Math Programming For Adaptive Experimentation

Hongseok Namkoong

Columbia Business School

Aug 2024, RLC Deployable RL Workshop

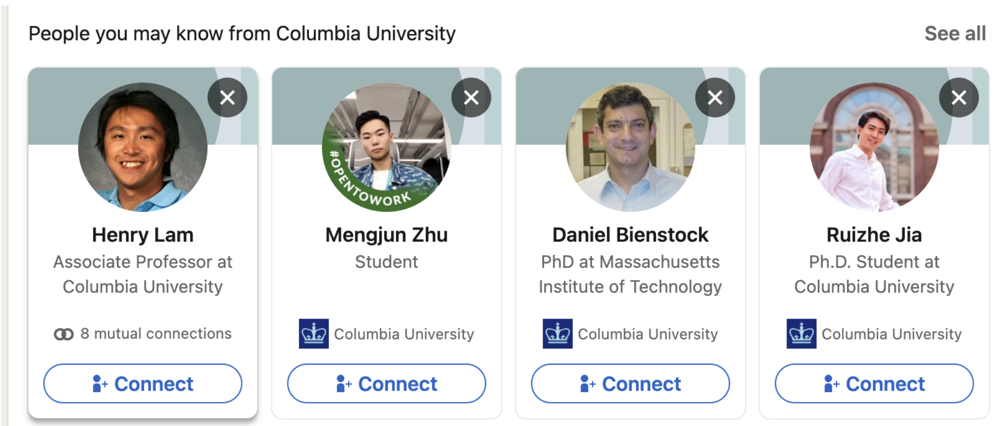
Ethan Che
Columbia

Daniel Jiang
Meta

Jimmy Wang
Columbia

Experimentation (prediction ⇒ decision)

- Imagine a ML engineer building a recommendation system



Configuration 1 2 … K

Goal: help users grow their professional network

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

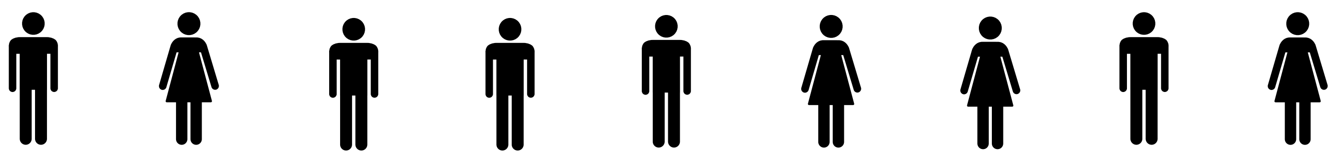
Adaptivity

- Adaptivity improves power => more testable hypotheses
  - Vast literature: Thompson ('33), Chernoff ('59), Robbins & Lai ('52, '85) + 1000s others
- Assumes unit-level continual reallocation
- Algo design guided by theory
  - Regret guarantees hold as # reallocation epochs $T \to \infty$
  - Changes to the objective requires ad hoc changes to algo

Batched Feedback

Practical setting: a **few, large batches**

(think $T = 7$ batches with $n = 100{,}000$ users per batch)



Due to delayed feedback or operational efficiency

Disclaimers

- This talk is about adaptive experiments, not continual interactions with an environment.
- As such, we don't care about $T \to \infty$
- For bandit experts
  - Forget sublinear regret as T grows
  - It's all about constants! We want 20% gains in experiment efficiency.

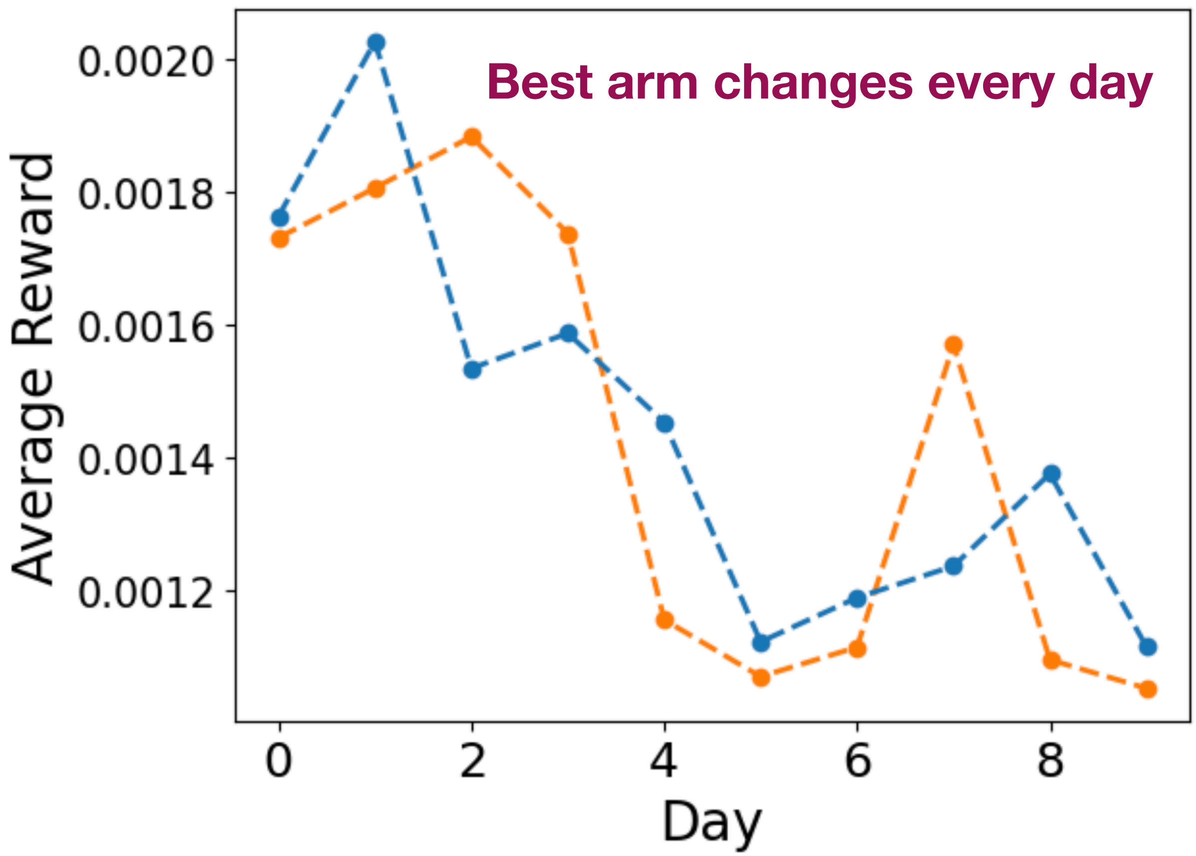
Non-stationarity

- Treatment effects change over day-of-the-week

ASOS Dataset

- Fashion retailer with > 26m active customers
- 78 real experiments with two arms and up to four metrics
  - Means and variances recorded every 12 or 24-hours
  - Duration range from 2~132 recorded intervals
- We generate 241 unique benchmark settings
  - Added additional arms (total 10 arms) with similar gaps as real ones

Non-stationarity



Vignette: Thomson sampling

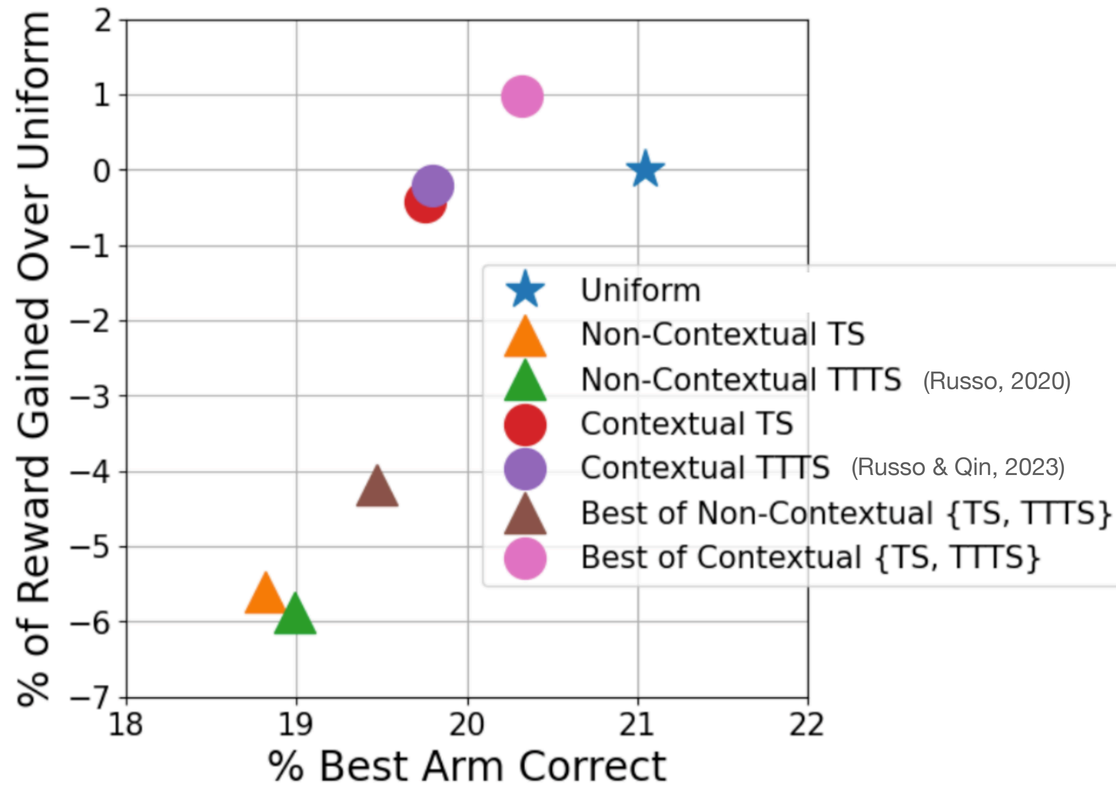
- TS: Select arms with P(Arm optimal | History)
  - Sample parameter $\theta \sim$ Posterior(History), pick best arm under θ
- Top-two TS: Same, but w.p. λ redraw θ until different arm selected
  - Equal to TS if $\lambda = 0$, less greedy as $\lambda \to 1$
  - Contextual variant: Explicitly model non-stationarity

Russo (2020), Russo & Qin (2023)

Vignette: Thomson sampling

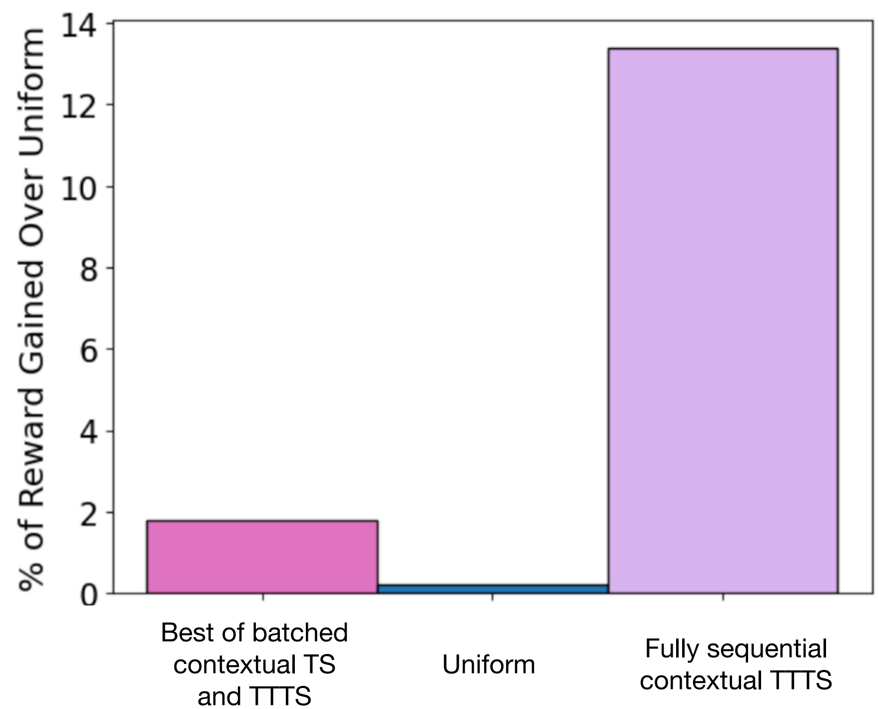
- Contextual policies explicitly model time-varying trends
- Batch size = 100K and horizon T = 10
- **Bandit algos worse than a static A/B test**
- Overfits on initial, temporary performance

180K different problem instances



Why is this happening?

- I didn't follow the instruction manual
- Algo only gets T = 10 chances to update policy; not much adaptivity
- When algo gets to update per person, performs really well!



People want different things

- **Best Arm Identification:** I want the best treatment (simple regret).
- **Top 5 Arm Identification:** Actually, I just want top-5 arms.
- **Personalization:** Learn a *policy* that assigns treatments to users.
- **Multiple Metrics:** Find best arm in a primary metric that's not worse than control in another guardrail metric.

Constraints

- **Sample Coverage:** I want at least 10% of samples for my control arm
- **Budget Constraint:** I can’t give too many discounts.
- **Quality of Service:** I don’t want a regression in this metric during the experiment (with 95% probability).
- **Pacing:** I want to use my budget of samples efficiently as possible over the experiment.

Problem

What is a good algorithmic design principle for…

Top 5 arm identification +

Batched Feedback +

Non-stationarity +

Sample coverage constraints + …

…that will actually materialize into practical performance?

Current art

- Step 1: Hire a person in this room for 1-2 years
- Step 2: Develop a variant of Thomson sampling or UCB adapted to the particular problem instance you have
- Step 3: Prove a nice regret bound for the said algorithm

Current art

- Step 1: Hire a person in this room for 1-2 years
- Step 2: Develop a variant of Thomson sampling or UCB adapted to the particular problem instance you have
- Step 3: Prove a nice regret bound for the said algorithm

- When infeasible, apply some algo not designed for your instance
    - Brittle performance: often even worse than uniform

Mathematical Programming

$$\begin{aligned} \text{minimize}_{\pi} \quad & \text{Objective}(\pi) \\ \text{subject to} \quad & \text{Constraint}(\pi) \leq B \end{aligned}$$

- Write down in a modeling language (e.g., CVX)
- Call a generic solver to get approximate solution (e.g., Gurobi)
- Good solvers should perform well across a wide set of problem instances, rather than focus only on a particular problem

Why do we design problem-specific algos?

Batched Experiments

For t **in range**(T)**:**

Two Treatment Arms: 

Sampling Allocation π_t

π_t

Batched Experiments

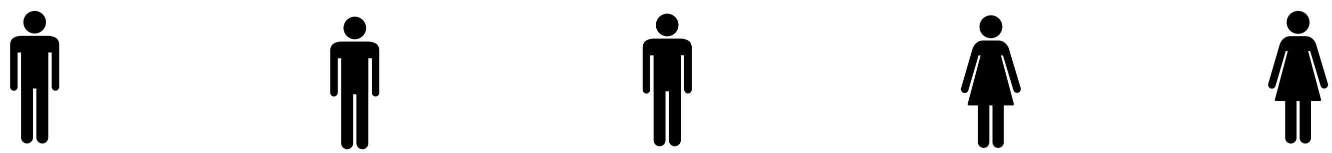
For t **in range**(T)**:**

Two Treatment Arms: 

Sampling Allocation π_t

π_t

Users x_t



Batched Experiments

For t **in range**(T)**:**

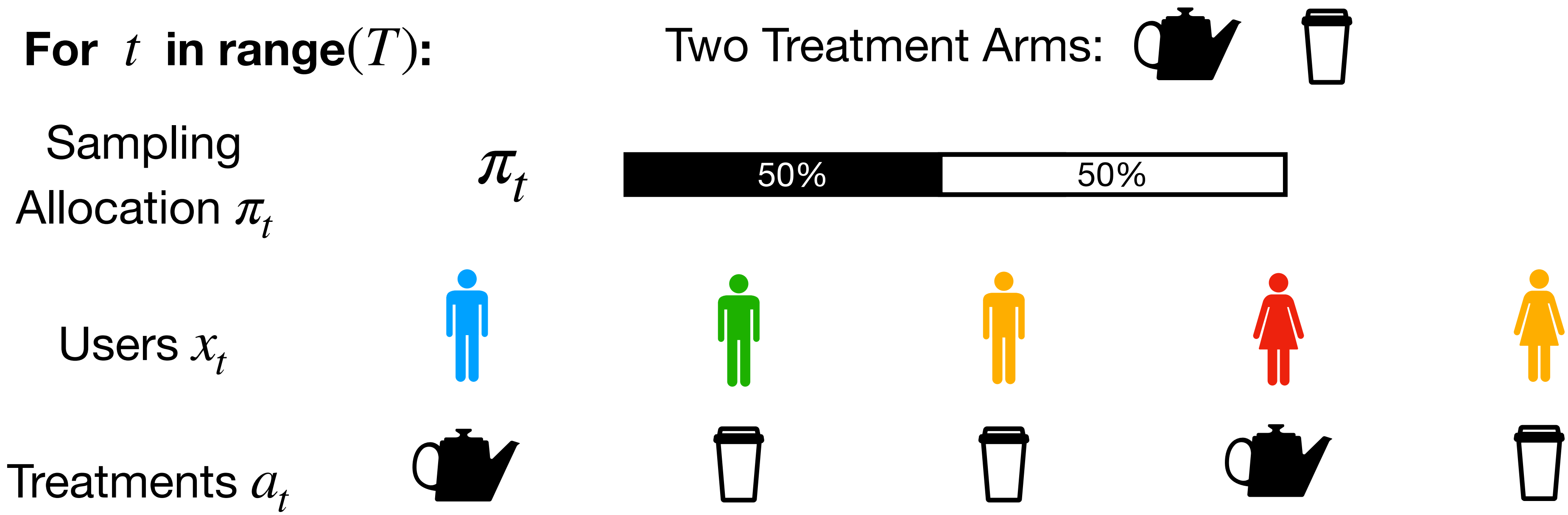
Two Treatment Arms: 

Sampling Allocation π_t

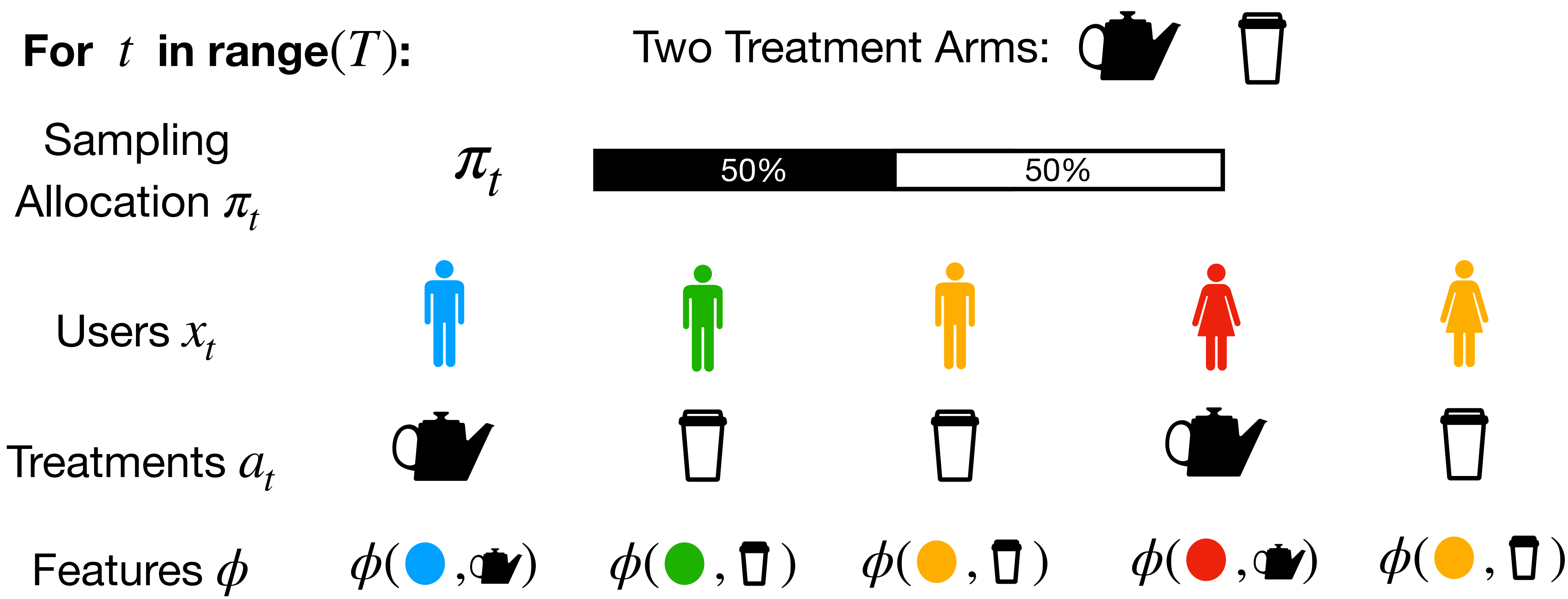
π_t

Users x_t

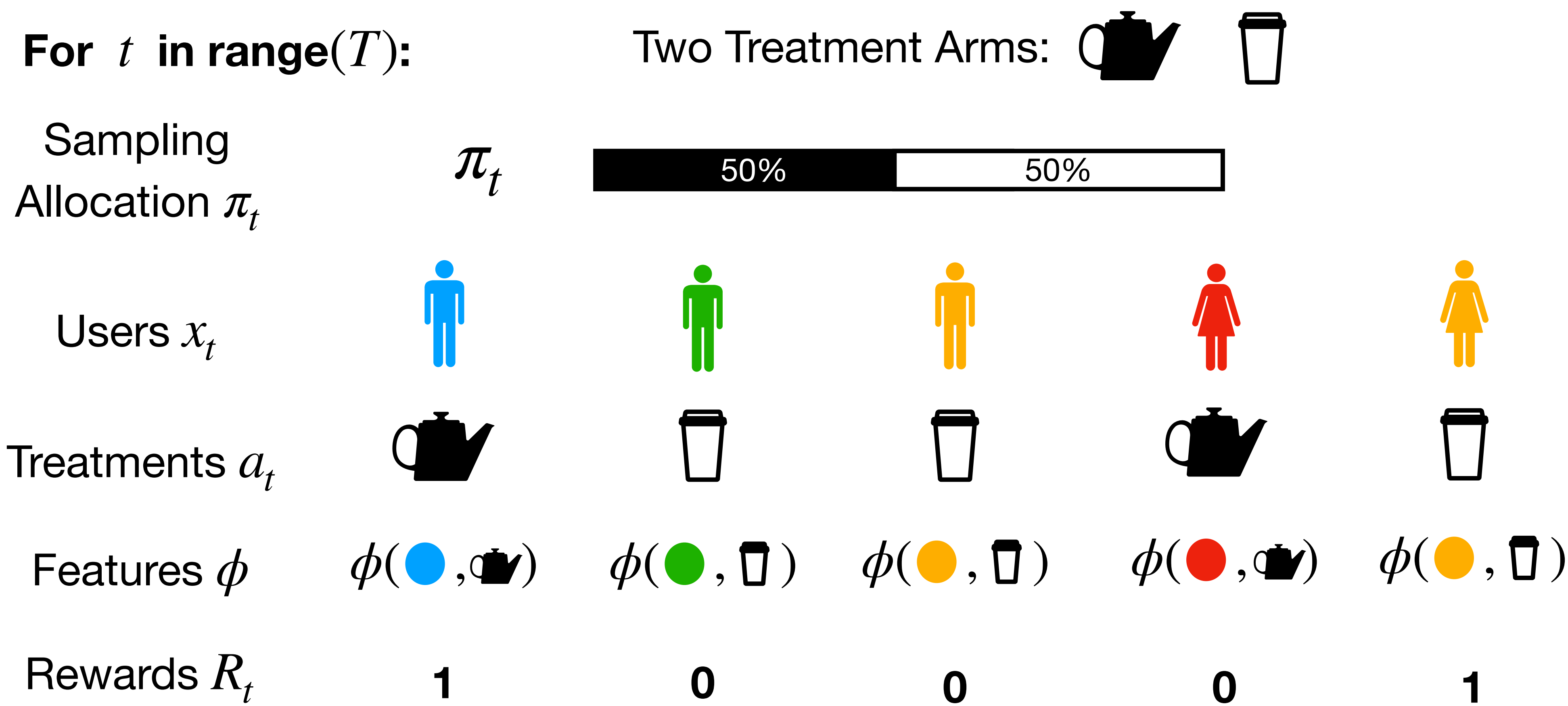
Batched Experiments



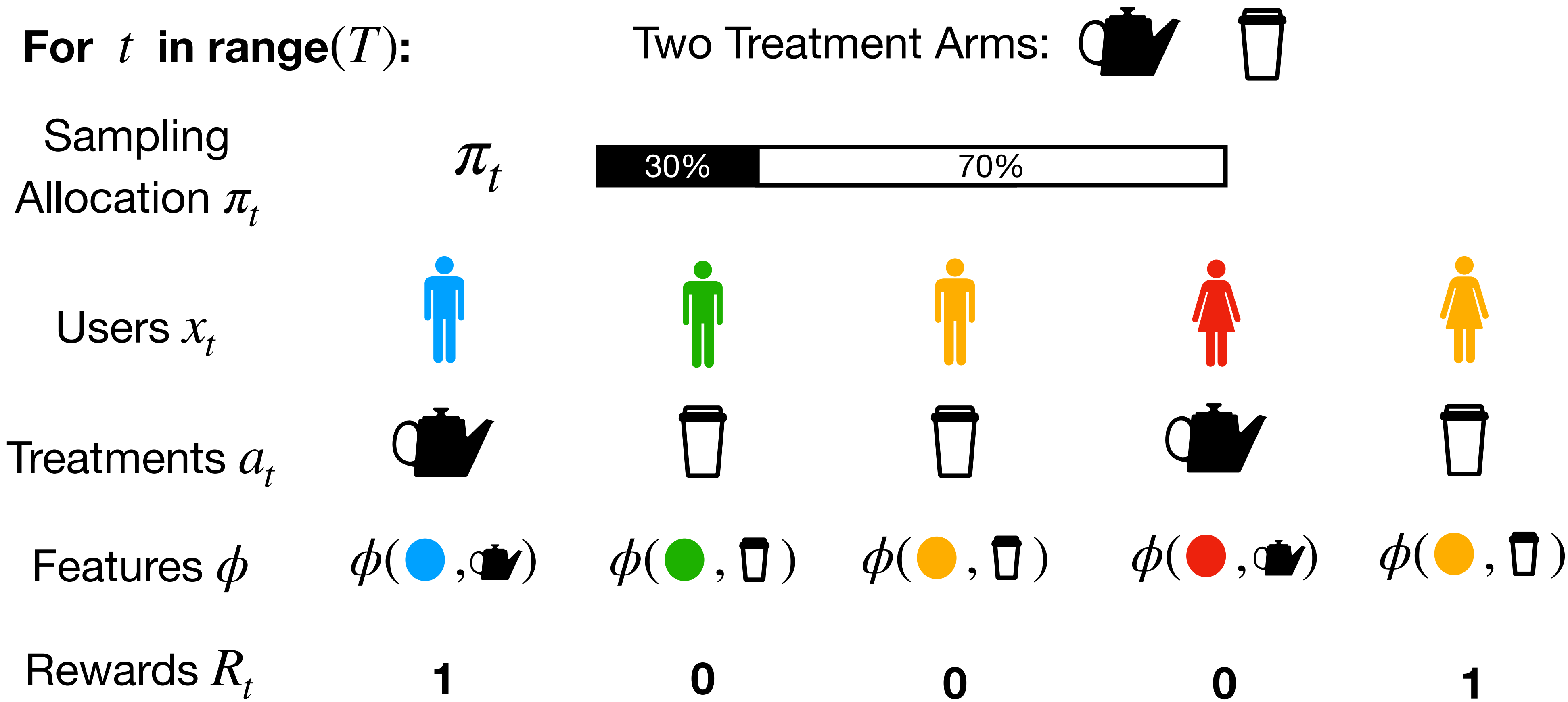
Batched Experiments



Batched Experiments



Batched Experiments



Adaptive experimentation as dynamic program

$$\text{minimize}_{\pi_t(H_t)} \quad \mathbb{E}\left[\sum_{t=0}^{T} \text{Objective}_t(\pi_t, H_t)\right]$$

$$\text{subject to} \qquad \text{Cost}(\pi_t; H_t) \leq c_t$$

$$\pi_t(H_t) \in \text{Simplex}$$

Reward/outcome distribution $R \sim \nu(\,\cdot\,)$ unknown

Bayesian MDP

- Adopt Bayesian principles to reason through uncertainty on ν
- Let Q_t be posterior on ν given the history H_t

$$\text{minimize}_{\pi_t(H_t, Q_t)} \quad \mathbb{E}\left[\sum_{t=0}^{T} \text{Objective}_t(\pi_t, H_t, Q_t)\right]$$

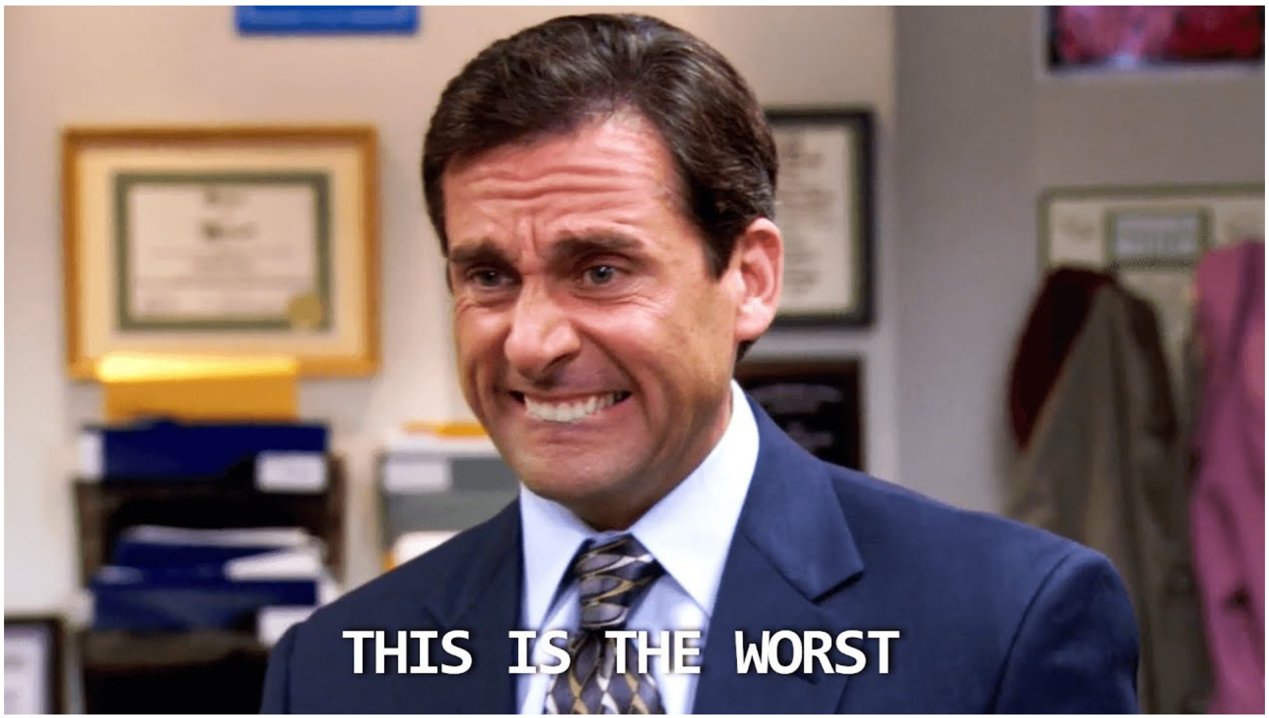
$$\text{subject to} \qquad \text{Cost}(\pi_t; H_t, Q_t) \le c_t$$

$$\pi_t(H_t, Q_t) \in \text{Simplex}$$

Bayesian MDP

- States
  - Observed data H_t; dimension = no. users
  - Posterior distribution Q_t; infinite dimensional in general
- Requires a Bayesian model for how each user behaves
- Even computing state transitions (posterior update) is a challenge

Bayesian MDP



Simplifying the Bayesian MDP

- Assume parametric model for *mean* rewards with true param $\theta^\star$
- Examples
  - Finite armed MAB: $\theta^\star$ = average reward across arms
  - Contextual model
    - Linear rewards: $\mathbb{E}[R \mid X = x, A = a] = \phi(x, a)^\top \theta_a^\star$
    - Logistic model: logistic$(R) = \phi(x, a)^\top \theta_a^\star$

Simplifying the Bayesian MDP

- Within each batch t, central limit theorem says

  maximum likelihood estimator $\widehat{\theta}_t \sim N(\theta^\star, n^{-1} g(\pi_t))$

- 99% of statistics; everyone uses this to calculate p-values
- CLT compress entire batch to sufficient statistic $\widehat{\theta}_t$

Bayesian Principle Over Batches

Governed by **posterior mean and variance** (β_t, Σ_t)

Prior	Likelihood	Posterior
$\theta^\star \sim N(\beta_0, \Sigma_0)$	$\widehat{\theta}_t \sim N(\theta^\star, n^{-1} g(\pi_t))$	$\theta^\star \sim N(\beta_1, \Sigma_1)$

Batch compressed to sufficient statistic

Bayesian Principle Over Batches

Governed by **posterior mean and variance** (β_t, Σ_t)

Prior | Likelihood | Posterior

$$\theta^\star \sim N(\beta_0, \Sigma_0) \longrightarrow \widehat{\theta}_t \sim N(\theta^\star, n^{-1}g(\pi_t)) \longrightarrow \theta^\star \sim N(\beta_1, \Sigma_1)$$

Bayesian Principle Over Batches

Governed by **posterior mean and variance** (β_t, Σ_t)

Prior | Likelihood | Posterior

$$\theta^\star \sim N(\beta_0, \Sigma_0) \longrightarrow \widehat{\theta}_t \sim N(\theta^\star, n^{-1} g(\pi_t)) \longrightarrow \theta^\star \sim N(\beta_1, \Sigma_1)$$

- Computationally, closed-form posterior state transitions
  - Posterior update formula for Gaussian conjugate family
  - Differentiable dynamics

Batch Limit Dynamic Program

$$\text{minimize}_{\pi_t(\beta_t,\Sigma_t)} \quad \mathbb{E}\left[\sum_{t=0}^{T} \text{Objective}_t(\pi_t, \beta_t, \Sigma_t)\right]$$

$$\text{subject to} \qquad \text{Cost}(\pi_t; \beta_t, \Sigma_t) \leq c_t$$

$$\pi_t(\beta_t, \Sigma_t) \in \text{Simplex}$$

- State dimension = $O(d^2)$

Batch Limit Dynamic Program

- Models any objective and constraint that can be written as a function of posterior states
  - Cumulative- and simple-regret, top-k regret
  - Budget constraints, minimum allocation constraints
  - Above applied to any number of rewards/outcomes/metrics

Residual Horizon Optimization

- At every epoch, given posterior state (β, Σ), solve for the optimal **static** sampling allocations
- Resolve every batch, based on new information

$$\text{minimize}_{\pi_t(\beta_t,\Sigma_t)} \quad \mathbb{E}\left[\sum_{t=s}^{T} \text{Objective}_t(\pi_t, \beta_t, \Sigma_t) \mid \beta_s, \Sigma_s\right]$$

$$\text{subject to} \qquad \text{Cost}(\pi_t; \beta_t, \Sigma_t) \le c_t \qquad t \ge s$$

$$\pi_t(\beta_t, \Sigma_t) \in \text{Simplex}$$

Residual Horizon Optimization

- At every epoch, given posterior state (β, Σ), solve for the optimal **static** sampling allocations
- Resolve every batch, based on new information

Constants

$$\text{minimize}_{\pi_t \cancel{(\beta_t, \Sigma_t)}} \quad \mathbb{E}\left[\sum_{t=s}^{T} \text{Objective}_t(\pi_t, \beta_t, \Sigma_t) \mid \beta_s, \Sigma_s \right]$$

$$\text{subject to} \qquad \text{Cost}(\pi_t; \beta_t, \Sigma_t) \leq c_t \qquad t \geq s$$

$$\pi_t \cancel{(\beta_t, \Sigma_t)} \in \text{Simplex}$$

Residual Horizon Optimization

- At every epoch, given posterior state (β, Σ), solve for the optimal **static** sampling allocations
- Resolve every batch, based on new information

$$\text{minimize}_{\pi_t} \quad \mathbb{E}\left[\sum_{t=s}^{T} \text{Objective}_t(\pi_t, \beta_t, \Sigma_t) \mid \beta_s, \Sigma_s\right]$$

$$\text{subject to} \qquad \text{Cost}(\pi_t; \beta_t, \Sigma_t) \le c_t \qquad t \ge s$$

$$\pi_t \in \text{Simplex}$$

Residual Horizon Optimization

$$\text{minimize}_{\pi_t} \quad \mathbb{E}\left[\sum_{t=s}^{T} \text{Objective}_t(\pi_t, \beta_t, \Sigma_t) \mid \beta_s, \Sigma_s\right]$$

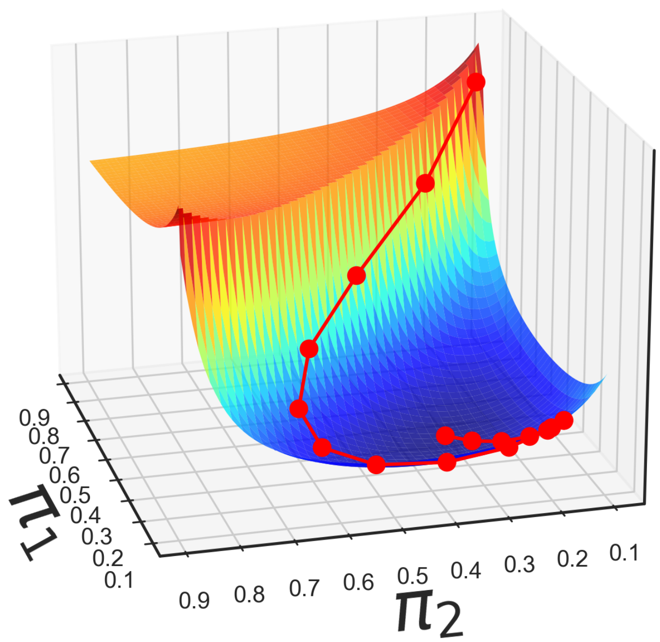
$$\text{subject to} \qquad \text{Cost}(\pi_t; \beta_t, \Sigma_t) \leq c_t \qquad t \geq s$$

$$\pi_t \in \text{Simplex}$$

- Closed-form dynamics means (β_t, Σ_t) can be expressed explicitly
- Use stochastic gradients to optimize allocations!

Residual Horizon Optimization

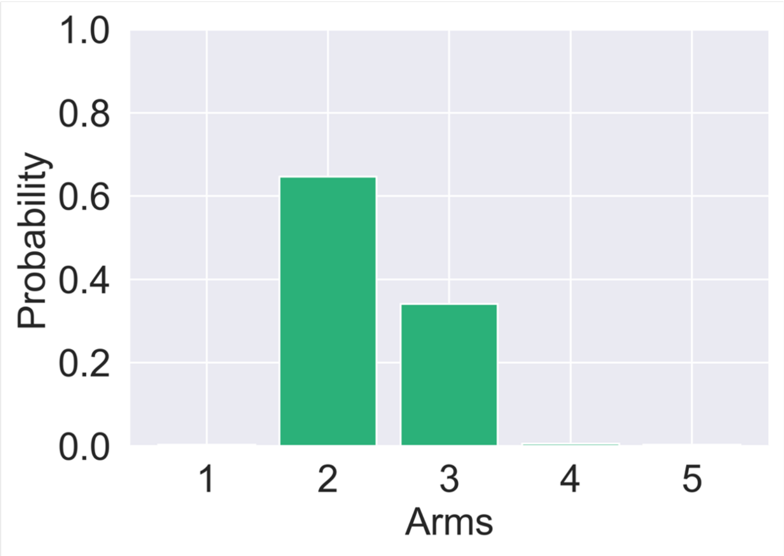
- Use stochastic gradients to optimize allocations!



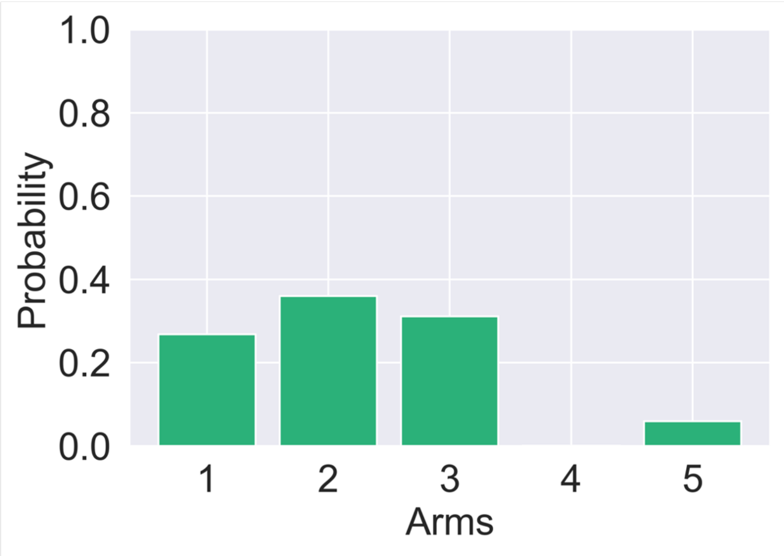
PyTorch

Residual Horizon Optimization

Why planning? Calibrate exploration to horizon



(c) RHO ($T - t = 1$)
(optimal)



(d) RHO ($T - t = 10$)

MPC Design Principle

Theorem: RHO achieves a smaller Bayesian regret than *any* static policy

- For any time horizon T
- For any constraints
- For any objective
- For any time non-stationarity

Why? The algorithm is **Policy Iteration on Static Designs**

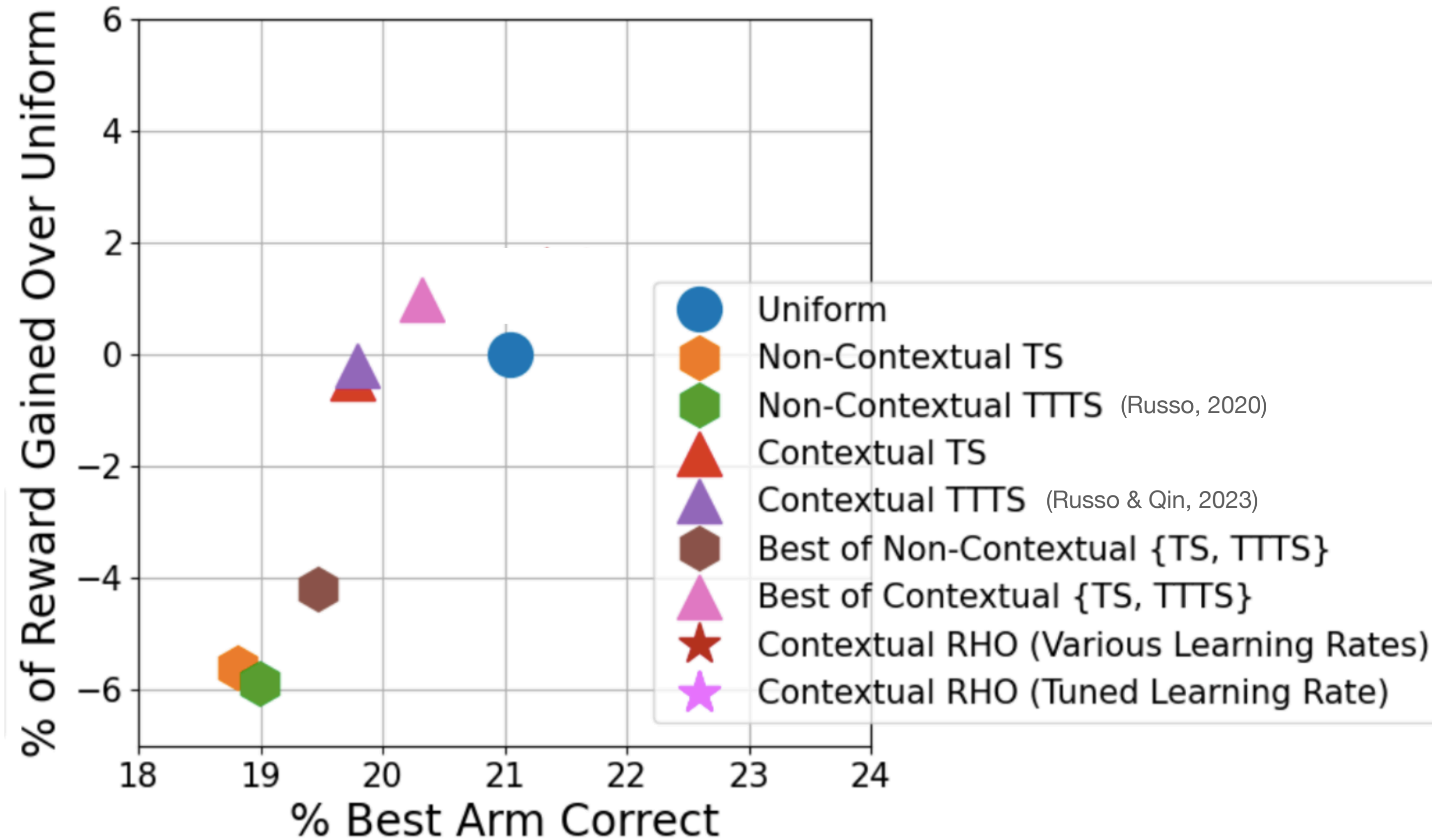
Back to non-stationarity

Benchmarking results over 180K different instances

Contextual = model time-varying trends

Batch size = 100K

Horizon T = 10



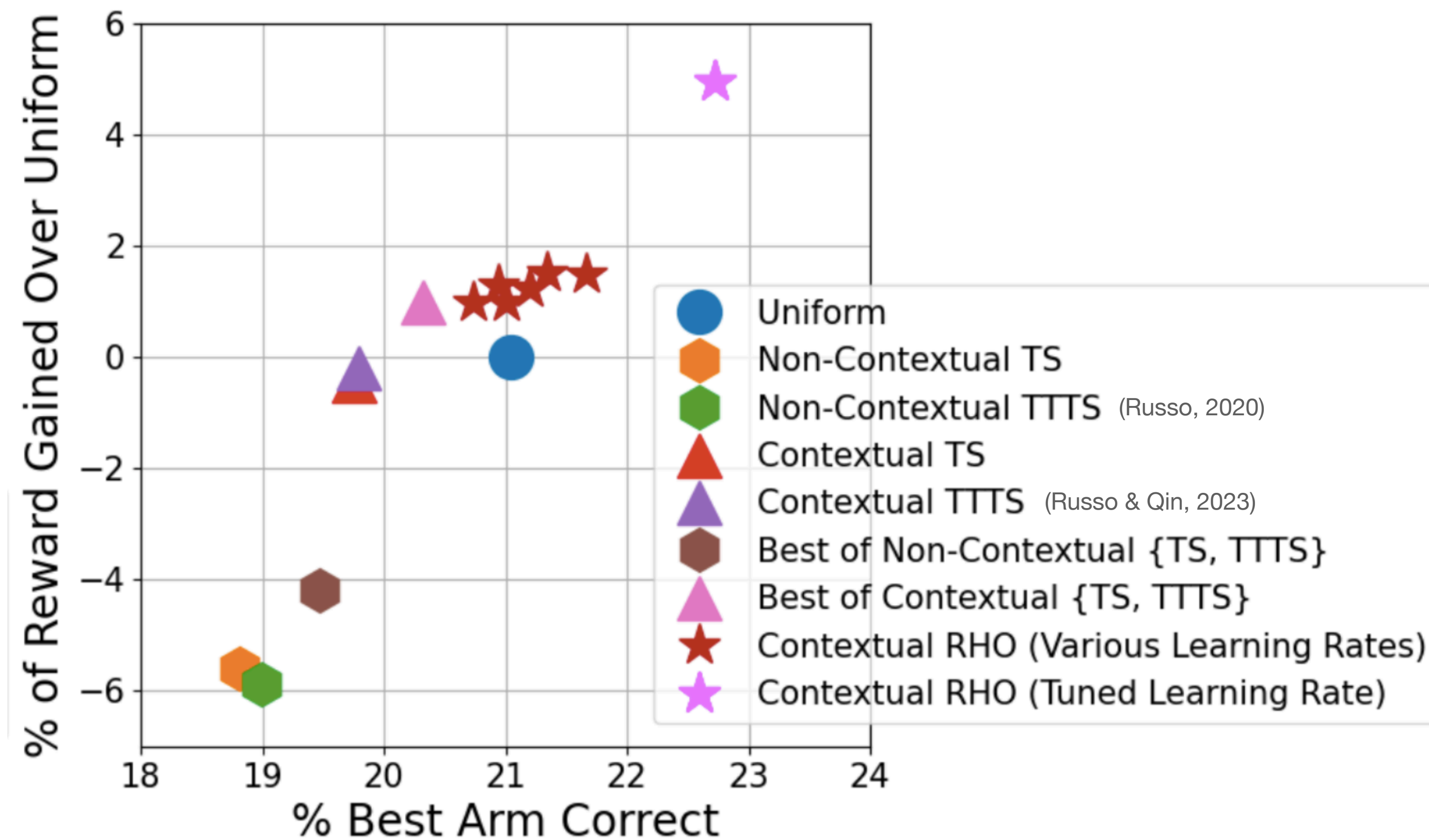
Back to non-stationarity

Benchmarking results over 180K different instances

Contextual = model time-varying trends

Batch size = 100K

Horizon T = 10



Encoding different objectives

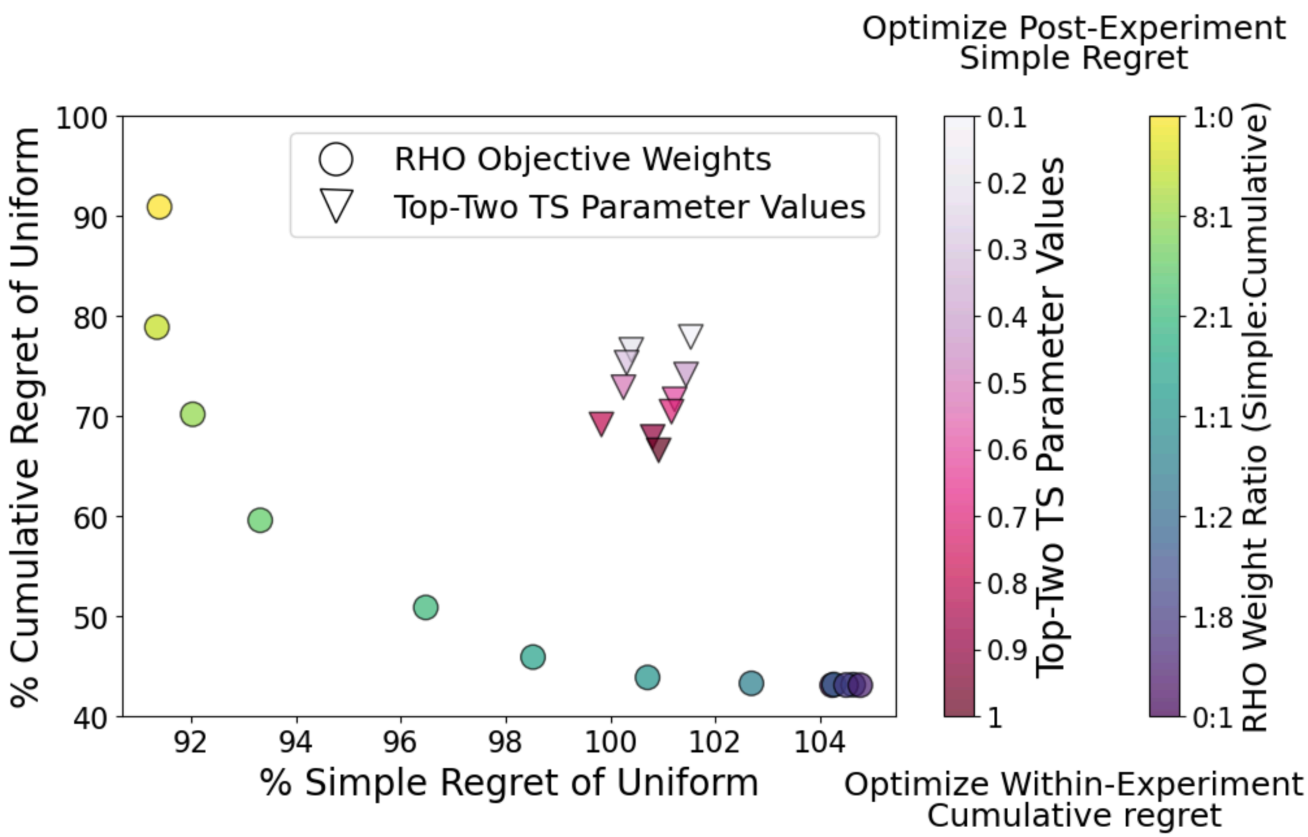
$$\text{minimize}_{\pi_t} \quad \mathbb{E}\left[\sum_{t=0}^{T-1} \text{Within-exp. Rewards}_t(\pi_t, \beta_t, \Sigma_t) + \lambda \cdot \text{Post-exp Rewards}(\pi_T, \beta_T, \Sigma_T)\right]$$

$$\text{subject to} \qquad \text{Cost}(\pi_t; \beta_t, \Sigma_t) \le c_t\,, \qquad \pi_t \in \text{Simplex}$$

- Natural candidate for λ: # in experiment / # affected by treatment
- Unlike TS-based policies, easy to balance within-experiment (simple) vs. post-experiment (cumulative) regret

Encoding different objectives

Batch size n = 100, Horizon T = 5



Summary

- Optimization-based planning for adaptive experimental design
  - Flexibly handles batches, objectives, constraints, and non-stationarity
  - Robustness guarantees against static A/B tests
- Normal approximations universal in statistical inference also delivers a tractable way to directly optimize experiments
- Intellectual foundation: sequential CLT
  - All quantities depend on previous observations; theory requires great care

Papers

hsnamkoong.github.io

- Mathematical Programming For Adaptive Experiments

  arXiv:2408.04570 with E. Che, D. Jiang, J. Wang

- AExGym: Benchmarks and Environments for Adaptive Experimentation

  arXiv:2408.04531 github.com/namkoong-lab/aexgym with J. Wang, E. Che, D. Jiang

- Adaptive Experimentation at Scale: A Computational Framework for Flexible Batches

  arXiv:2303.11582 with E. Che