Random-Bandit: An Online Planner

Alan Fern, Murugeswari Issakkimuthu and Prasad Tadepalli
School of EECS, Oregon State University
Corvallis, OR 97331, USA

Abstract

Random-Bandit is an online planner based on the ϵ-greedy algorithm for multi-armed bandit problems (Kuleshov and Precup 2000). Every planning step is regarded as an independent multi-armed bandit problem at the current state with the set of applicable actions as the arms of the bandit. The ϵ-greedy algorithm for the multi-armed bandit problem estimates the average reward of each arm by pulling the current best arm with probability $1-\epsilon$ and one of the remaining arms with probability ϵ, and finally returns the arm with the highest average reward. The ϵ-greedy algorithm of *Random-Bandit* estimates $Q^{\pi}_{h}(s, a)$ for the random policy (π) for each action (a) applicable in the current state (s) for horizon h and returns $\hat{a} = arg\max_a Q^{\pi}(s, a)$.

Introduction

The planner *Random-Bandit* has been implemented as a component of *Prost* (Keller and Eyerich 2012) as it relies on many existing functionalities in *Prost*. *Prost* is the state-of-the-art search-based online planner for *RDDL* domains. Figure 1 shows the schematic diagram of the entire planning system. *RDDLSim* (Sanner 2010) is the *RDDL* server used for evaluation in the competition. *Prost* initiates (and

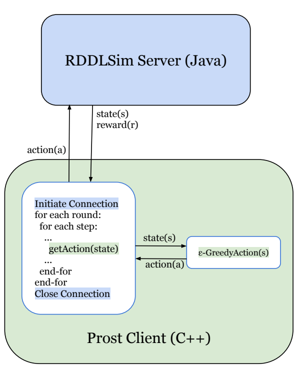

Figure 1: Schematic Diagram

also terminates) the communication with the server, receives and parses the *RDDL* domain and problem files, and initializes the required data structures. The nested *for* loops in the figure denote the evaluation loop in which *Prost* returns an action for the current state to the server and receives the reward and next state from the server. At each planning step *Prost* calls the *Random-Bandit* function ϵ-GreedyAction(s) with the current state s and returns the received action $\hat{a}$ to the server instead of invoking its own planning routines.

The ϵ-Greedy Algorithm

The ϵ-Greedy algorithm estimates $Q^{\pi}(s, a)$ for each action $a \in A_s$ applicable in state s for the random policy π for horizon h and returns $\hat{a} = arg\max_a Q^{\pi}(s, a)$. In Algorithm 1 below, the function *random-number*$(0, 1)$ returns a random number between 0 and 1, *random-action*$(A_s \backslash \{\hat{a}\})$ returns a random action from the set A_s excluding action $\hat{a}$, and *next-state*(s, a) returns the next state s' and reward r as a result of taking action a in state s.

Algorithm 1 ϵ-GreedyAction(s)

Initialize $Q^{\pi}(s, a) \leftarrow 0, \forall a \in A_s$
Initialize $N(a) \leftarrow 0, \forall a \in A_s$
Initialize $\hat{a} \leftarrow$ random-action(A_s)
repeat
  $r \leftarrow$ random-number$(0, 1)$
  if $r > \epsilon$ **then**
    $a \leftarrow \hat{a}$
  else
    $a \leftarrow$ random-action$(A_s \backslash \{\hat{a}\})$
  end if
  $N(a) \leftarrow N(a) + 1$
  $(s', r) \leftarrow$ next-state(s, a)
  $R \leftarrow r$
  $s \leftarrow s'$
  for $i = 1..h$ **do**
    $(s', r) \leftarrow$ next-state$(s, \pi(s))$
    $R \leftarrow R + r$
    $s \leftarrow s'$
  end for
  $Q^{\pi}(s, a) \leftarrow Q^{\pi}(s, a) + (R - Q^{\pi}(s, a))/N(a)$
  if $Q^{\pi}(s, a) > Q^{\pi}(s, \hat{a})$ **then**
    $\hat{a} \leftarrow a$
  end if
until time-limit is not reached
return $\hat{a}$

Implementation Details

The important functionalities in *Prost* used in implementing *Random-Bandit* are

1. The *IPPCClient* class for establishing (and terminating) the connection with the *RDDL* server, parsing the *RDDL* domain and problem files and initializing data structures, and running the evaluation loop receiving state and reward signals and sending actions
2. The *RandomWalk* class for simulating a trajectory from state s starting with action a and then following the random policy π for h steps accounting for steps 12 through 19 in algorithm 1
3. The *IDS* class to estimate the best rollout horizon h for the problem by means of iterative deepening search

Parameter Settings: The main parameters of the algorithm are ϵ, the rollout horizon h, and the decision-time for each planning step. ϵ is set to 0.5. The rollout horizon h is initialized to the minimum of 7 or the value returned by the *IDS* class and reduced to the number of remaining steps for planning steps near the end of an episode. The decision-time is set to 75% of the average time available for each step re-computed at the beginning of each round.

Acknowledgements

Many thanks to Dr. Thomas Keller for his help with resolving problems connected to *Prost* functionalities.

References

Keller, T., and Eyerich, P. 2012. PROST: Probabilistic Planning Based on UCT. In *Proceedings of the International Conference on Automated Planning and Scheduling*.

Kuleshov, V., and Precup, D. 2000. Algorithms for the multi-armed bandit problem. *Journal of Artificial Intelligence Research (1)* 1–48.

Sanner, S. 2010. Relational Dynamic Influence Diagram Language (RDDL): Language description.