Hierarchical Model-Based Reinforcement Learning: R-MAX + MAXQ

Nicholas K. Jong Peter Stone

Department of Computer Sciences
University of Texas at Austin

International Conference on Machine Learning, 2008

Outline

Outline

Introduction

Problem Learn behaviors in unknown environments

Criterion Minimize number of suboptimal actions taken

Idea 1 Model-Based Reinforcement Learning

- Probabilistic finite-time convergence
- Efficient use of sample data
- Robust exploration using model uncertainty

Idea 2 Hierarchical Reinforcement Learning

- Intuitive approach to scaling to large problems
- Decomposition of tasks into subtasks

Our Contribution

Integration of model-based and hierarchical RL for fully stochastic, finite problems

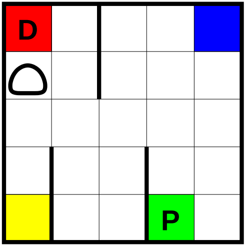
The Taxi Domain

State Variables

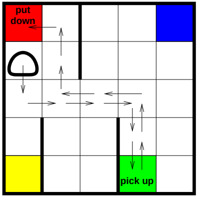
- *x* coordinate
- *y* coordinate
- Passenger location
  (at 1 of 4 landmarks or in the taxi)
- Destination location
  (at 1 of 4 landmarks)

Actions

North, South, East, West, PickUp, PutDown



The Taxi Hierarchy



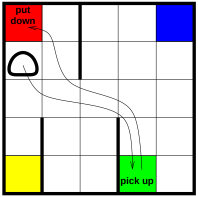
Optimal policy

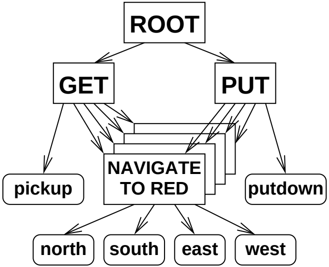
- Navigate to the passenger
- Pick up the passenger
- Navigate to the destination
- Put down the passenger

Composite actions

- Set of child actions A^i
- Set of terminal states $T^i \subseteq S$
- Goal rewards $\tilde{R}^i : T^i \to \mathbb{R}$

The Taxi Hierarchy





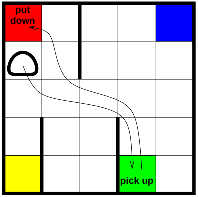
Optimal policy

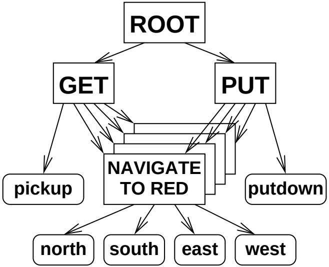
- Navigate to the passenger
- Pick up the passenger
- Navigate to the destination
- Put down the passenger

Composite actions

- Set of child actions A^i
- Set of terminal states $T^i \subseteq S$
- Goal rewards $\tilde{R}^i : T^i \to \mathbb{R}$

The Taxi Hierarchy





Optimal policy

- Navigate to the passenger
- Pick up the passenger
- Navigate to the destination
- Put down the passenger

Composite actions

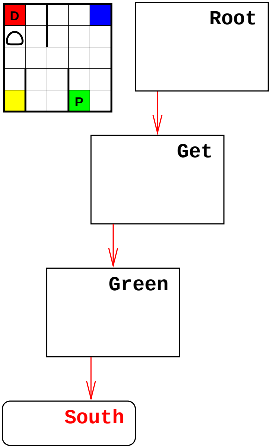
- Set of child actions A^i
- Set of terminal states $T^i \subseteq S$
- Goal rewards $\tilde{R}^i : T^i \to \mathbb{R}$

Outline

MAXQ Decomposition of the Value Function

Decompose value function

- $V^i(s) = \max_a Q^i(s, a)$
  Total expected reward (for action i)
- $Q^i(s, a) = V^a(s) + C^i(s, a)$
  Reward if i executes a first
- $C^i(s, a) = E_{k,s'}\left[\gamma^k V^i(s')\right]$
  Reward i expects after executing a

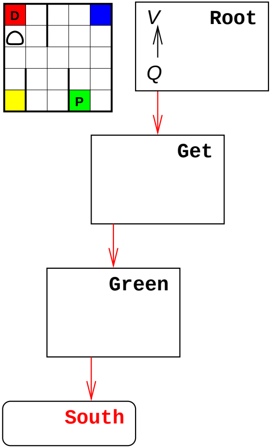


MAXQ Decomposition of the Value Function

Decompose value function

- $V^i(s) = \max_a Q^i(s,a)$
  Total expected reward (for action i)
- $Q^i(s,a) = V^a(s) + C^i(s,a)$
  Reward if i executes a first
- $C^i(s,a) = E_{k,s'}\left[\gamma^k V^i(s')\right]$
  Reward i expects after executing a

$V^{\text{Root}}($  $) \quad = \quad Q^{\text{Root}}($  $, \texttt{Get})$

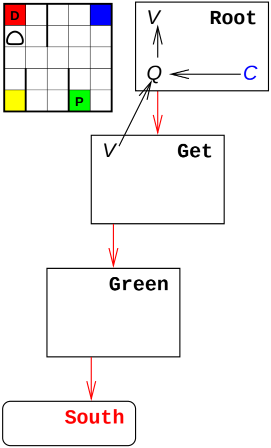


MAXQ Decomposition of the Value Function

Decompose value function

- $V^i(s) = \max_a Q^i(s,a)$
  Total expected reward (for action i)
- $Q^i(s,a) = V^a(s) + C^i(s,a)$
  Reward if i executes a first
- $C^i(s,a) = E_{k,s'}\left[\gamma^k V^i(s')\right]$
  Reward i expects after executing a

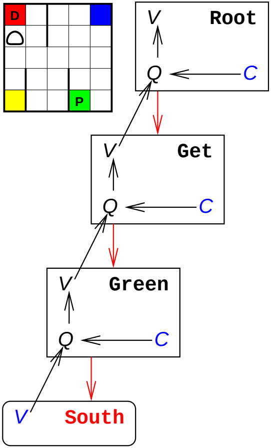
$$
\begin{aligned}
V^{\texttt{Root}}(\;) &= Q^{\texttt{Root}}(\;, \texttt{Get}) \\
&= V^{\texttt{Get}}(\;) + C^{\texttt{Root}}(\;, \texttt{Get})
\end{aligned}
$$



MAXQ Decomposition of the Value Function

Decompose value function

- $V^i(s) = \max_a Q^i(s,a)$
  Total expected reward (for action i)
- $Q^i(s,a) = V^a(s) + C^i(s,a)$
  Reward if i executes a first
- $C^i(s,a) = E_{k,s'}\left[\gamma^k V^i(s')\right]$
  Reward i expects after executing a



$$
\begin{aligned}
V^{\texttt{Root}}(\,) &= Q^{\texttt{Root}}(\,, \texttt{Get}) \\
&= V^{\texttt{Get}}(\,) + C^{\texttt{Root}}(\,, \texttt{Get}) \\
&= V^{\texttt{South}}(\,) + C^{\texttt{Green}}(\,, \texttt{South}) \\
&\quad + C^{\texttt{Get}}(\,, \texttt{Green}) + C^{\texttt{Root}}(\,, \texttt{Get})
\end{aligned}
$$

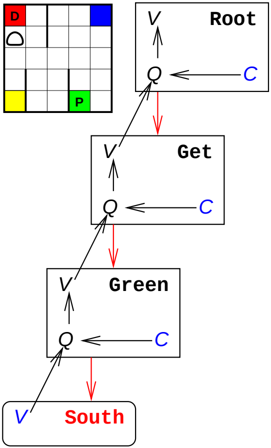
The MAXQ-Q Algorithm

Overview of MAXQ-Q

- Learn V for each primitive action
- Learn C for each composite action
- Use Q-learning-like update rules

Properties of MAXQ-Q

- Facilitates state abstraction:
  Different representation for each C, V
- Learning proceeds bottom-up
- Parameters tuned for each action
- Asymptotic convergence



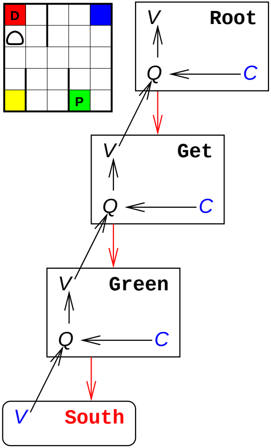
Model Decomposition

MAXQ Model Decomposition

1. Learn models of primitive actions
2. Plan using existing models
3. Compute abstract model
4. Apply induction

$$R^a(s) = R^{\pi^a(s)}(s) + \sum_{s' \in S \setminus T^a} P^{\pi^a(s)}(s, s') R^a(s')$$

$$P^a(s, x) = P^{\pi^a(s)}(s, x) + \sum_{s' \in S \setminus T^a} P^{\pi^a(s)}(s, s') P^a(s', x)$$



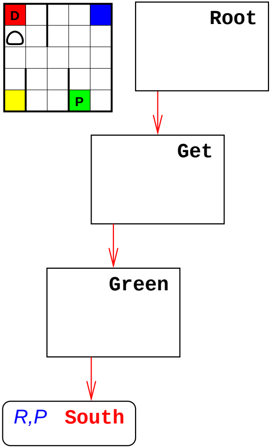
Model Decomposition

MAXQ Model Decomposition

1. Learn models of primitive actions
2. Plan using existing models
3. Compute abstract model
4. Apply induction

$$R^a(s) = R^{\pi^a(s)}(s) + \sum_{s' \in S \setminus T^a} P^{\pi^a(s)}(s, s') R^a(s')$$

$$P^a(s, x) = P^{\pi^a(s)}(s, x) + \sum_{s' \in S \setminus T^a} P^{\pi^a(s)}(s, s') P^a(s', x)$$



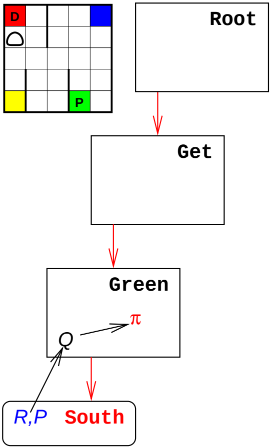
Model Decomposition

MAXQ Model Decomposition

1. Learn models of primitive actions
2. Plan using existing models
3. Compute abstract model
4. Apply induction

$$R^a(s) = R^{\pi^a(s)}(s) + \sum_{s' \in S \setminus T^a} P^{\pi^a(s)}(s, s') R^a(s')$$

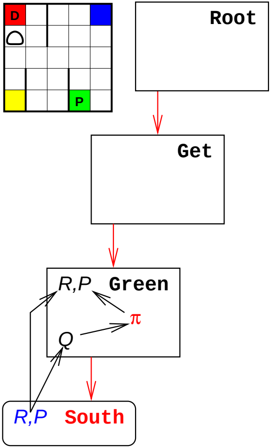
$$P^a(s, x) = P^{\pi^a(s)}(s, x) + \sum_{s' \in S \setminus T^a} P^{\pi^a(s)}(s, s') P^a(s', x)$$



Model Decomposition

MAXQ Model Decomposition

1. Learn models of primitive actions
2. Plan using existing models
3. Compute abstract model
4. Apply induction



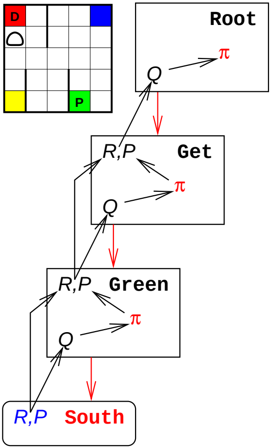
$$R^a(s) = R^{\pi^a(s)}(s) + \sum_{s' \in S \setminus T^a} P^{\pi^a(s)}(s, s') R^a(s')$$

$$P^a(s, x) = P^{\pi^a(s)}(s, x) + \sum_{s' \in S \setminus T^a} P^{\pi^a(s)}(s, s') P^a(s', x)$$

Model Decomposition

MAXQ Model Decomposition

1. Learn models of primitive actions
2. Plan using existing models
3. Compute abstract model
4. Apply induction



$$R^a(s) = R^{\pi^a(s)}(s) + \sum_{s' \in S \setminus T^a} P^{\pi^a(s)}(s, s') R^a(s')$$

$$P^a(s, x) = P^{\pi^a(s)}(s, x) + \sum_{s' \in S \setminus T^a} P^{\pi^a(s)}(s, s') P^a(s', x)$$

Outline

1. Learning with Hierarchies of Models
   - Learning in Structured Environments
   - MAXQ Decomposition

2. The R-MAXQ Algorithm
   - R-MAX Exploration
   - Results

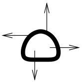
R-MAX Models of Primitive Actions

Maximum-likelihood estimation, given sufficient data

$$R^a(s) = \frac{\text{total reward}}{\text{\# of transitions}} \qquad P^a(s, s') = \frac{\text{\# of transitions to } s'}{\text{\# of transitions}}$$

Optimistic models, given insufficient data

$$R^a(s) = V^{\max} \qquad P^a(s, s') = 0$$



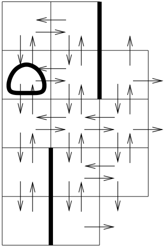
R-MAX Models of Primitive Actions

Maximum-likelihood estimation, given sufficient data

$$R^a(s) = \frac{\text{total reward}}{\text{\# of transitions}} \qquad P^a(s, s') = \frac{\text{\# of transitions to } s'}{\text{\# of transitions}}$$

Optimistic models, given insufficient data

$$R^a(s) = V^{\max} \qquad P^a(s, s') = 0$$



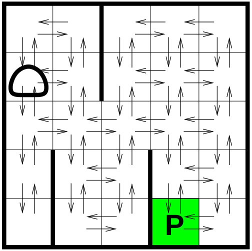
R-MAX Models of Primitive Actions

Maximum-likelihood estimation, given sufficient data

$$R^a(s) = \frac{\text{total reward}}{\text{\# of transitions}} \qquad P^a(s, s') = \frac{\text{\# of transitions to } s'}{\text{\# of transitions}}$$

Optimistic models, given insufficient data

$$R^a(s) = V^{\max} \qquad P^a(s, s') = 0$$



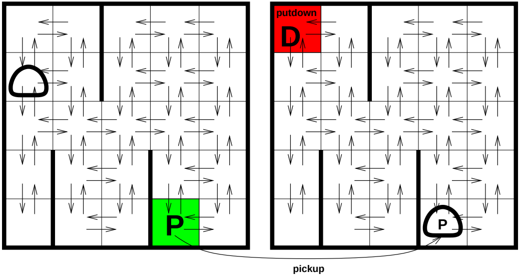
R-MAX Models of Primitive Actions

Maximum-likelihood estimation, given sufficient data

$$R^a(s) = \frac{\text{total reward}}{\text{\# of transitions}} \qquad P^a(s, s') = \frac{\text{\# of transitions to } s'}{\text{\# of transitions}}$$

Optimistic models, given insufficient data

$$R^a(s) = V^{\max} \qquad P^a(s, s') = 0$$

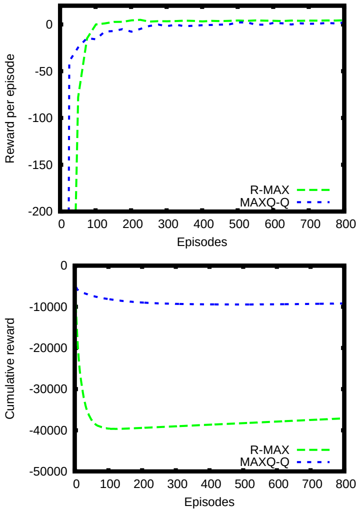


The R-MAX Algorithm

Procedure for each time step

1. Update model
2. Compute value function
3. Choose greedy action

- Thorough exploration due to initial optimism
- Very large negative rewards in exploratory episodes
- High-quality policy after initial exploration

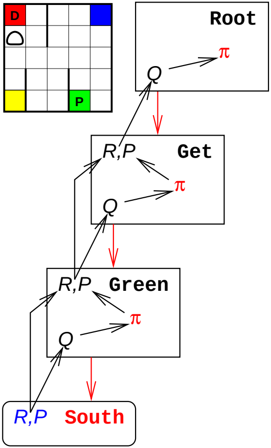


The R-MAXQ Algorithm

Procedure for each time step

1. Update R-MAX primitive models
2. Compute MAXQ composite models
3. Resume executing hierarchical policy

- Propagates optimism up hierarchy
- Memoizes models across time steps
- Employs prioritized sweeping

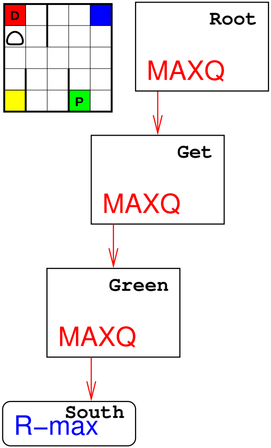


The R-MAXQ Algorithm

Procedure for each time step

1. Update R-MAX primitive models
2. Compute MAXQ composite models
3. Resume executing hierarchical policy

- Propagates optimism up hierarchy
- Memoizes models across time steps
- Employs prioritized sweeping



Outline

1. Learning with Hierarchies of Models
   - Learning in Structured Environments
   - MAXQ Decomposition
2. The R-MAXQ Algorithm
   - R-MAX Exploration
   - Results

Experimental Setup

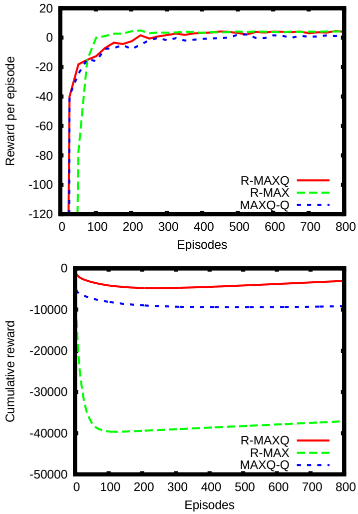
- Environment: stochastic Taxi
- MAXQ-Q
  - Replication of Dietterich's original algorithm
  - Boltzmann exploration
  - Parameters from Dietterich's implementation
- R-MAX primitive models
  - Each state-action optimistic until sample size $m = 5$
  - Planning with value iteration until $\epsilon = 0.001$
- State abstraction:

MAXQ-Q: All of Dietterich's abstractions

R-MAX: *Max-node irrelevance* for each primitive model
Example:
`South` ignores `Passenger` and `Destination`

R-MAXQ: Also *max-node irrelevance* for abstract models
Example: `Get` ignores `Destination`

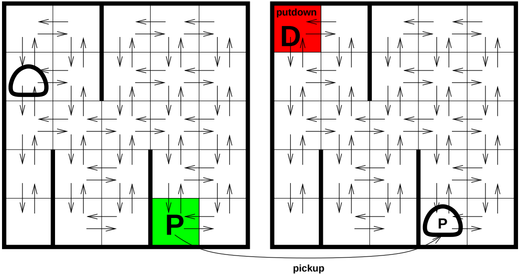
Empirical Results



- R-MAXQ learning curve dominates MAXQ-Q curve
- R-MAXQ converges to same asymptote as R-MAX
- R-MAXQ avoids most of the costly exploration of R-MAX

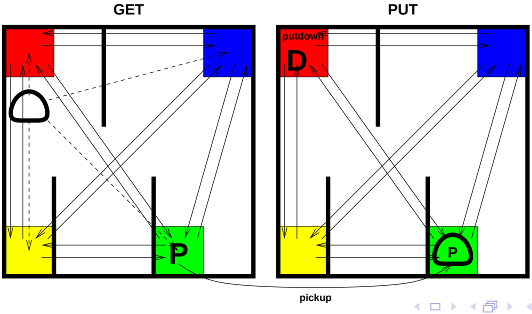
Eager Exploration Versus Lazy Exploration

- R-MAX experiments with `pickup` and `putdown` at all 50 states reachable from the initial state.
- R-MAXQ attempts `pickup` (`putdown`) at only 5 (4) reachable states in `Get` (`Put`).
- R-MAXQ never attempts `putdown` outside the four landmark locations.



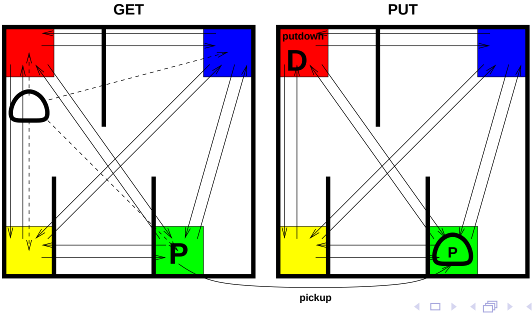
Eager Exploration Versus Lazy Exploration

- R-MAX experiments with `pickup` and `putdown` at all 50 states reachable from the initial state.
- R-MAXQ attempts `pickup` (`putdown`) at only 5 (4) reachable states in `Get` (`Put`).
- R-MAXQ never attempts `putdown` outside the four landmark locations.



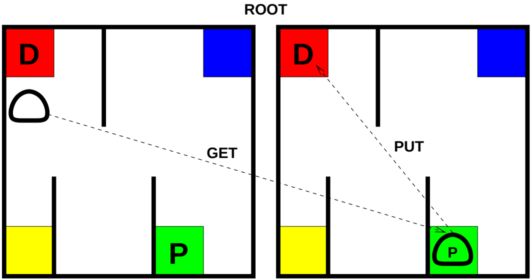
Eager Exploration Versus Lazy Exploration

- R-MAX experiments with `pickup` and `putdown` at all 50 states reachable from the initial state.
- R-MAXQ attempts `pickup` (`putdown`) at only 5 (4) reachable states in `Get` (`Put`).
- R-MAXQ never attempts `putdown` outside the four landmark locations.



The Role of Hierarchy

- Improve computational complexity (already known)
- Decompose tasks into smaller subtasks
  - Fewer primitive actions per subtask
  - Explicit state abstraction at lower levels
  - Smaller "completion sets" of reachable states at higher levels (related to *result distribution irrelevance*)



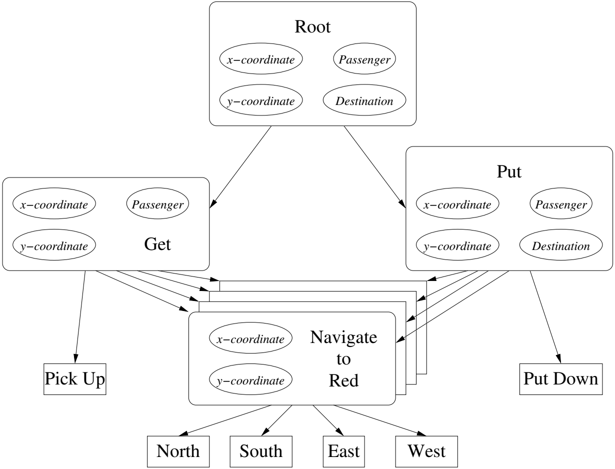
Summary

- R-MAXQ combines R-MAX's robust exploration with MAXQ's incorporation of hierarchical domain knowledge.
- With regard to sample complexity, a primary role of hierarchy may be to constrain unnecessary exploration.

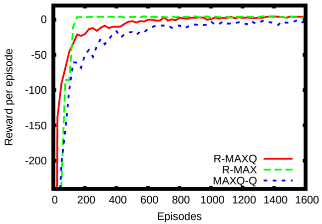
Future Work

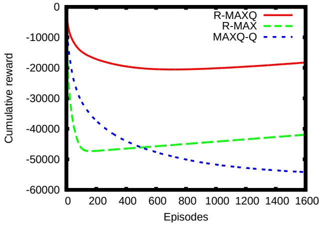
- Application to larger, even continuous, domains
- Guidelines for the design or discovery of hierarchies

The Abstract Taxi Hierarchy



Empirical Results Without State Abstraction





- R-MAX performs slightly worse.
  - Navigational actions require 16 times as much data, since they no longer ignore passenger location and destination.
  - `Pickup` requires 4 times as much data, since it no longer ignores passenger destination.
- R-MAXQ still benefits from never executing `putdown` outside of the four landmark locations.
- MAXQ-Q performs poorly without state abstraction.

The Sample Complexity of R-MAXQ I

- For the same threshold amount of experience per state-action, R-MAXQ will spend no more time exploring that R-MAX.
- However, the threshold required to ensure a given level of near-optimality may be exponentionally worse in the height of the hierarchy.
- These (weak) guarantees make no assumptions about the quality of the hierarchy! (In the same way that the R-MAX guarantees make no assumptions about the policy used to transform a bound on model error into a bound on value function error.)

The Sample Complexity of R-MAXQ II

Theorem

If m samples of each state-action guarantee that R-MAX converges to an ϵ-optimal policy with probability $1-\delta$, then $m' = O\left(m\left(\frac{TL}{1-\delta}\right)^{2h}\right)$ samples of each primitive state-action suffice for R-MAXQ to converge to a recursively ϵ-optimal policy with probability $1-\delta$.

- L is $O\left(\frac{\log \epsilon}{1-\gamma}\right)$
- T is the maximum number of reachable terminal states for any composite action
- h is the height of the hierarchy