Adaptive Experimentation at Scale

Hongseok Namkoong

Decision, Risk, and Operations Division
Columbia Business School
namkoong@gsb.columbia.edu

This work was led by **Ethan Che**

ethche.github.io

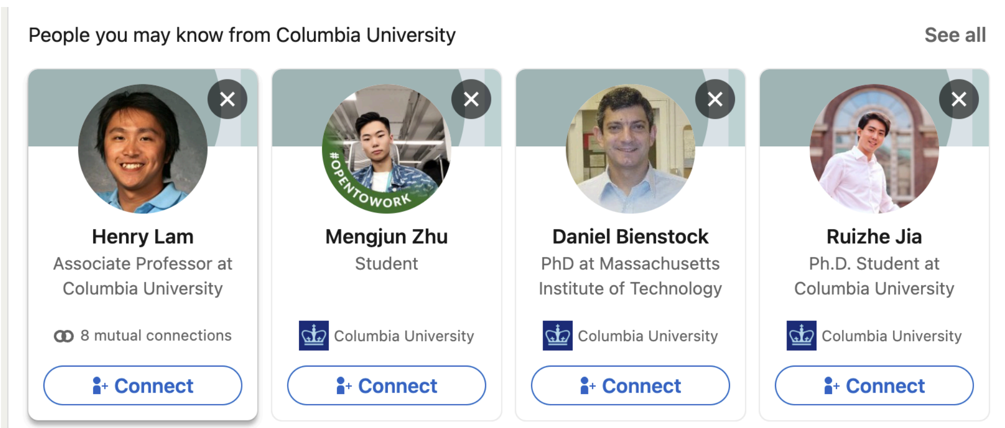
Adaptive Experimentation at Scale: A Computational Framework for Flexible Batches

Arxiv	arxiv.org/abs/2303.11582
Interactive plots	aes-batch.streamlit.app

Motivation

Experimentation (prediction ⇒ decision)

- Imagine a ML engineer building a recommendation system



Configuration 1 2 … K

Which of these help users grow their professional network the best?

- Underpowered: quality of service improvement at most 2%
  - Business impact can nevertheless be big!

Experimentation

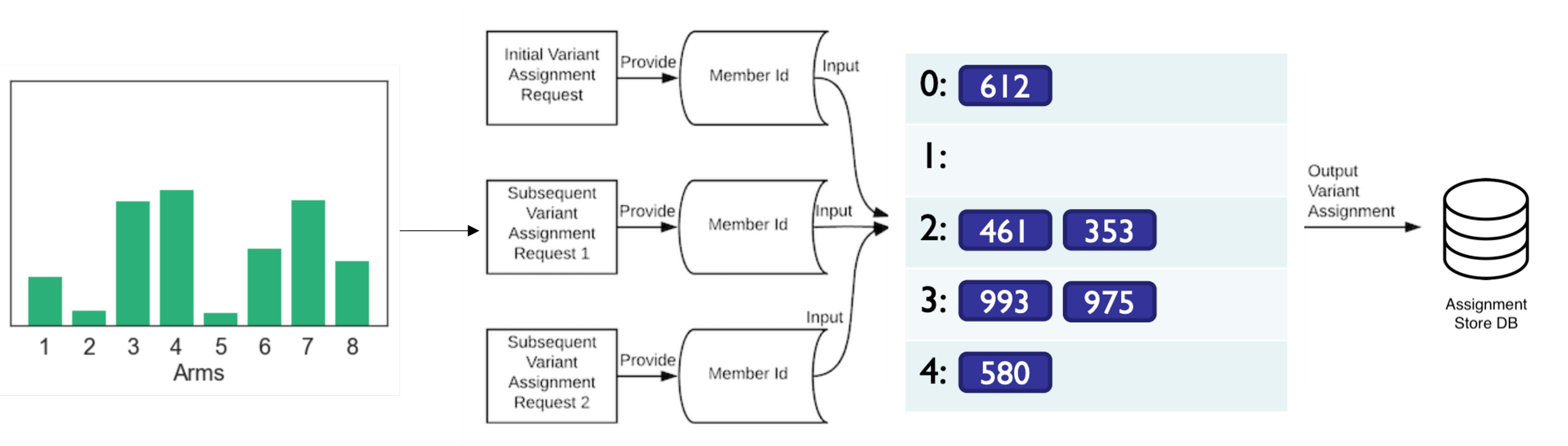
- Foundation of scientific decision-making
  - medical treatments, economic policy, product & engineering innovations
- Typically expensive or risky: cost of collecting data poses operational constraint
- **Statistical power** is of fundamental concern

Adaptivity

- Adaptive allocation of measurement effort can improve power
  - Vast literature: Thompson ('33), Chernoff ('59), Robbins & Lai ('52, '85) + 1000s others
- Assumes unit-level continual reallocation
- Algorithmic design largely guided by theory; "operational constraints" unmodeled
  - Guarantees hold as # reallocation epochs $T \to \infty$
  - Changes to the objective requires ad hoc changes to algo

Adaptivity

- Reallocating measurement costly in practice
  - Delayed feedback, engineering & organizational challenges
  - Latency -> offline computation of sampling probabilities
- No adaptivity in practice; ***at most few, large batches***



Overview

Computation over theory

- Optimize "constants": tailored to small # of reallocations
  - instance-specific signal-to-noise ratio
- Algorithmic design guided by modern computational tools
  - ML + optimization; handle multiple objectives flexibly
  - Policies trained via differentiable programming
- Must handle batch sizes flexibly
  - Cannot resolve if batch size changes

Goal

- Optimize "constants": tailored to small # of reallocations
  - instance-specific signal-to-noise ratio

A/B test:
no adaptivity

Bandits:
fully sequential

Present work

disproves conventional
wisdom that batching
complicates algo design

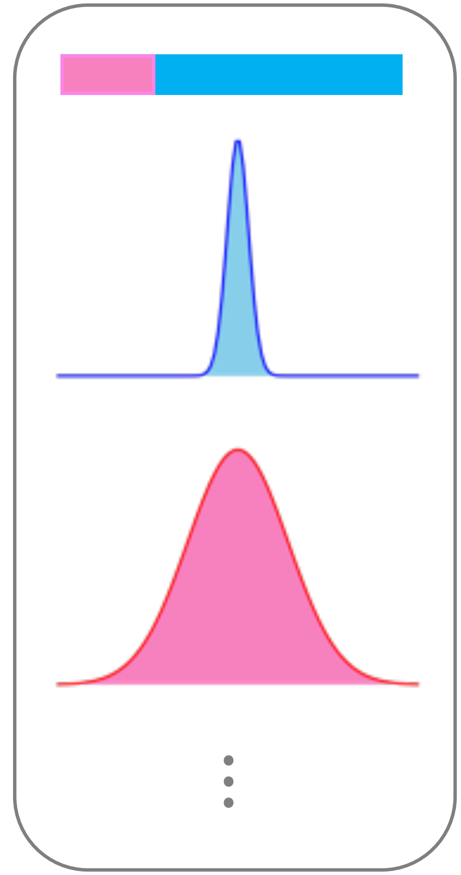
Batched bandits

Perchet+16, Jun+16, Agarwal+17, Gao+19,
Esfandiari+21, Kalkanli+21, Karbasi+21

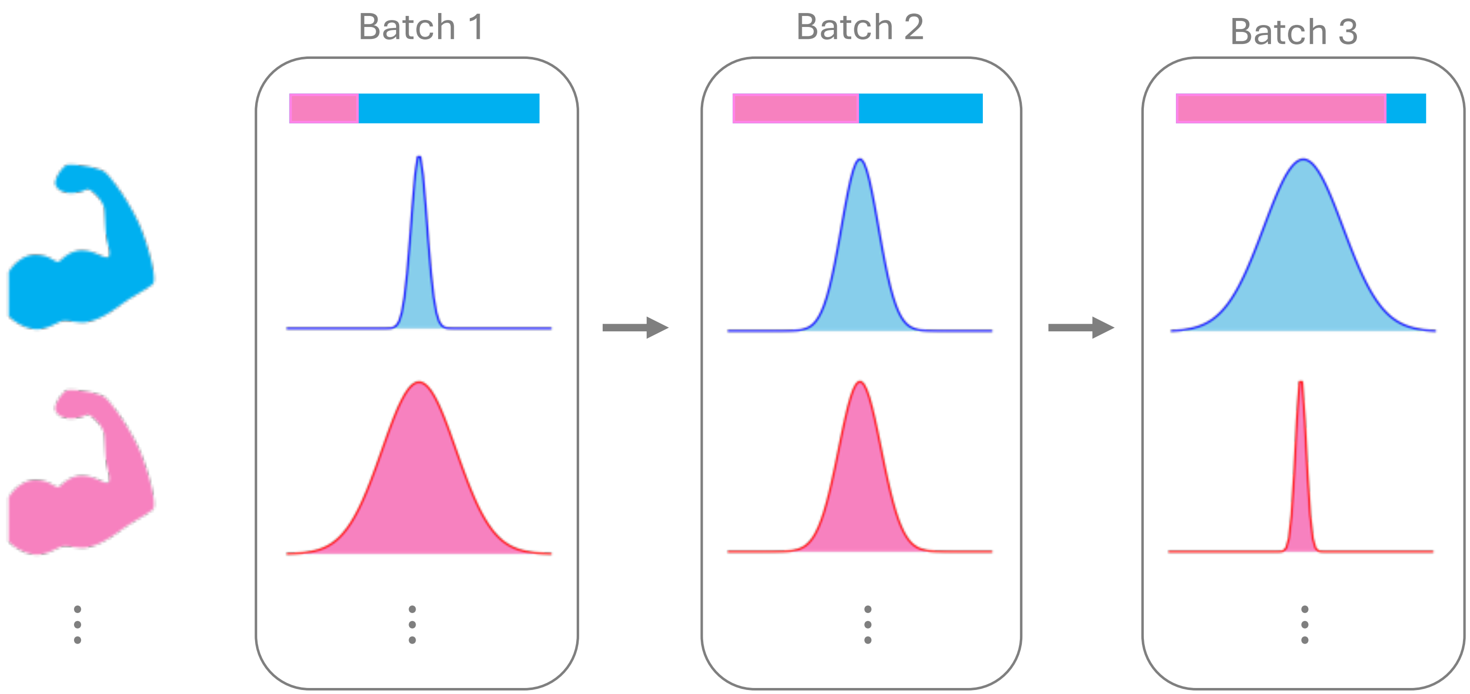
Gaussian approximations

Sample mean in a batch ~ Gaussian

- Allocation controls the effective sample size
  - Gaussian is skinny if the arm is sampled more
- Normal approximations, universal in inference, is also useful for design of adaptive algorithms



Gaussian sequential experiment



Sequence of Gaussian observations gives a tractable model for dynamic programming (DP)

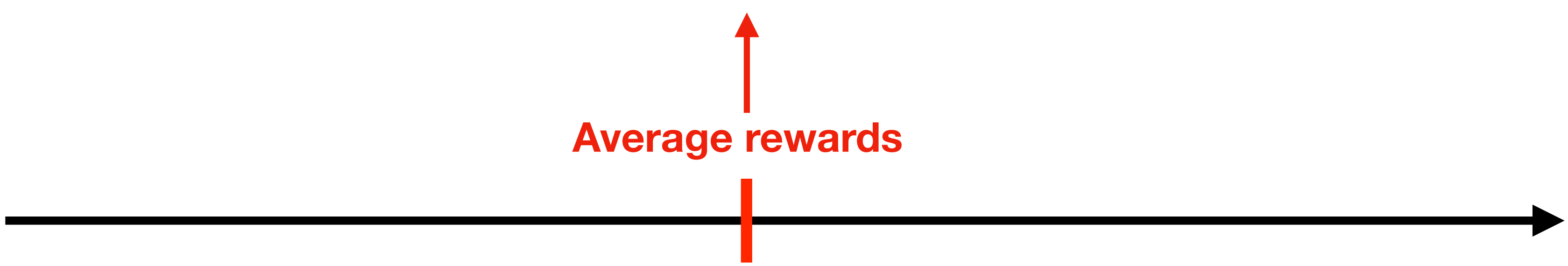
Formulation

Typically in practice, variance ~ 100K × mean reward
with large batches n ~ 100K

Scaling average rewards

- Model underpowered experiments by scaling **average rewards** with batch size n

Reward at arm a: $$R_a = \frac{h_a}{\sqrt{n}} + \varepsilon_a \quad \text{where} \quad \mathrm{Var}(\varepsilon_a) = s_a^2$$



Impossible $\ll$ Admissible $n^{-1/2}$ $\ll$ Trivial

Scaling average rewards

- Model underpowered experiments by scaling **average rewards** with batch size n

Reward at arm a: $R_a = \frac{h_a}{\sqrt{n}} + \varepsilon_a$ where $\text{Var}(\varepsilon_a) = s_a^2$

$\bar{R}_a$: sample mean for arm a, π_a : allocation/fraction

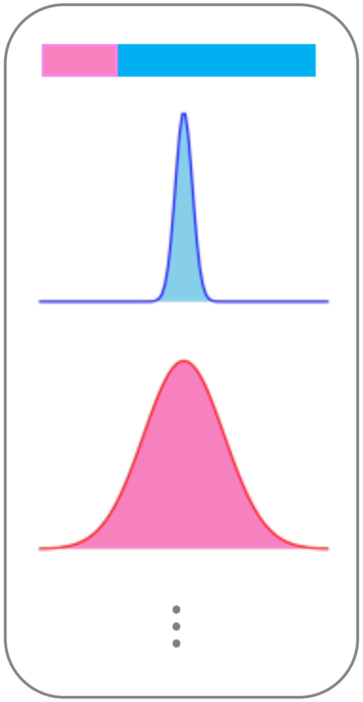
$$\sqrt{n} \cdot \bar{R}_a \sim N(\pi_a h_a, \pi_a s_a^2)$$

Gaussian sequential experiment

- For each arm a

$$\sqrt{n} \cdot \bar{R}_a \sim N(\pi_a h_a, \pi_a s_a^2)$$

- Each batch is an approximate Gaussian draw
  - Each "observation" provides info on average rewards h_a
  - Allocation π_a controls the effective sample size



Our scaling is related to diffusion limits for fully sequential problems, e.g., Wager & Xu (2021), Fan & Glynn (2021)

Local asymptotic normality

- Using successive normal approximations for each batch,

$$G_t \mid G_{0:t-1} \sim N\left(\pi_t \cdot h, \text{diag}(\pi_t \cdot s^2)\right)$$

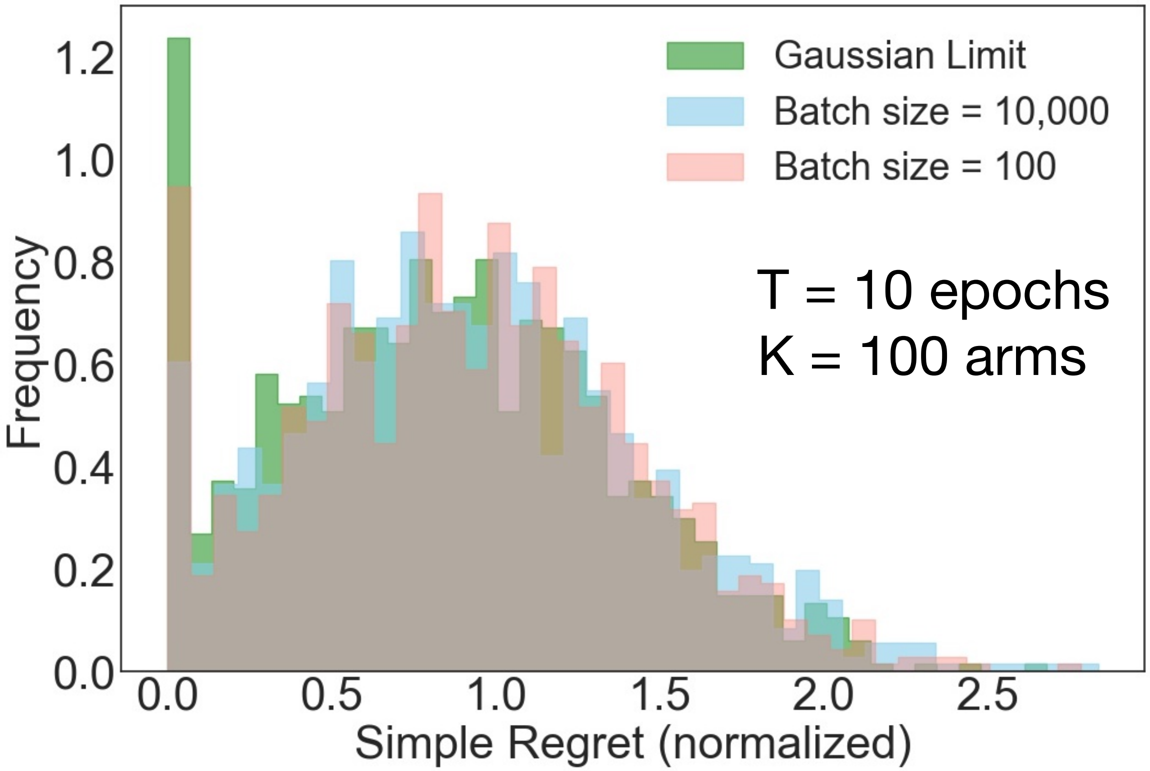
Theorem (Che & N. '23) If allocation π's only depends on batch means continuously, then

$$\left(\sqrt{n}\bar{R}_0, \ldots, \sqrt{n}\bar{R}_{T-1}\right) \Rightarrow (G_0, \ldots, G_{T-1})$$

We don't impose any assumption on the magnitude of π_t

Empirical validity

$$\max_a h_a - h_{\hat{a}} \quad \text{where} \quad \hat{a} \sim \pi_T\left(\sqrt{n}\bar{R}_{0:T-1}\right)$$



Normal approximation reasonable even for small batch sizes!

Convergence rate

Corollary Let L be the Lipschitz constant of allocations π_t .

Metrize weak convergence using bounded 1-Lipschitz functions. Then,

$$\text{dist}\left(\sqrt{n}\bar{R}_{0:T-1}, G_{0:T-1}\right) \lesssim L^T n^{-1/6}$$

- No assumption on the magnitude of π_t
  - If π_t uniformly lower bounded, our proof gives standard $O(n^{-1/2})$-bound
- Despite empirics, conservative convergence rates
  - Nevertheless, usually $T \lll n$ in online platforms

Markov decision process over posterior beliefs

Posterior beliefs as states

- Maintain *beliefs* over average rewards h

Prior	$h \sim \nu := N\left(\mu_0, \text{diag}(\sigma_0^2)\right)$
Likelihood	$G \mid h \sim N\left(\pi h, \text{diag}(\pi s^2)\right)$ ← **Gaussian approximation**

- Observe sample means G_t , then update posterior beliefs

- Goal: choose allocation π_t to maximize terminal reward

Posterior updates as state dynamics

Prior $h \sim \nu := N\left(\mu_0, \mathrm{diag}(\sigma_0^2)\right)$

Likelihood $G \mid h \sim N\left(\pi h, \mathrm{diag}(\pi s^2)\right)$ ← **Gaussian approximation**

- Bayes rule / posterior update gives state dynamics

Posterior variance $\sigma_{t+1}^{-2} = \sigma_t^{-2} + \pi_t / s^2$

Posterior mean $\mu_{t+1} = \sigma_{t+1}^2 / \sigma_t^2 \cdot \mu_t + \sigma_{t+1}^2 / s^2 \cdot G_t$

“Training”

Posterior variance $\sigma_{t+1}^{-2} = \sigma_t^{-2} + \pi_t / s^2$

Posterior mean $\mu_{t+1} = \sigma_{t+1}^2 / \sigma_t^2 \cdot \mu_t + \sigma_{t+1}^2 / s^2 \cdot G_t$

- Goal: plan using roll-outs and maximize terminal reward

$$\text{maximize} \left\{ \mathbb{E}^{\pi} \left[\max_a \mu_{T,a} \right] : \pi_t(\mu_t, \sigma_t), t = 1, \ldots, T-1 \right\}$$

Bayesian adaptive experiment

$$\text{maximize}_{\text{allocation}} \quad \text{Reward of arm chosen at the end of the experiment}$$

- Gaussian observations at each epoch; perform posterior updates over belief on the average rewards
- Prior only over ***average*** rewards
  - Unlike Thompson sampling, no distributional assumptions on *individual* rewards

Bayesian adaptive experiment

$$\text{maximize}_{\text{allocation}} \quad \text{Reward of arm chosen at the end of the experiment}$$

- Tailored to the signal-to-noise ratio in each problem instance and the number of reallocation opportunities T
- ***Offline*** updates: easily deployable to millions of units!
  - Only sample from a fixed allocation, regardless of batch size
  - TS difficult to implement due to real-time posterior inference

"Inference"

Idealized Gaussian

Posterior variance	$\sigma_{t+1}^{-2} = \sigma_t^{-2} + \pi_t / s^2$
Posterior mean	$\mu_{t+1} = \sigma_{t+1}^2 / \sigma_t^2 \cdot \mu_t + \sigma_{t+1}^2 / s^2 \cdot G_t$

“Inference”

Idealized Gaussian

Posterior variance	$\sigma_{t+1}^{-2} = \sigma_t^{-2} + \pi_t / s^2$
Posterior mean	$\mu_{t+1} = \sigma_{t+1}^2 / \sigma_t^2 \cdot \mu_t + \sigma_{t+1}^2 / s^2 \cdot O_t$

"Inference"

Scaled sample mean

Posterior variance	$\sigma_{t+1}^{-2} = \sigma_t^{-2} + \pi_t / s^2$
Posterior mean	$\mu_{t+1} = \sigma_{t+1}^2 / \sigma_t^2 \cdot \mu_t + \sigma_{t+1}^2 / s^2 \cdot \sqrt{n}\bar{R}_t$

- Calculate state transitions / posterior updates using observed sample mean
- Apply learned policy at the current state: $\pi_t(\mu_t, \sigma_t)$

Sanity check

Idealized Gaussian

Posterior variance	$\sigma_{t+1}^{-2} = \sigma_t^{-2} + \pi_t / s^2$
Posterior mean	$\mu_{t+1} = \sigma_{t+1}^2 / \sigma_t^2 \cdot \mu_t + \sigma_{t+1}^2 / s^2 \cdot G_t$

Sanity check

Idealized Gaussian

Posterior variance $\sigma_{t+1}^{-2} = \sigma_t^{-2} + \pi_t / s^2$

Posterior mean $\mu_{t+1} = \sigma_{t+1}^2 / \sigma_t^2 \cdot \mu_t + \sigma_{t+1}^2 / s^2 \cdot G_t$

Scaled sample mean $\sqrt{n}\bar{R}_t$

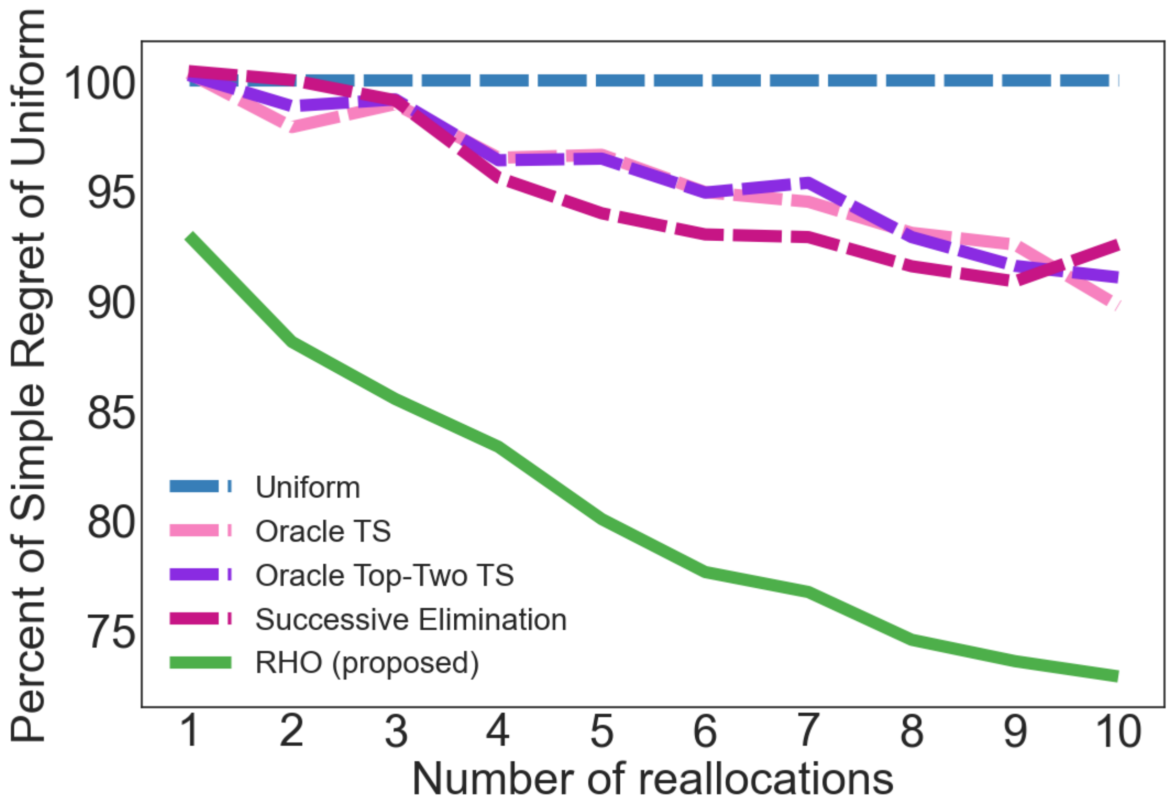
Theorem The two are equivalent for large batch n

Overview

- Despite conventional wisdom, batching simplifies algo design
- Gaussianity is a **result**, not an assumption
- Incorporate prior knowledge on ***average rewards***
- Differentiable dynamic program
  - Bring to bear full power of modern ML + opt tools
  - Objectives can be flexibly encoded

Adaptive designs from approximate dynamic programming

It actually works!



K = 100
n = 10K

Empirical Rigor

aes-batch.streamlit.app

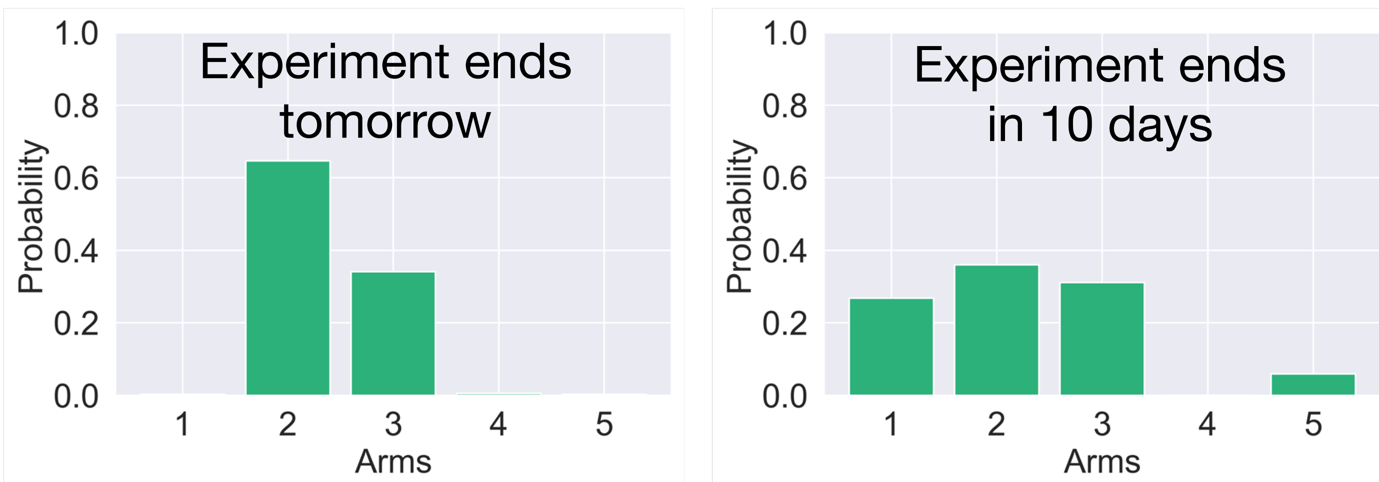
Residual Horizon Optimization

- DP is hard, so consider a simple open-loop policy
  - Optimize over future allocations that only depend on currently available information (μ_t, σ_t)
- Guaranteed to outperform allocations that only use (μ_t, σ_t)
- Sanity checks
  - Sample proportional to measurement noise as $s_a \to \infty$
  - Similar to Thompson sampling as $T \to \infty$

Residual Horizon Optimization

- Resolve planning problem via stoch. gradient descent

$$\min_{\text{allocation } \rho} \mathbb{E}\left[\text{reward} \mid \text{current posterior belief} \right]$$



Calibrate exploration to residual horizon by iterative planning

Pathwise policy gradient

- Differentiable dynamics over the policy parametrization $\pi_{t,a}(\theta)$

Posterior variance	$\sigma_{t+1}^{-2} = \sigma_t^{-2} + \pi_t / s^2$
Posterior mean	$\mu_{t+1} = \sigma_{t+1}^2 / \sigma_t^2 \cdot \mu_t + \sigma_{t+1}^2 / s^2 \cdot G_t$

- Instead of "zero-th order / score trick" estimates (e.g., PPO), use pathwise gradients using auto-differentiation!
  - a.k.a. 21st century infinitesimal perturbation analysis using PyTorch

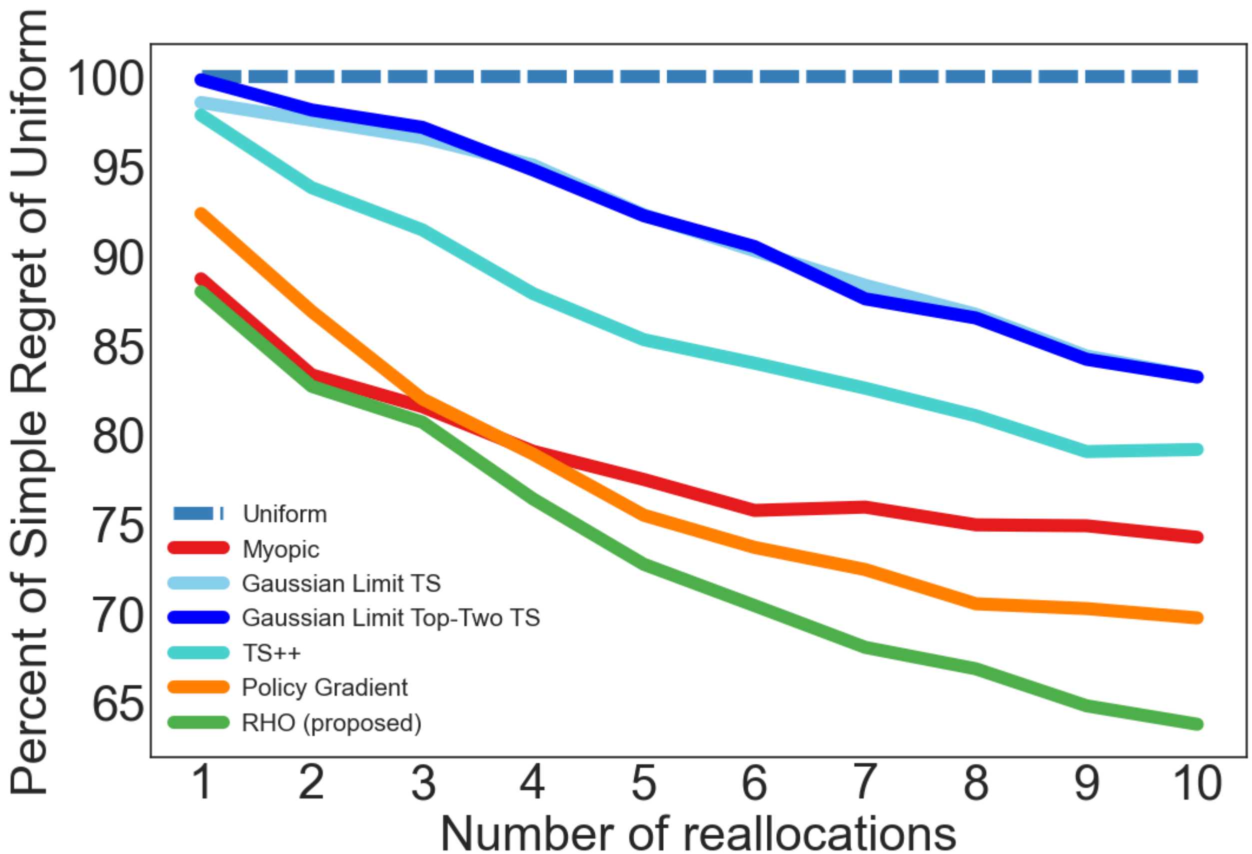
Pathwise policy gradient

- Differentiable dynamics over the policy parametrization $\pi_{t,a}(\theta)$
- Train policy through stochastic gradient ascent

$$\theta \leftarrow \theta + \hat{\nabla} V_0^{\pi(\theta)}(\mu_0, \sigma_0)$$

- Similar performance to RHO when # arms small (K = 10)
- Training challenging for many arms, large horizons, low noise
  - Noisy and vanishing gradients

Gaussian batch policies



K = 100
n = 10K

Comparison against standard methods

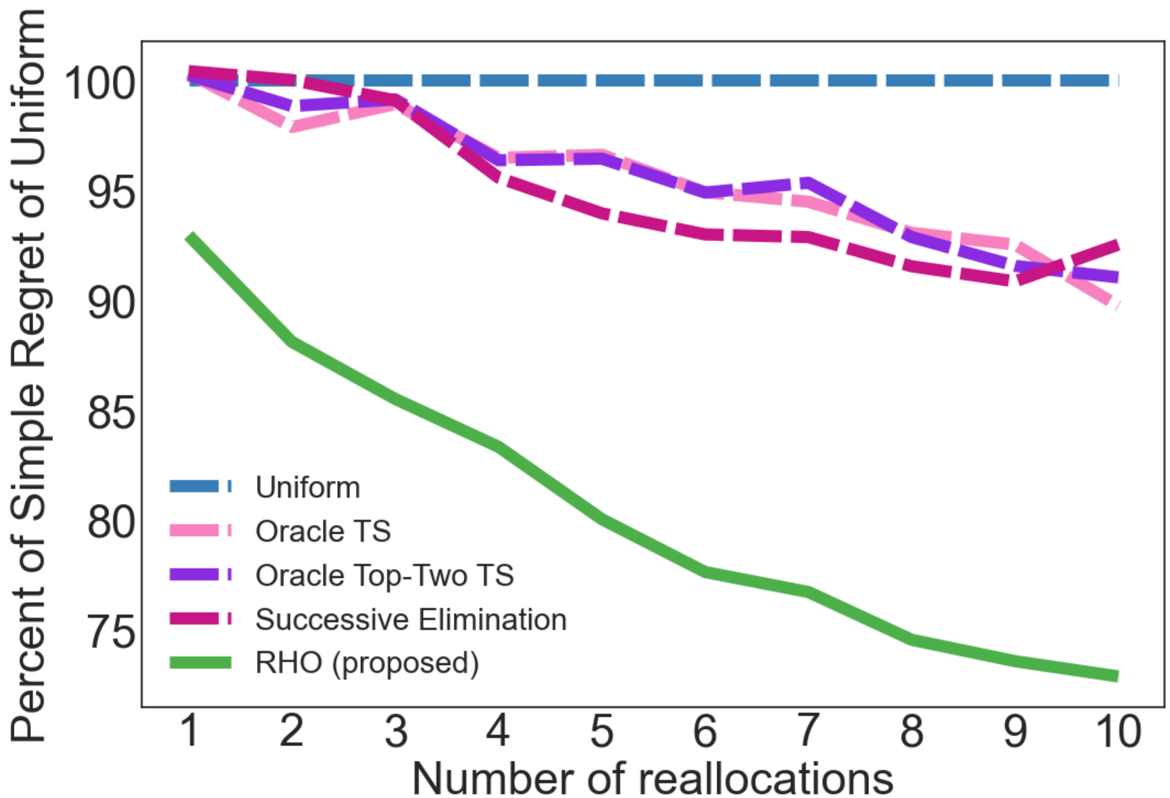
Baselines

- Uniform; static A/B testing
- Batch Successive Elimination
  - Remove arms whose UCB < LCB of other arms
- Batch Thompson sampling
- Batch Top-2 Thompson sampling

} Oracle policies

Batch size

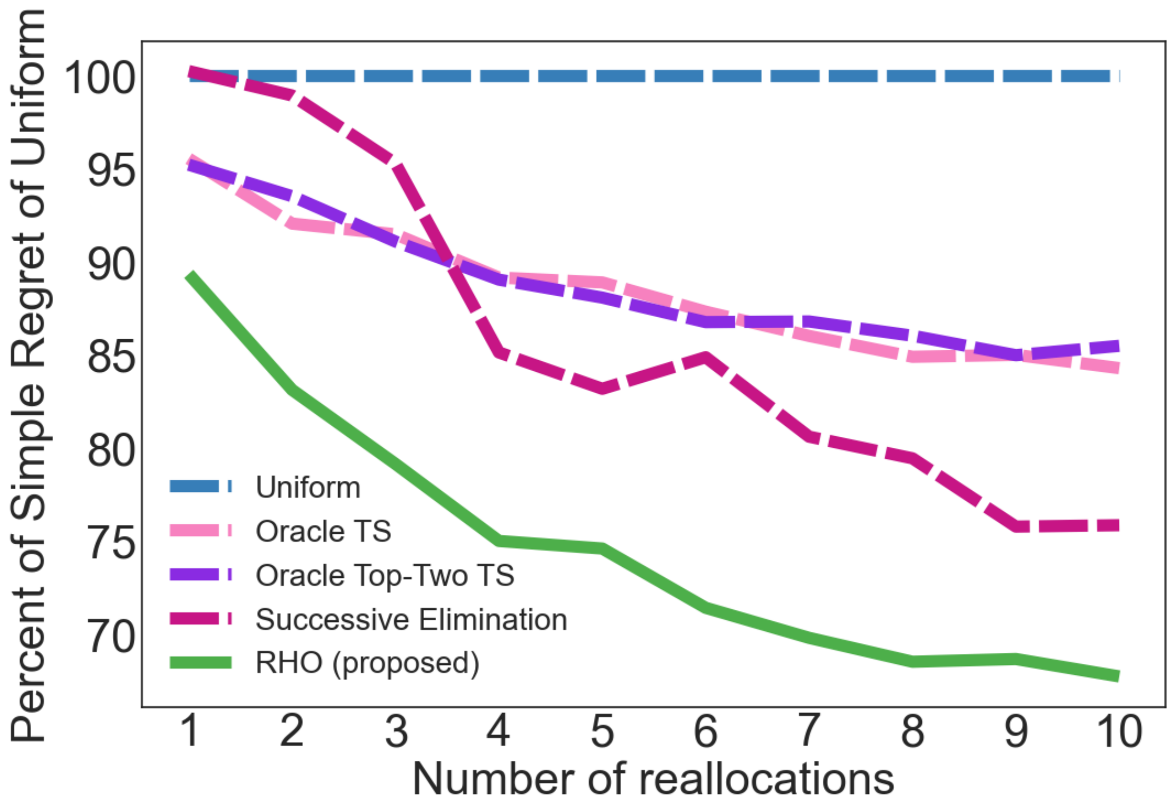
Large batch n = 10000



K = 100
n = 10K

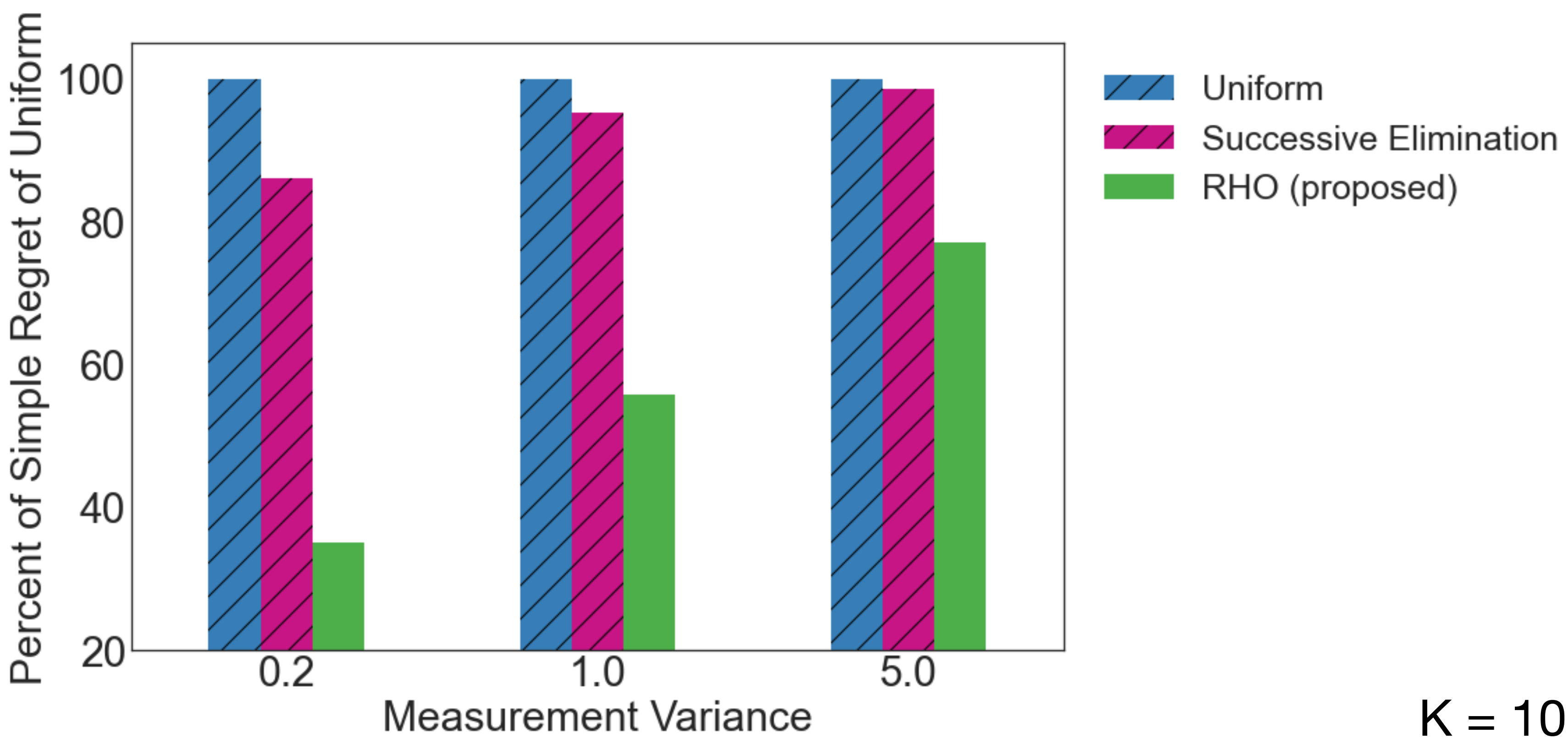
Batch size

Small batch n = 100



K = 100
n = 100

Measurement noise s_a^2



K = 100
n = 100

Empirics takeaways

- Gaussian approximation useful for experimental design
  - Even when batch sizes are small!
- Policies derived from our MDP outperform algos that require complete knowledge of the reward distribution, e.g., TS
- Among these, RHO achieves the largest performance gains
  - Gains large when underpowered: many treatment arms or high measurement noise, where standard adaptive policies struggle

aes-batch.streamlit.app

Recap

- Algorithmic design guided by modern computational tools
  - ML + optimization; trained through differentiable programming
- Optimize “constants”: tailored to small # of reallocations
  - instance-specific measurement noise and statistical power
- Handle batch sizes flexibly
- Empirically validated & ***deployable with ease***