Solving Sokoban with Forward-Backward Reinforcement Learning

Yaron Shoham, Gal Elidan
Google Research, Israel
yaronsh,elidan@google.com

Abstract

Despite seminal advances in reinforcement learning in recent years, many domains where the rewards are sparse, e.g. given only at task completion, remain quite challenging. In such cases, it can be beneficial to tackle the task both from its beginning and end, and make the two ends meet. Existing approaches that do so, however, are not effective in the common scenario where the strategy needed near the end goal is very different from the one that is effective earlier on.

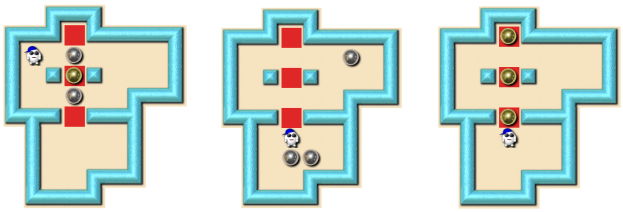
In this work we propose a novel RL approach for such settings. In short, we first train a backward-looking agent with a simple relaxed goal, and then augment the state representation of the forward-looking agent with straightforward *hint features*. This allows the learned forward agent to leverage information from backward plans, without mimicking their policy.

We demonstrate the efficacy of our approach on the challenging game of Sokoban, where we substantially surpass learned solvers that generalize across levels, and are competitive with SOTA performance of the best highly-crafted systems. Impressively, we achieve these results while learning from a small number of practice levels and using simple RL techniques.

Introduction

Many RL domains are challenging because the rewards are sparse, e.g., given only at task completion. Intuitively, the only thing we can leverage in such scenarios is the fact that the target goal is known and that we can somehow backtrack from this goal in order to assist the learning of the forward looking agent. Building on this intuition, two recent works use data augmentation or imitation, with the inherent assumption that the agent's behavior or policy near the goal is similar to the one at the beginning of the task (Edwards, Downs, and Davidson 2018; Goyal et al. 2019). In reverse curriculum learning (Florensa et al. 2017), we implicitly assume that the policy near the goal can be slowly transformed into the policy of the original problem. Such assumptions do not hold in general and the effective policy near the goal state can be quite different from the one that is beneficial earlier on.

We suggest a straightforward approach to cope with the above scenario. Instead of trying to imitate the policy of an agent that starts at the goal state and faces backwards, we use simple *hint features* of trajectories of that agent to augment the state representation when learning the forward looking agent. For example, a hint-feature can be the distance to the backward plan. If this feature is minimized to zero then a solution has been found. Thus, this feature can serve as a useful signal for the forward facing agent.



(a) Start (b) Intermediate (c) Goal

Figure 1: Sokoban level LOMA04-04. The strategy near the goal, pushing boxes towards targets and blocking pathways, is different from the strategy required at the beginning.

There are two important benefits to using these informed features instead of the backward policy itself. First, instead of imitating a specific (possibly sub-optimal) policy, we allow the forward procedure to "discover" advantageous properties of the backward experiences via the learned value function. Second, this decoupling of the forward and backward policies allows us to learn a backward-facing agent even in difficult settings, by using a relaxed backwards goal.

We apply our approach to the challenging puzzle game of Sokoban, where a player needs to push boxes into storage positions (see Figure 1). Solution paths in Sokoban are much longer than in Chess or Go, the agent faces the danger of irreversible moves, and the number of relevant training levels is limited. As a consequence, Sokoban remains a game where human performance far surpasses that of automated solvers.

Quite remarkably, using a small training set of just 155 beginner levels and simple RL and search algorithms, we are able to achieve state of the art results and solve 88 of the 90 benchmark XSokoban (Myers 1995) levels. To put this in perspective, without relying on the backward agent we solve only 60 levels, similarly to previously published works (Junghanns and Schaeffer 2001; Demaret, Van Lishout, and Gribomont 2008). In fact, using our approach we do not only dramatically surpass other ML-based attempts, but also improve the results of non-academic highly crafted systems.

Look Back Before Forward RL

As discussed, instead of mimicking the trajectories of a backward agent, we build simple *hint features* extracted from backward trajectories, and use these to augment the state representation of the forward agent. We now briefly describe these different components. For a more detailed exposition, see the full version of the paper in (Shoham and Elidan 2021).

The Backward Agent and Backward Trajectories

The backward model is learned by starting the agent in the goal position(s) where rewards are received. The first step is thus to design a reversed task via the construction of reversed rewards. Importantly, since we do not plan to imitate the policy of the backward agent, the reverse task is decoupled from the forward one and we can use a relaxed objective. This can be crucial in domains where the original reverse task is as difficult as the forward one.

As an example, in the case of Sokoban, the forward objective is to push boxes to target squares and the (hard) reverse task to to bring the boxes back to their original positions. A much simpler goal is defined by giving the backward agent a single reward for simply pulling all boxes away from their target locations, but not necessarily to their initial squares.

The learned backward value function $\mathcal{V}_B$ can now be used both at forward-learning and at inference time to generate a backward trajectory T: for a particular (train or test) instance the agent is initialized at goal state(s), and a planning algorithm is used to find T that maximizes $\mathcal{V}_B$.

The Forward Agent and Hint-Features

To learn a forward-facing value function $\mathcal{V}_F$, we use any standard RL algorithm with the original rewards and original starting position. The key difference is that we augment the state-space of the agent with *hint features* that are computed based on the current state s and the instance specific backward trajectory T. Intuitively, such features are informed (by the backward agent) hints that can point the forward agent in the right direction, both at learning and inference time. The forward learning procedure assigns the appropriate weights to these features, in the context of the original forward task. Importantly, as we shall see below, extremely simplistic *hint features* that involve minimal domain understanding can lead to substantial performance improvements.

Application to Sokoban

To apply our approach to the the game of Sokoban, we use linear approximation for both $\mathcal{V}_B$ and $\mathcal{V}_F$ with the following base features: : *Targets* (number of boxes already on target), *Distance* (between the boxes and the nearest target) and *Connectivity* (the number of distinct regions on the board).

Given the trajectory of the backward agent, we augment the base features set with two additional *hint features*: *Overlap* (distance of the state to the closest backward state) and *Perm* (packed boxes according to the order implied by the backward trajectory). The planning algorithm for maximizing $\mathcal{V}_B$ and $\mathcal{V}_F$ is a search tree with an ϵ-greedy traversal from the root. During the learning phase, expanding a leaf updates the value function using standard $TD(0)$.

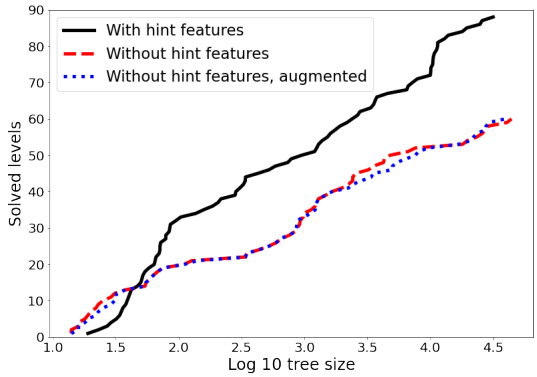


Figure 2: Summary performance on the 90 XSokoban test levels. The graph shows the number of solved levels as a function of the search tree size at inference time.

Experimental Evaluation

We train both the forward and backward agents using only the 155 Microban levels (Skinner 2000) which have been developed as "a good set for beginners and children".

We evaluate performance on the standard 90 XSokoban levels (Myers 1995). Each test level is processed at inference time as follows. We first let the backward agent tackle it, starting from the goal state. We then use the trajectory of that agent to compute *hint features*, and then apply the forward facing agent with these features.

We compare our solver to a baseline trained in an identical manner but without the *hint features* that are generated by the backward agent. We also compare to a baseline based on reverse curriculum learning (Florensa et al. 2017) by augmenting the training data with easier instances.

The results are summarized in Figure 2. Using our look backward before forward approach (solid black line), we solve 88 of the 90 XSokoban levels. This surpasses the highly crafted solvers Sokolution (Diedler 2017), Takaken (Takahashi 2008) and YASS (Damgaard 2000), and is second only to the Festival solver (Shoham and Schaeffer 2020).

The added value of the *hint features* is obvious and without them we are only able to solve 60 levels. Further, augmenting the experiences of the forward solver with near-goal trajectories leads only to a minor improvement. This emphasizes the importance of *deriving* features from the backward agent rather than trying to mimic it.

Summary

We presented a novel RL approach for sparse-reward scenarios where the effective policy near the goal state is quite different from the policy that is beneficial earlier on. Our approach relies on the idea of using *hint features* constructed from backward trajectories to help guide the forward facing agent, both at training and inference time.

Using this forward-backward RL approach, we were able to achieve SOTA results on the challenging game of Sokoban, while training only on a small set of practice levels, with a naive linear function approximation for the value function, and using straightforward RL and search techniques.

References

Damgaard, B. 2000. YASS Sokoban solver. https://sourceforge.net/projects/sokobanyasc/. Accessed: 2021-05-28.

Demaret, J.-N.; Van Lishout, F.; and Gribomont, P. 2008. Hierarchical planning and learning for automatic solving of sokoban problems. *Belgian/Netherlands Artificial Intelligence Conference* 57–64.

Diedler, F. 2017. Sokolution Sokoban solver. http://codeanalysis.fr/sokoban/. Accessed: 2021-05-28.

Edwards, A. D.; Downs, L.; and Davidson, J. C. 2018. Forward-Backward Reinforcement Learning. *CoRR* .

Florensa, C.; Held, D.; Wulfmeier, M.; Zhang, M.; and Abbeel, P. 2017. Reverse Curriculum Generation for Reinforcement Learning. In Levine, S.; Vanhoucke, V.; and Goldberg, K., eds., *Proceedings of the 1st Annual Conference on Robot Learning*, 482–495.

Goyal, A.; Brakel, P.; Fedus, L.; Singhal, S.; Lillicrap, T.; Levine, S.; Larochelle, H.; and Bengio, Y. 2019. Recall Traces: Backtracking Models for Efficient Reinforcement Learning. In *International Conference on Learning Representations*.

Junghanns, A.; and Schaeffer, J. 2001. Sokoban: Enhancing General Single-Agent Search Methods Using Domain Knowledge. *Artificial Intelligence* 129(1–2): 219–251.

Myers, A. 1995. XSokoban home page. http://www.cs.cornell.edu/andru/xsokoban.html. Accessed: 2021-05-28.

Shoham, Y.; and Elidan, G. 2021. Solving Sokoban with Forward-Backward Reinforcement Learning. https://arxiv.org/abs/2105.01904. Accessed: 2021-05-28.

Shoham, Y.; and Schaeffer, J. 2020. The FESS Algorithm: A Feature Based Approach to Single-Agent Search. In *2020 IEEE Conference on Games (CoG)*, 96–103. doi:10.1109/CoG47356.2020.9231929.

Skinner, D. 2000. Microban puzzle pack. https://www.sokobanonline.com/play/web-archive/david-w-skinner. Accessed: 2021-05-28.

Takahashi, K. 2008. Takaken Sokoban solver. http://www.ic-net.or.jp/home/takaken/e/soko/. Accessed: 2021-05-28.