ICLR

REGRET-GUIDED SEARCH CONTROL FOR EFFICIENT LEARNING IN ALPHAZERO

Yun-Jui Tsai[1,2], Wei-Yu Chen[2,3], Yan-Ru Ju[2], Yu-Hung Chang[2], Ti-Rong Wu[2]

[1] National Yang Ming Chiao Tung University Academia Sinica
[2] Academia Sinica
[3] College of Computing, Georgia Institute of Technology

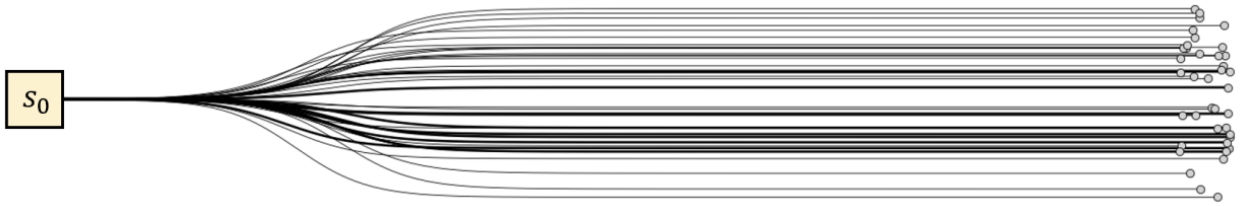
AlphaZero and Human's Learning Efficiency

AlphaZero Learning

- Always restarts from an empty board
- Required large training resources

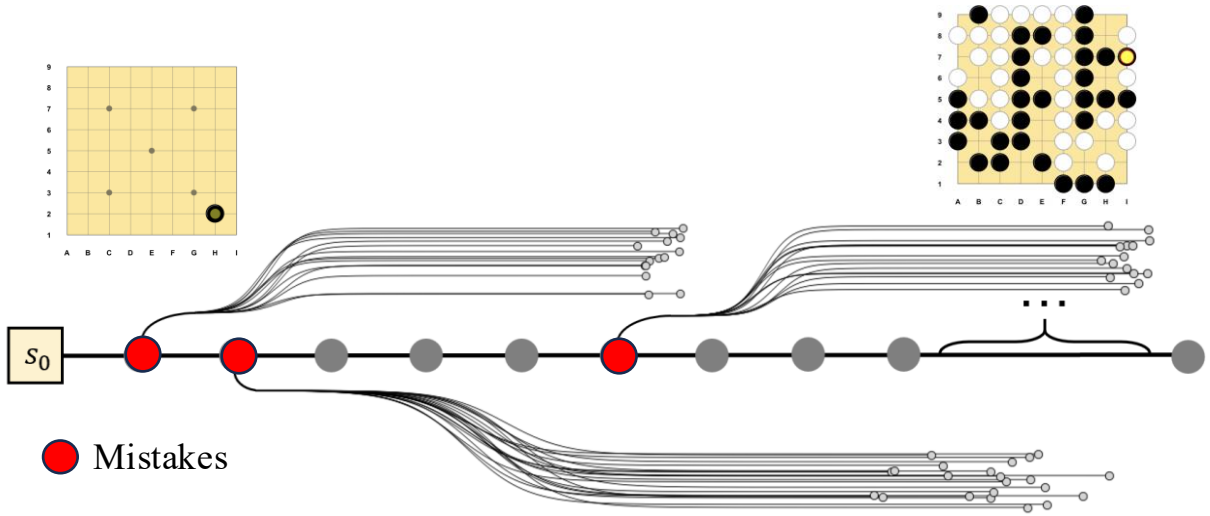
Human Learning

- Review the critical positions repeatedly
- Learn faster under restricted training resources
- A kind of self-play search control



AlphaZero Zero-knowledge Learning

S_0 represents the initial state



Human Learning

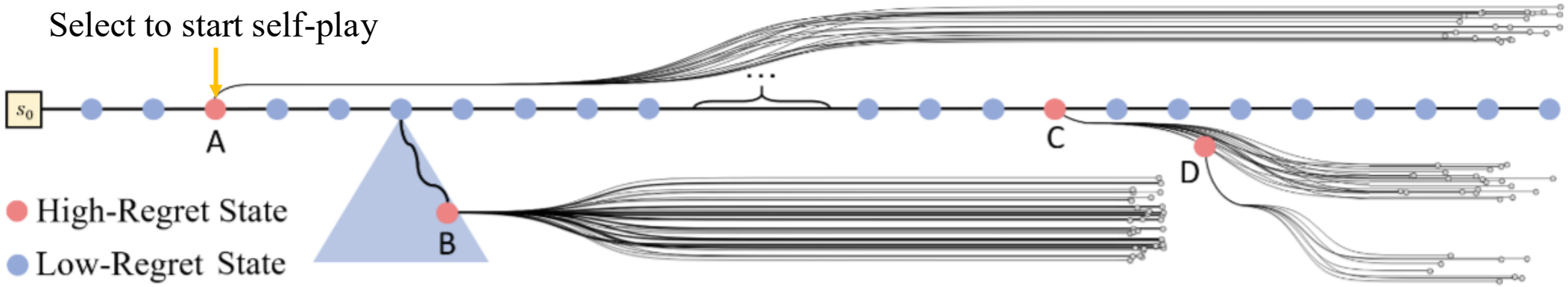
Search Control in AlphaZero

- Search Control: "*The process that selects the starting states and actions for the simulated experiences generated by the model*[1]"
- Previous works for search control in AlphaZero:
  - Starting from random states in the past trajectories
    - Accelerating Self-Play Learning in Go (KataGo)[2]
    - Targeted Search Control in AlphaZero for Effective Policy Improvement (Go-Exploit)[3]
- We proposed RGSC, which extends AlphaZero
  - Automatically identify unfamiliar (high-regret) states for the current model

[1] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. Adaptive Computation and Machine Learning Series. MIT Press, Cambridge, MA, USA, 2 edition, November 2018.
[2] David J. Wu. Accelerating Self-Play Learning in Go. In Proceedings of the AAAI Workshop on Reinforcement Learning in Games, November 2020.
[3] Alexandre Trudeau and Michael Bowling. Targeted Search Control in AlphaZero for Effective Policy Improvement. In Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '23, pp. 842–850, Richland, SC, May 2023. International Foundation for Autonomous Agents and Multiagent Systems.

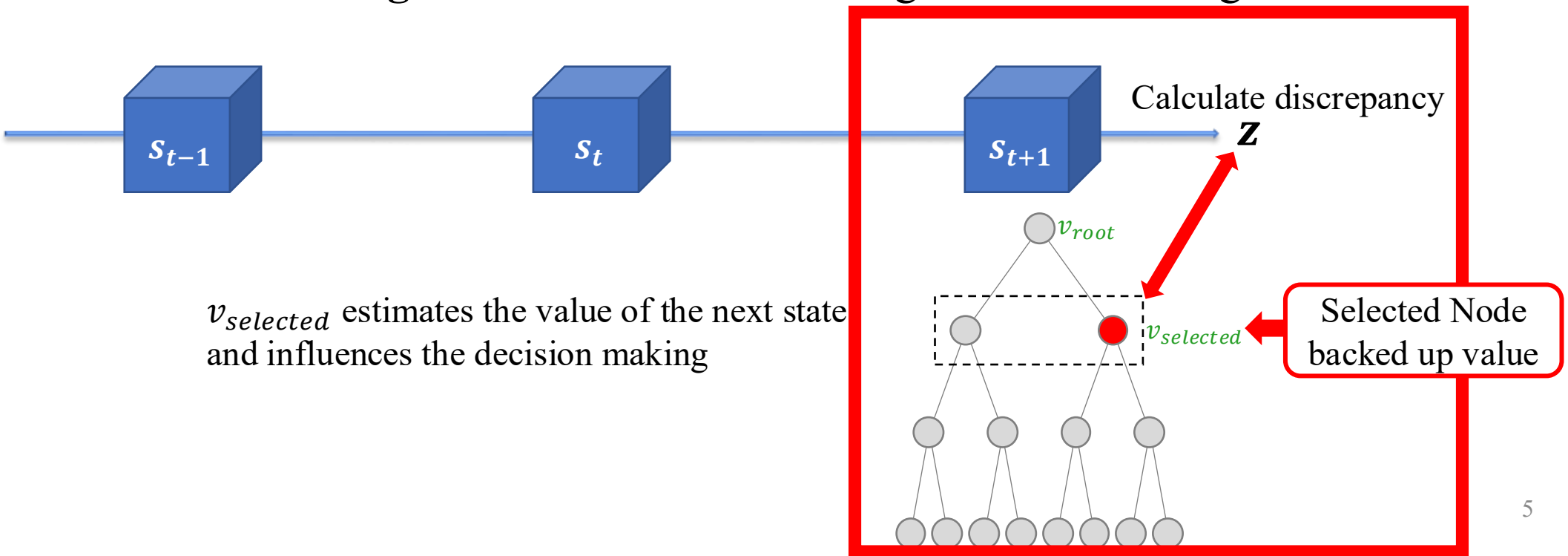
Regret in AlphaZero

- Motivation:
  - Aim to select critical states as starting states, like humans reviewing their mistakes
- Regret is used to determine the critical state
  - Regret approximates the distance between the current policy and the optimal policy



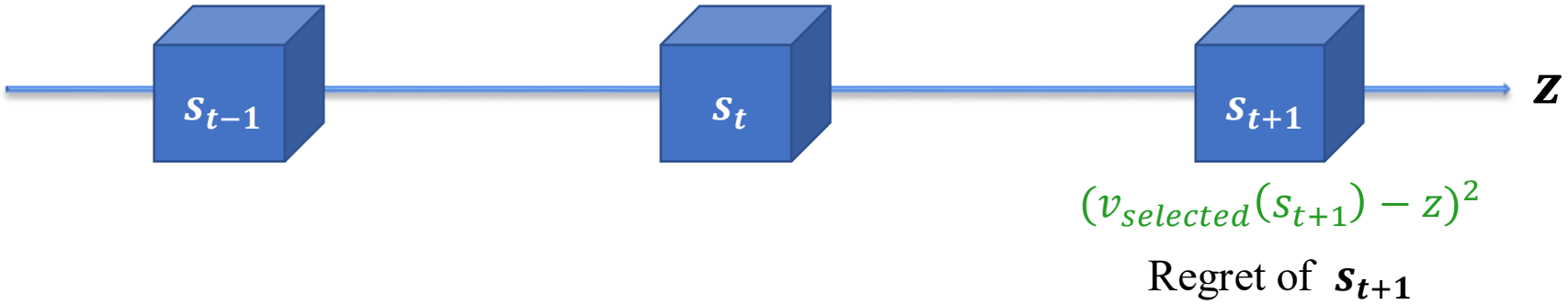
Regret in AlphaZero

- For every single self-play trajectory $s_0, s_1, s_2 \cdots s_T$, we aim to find a state with the largest regret
- In RGSC, we estimate regret by measuring the average discrepancy between the **agent's evaluation** and **the game outcome** when the agent start from a given state



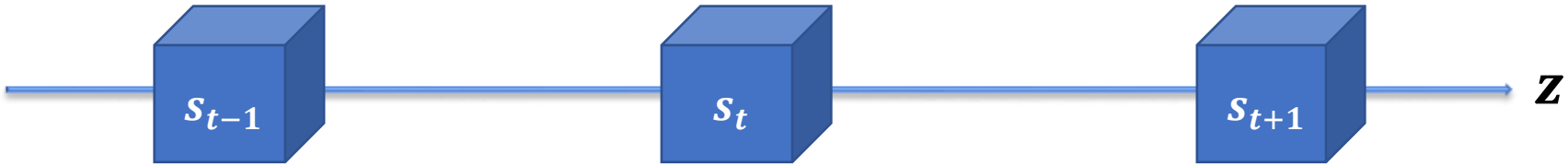
Regret in AlphaZero

- For every single self-play trajectory $s_0, s_1, s_2 \cdots s_T$, we aim to find a state with the largest regret
- In RGSC, we estimate regret by measuring the average discrepancy between the **agent's evaluation** and **the game outcome** when the agent start from a given state



Regret in AlphaZero

- For every single self-play trajectory $s_0, s_1, s_2 \cdots s_T$, we aim to find a state with the largest regret
- In RGSC, we estimate regret by measuring the average discrepancy between the **agent's evaluation** and **the game outcome** when the agent start from a given state

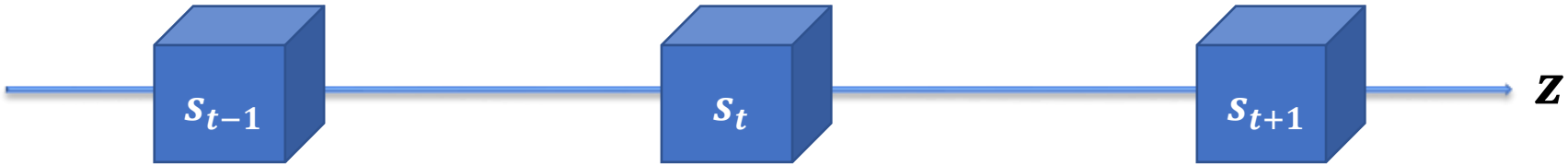


$$\frac{1}{2}\Big\{(v_{selected}(s_{t+1}) - z)^2 + (v_{selected}(s_t) - z)^2$$

Regret of s_t represents the average discrepancy when an agent starts from s_t

Regret in AlphaZero

- For every single self-play trajectory $s_0, s_1, s_2 \cdots s_T$, we aim to find a state with the largest regret
- In RGSC, we estimate regret by measuring the average discrepancy between the **agent's evaluation** and **the game outcome** when the agent start from a given state

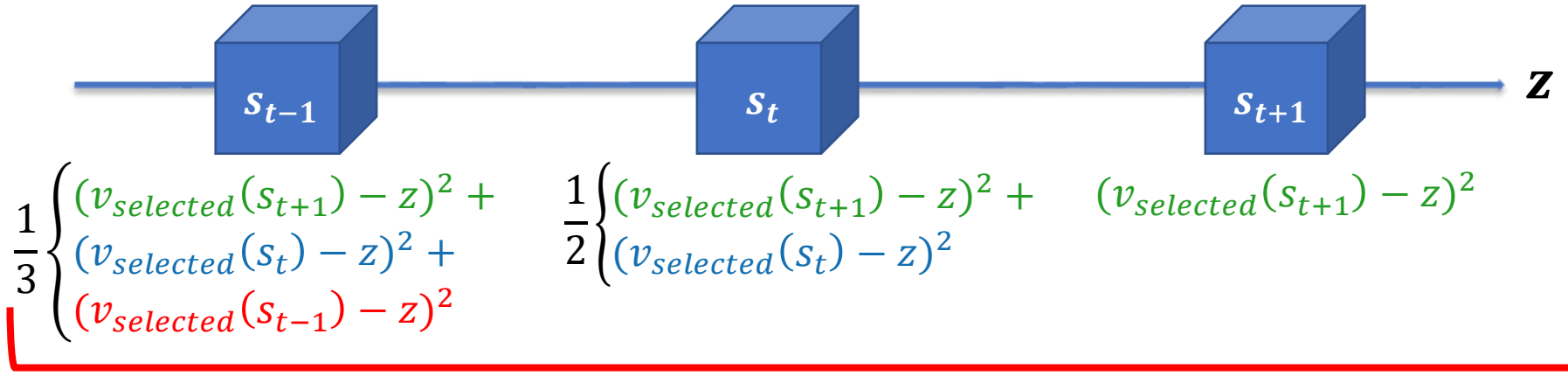


$$\frac{1}{3}\begin{cases}(v_{selected}(s_{t+1}) - z)^2 + \\ (v_{selected}(s_t) - z)^2 + \\ (v_{selected}(s_{t-1}) - z)^2\end{cases}$$

Regret of $\boldsymbol{s_{t-1}}$ represents the average discrepancy when an agent starts from $\boldsymbol{s_{t-1}}$

Regret in AlphaZero

- For every single self-play trajectory $s_0, s_1, s_2 \cdots s_T$, we aim to find a state with the largest regret
- In RGSC, we estimate regret by measuring the average discrepancy between the **agent's evaluation** and **the game outcome** when the agent start from a given state



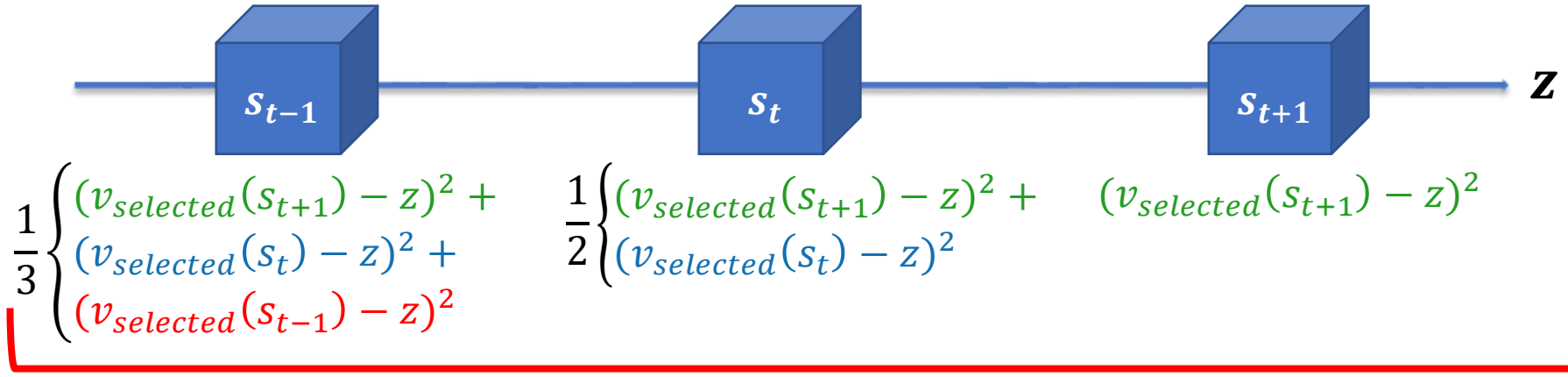
Compare regrets to decide the critical state

Regret in AlphaZero

- We define regret as :

$$\mathcal{R}(s_t) = \frac{1}{T-t}\sum_{i=t}^{T}(v_{selected}(s_i) - z)^2$$

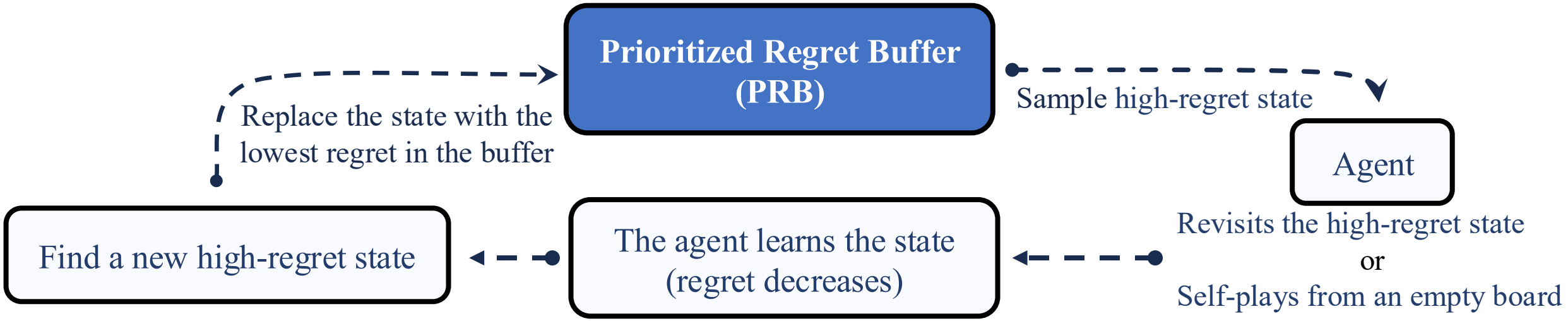
where, $V_{selected}(s_i)$ represents MCTS value of the selected action at state s_i



Compare regrets to decide the critical state

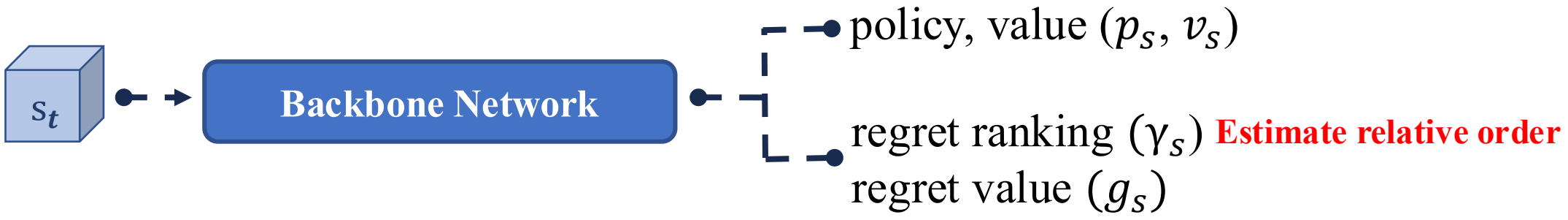
Prioritized Regret Buffer (PRB)

- After getting high-regret states, **Prioritized Regret Buffer (PRB)** is used to mimic how humans learn



Approximate Regret for MCTS Node

- Motivation:
  - Measure and select critical state among MCTS nodes
  - Make the model focus on learning the **relative order** among all nodes
- Regret ranking head predicts the ranking value γ_s for selecting the opening
- Regret value head predicts the regret value g_s for initializing the opening



Regret Ranking Head Training

- According to Exploring Restart Distribution[1], the training objective with search control is

$$L = \sum_{s \in S} \rho(s) \sum_{a \in A} \pi(a|s) \left(q(s,a) - \hat{q}(s,a)\right)^2$$

where $\rho(s)$ is the probability for sampling state s

- Similarly, ranking training objective aims to maximize the regret of sampled states,

$$\mathcal{T}_{rank} = \sum_{s \in S} \rho(s)\, \mathcal{R}(s)$$

- Regret ranking head aims to approximate $\rho(s)$ to get the rank of states

$$\mathcal{T}_{rank} = \sum_{s \in S} softmax(\gamma_s) \mathcal{R}(s)$$

where γ_s is the ranking head output, higher $\mathcal{R}(s)$ implies higher γ_s

Regret Ranking Head Training

- Optimized by using a surrogate objective

$$\mathcal{T}_{rank} = \sum_{s \in S} softmax(\gamma_s)\mathcal{R}(s);\ \tilde{\mathcal{T}}_{rank} = \sum_{s \in S} softmax(\gamma_s)\, exp\{\mathcal{R}(s)\}$$

  - Since $exp\{\mathcal{R}(s)\}$ is a strictly increasing function, the order of γ_s to optimize $\mathcal{T}_{rank}$ equals the order to optimize $\tilde{\mathcal{T}}_{rank}$

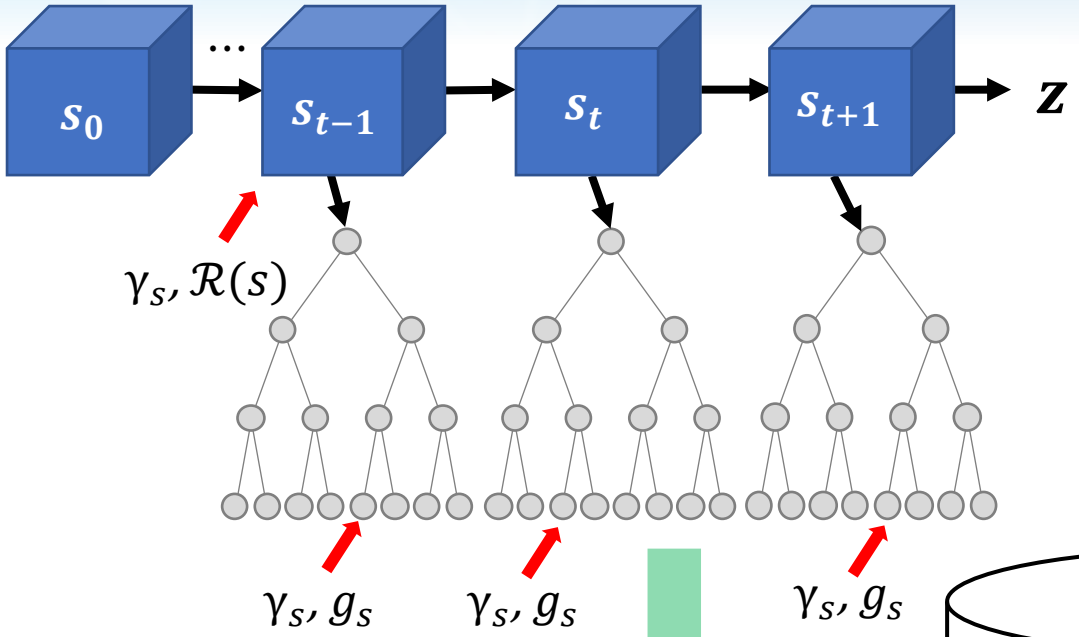
- The loss function of regret ranking head is

$$\mathcal{L}_{rank} = -log\, \tilde{\mathcal{T}}_{rank} = -log \sum_{s \in S} softmax(\gamma_s)\, exp\{\mathcal{R}(s)\}$$
$$= -log \sum_{s \in S} exp(\, log(softmax(\gamma_s))\} exp\{\mathcal{R}(s)\})$$
$$= -log \sum_{s \in S} exp(\, log(softmax(\gamma_s)) + \mathcal{R}(s)\,)$$

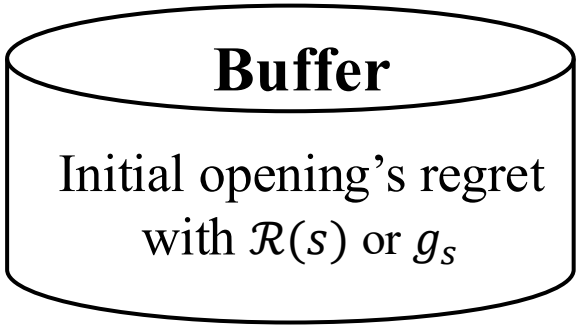
Regret-Guided Search Control (RGSC)

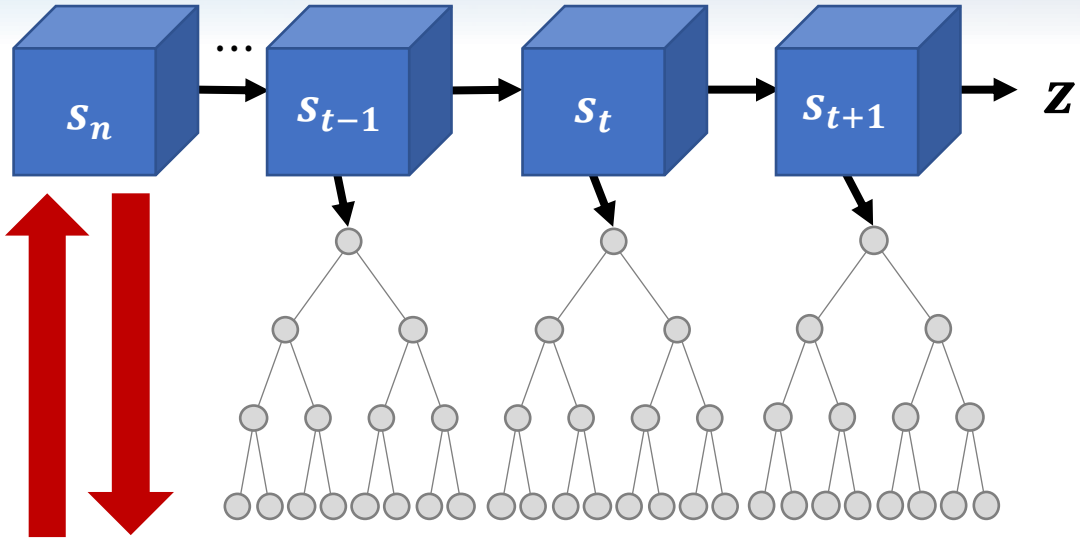
- RGSC consists of two parts:
  - λ probability to start self-play from the opening sampled from PRB
    - Update the opening's regret in PRB
  - $1 - \lambda$ probability to start self-play from the initial state
    - Selecting a new opening into PRB

RGSC Overview



Select a node with the **highest ranking γ_s**





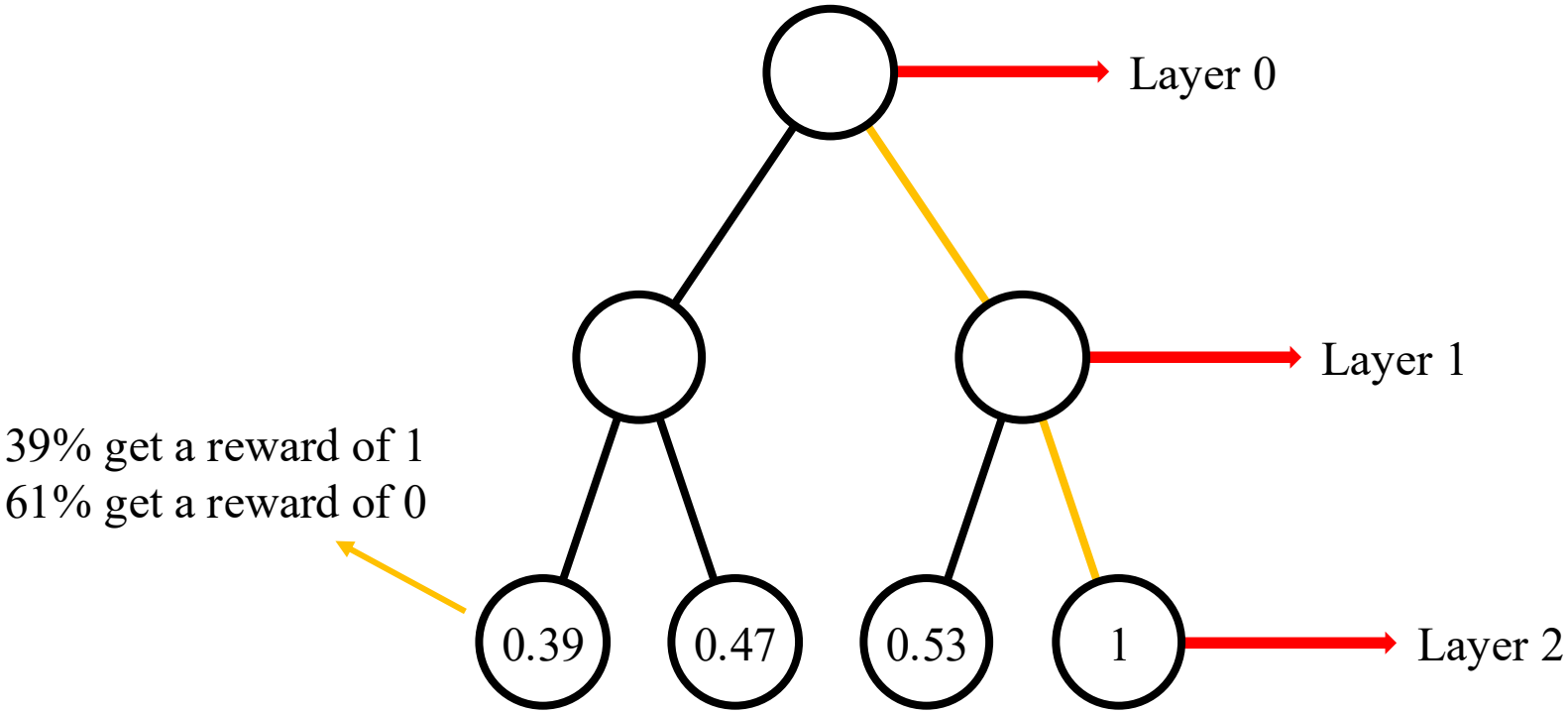
- Start self-play from the sampled opening s_n
- Calculate $\mathcal{R}(s_n)$ and update opening when the game finishes

Update the opening with **EMA**

$$\mathcal{R}^{new}(s_n) \leftarrow (1-\alpha) \times \mathcal{R}^{old}(s_n) + \alpha \times \mathcal{R}(s_n)$$

Toy Example

- Motivation: Prove regret method accelerates Q-learning in a small binary tree
- Goal: Find the optimal path (orange path)

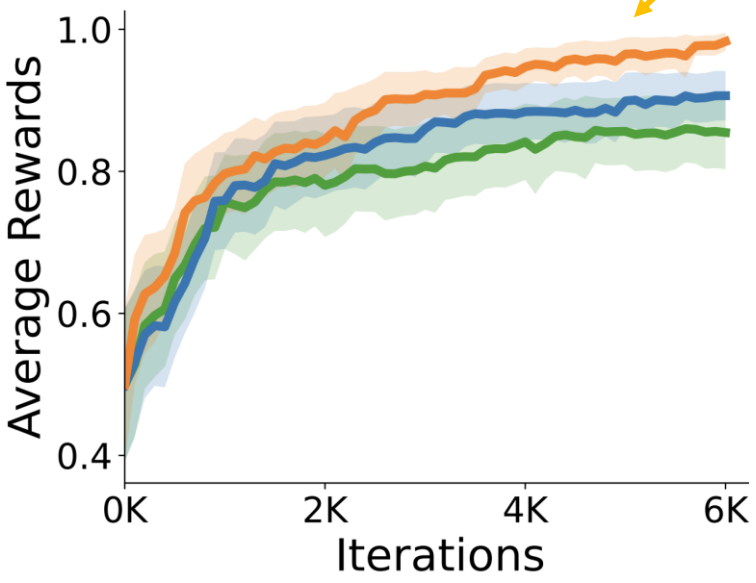


An example for 2-layer binary tree

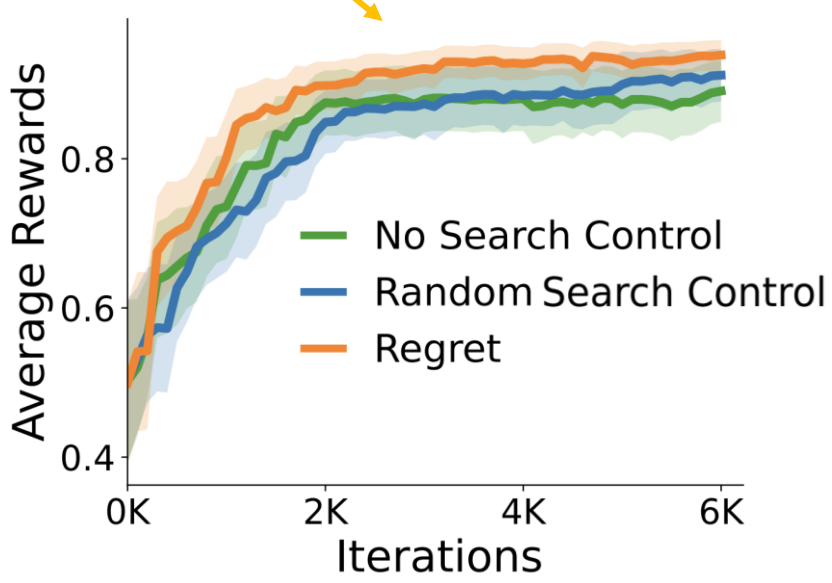
Toy Example

- Average Reward: Regret > Random > No Search Control
- These results show the effectiveness of the regret-guided search control

Higher rewards with the regret method



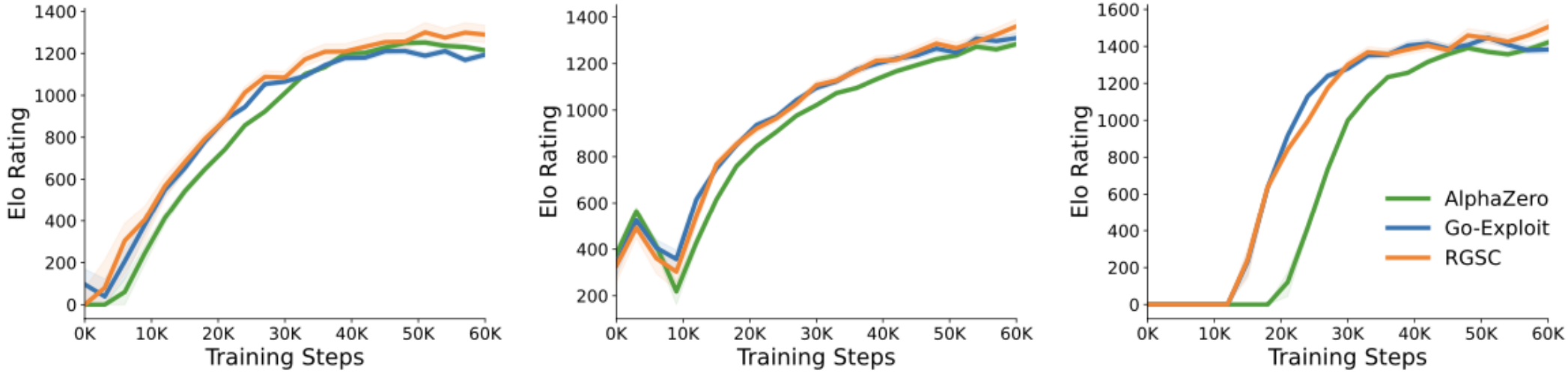
Layer 5 Binary Tree



Layer 6 Binary Tree

RGSC in Board Games

- Although Go-Exploit achieves a higher Elo than AlphaZero in the early stages of training, its advantage diminishes as training converges
- RGSC, however, **maintains its advantage** in the later stages of training



(a) 9x9 Go (b) 10x10 Othello (c) 11x11 Hex

Playing performance of AlphaZero, Go-Exploit, and RGSC

RGSC in Board Games

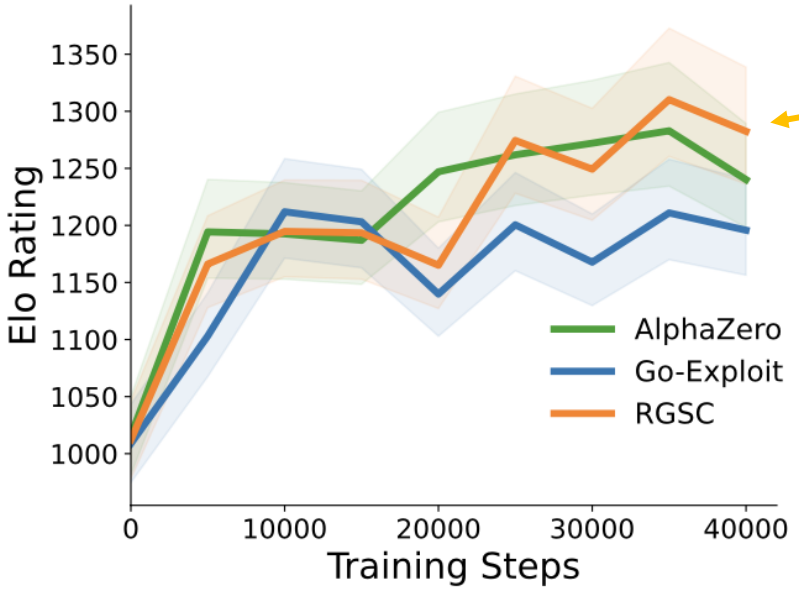
- Although Go-Exploit achieves a higher Elo than AlphaZero in the early stages of training, its advantage diminishes as training converges
- RGSC, however, **maintains its advantage** in the later stages of training

	AlphaZero	**Go-Exploit**	**RGSC**
9x9 Go	45.5% ± 1.5%	49.5% ± 2.0%	**53.6% ± 2.4%**
10x10 Othello	51.7% ± 2.5%	52.9% ± 3.3%	**57.8% ± 3.2%**
11x11 Hex	83.6% ± 1.6%	89.2% ± 1.8%	**91.1% ± 2.0%**

Winrate at 60000 steps against established open-source programs
(KataGo, Ludii alpha-beta search and Mohex)

RGSC on Well-trained Model

- Motivation: Prove RGSC surpasses other methods when training on a well-trained model
- Training on a well-trained model for 40,000 steps
- RGSC achieves a substantially higher win rate, outperforming all baselines



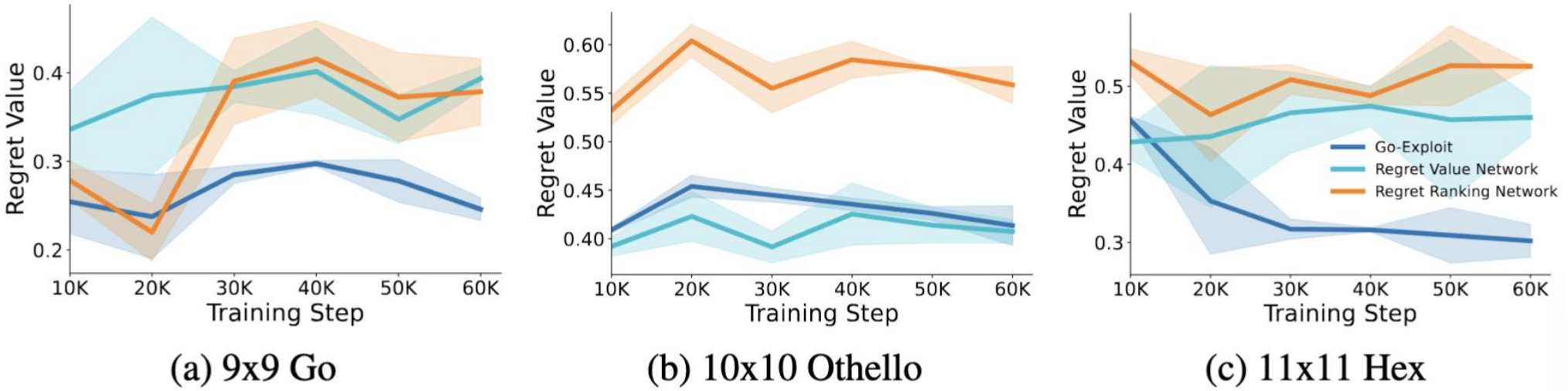
40000 steps

Baseline(0 step)	AlphaZero	Go-Exploit	RGSC
69.3% ± 2.6%	70.2% ± 2.7%	69.2% ± 2.7%	**78.2% ± 2.5%**

Winrate at 40000 steps against KataGo

Comparison between Ranking and Regret

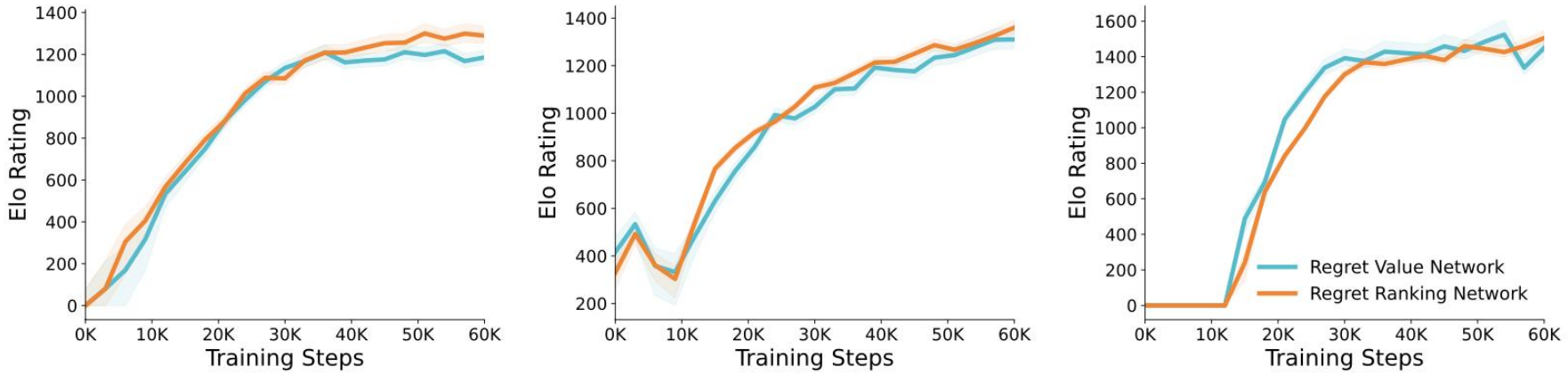
- The regret ranking network consistently selects states with higher true regret than the regret value network



Average regret value of nodes selected by different methods

Comparison between Ranking and Regret

- Directly compare the training performance with regret value network and regret ranking network
- These results demonstrate that the ranking objective improves the quality of selected states by prioritizing those with greater learning potential

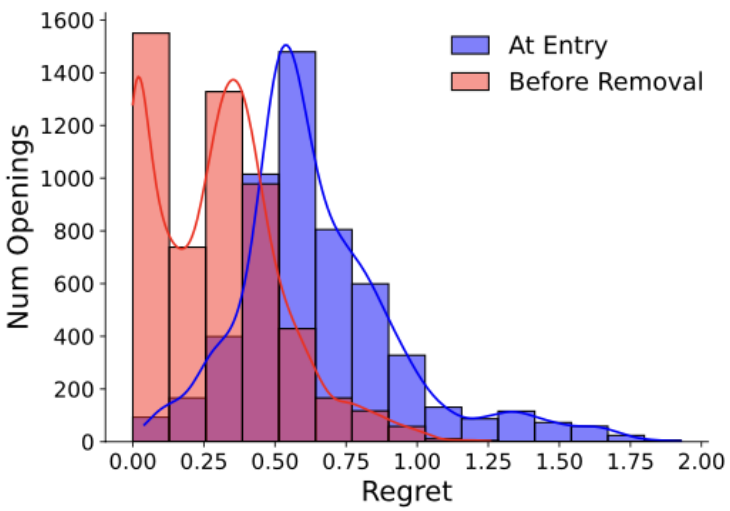


(a) 9x9 Go (b) 10x10 Othello (c) 11x11 Hex

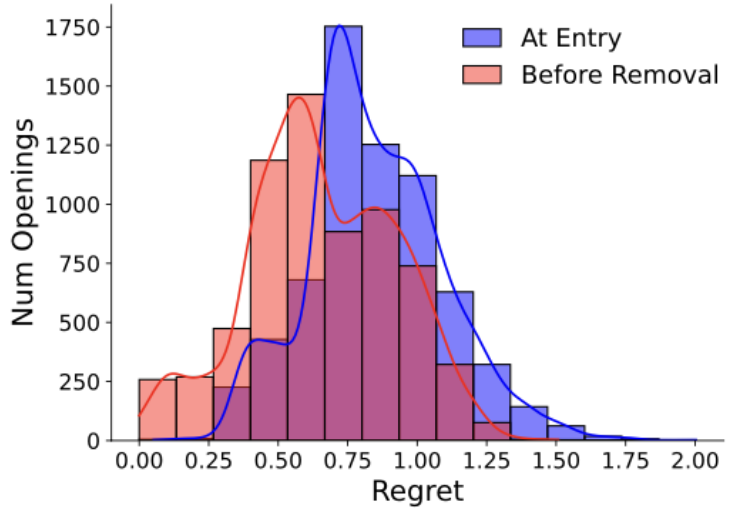
Playing performance of model selecting MCTS node using regret ranking network and regret value network

Regret Decay in Buffer

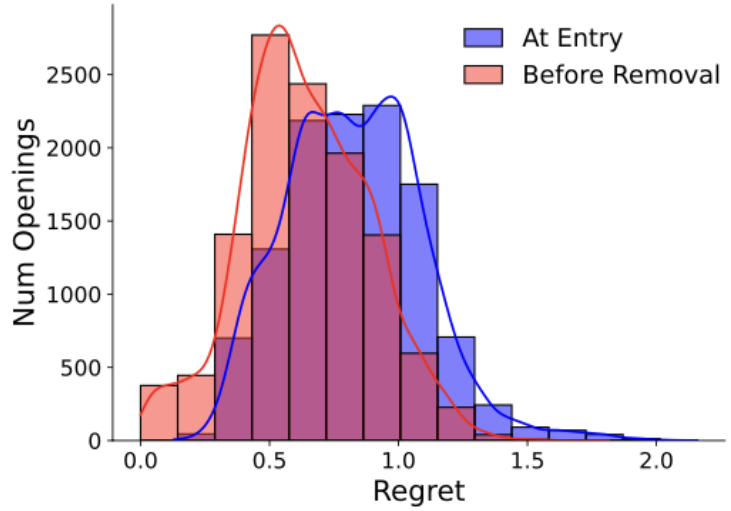
- High-regret states in the buffer gradually decrease during training
  - The model can actually correct its mistakes by repeatedly revisiting those states
- This experiment confirms that states initially associated with high regret are eventually corrected through repeated replay, resulting in reduced regret over time



(a) 9x9 Go



(b) 10x10 Othello



(c) 11x11 Hex

Summary

- RGSC: An extension of AlphaZero that
  - **Identifies high-regret states**
  - Focus on **correcting its most critical mistakes**, mimicking how humans learn
  - Outperform AlphaZero and Go-Exploit when
    - Start training from a random weight
    - Continue training from a well-trained 9×9 Go model weight
- Future work:
  - Extend RGSC to MuZero (already a preliminary experiment)
  - Enhance model explanation according to human preference
- Our results highlight regret-guided search control as a promising direction for **improving the efficiency and robustness** of reinforcement learning

Thank You for Your Attention

Our code and data are available at

https://rlg.iis.sinica.edu.tw/papers/rgsc/