Online and Reinforcement Learning (2025)
Home Assignment 8

Davide Marchi 777881

Contents

1 MDP Classes

In this exercise we classify three MDPs—**RiverSwim**, **RiverSwim-2** (a modified version of RiverSwim), and the **4-room grid-world**—according to the following classes:

- **Ergodic MDPs**: Every stationary policy induces a Markov chain that has a unique recurrent class (i.e., it is a unichain).
- **Communicating MDPs**: For every pair of states $s, s' \in \mathcal{S}$, there exists some policy under which s' is reachable from s (and vice versa).
- **Weakly Communicating MDPs**: The state space can be partitioned into a communicating set of recurrent states and a set of transient states.

Since an ergodic MDP is a special case of a communicating MDP, and every communicating MDP is weakly communicating, the classes satisfy

$$\text{Ergodic} \subseteq \text{Communicating} \subseteq \text{Weakly Communicating}.$$

Below, I analyze each MDP and provide counterexamples where needed.

(i) RiverSwim

Ergodicity: Answer: `False`.

Explanation: Although the underlying transition graph of RiverSwim is (in principle) strongly connected (since for any two states one can design a policy to move from one to the other), there exist stationary policies that break the unichain property. For example, consider the policy that always selects the *left* action. Under this policy the agent remains confined to the left-hand bank (state 1) and never reaches any other state. This shows that not every stationary policy yields a single recurrent class; hence, the MDP is not ergodic.

Communicating: Answer: `True`.

Explanation: By the structural properties of RiverSwim, for any pair of states s and s' one can construct a policy that uses the *right* action (to move toward higher-indexed states) and the *left* action (to move back) appropriately such that s' is reachable from s. This mutual reachability implies that RiverSwim is a communicating MDP.

Weakly Communicating: Answer: `True`.

Explanation: Since being communicating implies that there exists at least one policy connecting every pair of states, the MDP is, in particular, weakly communicating.

(ii) RiverSwim-2

The modified MDP, RiverSwim-2, is defined as RiverSwim but with an extra state s_{extra} that has two actions:

- Under action a_1, the agent moves deterministically to state 1 (the left bank).
- Under action a_2, the agent moves to state 1 with probability 0.5 or to state 2 with probability 0.5.

All other transitions remain the same as in RiverSwim.

Ergodicity: Answer: `False`.

Explanation: Similar to the original RiverSwim, one can choose stationary policies (e.g., always taking an action that avoids exiting a subset of states) that do not visit all states. Thus, the unichain property does not hold.

Communicating: Answer: `False`.

Explanation: A key modification in RiverSwim-2 is that the extra state s_{extra} is not reachable from any of the original RiverSwim states. While once in s_{extra} the agent can move to state 1 (or 2), there is no action in the main chain that will take the agent to s_{extra}. Hence, there exist state pairs (namely, any state in the original set and s_{extra}) for which no policy can ensure mutual reachability. Therefore, RiverSwim-2 is not communicating.

Weakly Communicating: Answer: `True`.

Explanation: Even though the whole state space is not mutually reachable, the original states (the main chain) form a communicating recurrent class. The extra state s_{extra} is transient because whenever it is visited the agent will eventually be forced into the communicating set. This structure fits the definition of a weakly communicating MDP.

(iii) 4-room Grid-world

The 4-room grid-world (often also called the frozen lake MDP in this context) has the following characteristics:

- The agent has 4 actions (up, down, left, right) when not adjacent to walls.
- Due to a slippery floor, the chosen action results in moving in the intended direction with probability 0.7, staying in the same state with probability 0.1, or moving in one of the two perpendicular directions (each with probability 0.1).
- Walls act as reflectors (i.e., if an action leads into a wall, the agent remains in the current state).

- Upon reaching the rewarding state (highlighted in yellow), the agent is teleported back to the initial state.

Ergodicity: Answer: `False`.

Explanation: Even though the grid is fully connected, there exist stationary policies (for instance, those that consistently choose actions that lead to self-loops—by always hitting the wall) that confine the agent to a strict subset of states. Thus, not every stationary policy produces a Markov chain with a single recurrent class.

Communicating: Answer: `True`.

Explanation: The underlying structure of the grid is such that for any two states there exists a policy (using the stochastic nature of the transitions and the teleportation mechanism) that connects them. In other words, even if some policies lead to degenerate behavior, one can always design a policy that makes every state reachable from any other state.

Weakly Communicating: Answer: `True`.

Explanation: Since the grid-world is communicating (in the structural sense described above), it is trivially weakly communicating.

Summary Table

MDP	Ergodic	Communicating	Weakly Communicating
RiverSwim	False	True	True
RiverSwim-2	False	False	True
4-room grid-world	False	True	True

Table 1: Classification of the MDPs based on their connectivity properties.

2 Infinitely Many Choices for Optimal Bias in Average-Reward MDPs

Let $M = (\mathcal{S}, \mathcal{A}, P, r)$ be a finite weakly-communicating average-reward MDP and let

$$b^* : \mathcal{S} \to \mathbb{R}$$

be an optimal bias function for M, i.e., it satisfies the optimality (Bellman) equation

$$g^* + b^*(s) = \max_{a \in \mathcal{A}(s)} \Big\{ r(s,a) + \sum_{s' \in \mathcal{S}} P(s'|s,a)\, b^*(s') \Big\}, \quad \forall s \in \mathcal{S},$$

where g^* is the optimal gain. Now, for any constant $c \in \mathbb{R}$, consider the function

$$b'(s) = b^*(s) + c, \quad \forall s \in \mathcal{S}.$$

We need to show that b' is also an optimal bias function for M.

Solution

Since b^* is an optimal bias function, we have

$$g^* + b^*(s) = \max_{a \in \mathcal{A}(s)} \left\{ r(s,a) + \sum_{s' \in \mathcal{S}} P(s'|s,a)\, b^*(s') \right\} \quad \forall s \in \mathcal{S}. \tag{1}$$

Define the function

$$b'(s) = b^*(s) + c.$$

Then, for any state $s \in \mathcal{S}$, the left-hand side of the Bellman equation becomes:

$$g^* + b'(s) = g^* + b^*(s) + c.$$

Next, consider the right-hand side when replacing $b^*(s')$ with $b'(s')$:

$$r(s,a) + \sum_{s' \in \mathcal{S}} P(s'|s,a)\, b'(s') = r(s,a) + \sum_{s' \in \mathcal{S}} P(s'|s,a) \left(b^*(s') + c \right).$$

Since c is a constant, we can rewrite the sum as:

$$r(s,a) + \sum_{s' \in \mathcal{S}} P(s'|s,a)\, b^*(s') + c \sum_{s' \in \mathcal{S}} P(s'|s,a).$$

Because $P(\cdot|s,a)$ is a probability distribution (i.e., $\sum_{s'} P(s'|s,a) = 1$), this simplifies to:

$$r(s,a) + \sum_{s' \in \mathcal{S}} P(s'|s,a)\, b^*(s') + c.$$

Now, taking the maximum over all actions $a \in \mathcal{A}(s)$, we have:

$$\max_{a \in \mathcal{A}(s)} \left\{ r(s,a) + \sum_{s' \in \mathcal{S}} P(s'|s,a)\, b'(s') \right\} = \max_{a \in \mathcal{A}(s)} \left\{ r(s,a) + \sum_{s' \in \mathcal{S}} P(s'|s,a)\, b^*(s') + c \right\}.$$

Since c is independent of the choice of a, it factors out of the maximization:

$$= \left(\max_{a \in \mathcal{A}(s)} \left\{ r(s,a) + \sum_{s' \in \mathcal{S}} P(s'|s,a)\, b^*(s') \right\} \right) + c.$$

Using (1) we substitute for the term in the parentheses:

$$= g^* + b^*(s) + c.$$

Thus, we obtain

$$g^* + b'(s) = \max_{a \in \mathcal{A}(s)} \left\{ r(s,a) + \sum_{s' \in \mathcal{S}} P(s'|s,a)\, b'(s') \right\}, \quad \forall s \in \mathcal{S}.$$

Conclusion

We have shown that if b^* is an optimal bias function, then for any constant $c \in \mathbb{R}$, the function

$$b'(s) = b^*(s) + c$$

also satisfies the Bellman optimality equation. Hence, b' is an optimal bias function for M. This result confirms that the optimal bias function is determined only up to an additive constant.

3 Comparison between UCRL2 and UCRL2-L

In this exercise, I empirically evaluate the performance of the UCRL2 (Jaksch et al., 2010) and UCRL2-L algorithms on a 6-state RiverSwim MDP. The main differences between the two algorithms lie in the construction of their confidence intervals. For UCRL2, the confidence sets are defined as

$$\left|\hat{R}_t(s,a) - R'\right| \leq \sqrt{\frac{3.5 \log\left(\frac{2SAn}{\delta}\right)}{n}}, \quad \|\hat{P}_t(\cdot|s,a) - P'\|_1 \leq \sqrt{\frac{14S \log\left(\frac{2An}{\delta}\right)}{n}},$$

where $n = \max(1, N_t(s,a))$. In contrast, UCRL2-L uses improved (Laplace-based) confidence intervals.

(i) Implementation Modifications

To implement UCRL2, I modified the provided `UCRL2_L.py` file. The modifications consisted of replacing the confidence set definitions used in UCRL2-L with those above. A code snippet illustrating the essential differences is as follows:

```
self.confR[s, a] = sqrt((3.5 * log((2 * self.nS * self.nA * n) / self.
    delta)) / n)
self.confP[s, a] = sqrt((14 * self.nS * log((2 * self.nA * n) / self.
    delta)) / n)
```

This ensures that UCRL2 uses the appropriate theoretical confidence intervals as in Jaksch et al. (2010).

(ii) Cumulative Regret Evaluation

The experiments were conducted on the 6-state RiverSwim MDP with the following settings:

- Time horizon $T = 3.5 \times 10^5$
- Initial state: left-most state (state 1)

- Failure probabilities: $\delta_{\text{UCRL2}} = 0.05$ and $\delta_{\text{UCRL2-L}} = 0.0125$

The regret is defined as

$$R(T) = \sum_{t=1}^{T} (g^\star - r_t)\,,$$

where $g^\star$ is the optimal gain computed via Value Iteration.

Figure 1 shows the cumulative regret curves for UCRL2 and UCRL2-L, averaged over 50 independent runs.

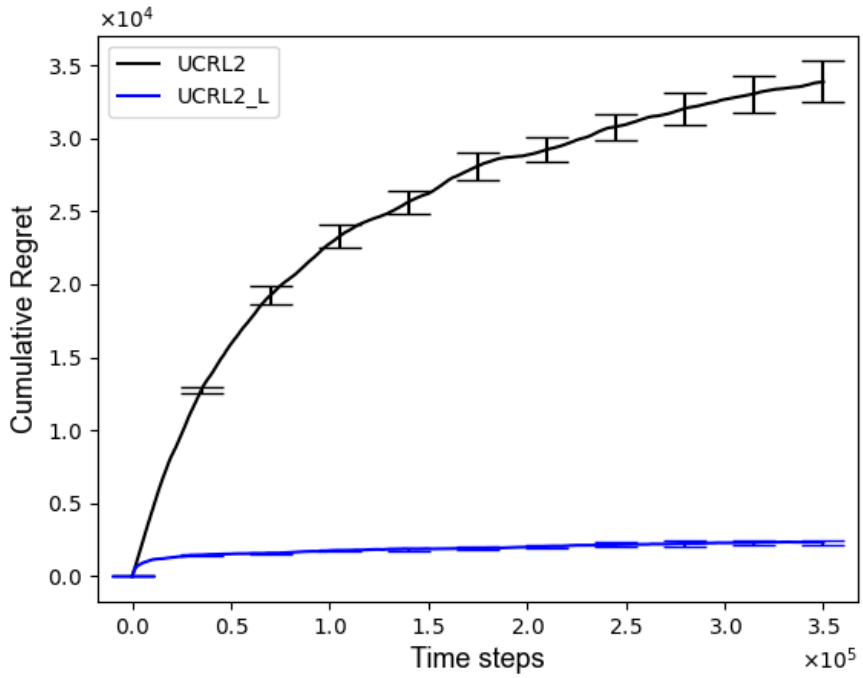

Figure 1: Average Cumulative Regret for UCRL2 and UCRL2-L over 50 runs.

(iii) Empirical Gain Comparison

I also plotted the empirical average gain,

$$\bar{r}(t) = \frac{1}{t} \sum_{t'=1}^{t} r_{t'},$$

for both algorithms. In the plot, I added a horizontal dashed red line corresponding to the optimal gain $g^\star$.

Figure 2 shows the gain curves along with the line for $g^\star$.

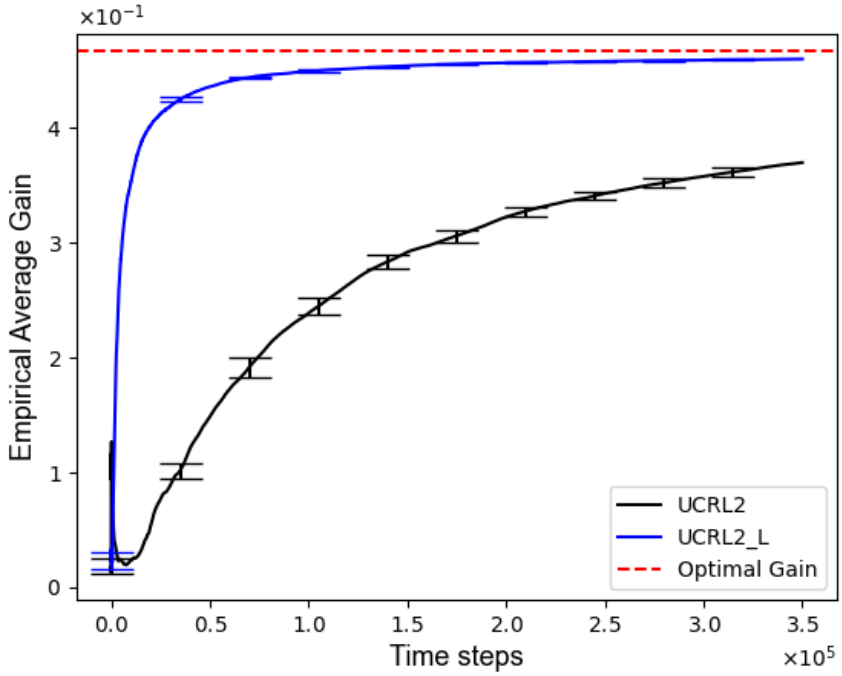

Figure 2: Empirical Average Gain for UCRL2 and UCRL2-L with the optimal gain $g^\star$ indicated by the horizontal red dashed line.

(iv) Average Number of Episodes

The average number of episodes initiated by the two algorithms across 50 runs is reported below:

- UCRL2: 75.8 episodes
- UCRL2-L: 56.16 episodes

(v) Discussion

The experimental results reveal several important differences:

1. **Cumulative Regret:** The cumulative regret curve for UCRL2-L is lower than that of UCRL2, indicating that UCRL2-L achieves better performance in terms of regret minimization.
2. **Empirical Gain:** Both algorithms approach the optimal gain $g^\star$ as time progresses; however, the gain curve for UCRL2-L remains closer to $g^\star$ compared to that of UCRL2. The horizontal line marking $g^\star$ confirms that UCRL2-L has a smaller gap.

3. **Number of Episodes:** UCRL2-L initiates fewer episodes on average (56.16) than UCRL2 (75.8). A smaller number of episodes is indicative of more efficient exploration and fewer policy restarts.

Overall, the refined confidence sets in UCRL2-L appear to lead to improved empirical performance. They not only reduce the cumulative regret but also result in fewer episodes needed during learning, which suggests that UCRL2-L is more sample efficient when compared to UCRL2.

Conclusion: Based on the empirical evaluation, I conclude that UCRL2-L outperforms UCRL2 in this particular RiverSwim setting. The combination of lower cumulative regret, more stable gain curves, and fewer episodes suggests that the modifications introduced in UCRL2-L provide a practical advantage.

4 Leveraging Prior Knowledge on the Graph of MDP to Reduce Regret

In this exercise, I study a variant of the UCRL2-L algorithm that exploits prior knowledge about the transition graph of the MDP. Specifically, for each state-action pair (s, a), I assume that the support set

$$K_{s,a} \;=\; \{\, x \in \mathcal{S} \;\mid\; P(x \mid s, a) > 0\}$$

is known a priori. By restricting candidate transition models to those assigning zero probability to states outside $K_{s,a}$, I aim to reduce over-exploration of impossible transitions and thereby lower regret.

(i) Modified Confidence Sets and the Implementation of `UCRL2_L_supp`

In the standard UCRL2-L algorithm, the L^1-type confidence set for $P(\cdot \mid s, a)$ involves the entire state space $\mathcal{S}$. Concretely, one typically has

$$\|\hat{P}_t(\cdot \mid s, a) - P'(\cdot \mid s, a)\|_1 \;\;\leq\;\; \sqrt{\frac{2\left(1+\frac{1}{n}\right)\,\ln\!\left(\sqrt{n+1}\,\frac{2^{|\mathcal{S}|}-2}{\delta}\right)}{n}},$$

where $n = \max\{1, N_t(s, a)\}$ is the number of visits to (s, a) up to time t. If I know that $P'(x) = 0$ for all $x \notin K_{s,a}$, then I can replace $|\mathcal{S}|$ by $|K_{s,a}|$ in the logarithmic term, reducing the size of the confidence set. Formally, I define

$$C'_{s,a,\text{supp}} \;=\; \Big\{\, P' \in \Delta(\mathcal{S}) \;:\; P'(x) = 0 \;\text{ for all }\; x \notin K_{s,a},$$
$$\|\hat{P}_t(\cdot \mid s, a) - P'(\cdot \mid s, a)\|_1 \;\leq\; \sqrt{\frac{2\left(1+\frac{1}{n}\right)\,\ln\!\left(\sqrt{n+1}\,\frac{2^{|K_{s,a}|}-2}{\delta}\right)}{n}}\Big\}.$$

This modification is straightforward to implement by forcing the empirical estimate $\hat{P}_t(\cdot \mid s, a)$ to zero outside the known support and adjusting the extended value iteration (EVI) routine so that it only considers states in $K_{s,a}$. Below is a code snippet (in Python-like pseudocode) showing the essential changes. I highlight two key points:

- *Confidence Bound Update*: Use $|K_{s,a}|$ instead of $|\mathcal{S}|$ in the log term.
- *Support Enforcement*: Immediately zero out probabilities for any state not in $K_{s,a}$ after updating the empirical model.

Below is a Python code snippet that shows the main changes implemented for the support-based variant:

```
import numpy as np
from math import sqrt, log

class UCRL2_L_supp(UCRL2_L):
    def __init__(self, nS, nA, gamma, known_support, epsilon=0.01, delta=
    0.05):
        super().__init__(nS, nA, gamma, epsilon, delta)
        self.known_support = known_support  # known_support is a list of
    lists: known_support[s][a] gives allowed states

    def confidence(self):
        d = self.delta / (self.nS * self.nA)
        for s in range(self.nS):
            for a in range(self.nA):
                n = max(1, self.Nk[s, a])
                # Use the size of the known support for state-action (s,a
    )
                k = len(self.known_support[s][a])
                factor = (2 ** k - 2) if (2 ** k - 2) > 0 else 1
                self.confR[s, a] = sqrt(((1 + 1/n) * log(2 * np.sqrt(n +
    1) / d)) / (2 * n))
                self.confP[s, a] = sqrt((2 * (1 + 1/n) * log(np.sqrt(n +
    1) * factor / d)) / n)

    def new_episode(self):
        self.updateN()
        self.vk = np.zeros((self.nS, self.nA))
        for s in range(self.nS):
            for a in range(self.nA):
                div = max(1, self.Nk[s, a])
                self.hatR[s, a] = self.Rsa[s, a] / div
                for next_s in range(self.nS):
                    self.hatP[s, a, next_s] = self.Nsas[s, a, next_s] /
    div
                # Enforce prior knowledge: set estimated probability to 0
     outside K_{s,a}
                for next_s in range(self.nS):
                    if next_s not in self.known_support[s][a]:
```

```
                    self.hatP[s, a, next_s] = 0.0
        self.confidence()
        self.policy = self.EVI_supp()
```

Next, I tested this `UCRL2_L_supp` algorithm on a 6-state ergodic RiverSwim environment. The time horizon was $T = 4 \times 10^5$, $\delta = 0.05$, and I used 40 independent runs. The cumulative regret is computed as

$$R(T) = \sum_{t=1}^{T} (g^* - r_t),$$

where g^* is the optimal gain found via Value Iteration. Figure 3 shows the results for UCRL2-L-supp (mean and 95% confidence intervals).

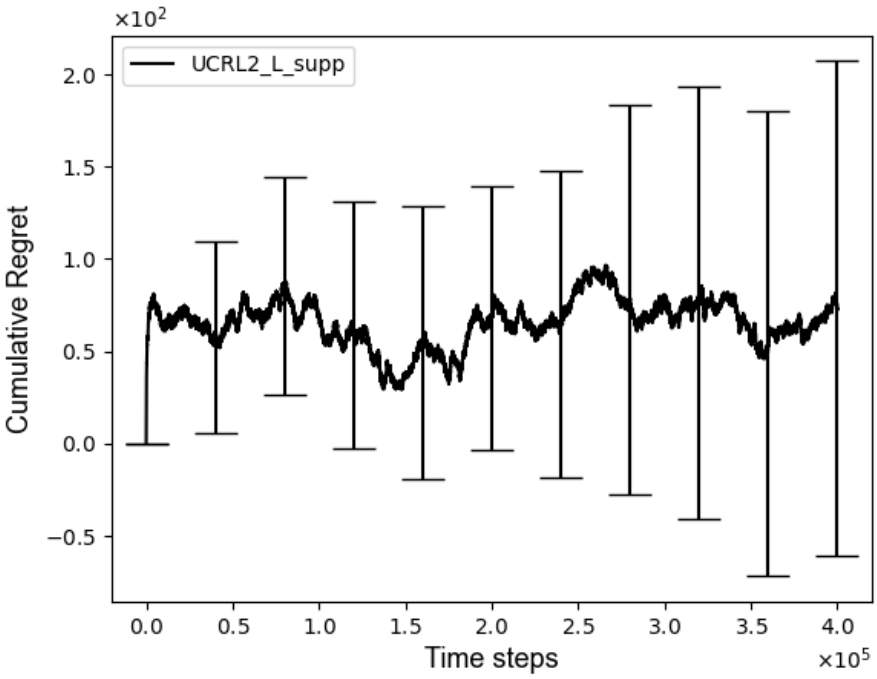

Figure 3: Cumulative regret for UCRL2-L-supp (with support knowledge) over 40 runs. Error bars indicate 95% confidence intervals.

(ii) Comparison with UCRL2-L (Without Prior Knowledge)

I ran the same experiment using the standard UCRL2-L algorithm under identical conditions. In principle, restricting the state space in the confidence sets should reduce uncertainty because the factor $2^{|K_{s,a}|} - 2$ is (typically) much smaller than $2^{|\mathcal{S}|} - 2$. In practice, early in the learning process, certain transitions in the smaller support might be visited so rarely that the resulting bounds can still be large. However, as $N_t(s, a)$ grows, focusing

exploration on the states that genuinely matter eventually yields a more confident model in those transitions.

Figure 4 shows the cumulative regret curve for the original UCRL2-L (again, mean and 95% confidence intervals over 40 runs). Comparing Figures 3 and 4, I observe that UCRL2-L-supp typically achieves lower cumulative regret. This observation is consistent with the idea that excluding impossible transitions makes exploration more efficient, even if small-sample effects initially cause some confidence bounds to appear large.

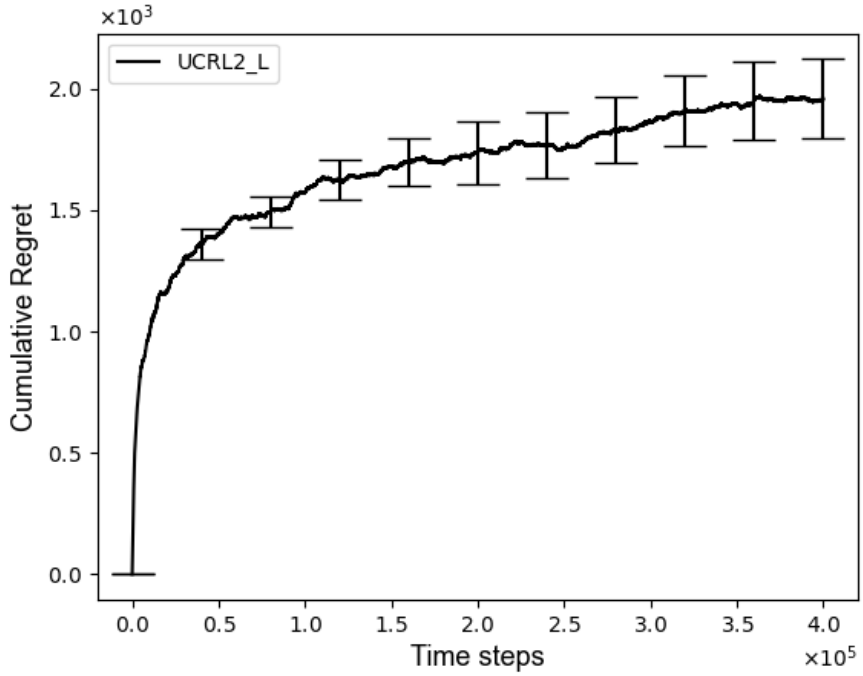

Figure 4: Cumulative regret for UCRL2-L (no prior support knowledge) over 40 runs. Error bars indicate 95% confidence intervals.

Conclusion:
By incorporating prior knowledge of the transition graph, I remove the need to explore impossible transitions, thereby tightening the effective confidence sets once enough visits have occurred in the known support. Although small sample sizes can produce larger variance at the start, the reduced modeling uncertainty ultimately leads to lower cumulative regret compared to the original UCRL2-L. Overall, UCRL2-L-supp leverages the support sets to achieve more focused exploration and improved empirical performance in the 6-state RiverSwim MDP.