Algorithm and Parameters

In the following, we give a brief description of LEAP (Learning Entities Adaptive Partitioning), a new model-free adaptive reinforcement learning algorithm, that we have benchmarked.

In LEAP, the state space is initially decomposed into different partitions, each entrusted to a different Learning Entity (LE) that runs a Q-learning-like learning algorithm. Each partition aggregates states of the original state space into macrostates and for each \((\text{macrostate}, \text{action})\) pair LE computes the Q-value and a measurement of the reliability of such a value. The learning process of LEAP is on-line and does not need any information about the transition model of the environment. The action selection phase is performed by a Learning Mediator (LM) that merges all the action-values learned by LEs and computes the best action in the current state. In all the experiments in the benchmark we adopted a simple mean of Q-values weighted by their variability, in order to strengthen only reliable estimations. During the update phase, each LE compares the expected reward with the target actually received and through a heuristic criterion, the consistency test, detects whether the resolution of its own partition in the current state should be increased, in that case the LE is said to be inconsistent. When more than one LE is inconsistent, the LM creates a new Joint Learning Entity (JLE), that will operate on a new single macrostate obtained as the intersection of the macrostates of the inconsistent LEs. As a consequence, the basic LE will be deactivated on all the states covered by the more specialized entity. At the same time, an opposite mechanism, the pruning mechanism, detects, during the action selection, when a JLE can be removed from the list of LEs. This mechanism simply compares the action proposed by a JLE with the action that would be proposed by the deactivated LEs, and, when those actions are the same, the JLE can be removed.

Through these mechanisms (consistency test and pruning), LEAP builds a multi-resolution state representation that is specialized only where it is needed to learn a near optimal policy. LEAP needs some parameters to be set depending on the specific problem to be solved. Most of them are typical of on-line algorithms:

- **Learning Rate**: the learning rate used by all the LEs and JLEs during their update phase.
- **Exploration Factor**: the value of \(\epsilon\) in a simple \(\epsilon\)-greedy exploration strategy.
- **Decreasing Rate**: the rate used to decrease both learning rate and exploration factor.\(^1\)

Furthermore, LEAP introduces two additional parameters used to put a lower threshold on the minimum exploration of macrostates needed before consistency test and pruning mechanism actually taking place in a macrostate:

- **MinExplorationC**: number of times the least taken action must be executed before the consistency test takes place.
- **MinExplorationP**: number of times the least taken action must be executed before the pruning mechanism takes place.

Both these parameters must be chosen in order to avoid, because of initialization effects, incorrect refining of the state space (i.e. the consistency test) and early pruning of macrostates (i.e. pruning mechanism).

Results

Since LEAP is an adaptive algorithm whose goal is to give compact solutions to complex problems, in all the following results we pursue a tradeoff between optimality and number of macrostates used to compute the solution.\(^1\)

\(^1\)Since aggregation induces additional stochasticity in the environment, even in deterministic problems the learning rate cannot be constant.