Conditional Abstraction Trees for Sample-Efficient Reinforcement Learning

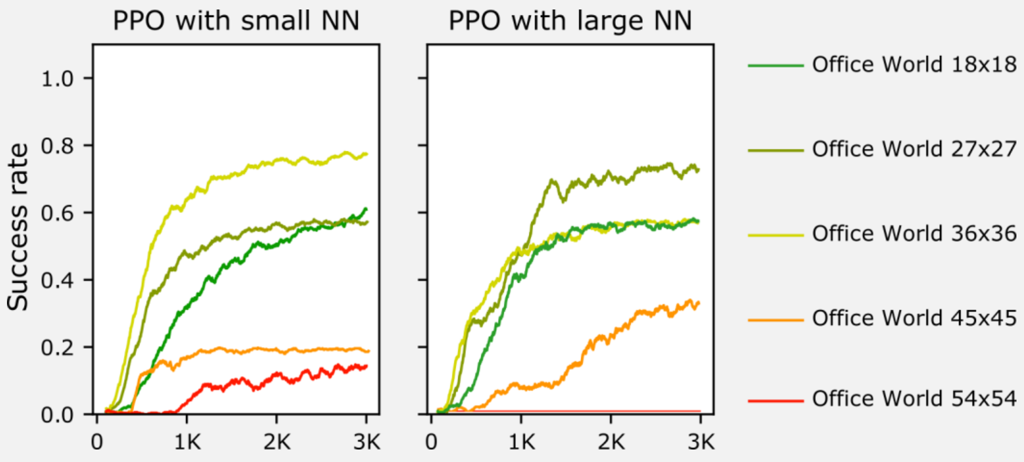
MEHDI DADVAR, RASHMEET KAUR NAYYAR, AND SIDDHARTH SRIVASTAVA

39th Conference on Uncertainty in Artificial Intelligence
Pittsburgh, PA, USA
31st July – 4th August 2023
uai2023

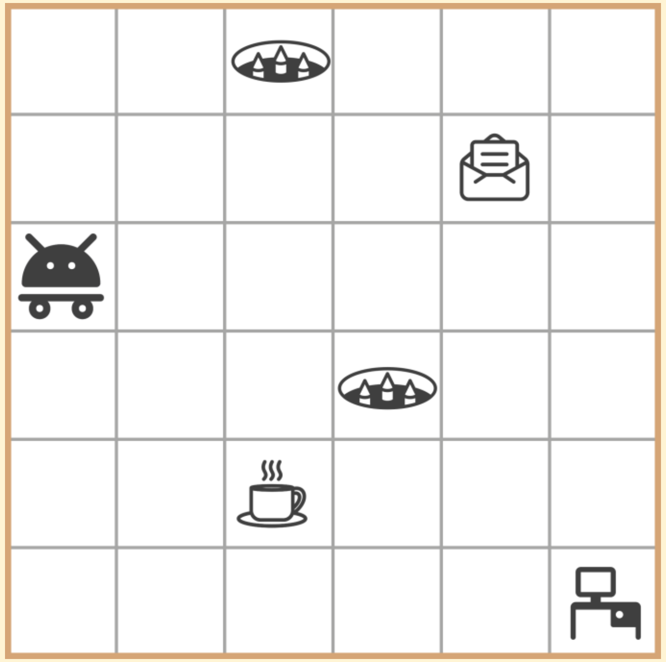
AAIR
Autonomous Agents and Intelligent Robots
ASU Arizona State University

RL Algorithms Are Not Scalable in Sparse Reward Settings

Solution?
A ***good abstraction*** can improve the scalability of RL.



Example: Office World



State Abstraction Can Make RL Algorithms More Scalable

- **Abstraction** maps the original problem representation to a new reduced representation.
- Let $M = \langle S, A, T, R, \gamma \rangle$ be a ground MDP from which an abstract MDP $M = \langle \bar{S}, A, \bar{T}, \bar{R}, \gamma \rangle$ can be derived.
- $\phi: S \rightarrow \bar{S}$ maps a concrete state s to an abstract state s.t. $\bar{s} = \phi(s)$.
- $\bar{T}$ and $\bar{R}$ are defined as follows:

$$\bar{\mathcal{R}}(\bar{s}, a) = \sum_{s \in \phi^{-1}(\bar{s})} w(s) \mathcal{R}(s, a),$$

$$\bar{\mathcal{T}}(\bar{s}, a, \bar{s}') = \sum_{s \in \phi^{-1}(\bar{s})} \sum_{s' \in \phi^{-1}(\bar{s})} w(s) \mathcal{T}(s, a, s').$$

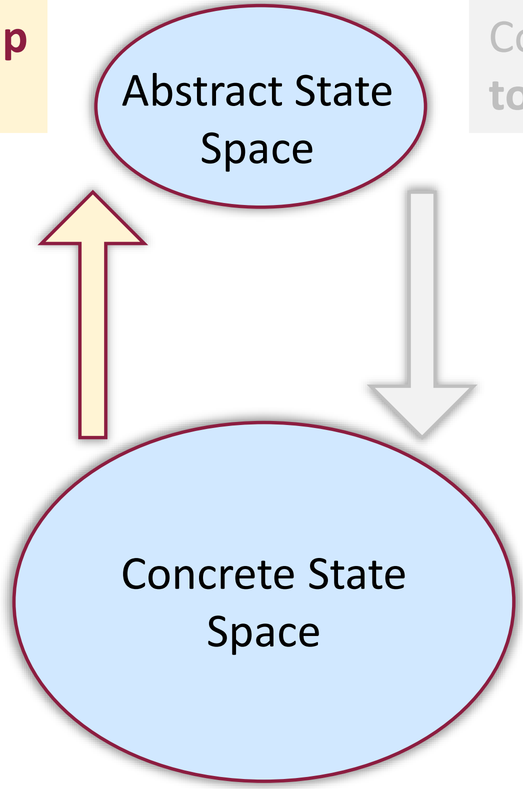
Why Abstraction Refinement?

Constructing state abstraction using **bottom-up** approaches may suffer from scalability issues.

Offline methods [Dietterich, 1999, Jons- son and Barto, 2000, Givan et al., 2003].

Graph-theoretic state abstraction methods [Mannor et al., 2004, Chiu and Soo, 2010].

Abstraction based on Monte-Carlo tree search [Kocsis and Szepesvári, 2006, Jiang et al., 2014].



Constructing state abstraction efficiently using **top-down** requires an abstract RL routine.

Abstraction refinement for classical planning [Seipp and Helmert, 2018].

Categorization of concrete transitions [Uther and Veloso 1998].

Abstraction refinement using a deterministic model of the world [Whiteson, 2010].

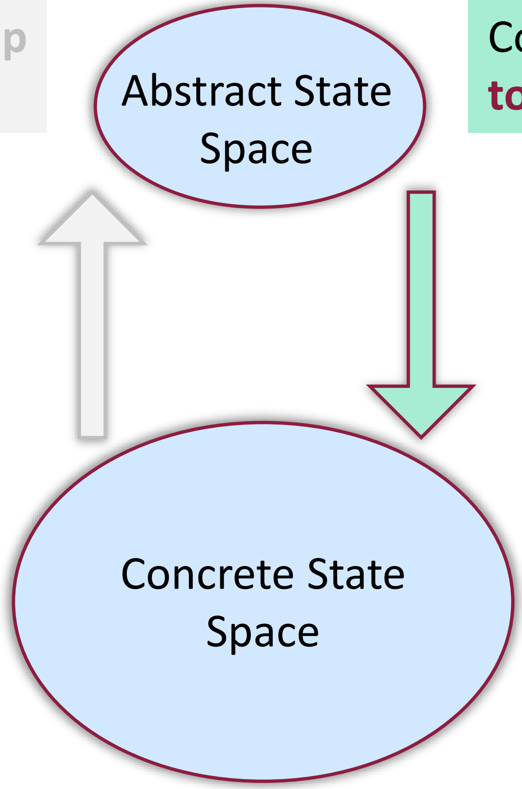
Why Abstraction Refinement?

Constructing state abstraction using **bottom-up** approaches may suffer from scalability issues.

Offline methods [Dietterich, 1999, Jons- son and Barto, 2000, Givan et al., 2003].

Graph-theoretic state abstraction methods [Mannor et al., 2004, Chiu and Soo, 2010].

Abstraction based on Monte-Carlo tree search [Kocsis and Szepesvári, 2006, Jiang et al., 2014].



Constructing state abstraction efficiently using **top-down** requires an abstract RL routine.

Abstraction refinement for classical planning [Seipp and Helmert, 2018].

Categorization of concrete transitions [Uther and Veloso 1998].

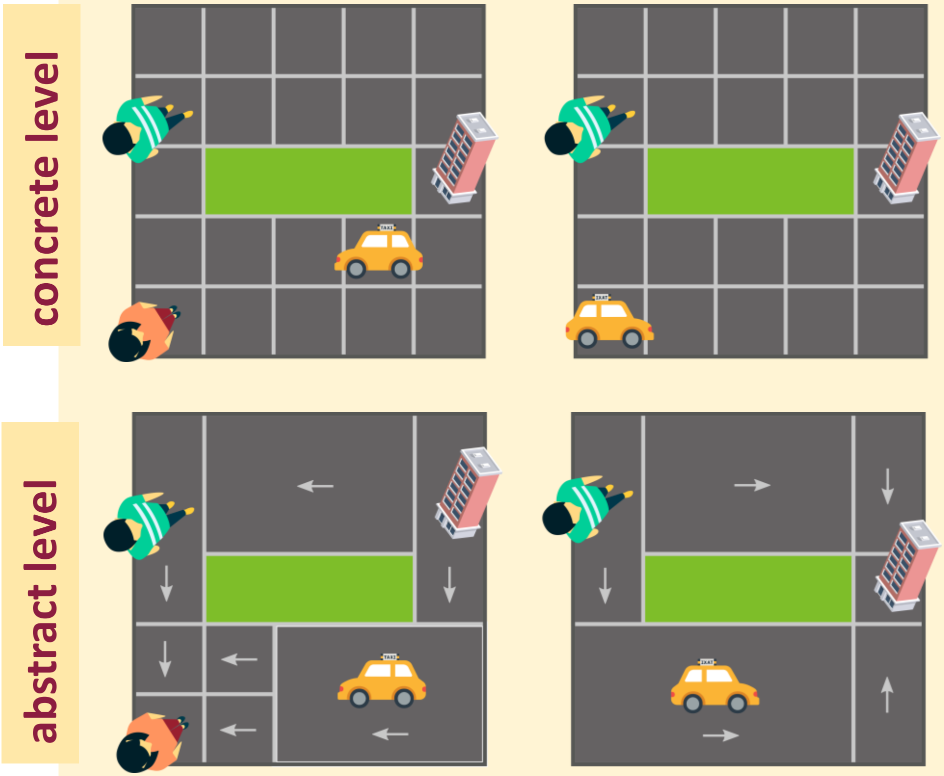
Abstraction refinement using a deterministic model of the world [Whiteson, 2010].

What Is a Good Abstraction?

Intuitively, a **good abstraction** should capture more detail on the more salient parts of the state space.

The abstraction on a state variable should be **contingent** on the value of other state variables.

Example: Taxi World



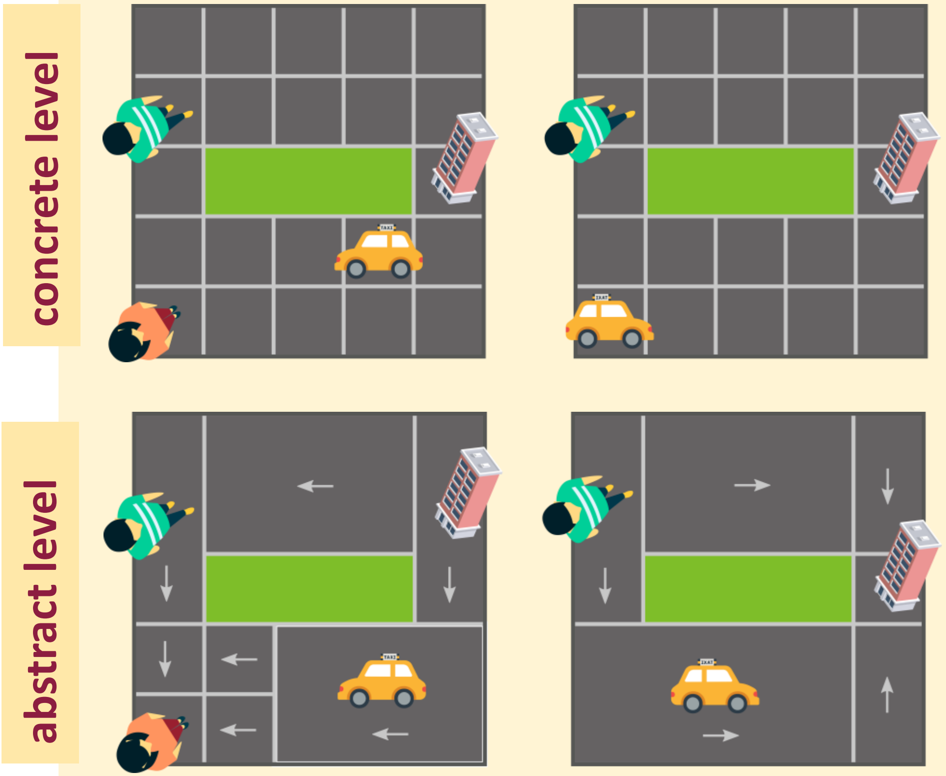
What Is a Good Abstraction?

Intuitively, a **good abstraction** should capture more detail on the more salient parts of the state space.

The abstraction on a state variable should be **contingent** on the value of other state variables.

Our method learns those abstractions in the form of **conditional abstraction trees (CATs)** while doing RL.

Example: Taxi World



CAT+RL Constructs Conditional Abstractions

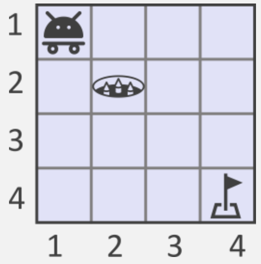
- CAT+RL deals with ranges of state variables.
- Partitioning these ranges constructs abstractions.

x and y represent the location of the agent:

- Range of x: [1,4]
- Range of y: [1,4]

Example: Wumpus World

- The pitfall and goal are terminal states.
- The agent can move to cardinal adjacent cells.



CAT+RL Constructs Conditional Abstractions

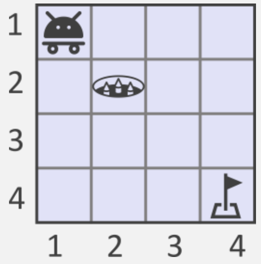
- CAT+RL deals with ranges of state variables.
- Partitioning these ranges constructs abstractions.

x and y represent the location of the agent:

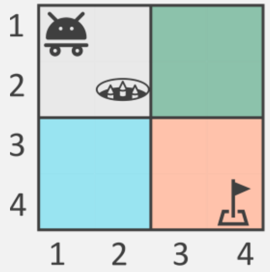
- Range of x: [1,2], [3,4]
- Range of y: [1,2], [3,4]

Example: Wumpus World

- The pitfall and goal are terminal states.
- The agent can move to cardinal adjacent cells.



Coarse Abstraction



CAT+RL Constructs Conditional Abstractions

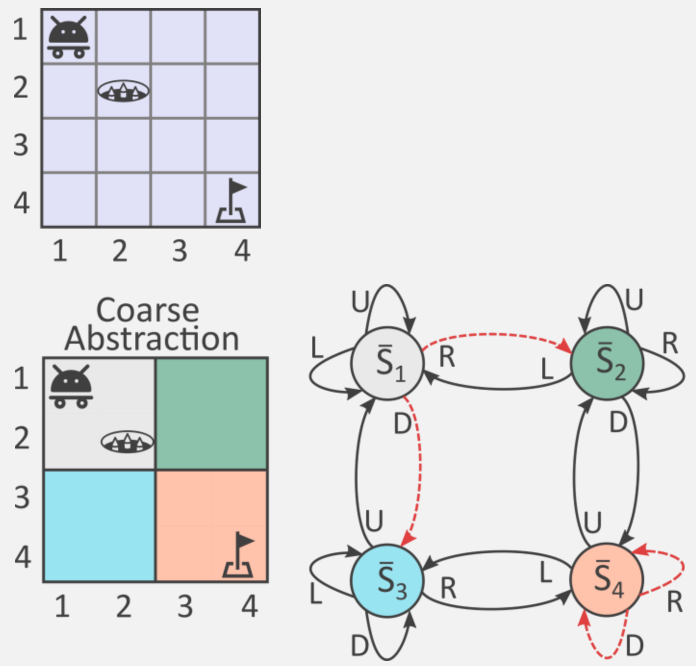
- The refinement of conditional abstractions is guided by the **dispersion of TD errors**.

State-action pairs with high variation of TD error:

$(\bar{S}1, right)$, $(\bar{S}1, down)$, $(\bar{S}4, right)$, and $(\bar{S}4, down)$

Example: Wumpus World

- The pitfall and goal are terminal states.
- The agent can move to cardinal adjacent cells.



CAT+RL Constructs Conditional Abstractions

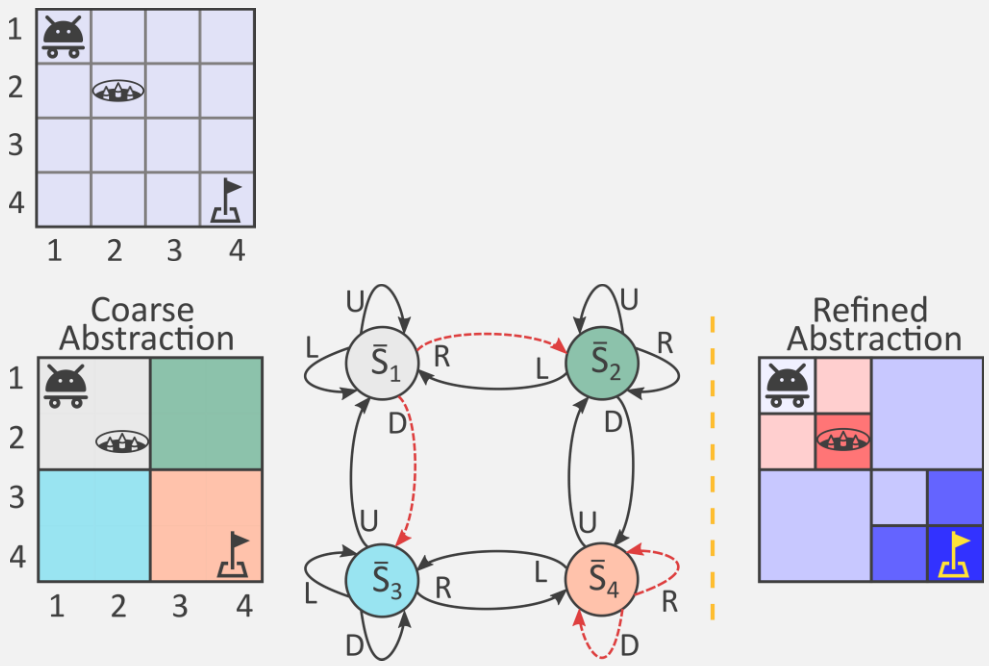
- The refinement of conditional abstractions is guided by the **dispersion of TD errors**.

when y > 2, the domain of x is abstracted into sets {1, 2}, {3}, and {4}.

when y ≤ 2, the domain of x is abstracted into sets {1}, {2}, and {3, 4}.

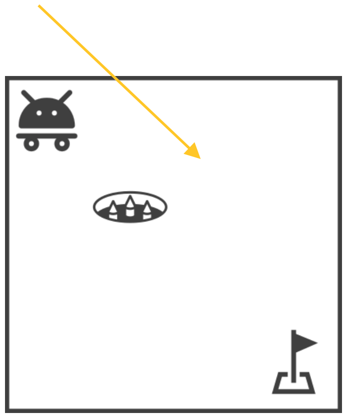
Example: Wumpus World

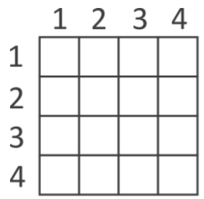
- The pitfall and goal are terminal states.
- The agent can move to cardinal adjacent cells.



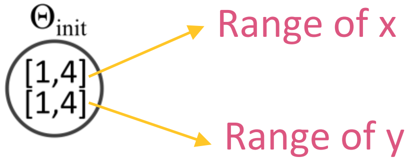
Conditional Abstraction Trees (CATs)

High variation of
TD error





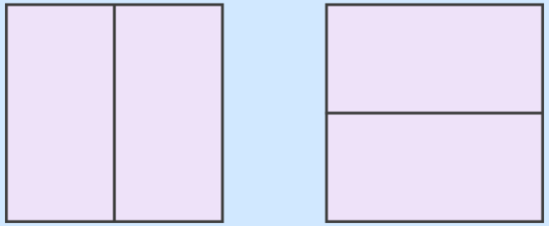
Root of CAT



Finding contributing state variable:

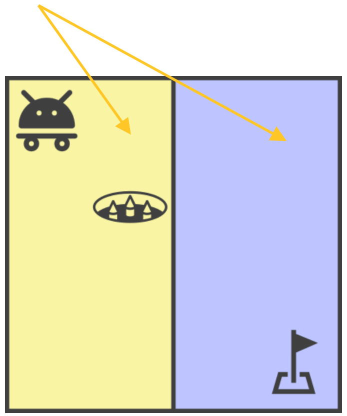
refine on x

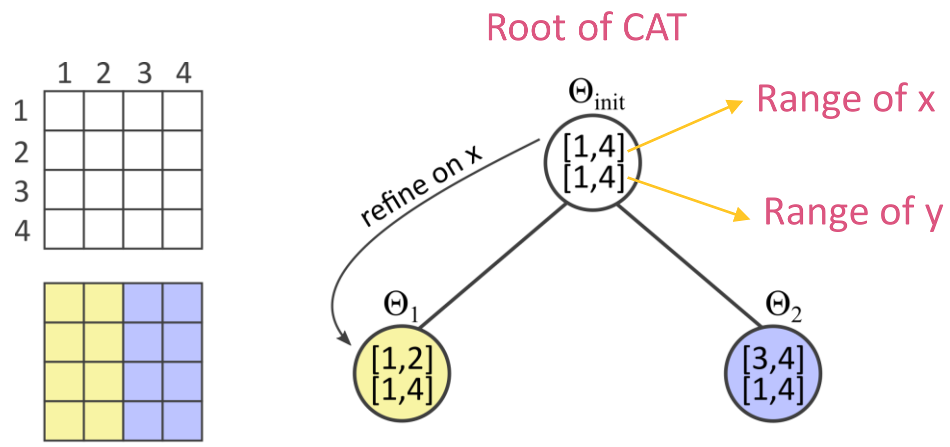
refine on y



Conditional Abstraction Trees (CATs)

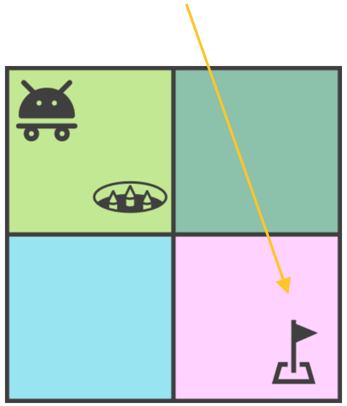
High variation of TD error

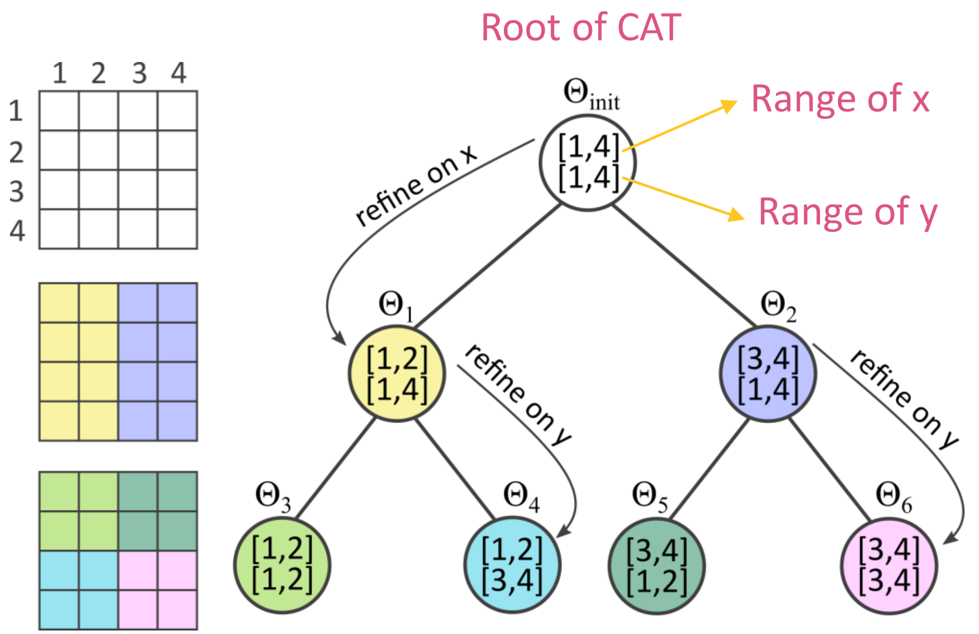




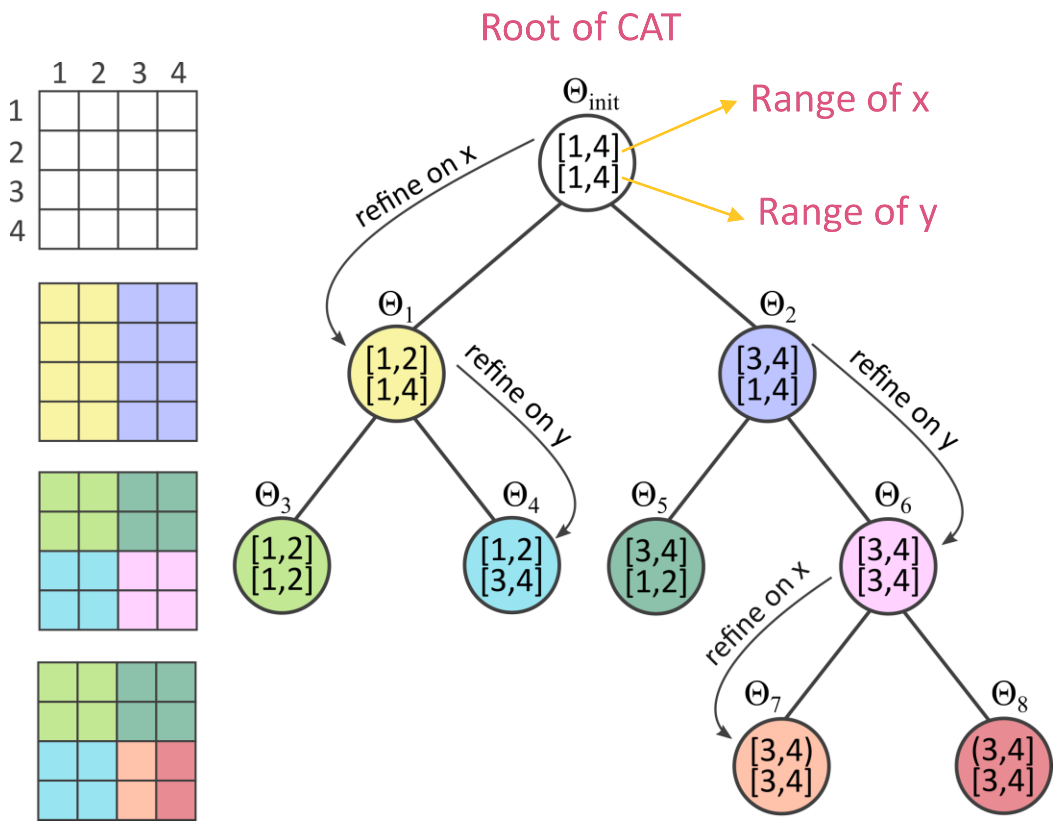
Conditional Abstraction Trees (CATs)

High variation of
TD error





Conditional Abstraction Trees (CATs)



Learning a CAT is Synthesized with RL

Algorithm 2: Learning Dynamic Abstractions

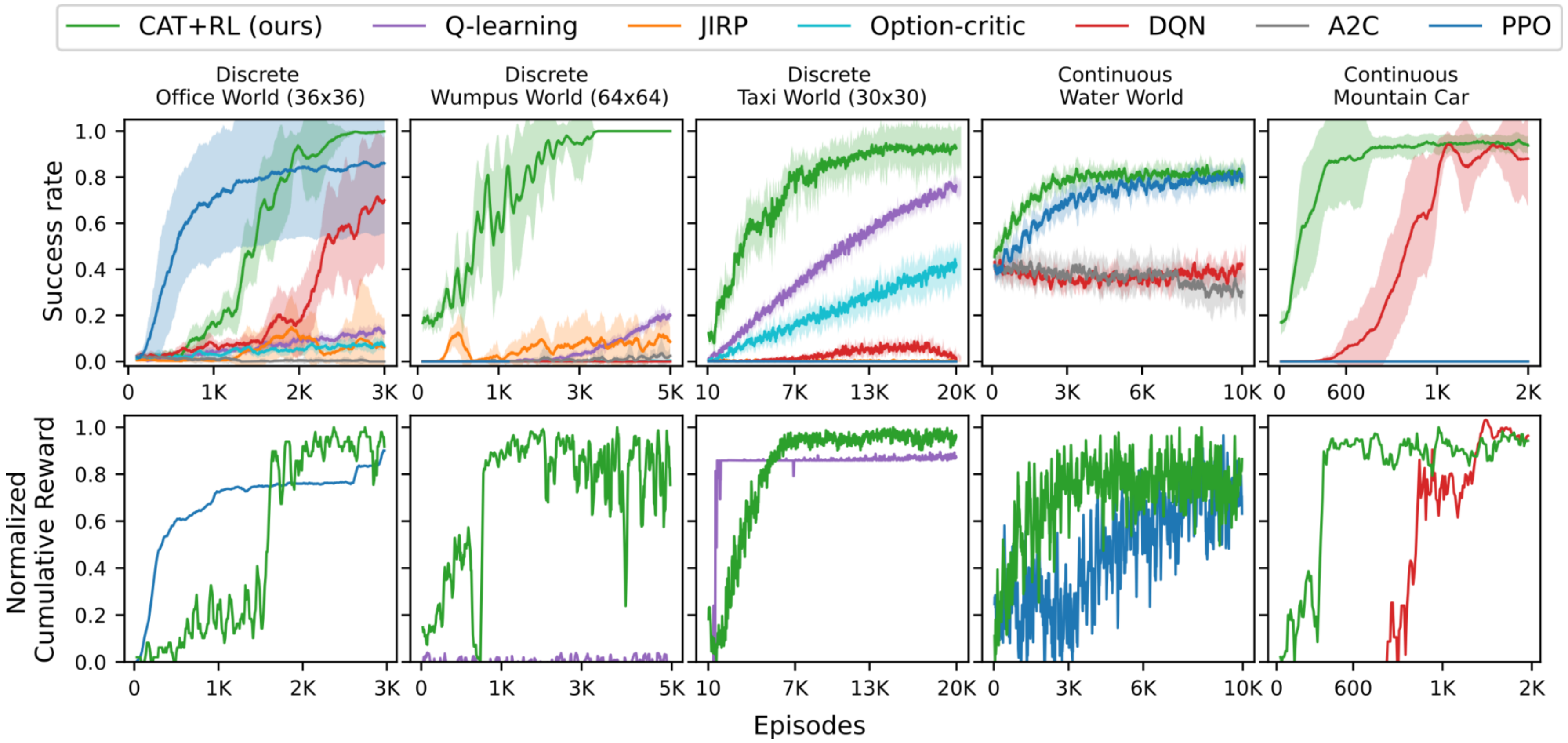
Input: M, f
Output: $\bar{M}, \xi, \bar{\pi}$
initialize Θ_{init}, ξ, and $\bar{Q}$
for $episode = 1, n_{epi}$ **do**
  $s \leftarrow \texttt{reset()}$
  for $steps$ in $episode$ **do**
    $\bar{s} \leftarrow \texttt{FindAbstract}(\xi, \Theta_{init}, s)$
    $a \leftarrow \bar{\pi}(\bar{s})$
    $s', \bar{r}, done \leftarrow \texttt{step}(\texttt{extend}(a))$
    $\bar{s}' \leftarrow \texttt{FindAbstract}(\xi, \Theta_{init}, s')$
    $\bar{\pi} \leftarrow \texttt{train}^{\bar{\pi}}(\bar{s}, \bar{s}', a, \bar{r})$
    $s, \bar{s} \leftarrow s', \bar{s}'$
  if $\bar{M}$ needs refinement **then**
    $\Gamma \leftarrow \texttt{evaluate}(M, \xi, \bar{\pi}, n_{eval})$
    $unstable \leftarrow \texttt{UnstableStates}(\Gamma)$
    for each Θ in $unstable$ **do**
      $i \leftarrow \texttt{UnstableVar}(\Gamma, \Theta)$
      $nodes \leftarrow \texttt{refine}(\Theta, i, f)$
      $\xi \leftarrow \texttt{UpdateTree}(\xi, \Theta, nodes)$
return $\bar{M}, \xi, \bar{\pi}$

Learning Phase:
- Starting with an initial coarse abstraction, the RL agent interacts with the environment and **learns the abstract policy $\bar{\pi}$**.

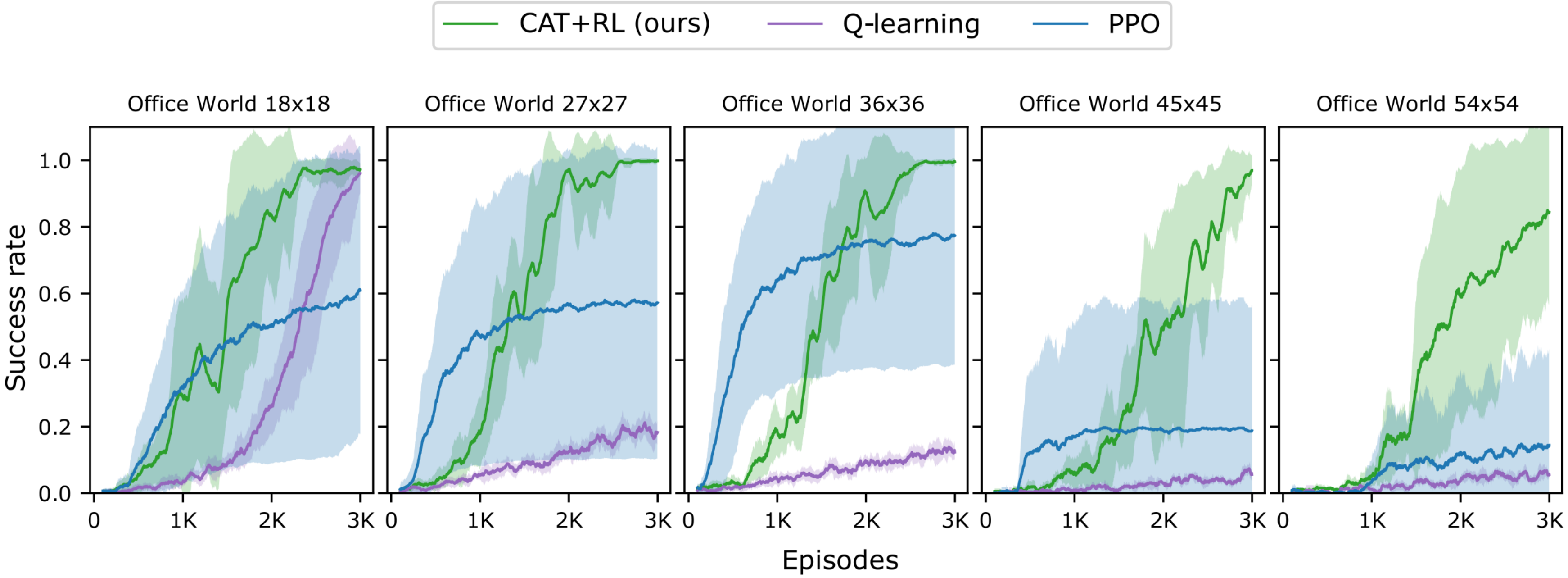
Evaluation Phase
- CAT+RL initiates this phase if the success rate is below a threshold.
- In this phase, **CAT+RL identifies unstable abstract states**.

Refinement Phase
- Given the logs of the TD errors, CAT+RL finds the unstable states on which **the CAT will be refined** with respect to the contributing state variables.

Empirical Results Show Improved Sample Efficiency



Empirical Results Show Improved Scalability



Empirical Results Show Improved Runtime

Total time taken (mean and standard deviation) by CAT+RL, Q-learning, and PPO to solve Office World problems with increasing complexity.

Office World problem size	Time (s) $\pm$ std dev by CAT+RL	Time (s) $\pm$ std dev by Q-learning	Time (s) $\pm$ std dev by PPO
18x18	302.75 ± 29.0	97.69 ± 4.7	2843.42 ± 959.17
27x27	391.36 ± 28.5	441.8 ± 13.85	4956.8 ± 2458.74
36x36	535.71 ± 54.6	1174.84 ± 46.23	8428.24 ± 2867.52
45x45	416.41 ± 58.94	1322.26 ± 45.87	11463.28 ± 3309.09
54x54	1010.52 ± 219.98	7750.53 ± 308.79	15293.57 ± 5815.63

Conclusions

- CAT+RL learns **conditional abstraction trees** on-the-fly while doing purely abstract RL.
- CAT+RL enables vanilla Q-learning to outperform SOTA baselines by significantly **improving its sample efficiency.**
- CAT+RL learns well-defined abstract representations and **draws out similarities** across the state space.
- CAT+RL requires significantly **less hyperparameter tuning** in comparison to many of the baselines.

Conclusions

- CAT+RL learns **conditional abstraction trees** on-the-fly while doing purely abstract RL.
- CAT+RL enables vanilla Q-learning to outperform SOTA baselines by significantly **improving its sample efficiency**.
- CAT+RL learns well-defined abstract representations and **draws out similarities** across the state space.
- CAT+RL requires significantly **less hyperparameter tuning** in comparison to many of the baselines.

Conclusions

- CAT+RL learns **conditional abstraction trees** on-the-fly while doing purely abstract RL.
- CAT+RL enables vanilla Q-learning to outperform SOTA baselines by significantly **improving its sample efficiency.**
- CAT+RL learns well-defined abstract representations and **draws out similarities** across the state space.
- CAT+RL requires significantly **less hyperparameter tuning** in comparison to many of the baselines.

Conclusions

- CAT+RL learns **conditional abstraction trees** on-the-fly while doing purely abstract RL.
- CAT+RL enables vanilla Q-learning to outperform SOTA baselines by significantly **improving its sample efficiency.**
- CAT+RL learns well-defined abstract representations and **draws out similarities** across the state space.
- CAT+RL requires significantly less hyperparameter tuning in comparison to many of the baselines.

Conclusions

- CAT+RL learns **conditional abstraction trees** on-the-fly while doing purely abstract RL.
- CAT+RL enables vanilla Q-learning to outperform SOTA baselines by significantly **improving its sample efficiency**.
- CAT+RL learns well-defined abstract representations and **draws out similarities** across the state space.
- CAT+RL requires significantly **less hyperparameter tuning** in comparison to many of the baselines.

mdadvar@asu.edu

www.mdadvar.net

References

Thomas G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. Journal of artificial intelligence research, 13:227–303, 2000.

Anders Jonsson and Andrew Barto. Automated state abstraction for options using the u-tree algorithm. Advances in neural information processing systems, 13, 2000.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. Artificial Intelligence, 147(1-2):163–223, 2003.

Shie Mannor, Ishai Menache, Amit Hoze, and Uri Klein. Dynamic abstraction in reinforcement learning via clustering. In Proceedings of the twenty-first international conference on Machine learning, page 71, 2004.

Chung-Cheng Chiu and Von-Wun Soo. Automatic complexity reduction in reinforcement learning. Computational Intelligence, 26(1):1–25, 2010.

Levente Kocsis and Csaba Szepesvári. Bandit based montecarlo planning. In European conference on machine learning, pages 282–293. Springer, 2006.

Nan Jiang, Satinder Singh, and Richard Lewis. Improving uct planning via approximate homomorphisms. In Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems, pages 1289– 1296, 2014.

Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In International Conference on Computer Aided Verification, pages 154–169. Springer, 2000.

Rohit Chadha and Mahesh Viswanathan. A counterexampleguided abstraction-refinement framework for markov decision processes. ACM Transactions on Computational Logic (TOCL), 12(1):1–49, 2010.