Feature Reinforcement Learning In Practice

September 9, 2011

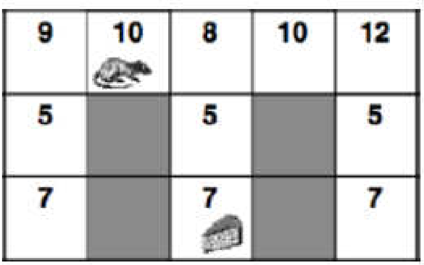
Phuong Nguyen[1,2] (PhD student)
Joint work with **Peter Sunehag**[1] and **Marcus Hutter**[1,2]
Australian National University[1], *National ICT Australia*[2]

Problems

- Robotic control in an unknown environment

Problems

- Perceptual aliasing



Problem formulation

$$
\begin{aligned}
h_t &= a_1 o_1 r_1 o_2 r_2 a_2 \ldots o_t r_t \\
a_t &= \mathbf{Agent}(h_t) \\
o_{t+1} r_{t+1} &= \mathbf{Environment}(h_t a_t)
\end{aligned}
$$

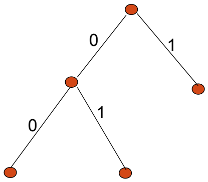
$$\Phi\text{: Histories} \rightarrow \text{States}$$

$$s_t = \Phi(h_t)$$

- Φ is to **reduce the general RL problem to an MDP**, so that we can use MDP solvers to find the solution
- Aim at finding Φs that result in MDPs with **good reward-prediction capability**

ΦMDP framework

- What does the function Φ look like?
  ⇒ one of the most useful classes of maps is **context trees**
- Example:
  $\mathcal{S} = \{s_1, s_2, s_3\} = \{00, 10, 1\}$
  $\Phi_{\mathcal{S}}$ is the map represented by $\mathcal{S}$
  $h_6 = 011001$
  - $\Phi_{\mathcal{S}}(h_4) = 10(s_2)$
  - $\Phi_{\mathcal{S}}(h_5) = 00(s_3)$
  - $\Phi_{\mathcal{S}}(h_6) = 1(s_1)$



ΦMDP framework

- How good is a Φ?
  ⇒ **predictive ability**

$$\mathbf{Cost}(\Phi|h_n) = \mathbf{CL}_\Phi(s_{1:n}|a_{1:n}) + \mathbf{CL}_\Phi(r_{1:n}|s_{1:n}, a_{1:n})$$

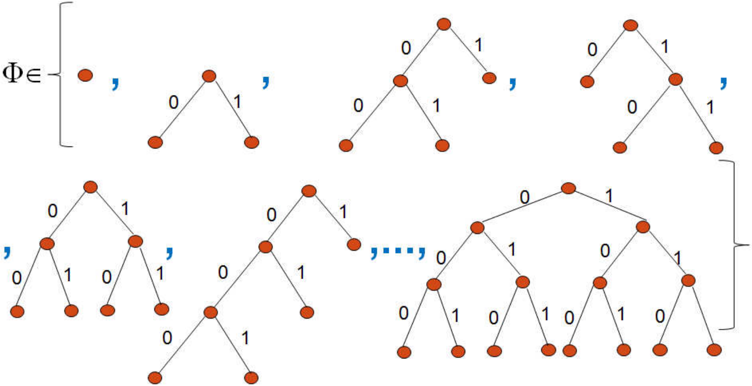
$$(\mathbf{CL} = \text{Code Length})$$

  - Inspired by MDL(Minimum Description Length) principle

M. Hutter, *Feature Reinforcement Learning: Part I: Unstructured MDPs*, Journal of General Artificial Intelligence, 2009

ΦMDP framework

- The optimal solution $\Phi = \arg\min_{\Phi} \textbf{Cost}(\Phi | h_n)$



Generic stochastic ΦMDP search (GSΦA)

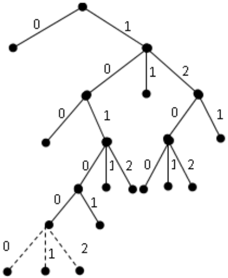
1. Generate a history using random policy

2. **Given the initial history, run stochastic search to find an estimate $\hat{\Phi}$ of Φ^{optimal}**
3. **Solve the MDP induced from $\hat{\Phi}$ using Action-Value Iteration (AVI)**
4. **Start acting based on the AVI solution, and further apply Q-Learning to refine the MDP solution**
5. **Add the history obtained from Q-Learning to the old history**

6. Go back to step 2
7. Return the $\hat{\Phi}$ with lowest cost, and the corresponding optimal policy computed from Q^*

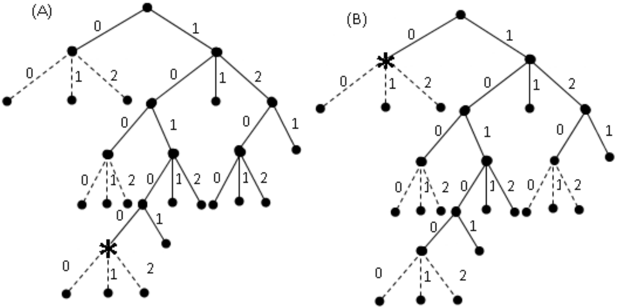
Stochastic search - Parallel tempering

- Search space: Markov Action-Observation Context Trees (Closed Finite State Machines)
  - Markov trees are trees where given s_t and a_t, o_{t+1}, we know s_{t+1}
- Parallel tempering algorithm:
  - Run a number of traditional simulated annealings in parallel
  - Swap configurations to speed up the search process



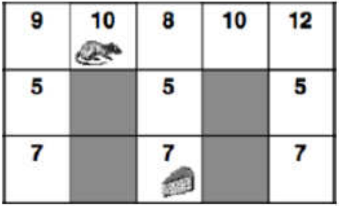
Stochastic search - Parallel tempering

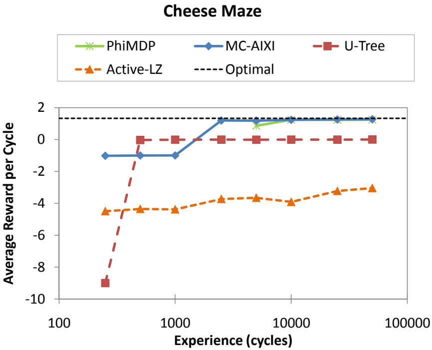
- Proposal distribution
  - Propose to split and merge some leaf node
  - Keep the search trees in the space of Markov action-observation context trees. In order to do this, we might have to perform a chain of splits or merges
  - Keep useful short-term memory if found, and share it with all other trees



Domain and results

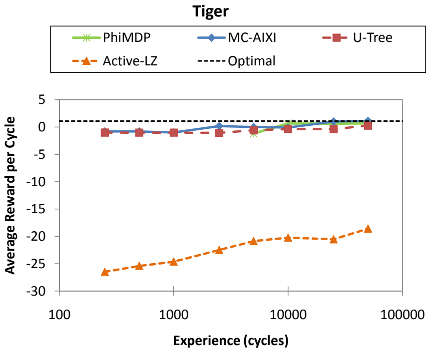
- Cheese maze





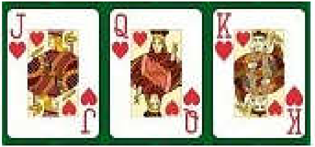
Domain and results

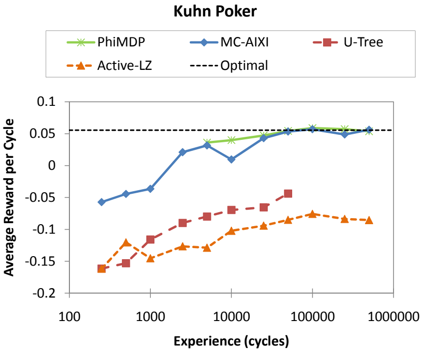
- Tiger



Domain and results

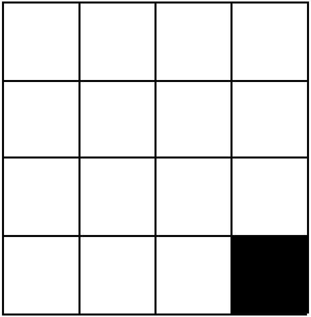
- Kunh poker

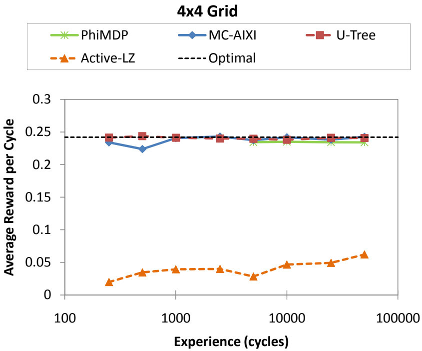




Domain and results

- 4×4 grid





Main contributions

- Limiting search space to Markov action-observation context trees
- Proposing the GSΦA algorithm
- Providing the first empirical analysis of ΦMDP
- Designing a specialized proposal distribution for stochastic search

Key conclusions

- ΦMDP outperforms U-tree, Active-LZ
- ΦMDP is competitive with MC-AIXI in short-term memory domains
- ΦMDP is more efficient than MC-AIXI in both computation and memory usage
- ΦMDP is more flexible in environment modelling than U-tree, Active-LZ, and MC-AIXI through the choice of any class of maps Φ, though other approaches can be combined with predicates

Thank you!