A Batch Sequential Halving Algorithm without Performance Degradation and its Application to Monte Carlo Tree Search

Sotetsu Koyamada

Publication

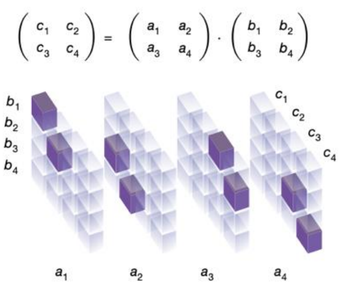
A Batch Sequential Halving Algorithm without Performance Degradation
By *Sotetsu Koyamada*, *Soichiro Nishimori*, and *Shin Ishii*
Reinforcement Learning Journal, vol. 5, 2024, pp. 2218–2232.

Monte Carlo Tree Seach (MCTS)

2000s ~ : Developed in computer Go research

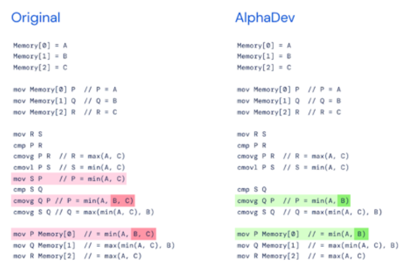
- Kocsis&Szepesvári (2006): UCT
- Coulom (2006) : Monte Carlo evaluation + tree search
- …
- Silver et al., 2016,17,18 : AlphaGo and AlphaZero family

Discovering tensor decomposition algorithm



[Mankowitz et al., 2023]

Discovering sort algorithm



[Fawzi et al., 2022]

Continuous control by Sampled AlphaZero

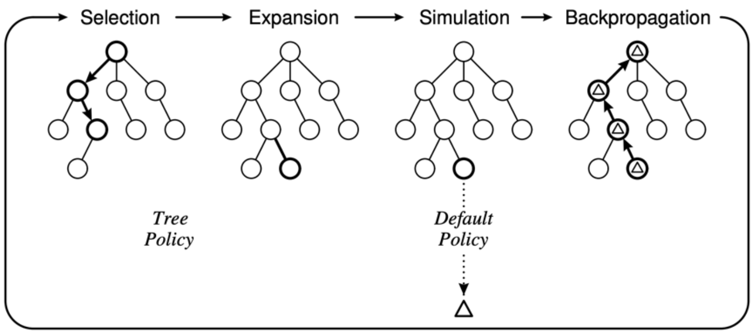
From https://research.google/blog/multi-task-robotic-reinforcement-learning-at-scale/

[Hubert et al., 2021]

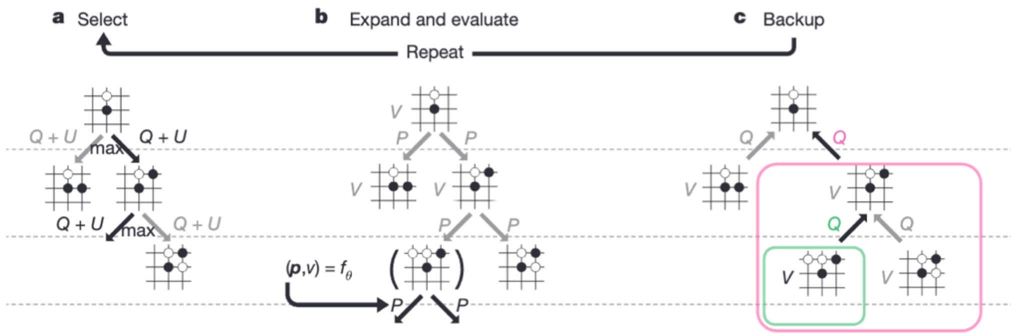
Heruristics but still in the center of planning algorithm

Review: Monte Carlo Tree Search (MCTS)

From Browne et al. (2012)



From Silver et al. (2017)



Evaluation by **rollout**

CPU-intensive

Evaluation by **policy/value net**

GPU-intensive

Is the same algorithm still efficient?

"The Monte-Carlo tree search … is challenging to parallelize"
Hafner et al., 2021 (Dreamer v2)

MCTS is known as **sequential** algorithm --- search tree grows step by step
To obtain 100 additional nodes, 100 NN inference is required 😢

Monte Carlo tree search x Neural Network = 🙅

Our focus: SH in pure exploration problem

Literature

Multi-armed bandit		Tree search
UCB (Auer et al., 2002)	→	**MCTS in AlphaZero**
Sequential Halving (SH; Karning et al., 2013)	→	**MCTS in Gumbel AlphaZero** (Danihelka et al., 2022)

(pure exploration problem)

This study

Step. 1 **Batch SH** → Step. 2 **Batch MCTS w/ SH**

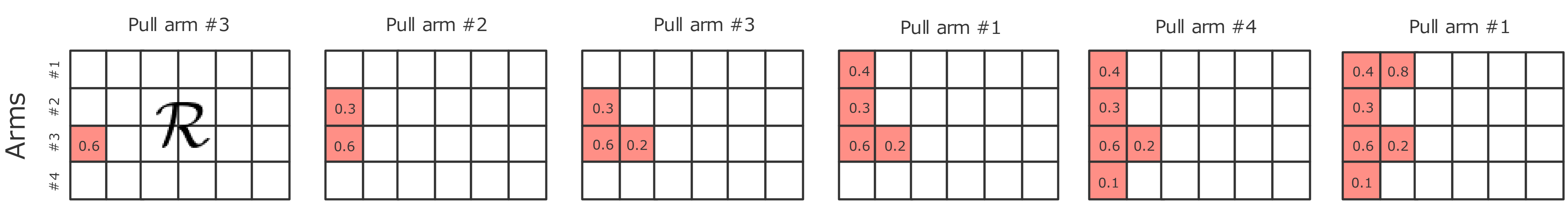
Pure exploration problem and SH

- A variant of multi-armed bandit problem
- Applicable to action selection at root node in MCTS

Pure exploration problem

- #arms: n Reward mean: $1 \geq \mu_1 \geq \mu_2 \geq \ldots \geq \mu_n \geq 0$
- Total budget: T After T arm pulls, select one arm a_T
- Reward matrix: $\mathcal{R} \in [0,1]^{n \times T}$ $\mathcal{R}_{i,j} \in [0,1]$ represents the reward of the j-th pull of arm i (counted independently for each arm)
- Algorithm: $\pi : [0,1]^{n \times T} \to [n]$ ($[n] := \{1, \ldots, n\}$)
- Simple regret:

$$\mathbb{E}_{\mathcal{R}}[\mu_1 - \mu_{a_T}]$$



$n = 4, T = 6$

SH: Sequential Halving [Karnin et al., 2013]

Algorithm 1 SH: Sequential Halving (Karnin et al., 2013)

---

input number of arms: n, budget: T
initialize best arm candidates $\mathcal{S}_0 := [n]$
for round $r = 0, \ldots, \lceil \log_2 n \rceil - 1$ **do**
    pull each arm $a \in \mathcal{S}_r$ for $J_r = \left\lfloor \frac{T}{|\mathcal{S}_r| \lceil \log_2 n \rceil} \right\rfloor$ times
    $\mathcal{S}_{r+1} \leftarrow$ top-$\lceil |\mathcal{S}_r|/2 \rceil$ arms in $\mathcal{S}_r$ w.r.t. the empirical rewards
return the only arm in $\mathcal{S}_{\lceil \log_2 n \rceil}$

Initialize the best arm candidates with n arms

Pull equally the remaining arms

Remove the half of arms

Round

After $\log_2$ n rounds, the only one arm is select

✓ Simple

✓ No task-dependent hyperparameters

✓ Efficient (simple regret is optimal except log factor [Zhao et al., 2023]) $\tilde{O}(\sqrt{n/T})$

Applications

- Hyperparameter search [Jamieson&Talwalkar, 2016]
- State-of-the-art AlphaZero/MuZero family [Danihelka et al., 2022]

Pure exploration in fixed-batch setting

Instead of T sequneital pulls, we simultaneously pull b arms for B times

Pros: Computational efficiency

b: batch size
B: batch budget

Especially with GPU accelerators

Cons: Delayed feedback & Reduced adaptability

The performance may degrade due to reward observation delay

Example

Let's consider two scenarios

Both total budgets = 100K

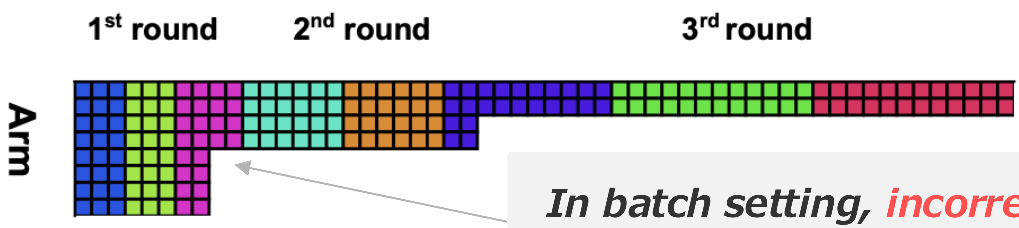
- (A) **100K** sequentail pulls: update poilcy 100K times
- (B) **20** batch pulls with batch size **5K** (B=20, b=5K): update policy only 20 times

(A) Performs better in general

Two batched variants of SH

BSH (Breadth-first SH)

selects arms as equally as possible



The **3rd batch pull** spans two rounds and the arm promotion is determined based solely on the completion of 6 out of 8 pulls.

In batch setting, incorrect arm promotion may happen because the arm promotion is determined only with uncompleted pulls

ASH (Advance-first SH)

defers arm promotion as much as possible

The **3rd batch pull** selects the arm to be promoted from among those that have completed the pulling

The same color indicates the same batch pull — For example, in **the first batch pull (blue),** BSH pulls each of the 8 arms 3 times, while ASH pulls 3 arms 8 times each.

Equivalence of SH and ASH: No wrong promotion occurs

Theorem 1 *Given a stochastic bandit problem with* $n \geq 2$ *arms, let* $b \geq 2$ *be the batch size and* B *be the batch budget satisfying* $B \geq \max\{4, n\}\lceil \log_2 n \rceil$. *Then, the ASH algorithm is algorithmically equivalent to the SH algorithm with the same total budget* $T = b \times B$ *— the mapping* π_{ASH} *is identical to* π_{SH}.

With the same total budget（$T = b \times B$）, when the condition

$$B \geq \max\{4, \tfrac{n}{b}\}\lceil \log_2 n \rceil$$

T: total budget
b: batch size
B: batch budget

As long as the batch budget B is not extremely small

holds，SH and ASH select the same arm

例

When $n = 32$

Note: Both have the same total budget 100K

- (A) **100K** sequential pulls (T=100K)
- (B) **20** batch pulls with batch size **5K** (B=20, b=5K)

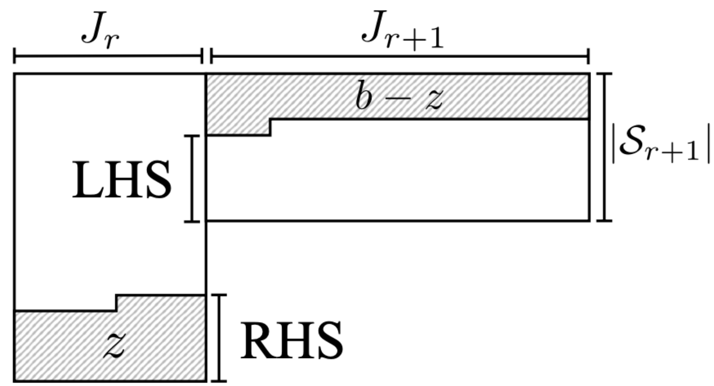
As the condition suffices, the selected arms are identical

Proof overview

When the batch spawns two rounds, prove that the wrong arm promotion does not occur

For any z, assume the batch with size b is splitted into z and b-z

$$\underbrace{|\mathcal{S}_{r+1}| - \left\lceil \frac{b-z}{J_{r+1}} \right\rceil}_{\substack{\text{the number of arms promoting to the} \\ \text{subsequent round post-batch pull}}} \geq \underbrace{\left\lceil \frac{z}{J_r} \right\rceil}_{\substack{\text{the arms pending completion of their} \\ \text{pulls at the batch pull juncture}}}$$

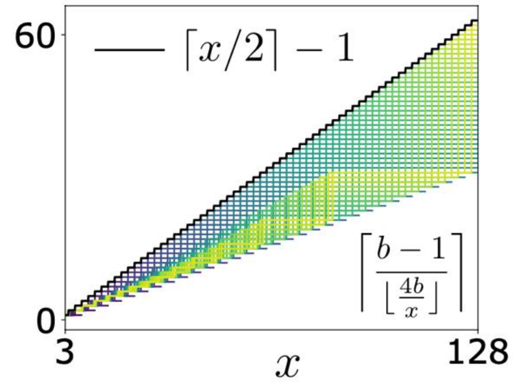


Even in the worst,all correct arms can promote

J_r : # of each arm pulls at round r
S_r : Arms living at round r

Proving the following inequality is enough (it holds as the right visualization)

$$\left\lceil \frac{x}{2} \right\rceil - 1 \geq \left\lceil \frac{b-1}{\lfloor 4b/x \rfloor} \right\rceil$$



Discussion on the condition $B \geq \max\{4, \frac{n}{b}\}\lceil \log_2 n \rceil$

(C1) $B \geq \frac{n}{b}\lceil \log_2 n \rceil$ かつ (C2) $B \geq 4\lceil \log_2 n \rceil$

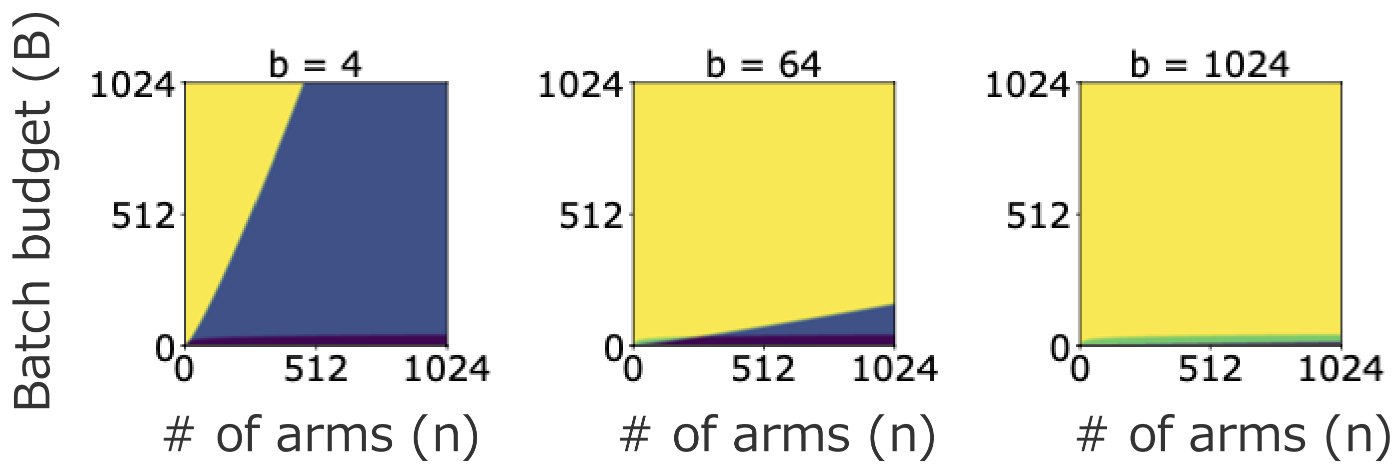
Required to execute SH itself

(all arms should be pulled at least once)

Necessary fo SH and ASH equivalence

Note: C2 is tight

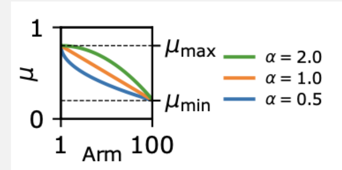
C1 is dominant; C2 is not problematic



Both (C1) and (C2) hold (i.e., ASH is equivalent to SH).
Only (C1) holds (i.e., SH is executable but ASH may not be equivalent to SH).
Only (C2) holds (i.e., SH is not executable).
Neither (C1) nor (C2) holds.

Empirical validation

Setup



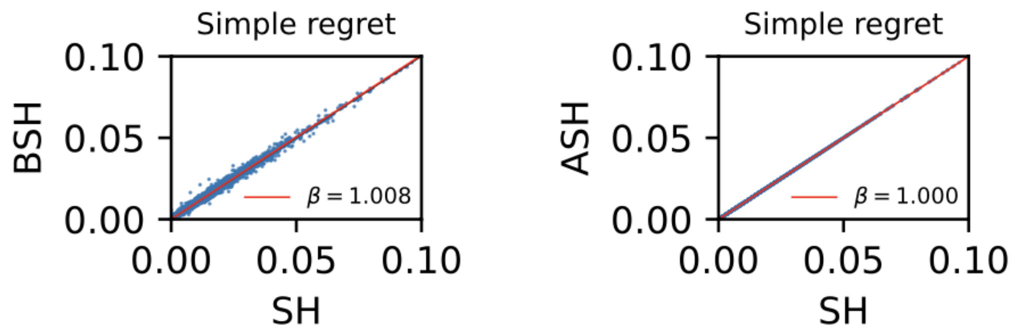
- 10K synthetic stochastic bandit problem instances
- Reward gap $\Delta_a := \mu_1 - \mu_a$ follows $\Delta_a \propto (n/a)^\alpha$ [Zhao et al., 2023]
- For each instance, applied SH and ASH with 100 different random seeds

When the condition holds, we confirmed that the selected arms of ASH and SH are identical in all 10K instances and 100 seeds

✓ Claim supported

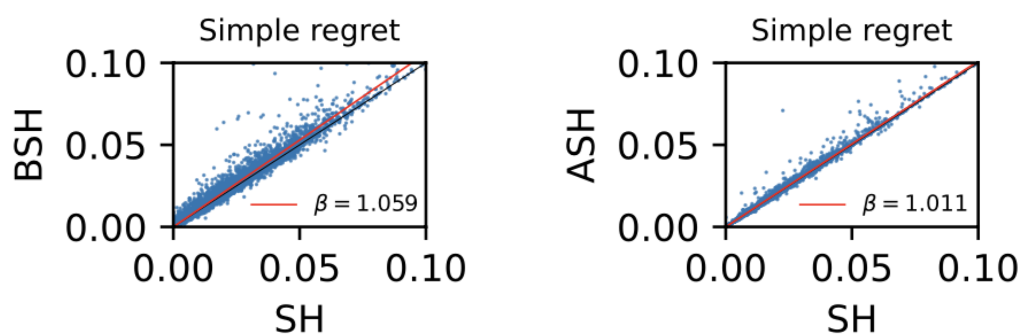
Empirical validation

when batch budget is large: $B \geq 4\lceil \log_2 n \rceil$



β :fitted slope

when batch budget is large: $B < 4\lceil \log_2 n \rceil$

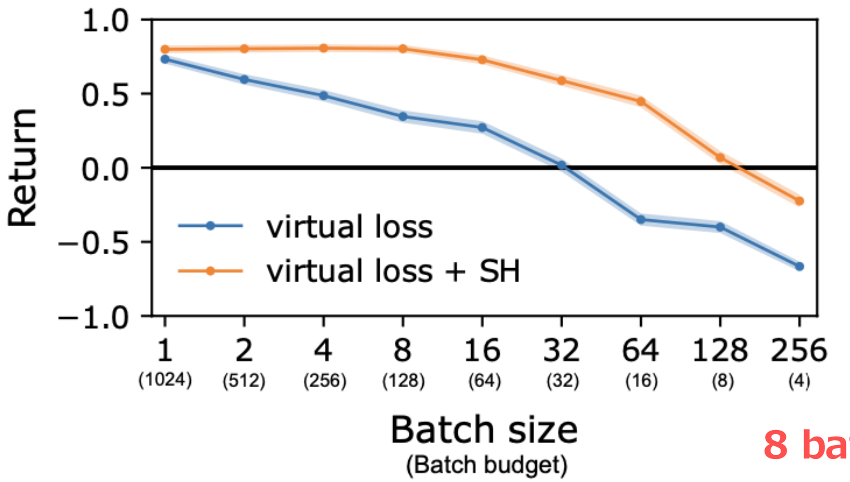


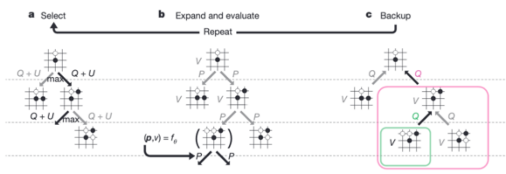
Application to MCTS in 9x9 Go

Example scenario

- Real-time evaluation (match) against top-human player with time limit (e.g., 5min per move).
- NN throughput determines how many times NN inference can be executed (i.e., batch budget).
- We want to select the best action in the given number of NN inferences.

Total budget T = 1024 (fixed)





From Silver et al., 2017

9x9 Go

vs budget = 100 (sequential)

8 batch budget = 100 total budget (sequential)

Virtual loss [Chaslot et al., 2008] Current de-facto parallelization method

Virtual loss + SH Use batch SH at root node

Note: theoretical guarantee does not hold

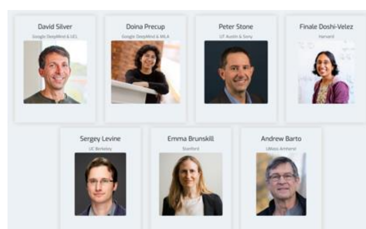
Conclusion

- Overall, we demonstrated the robust nature of SH in the fixed-size batch setting in pure exploration problem
- Clarified the condition where SH and ASH are algorithmically equivalent
  - E.g., ASH can match SH's choice with 100K sequential pulls using just 20 batch pulls, each of size 5K when n = 32
- Demonstrated the combination of MCTS and SH can provide the robust batch MCTS algorithm
- Future work: efficient parallelization for internal nodes?

Publication

A Batch Sequential Halving Algorithm without Performance Degradation
By *Sotetsu Koyamada*, *Soichiro Nishimori*, and *Shin Ishii*
Reinforcement Learning Journal, vol. 5, 2024, pp. 2218–2232.

Reinforcement Learning Conference



RLC2025 @ U. Alberta!