies action sequences either effective all the time or not at any time. However, in many domains, most of the actions are conditionally effective, an action has a unique outcome for *some* states, but not for all states in the domain. We propose Dominate Action Sequence Avoidance (DASA), which learns conditional action sequence sets which are dependent on the current context of the plan.

Let $p(\mathbf{a}, t)$ be the fraction of new nodes generated by action sequence $\mathbf{a}$ in the t-th planning episode. We define $p^*(\mathbf{a}, t+1)$ as:

$$p^*(\mathbf{a}, t+1) = (p(\mathbf{a}, t) + \alpha p^*(\mathbf{a}, t))/(1+\alpha), \quad (1)$$

where α is a discount factor. $p^*(\mathbf{a}, t+1)$ is an estimate for the ratio of new nodes by action sequence $\mathbf{a}$ on $t+1$-th planning episode based on the experience of previous t planning episodes. If the number of nodes generated (including new/old nodes) by action sequence $\mathbf{a}$ in the t-th planning episode is 0, then $p^*(\mathbf{a}, t+1) = p^*(\mathbf{a}, t)$. $p^*(\mathbf{a}, 0)$ = 1.

DASA applies actions with higher p value more frequently, and action with lower p value less frequently. The trajectory to reach state s is given as $T = (a_0, a_1, ..., a_n)$. For each node expansion, action a is applied and $s' = succ(a, t)$ is generated with a probability:

$$P(a,t) = (1-\epsilon)s(p^*(a,t))s(p^*((a_{n-1},a),t)) \\ ...s(p^*((a_{n-l},...,a_n,a),t)) + \epsilon, \quad (2)$$

where $s(x)$ is a sigmoid function, ϵ is a parameter for the minimal probability for applying an action, and l is the length of the longest dominated action sequences to detect.

As the definition of new/old depends on the ordering of actions, the action sequences should properly be ordered. DASA finds an action sequence set A^L using Algorithm 1. DASA orders the action sequences preferring sequences in A^L and breaking ties in favor of higher p value.

The additional overhead on the runtime due to DASP/DASA should be negligible when node expansion is slow enough, which is likely in many blackbox domains.

Table 1: Comparison of the number of games scored the best. #Best includes ties, but excludes when all 5 algorithms have the same score. p-IW(1) (2000) is limited to 2000 simulation frames, while all others are limited to 10000 frames. DASA2: 2-step DASA ($L = 2$), DASA1: 1-step DASA ($L = 1$), DASP1: 1-step DASP ($L = 1$), default: use all available action set, restricted: use (hand-coded) restricted action set.

search method	DASA2	DASA1	DASP1	default	restrict
p-IW(1)	**22**	10	4	6	10
p-IW(1) (2000)	**24**	14	6	5	7
IW(1)	**22**	9	7	7	8
BrFS	**18**	11	11	6	11

4 Experimental Evaluation

We evaluated proposed methods on p-IW(1) (Shleyfman, Tuisov, and Domshlak 2016), IW(1)(Lipovetzky and Geffner 2012), and Breadth-first search (BrFS) algorithms.

For DASP we use all available actions until 12 planning frames (5x12=60 in-game frames). As the minimal vertex cover is NP-hard (Karp 1972), We calculate the optimal vertex cover if there are ≤ 5 nodes, and otherwise use a greedy algorithm which adds a vertex with the highest number of uncovered edges one by one. For DASA, we used a sigmoid function $s(x) = \frac{1}{1+e^{-5(x-0.5)}}$, minimal chance of applying action sequence $\epsilon = 0.04$, discount factor for p value $\alpha = 0.95$. The other parameters follow (Lipovetzky, Ramirez, and Geffner 2015). To reduce the variance, each game was played 5 times, with the reported results averaged across these runs. Overall, DASA ($L = 2$) outperformed other algorithms (Table 1).

5 Conclusion

We proposed DASP, a method to learn a static minimal action set which is valid throughout the course of a game, and DASA, a method to learn conditional minimal action sets which are dependent on the current context of the game. We evaluated DASP and DASA on 53 games in the ALE arcade game environment, and showed that DASD significantly improves the performance of black-box planning in these domains compared to baseline algorithms without DASD.

References

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47:253–279.

Jinnai, Y., and Fukunaga, A. 2017. Learning to prune dominated action sequences in online black-box domain. In *Proc. AAAI*.

Karp, R. M. 1972. Reducibility among combinatorial problems. In *Complexity of computer computations*. Springer. 85–103.

Lipovetzky, N., and Geffner, H. 2012. Width and serialization of classical planning problems. In *Proc. ECAI*, 540–545.

Lipovetzky, N.; Ramirez, M.; and Geffner, H. 2015. Classical planning with simulators: Results on the Atari video games. In *Proc. IJCAI*, 1610–1616.

Shleyfman, A.; Tuisov, A.; and Domshlak, C. 2016. Blind search for Atari-like online planning revisited. In *Proc. IJCAI*, 3251–3257.

Taylor, L. A., and Korf, R. E. 1993. Pruning duplicate nodes in depth-first search. In *Proc. AAAI*, 756–761.