Combinatorial Rising Bandits

ICLR 2026 poster

Seockbean Song[1] , Youngsik Yoon[1], Siwei Wang[2], Wei Chen[2], Jungseul Ok[1]

POSTECH[1], Microsoft Research Asia[2]

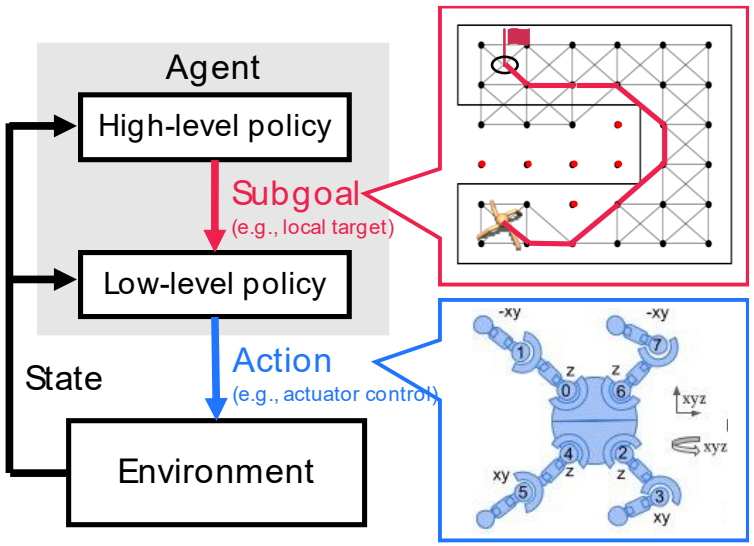
Outline

- Motivation
- Problem formulation
- Method: Combinatorial Rising Upper Confidence Bound (CRUCB)
- Regret analysis
- Experiment

Foundation: Online combinatorial learning

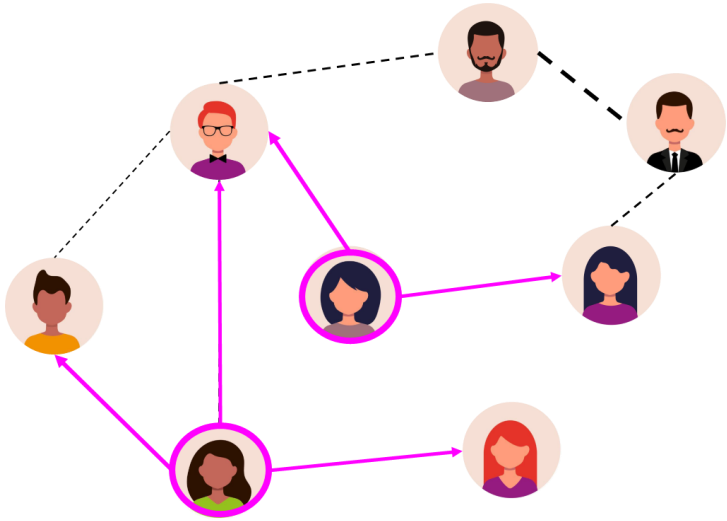
- Online combinatorial learning involves a sequential selection of a combination of actions

Set of subgoals



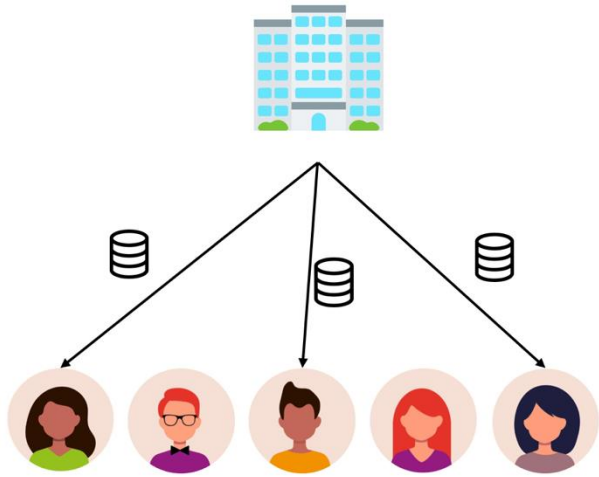
Hierarchical reinforcement learning (Robotics)

Set of seeds



Social advertisement

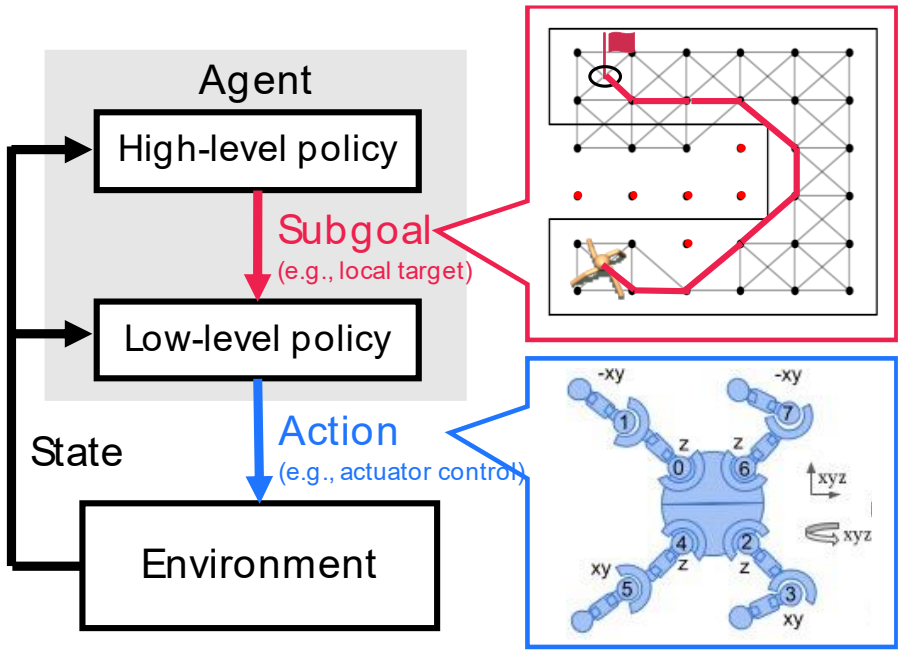
Set of workers

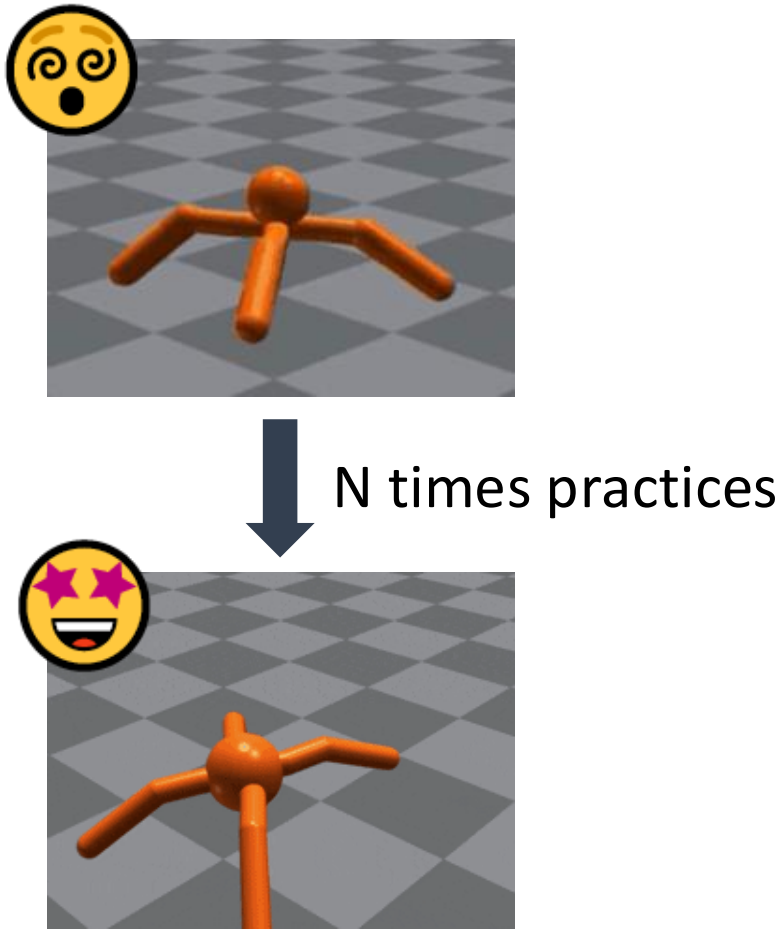


Worker selection in crowdsourcing

Observation: Reward rises with repetitive practice

- Robotics: The more a robot practices, the more skilled it becomes





Proposed Framework: Combinatorial Rising Bandit

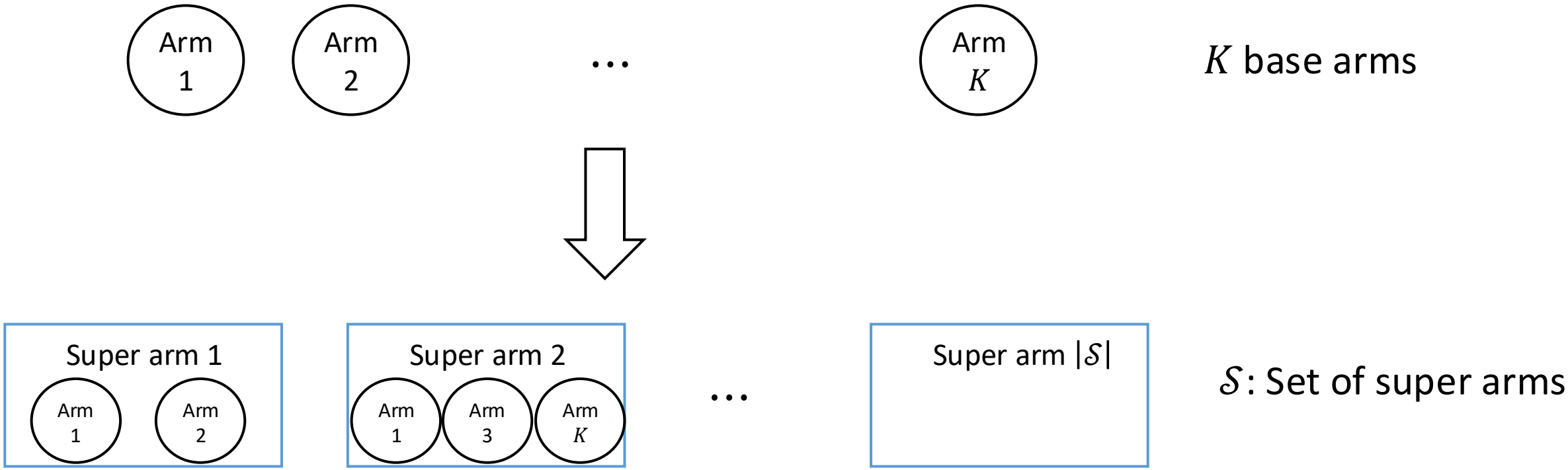
- **Problem:** Existing combinatorial bandit frameworks assume stationary rewards → They cannot model this rising behavior

- Therefore, we formulate **Combinatorial Rising Bandits (CRB)**, a new framework that combines combinatorial action selection with rising rewards

Problem formulation

- A super arm $S_t \in \mathcal{S} \subset 2^{[K]}$ = a combination of base arms
- Each base arm i associated with a rising reward function $\mu_i: [T] \mapsto [0,1]$ such that

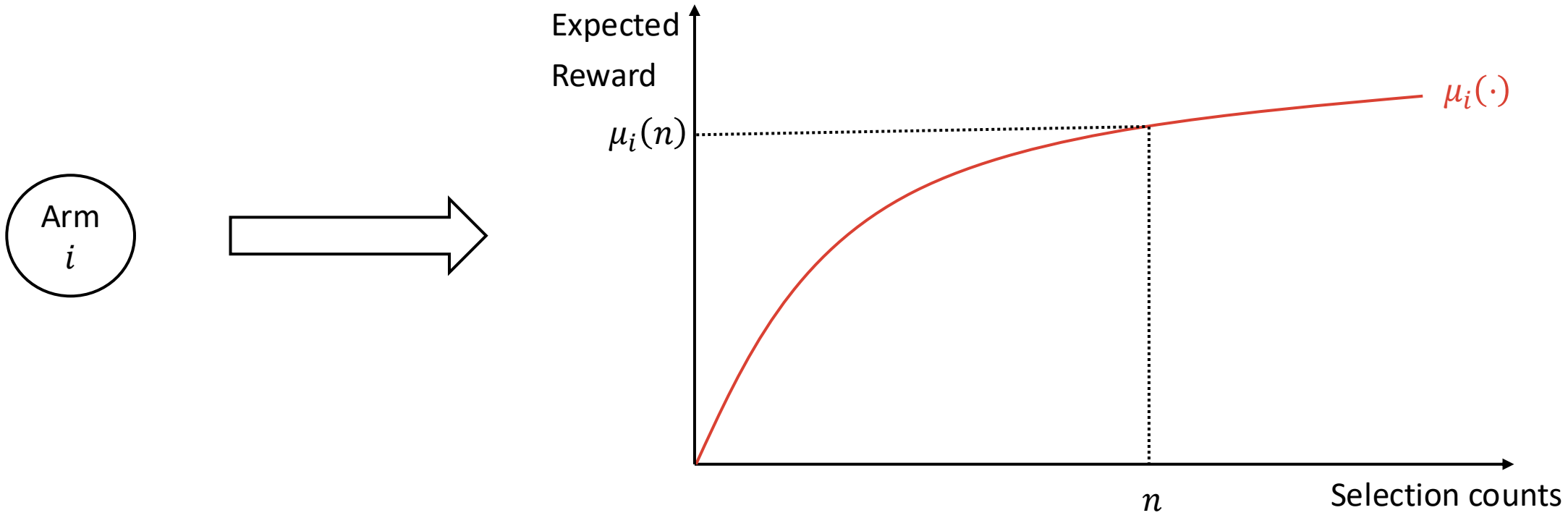
Reward slope $\gamma_i(n) \coloneqq \mu_i(n+1) - \mu_i(n) \geq 0$



Problem formulation

- A super arm $S_t \in \mathcal{S} \subset 2^{[K]}$ = a combination of base arms
- Each base arm i associated with a rising reward function $\mu_i : [T] \mapsto [0,1]$ such that

Reward slope $\gamma_i(n) \coloneqq \mu_i(n+1) - \mu_i(n) \geq 0$

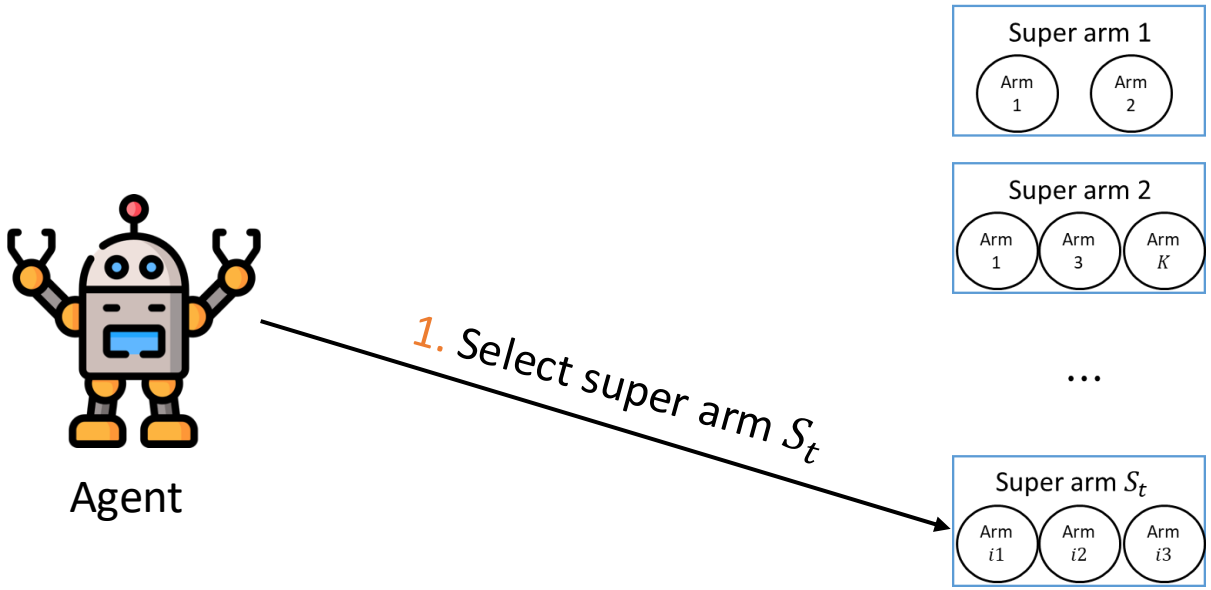


Problem formulation

- A super arm $S_t \in \mathcal{S} \subset 2^{[K]}$ = a combination of base arms
- Each base arm i associated with a rising reward function $\mu_i : [T] \mapsto [0,1]$ such that

$$\text{Reward slope } \gamma_i(n) \coloneqq \mu_i(n+1) - \mu_i(n) \geq 0$$

- For each $t \in [T]$:
  1. Agent chooses a super arm S_t
  2. Agent observes reward $X_{S_t}(t)$ and outcome $X_i(t)$ for each base arm $i \in S_t$

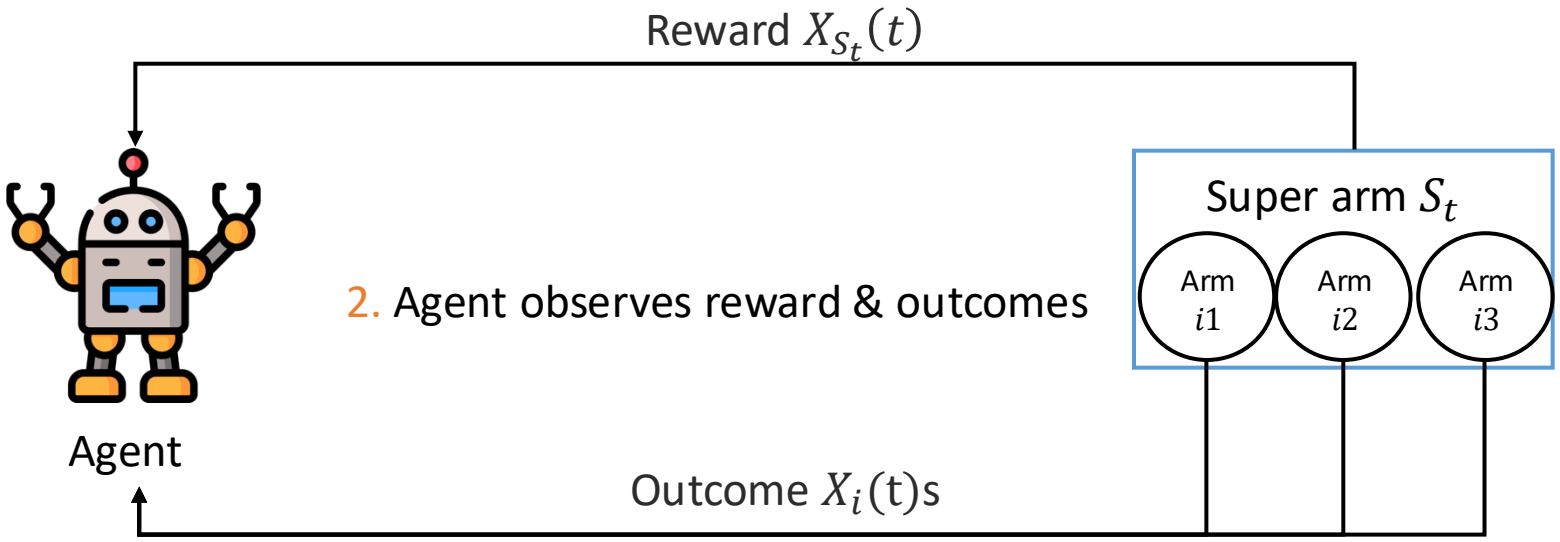


Problem formulation

- A super arm $S_t \in \mathcal{S} \subset 2^{[K]}$ = a combination of base arms
- Each base arm i associated with a rising reward function $\mu_i : [T] \mapsto [0,1]$ such that

$$\text{Reward slope } \gamma_i(n) \coloneqq \mu_i(n+1) - \mu_i(n) \geq 0$$

- For each $t \in [T]$:
  1. Agent chooses a super arm S_t
  2. Agent observes reward $X_{S_t}(t)$ and outcome $X_i(t)$ for each base arm $i \in S_t$



Regret minimization

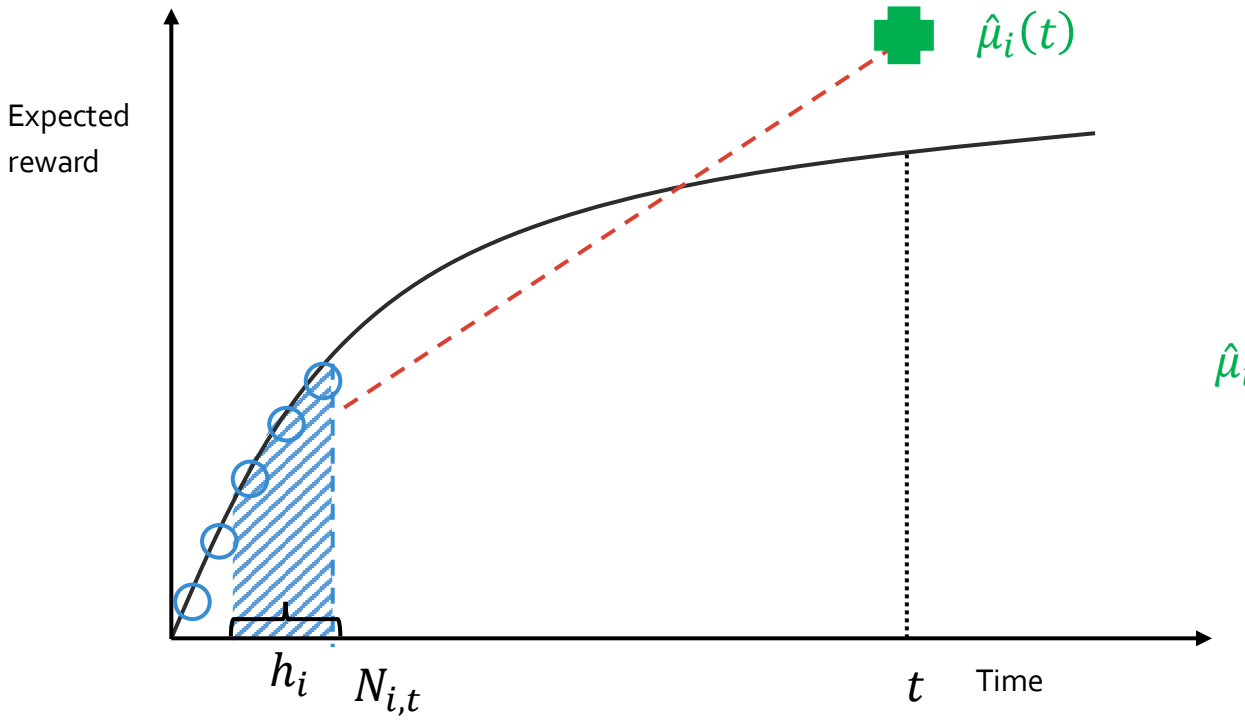
- We aim to design a policy π to minimize regret $R_{\boldsymbol{\mu}}(\pi, T)$:

$$R_{\boldsymbol{\mu}}(\pi, T) \coloneqq \underbrace{\mathbb{E}_{S_t \sim \pi^*}\left[\sum_{t \in [T]} X_{S_t}(t)\right]}_{\text{Reward of optimal policy } \pi^*} - \underbrace{\mathbb{E}_{S_t \sim \pi}\left[\sum_{t \in [T]} X_{S_t}(t)\right]}_{\text{Reward of given policy } \pi}$$

- To minimize regret, policy π should behave similarly to the optimal policy π^*

Algorithm: Combinatorial Rising Upper Confidence Bound

- Step 1: Predict future reward for each base arm
  - For each base arm $i \in [K]$, CRUCB estimates the recent mean and slope

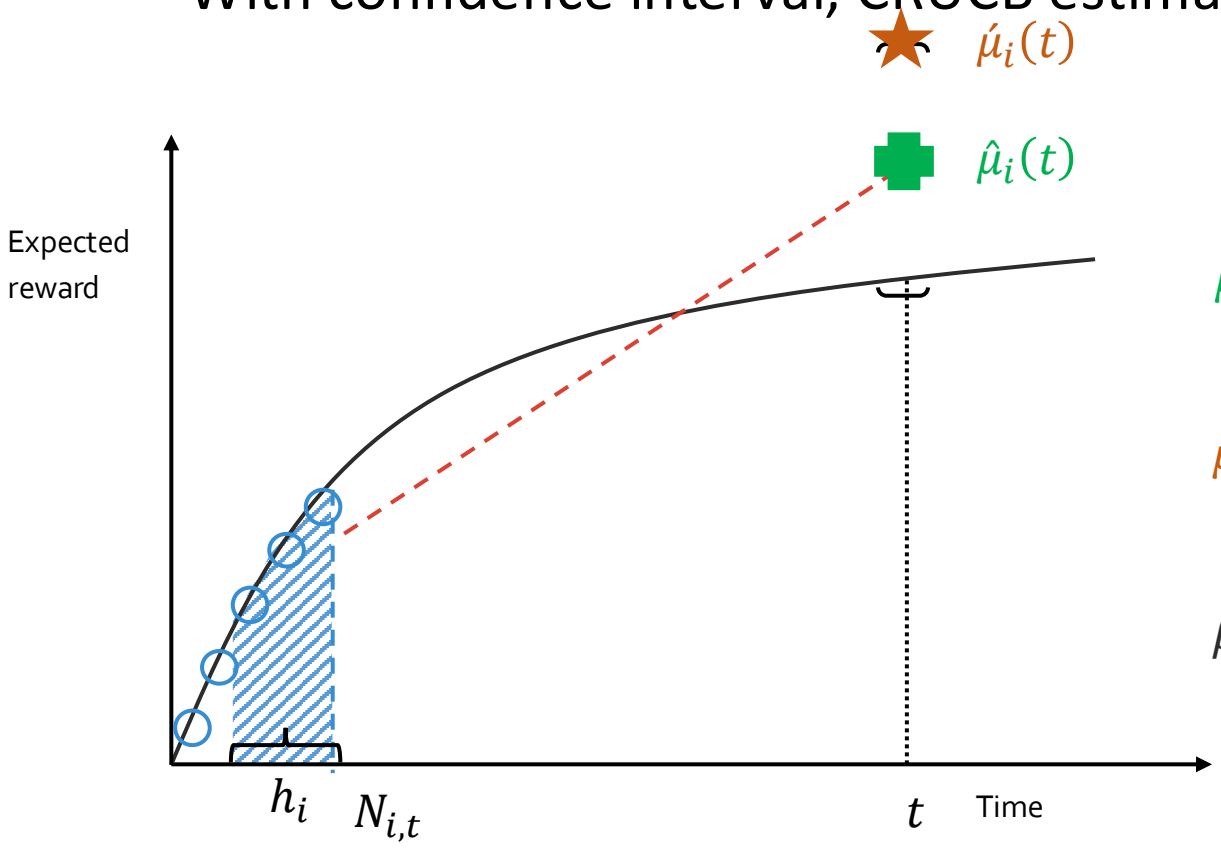


$$\hat{\mu}_i(t) := \frac{1}{h_i}\left(\sum_{l=N_{i,t-1}-h+1}^{N_{i,t-1}}\left(\underbrace{X_i(l)}_{\text{Recent mean}} + \underbrace{(t-l)\frac{X_i(l)-X_i(l-h_i)}{h_i}}_{\text{Slope}}\right)\right)$$

Sliding window

Algorithm: Combinatorial Rising Upper Confidence Bound

- Step 1: Predict future reward for each base arm
  - For each base arm $i \in [K]$, CRUCB estimates the recent mean and slope
  - With confidence interval, CRUCB estimates each base arm by $\acute{\mu}_i(t)$



$$\hat{\mu}_i(t) \coloneqq \frac{1}{h_i}\left(\sum_{l=N_{i,t-1}-h+1}^{N_{i,t-1}}\left(X_i(l) + (t-l)\frac{X_i(l) - X_i(l-h_i)}{h_i}\right)\right)$$

$$\acute{\mu}_i(t) \coloneqq \hat{\mu}_i(t) + \beta_i(t)$$

$$\beta_i(t) \coloneqq \sigma\left(t - N_{i,t} + h_i - 1\right)\sqrt{\frac{10\log t^3}{h_i^3}}$$

Algorithm: Combinatorial Rising Upper Confidence Bound

- Step 2: Select the best combination based on the estimated future UCB
  - Solver is a task-specific algorithm

Algorithm 1: `Combinatorial Rising UCB (CRUCB)`

Input $N_{i,0} \leftarrow 0$ for all $i \in [m]$, Sliding window h_i for all $i \in [m]$.
Initialize Play arbitrary super arm including base arm i two times for each $i \in [m]$.
for $t \in (1, \ldots, T)$ **do**
  For each base arm i, set $\acute{\mu}_i(t) = \hat{\mu}_i(t) + \beta_i(t)$.
  $S_t \leftarrow \text{Solver}(\acute{\mu}_1, \acute{\mu}_2, \cdots, \acute{\mu}_m)$.
  Play S_t and observe reward $X_{S_t}(t)$.
  Update $\hat{\mu}_i(t)$ and $N_{i,t}$.
end for

Regret lower bound of CRB without any constraints

- General regret lower bound of CRB is given as follows:

Theorem 1. ***(Informal)*** *The regret lower bound of CRB is given as:*

$$\max_{\boldsymbol{\mu}} R_{\boldsymbol{\mu}}(\pi, T) \geq \Omega(T)$$

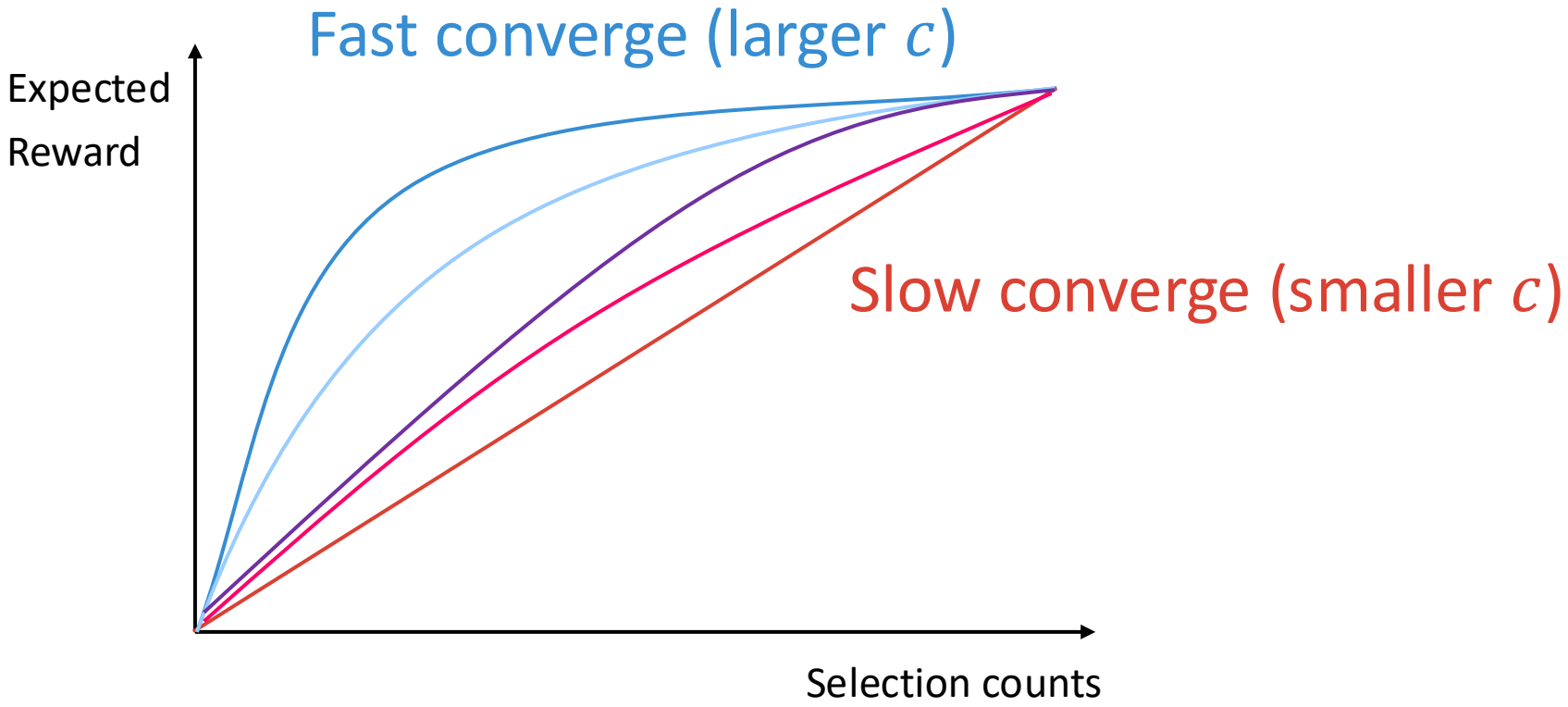
- It means that without any constraints, CRB is impossible to learn

Problem abstraction with curvature constant

- To abstract problem instance, we consider **curvature constant** c

$$\gamma_i(n) \leq (n+1)^{-c} \quad \forall i \in [K]$$

Reward slope $\gamma_i(n) \coloneqq \mu_i(n+1) - \mu_i(n)$



Regret lower bound of CRUCB with constant

- For a given curvature constant c, the regret lower bound of CRB is given as:

Theorem 2. ***(Informal)*** *For a given curvature constant c, any policy π for CRB suffers:*

$$\max_{\boldsymbol{\mu}} R_{\boldsymbol{\mu}}(\pi, T) \geq \Omega\left(\max\left(T^{1/2}, T^{2-c}\right)\right)$$

- For a large c (fast converge), the lower bound is about $\Omega(\mathrm{T}^{1/2})$
- For a small c (slower converge), the lower bound is about $\Omega(T)$ (i.e., impossible to solve)

Regret upper bound of CRUCB

- The regret upper bound of CRUCB is given as follows:

Theorem 3. ***(Informal)*** *the regret upper bound of CRUCB is given as:*

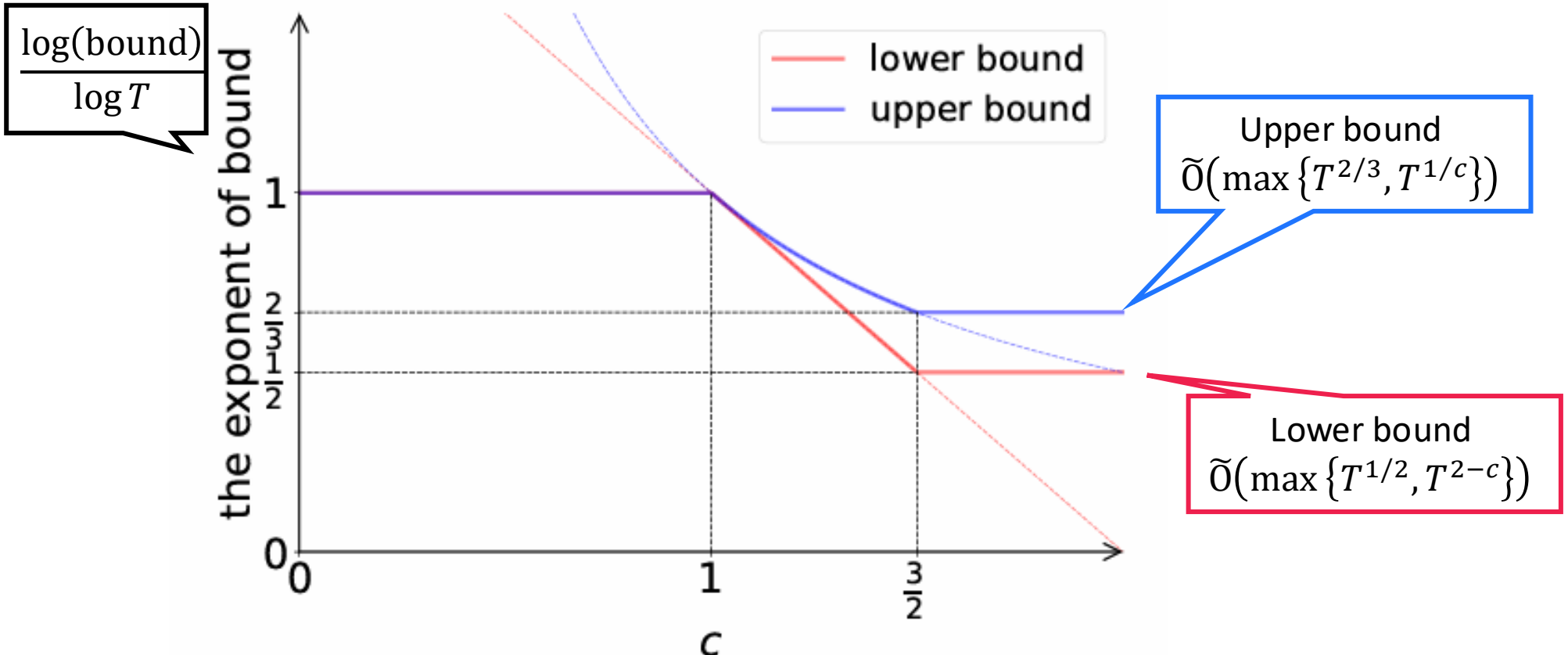
$$R_{\boldsymbol{\mu}}(CRUCB, T) \leq \tilde{O}\left(\max\left(T^{q}\Upsilon_{\boldsymbol{\mu}}(T, q), T^{\frac{2}{3}}\right)\right),$$

where $q \in [0,1]$ *is a fixed constant*

$$\Upsilon_{\boldsymbol{\mu}}(T, q) \coloneqq \sum_{l \in [T-1]} \max_{i \in [K]} \{\gamma_i(l)^q\}$$

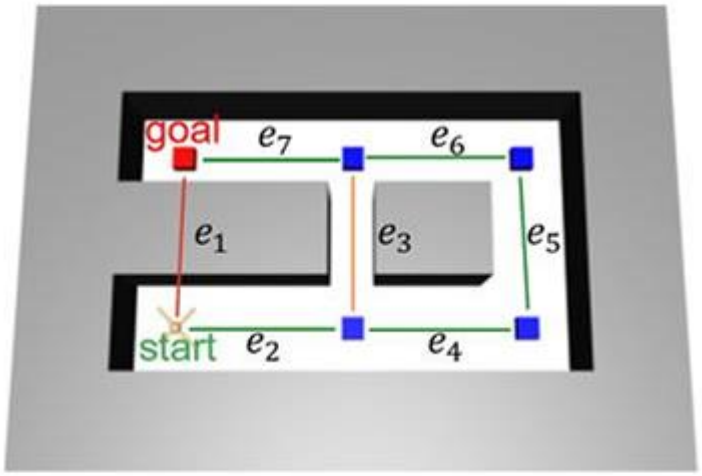
Summary: CRUCB is provably efficient

- Combining Thm 1~3, we show that CRUCB is nearly optimal,
  i.e., **CRUCB is provably efficient**
- We remark that CRUCB **does not require the information of c**

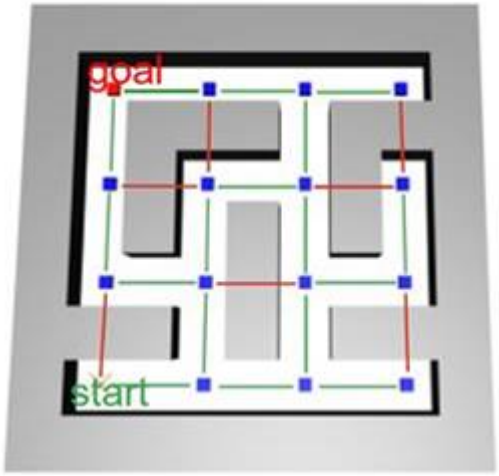


Experiment: Ant-maze

- Ant-maze (Hierarchical reinforcement learning)
  - Objective: finding the shortest path to goal
  - Base arm: each edge in graph
  - Super arm: Simple path from start to goal
  - Reward: Propositional to efficiency

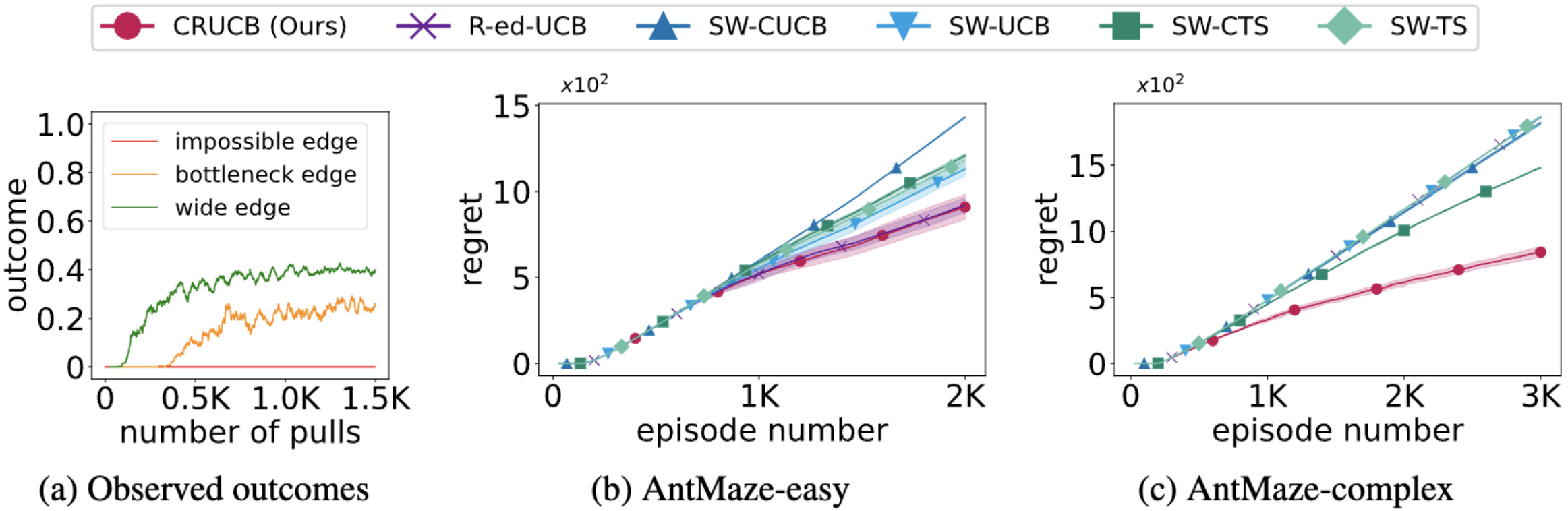


(b) AntMaze-easy



(c) AntMaze-complex

Experiment results

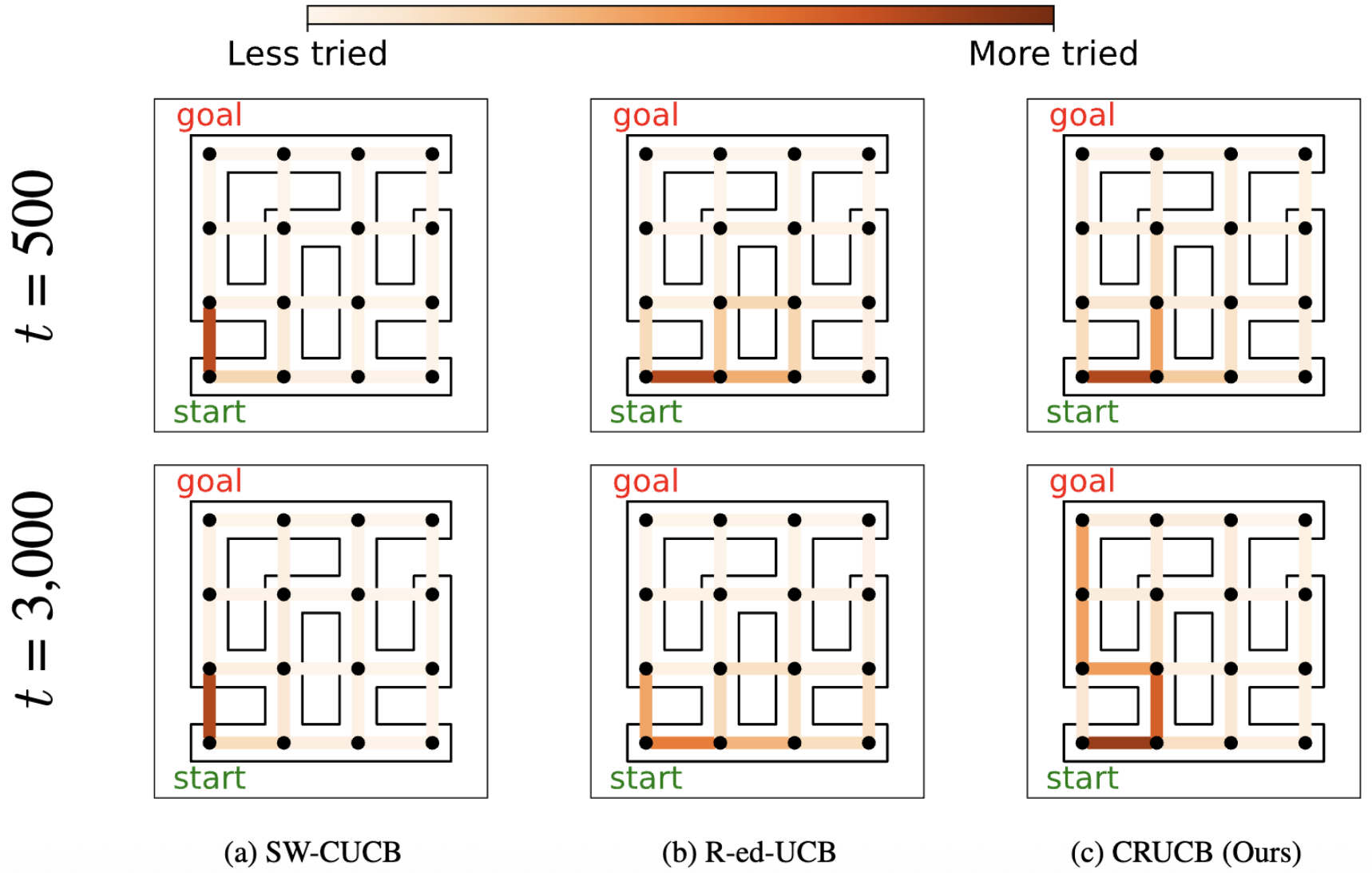


(a) Observed outcomes

(b) AntMaze-easy

(c) AntMaze-complex

Experiment results



Conclusion

- We formulate Combinatorial Rising Bandits (CRB), a new framework that captures rising rewards in combinatorial action selection
- We propose CRUCB, a provably efficient algorithm that does not require knowledge of the curvature constant
- We establish matching upper and lower bounds, showing CRUCB is near-optimal
- Experiments on Ant-maze validate that CRUCB outperforms existing baselines

Thank you for listening!