Research Proposal for PhD Thesis

Erwan Lecarpentier

The 11th of January 2017

1 Abstract

How to learn optimal controller for a learning system within an uncertain non-stationary environment? Imagine yourself trapped into a maze whose walls are moving. Your goal is to reach the exit as quickly as possible but you do not have a clue of its location. One could chose to evolve randomly into this environment, one could try to draw a mental map of the labyrinth based on one's experience, one could try to figure out "How do the walls move? Is it randomly? Is there a kind of logic?" and so forth. This example can be characterized as a problem of sequential decision making within an uncertain non-stationary environment. Reinforcement Learning (RL) algorithms provide a variety of solutions to the latter. Including uncertainty about the environment is a common consideration, time dependency to a lesser extent since it is a challenging concern. This PhD Thesis aims at studying, designing and evaluating learning algorithms for the generation of optimal controllers in this context. We investigate various RL architectures among model-free and model-based approaches and focus on the application to the control of Unmanned Aerial Vehicles (UAVs) motivated by the fact that flying environments are rarely stationary, e.g. unsteady wind-field, thermals magnitude varying over time, obstacles.

Keywords: Reinforcement Learning; Planning; Uncertain Non-stationary Environment; Monte Carlo Tree Search; UAV Control.

2 Proposed approach

In the matter of deriving an optimal controller, Reinforcement Learning (RL) approaches usually follow one of the three alternative paths presented in 1, classically lying under the paradigm of Markov Decision Processes (MDPs) [9, 10]. In each cases, the goal is to learn a policy, optimal in the sense of a scalar reward function, based on data collected during interactions between the agent and the environment. The class of problems considered here, a.k.a. acting under non-stationarity and uncertainty, requires a quick learning of the policy due to the changes of the environment. It is in a sense comparable to game theory where the unpredictable actions of the opponent can be seen as a non-stationarity of the environment, possibly in a worst case sense. In this field, Search based algorithms encountered a great success in the past decades due to their capability to quickly identify a close-to-optimal action and to make predictions on the long term [1, 5, 4, 3, 8]. Therefore, this class of algorithms suits well to the considered problem. However, pure Tree Search methods often learn a close-to-optimal action given a specific state but do not store the collected informations. In

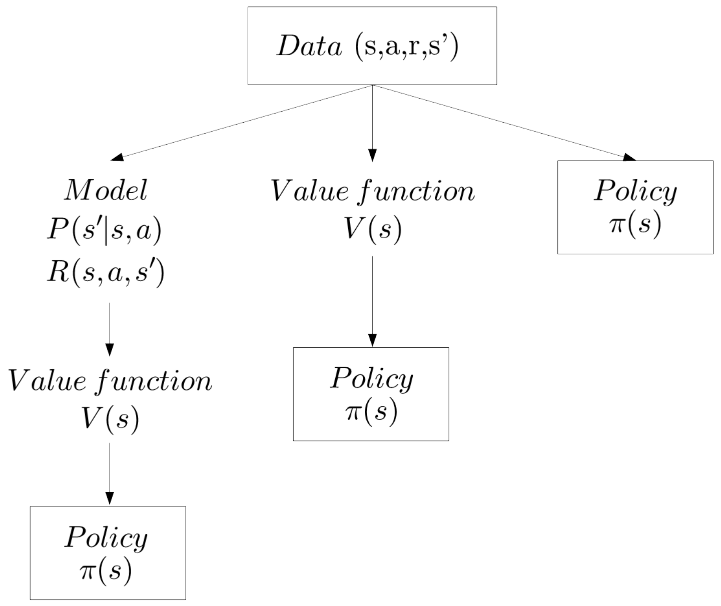

Figure 1: Three standard RL approaches, from left to right: Model-based RL where assumptions on the dynamic of the environment and the reward model are exploited in order to derive a value function; Value-based RL where a value function is learned from samples and used to derive a policy; and Direct-RL where a policy is directly learned from samples.

other words, they do not learn a global policy or value function which is justified by the fact that such a global function may not exist given the non-stationarity of the problem. The latter fact is questionable. For instance one can rely on a slow evolution of the environment thanks to which a global policy may exist. One can also imagine to capture some global features of the environment in a single policy to which local corrections would be added. In other words, one can believe that we could learn something valid for the whole environment thanks to local data samples, this point being a question to which we would like to answer in the Thesis.
Back to the big picture of 1, Search-based algorithms can be found in the left part of the figure, alias Model-based RL. Particularly, the Dyna architecture proposed in [9] describes an agent simultaneously learning a value function and a model of the environment onto which planning is performed at each control iteration. Here, planning has to be understood as an additional learning step based on data generated by the model. Doing so accelerates the learning process assuming that the model is valid, hence consisting in a very useful feature for non-stationary problems. From the points discussed so far, several interesting features of what would be a "good learner" in our case of study can be listed as follows:

- Learning a global policy/value function, universal in the sense of the state i.e. always valid;
- Learning a local policy/value function, capturing the specificities of the environment at a certain state at a certain time;

- Learning a model of the environment based on data;
- Performing planning with search-based methods on this model in order to accelerate the policy/value learning;

From the general considerations listed above, we describe now a specific algorithm that we hope consisting in a good search direction for the beginning of the Thesis. The Dyna-2 algorithm [7] is based on a Dyna architecture and presents several advantages discussed above, namely the learning of a global and a local approximated Q-function and a planning step improving the policy at each control iteration. Both planning and learning use Sarsa updates to improve their respective state-action value functions which are considered linear in features. Some difficulty of such an approach is the tuning of the learning rate and the selection of the feature functions. A way of avoiding the first issue can be to use policy evaluation methods such as LSTDQ [6] and in order to prevent from feature selection, one can use a kernel-based version of this method, namely K-LSTDQ [11]. Those modifications of Dyna-2 consist finally in having an actor-critic approach within the Dyna architecture. If this kind of approach is often associated with batch-learning, one can also apply those methods online with some considerations about the computational cost [2]. Figure 2 illustrates the architecture of the suggested algorithm.

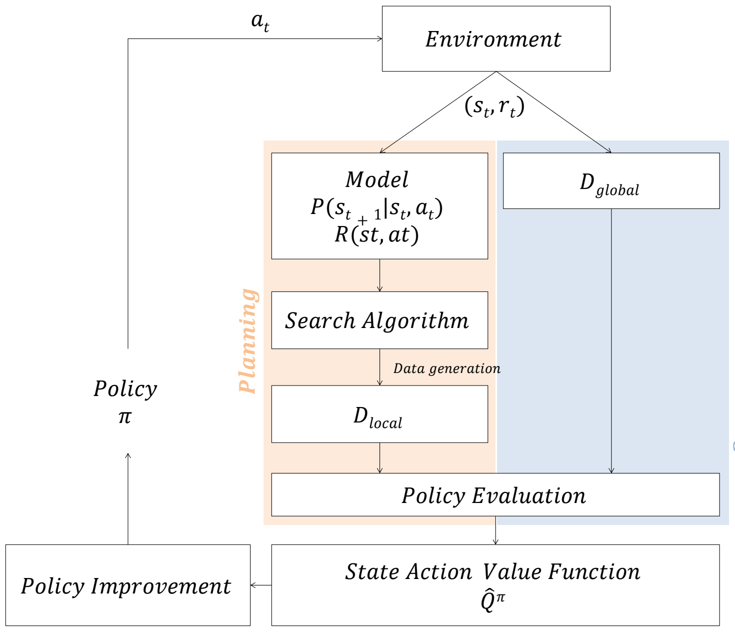

Figure 2: General architecture of an actor-critic planning and learning online algorithm.

3 Results

In order to evaluate the performances of the derived algorithms, we plan to perform numerical simulations based on the problem of thermal soaring with a glider UAV. An interesting opportunity is to make use of a C++ coded simulator currently developed within the project Learning2Fly at ISAE-Supaero. This simulator provides a realistic model of a dynamic environment including thermals.

References

[1] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[2] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement learning and dynamic programming using function approximators*, volume 39. CRC press, 2010.

[3] Sylvain Gelly and David Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856–1875, 2011.

[4] Thomas Keller and Patrick Eyerich. Prost: Probabilistic planning based on uct. In *ICAPS*, 2012.

[5] Thomas Keller and Malte Helmert. Trial-based heuristic tree search for finite horizon mdps. 2013.

[6] Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4(Dec):1107–1149, 2003.

[7] David Silver, Richard S Sutton, and Martin Müller. Sample-based learning and search with permanent and transient memories. In *Proceedings of the 25th international conference on Machine learning*, pages 968–975. ACM, 2008.

[8] David Silver, Richard S Sutton, and Martin Müller. Temporal-difference search in computer go. *Machine learning*, 87(2):183–219, 2012.

[9] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[10] Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

[11] Xin Xu, Dewen Hu, and Xicheng Lu. Kernel-based least squares policy iteration for reinforcement learning. *IEEE Transactions on Neural Networks*, 18(4):973–992, 2007.