A Framework for Reinforcement Learning and Planning: Extended Abstract*

Thomas M. Moerland,^{1,2} **Joost Broekens,**^{2} **Catholijn M. Jonker**^{1,2}
^{1} Interactive Intelligence, Delft University of Technology, The Netherlands
^{2} Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands

Abstract

Two successful approaches to Markov Decision Process optimization are planning and reinforcement learning. Both research communities operate large separate. This framework attempts to bridge both fields, by disentangling their common algorithmic space, showing that both fields face exactly the same algorithmic decisions. The full paper is available from https://arxiv.org/pdf/2006.15009.pdf.

Sequential decision making, commonly formalized as Markov Decision Process (MDP) optimization, is a key challenge in artificial intelligence research. The two prime research directions in this field are *reinforcement learning* (Sutton and Barto 2018), a subfield of machine learning, and *planning* (also known as *search*), of which the discrete and continuous variants have been studied in the fields of artificial intelligence (Russell and Norvig 2016) and control (Bertsekas 1995), respectively. Planning and learning approaches differ with respect to a key assumption: is the dynamics model of the environment known (planning) or unknown (reinforcement learning).

Departing from this distinctive assumption, both research fields have largely developed their own methodology, in relatively separated communities. There has been cross-breeding as well, better known as 'model-based reinforcement learning' (recently surveyed by Moerland, Broekens, and Jonker (2020)). While the combination of planning and learning has shown great empirical success (Silver et al. 2017), literature still lacks a fundamental view on the relation between both fields, and how their approaches overlap and differ.

Therefore, this paper* introduces the Framework for Reinforcement learning and Planning (FRAP), which identifies the essential algorithmic decisions that any planning or RL algorithm has to make. It consists of six main dimensions, which we will shortly discuss in more detail. However, the main message of the framework is that any RL or planning algorithm, from Q-learning (Watkins and Dayan 1992) to A* (Hart, Nilsson, and Raphael 1968), will have to make a decision on each of these dimensions. Therefore, planning and learning are not only related, but really two sides of the same coin. We illustrate this point in the full paper*, by formally comparing a variety of planning and RL papers along the dimensions of our framework.

Framework

We will here shortly introduce the structure of FRAP. It consists of six main dimensions, of which some have multiple sub-considerations, which are summarized in Table 1. FRAP centers around the concept of *trials* and *back-ups*. A trial is a single call to the environment, where we impute a state-action pair and get back a next state (distribution) and associated reward (distribution). Trials are the fundamental way in which we get information about the environment. After one or more trials, we want to back-up the acquired information to better decide where we want to make the next trial. We may disentangle this process into six key questions:

1. *Where to put our computational effort?*
   We first determine for which states we seek a solution at all. As a crucial distinction, we may either consider all states (as used by Dynamic Programming approaches), or only the reachable states (which we can track by only sampling from a start state distribution).
2. *Where to make the next trial?*
   We then determine where to make the next trial. There are several relevant considerations, like which set of state-action pairs are candidate for selection in the current iteration (e.g., all actions at the current state, or all state-actions at the frontier), and how to add exploration (since greedy selection leads to suboptimal behaviour).
3. *How to estimate the cumulative return?*
   After we make the trial, we need an estimate of the remaining cumulative reward after the trial. We can either sample and/or bootstrap, which we both need to decide on.
4. *How to back-up?*
   We then want to back-up this new information (obtained from the trial). We need to decide on the back-up policy, and on how to deal with the expectations over the actions and dynamics in the one-step Bellman equation. As an example, both a well-known planning algorithm like MCTS

*Full paper available at: https://arxiv.org/pdf/2006.15009.pdf

Table 1: Overview of dimensions in the Framework for Reinforcement learning and Planning (FRAP). For any planning or reinforcement learning algorithm, we should be able to identify the decision on each of the dimensions. The subconsiderations and possible options are shown in the right columns. IM = Intrinsic Motivation.

Dimension	**Consideration**	**Choices**
1. Comp. effort	- State set	All $\leftrightarrow$ reachable $\leftrightarrow$ relevant
2. Trial selection	- Candidate set	Step-wise $\leftrightarrow$ frontier
	- Exploration	Random $\leftrightarrow$ Value-based $\leftrightarrow$ State-based -For value: mean value, uncertainty, priors -For state: ordered, priors (shaping), novelty, knowledge IM, competence IM
	- Phases	One-phase $\leftrightarrow$ two-phase
	- Reverse trials	Yes $\leftrightarrow$ No
3. Return estim.	- Sample depth	$1 \leftrightarrow n \leftrightarrow \infty$
	- Bootstrap func.	Learned $\leftrightarrow$ heuristic $\leftrightarrow$ none
4. Back-up	- Back-up policy	On-policy $\leftrightarrow$ off-policy
	- Policy expec.	Expected $\leftrightarrow$ sample
	- Dynamics expec.	Expected $\leftrightarrow$ sample
5. Representation	- Function type	Value $\leftrightarrow$ policy $\leftrightarrow$ both (actor-critic) - For all: generalized $\leftrightarrow$ not generalized
	- Function class	Tabular $\leftrightarrow$ function approximation - For tabular: local $\leftrightarrow$ global
6. Update	- Loss	- For value: e.g., squared -For policy: e.g., (det.) policy gradient $\leftrightarrow$ value gradient $\leftrightarrow$ cross-entropy, etc.
	- Update	Gradient-based $\leftrightarrow$ gradient-free - For gradient-based, special cases: replace & average update

(Kocsis and Szepesvári 2006) and a well-known RL algorithm like SARSA (Rummery and Niranjan 1994) make the same algorithmic choice here (an on-policy, sample action, sample dynamics back-up).

5. *How to represent the solution?* We also want to be able to store the new information. Here, planning and reinforcement learning have emphasized different approaches, since planning methods mostly focus on tabular/atomic representations (like nodes), while reinforcement learning approaches have emphasized approximate (learned) representations of the solution.
6. *How to update the solution?* Finally, we need to update our solution (from 5) based on the back-up estimate (from 4). We can distinguish gradient-based and gradient-free updates. FRAP also shows how common planning updates can be cast into this categorization.

The framework shows that planning and learning essentially do the same thing. As an illustration, note that a MCTS of 500 traces is conceptually not too different from 500 episodes of a model-free Q-learning agent in the same environment. In both cases, we repeatedly move forward in the environment to acquire new information, make back-ups to store this information, with the goal to make better informed decisions in the next trace/episode. The model-free RL agent is restricted in the order in which it can visit states, but otherwise, the methodology of exploration, back-ups, representation and updates is the same.

The main paper includes a large table comparing a variety of planning, model-free RL and model-based RL papers along the dimensions of our framework, which illustrates the validity of FRAP. In short, FRAP provides a common language to categorize algorithms in both fields, hopefully serving as a bridge between both. We hope it also inspires new research, for example by identifying novel possible combinations of planning and learning, or stimulating the design of a new algorithm in one field based on inspiration from the other.

References

Bertsekas, D. P. 1995. *Dynamic programming and optimal control*, volume 1.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* 4(2):100–107.

Kocsis, L., and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *ECML*, volume 6, 282–293. Springer.

Moerland, T. M.; Broekens, J.; and Jonker, C. M. 2020. Model-based Reinforcement Learning: A Survey. *arXiv preprint arXiv:2006.16712*.

Rummery, G. A., and Niranjan, M. 1994. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, England.

Russell, S. J., and Norvig, P. 2016. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Silver, D.; van Hasselt, H.; Hessel, M.; Schaul, T.; Guez, A.; Harley, T.; Dulac-Arnold, G.; Reichert, D.; Rabinowitz, N.; Barreto, A.; et al. 2017. The predictron: End-to-end learning and planning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 3191–3199. JMLR. org.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Watkins, C. J., and Dayan, P. 1992. Q-learning. *Machine learning* 8(3-4):279–292.