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 the real-world tasks.

- The required amount of sampling for <u>optimization</u> is not feasible in a realistic time frame.
- The required amount of exploration for <u>optimization</u> is not feasible in a realistic time frame.

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

- We focus on **satisficing**, a learning tendency of humans.
- We introduce the concept of an **aspiration level** into reinforcement learning.
- We generalize the goal of reinforcement learning from optimization into satisficing (but optimization is also possible).

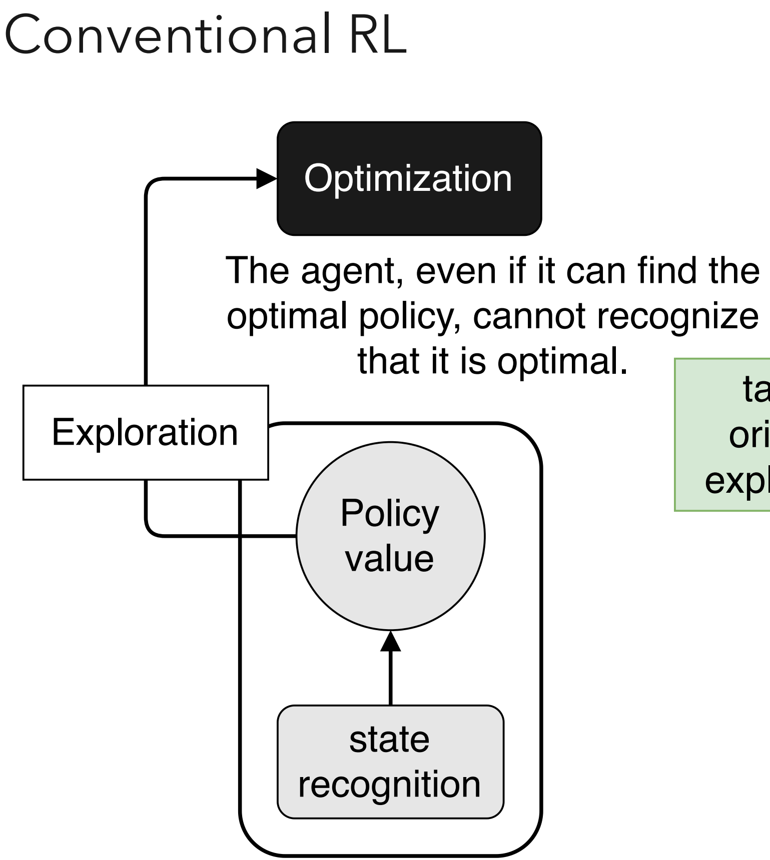
We implemented **target-oriented exploration**,
which is a learning approach that involves aiming for
achieving a specific aspiration level.

The goal of this study

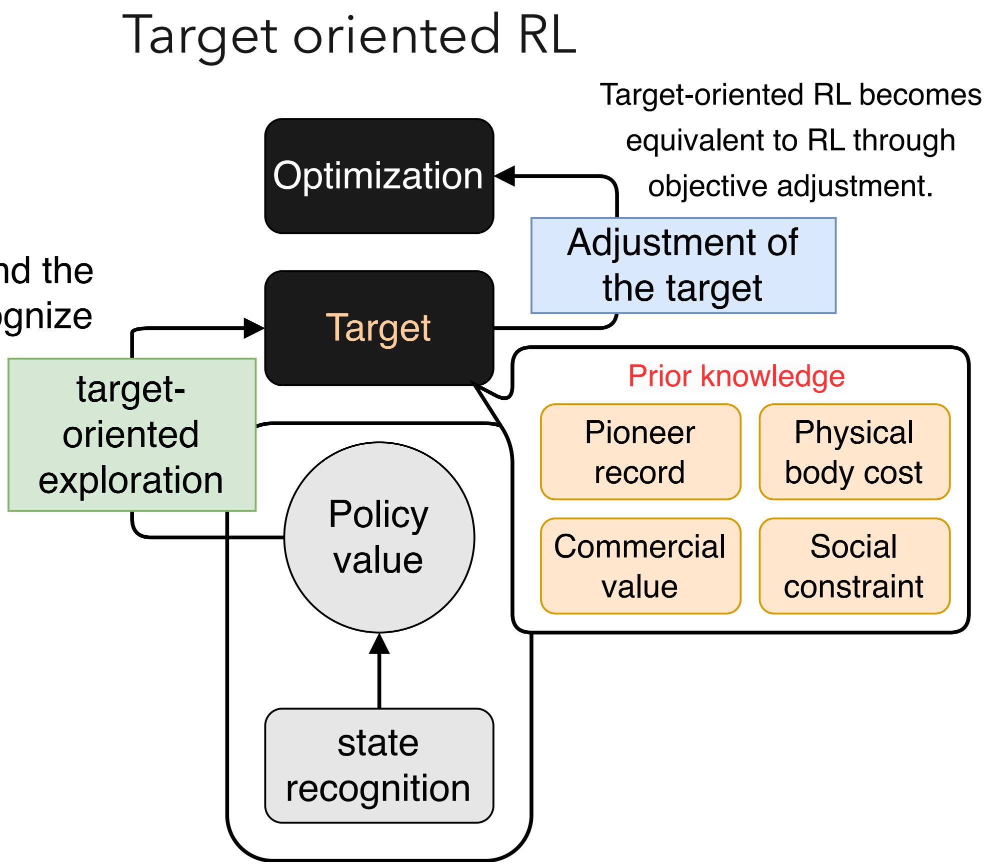
[Our main goal]
Application of target-oriented exploration to deep reinforcement learning

[Our sub goal]
Stochastic generalization of the action selection of target-oriented exploration methods

- In this study, we generalize the existing state approximation methods.
- We show that our new method is a successful generalization (i.e. it works without performance degradation).
- Our goal is to show that the new method performs equal to or better than representative methods.



Conventional RL
Optimization
The agent, even if it can find the optimal policy, cannot recognize that it is optimal.
Exploration
Policy value
state recognition



Target oriented RL
Target-oriented RL becomes equivalent to RL through objective adjustment.
Optimization
Adjustment of the target
Target
target-oriented exploration
Prior knowledge
Pioneer record
Physical body cost
Commercial value
Social constraint
Policy value
state recognition

Related research

Risk-sensitive Satisficing (RS)

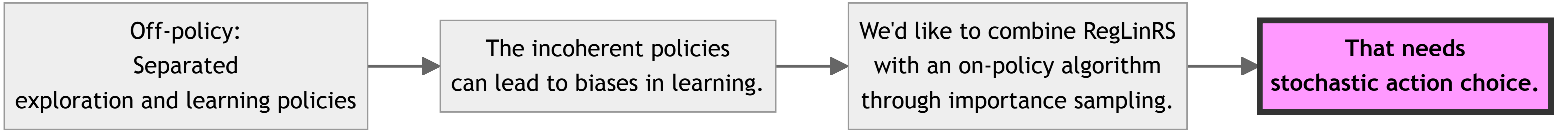
- Overview: RS is a method that incorporated an aspiration level into RL (target-oriented RL).
- Mechanism: Off-policy, Deterministic action selection
- Features: RS showed better performance than other methods in bandit problems and RL problems.
  - Takahashi et al., 2016
  - Tamatsukiri et al., 2019

Related research

Regional Linear RS (RegLinRS)

- Overview: RegLinRS is one of the RS methods that can identify states.
- Mechanism: Off-policy, Deterministic action selection
- Features: RegLinRS showed better performance than LinUCB and LinTS in contextual bandit problems.
  - Tsuboya et al., 2023

We want to generalize deterministic action selection into stochastic action selection.



Contextual Bandit Problems

- Experimental task: Linear contextual bandit problems
- The agent calculates the reward expectation values $p_{t,i}$ of each action a_i by the context $\mathbf{x}_t$ and the parameter θ_i.
- The agent observes the context $\mathbf{x}_t$ at time t and chooses an action a_i.
  - As the 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 performing well in contextual bandit problems

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

Regret

- We use **regret** as an evaluation index.

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

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

- Properties of regret
  - Regret is the loss expectation value of the agent, and it is a weakly increasing function.
  - The minimum value of regret is 0 (when the agent continues to choose the optimal action).

Subjective regret

Implementation of target-oriented exploration

- When we use target-oriented exploration, we can use a subjective index instead of regret.
  - We call this **subjective regret** (SR).

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

- $\aleph$: Aspiration level

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

Risk-sensitive Satisficing (RS)

Implementation of target-oriented exploration

- The formula for the core metric of target-oriented exploration
  - We define the RS value function as $I_i^{\mathrm{RS}} := -I_i^{\mathrm{SR}}$.
  - The agent chooses an action by taking the argmax from I_i^{RS}.

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

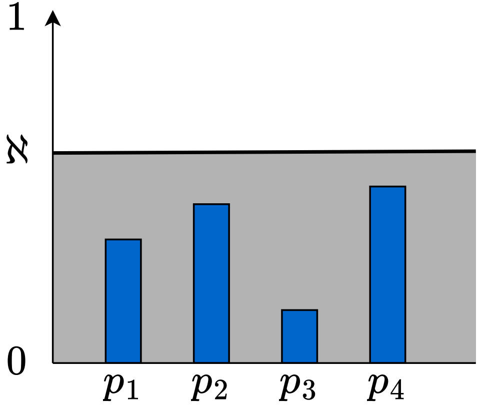
- p_i: Reward expectation value of action a_i
- n_i: The number of times the agent chose action a_i
- N: The total number of times the agent chose an action
- n_i/N: Reliability (Choice probability) of action a_i

- δ_i: Reflection effect of prospect theory → Difference between aspiration level $(p_i - \aleph)$
  - By multiplying the reliability and δ_i, the agent makes optimistic or pessimistic action choices depending on the situation.

Under-archieved and Over-archieved situations

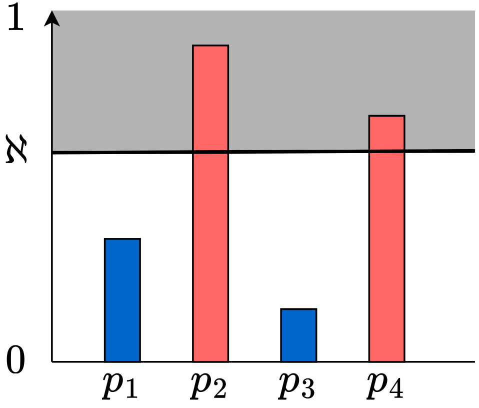
Under-archieved situation

All reward expectation values are less than $\aleph$.



Over-archieved situation

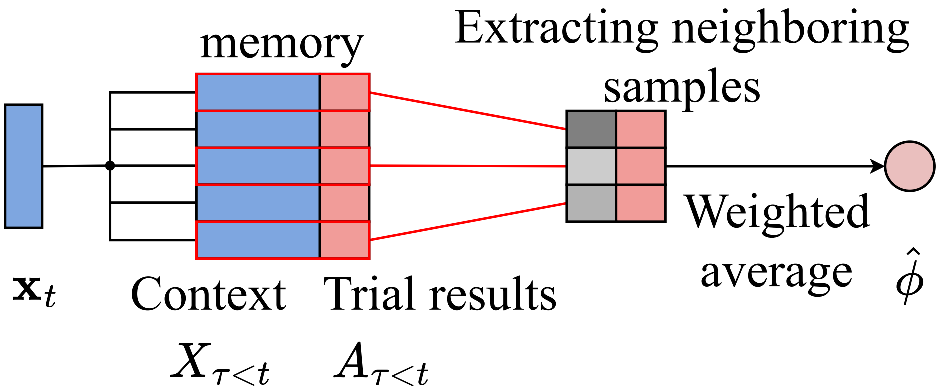
At least one reward expectation value is greater than or equal to $\aleph$.



Exploration area

Local approximation of reliability

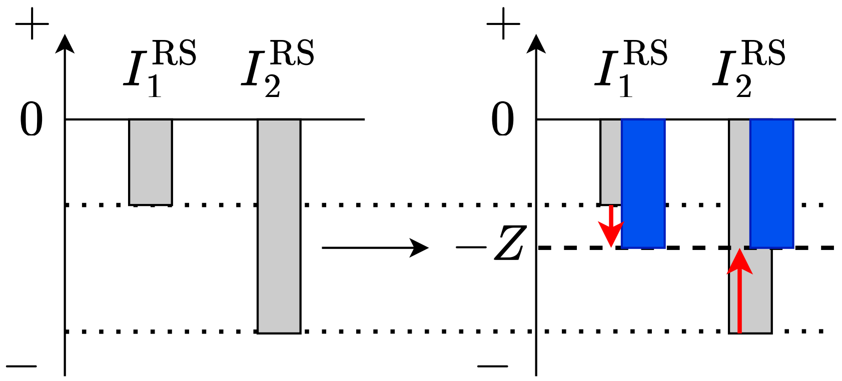
- The agent approximates and estimates reliability using episodic memory and k-nearest neighbor.
- **Regional Linear RS** (RegLinRS)
  - Tsuboya et al., 2023



Inverse calculation of the choice distribution of RS

- We can estimate the internal choice ratio of RS in the under-archieved situation.
- The agent generates a probability distribution from the difference between the estimated reliability and the actual reliability.
  - The agent generates the estimated reliability ρ_i^z using the RS equilibrium value $-Z$.
- **Stochastic RS** (SRS)

RS equilibrium value $-Z$



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

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

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

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

How to calculate the policy of SRS

Under-archieved situation

- We can calculate $-Z$.

$$b_i = \frac{n_i}{\rho_i^z} - N + \epsilon$$

$$I_i^{\mathrm{SRS}} = (N + \max_i(b_i))\rho_i^z - n_i > 0$$

$$\pi_i = I_i^{\mathrm{SRS}} / \sum_{i=1}^{K} I^{\mathrm{SRS}_i}$$

- b: Adjustment parameter for preventing negative ratios
- ϵ: An extremely small value to prevent zero division

Over-achieved situation

- We cannot calculate $-Z$.
- Instead, we calculate the probability of choosing an action that has 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

- **Our new method**
- With this method, the reliability estimation part of RegLinRS is combined with SRS.
  - The formula of SRS contains n_i and N.
  - We can approximate n_i/N, but we cannot approximate n_i and N.

Transformation of the equations in SRS

$$
\begin{aligned}
\frac{b_i}{N_x} &= \frac{1}{\rho_i^z} \cdot \frac{n_i}{N_x} - 1 + \epsilon = \frac{\rho_i}{\rho_i^z} - 1 + \epsilon \\
\frac{I_i^{\mathrm{SRS}}}{N_x} &= \left(\frac{N_x + \max_i(b_i)}{N_x} \right) \\
&= \left\{ \max_i \left(\frac{\rho_i}{\rho_i^z} \right) + \epsilon \right\} \rho_i^z - \rho_i
\end{aligned}
$$

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

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

→ We can extend SRS to RegLinSRS.

Artificial dataset

- We created an artificial dataset in which the aspiration level $\aleph$ is always constant.
  - The purpose is to compare RS methods (RegLinRS and RegLinSRS) with other methods using the same evaluation index.

- We used 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

※ $\aleph_{\mathrm{opt}}$ is the reference value set between the optimal and suboptimal.

Experiment 1

RL methods

LinUCB, LinTS, and RS methods
(RegLinRS, RegLinSRS)

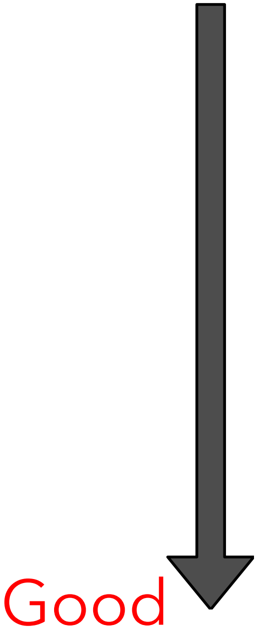
Experimental settings

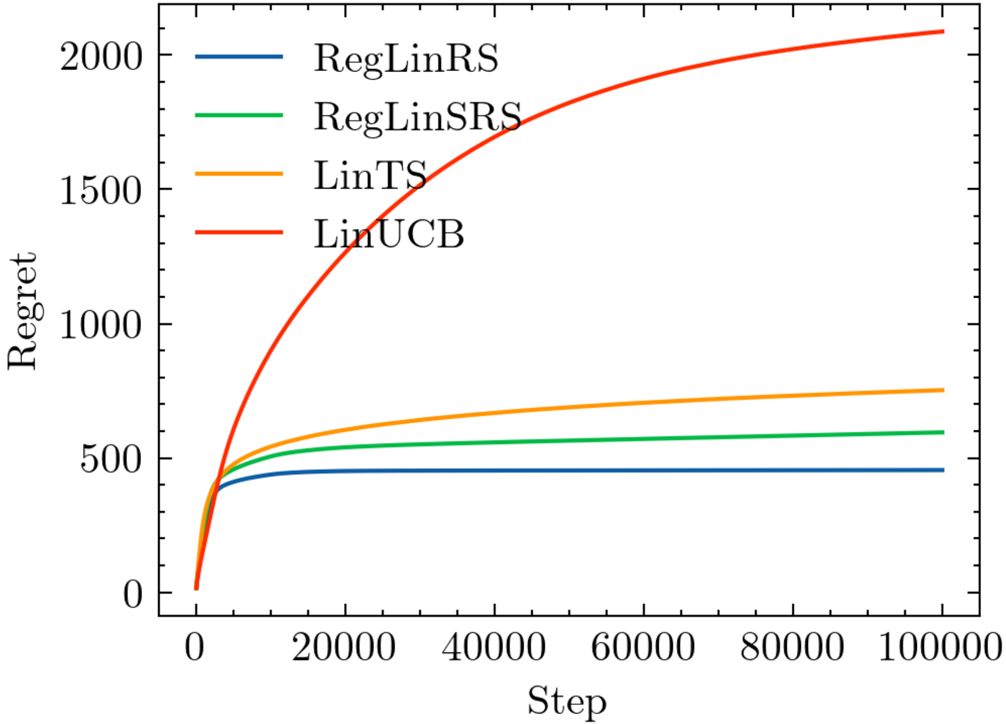
- We ran 1,000 simulations, with 100,000 steps per simulation.
  - We calculated the average values and used them as the result.
- The agent initially selects each action 10 times.
  - This setting is necessary for parameter initialization.
- We set the batch size to 20 for all methods.

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





Result

Experiment 1

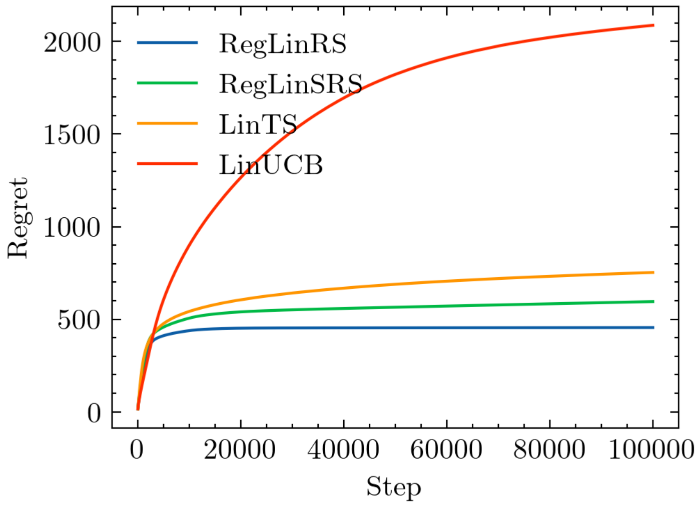
- RegLinRS, RegLinSRS, LinTS, LinUCB performed well in that order.
- LinTS and LinUCB have a logarithmic increase in regret.
- RegLinRS and RegLinSRS have almost converged regret.

- LinTS is one of the state-of-the-art methods (Agrawal et al., 2019).
- However, from the early steps, the regret of LinTS is larger than that of RegLinRS and RegLinSRS.

↓

A question:

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?

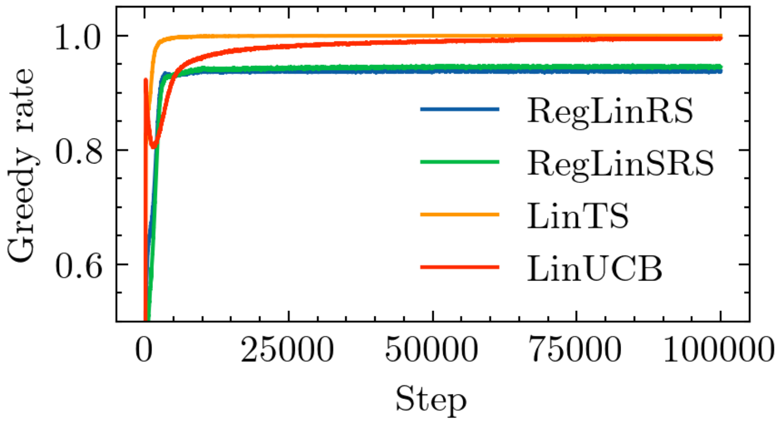


Discussions

Experiment 1

- LinTS and LinUCB have reached a Greedy rate of 1.0.
- RegLinRS and RegLinSRS have stopped at a Greedy rate of over 0.9.
  - About once in 10 times, they were not greedy.

→ RegLinRS and RegLinSRS can choose the truly optimal action even if the action is overestimated.



Hypothesis

Experiment 1

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

- These methods can choose the truly optimal action even if the action is overestimated due to approximation errors.

[Hypothesis] Are RegLinRS and RegLinSRS achieving this using reliability?

- RS does not necessarily make greedy action selection, because it uses the reflection effect of reliability.
- This property is effective in the sense that the agent can choose a satisfactory action with certainty.

→ We conducted an experiment to verify this property
by intentionally adding noise to the reward expectation values.

Experiment 2

Purpose of Experiment

- **To verify the robustness of RS against approximation errors**
  - We added noise to the estimated reward expectation value.
  - We intentionally created a situation where it is easy to select a non-optimal action.

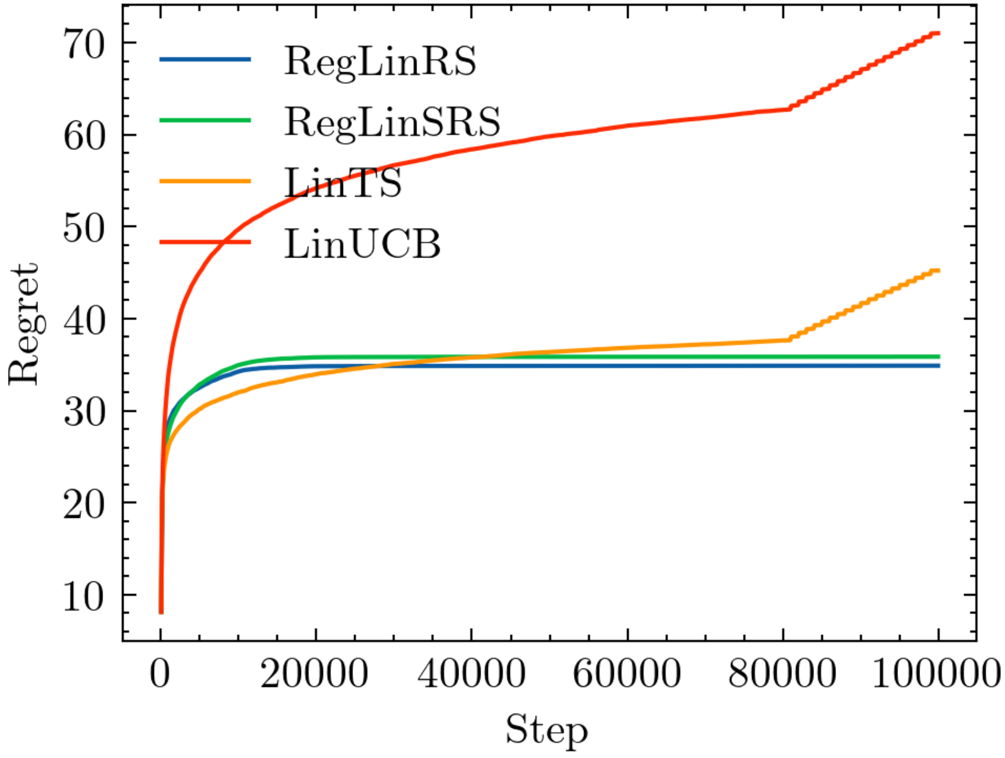
Experimental settings

- We set the number of actions to $K = 2$, to simplify Experiment 1.
- We added noise to the estimated reward expectation value at equal intervals after 80,000 steps.
- We set the other settings the same as in Experiment 1.

Result

Experiment 2

Good

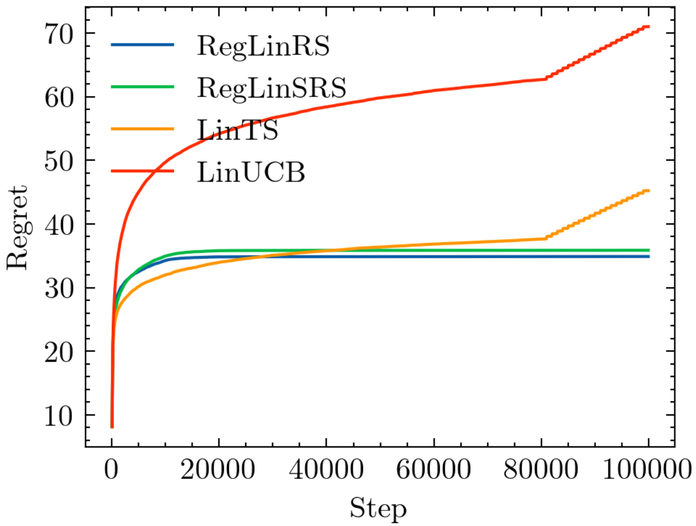


Result

Experiment 2

- After the step we added noise, LinTS and LinUCB have a sharp increase in regret.
- RegLinRS and RegLinSRS have no increase in regret.
  - The latter two methods can partially mitigate the effects of approximation errors by the reflection effect of reliability.

→ It shows the robustness of RS to approximation errors.



Conclusion

- We generalized deterministic action selection (RegLinRS) into **stochastic action selection (RegLinSRS)**.
- We showed that
  - RegLinSRS **does not have a significant performance degradation** compared to RegLinRS.
  - RegLinSRS **has better performance than LinTS and LinUCB**.
- We showed **the robustness of RS to approximation errors**.
  - We think this property is useful when we extend RS to deep RL in the future.

→ We are now prepared for the application of RS to deep RL.