POLITECNICO
MILANO 1863

Online Learning Applications Project

Dynamic Pricing

Authors:
Carminati Gabriele
Compagnoni Riccardo Domingo
De Introna Federico
Di Giore Francesco
Fossa’ Chiara

Academic Year 2024/2025

School of Industrial and Information Engineering
Master of Science - Computer Science

Project: Overview

The goal of the project is to design online learning algorithms to sell **multiple** types of products under **production constraints**.

Parameters

- Number of rounds T
- Number of types of products N
- Set of possible prices P (small and discrete set)
- Production capacity B (for simplicity, there is a total number of products B that the company can produce)

Buyer behavior

- Has a valuation vi for each type of product in N
- Buys all products priced below their respective valuations

At each round $t \in T$:

1. The company chooses which types of product to sell and set price pi for each type of product
2. A buyer with a valuation for each type of product arrives
3. The buyer buys a unit of each product with price smaller than the product valuation

<u>Requirement 1: Single product and stochastic environment</u>

Environment

- We simulate **a seller offering a single product**.

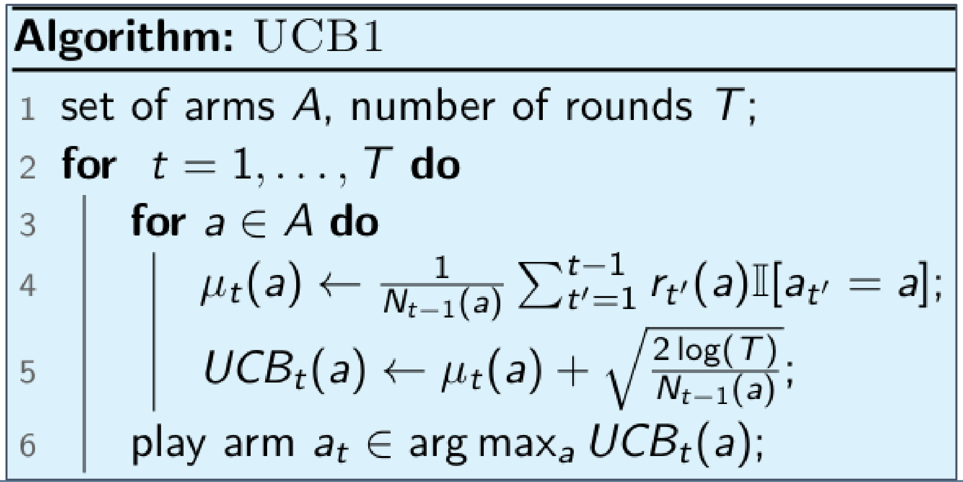
- Customers' **valuations** are random and drawn from distributions bounded in [0,1]:
    - Truncated Normal
    - Uniform
    - Beta
    - Truncated Exponential
- At each round:
    - Seller chooses a **price**
    - A customer arrives with a random valuation
    - If valuation ≥ price → product sold (d=1), else not sold (d=0)
    - **Reward**: $r_t = (p_t - cost) \cdot d$

Agent Case 1: UCB1 without inventory constraint

- Treats each discrete price as an **arm in a multi-armed bandit**.
- Algorithm:
  - First, try each price once.
  - Then, select price with the largest **Upper Confidence Bound (UCB)**:

$$UCB(p) = \bar{r}(p) + \sqrt{\frac{2 \cdot \log(T)}{N(p)}}$$

  - Learns the optimal price by balancing **exploration vs exploitation**.



Algorithm: UCB1

```
set of arms $A$, number of rounds $T$;
for $t = 1, \ldots, T$ do
    for $a \in A$ do
        $\mu_t(a) \leftarrow \frac{1}{N_{t-1}(a)} \sum_{t'=1}^{t-1} r_{t'}(a) \mathbb{I}[a_{t'} = a]$;
        $UCB_t(a) \leftarrow \mu_t(a) + \sqrt{\frac{2 \log(T)}{N_{t-1}(a)}}$;
    play arm $a_t \in \arg\max_a UCB_t(a)$;
```

Agent Case 2: UCB-like with inventory (budget) constraint

- Adds a constraint: limited number of products B.
- Extends UCB by:
    - Computing **UCB for utility** and **LCB for costs**.
    - Solving a **Linear Program (LP)** to choose a price distribution while respecting budget:

$$\max \sum_{p} \gamma(p) \cdot f(p) \quad s.t \sum_{p} \gamma(p) \cdot c(p) \le \rho, \qquad \sum_{p} \gamma(p) = 1$$

- Stops selling once budget is exhausted.

- Seller's utility: $f_t(p_t) = p_t 1[v_t \ge p_t]$
- Seller's cost: $c_t(p_t) = 1[v_t \ge p_t]$

$$f_{\text{ucbs}} = \text{avg\_f} + \text{range} \cdot \sqrt{\frac{2\ln(T)}{N_{\text{pulls}}}}$$

$$c_{\text{lcbs}} = \text{avg\_c} - \text{range} \cdot \sqrt{\frac{2\ln(T)}{N_{\text{pulls}}}}$$

$$\begin{aligned} \text{maximize } & f_{\text{ucbs}}^{\top}\gamma \\ \text{subject to } & c_{\text{lcbs}}^{\top}\gamma \le \rho, \\ & \sum_{i=1}^{K} \gamma_i = 1, \\ & 0 \le \gamma_i \le 1, \quad i = 1, \ldots, K \end{aligned}$$

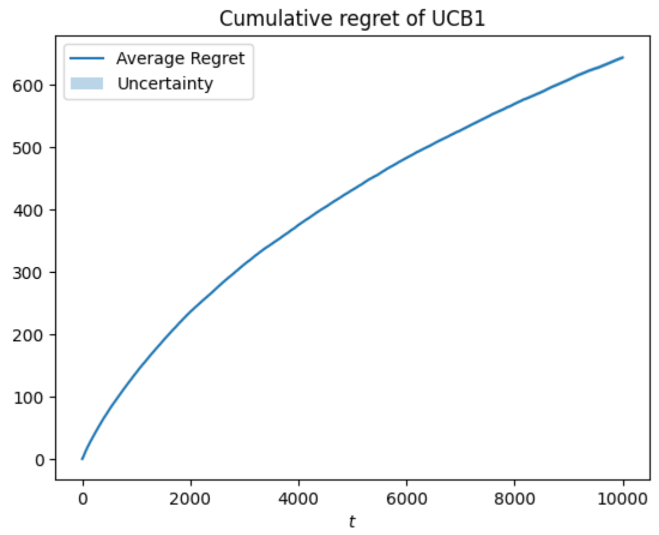
Baseline (Clairvoyant)

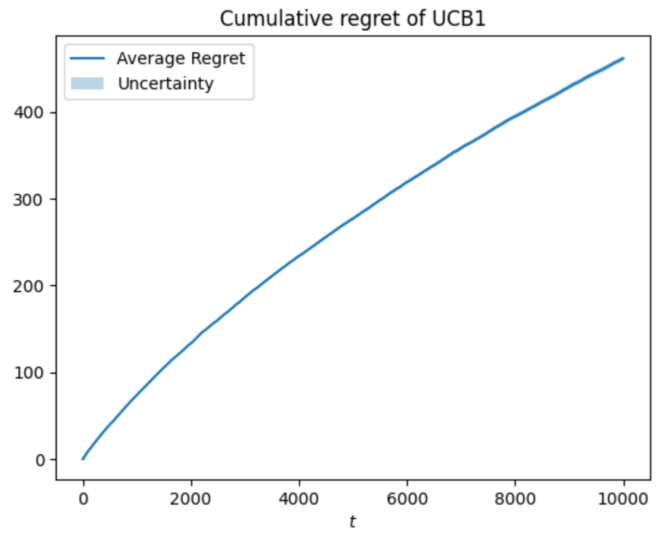
- Knows the **true distribution of valuations**.
- Chooses the price (or mix of prices) that **maximizes expected profit** under budget constraints.
- Computed by solving an LP.
- Serves as the **upper bound** against which we compare regret.

$$
\begin{aligned}
\text{maximize} \quad & \sum_i \gamma_i(\text{price}_i - \text{cost}) \cdot \text{win\_probability}_i \\
\text{subject to} \quad & \sum_i \gamma_i \cdot \text{win\_probability}_i \leq \rho, \\
& \sum_i \gamma_i = 1, \\
& 0 \leq \gamma_i \leq 1 \quad \forall i
\end{aligned}
$$

N.B.: cost is {0,1}

Trials and graphs without budget constraints





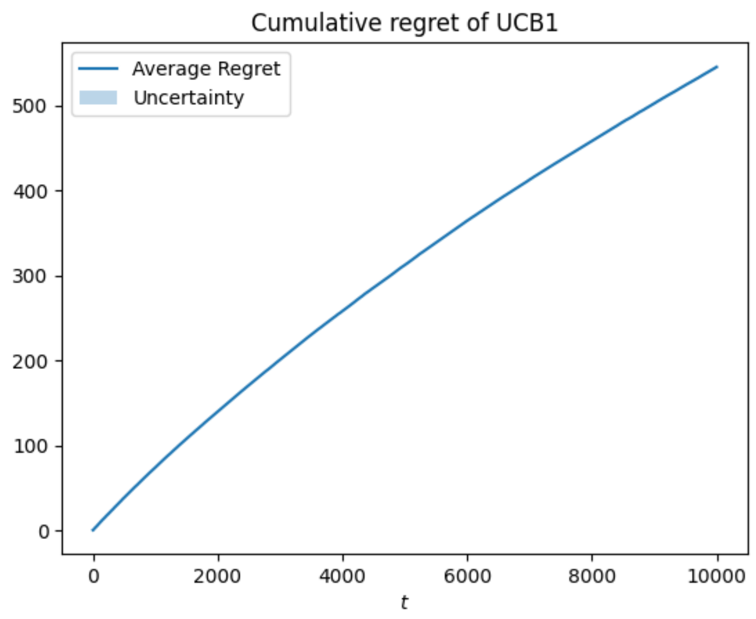
Truncated Normal Distribution

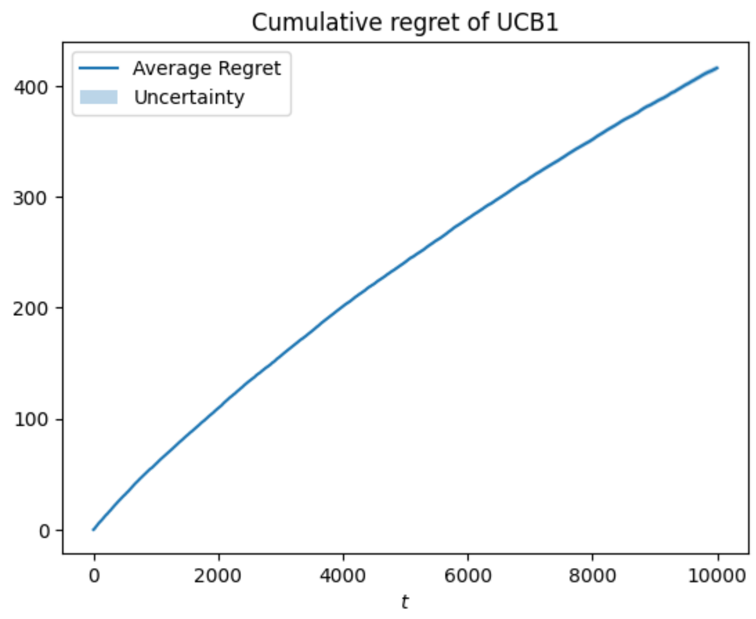
- **Cumulative Regret**: grows sublinearly → UCB1 converges.
- **Number of Pulls**: most mass on the optimal price, with some exploration.

Uniform Distribution

- **Cumulative Regret**: sublinear, but noisier.
- **Pulls**: spread across several prices → higher exploration due to uncertainty.

Trials and graphs without budget constraints





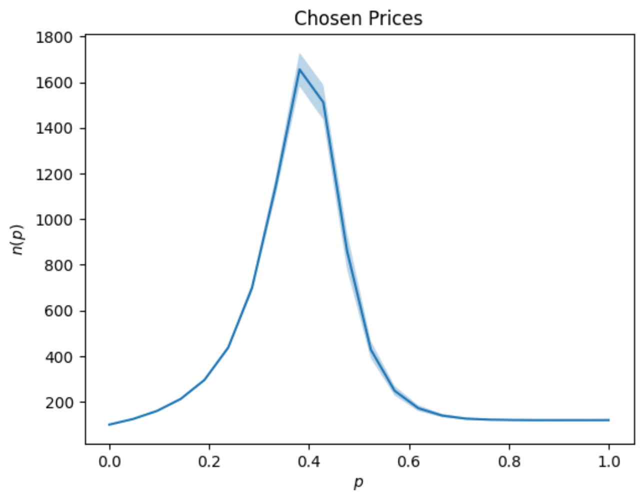
Beta Distribution

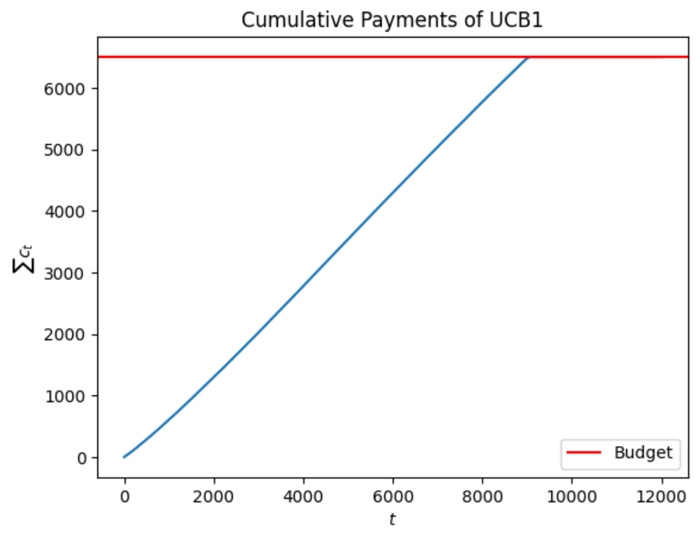
- Optimal price is lower (since mean valuation is low).
- Agent converges well: pulls mostly the optimal lower prices.

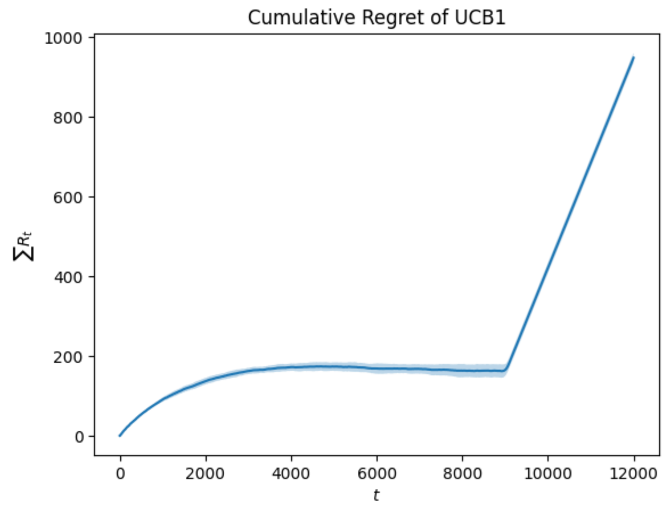
Truncated Exponential

- Optimal price shifts higher compared to Beta.
- Again, UCB1 identifies the best price and regret stays sublinear.

Trials and graphs with budget constraints

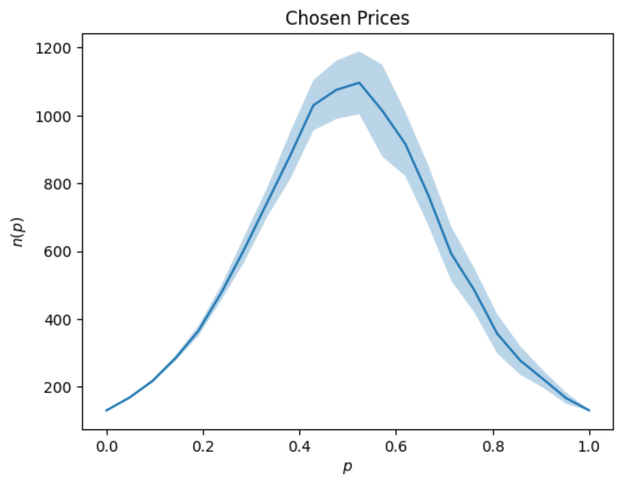


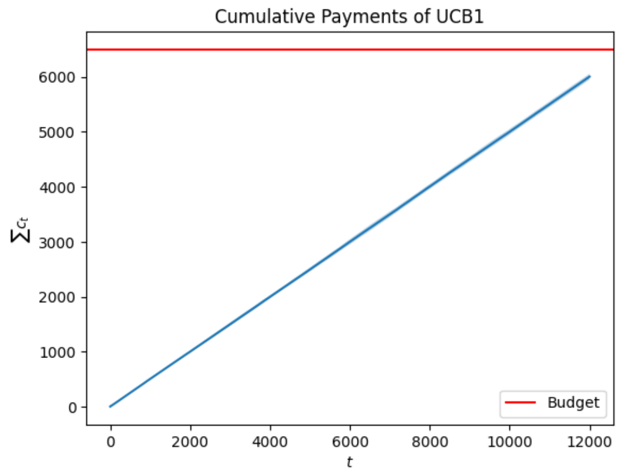


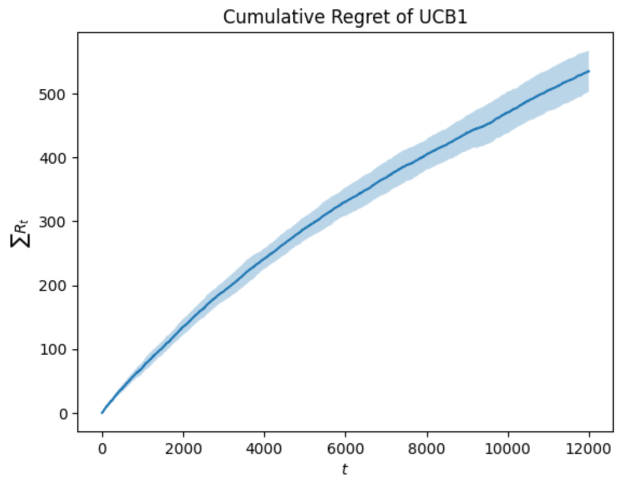


Normal Distribution
Chosen Prices: Agent concentrates on 2 prices near 0.5
Cumulative Payments: Grows steadily → exactly stops at budget.
Cumulative Regret: Sublinear, then flattens once budget is exhausted.

Trials and graphs with budget constraints







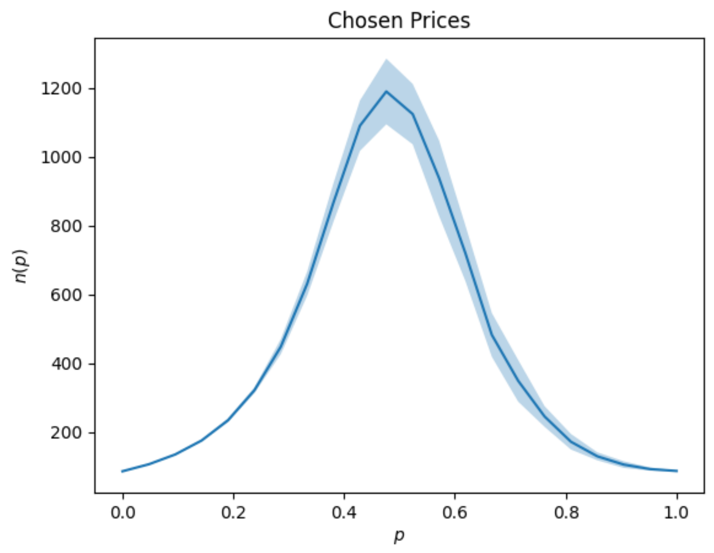
Uniform Distribution

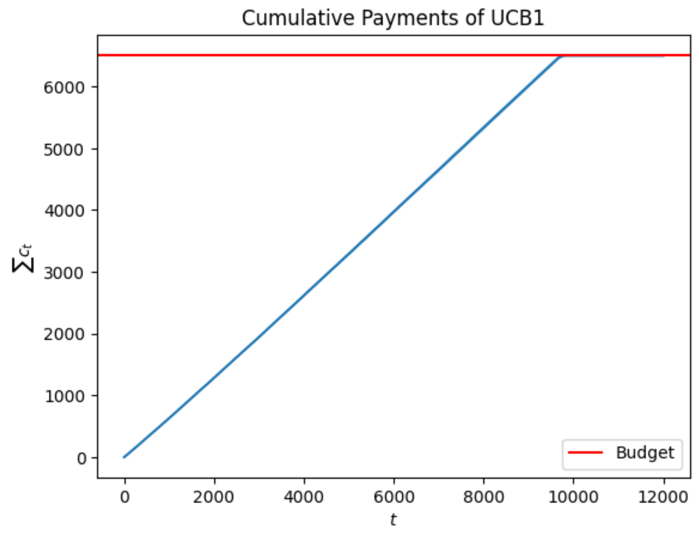
Agent does not always spend the full budget.

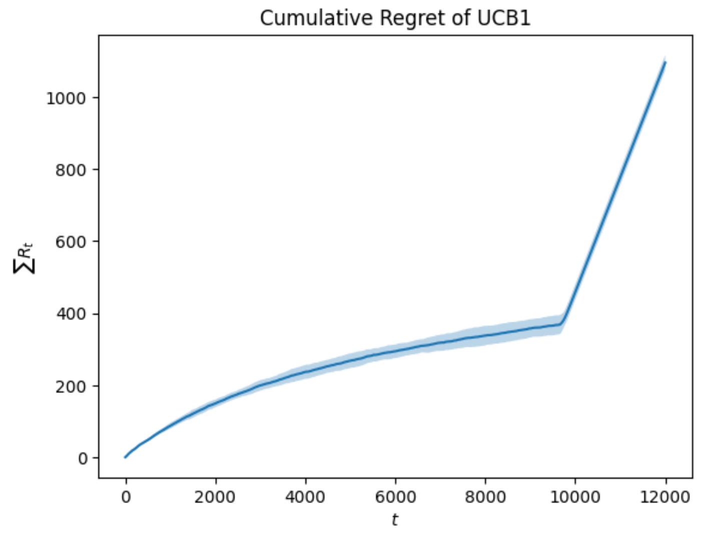
Pulls spread over multiple prices → higher variability.

Regret grows with wider uncertainty.

Trials and graphs with budget constraints





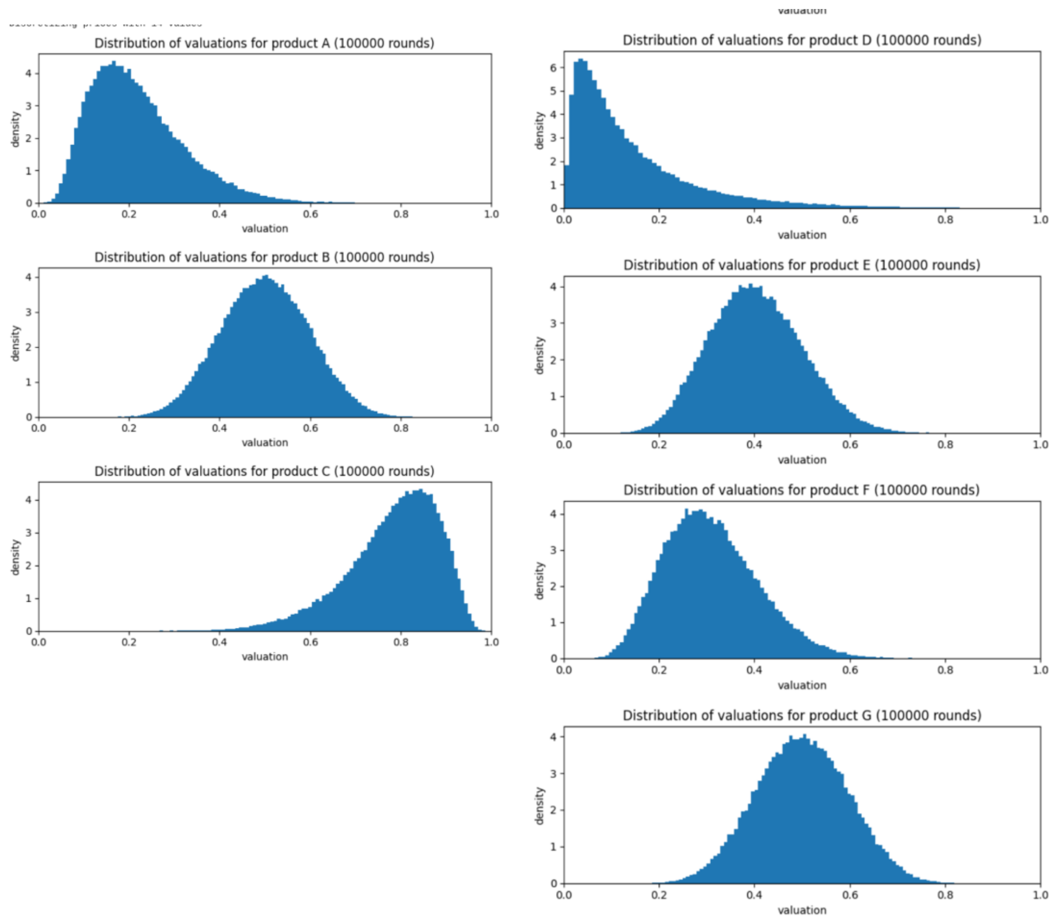


Beta Distribution
Like normal case, budget is fully used.
Optimal prices are lower

Requirement 2: Multiple products and stochastic environment

Environment

- We simulate a **seller offering multiple products**.
- Customer's **valuations** are modeled with **joint distribution** over all the products.
  - Specifically we used a **multivariate logit normal distribution**
  - Approximating the latent gaussian parameters with the **delta method**
- At each round:
  - The seller chooses a **price for each product**
  - A customer arrives with a **valuation** for each product sampled from the distribution
  - If **valuation>price** for a given product a purchase happens (d=1)
  - For each product the reward is computed as **reward = price * d**, so the seller agent receives a cumulative reward of **sum(rewards)**



Agent: Combinatorial UCB with Gaussian Processes

- The agent uses a **GP with RBF kernel** to model each product's **demand curve**
- At each round the agent:
  - Estimates demand by computing for each product
    - **UCB:** to estimate the probability of the product to get sold at each price
    - **LCB:** to estimate the expected reward for each price
  - Optimizes budget by solving a linear problem
    - Check the image for the LP
    - Maximizes the reward without exceeding the cost of the per-round budget
  - **Samples a discrete action** from the optimal fractional solution
  - **Updates GPs** with observations from the environment

$$\max_{x \in \Delta} \sum_{i,p} r_{i,p}^{\text{ucb}} x_{i,p} \quad \text{s.t.} \quad \sum_{i,p} c_{i,p}^{\text{lcb}} x_{i,p} \le \rho_t, \ \sum_{p} x_{i,p} = 1 \ \forall i$$

Agent: Combinatorial UCB with Gaussian Processes

UCB: $r_{ucb}(x) = \left(\mu_t(x) + \beta_t \cdot \sqrt{\sigma_t(x)}\right) \cdot prices(x)$
LCB: $c_{lcb}(x) = \max(\mu_t(x) - \beta_t \cdot \sigma_t(x),\ \varepsilon)$
where x is the index of price, μ_t the mean estimated by the GP, σ_t the std, and beta the exploration term of the GP-UCB approach

The LCBs for the costs are calculated using the usual formula of the GP-UCB approach, aside from the clipping that ensures that all the prices can be given a positive probability by the LP solver.
The UCBs, instead, are calculated using the square root of sigma, because experimentally the bounds shrank too much after just a few samples with the classical formula.

Moreover, we didn't keep a separate GP for the reward, because it proved to be inefficient, since we know a priori that the reward is equal to cost*price.

Baseline: clairvoyant

- Clairvoyant **knows** the real **purchasing probabilities** of each product.
- By solving the following LP problem it choose the mix of price for each product that maximizes the reward:

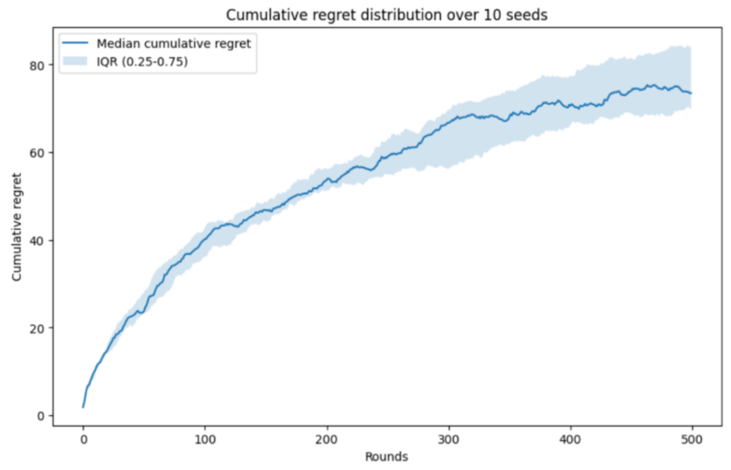
$$\begin{aligned}
\max_{X} \quad & \sum_{i=1}^{N}\sum_{j=1}^{K} q_i(p_j)\, p_j\, X_{i,j} \\
\text{s.t.} \quad & \sum_{i=1}^{N}\sum_{j=1}^{K} q_i(p_j)\, X_{i,j} \leq \rho \\
& \sum_{j=1}^{K} X_{i,j} = 1 \quad \forall i = 1, \dots, N \\
& X_{i,j} \geq 0
\end{aligned}$$

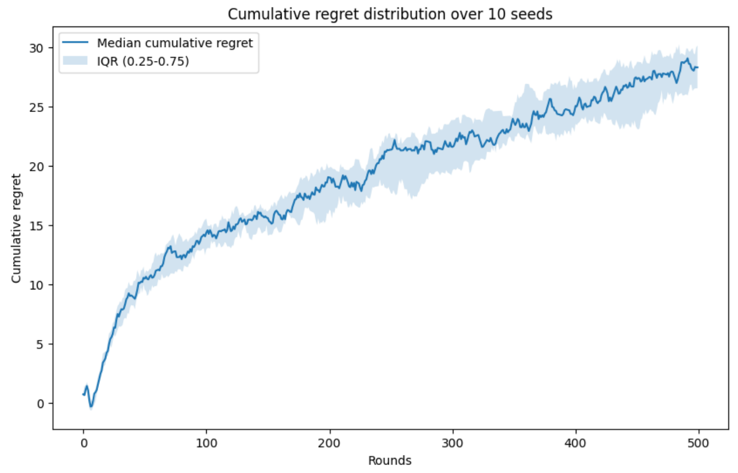
Where:

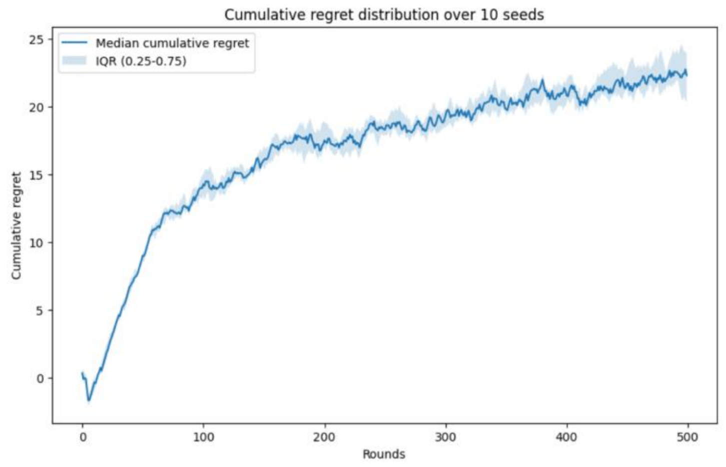
- *X_i,j*: Fractional probability of selecting price (p_j) for product (i)
- *p_j*: Candidate price for the (j)-th price level.
- *q_i(p_j)*: Probability that product (i) is purchased at price (p_j).
- *rho*: Maximum expected units that can be sold in a round (budget/pacing constraint).

N.B.: cost is {0,1}, so omitted in the constraint

Trials with uncorrelated products







With **B >> T:**

- The **median cumulative regret** over 10 trials is clearly **sublinear**

With **B = T:**

- The **median cumulative regret** over 10 trials is again **sublinear**

With **B << T:**

- The **median cumulative regret** over 10 trials is still **sublinear**

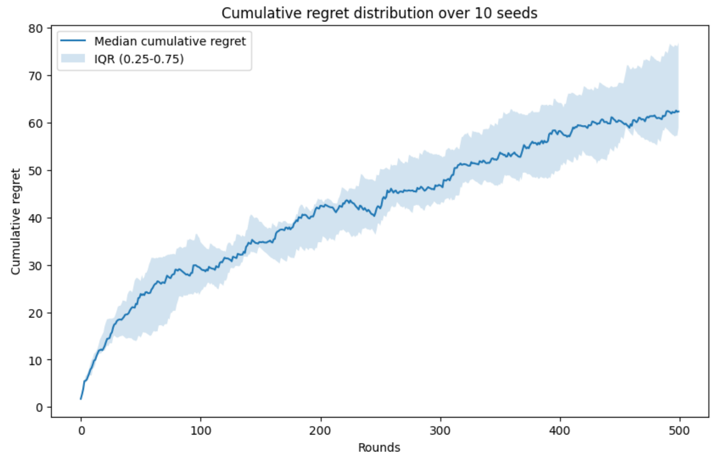
Trials with uncorrelated products

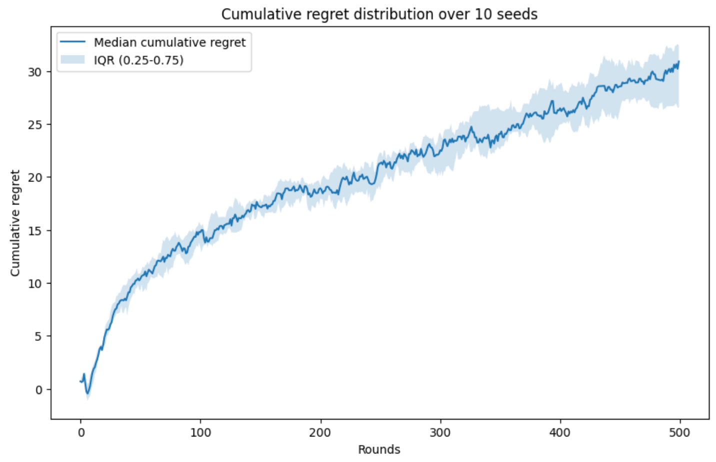
- From the previous graphs we can see that the cumulative regret is **sublinear** even in the worst case where the budget is small compared to T

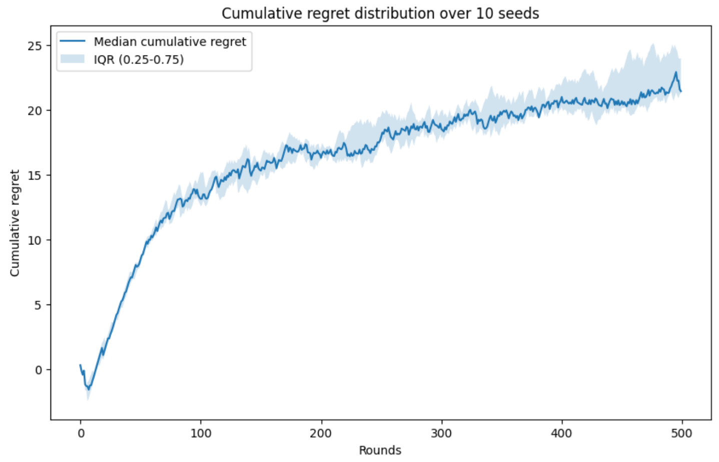
- We can observe that in the more complicated situations the regret tend to grow faster in the beginning bust still reach a sublinear growth at the end. This is likely due to less exploration constrained by the budget.

- In the more complicated cases the grow is also a bit **noisier** than in the most favorable one.

Trials with correlated products







With **B >> T:**

- The **median cumulative regret** over 10 trials is sublinear

With **B = T:**

- The **median cumulative regret** over 10 trials is again **sublinear**

With **B << T:**

- The **median cumulative regret** over 10 trials is still **sublinear**

Trials with correlated products

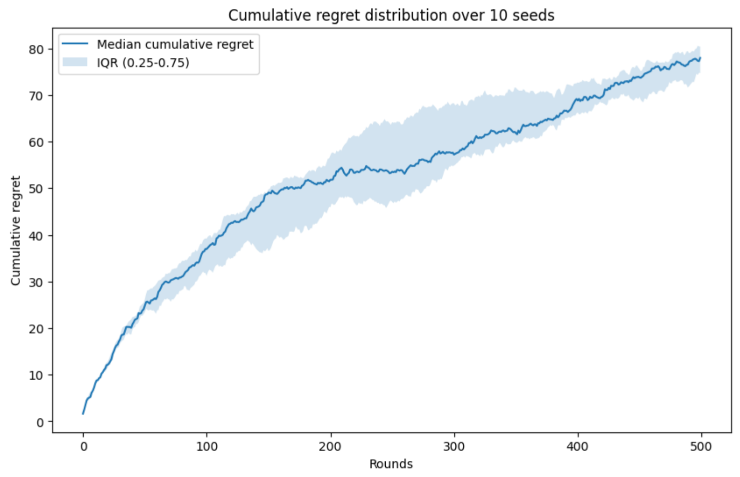
- As in the uncorrelated product case we can see from the graphs that the regret is **sublinear** even when the budget is really small.

- We can observe a similar trend to the uncorrelated product case where reducing the budget makes regret to grow faster in the first rounds and reach a sublinear growing later.

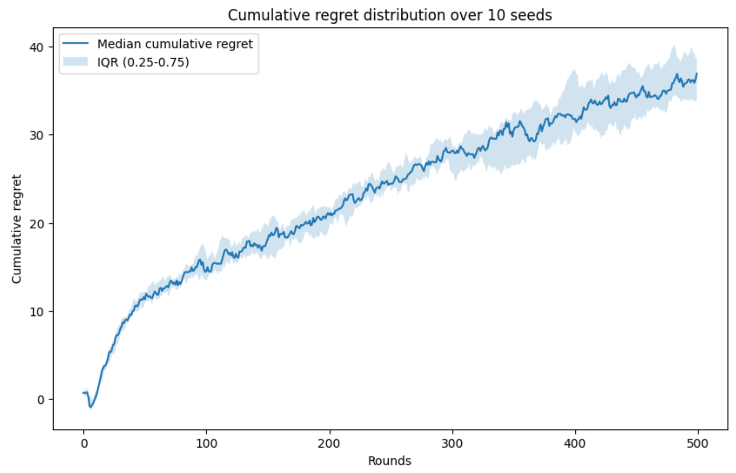
Trials with custom covariance matrix

Up to this point, we used a covariance matrix whose values outside the principal diagonal were all the same.

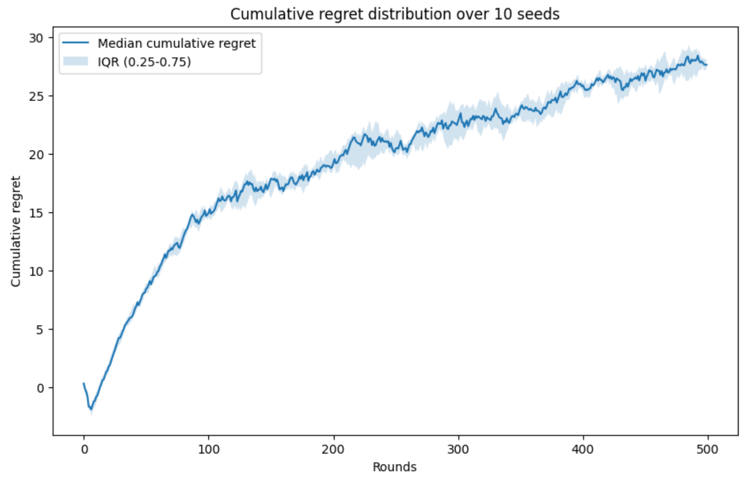
From now on, we use a covariance matrix whose values are different and there are **products that are correlated and products that are not correlated.**



With **B >> T**



With **B = T**



With **B << T**

Trials with custom covariance matrix

- Even in this case the agent is able to achieve **sublinear regret** in every situation, proving that is a consistent and effective learning agent in these scenarios.
- By the previous trials we can highlight how increasing product correlation leads to an increased variance in the regret, and how reducing the budget limits the agent exploration leading to steeper regret graphs in the initial rounds.

Requirement 3: Best-of-both-worlds algorithms with a single product

Requirement features

Dynamic pricing in rapidly changing markets requires algorithms that can:

- **Adapt** to non-stationary demand patterns.
- **Respect global budget limits** over long horizons.
- **Balance exploration and exploitation** without prior knowledge of distributions.

We test a **Primal-Dual agent** based on EXP3.P in multiple nonstationary valuation environments.

Nonstationary Pricing Environments

We model customer valuations with **three dynamic probability distributions** restricted to [0,1]:

- **Beta distribution**: allows flexible skewness and concentration, capturing diverse preference profiles.
- **Truncated Normal distribution**: captures valuations centered around a mean, with varying spread.
- **Truncated Exponential distribution**: produces heavily skewed valuations, either toward low or high willingness to pay.

Nonstationarity is introduced through random **mode changes** at every round:

- **Jump**: sudden reinitialization of parameters.
- **Drift**: gradual evolution, mimicking slow market trends.
- **Spike**: temporary concentration around extreme values.

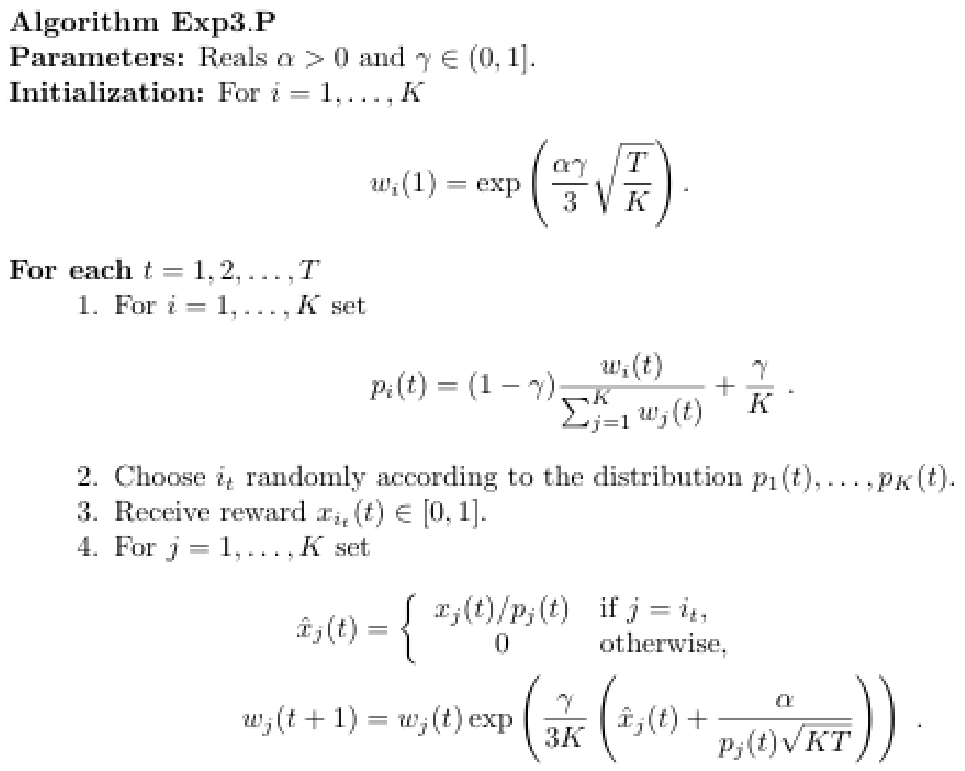
This creates highly volatile demand conditions where a static pricing strategy would fail.

Learning Agent

We design a **Primal-Dual EXP3.P agent**.

- **Primal step (EXP3.P):**
  - Maintains weights over possible prices.
  - Chooses prices with probabilities based on exponential weights.
  - Corrects bias and balances exploration (γ) with exploitation.
- **Dual step (budget pacing):**
  - Updates a **Lagrangian multiplier λ** via online gradient descent.
  - Penalizes overspending, steering the agent to respect average budget
  - Conservatism emerges naturally when λ increases.

This ensures the agent remains feasible while adapting to changes.



```
Algorithm Exp3.P
Parameters: Reals $\alpha > 0$ and $\gamma \in (0,1]$.
Initialization: For $i = 1, \dots, K$
    $w_i(1) = \exp\left(\frac{\alpha\gamma}{3}\sqrt{\frac{T}{K}}\right)$.
For each $t = 1, 2, \dots, T$
    1. For $i = 1, \dots, K$ set
        $p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K}$.
    2. Choose $i_t$ randomly according to the distribution $p_1(t), \dots, p_K(t)$.
    3. Receive reward $x_{i_t}(t) \in [0,1]$.
    4. For $j = 1, \dots, K$ set
        $\hat{x}_j(t) = \begin{cases} x_j(t)/p_j(t) & \text{if } j = i_t, \\ 0 & \text{otherwise,} \end{cases}$
        $w_j(t+1) = w_j(t)\exp\left(\frac{\gamma}{3K}\left(\hat{x}_j(t) + \frac{\alpha}{p_j(t)\sqrt{KT}}\right)\right)$.
```

Baseline

We compare against a **clairvoyant benchmark**:

- Knows the **true win probability** at each round.
- Chooses the **optimal distribution over prices** by solving a linear program:
    - Maximize expected reward (price – cost) × probability of sale.
    - Constrained by expected budget usage ≤ ρ.

This provides the **upper bound** on performance.

- **Regret** is defined as the gap between the clairvoyant and the agent.

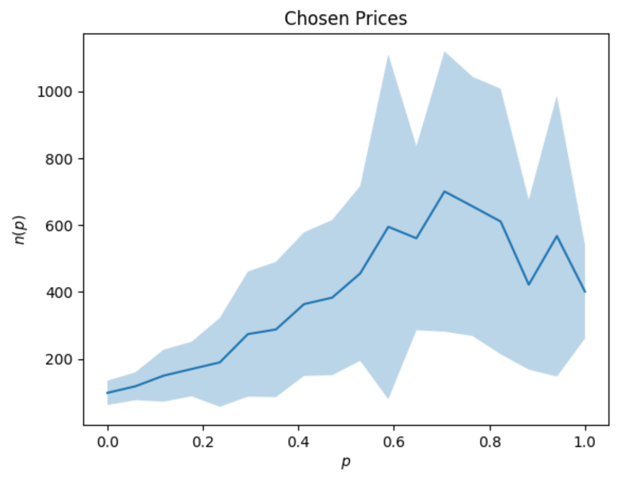
Experiment Setup

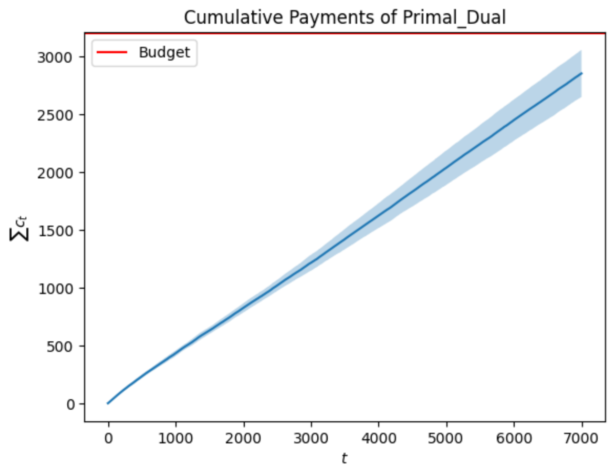
- **Horizon:** T = 7000 rounds.
- **Budget:** B = 3200 (average $\rho \approx 0.46$ per round).
- **Number of arms:** K = O(T^⅓) ≈ 18 price points in [0,1].
- **Agent step size (η):** T^(-½).
- **Trials:** 20 independent epochs with different random seeds.

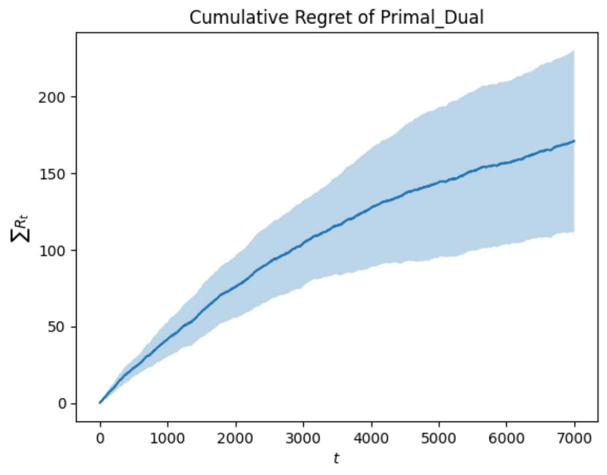
Metrics:

- **Chosen price distribution** (which prices the agent prefers).
- **Cumulative payments** (how budget is spent over time).
- **Cumulative regret** (distance to clairvoyant benchmark).

Results: Beta Environment





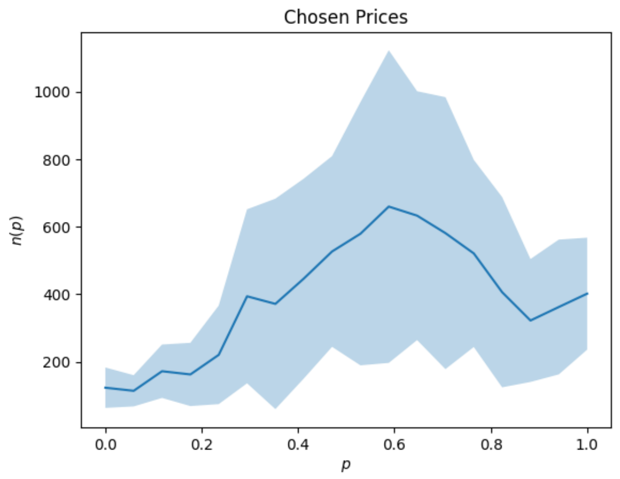


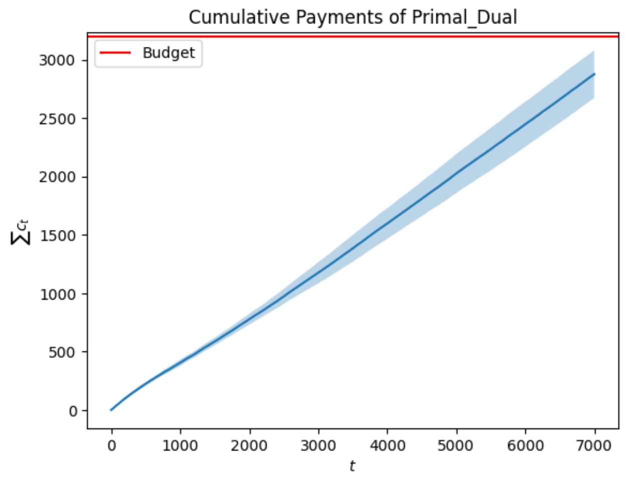
- **Chosen Prices:** Agent develops a strong preference around price ≈ 0.7. Variability remains high due to both nonstationarity and EXP3.P exploration. Peaks shift between runs, reflected in wide uncertainty bands.

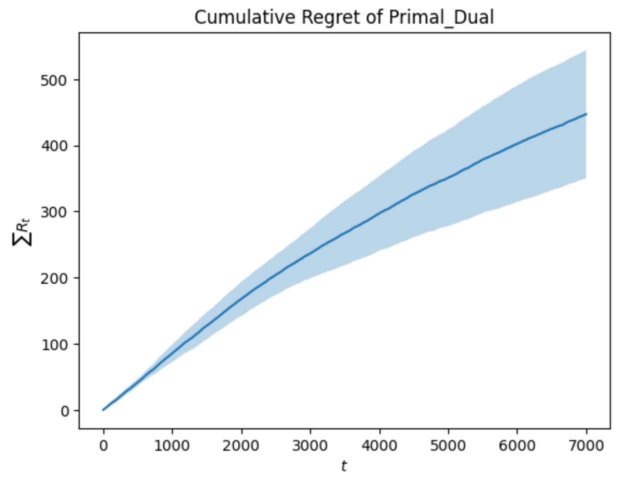
- **Cumulative Payments:** Payments grow sublinearly. Budget never fully exhausted, indicating the pacing mechanism prevents overspending but leaves resources unused.

- **Cumulative Regret:** Increases sublinearly, which aligns with no-regret theory. However, regret variance is large, suggesting sensitivity to environment fluctuations.

Results: Truncated Normal Environment





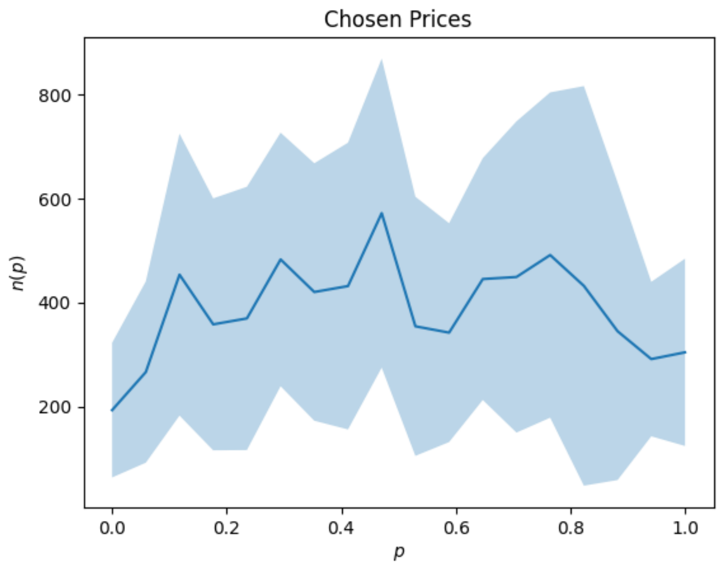


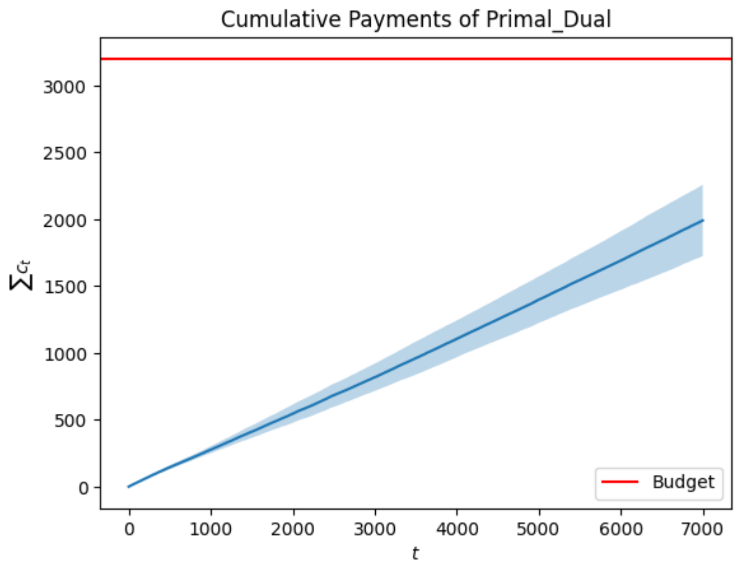
Chosen Prices: Clear concentration around one price peak (≈ 0.6–0.7). Still, large variance remains due to exploration and shifting optimal solutions. The agent frequently re-adjusts rather than locking onto a single value.

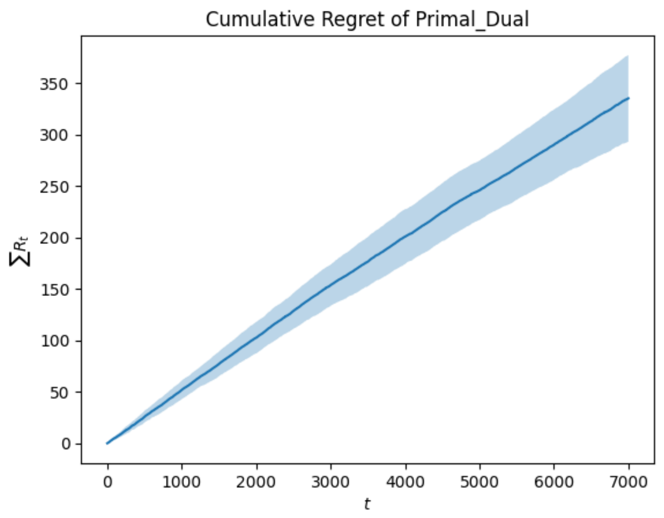
Cumulative Payments: Payments steadily rise but plateau well below the budget line. The agent avoids overspending but sacrifices potential revenue. Conservative budget pacing dominates.

Cumulative Regret: Sublinear growth. Regret per round decreases over time, confirming learning effectiveness. However, the widening uncertainty region reflects the instability of the clairvoyant benchmark in highly nonstationary settings.

Results: Truncated Exponential Environment







Chosen Prices: Unlike Beta or Normal, no dominant peak emerges. Prices are scattered, reflecting the steeply shifting nature of exponential valuations. Exploration dominates, with wide uncertainty bands.

Cumulative Payments: Extremely conservative. Payments stay far below budget, revealing inefficiency. The agent adapts to frequent low-valuations but non utilizes opportunities for higher sales.

Cumulative Regret: Nearly linear growth compared to other environments, but variability across runs is smaller. Sublinearity is still present but weaker, suggesting the environment is hardest to learn in.

Key Insights

Budget Safety: Agent never overspends, always satisfies global constraint.

Conservatism: Leaves a significant fraction of the budget unused, especially in Exponential environments.

Exploration Costs: Wide variance in chosen prices shows persistent exploration.

Performance Differences:

- Beta & Normal: identifiable dominant price.
- Exponential: scattered choices, harder to exploit.

Regret: Always sublinear, confirming theoretical guarantees, but higher in harsher environments.

The proposed algorithm demonstrates the **best-of-both-worlds** property: it is both **no-regret** and **budget-feasible**.

The pacing mechanism successfully prevents overspending but may **underutilize resources**, especially in exponential environments.

Requirement 4: Best-of-both-worlds with multiple products

Goal:

Design algorithms that adapt to **non-stationary multi-product pricing** environments.

Must balance:

- **Exploration vs Exploitation** (learn valuations while maximizing revenue).
- **Budget / resource constraints**.
- **Best-of-both-worlds**: near-optimal against stochastic and adversarial demand.

Environment: Multi-Product Pricing

Models consumer valuations as **logit-normal distribution**.

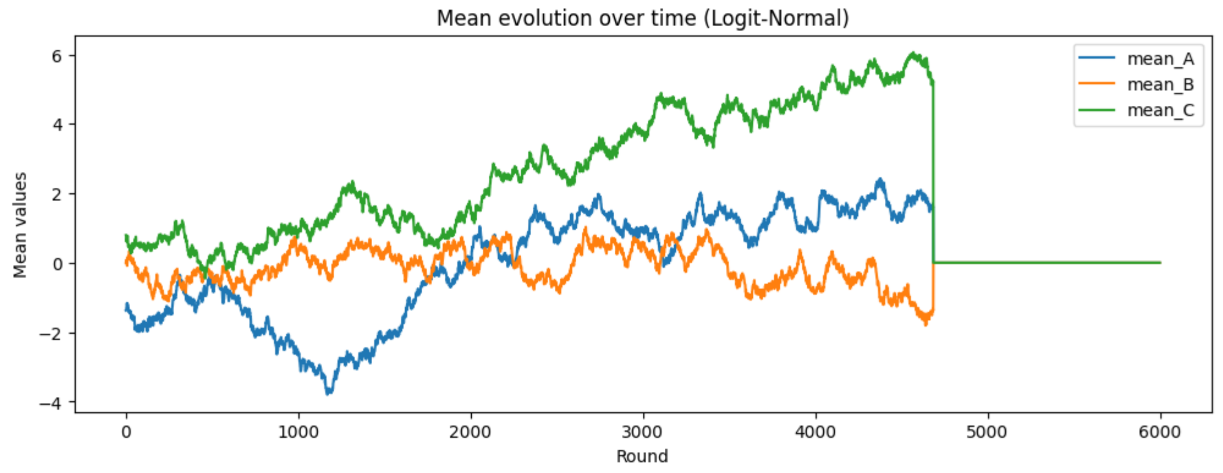
Supports **multiple correlated products**.

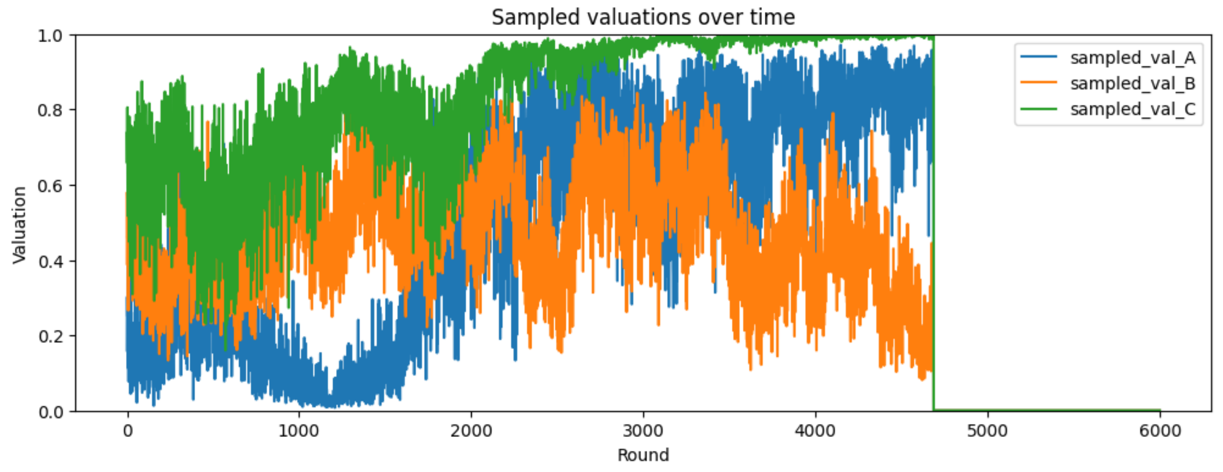
Key features:

- Randomized valuations per round.
- Valuations drift slowly over time (`update_means`).
- Consumer buys if valuation ≥ price.

Provides:

- `round()` → simulate purchases + revenue.
- `compute_Q()` → probability demand curve per price.





Agent: Discounted EXP3.P

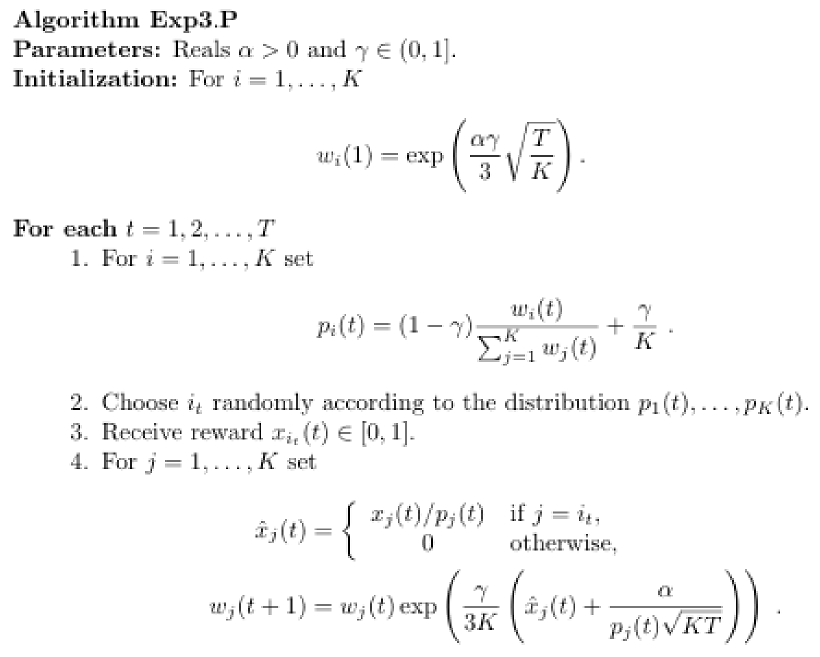
An **adversarial bandit algorithm** adapted for non-stationary environments.

Core mechanics:

- Maintains exponential weights over arms (prices).
- Uses **bias correction** for regret guarantees.
- Adds **discounting** → adapts to changing environments.

Outputs:

- Chosen **arm** (price).
- Updated **probabilities** over arms.



```
Algorithm Exp3.P
Parameters: Reals $\alpha > 0$ and $\gamma \in (0, 1]$.
Initialization: For $i = 1, \ldots, K$
    $w_i(1) = \exp\left(\frac{\alpha\gamma}{3}\sqrt{\frac{T}{K}}\right)$.
For each $t = 1, 2, \ldots, T$
    1. For $i = 1, \ldots, K$ set
        $p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K}$.
    2. Choose $i_t$ randomly according to the distribution $p_1(t), \ldots, p_K(t)$.
    3. Receive reward $x_{i_t}(t) \in [0, 1]$.
    4. For $j = 1, \ldots, K$ set
        $\hat{x}_j(t) = \begin{cases} x_j(t)/p_j(t) & \text{if } j = i_t, \\ 0 & \text{otherwise,} \end{cases}$
        $w_j(t+1) = w_j(t)\exp\left(\frac{\gamma}{3K}\left(\hat{x}_j(t) + \frac{\alpha}{p_j(t)\sqrt{KT}}\right)\right)$.
```

Weights are multiplied by a discount factor

Agent: Primal-Dual Pricing

Wraps multiple EXP3.P agents (one per product).

Uses **dual variable λ** to enforce budget constraints.

Decision process:

1. Each product chooses price via its EXP3.P agent.
2. Agent updates weights based on **Lagrangian reward** (revenue – λ × constraint).
3. Updates λ to balance **budget vs. revenue**.

Provides:

- Near-optimal pricing under global constraints.
- History of price distributions, λ evolution.

Baselines for Evaluation

Clairvoyant LP:

- Solves linear program with full knowledge of demand.
- Used as benchmark upper bound.

Best fixed hindsight distribution:

- Computes optimal fixed pricing strategy for entire horizon.
- Provides another regret baseline.

Experiment Setup

Parallelized trial runs (`joblib`, multiprocessing).

Progress tracked with `tqdm`.

Each trial simulates:

- **Multi-product environment.**
- **Agent interaction** (Primal-Dual EXP3.P).
- **Logging**: rewards, payments, regrets.

Visualizations:

- Regret vs horizon.
- Payments vs budget.
- Evolution of valuations and λ.

Trial Setup: Budget < Horizon

- Parameters:
  - Products = {A, B, C}
  - Budget = 4000, Horizon = 6000 rounds
  - Seeds: 20 randomized runs
  - Price discretization: ~17 levels between 0.01 and 1.0
- Two modes:
  - **Single Trial (run_one = 1):** visualization of one trajectory.
  - **Multiple Trials (run_one = 0):** averages + confidence bands.

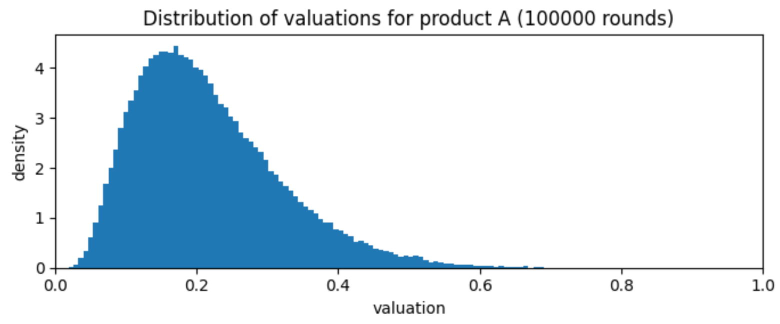
Single Trial Results

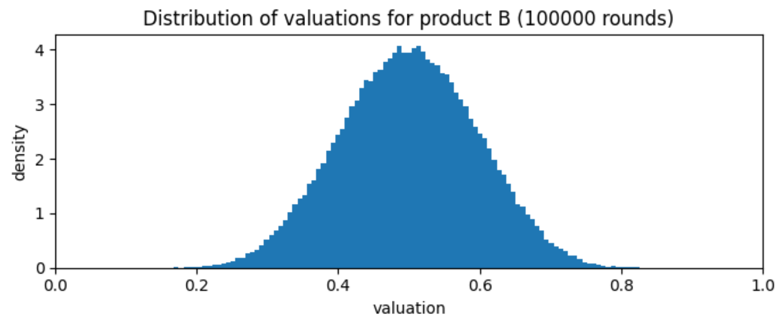
Observed Sales:

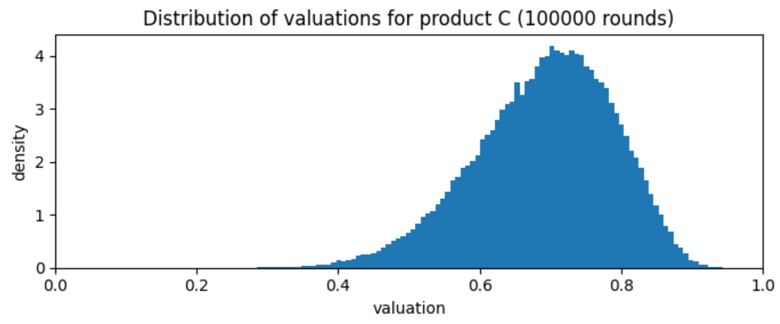
- Product C sold most (highest mean valuation).
- Product B sold moderately.
- Product A rarely sold (low mean valuation).
- Sometimes multiple products sold in the same round.

Regret Behavior:

- Regret vs clairvoyant oracle is consistently lower.
- Regret vs fixed hindsight oracle decreases slowly but steadily.
- Sublinear growth → algorithm adapts effectively.







Single Trial Visuals

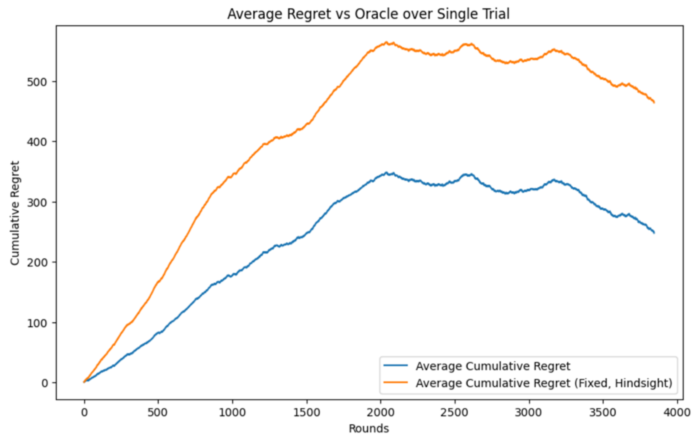


Figure 1: **Cumulative regret vs oracle**

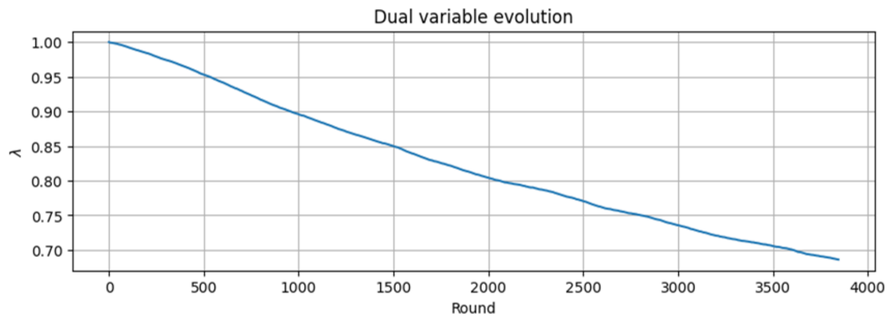
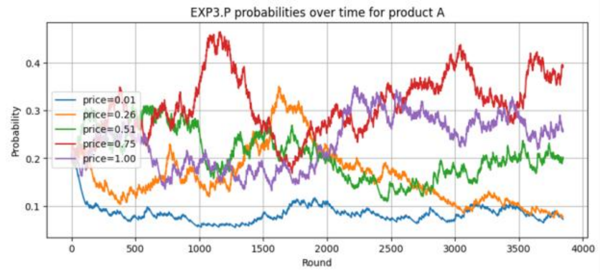
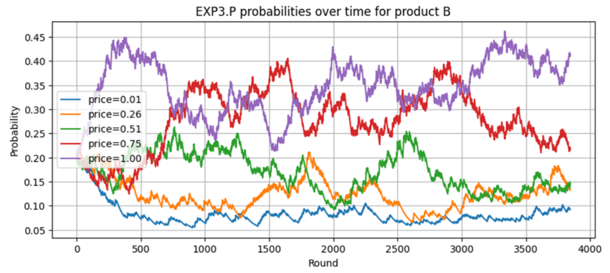


Figure 2: **Dual variable λ evolution**





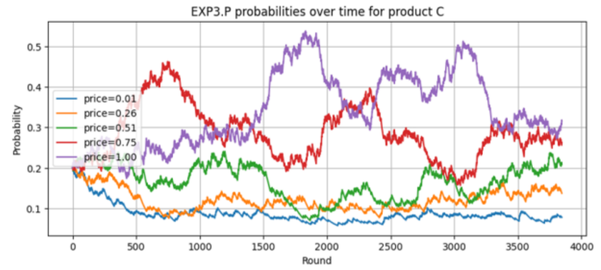


Figure 3–5: **EXP3.P price probabilities over time for each product**

Multiple Trials

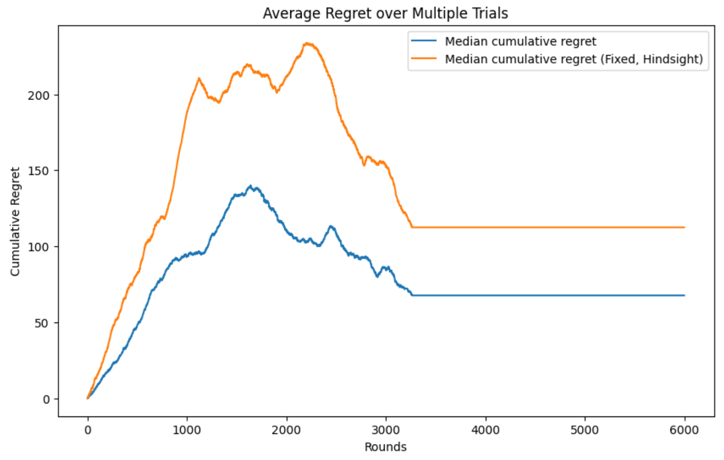
Average over 20 seeds.

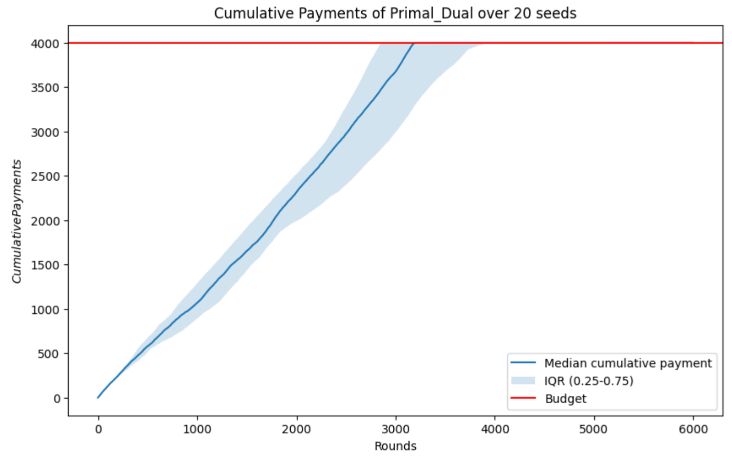
Regret:

- Sublinear and lower vs clairvoyant oracle.
- Sublinear vs fixed hindsight oracle.

Revenue / Payments:

- Average cumulative payments track budget.
- Variability across runs modest.





Price Discretization Sensitivity

Case 1: ~17 prices per product (fine grid)

- More exploration, higher computational cost.
- Smooth regret curves, good adaptation.

Case 2: 5 fixed prices per product (coarse grid)

- Less granularity, but still learns profitable prices.
- Sublinear regret preserved.
- Revenue still dominated by product C > B > A.

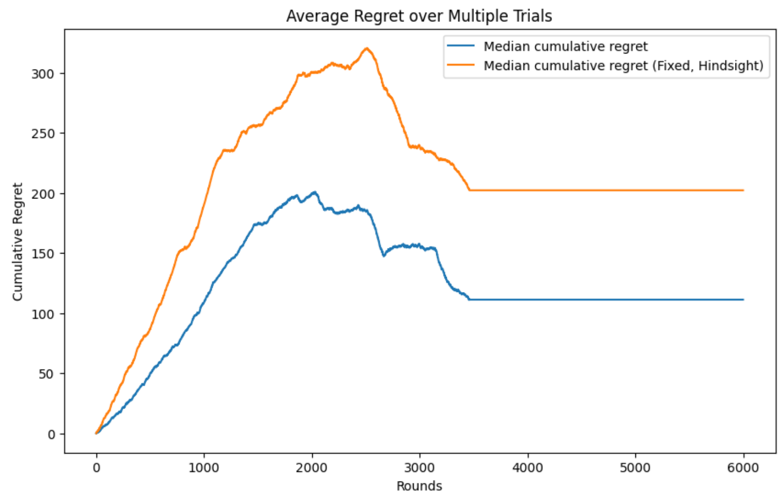
Revenue and Regret Comparison

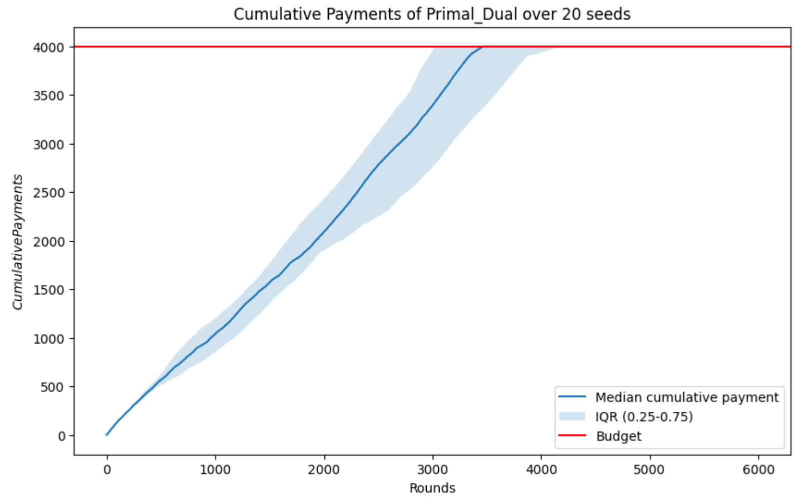
Revenue grows steadily, dominated by high-valuation product C.

Regret analysis:

- **Fixed hindsight oracle** = hard benchmark → higher regret.
- **Clairvoyant oracle** = easier benchmark → lower regret.

Sublinear regret growth = hallmark of good bandit learning.





Key Takeaways

Primal-Dual + Discounted EXP3.P adapts to **non-stationary, multi-product, budget-constrained pricing**.

Sublinear regret vs oracles confirms effectiveness.

Discounting not so relevant for tracking drifting valuations.

Works **robustly** across:

- Single vs multiple trials.
- Different price discretizations.

Case: Budget = Horizon (B = T)

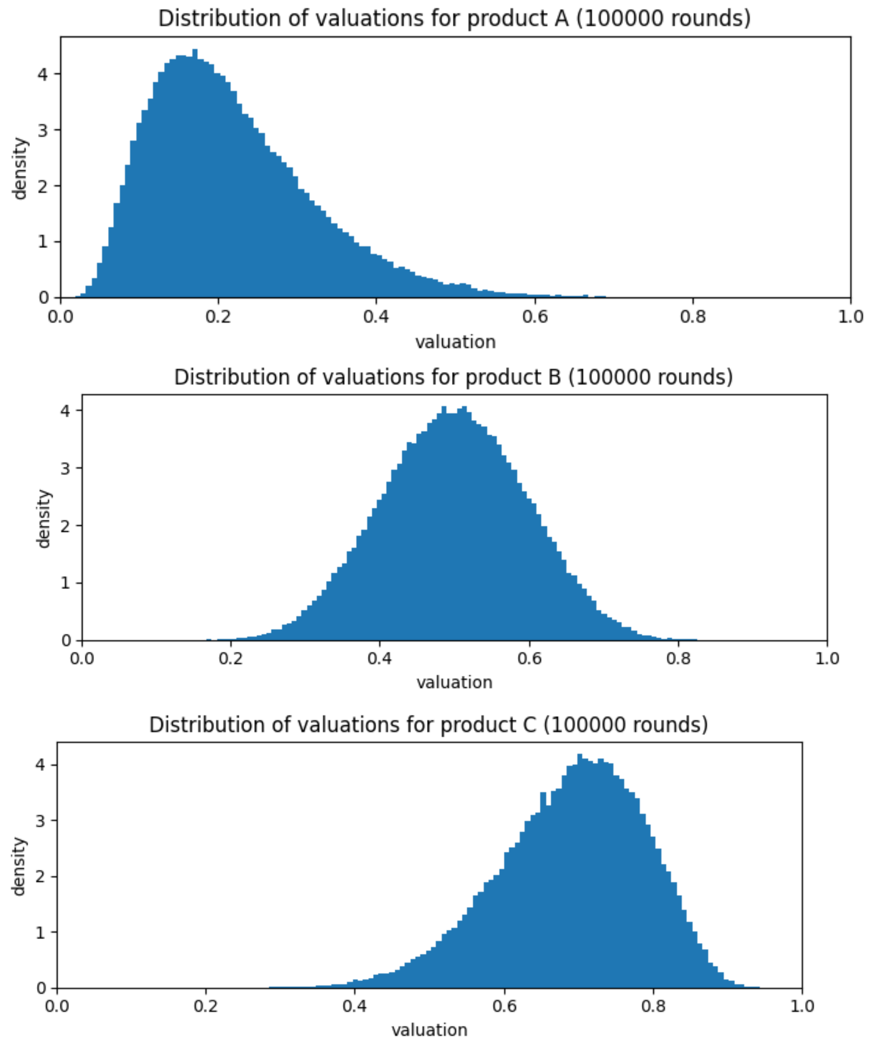
Setup:

- **Budget** = 6000, **Horizon** = 6000 rounds
- **Products** = {A, B, C}
- **Price discretization**: fine grid (~17 prices) or coarse (10 prices)

Goal: Compare learning behavior under matched budget and horizon.

Methodology:

- Single trial (visualization).
- Multiple trials (averaged results).



Single Trial (Fine Grid)

Agent gradually learns profitable prices.

Early variability: high prices sometimes exceed valuations → no sales.

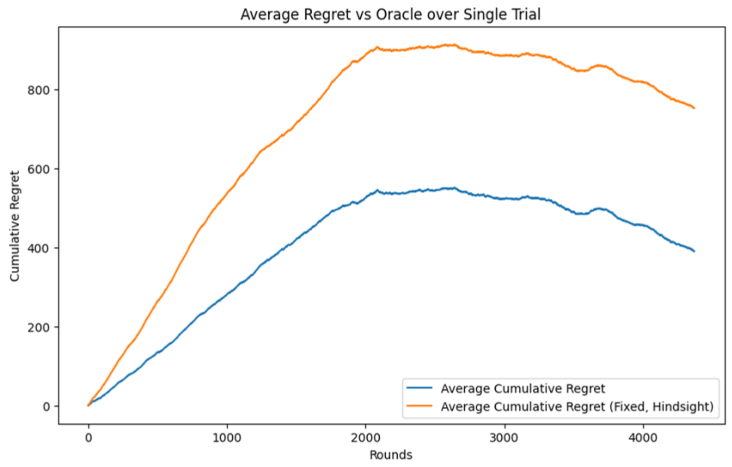
Over time:

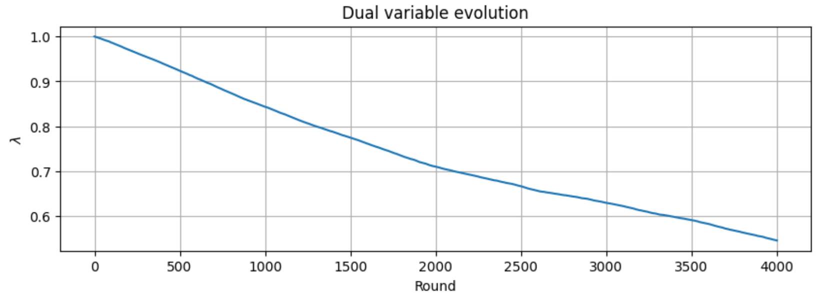
- Product C and B have a price that is distinguishable from the others
- A has more uniformly probable prices

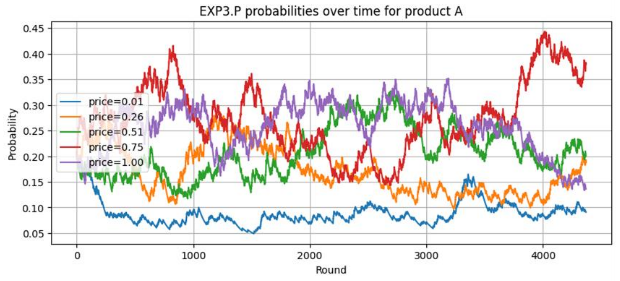
Cumulative regret:

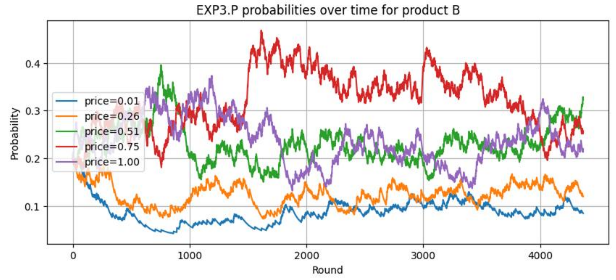
- Sublinear and lower vs clairvoyant oracle.
- Sublinear vs fixed hindsight oracle.

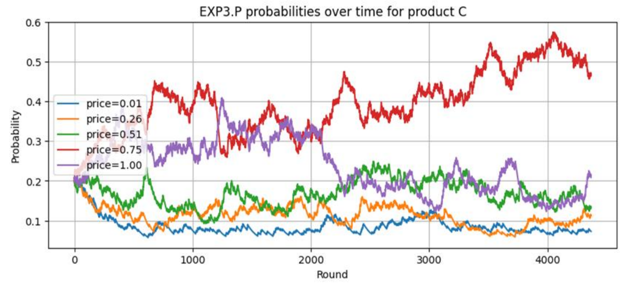
Discretization (17 prices) balances exploration and convergence.











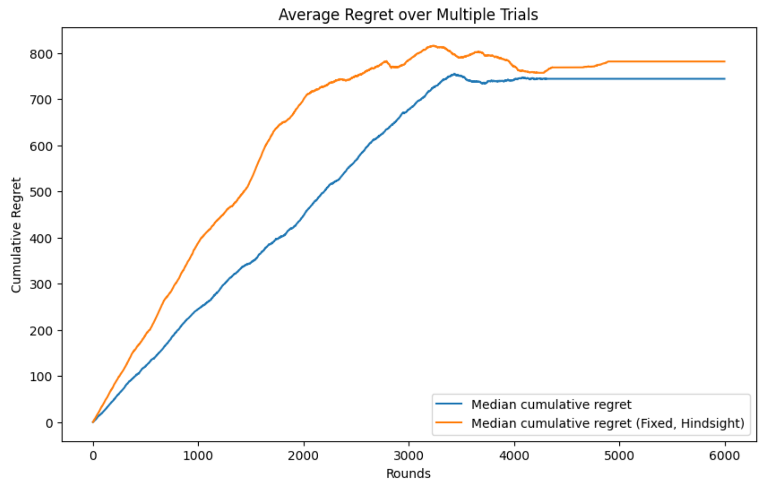
Multiple Trials (Fine grid)

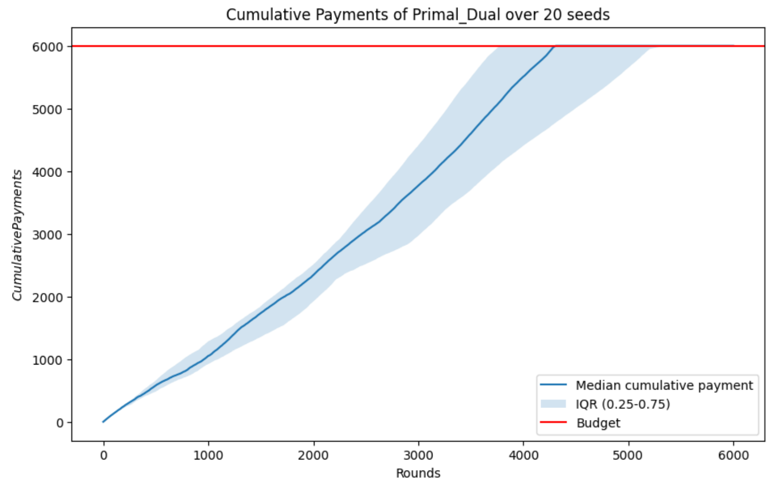
Across 20 seeds:

- Consistent adaptation across trials.
- Sublinear regret confirmed.
- Fixed hindsight benchmark harder to track → higher regret.

Payments analysis:

- Average cumulative payments approach budget.
- Variability across runs smoothed out.





Multiple Trials (Coarse Grid: 5 Prices)

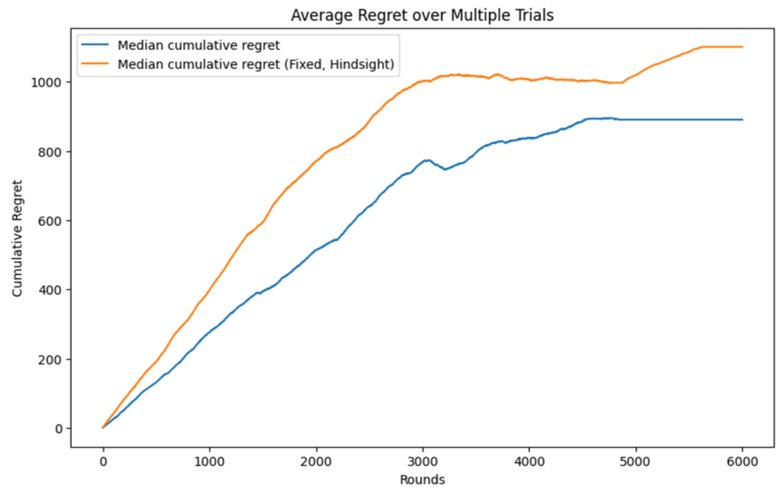
Restricts pricing to 5 discrete levels.

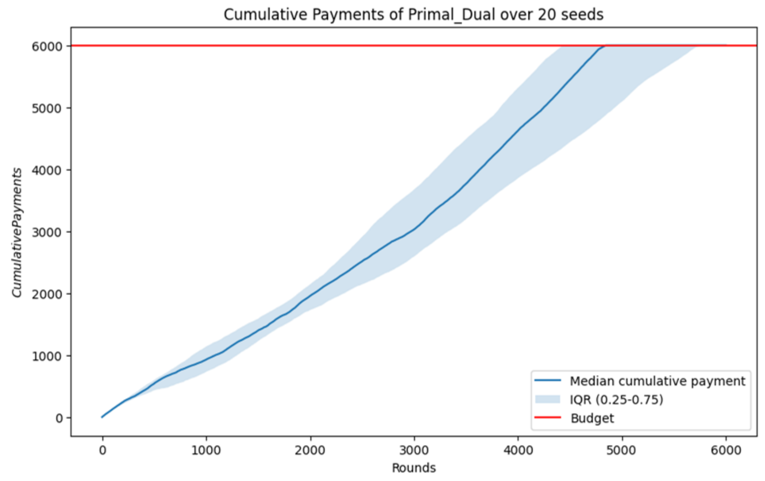
Outcomes:

- Almost same regret than fine grid
- Still shows sublinear growth across trials.

Features:

- **Coarse grid → faster computation and same regret.**



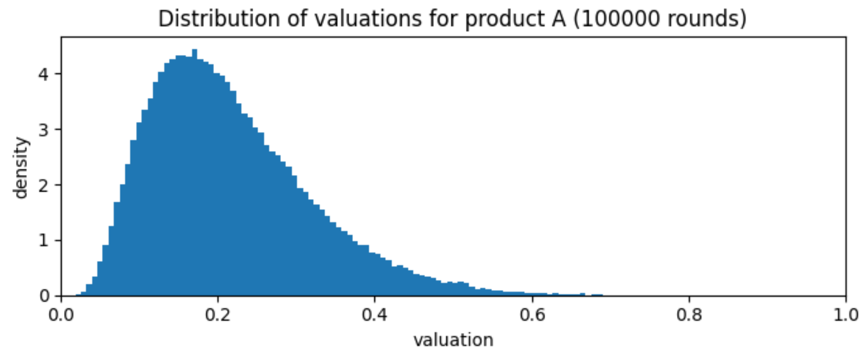


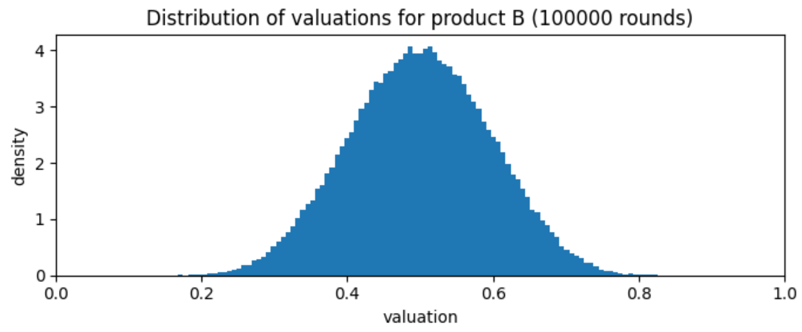
Key Takeaways

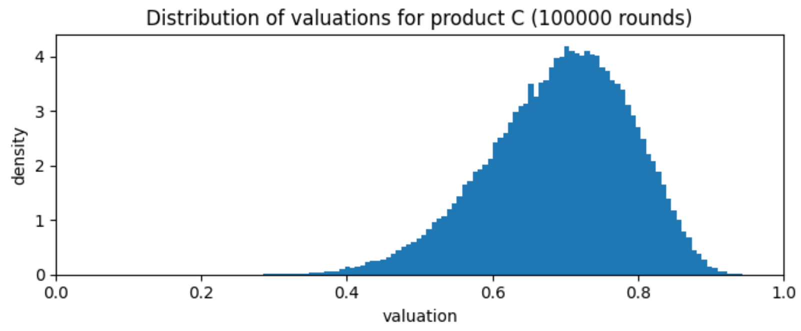
- When **budget = horizon**, exploration pressure is high:
  - Finer grids (17+ prices) allow **near-optimal** adaptation.
  - Coarse grids (5 prices) keeps same performances.
- **Sublinear regret** persists across all cases.
- Discounted EXP3.P + primal-dual remains **robust**:
  - Tracks valuations under nonstationarity.
  - Enforces budget constraints effectively.

Case: Budget > Horizon (B > T)

- **Setup**:
  - Budget = 10,000
  - Horizon = 6,000
  - Products = {A, B, C}
  - Price discretization: fine grid (~17 prices) or coarse grid (10 prices)
- **Key difference** from earlier cases:
  - Budget per-round $\rho > 1 \rightarrow$ more products sold per round more often.







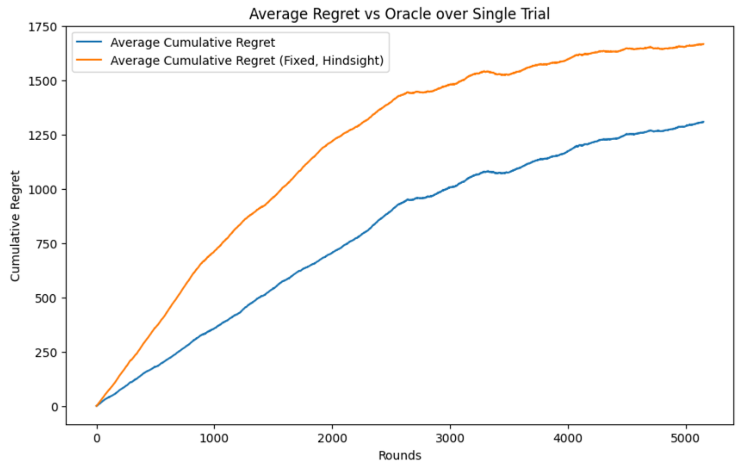
Single Trial (Fine Grid)

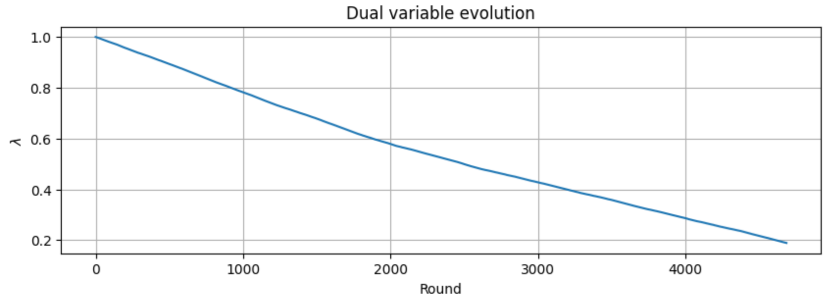
Agent explores broadly in early rounds.

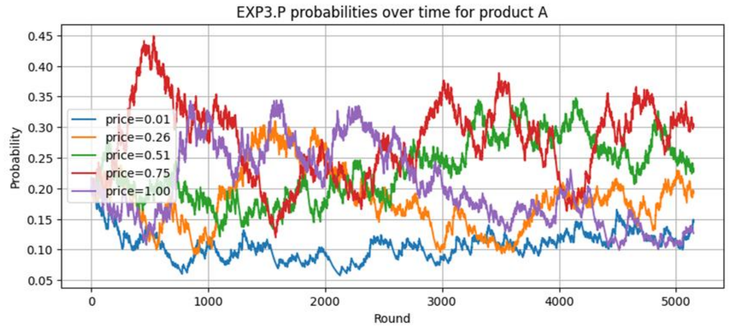
Cumulative regret decreases steadily:

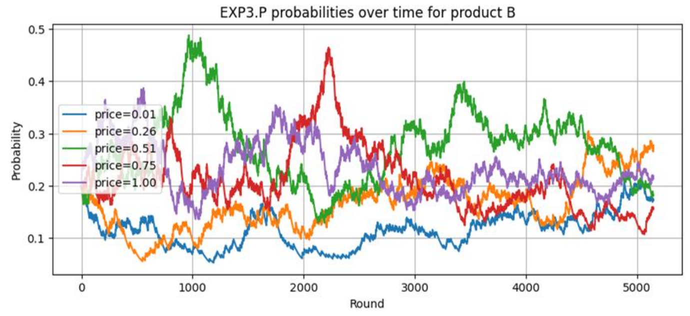
- Sublinear vs fixed hindsight oracle.
- Sublinear vs clairvoyant oracle.

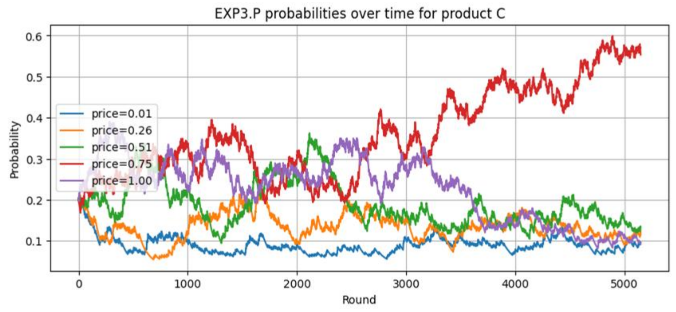
λ evolution: decreases up to 0.2 due to ρ > 1 .











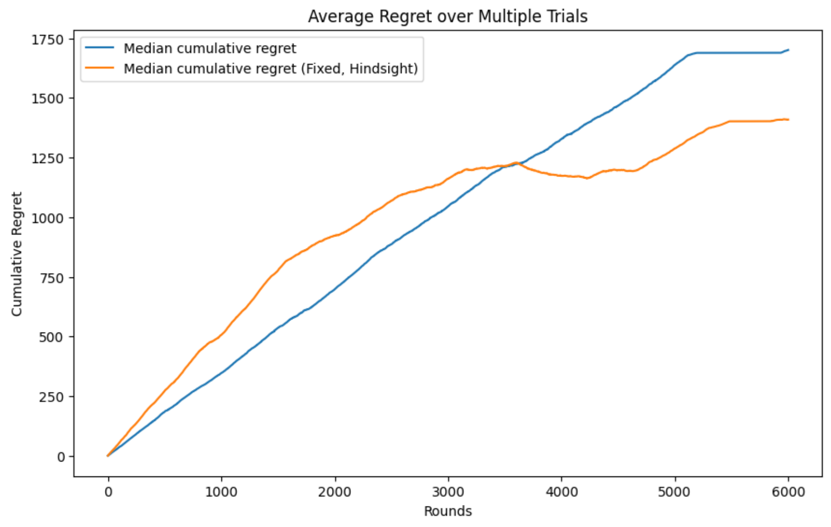
Multiple Trials (Fine Grid)

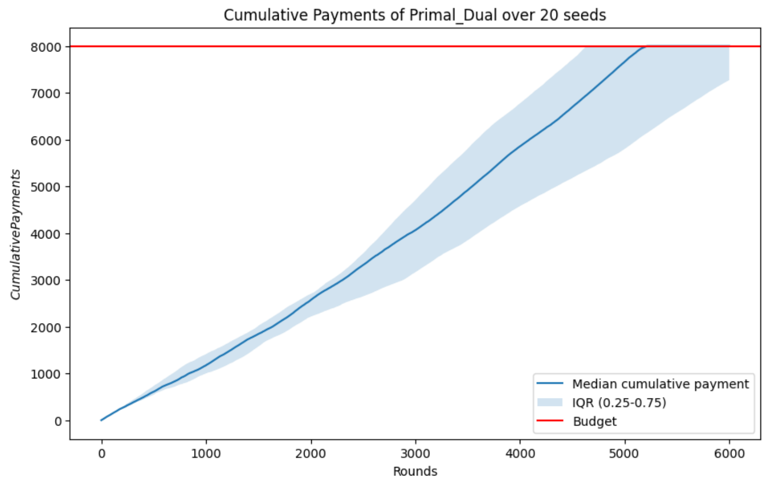
20 randomized seeds averaged.

Observations:

- Sublinear cumulative regret holds across trials.
- Average payments reaches budget.
- Revenue ranking remains C > B > A.

Exploration early, exploitation late → consistent adaptation.





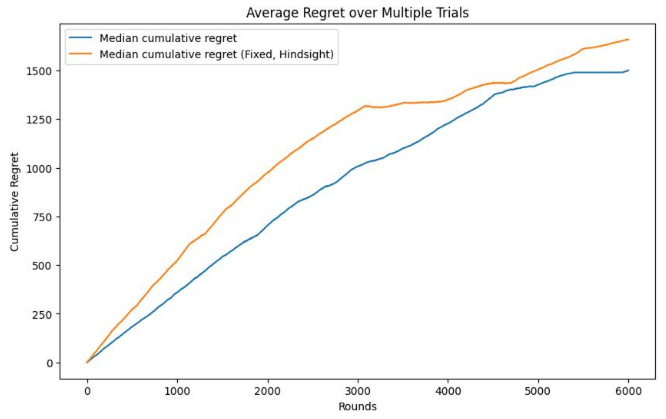
Multiple Trials (Coarse Grid: 5 Prices)

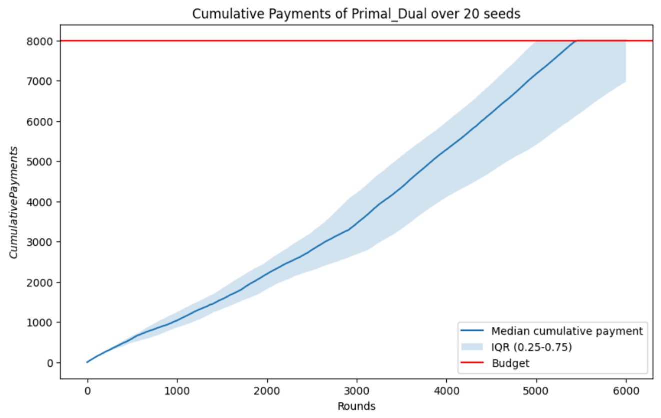
Restricting to 5 prices per product.

Features:

- Smoother regret for clairvoyant.
- Still maintains sublinear growth.

Practical insight: coarse grids may suffice when computation is limited.





Key Takeaways

- With budget > horizon, **sell multiple products more often at each round**.
- Discounted EXP3.P + primal-dual continues to:
  - Achieve sublinear regret.
  - Allocate prices toward profitable products.
  - Stay robust under both fine and coarse discretizations.
- Confirms flexibility of algorithm across different budget regimes.

<u>Requirement 5: Slightly non-stationary environments with multiple products</u>

Environment

- The base environment is the same used for requirement 2:
    - Uses a **multivariate logit-normal distribution** to model customer preferences.
    - Product valuations are obtained by sampling from the distribution and applying a **sigmoid** function to rescale the value in [0, 1] range
- To make it **slightly non-stationary** we define a specific number of intervals by considering the number of rounds each interval lasts and the distribution:
    - remain **fixed during each interval**
    - **changes** by randomly sampling new mean and standard deviation values **between each interval**.

Agent Case 1: UCB1 with GP and sliding window

- We used the same agent used for requirement 2, a UCB agent with GP to model the products demand curve
- To adapt it to the slightly non-stationary environment it was equipped with a **sliding window** mechanism:
  - During the GP update if the current sample exceeds the sliding window size the oldest sample is removed from the GP
- This mechanism allows the agent to “forget” old samples and adapt to the variability of the environment

Agent Case 2: Primal-Dual with modified EXP3.P

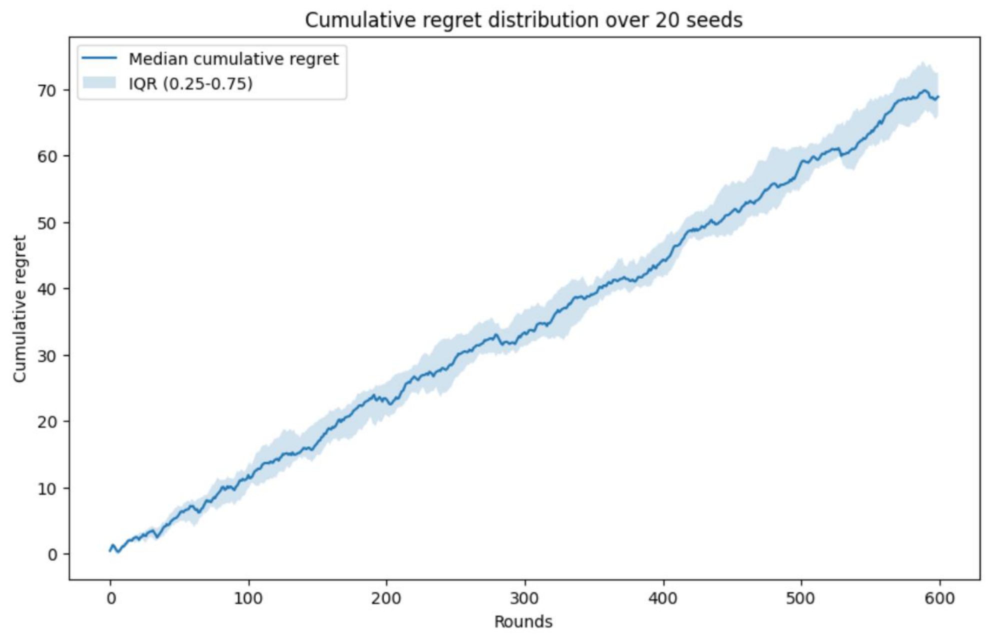
- This is the same primal dual agent used for requirement 4, which used an instance of EXP3.P for each product
- Given that the environment is **non-stationary** we modified the EXP3.P algorithm by introducing a **discount factor** to discount the weights over time to somehow “forget” the past
- For the Dual, we consider a smoothed update for each EXP3.P agent, each updated independently of the others

Baselines

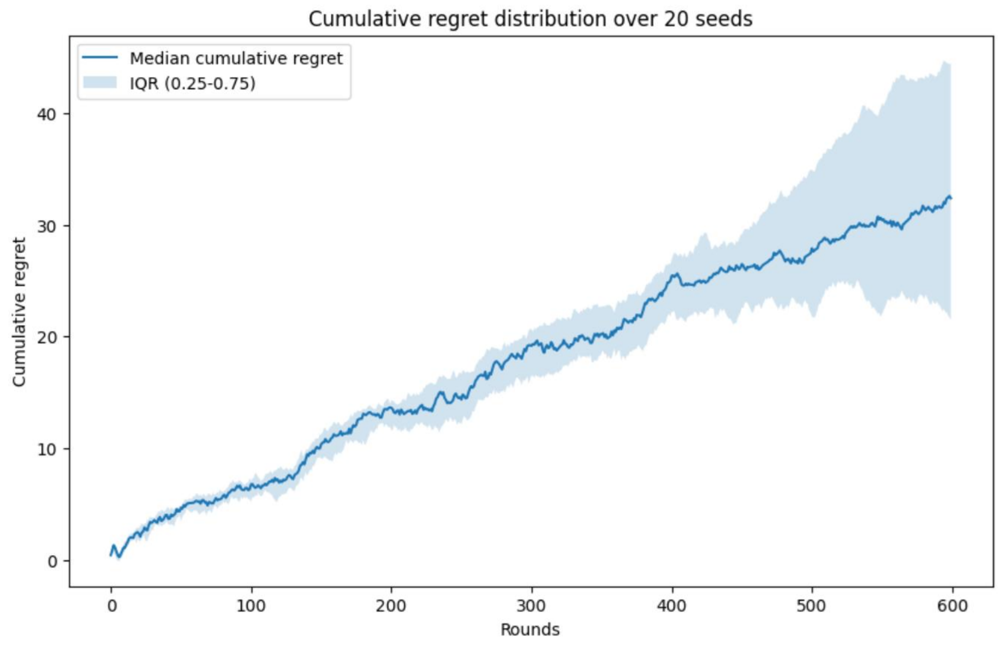
We used 2 different baselines for comparison:

- **Clairvoyant**:
  - Solves a linear program to compute the profit-maximizing price distribution for N products over K prices, subject to demand and unit constraints.
  - Returns the optimal probabilities, expected profit, and expected units used
- **Best fixed distribution hind slack**:
  - computes the hindsight-optimal fixed price distribution across all rounds by solving a linear program that maximizes average expected revenue under a global sales constraint
  - This is used only with the Primal-Dual.

Trials with CombUCB1 agent

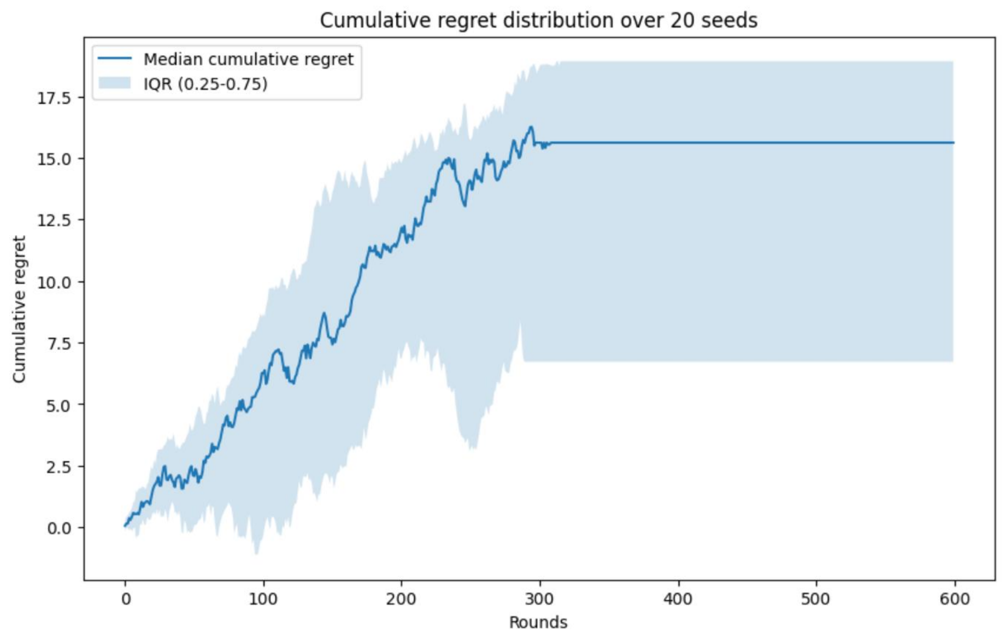


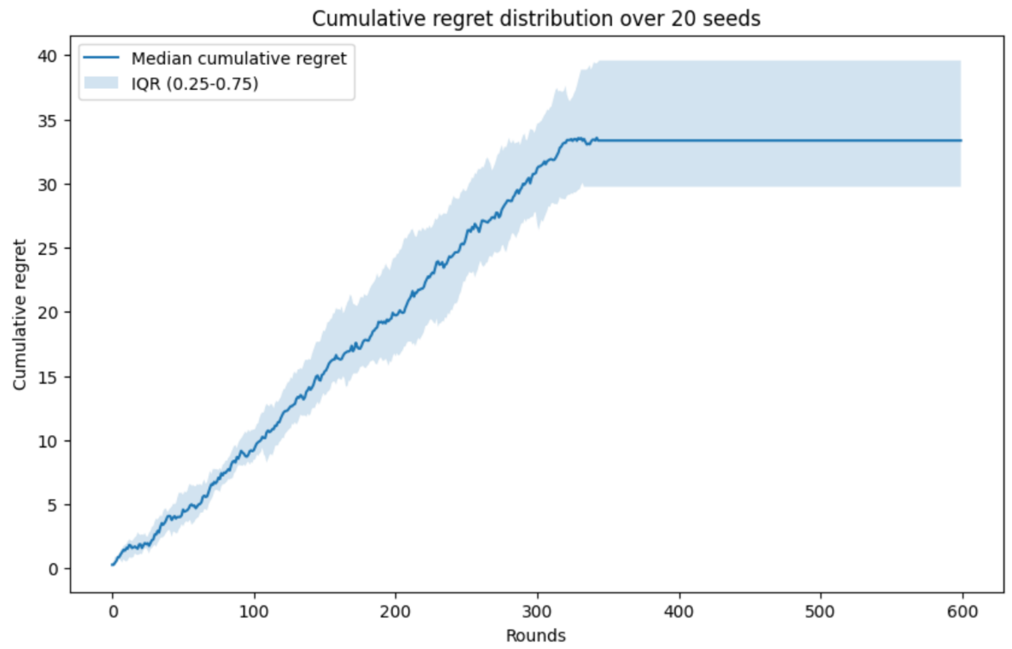
When the environment changes frequently the agent is struggling to achieve a sublinear regret. This is because the environment is almost non-stationary and the CombUCB1 agent is not suited for non-stationary environments.



Increasing the number of rounds after which the environment distribution changes the agent is able to achieve **sublinear regret.**

Trials with Primal-Dual agent

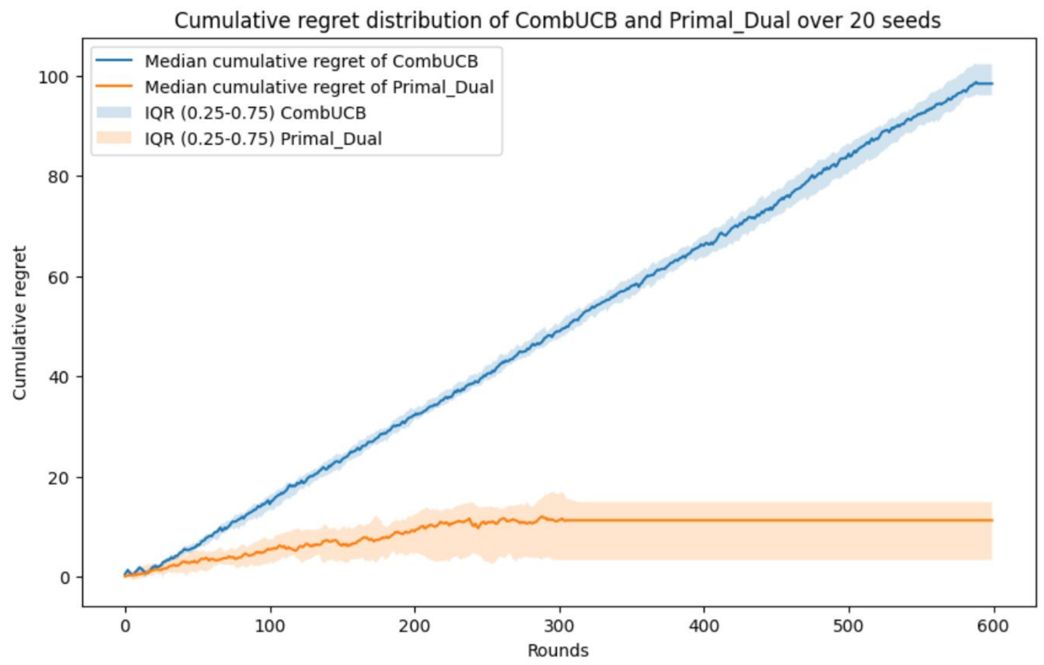


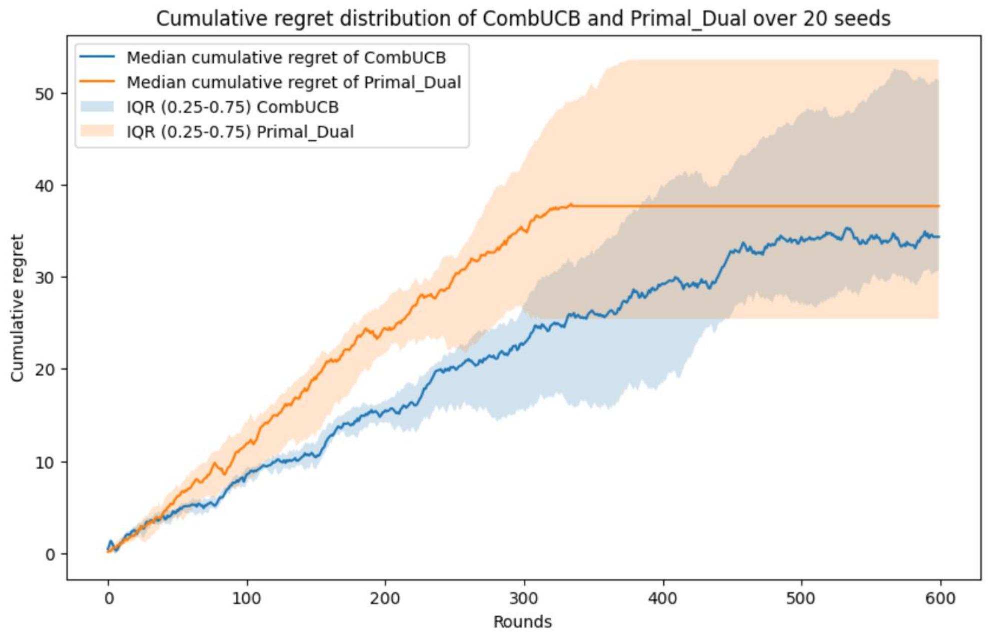


When the environment changes frequently the primal-dual agent show best performances since it's specialized for non-stationary environments

When the environment becomes almost stationary the regret is still sublinear but the agent performances are worse.

Agents comparison B < T

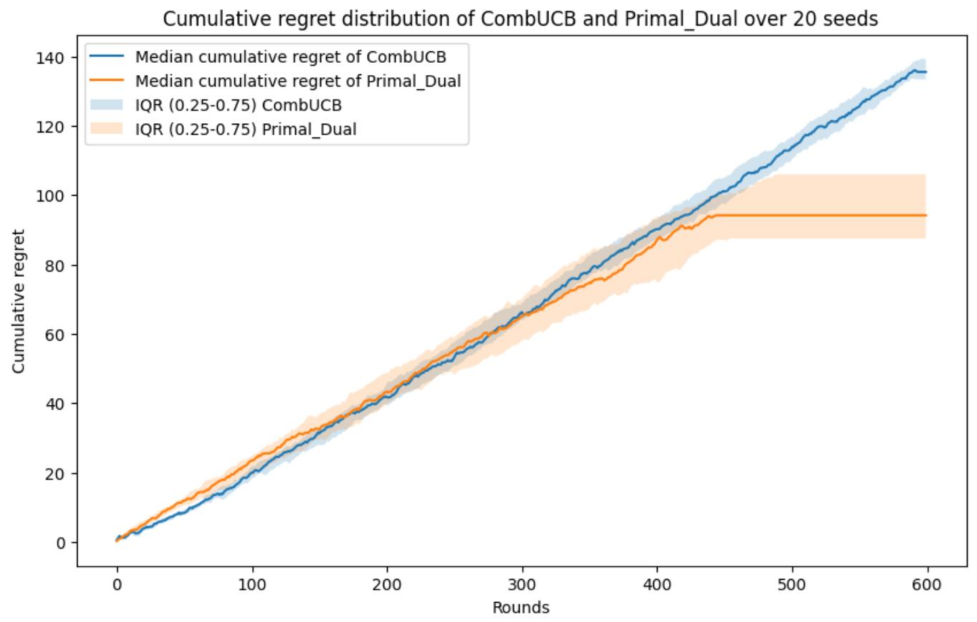


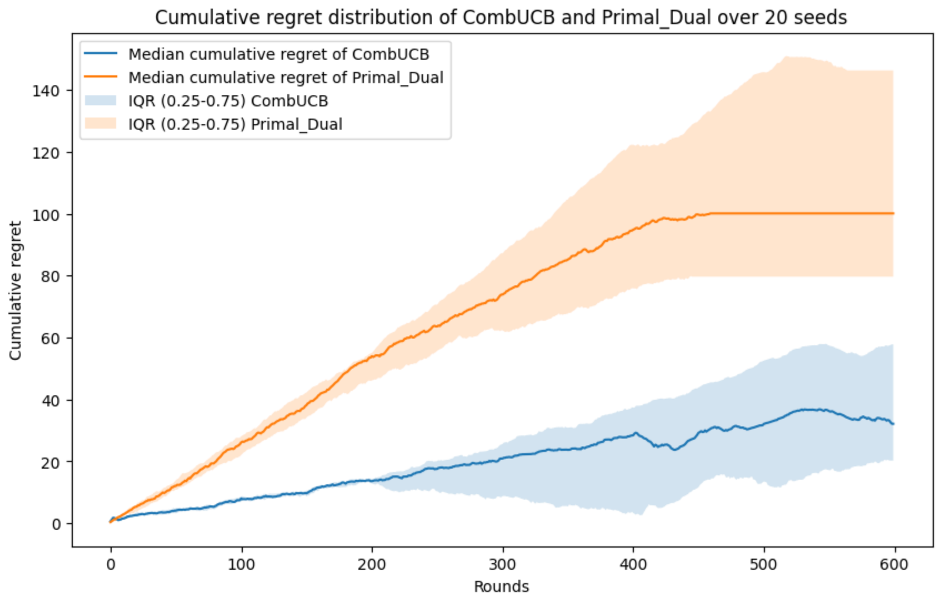


With small change intervals the primal dual agent clearly outperform the CombUCB1 which is not effective in highly non-stationary environments.

While with an almost stationary environment the CombUCB1 agent is superior.

Agents comparison B = T

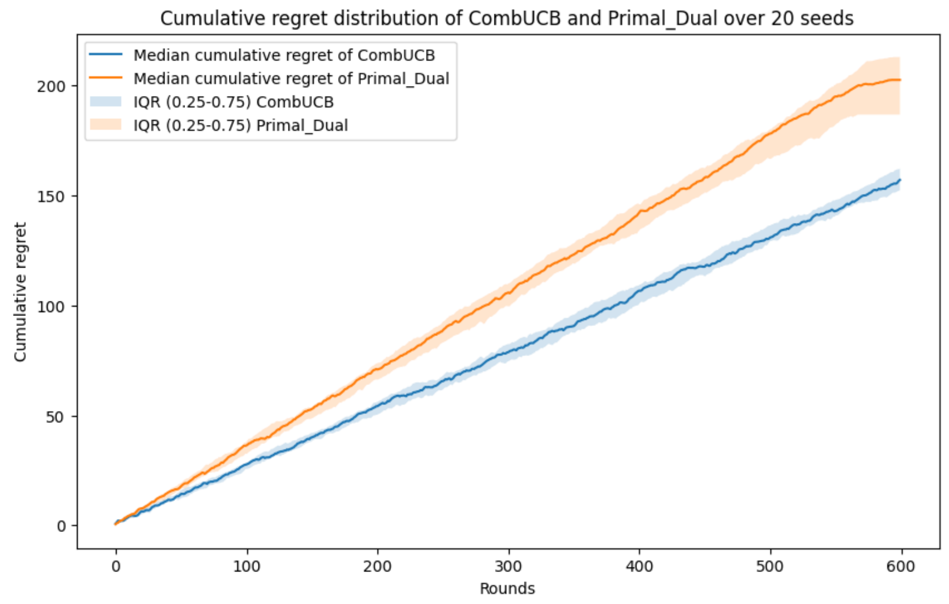


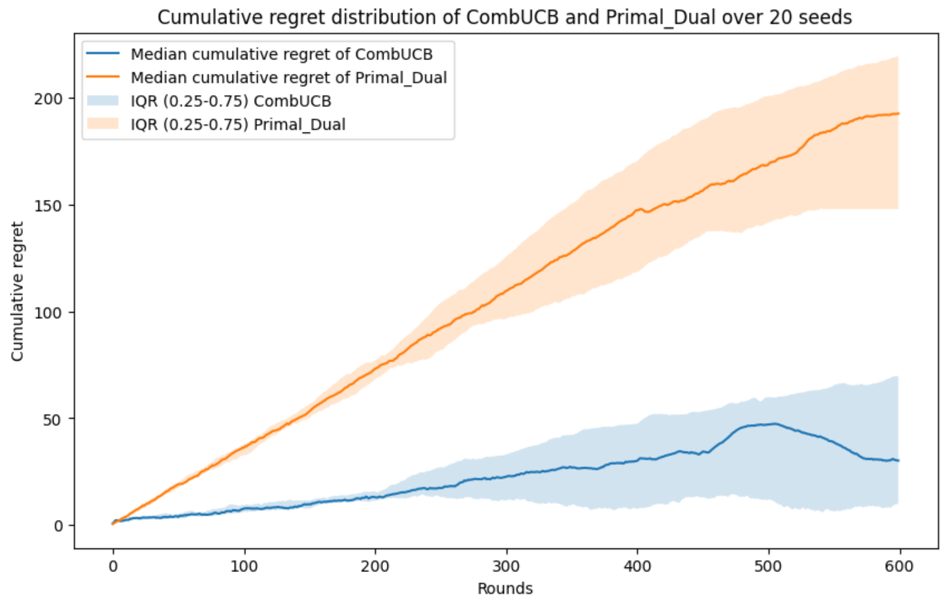


As expected with an environment which is almost completely non-stationary the CombUCB1 agent struggle to achieve sublinear regret.

Decreasing the number of intervals, the primal dual agent show worse performance as already seen before, while the almost stationary environment let the CombUCB1 agent outperform the primal dual one.

Agents comparison B > T





Increasing budget make the primal dual agent being outperformed by the CombUCB1 even in almost non-stationary environment.

As expected the CombUCB1 agent thrive with almost stationary environment also when B > T.

Conclusions

Requirement 1:

- UCB1 no budget constraint –> sublinear regret
- UCB1 with budget constraint –> sublinear regret

Requirement 2:

- Combinatorial UCB (with GP) –> sublinear regret

Requirement 3:

- Primal-Dual (EXP3.P) single product –> sublinear regret

Requirement 4:

- Primal-Dual (EXP3.P) multiple products –> sublinear regret

Requirement 5:

- Combinatorial UCB (with GP and SW) –> sublinear regret
- Comparison between Combinatorial UCB (with GP and SW) and Primal-Dual –> coherent results

Further improvements:

- Create new environments
- Test each algorithm under different conditions (limit cases, new environments, different parameters,…)
- Further optimize each algorithms by hyperparameters tuning

Useful Links

Github link to the Repository of the Project:

- https://github.com/Digioref/OLA-Pricing.git

Paper of EXP3.P:

- https://cesa-bianchi.di.unimi.it/Pubblicazioni/J18.pdf