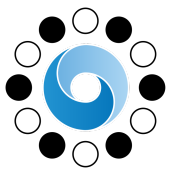
Robust probabilistic target-oriented exploration with reliability approximation

Tokyo Denki University

Moto Shinriki, Yu Kono, Tatsuji Takahashi

Background

Reinforcement learning (RL) agents have reached superhuman levels in games such as Go and chess.



AlphaGo >

Task complexity

In terms of the complexity of the environment,
real-world tasks are more challenging than games.

>

Reinforcement learning and human learning

[Problem] RL is still too costly to use in practice

- Sampling time for optimization is not feasible in a realistic time frame
- Exploration time for optimization is not feasible in a realistic time frame

[Idea] Can we solve this by mimicking human learning?

- Refer to **satisficing**, which is a human learning tendency
- Introduce an aspiration level to RL
- Change the goal of RL (optimizing) to satisficing (but it is also possible to optimize it)

We implemented **target-oriented exploration**
in order to achieve the aspiration level through learning.

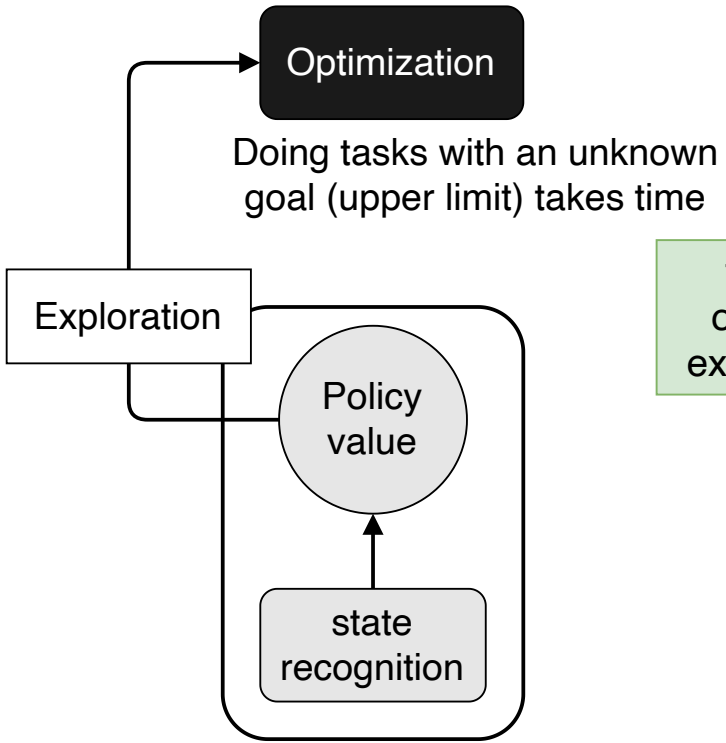
Objectives of this study

[Main objective]
Application of target-oriented exploration to deep reinforcement learning

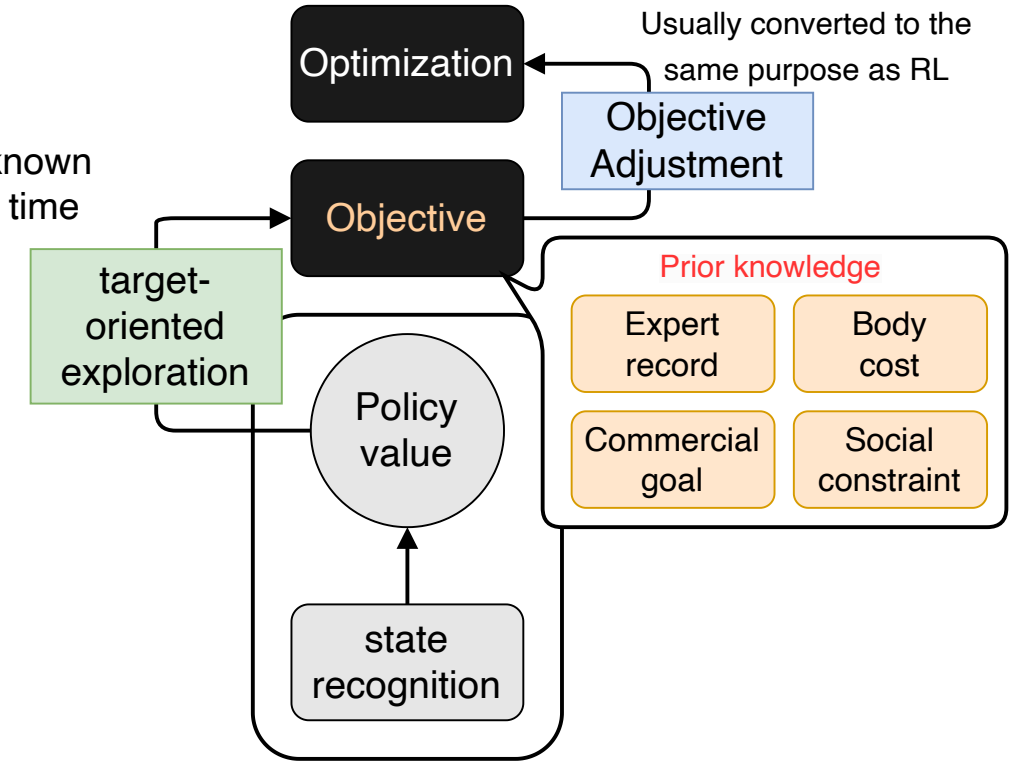
[Sub objective]
Generalize the action choice of target-oriented exploration methods to be stochastic

- Generalize methods that can approximate states
- Demonstrates that new method generalizes related method stochastically without significant performance degradation
- Aim to show the same or better performance as representative methods

Conventional RL



Target oriented RL



Related research

Risk-sensitive Satisficing (RS)

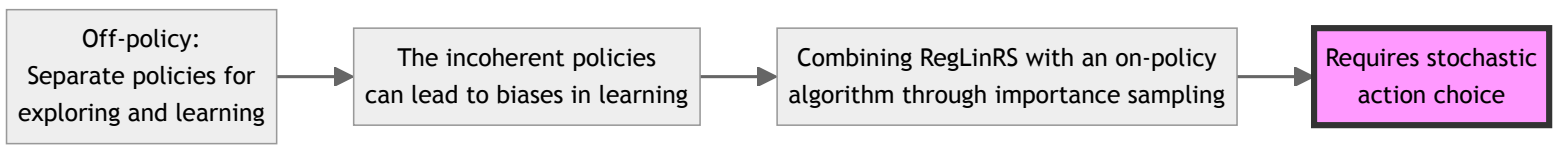
- Overview: Incorporates an aspiration level into RL (target-oriented RL)
- Properties: Off-policy, Deterministic action selection
- Features: Better performance than other methods in bandit problems and reinforcement learning problems
  - Takahashi et al., 2016
  - Tamatsukiri et al., 2019

Related research

Regional Linear RS (RegLinRS)

- Overview: One of the RS methods that can identify states
- Properties: Off-policy, Deterministic action selection
- Features: Better performance than LinUCB and LinTS in contextual bandit problems
  - Tsuboya et al., 2023

Expanding to deep reinforcement learning, probabilistic action selection is preferable



Contextual Bandit Problems

- Experiment task: Linear contextual bandit problems
- Context $\mathbf{x}_t$ and weight θ_i determine the reward expectation value $p_{t,i}$ of each action a_i
- The agent observes the context $\mathbf{x}_t$ at time t and selects the action a_i
  - As a result, the agent observes the reward r_t (in this study, the reward expectation value $p_{t,i}$)
- The calculation method of $p_{t,i}$ is as follows

$$p_{t,i} = \mathbf{x}_t^T \theta_i + \epsilon_t$$

  - θ_i: The parameter of the reward expectation value
  - ϵ_t: The error term with an expected value of 0

Methods that have performed well in contextual bandit problems

- LinUCB (Li et al., 2010)
- LinTS (Riquelme et al., 2018)

Regret

- We use **regret** as an evaluation index.
  - Expected reward loss

$$\text{regret} = \sum_{t=1}^{T} (p_{\max} - p_{t,\,\text{chosen}})$$

    - $p_{\max}$: The highest reward expectation value
    - $p_{t,\,\text{chosen}}$: The reward expectation value of the action selected in the t-th round

- Properties of regret
  - Regret is the expected loss of the agent and is a monotonically increasing function
  - The minimum value of regret is 0 (when the optimal action is selected)

Subjective regret

Implementing target-oriented exploration

- If it is target-oriented exploration, we can use a subjective index instead of regret
- Use subjective regret (SR)

$$I_i^{\mathrm{SR}} = \sum_{t=1}^{T} (\aleph - p_{t,\,\mathrm{chosen}})$$

  - $\aleph$: Aspiration level

- Properties of SR
  - If the newly acquired reward is greater than or equal to $\aleph$ (i.e., sufficient), I_i^{SR} decreases
  - If the newly acquired reward is less than $\aleph$ (i.e., insufficient), I_i^{SR} increases
  - This index can also be interpreted as a risk-sensitive value function

Risk-sensitive Satisficing (RS)

Implementing target-oriented exploration

- The formula for the index that is the core of target-oriented exploration
- Define the RS value function by $I_i^{\mathrm{RS}} := -I_i^{\mathrm{SR}}$
- Select an action by taking argmax from I_i^{RS}

$$I_i^{\mathrm{RS}} = \frac{n_i}{N}(p_i - \aleph) = \frac{n_i}{N}\delta_i$$

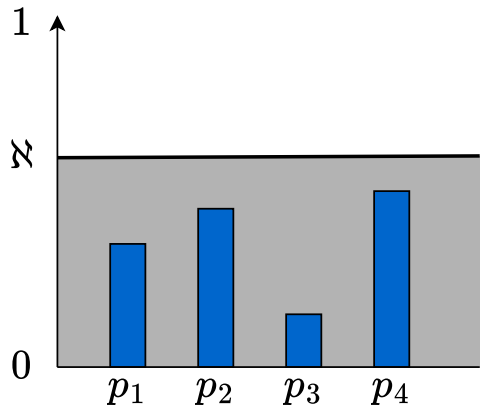
  - p_i: Reward expectation value of action a_i
  - n_i: The number of times action a_i was selected
  - N: The number of action selections so far
  - n_i/N: Reliability, Choice propability

- δ_i: Reflection effect of prospect theory → Difference between aspiration level: $p_i - \aleph$
  - Optimistic or pessimistic action selection depending on the situation by multiplying with reliability

Under-archieved and Over-archieved situation

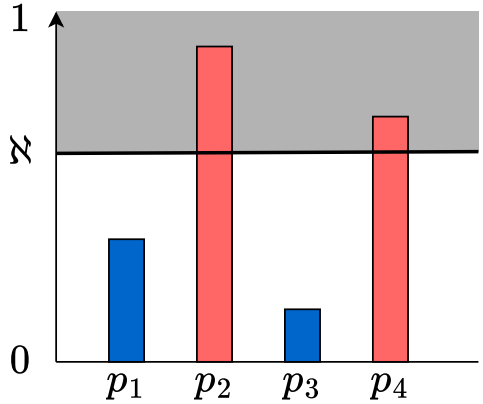
Under-archieved situation

All reward expectation values are less than ℵ



Over-archieved situation

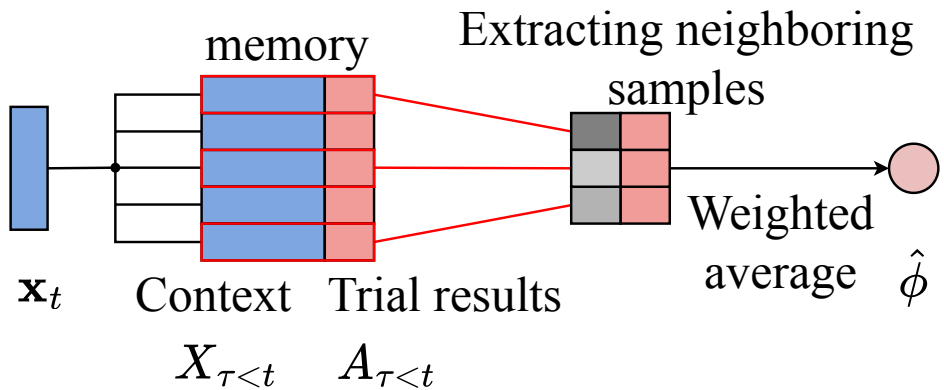
At least one reward expectation value is greater than or equal to ℵ



Exploration area

Local approximation of the reliability

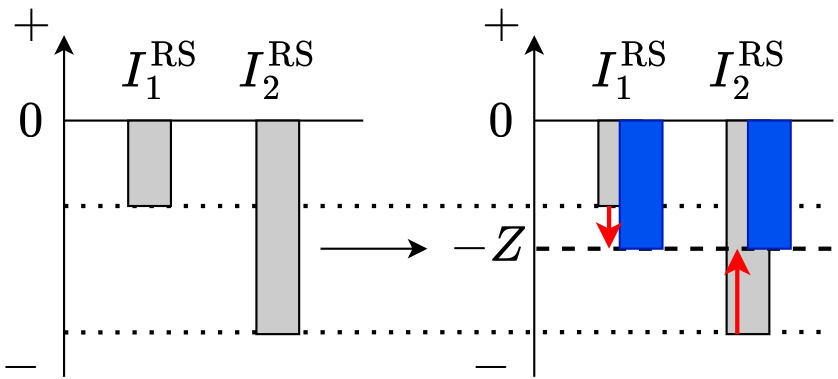
- Approximate the reliability using episodic memory and k-nearest neighbors
- **Regional Linear RS** (RegLinRS)
  - A method proposed by Tsuboya et al., 2023



Inversly calculating the choice distribution of RS

- RS is estimable for internal choice ratio in under-archieved situation
- Generate a probability distribution from the difference between the estimated and actual reliability
  - Generate the estimated reliability ρ_i^z using the RS equilibrium value $-Z$
- **Stochastic RS** (SRS)

RS equilibrium value $-Z$



$$I_i^{\mathrm{RS}} = -Z$$

$$\rho_i = n_i/N = Z/(\aleph - p_i)$$

$$\sum_{i=1}^{K} \rho_i = \sum_{i=1}^{K} Z/(\aleph - p_i) = 1$$

$$Z = 1/\sum_{i=1}^{K} \frac{1}{\aleph - p_i}$$

Calculate policy of SRS

Under-archieved situation

- $-Z$ is calculable in under-archieved situation

$$b_i = \frac{n_i}{\rho_i^z} - N + \epsilon$$
$$I_i^{\mathrm{SRS}} = (N + \max_i(b_i))\rho_i^z - n_i > 0$$
$$\pi_i = I_i^{\mathrm{SRS}} / \sum_{i=1}^{K} I^{\mathrm{SRS}_i}$$

- b: Adjustment parameter to prevent negative ratio
- ϵ: Extremely small value to prevent zero division

Over-achieved situation

- $-Z$ is not calculable in over-achieved situation
- Calculate the probability of selecting an action with a reward expectation value greater than $\aleph$

$$I_i^{\mathrm{RS}'} = \begin{cases} I_i^{\mathrm{RS}} + \epsilon, & \text{if } p_i \geq \aleph \\ 0, & \text{if } p_i < \aleph \end{cases}$$
$$\pi_i = I_i^{\mathrm{RS}'} / \sum_{j=1}^{K} I_j^{\mathrm{RS}'}$$

Regional Linear SRS

- **New method**
- A method that combines the reliability estimation part of RegLinRS and SRS
  - The formula of SRS contains n_i and N
  - n_i/N can be approximated, but n_i and N cannot be approximated

Transformation of the equations in SRS

$$\frac{b_i}{N_x} = \frac{1}{\rho_i^z} \cdot \frac{n_i}{N_x} - 1 + \epsilon = \frac{\rho_i}{\rho_i^z} - 1 + \epsilon$$

$$\begin{aligned}\frac{I_i^{\mathrm{SRS}}}{N_x} &= \left(\frac{N_x + \max_i(b_i)}{N_x}\right) \\ &= \left\{\max_i\left(\frac{\rho_i}{\rho_i^z}\right) + \epsilon\right\}\rho_i^z - \rho_i\end{aligned}$$

$$\frac{I_i^{\mathrm{SRS}}}{N_x} = \left\{\max\left(\frac{\hat{\phi}_i}{\rho_i^z}\right) + \epsilon\right\}\rho_i^z - \hat{\phi}_i$$

$$\pi_i = \frac{I_i^{\mathrm{SRS}}}{N_x} \Big/ \sum_{j=1}^{K} \frac{I_j^{\mathrm{SRS}}}{N_x} = I_i^{\mathrm{SRS}} \Big/ \sum_{j=1}^{K} I_j^{\mathrm{SRS}}$$

→ SRS can be extended to RegLinSRS

Artificial dataset

- Create an artificial dataset with a constant aspiration level $\aleph$
  - For comparison with RegLinRS, RegLinSRS, and other methods using the same evaluation index

- Use the same dataset as Tsuboya et al., 2023
- We designed the dataset so that the optimal action would not be biased in order to properly evaluate the balance between exploration and exploitation

Configuration Item	Configuration Value
Feature vector dimension d	128
Number of actions K	8
Optimal Aspiration level $\aleph_{\mathrm{opt}}$	0.7
Data size N	100,000

※ $\aleph_{\mathrm{opt}}$: The reference value set between the optimal and suboptimal, see the paper for details

Experiment 1

- Comparison methods
  - LinUCB
  - LinTS
  - RS series
    - RegLinRS
    - RegLinSRS

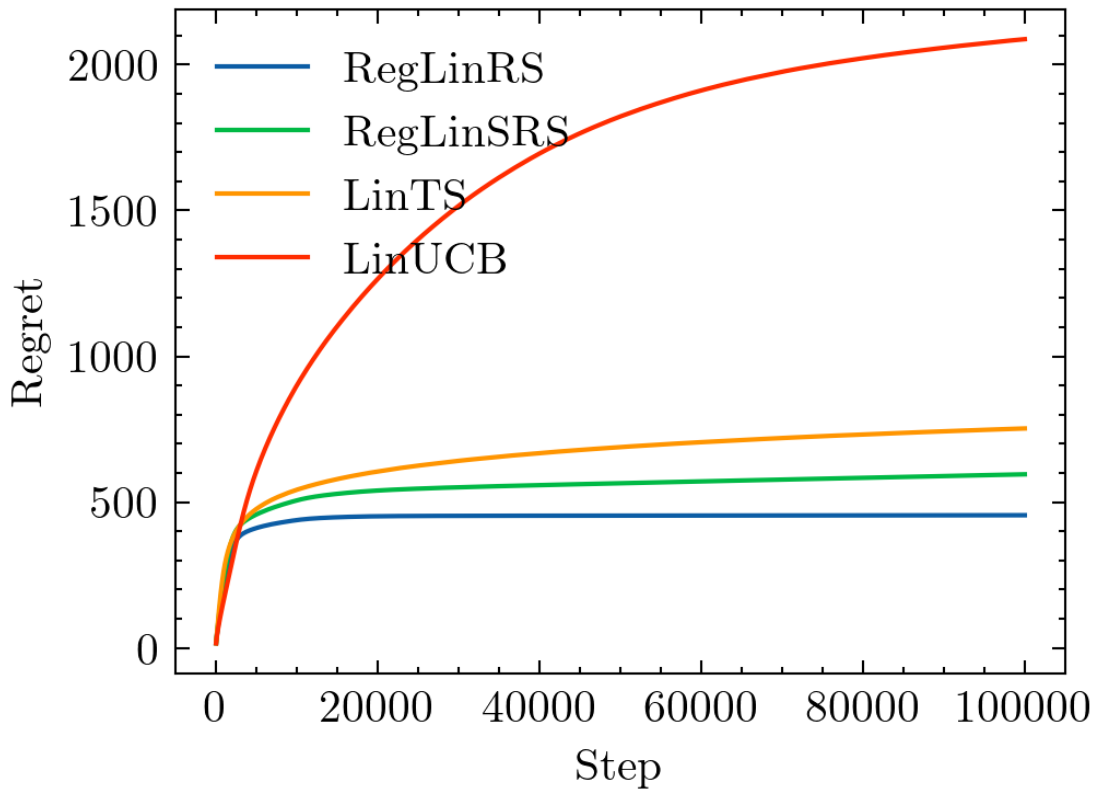
- 1,000 simulations with 100,000 steps per simulation
  - Calculate the average value of the results
- Select each action 10 times at first
  - For parameter initialization
- Set the batch size to 20 for all methods

Value Name	Value
ϵ	sys.float_info.epsilon in Python
episodic memory size	10,000
k of K-nearest neighbors	50
$\aleph$	0.6
α of LinUCB	0.1
λ of LinTS	0.25
α of LinTS	6
β of LinTS	6
$\mathbf{b}_i$	All 0
$\mathbf{A}_i$	Identity matrix $\mathbf{I}$

Result

Experiment 1

Good



Result

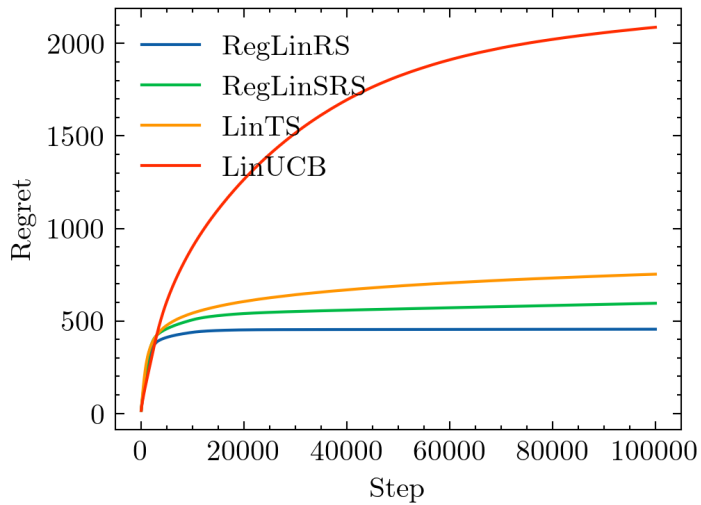
Experiment 1

- RegLinRS, RegLinSRS, LinTS, LinUCB performed well in that order
- LinTS, LinUCB have a logarithmic increase in regret
- RegLinRS, RegLinSRS have almost converged regret

- LinTS is one of the state-of-the-art methods (Agrawal et al., 2019)
- RegLinRS, RegLinSRS are inferior to LinTS in the initial rise of regret

↓

RegLinRS, RegLinSRS can stop learning faster than LinTS and can select the truly optimal action even in situations where accurate approximation has not yet been achieved?

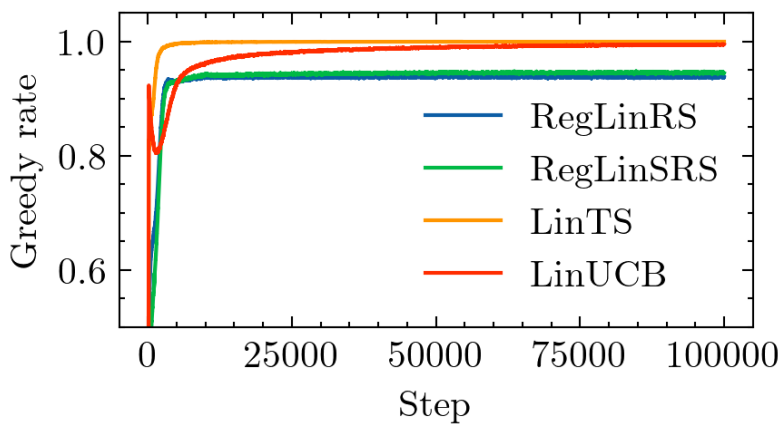


Discussions

Experiment 1

- LinTS, LinUCB have reached a Greedy rate of 1.0
- RegLinRS, RegLinSRS have stopped at a Greedy rate of over 0.9
  - Not greedy about 1 in 10 times

→ RegLinRS, RegLinSRS can select the truly optimal action even if the action is overestimated



Hypothesis

Experiment 1

[Fact] RegLinRS, RegLinSRS can partially mitigate the effects of approximation errors

- Can select the truly optimal action even if the action is overestimated

[Hypothesis] Is RegLinRS and RegLinSRS achieving this by reliability?

- RS does not necessarily make greedy action selection due to the reflection effect of reliability
- It is rational in the sense that the agent can definitely select a satisfactory action

→ Verify this property by intentionally adding noise to the reward expectation value

Experiment 2

Experimental purpose

- To verify the robustness of RS to approximation errors
  - Add noise to the estimated reward expectation value
  - Create a situation where it is easy to select a non-optimal action

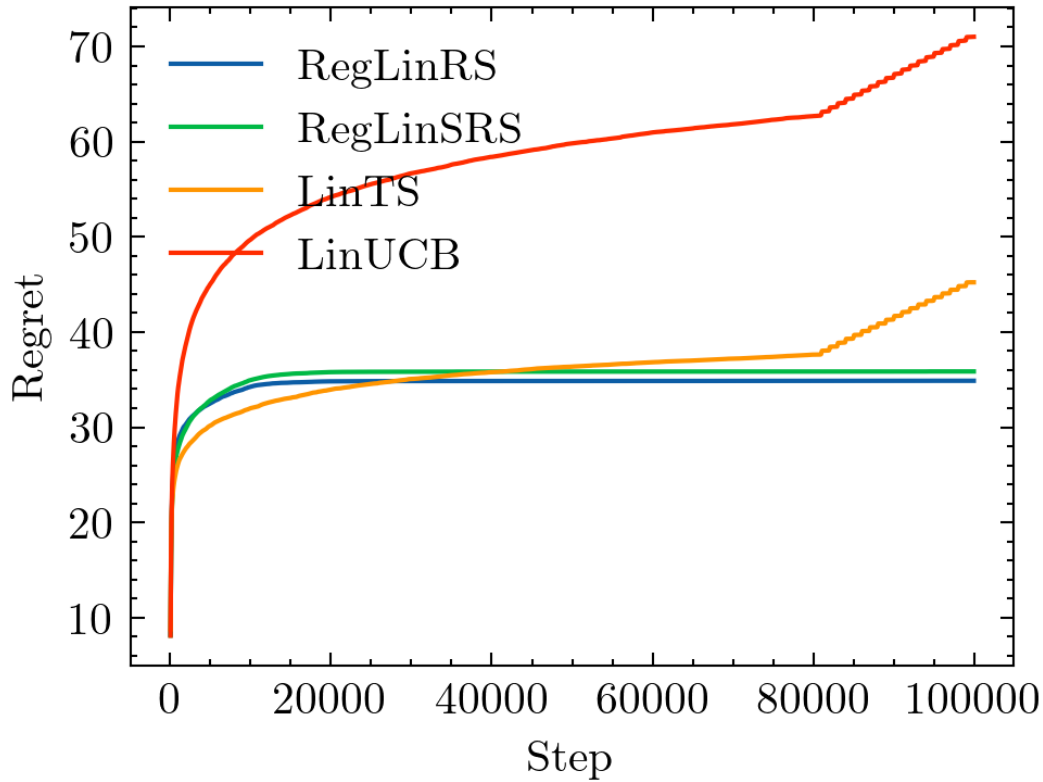
Experimental settings

- To simplify the experiment of Experiment 1, set the number of actions $K = 2$
- Add noise to the estimated reward expectation value at equal intervals after 80,000 steps
- Other settings are the same as Experiment 1

Result

Experiment 2

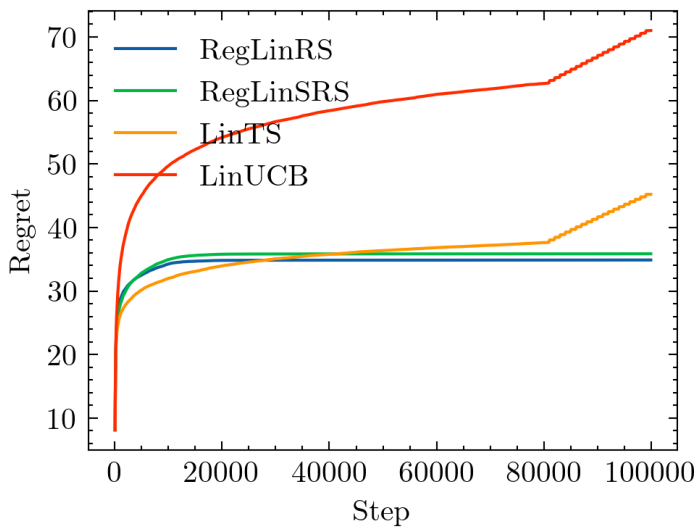
Good



Discussions

Experiment 2

- After the step on which the noise is added, LinTS, LinUCB have a sharp increase in regret
- RegLinRS, RegLinSRS have no increase in regret
  - Partially mitigates the effects of reward expectation value estimation errors due to the reflection effect of reliability
  - RS is robust to approximation errors



Conclusion

- We were able to realize the probabilistic generalization of RegLinRS (RegLinSRS) without significant performance degradation
- We were able to show that it performs better than LinTS, LinUCB

- We were able to show the robustness of RS to approximation errors
  - This property is useful for future deepening

→ We were able to prepare for the application of RS to deep reinforcement learning