Bandits Algorithms for Factorial Design

Yutong Yan
joint work with
Audrey Durand, Joelle Pineau

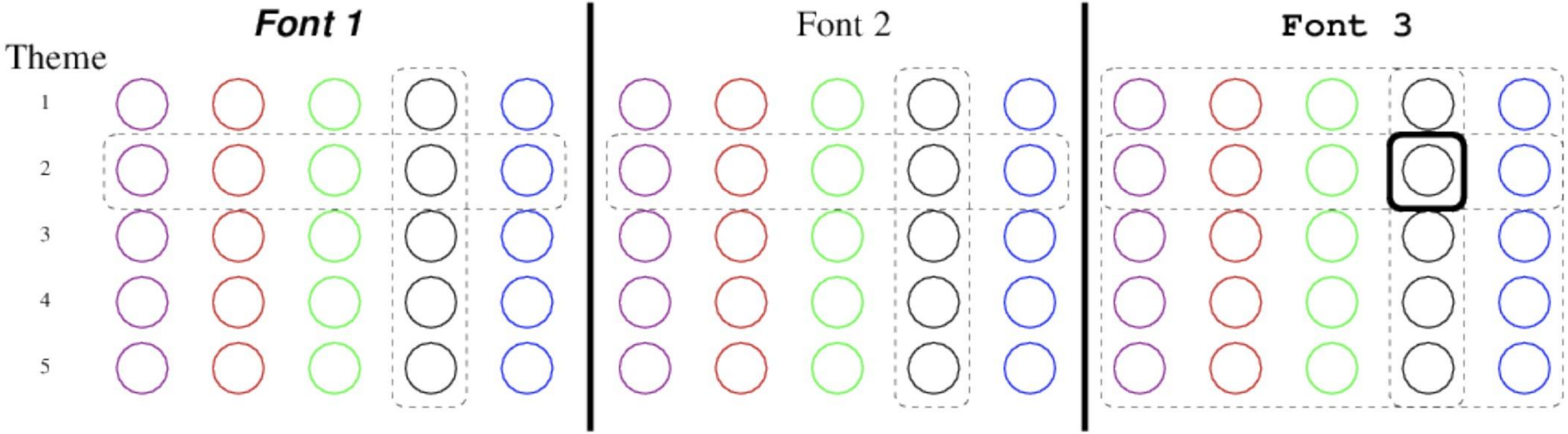
Université Laval
Mila
McGill
School of Computer Science

Outline

- Factorial Experiment
- Multi-armed Bandits
- Upper Confidence Bound vs Thompson Sampling
- Linear Bandits
- Bandits in Tree Search
- Proposed UCT-Laplace
- Results

Factorial Experiment:

- An observation provides partial information on several *configurations*
- 3 fonts X 5 themes X 5 colors = 75 configurations
- A single run of configuration (Font=3, Color=4, Theme=2) … gives partial information on all configurations with Color=4, Theme=2, or Font=3
- Each observation teaches us something about 43 configurations



Multi-armed bandits (MAB)

- Sequential experimental design
- Produces the reward under an uncertain payoff distribution

Multi-armed bandit: Problem statement

Algorithm 1: MAB

Input: Action set $\mathcal{A}$

for *each time step* $t := 1, ..., T$ **do**

  Select a_t from $\mathcal{A}$

  Get reward $r_t \sim \mathcal{D}_{a_t}$

  Update a_t using r_t

end

- **Maximize** $\sum_{t=1}^{T} r_t$
- **By choosing optimal action:**

$$a_\star = \arg\max_{a \in \mathcal{A}} \mu_a$$, where

$$\mu_a = \mathbb{E}[\mathcal{D}_a]$$

- **Minimize expected regret:**

$$\sum_{t=1}^{T} [\mu_{a_\star} - \mu_{a_t}]$$

Upper Confidence Bound (UCB)

- **Optimism in the face of uncertainty (OFU)**
- **Pick the arm with the highest upper confidence bound**
- **[Auer *et al.*(2002)] showed UCB satisfies optimal rate of exploration [Lai and Robbins(1995)]**
- **The most popular choice is UCB1:**

$$B_{a,t} = \hat{\mu}_{a,t} + 2C\sqrt{\frac{\ln t}{N_{a,t}}}$$

Thompson Sampling (TS)

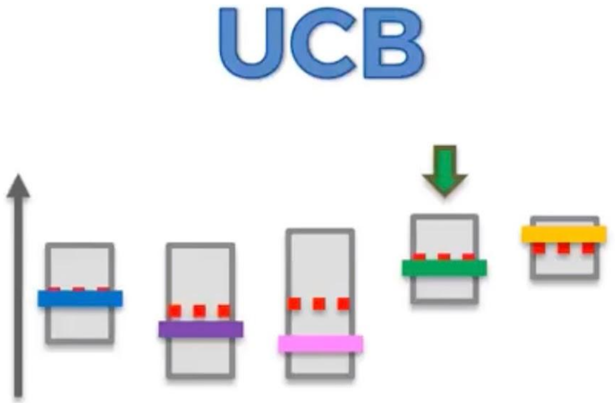
- **Thompson sampling was first induced by [Thompson (1933)]**
- **Bayes theorem:** $\mathrm{P}(\theta|r_a^1,\ldots,r_a^n) \propto \mathrm{P}(\theta)\mathrm{P}(r_a^1,\ldots,r_a^n|\theta)$
- **Sample from posterior distribution (conditioned on the observed rewards) for arm *a*:** $\tilde{\mu}_a \sim \mathrm{P}_a$
- **Selection rule:** $\arg\max_{a \in \mathcal{A}} \tilde{\mu}_a$
- **Example: Gaussian bandits sampling:**

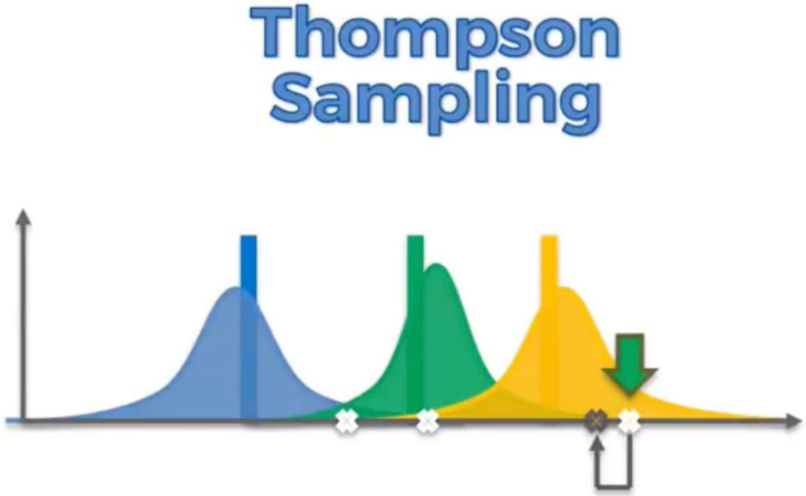
$$\tilde{\mu}_{a,t} \sim \mathcal{N}(\hat{\mu}_{a,t}, \frac{1}{N_{a,t}+1})$$

UCB vs TS

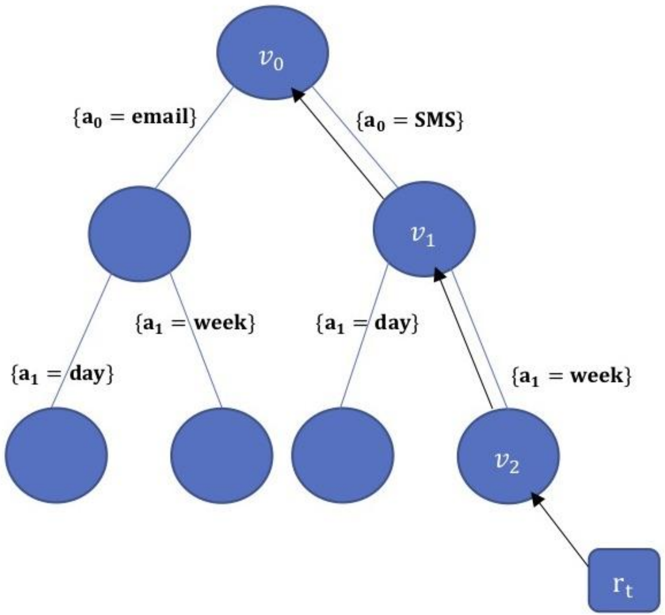
- Deterministic
- Has theoretical guarantees

- Probabilistic
- Better empirical performance





Factorial Experiment as Bandits



- MAB:
  - Each configuration as an arm
  - K=4 arms

Linear Bandits

Algorithm 2: Linear bandits

```
for each time step t := 1, ..., T do
    Observe feature set 𝒳_t
    Select x_t from 𝒳_t
    Get reward r_t = ⟨θ_⋆, x_t⟩ + ε_t
    Update θ̂ using r_t and x_t
end
```

- **Maximize** $\sum_{t=1}^{T} r_t$
- **By choosing optimal action:**

$$x_\star = \arg\max_{x \in \mathcal{X}_t} \underbrace{\langle \theta_\star, x \rangle}_{\mu_x}$$

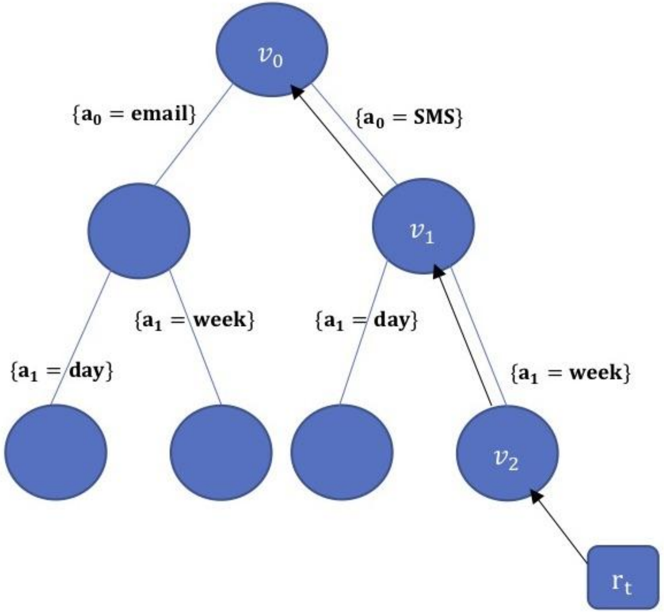
- **Minimize expected regret:**

$$\sum_{t=1}^{T} [\mu_{x_\star} - \mu_{x_t}]$$

- **Linear Thompson Samping (LinTS):**

$$\tilde{\theta}_t \sim \mathcal{MVN}(\hat{\theta}_t, v^2 \overline{V}_t)$$

Factorial Experiment as Bandits



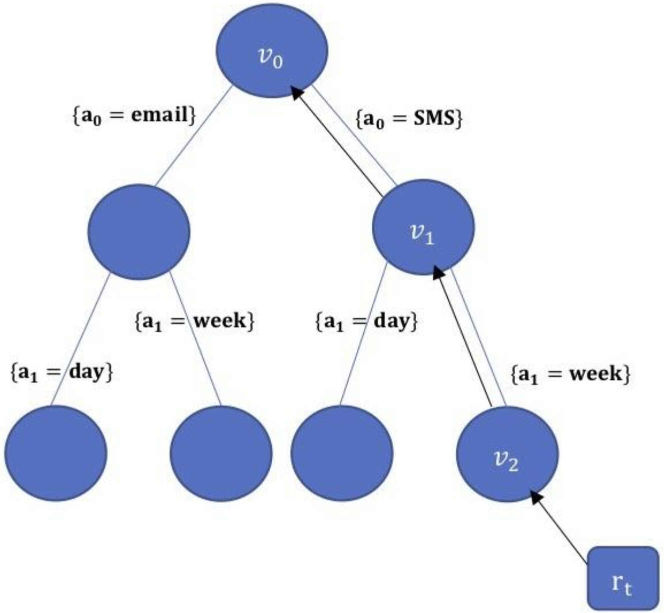
- MAB:
  - Each configuration as an arm
  - *K*=4 arms
- Linear Bandits:
  - Binary feature vector dimension *d*=4
  - [email(no), SMS(yes), day(no), week(yes)]
  - [0 1 0 1]

Bandits Algorithms for Tree Search (BATS)

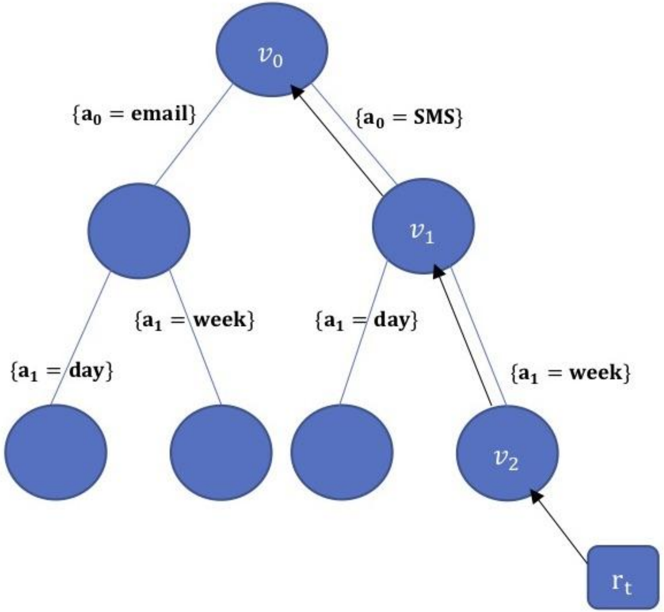
Algorithm 3: BAST

for *each time step* $t := 1, 2, 3, \ldots, T$ **do**
- $v_d \leftarrow v_0$
- Start running trajectory
- **for** *depth* $d := 1, 2, \ldots, D$ **do**
  - Select $v' \in \mathcal{C}(v_d)$
  - $v_d \leftarrow v'$
- **end**
- Reach the leaf $l_t \leftarrow v_D$
- End trajectory
- Receive reward $r_t \sim \mathcal{D}_{l_t}$
- Update nodes visited in the trajectory

end



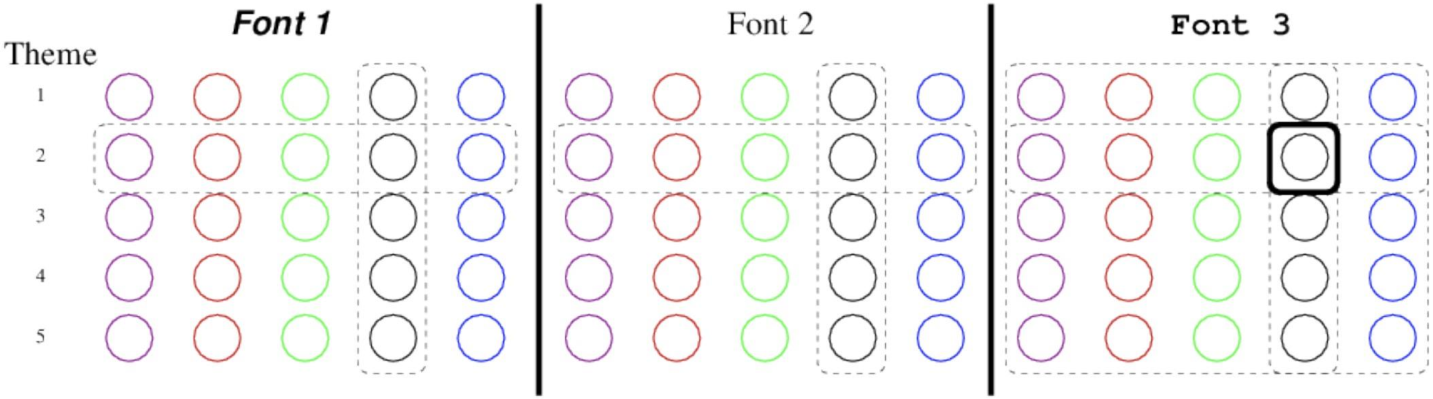
Factorial Experiment as Bandits



- **MAB:**
  - **Each configuration as an arm**
  - ***K*=4 arms**
- **Linear Bandits:**
  - **Binary feature vector dimension is 4**
  - **[email(no), SMS(yes), day(no), week(yes)]**
  - **[0 1 0 1]**
- **BATS:**
  - **2 *factors* and 2 *choices* per factor**
  - ***2 x 2 x 2* factorial design**
  - **Tree Depth *D*=2**
  - **At depth *d*=1, email or SMS**
  - **At depth *d*=2, every day or every week**

Recall: Factorial Experiment

- 3 fonts X 5 themes X 5 colors = 75 configurations
- MAB: *K*=75 arms
- LinTS: Feature vector dimension=13
- BATS: Tree depth *D*=3
  - 2 factors with 5 choices per factor, 1 factor with 3 choices per factor
  - *3 X 5 X 5 factorial design*

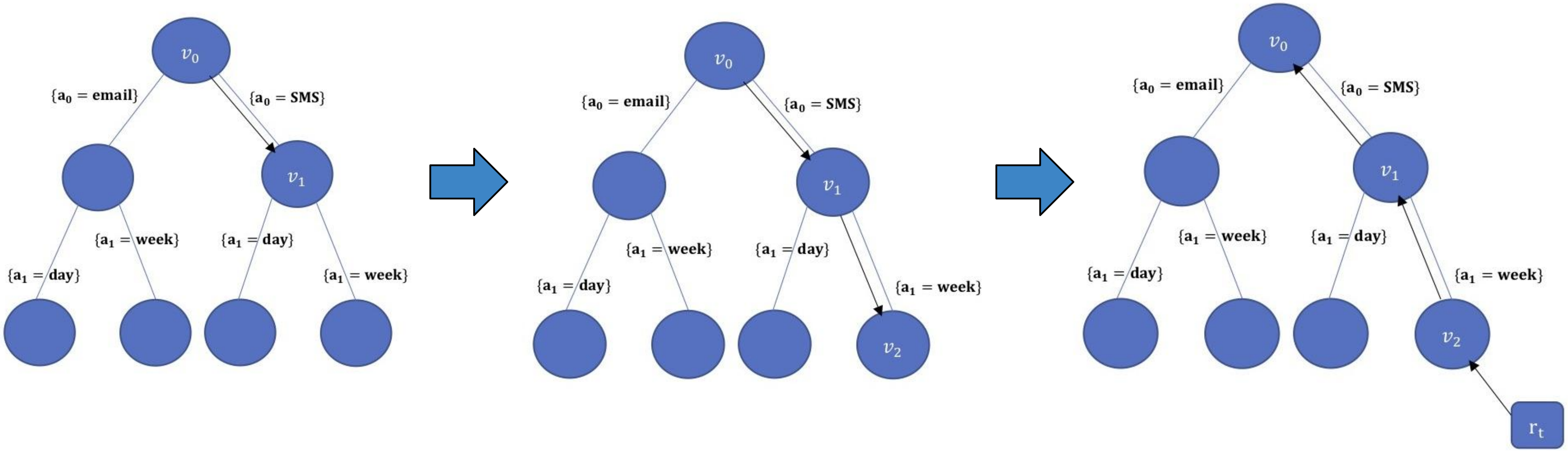


Upper Confidence Bound applied to Trees (UCT)

- **UCB1 in Tree Search!**
- **Compute the *B*-value for every children node** $v' \in \mathcal{C}(v)$
- ***B*-value in tree search:**

$$B_{v',t} = \hat{\mu}_{v',t} + 2C\sqrt{\frac{\ln N_{v,t}}{N_{v',t}}}$$

UCT example: medical app



$$v_{\text{SMS}} \in \mathcal{C}(v_0)$$
$$B_{\text{email}} < B_{\text{SMS}}$$

$$v_{\text{week}} \in \mathcal{C}(v_1)$$
$$B_{\text{day}} < B_{\text{week}}$$

Update the nodes in the trajectory $\{v_0, v_1, v_2\}$

Benefits of UCT

- Computational-efficient
- Information-sharing
- Theoretical guarantees
- Weaker assumption on unknown reward function

UCT-Laplace

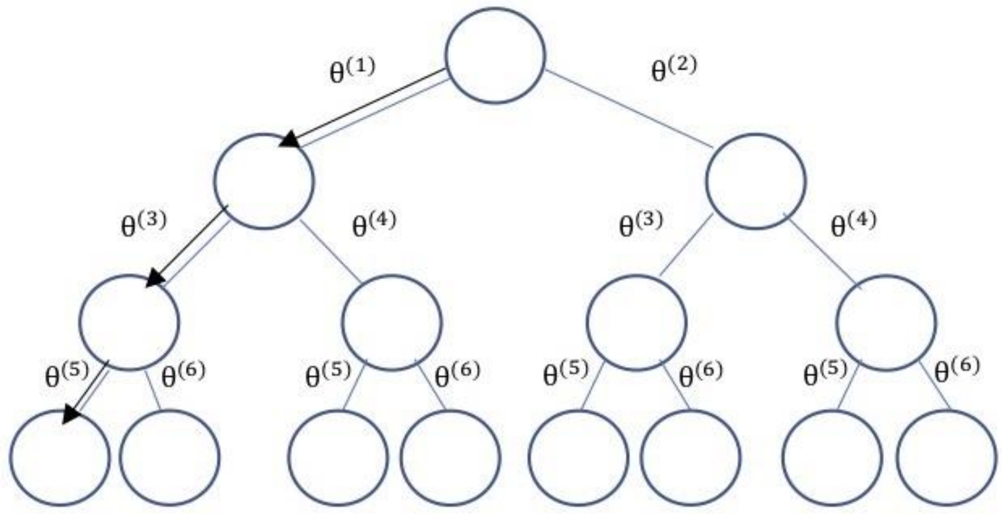
- **UCB-Laplace has been proved in [Abbasi-Yadkori *et al.*(2011)] using the Laplace method (method of mixtures for sub-Gaussian r.v.)**
- **Explicit control of success probability**
- **In UCT-Laplace, *B*-value of a children node is:**

$$B_{v',t} = \hat{\mu}_{v',t} + \sqrt{\frac{\left(1 + \frac{1}{N_{v',t}}\right) \ln\left(K\sqrt{N_{v',t} + 1}/\delta\right)}{2N_{v',t}}}$$

Experiments

- **Algorithms to be compared:**
  - **MAB: TS**
  - **Linear bandits: LinTS (tuned)**
  - **BATS: UCT and UCT-Laplace**
- **Experiments were run in different environments to verify the robustness, optimality, and fast-convergence of UCT-Laplace:**
  - **Two settings with *2 and 6 choices per factor***
  - **Linear and non-linear reward function**

Reward Function in Trees



$$\theta_\star = [\theta^{(1)}\ \theta^{(2)}\ \theta^{(3)}\ \theta^{(4)}\ \theta^{(5)}\ \theta^{(6)}\ ]$$

- **Feature vector:**

$$x_t = [1\ 0\ 1\ 0\ 1\ 0]$$

- **Linear reward:**

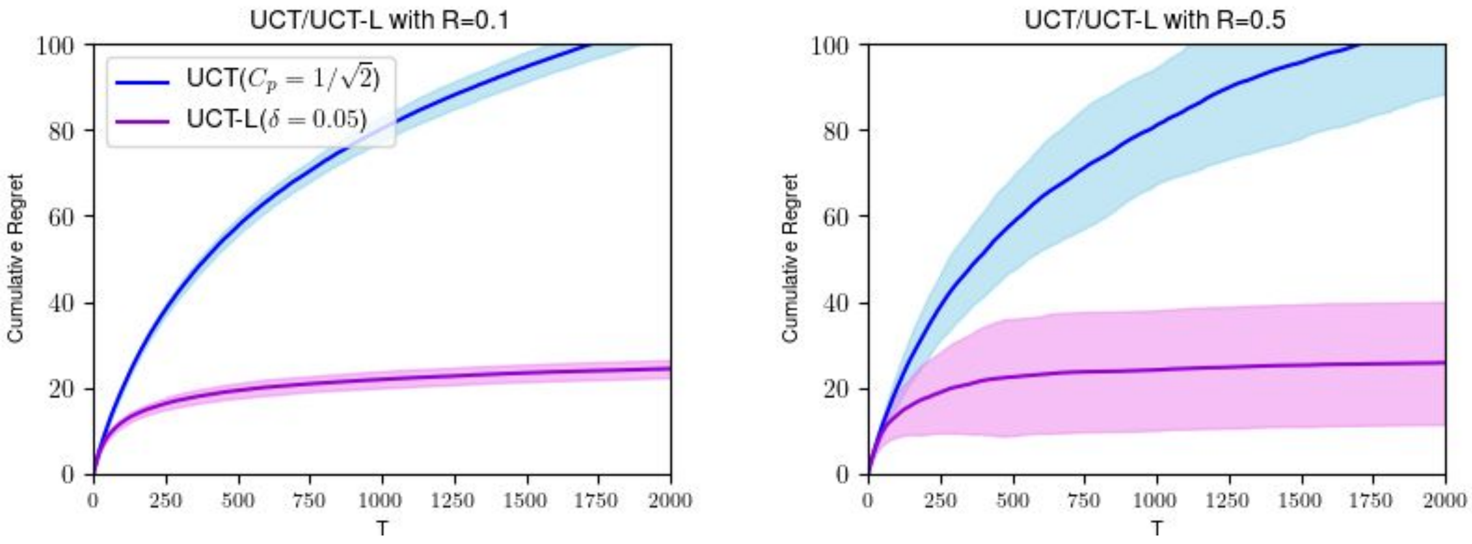
$$r_t = \langle \theta_\star, x_t \rangle$$

- **Max reward:**

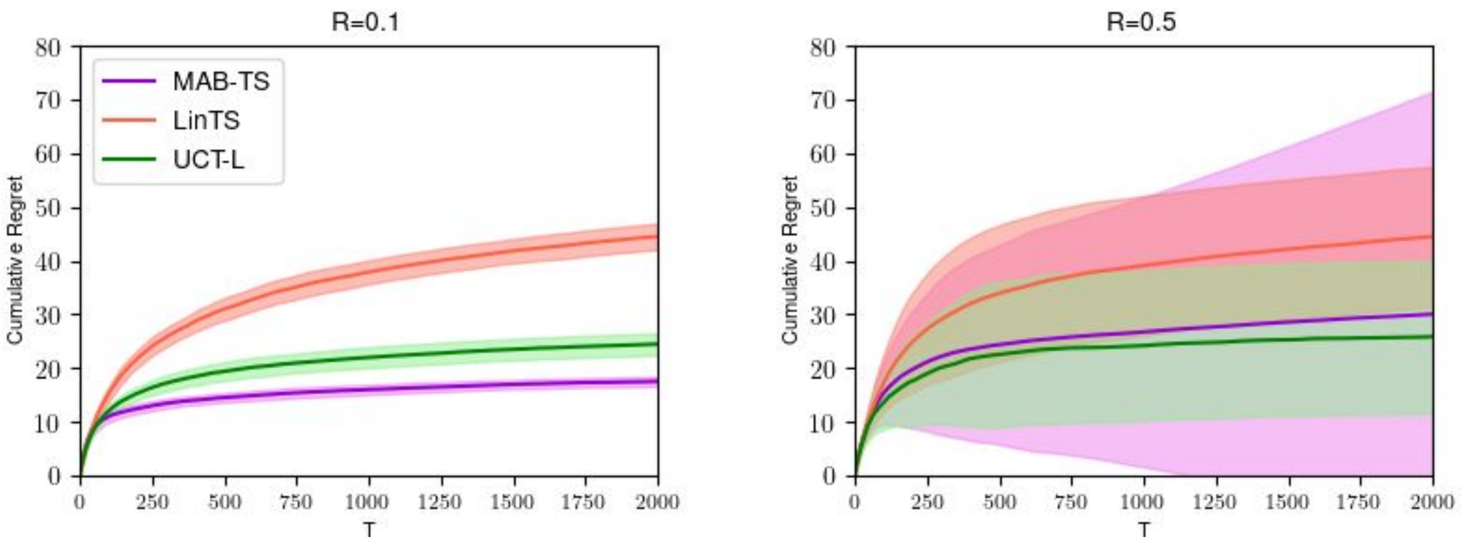
$$r_t = \max_{i \in \{1,2,\ldots,|\theta_\star|\}} \theta^{(i)} \cdot x_t^{(i)}$$

UCT vs UCT-L: ***2 x 2 x 2** factorial design*

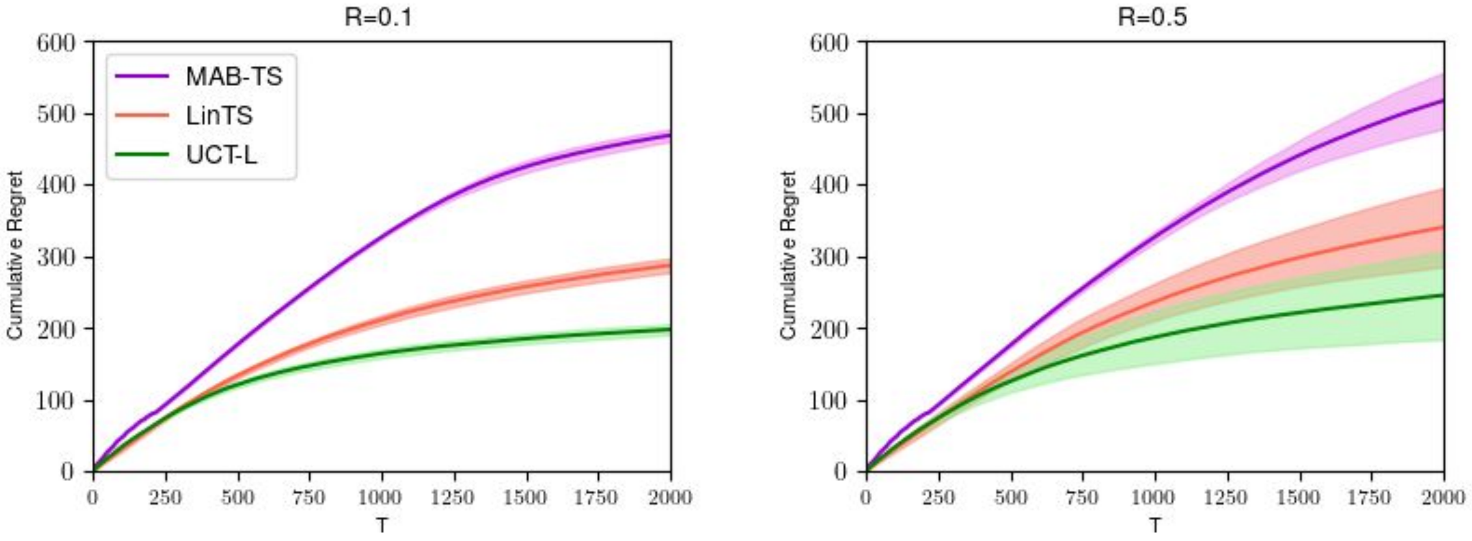
with ***Gaussian*** rewards and ***linear*** reward function



Results: *2 x 2 x 2 factorial design* with *Gaussian* rewards and *linear* reward function

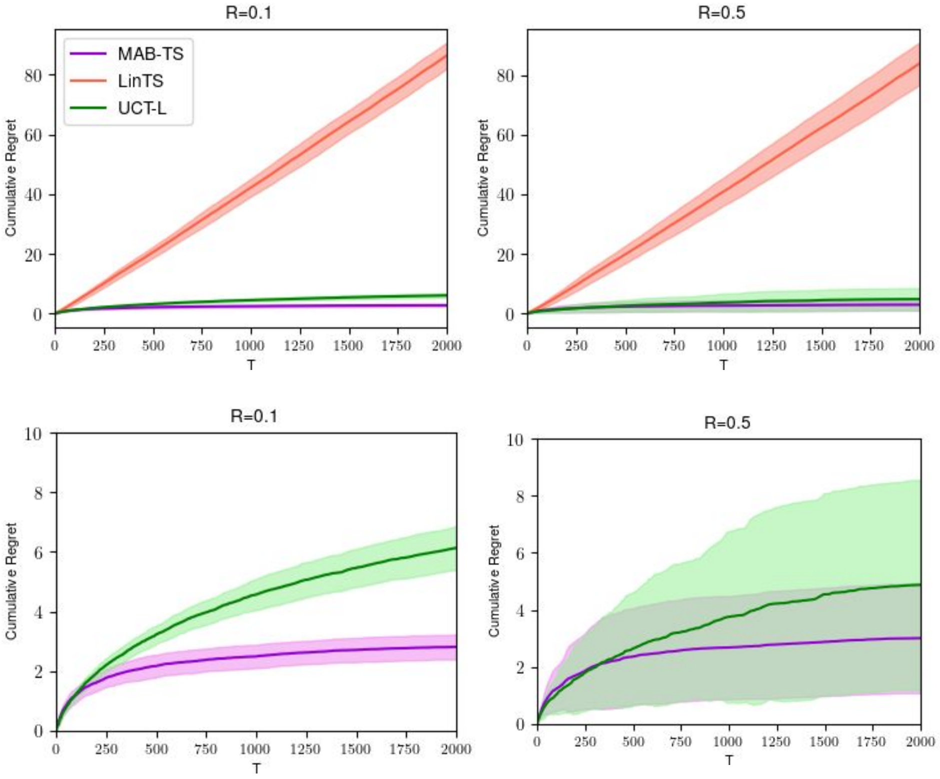


Results: *6 x 6 x 6 factorial design* with *Gaussian* rewards and *linear* reward function



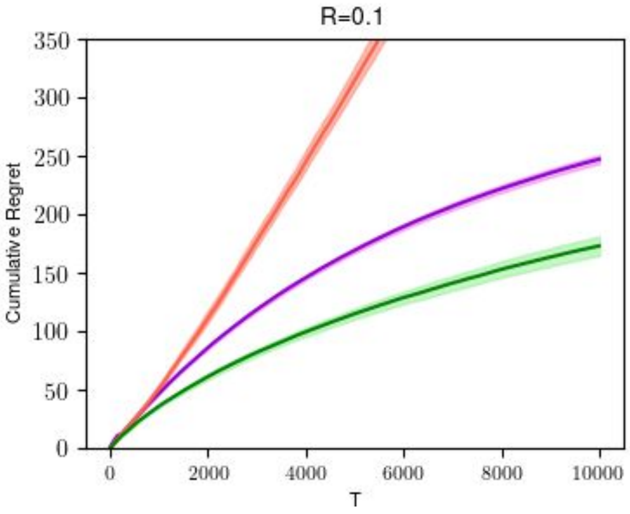
Results: ***2 x 2 x 2*** *factorial design*

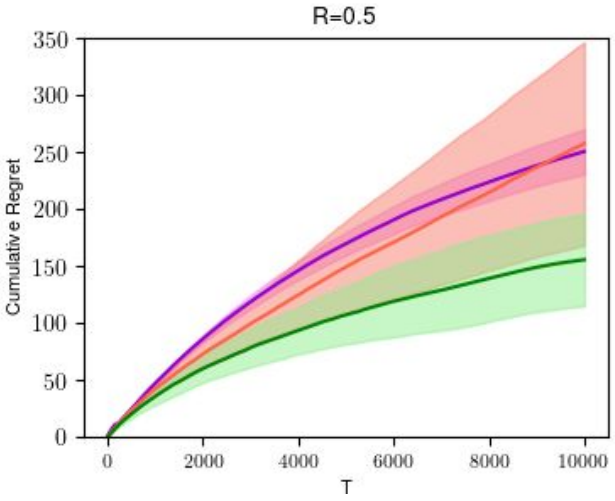
with ***Gaussian*** rewards and ***max*** reward function



Results: *6 x 6 x 6 factorial design*

with *Gaussian* rewards and *max* reward function





Future work

- Propose a randomized algorithm