Online Learning Applications

Project by:
Jan Bartolini
Lorenzo Biraghi
Francesco Inzerillo
Pratik Rameshwar Potdukhe
Matteo Zanetti

Project Overview

Goal:
Design online learning algorithms for dynamic pricing of multiple products under production constraints

Key Components:

- Stochastic and non-stationary environments
- Budget/inventory constraints
- Multi-armed bandit algorithms
- Combinatorial optimization

Business Context:
Company dynamically prices products with limited production capacity

Problem Setting

Parameters:

- T: Number of rounds (time horizon)
- N: Number of product types
- P: Set of possible prices (discrete)
- B: Production capacity (budget constraint)

Buyer Behavior:

- Has valuation v_i for each product type
- Buys all products priced below their valuations

Interaction per Round:

1. Company sets prices for each product type
2. Buyer arrives with product valuations
3. Buyer purchases products priced below their valuations

R1.1: Single Product - Stochastic Environment

Environment: Beta Distribution

- Stochastic valuations: $v_t \sim \text{Beta}(a, b)$
- Single product type
- Fixed distribution over time

Baseline:
Reward of a Clairvoyant agent i.e. an agent aware of the underlying distribution

Goal:
Achieve Sublinear Pseudo-Regret

Algorithm UCB1:

Input: set of arms A, number of rounds T.

For each round t=1,...,T:

- For each arm $a \in A$:
  - Estimate mean reward:

$$\mu_t(a) = \frac{1}{N_{t-1}(a)} \sum_{\tau=1}^{t-1} r_\tau(a) I\left[a_\tau = a\right]$$

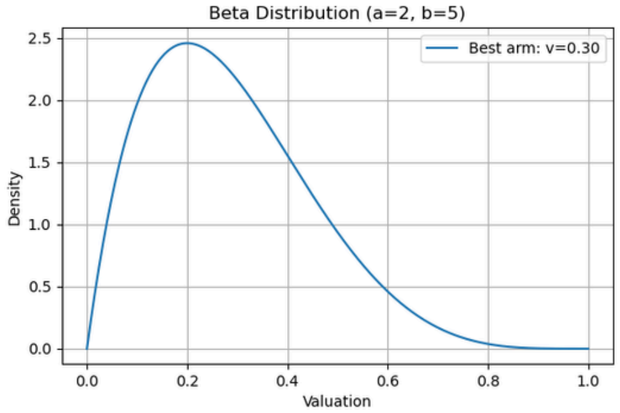
  - Compute confidence bound:

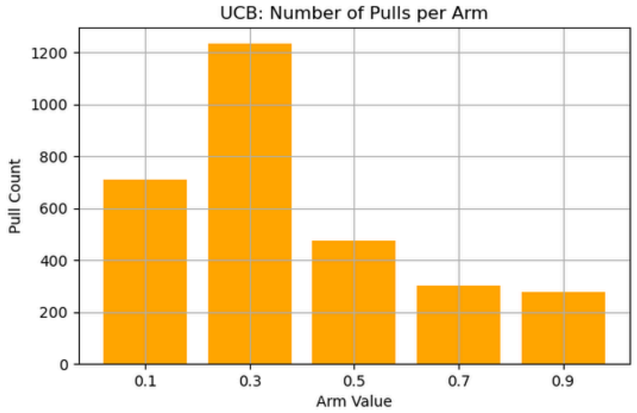
$$UCB_t(a) = \mu_t(a) + \sqrt{\frac{2logT}{N_{t-1}(a)}}$$

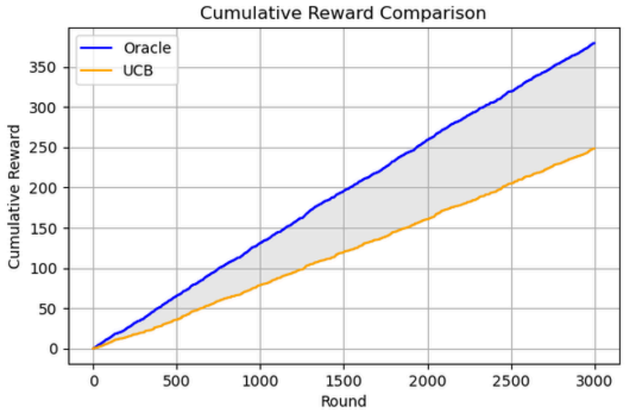
- Play arm

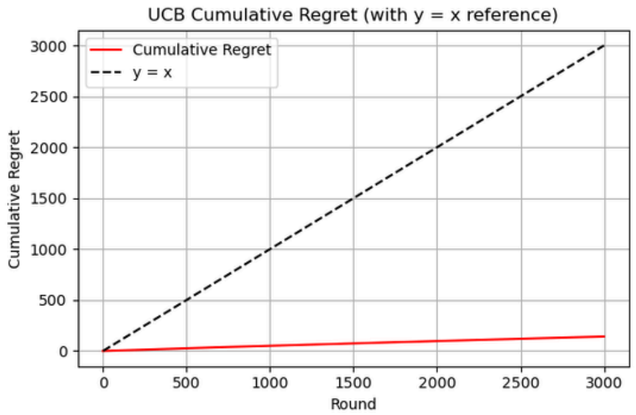
$$a_t = \arg\max_a UCB_t(a)$$

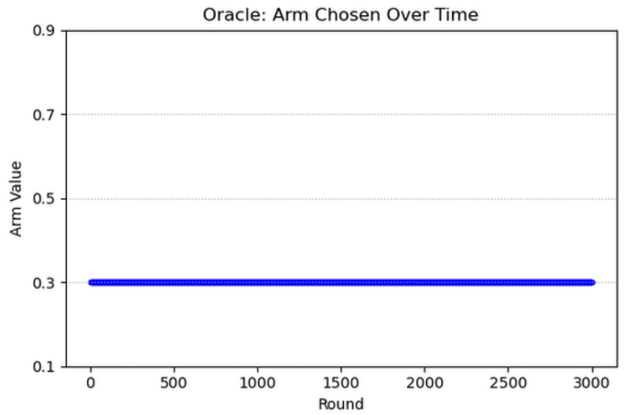
Performance Comparison - No Budget

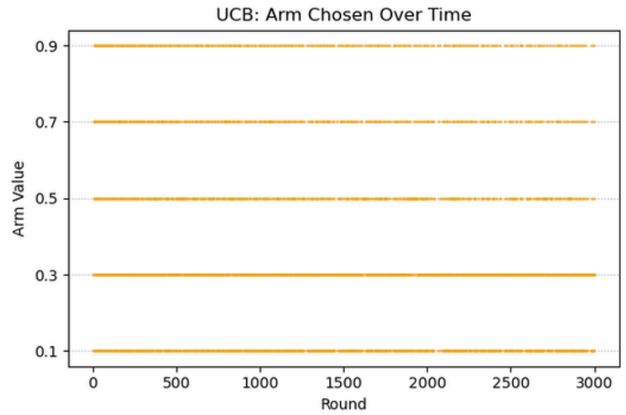










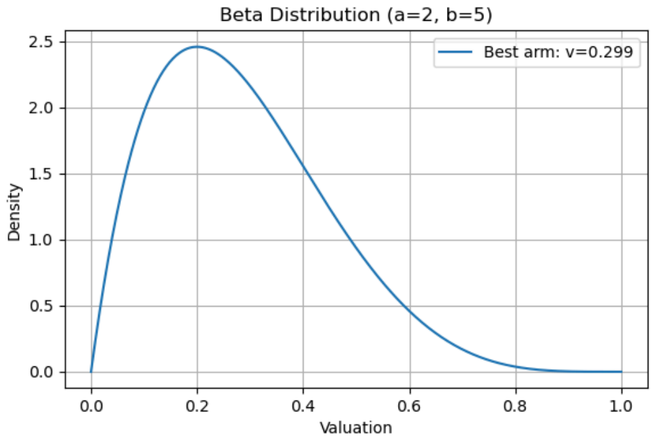


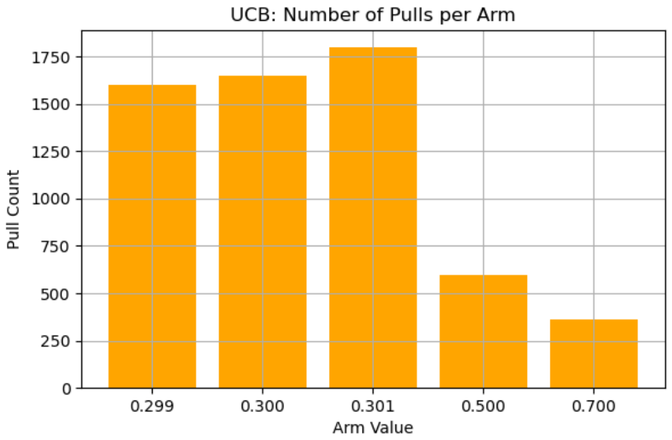
After an initial exploration the UCB agents manages to identify the optimal arm.

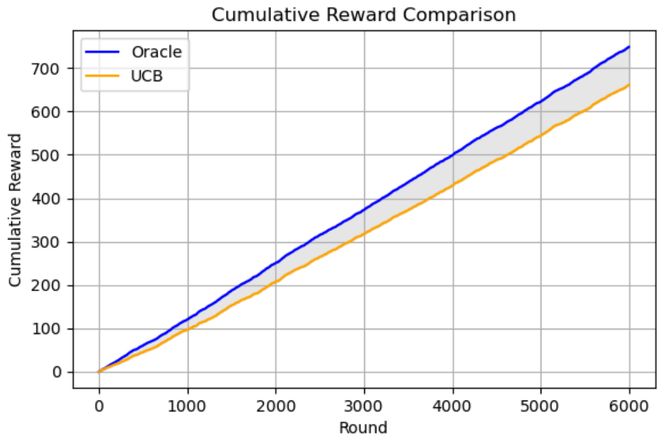
Sublinear regret is easily achieved

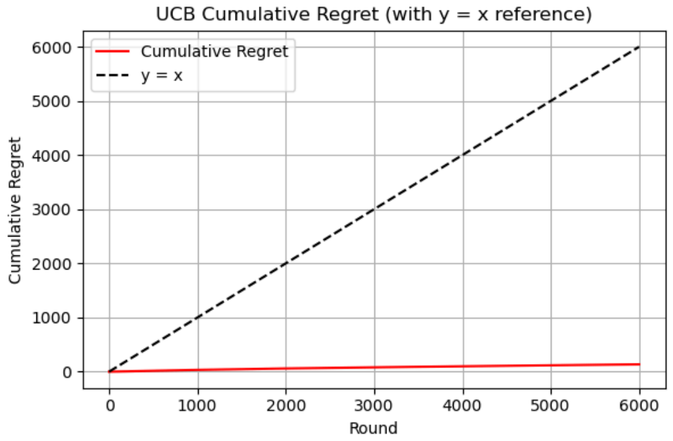
Empirical results are coherent with theoretical ones

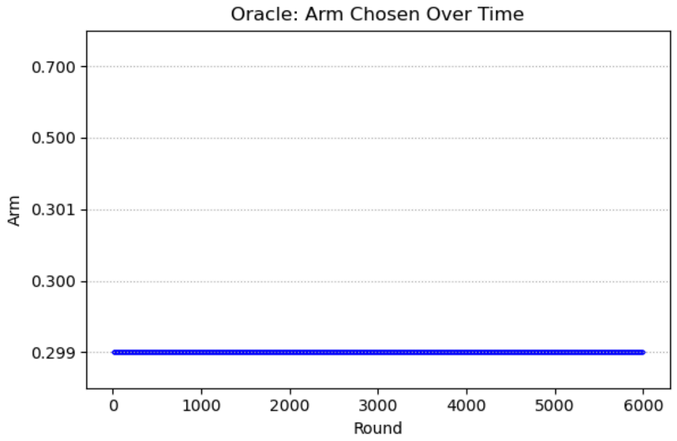
R1.1: Notable case

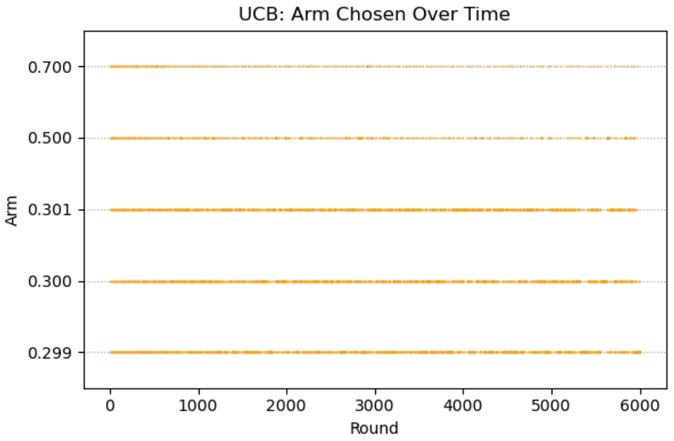












With arms really close in values, the algorithm still achieves sublinear regret.

R1.2: Single Product - Stochastic Environment

Environment: Beta Distribution

- Stochastic valuations: $v_t \sim \text{Beta}(a, b)$

LP Formulation:

$$\begin{aligned} \max \quad & \sum_i r_i \times x_i \\ \text{s.t.} \quad & \sum_i p_i \times x_i \leq \frac{b}{T-t} \\ & \sum_i x_i = 1 \\ & 0 \leqslant x_i \leqslant 1 \end{aligned}$$

Baseline:

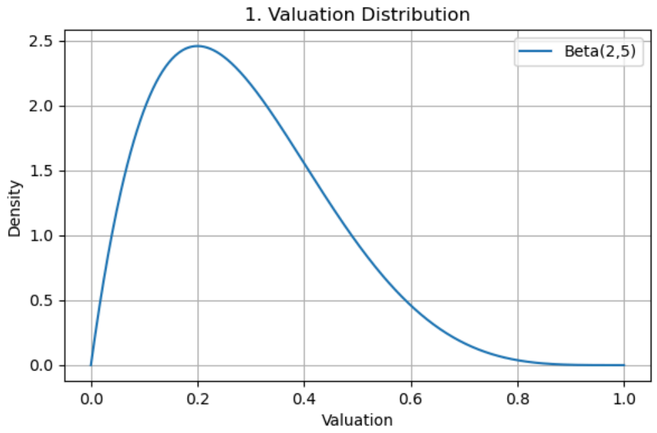
The reward of the best dynamic policy when the decision maker is aware of the underlying distribution

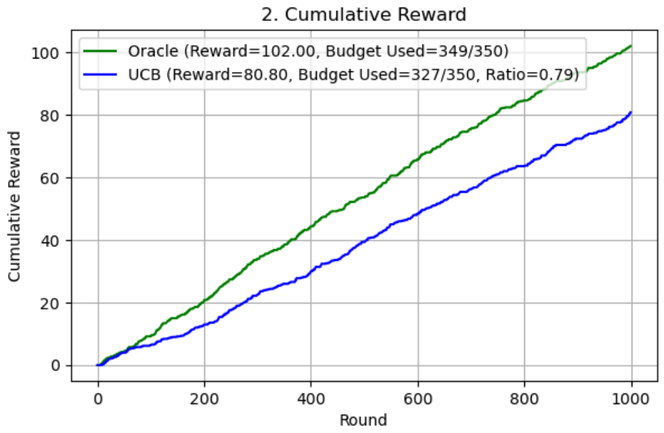
Algorithm UCB1 + Budget:

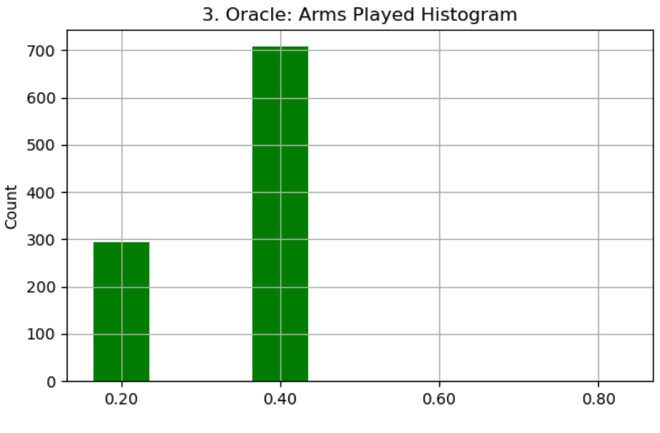
for t=1...T:

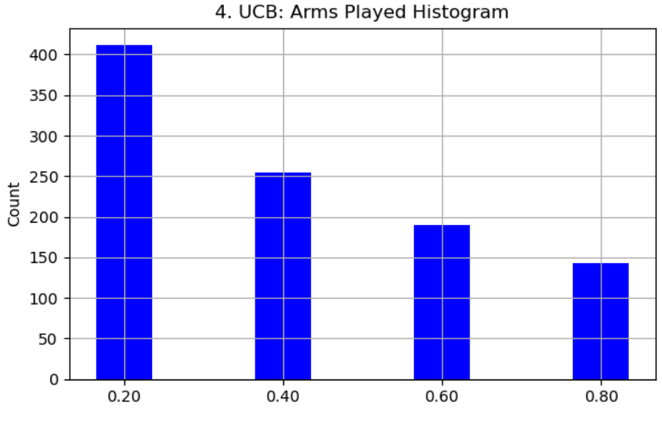
1. Initialization phase:
   - Play each arm once
2. Compute estimates and confidence. For each arm b:
   - $f_t(b) \leftarrow R_b/N_b$ (mean reward)
   - $c_t(b) \leftarrow S_b/N_b$ (mean cost, i.e. success probability)
   - $conf_t(b) \leftarrow \sqrt{\frac{2logT}{N_b}}$
   - $f^{UCB}{}_t(b) \leftarrow f_t(b) + conf_t(b)$
   - $c^{LCB}{}_t(b) \leftarrow max\{c_t(b) - conf_t(b), 10^{-6}\}$
3. Compute per-round budget: $\rho_t \leftarrow \frac{remainingbudget}{T-t+1}$
4. Solve Linear Program and obtain distribution γ_t over arms
5. Sample and play an arm from γ_t
6. Update statistics:
   - $N_b \leftarrow N_b + 1$
   - $R_b \leftarrow R_b + r_t$
   - If T+ > 0:
     - $S_b \leftarrow S_b + 1$
     - Remaining budget ← Remaining budget - 1

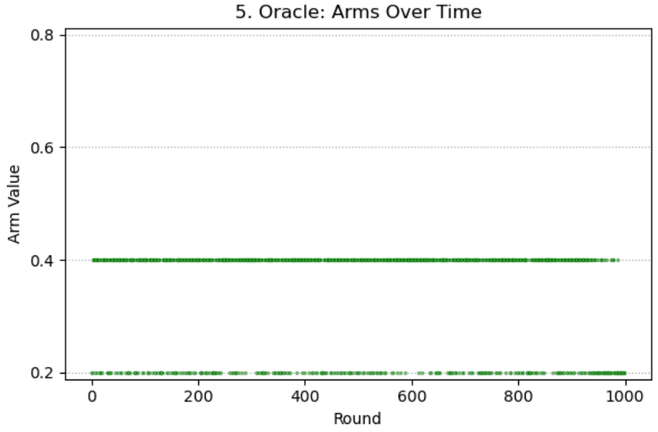
Performance Comparison - With Constraint

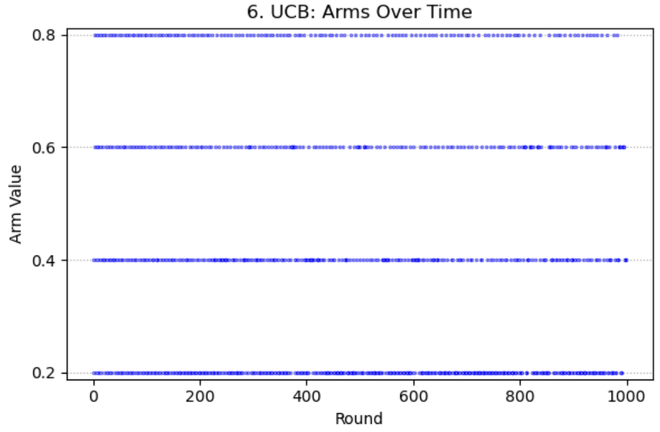






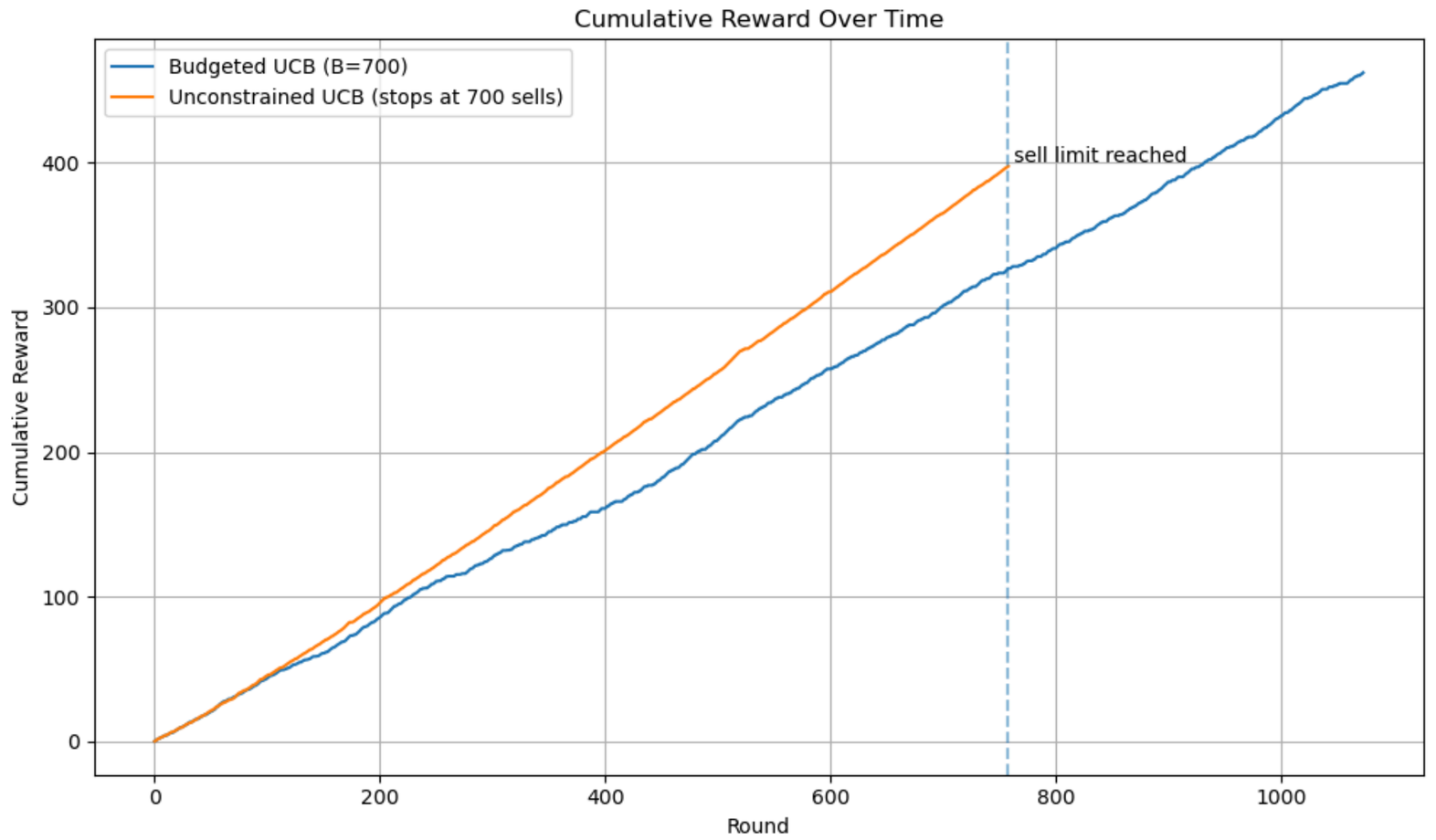






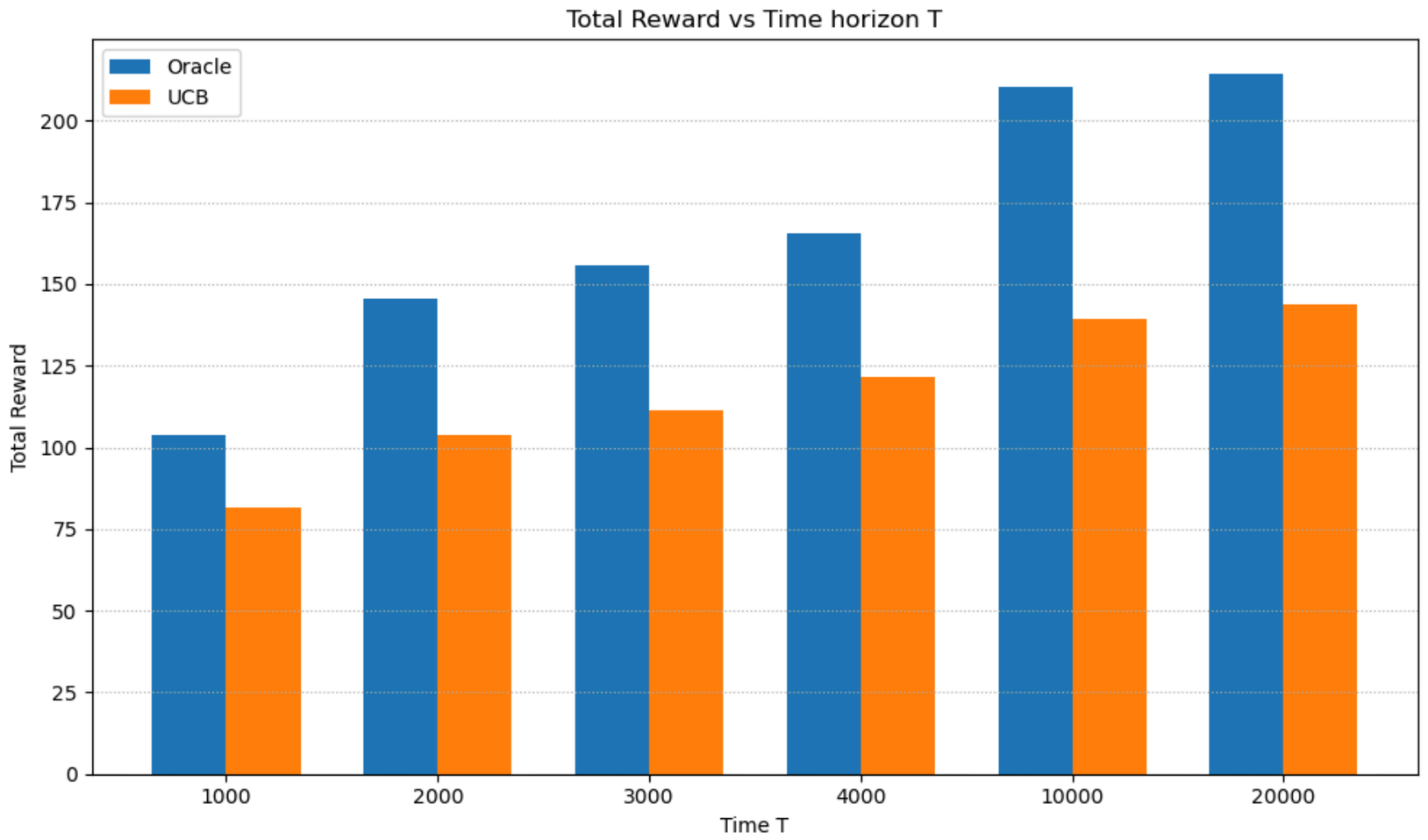
The Budget constraint is effective for the UCB agent to avoid overspending, without preventing it from converging to the optimal plan.

Performance Comparison - With Constraint



The Budgeted agent outperforms the Classic one leveraging hte greater amount of round forcing more expensive arms

Performance Comparison - With Constraint



If we keep the budget B constant and incrase t we see the performances of both agents improve, although exists an upperbound for the total utility the Clairvoyant can achieve which is:

$$B \times k^{opt}$$

$$k^{opt} = max_{k \in K}\{k\}$$

And in the same way the UCB agent cannot achieve more than a fraction of the total utility of the Clairvoyant because it has to expend some rounds exploring obtaining sub-optimal rewards

R2: Multi Product - Stochastic Environment

Environment: Beta Distribution

- **Stochastic valuations:** $v_{t,i} \sim \text{Beta}(a_i, b_i)$
- **Multiple product types**

LP Formulation:

$$\max \sum_s R_s \times x_s$$

$$\text{s.t.} \quad \sum_s C_s(t) \times x_s \leq \frac{b}{T-t}$$

$$\sum_s x_s = 1 \qquad 0 \leqslant x_s \leqslant 1$$

$$S = \{S \subseteq I : |S \cap I_p| \leqslant 1, p \in [1, k]\}$$

$$R_s = \sum_{i \in s} r_i \qquad C_s = \sum_{i \in s} p_i$$

$$r_i = \hat{\mu}_{i,t} + \sqrt{\frac{2 \log T}{N_{i,t}}} \qquad p_i = 1 - F_{beta(a,b)}(v_i)$$

Budgeted bandit strategy

For each round $t = 1, \ldots, T$**:**

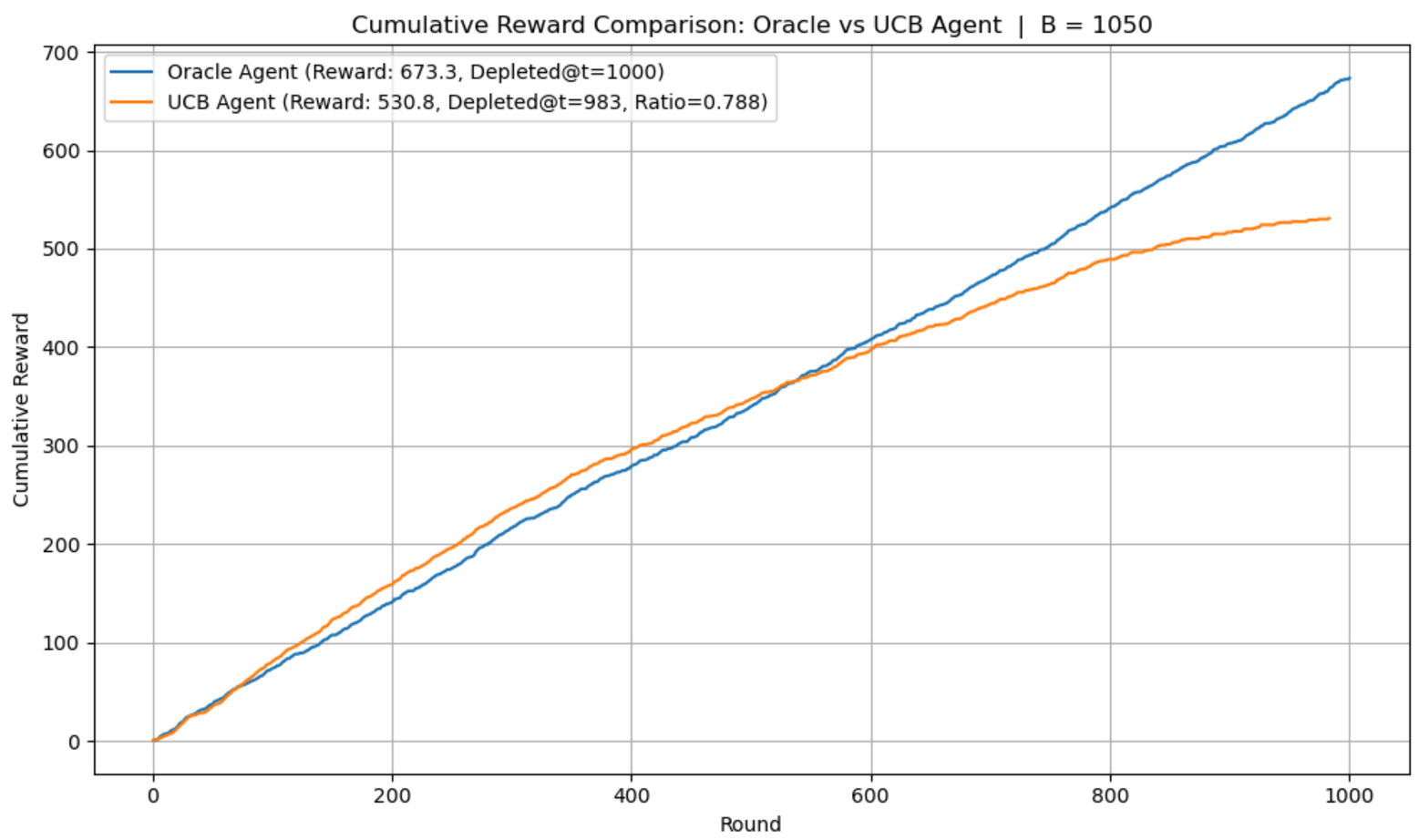
- **Termination: stop if budget ≤ 0.**
- **Initialization: if** $t \leq n$**, play each product's t-th arm.**
- **Estimate reward/cost for each arm a:**
  - **If** $N[p,a] = 0$**: set upper/lower bounds to** $+\infty/0$**.**
  - **Else:**

$$avg.\,reward[p,a] = R[p,a]/N[p,a] \qquad bonus = \sqrt{\frac{2log(1/\delta)}{N[p,a]}}$$

$$ucb\_reward[p,a] = \min(1, avg.\,reward + bonus),\ lcb\_cost[p,a] = \max(0, prob.\,success - bonus)$$

- **Build superarms: combine arms and compute total reward/cost estimates.**
- **Solve LP: choose superarm maximizing total reward, subject to budget constraint.**
- **Play chosen arms, update counters, rewards, costs.**
- **Update budget:** $B \leftarrow B - \text{sales cost}$**, increment round.**

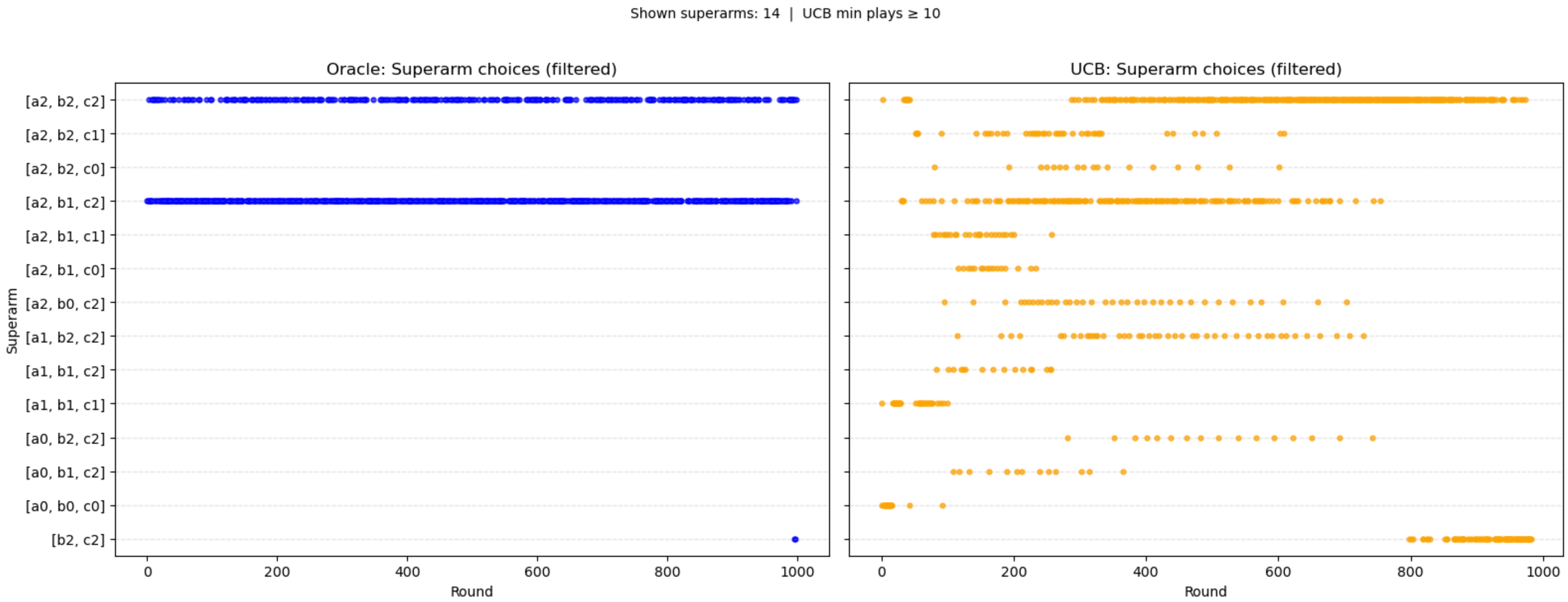
Performance: Rewards



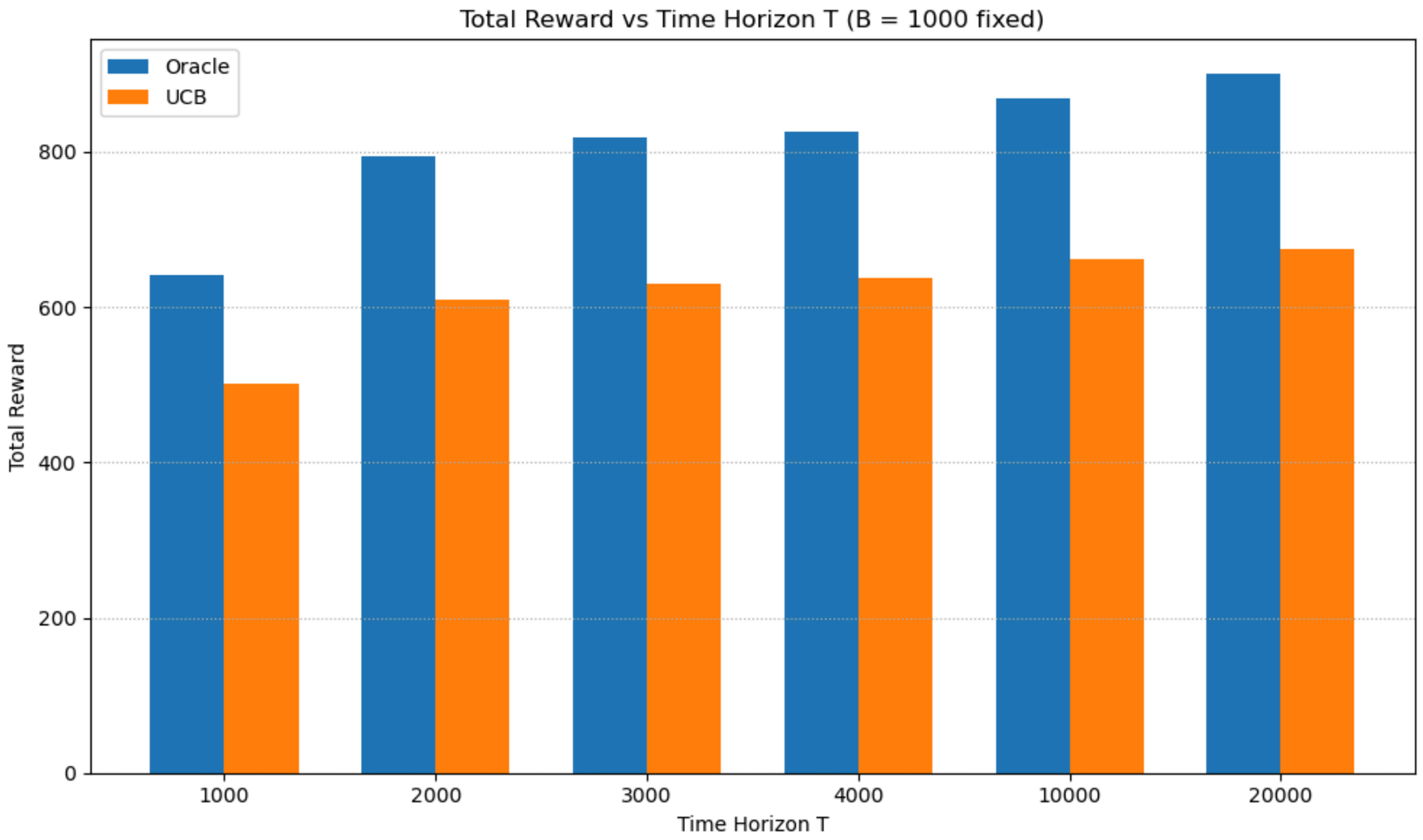
The UCB agent achieves a good fraction of the oracle utility and it uses most of the rounds at his disposal to maximize the global reward

The shape of the graph is also one we encounterd many times:
The UCB agent has higher rewards in the first part of the simulation which decay significantly in the second part, resulting in a concave function.
The Clairvoyant, being aware in advance of the distribution doesn't waste plays on sub-optimal arms leveraging the amount of rounds at its disposal to exploit the more profitable arms

Performance: Arms Selection



Performance Comparison - With Constraint

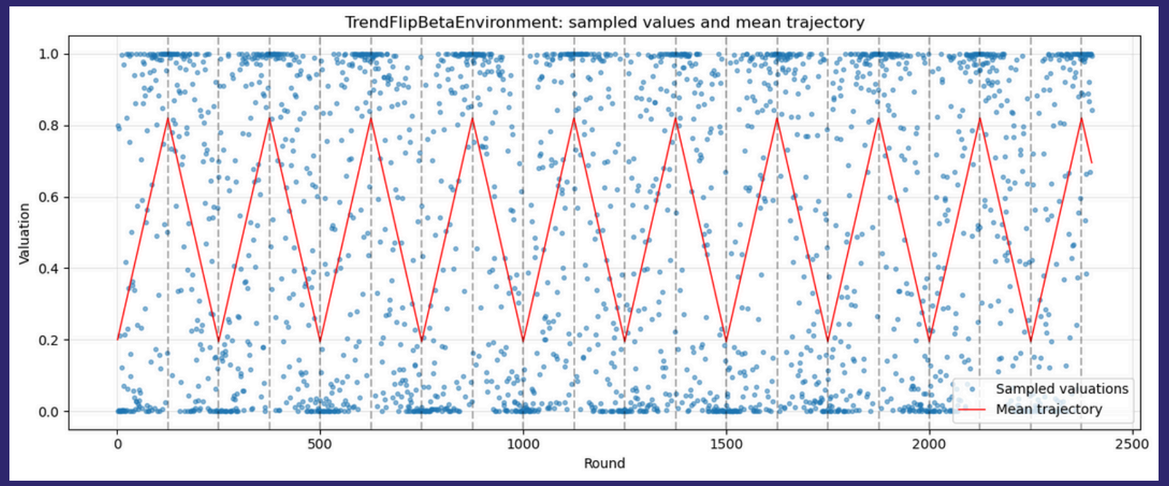


Exactly as we saw in the univariate case as T approaches infinity the Clairvoyant's reward and the UCB's reward approach their respective upperbound.

3: Single-Product Best-of-both-worlds

Environment 1: Beta Distribution

- Stochastic valuations: $v_t \sim \mathrm{Beta}(a, b)$
- Single product type
- Fixed distribution over time



Environment 2: Highly Non-Stationary

Parameters:

$v_t \sim \mathrm{Beta}(a, b)$

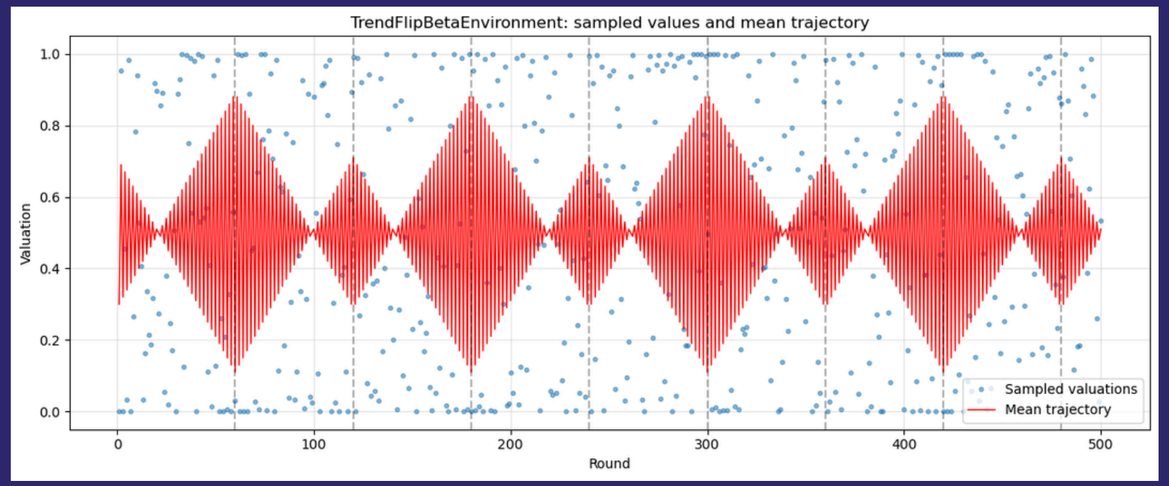
T: number of rounds

u0: starting mean

v: variance

u: drift of the mean

k: number of rounds after flipping the trend



3: Single-Product Best-of-both-worlds

Algorithm: Primal-Dual Method with EXP3.P (Pacing Strategy)

Initialization: $\rho \leftarrow \frac{B}{T}, \lambda_0 \leftarrow 0;$

For t = 1,2,...T DO

- **Primal decision: obtain γ_t via regret minimizer EXP3.P**
- **Choose price $p_t \sim \gamma_t$;**
- **Observe $f_t(b_t), c_t(b_t)$**
- **Dual Variable Update: $\lambda_{t+1} = \Pi_{[0,1/\rho]}(\lambda_{t-1} - \eta(\rho - c_t(b_t)))$**
- **Update budget**
- **If $B < 1$: TERMINATE**

Baseline:

The reward of the best dynamic policy when the decision maker is aware of the underlying distribution

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

$$p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K}.$$

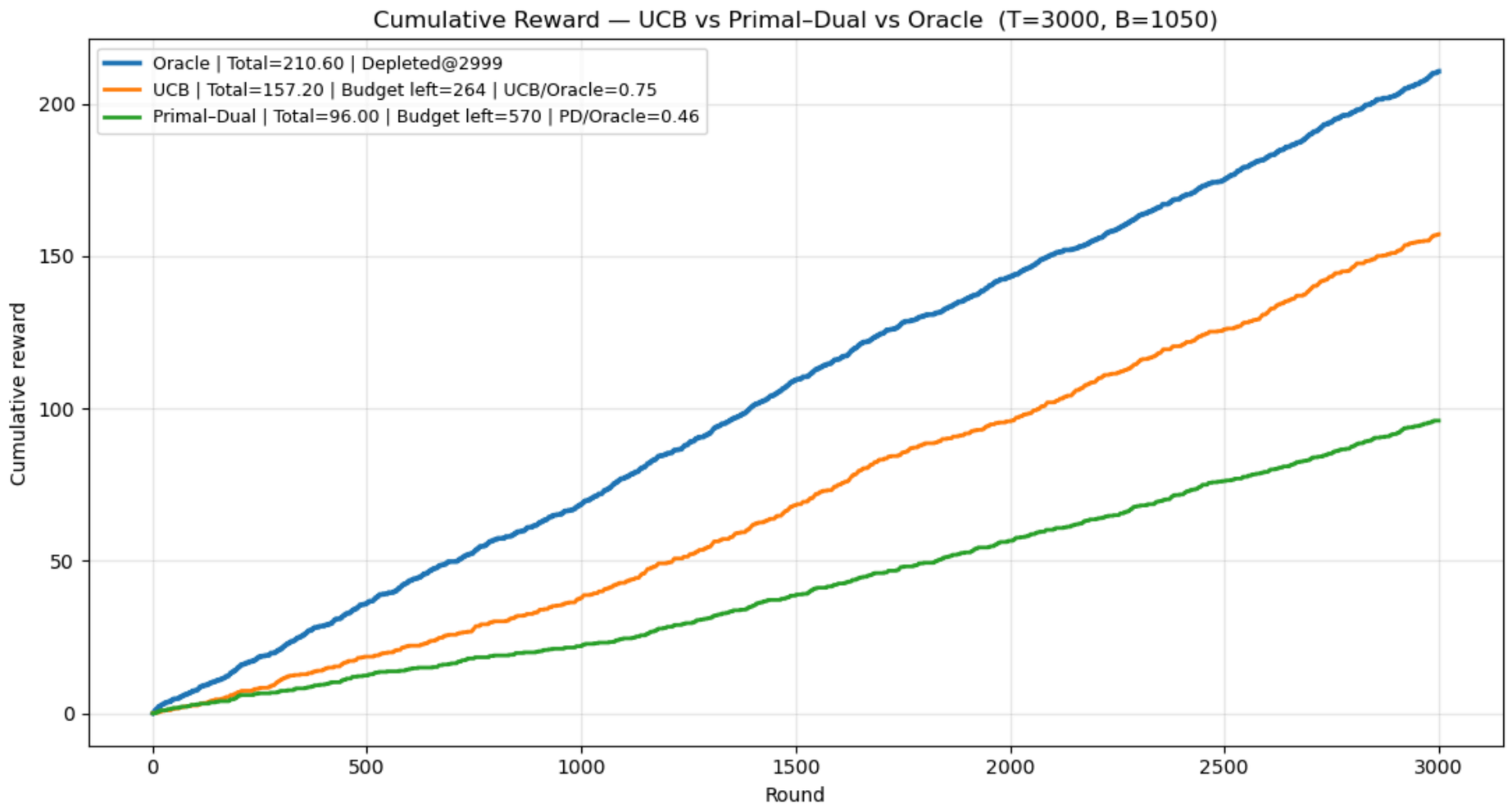
2. Choose i_t randomly according to the distribution $p_1(t), \ldots, p_K(t)$.
3. Receive reward $x_{i_t}(t) \in [0, 1]$.
4. For $j = 1, \ldots, K$ set

$$\hat{x}_j(t) = \begin{cases} x_j(t)/p_j(t) & \text{if } j = i_t, \\ 0 & \text{otherwise,} \end{cases}$$

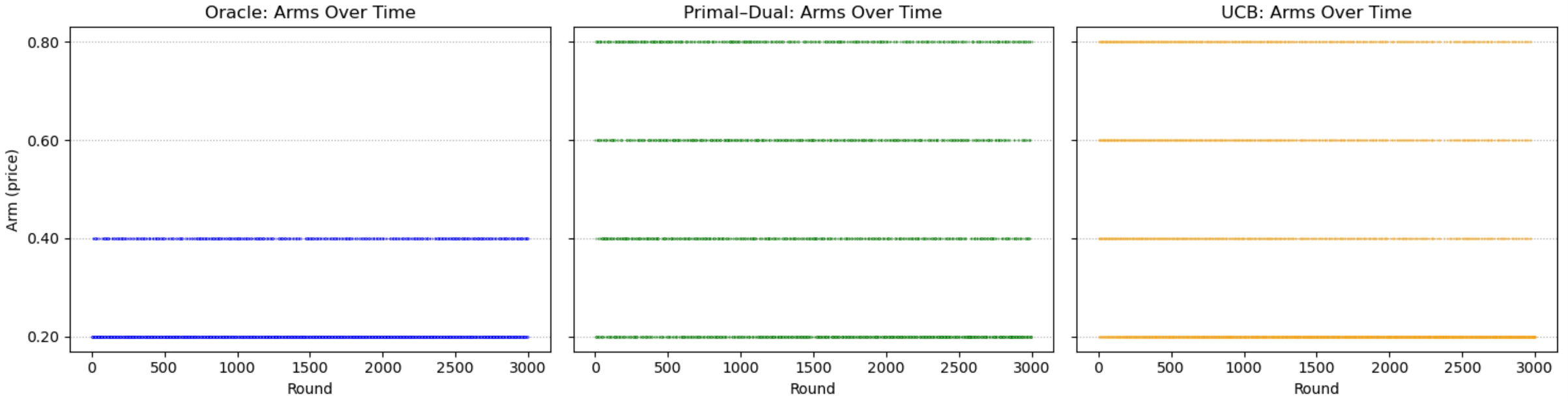
$$w_j(t+1) = w_j(t)\exp\left(\frac{\gamma}{3K}\left(\hat{x}_j(t) + \frac{\alpha}{p_j(t)\sqrt{KT}}\right)\right).$$

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. SIAM journal on computing, 32(1):48–77, 2002.

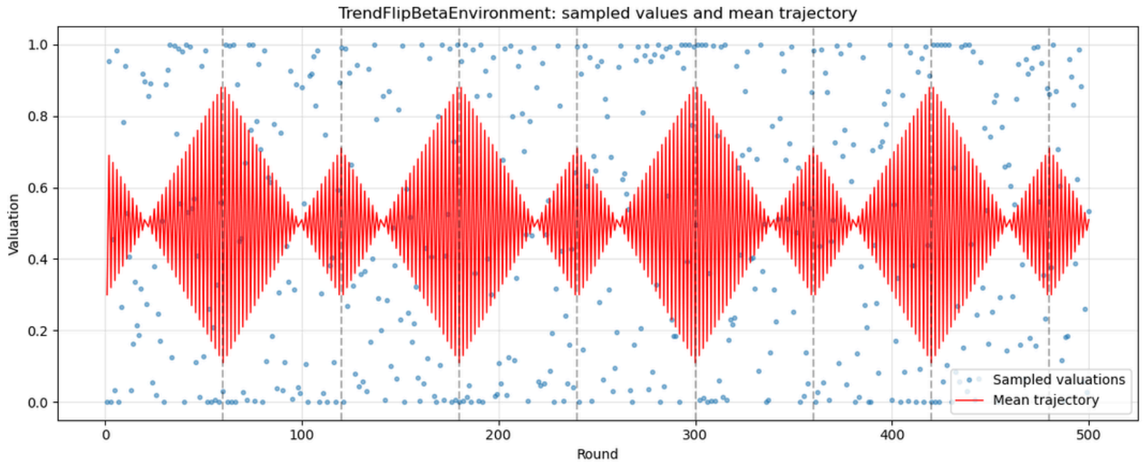
3: Performance on Stochastic Environment

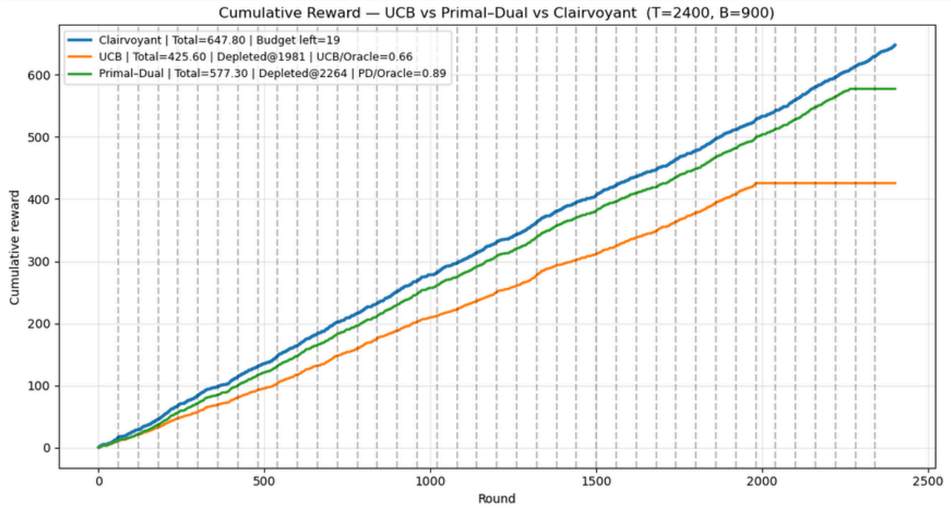


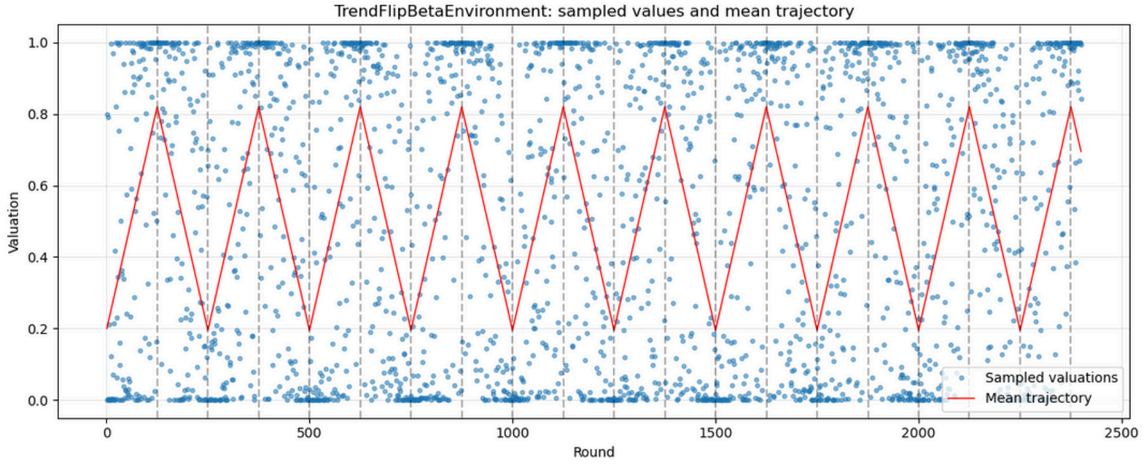
3: Stochastic Environment Arm Selection

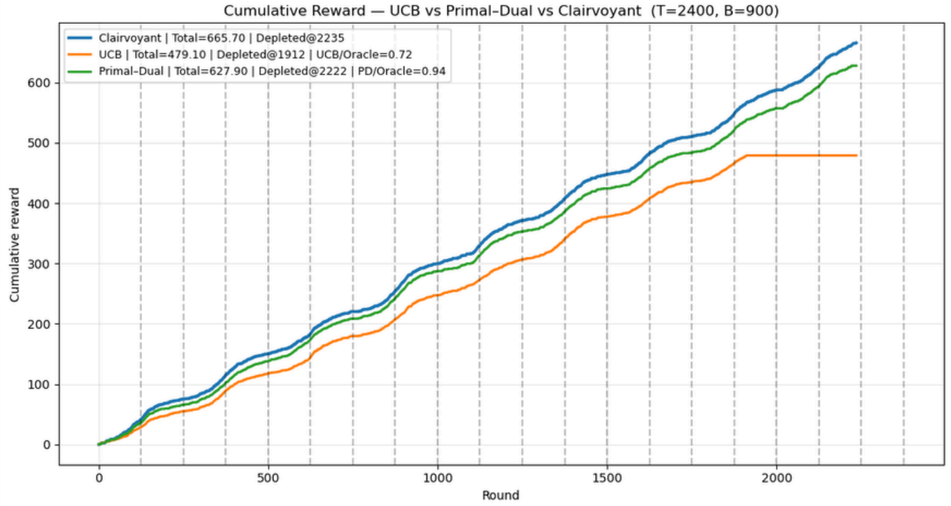


3: Performance on Highly non-Stationary Environment

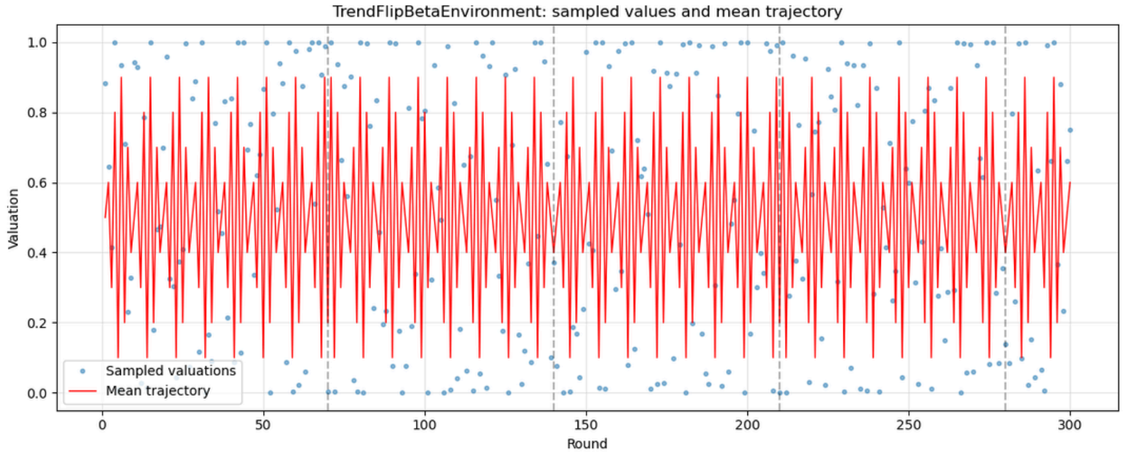


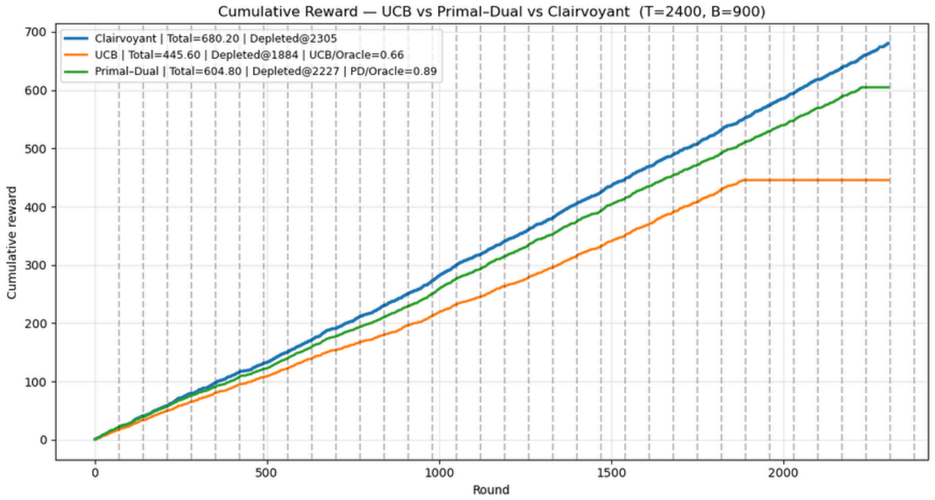


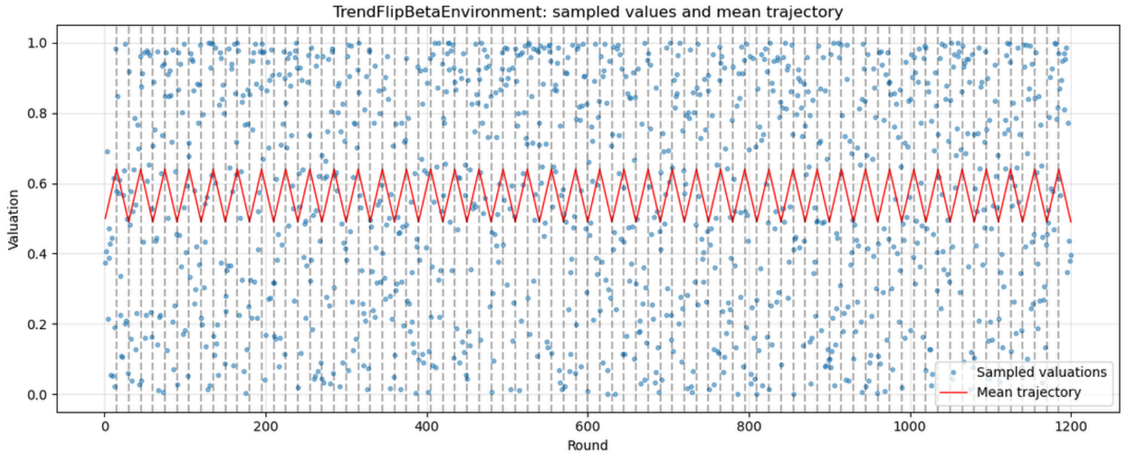


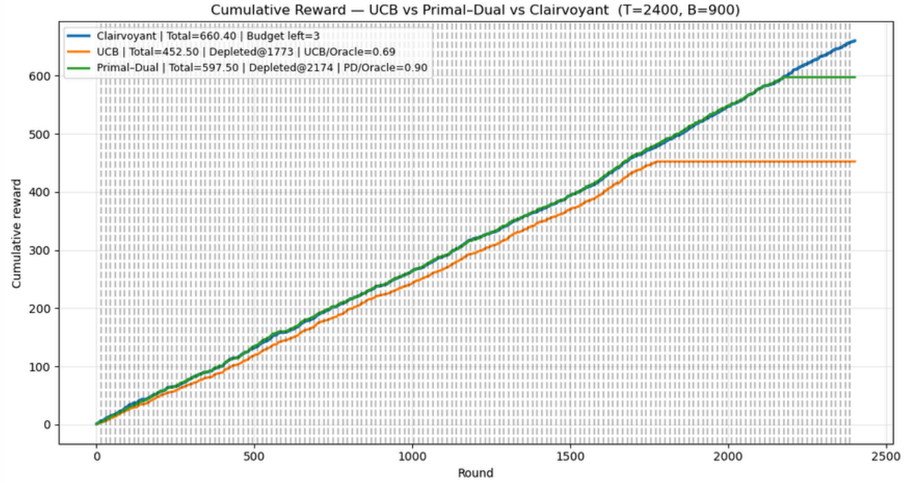


3: Performance on Highly non-Stationary Environment

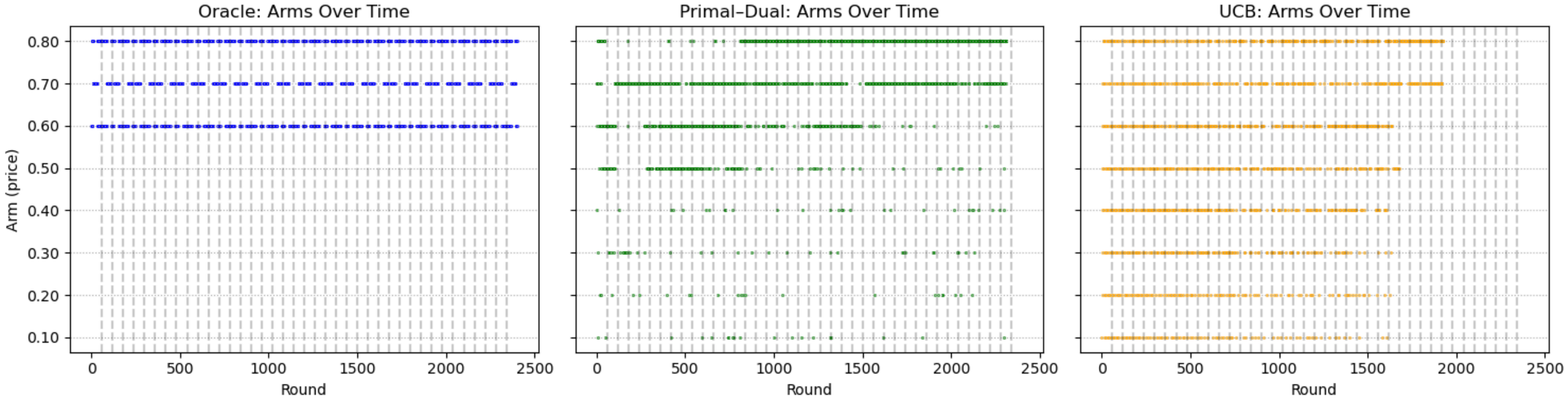








3: Stochastic Environment Arm Selection



4: Multi-Product Best-of-both-worlds

Multi-product pacing strategy

Input: budget B, horizon T, learning rate η, K regret minimizers.

Init: average budget $\rho = B/T$ **, dual variable** $\lambda_0 = 0$.

For each round $t = 1, \ldots, T$:

1. **For each product** $k = 1, \ldots, K$:
   - **Get distribution** ν_t^k **from regret minimizer.**
   - **Sample a bid** b_t^k.
   - **Observe reward** $f_t^k(b_t^k)$ **and cost** $c_t^k(b_t^k)$.
   - **Update regret minimizer with payoff = reward** $-\lambda_{t-1}$**·cost.**
2. **Compute total spend:** $C_t = \sum_{k=1}^{K} c_t^k(b_t^k)$
3. **Update pacing multiplier:** $\lambda_t = \Pi_{[0,1/\rho]}\left(\lambda_{t-1} - \eta(\rho - C_t)\right)$
4. **Subtract spend from budget:** $B \leftarrow B - C_t$.
5. **If** $B < 1$**, stop.**

Baseline:

The reward of the best dynamic policy when the decision maker is aware of the underlying distribution

Regret decomposition:

$$max \sum_{t=1}^{T} \sum_{k=1}^{K} f_t^k \left(b_t^k\right)$$

$$s.t. \sum_{t=1}^{T} \sum_{k=1}^{K} c_t^k \left(b_t^k\right) \leqslant B$$

$$L_t\left(b_t, \lambda\right) = \sum_{k=1}^{K} f_t^k \left(b_t^k\right) - \lambda \left(\sum_{k=1}^{K} c_t^k \left(b_t^k\right) - \rho \right)$$

$$L_t\left(b_t, \lambda\right) = \sum_{k=1}^{K} f_t^k \left(b_t^k\right) - \lambda \sum_{k=1}^{K} c_t^k \left(b_t^k\right) + \lambda\rho$$

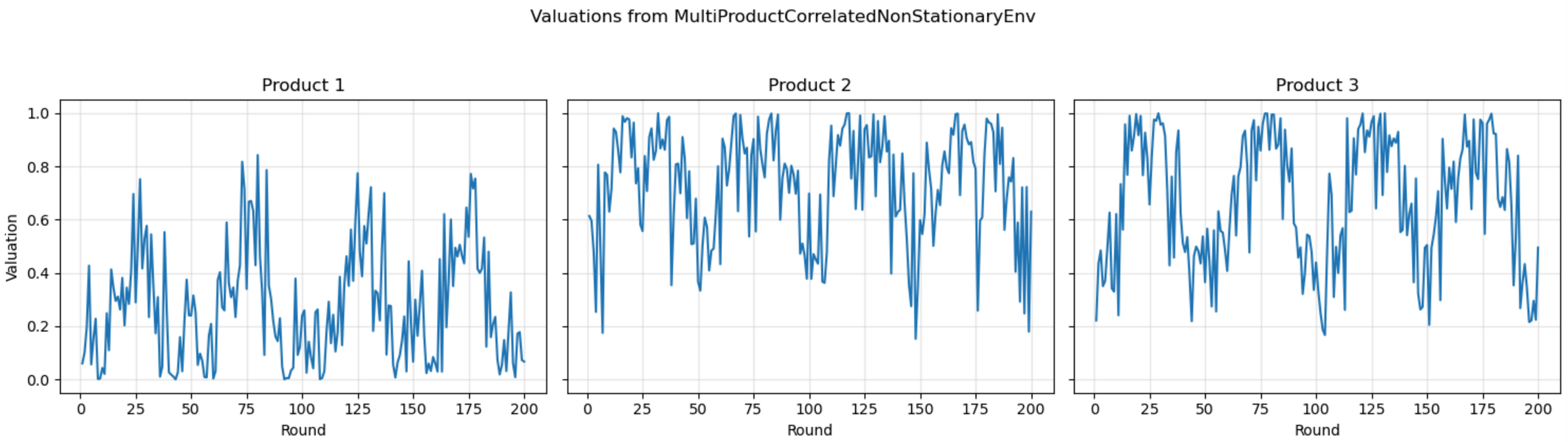
Product Valuations Distribution

Environment 1: Multi-Beta Distribution

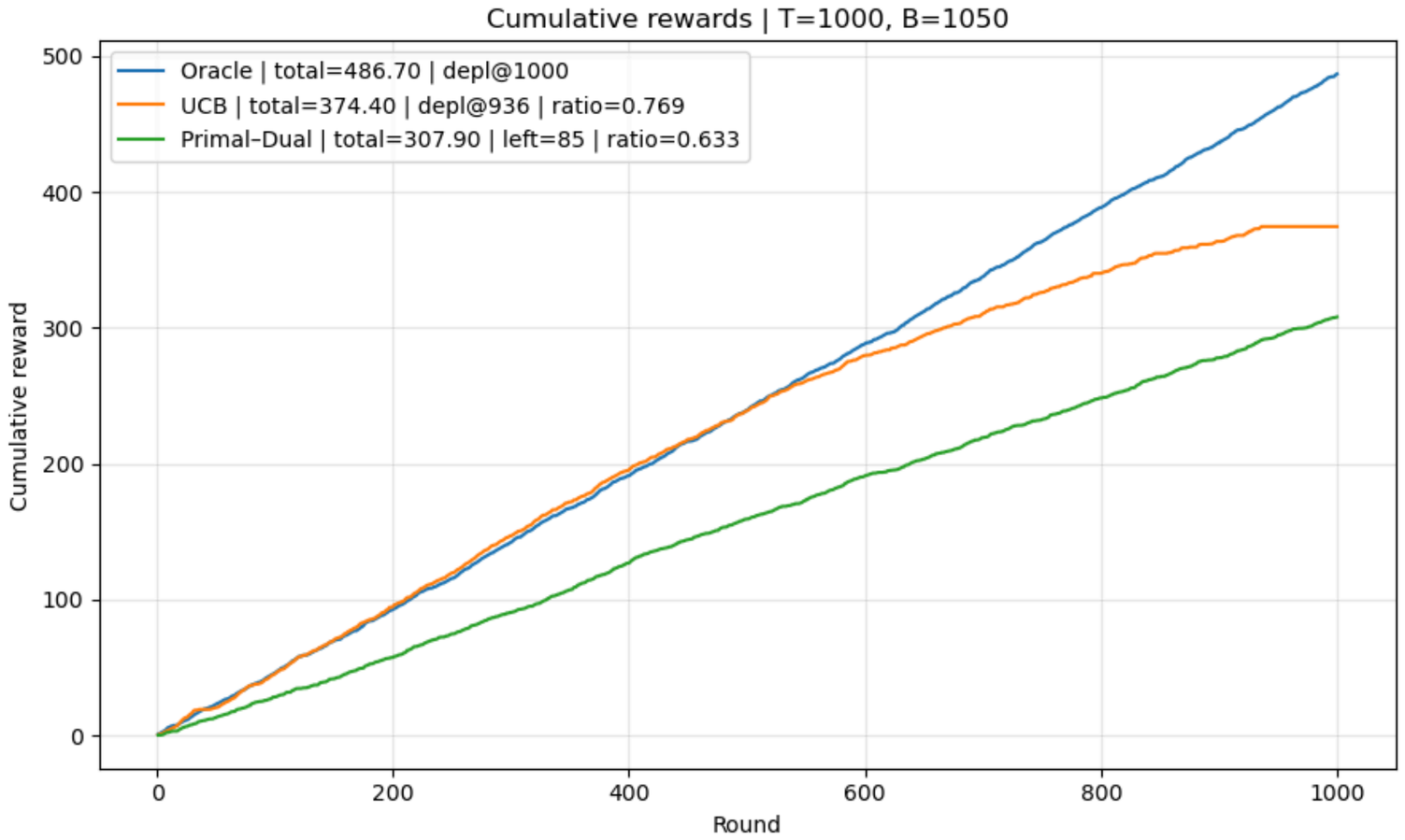
- Stochastic valuations: $v_{t,i} \sim \text{Beta}(a_i, b_i)$
- Multiple product types

Environment 2: Multi highly non-stat. Distribution

- Multiple products
- Correlated valuations

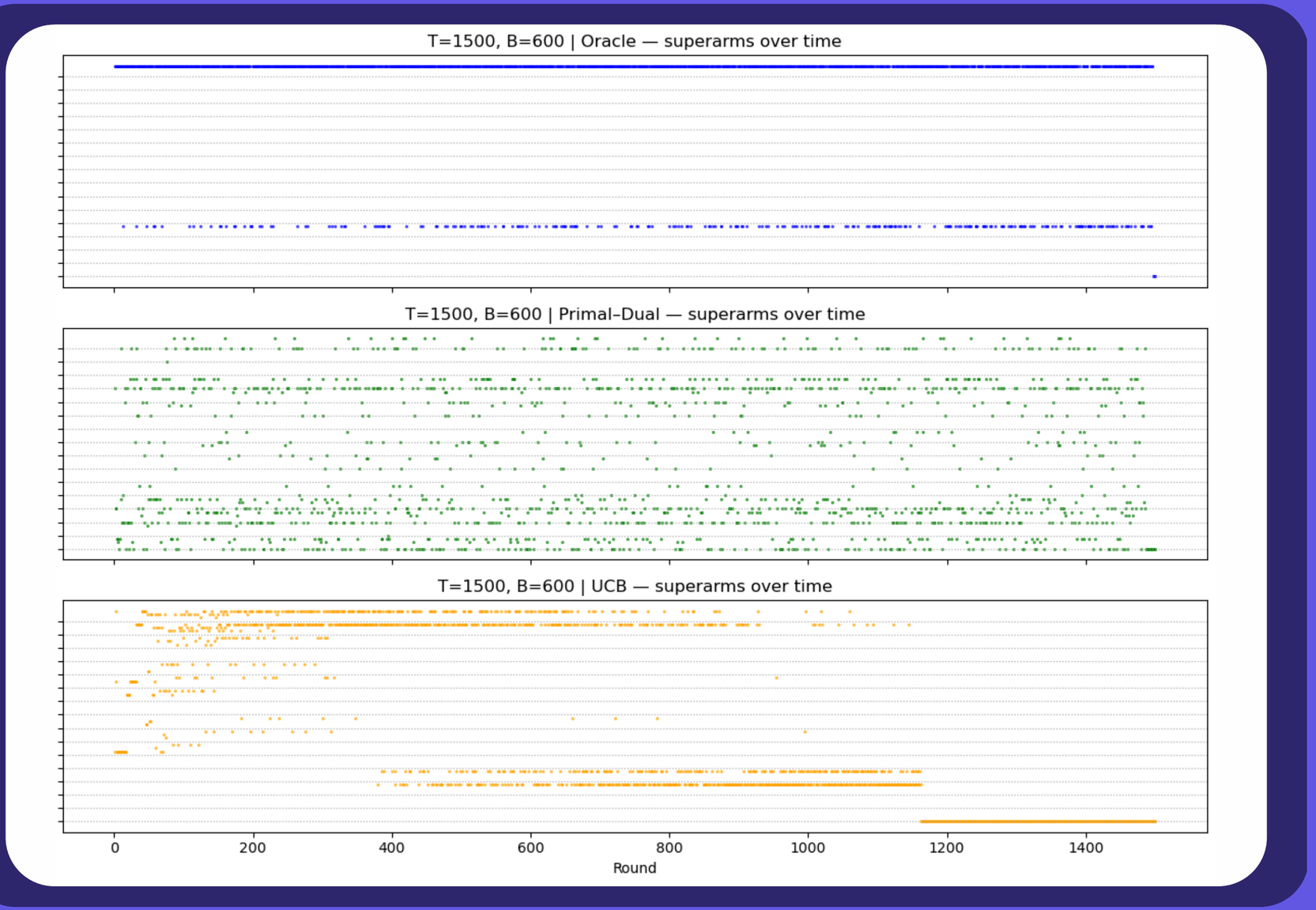


Performance on Stochastic Environment

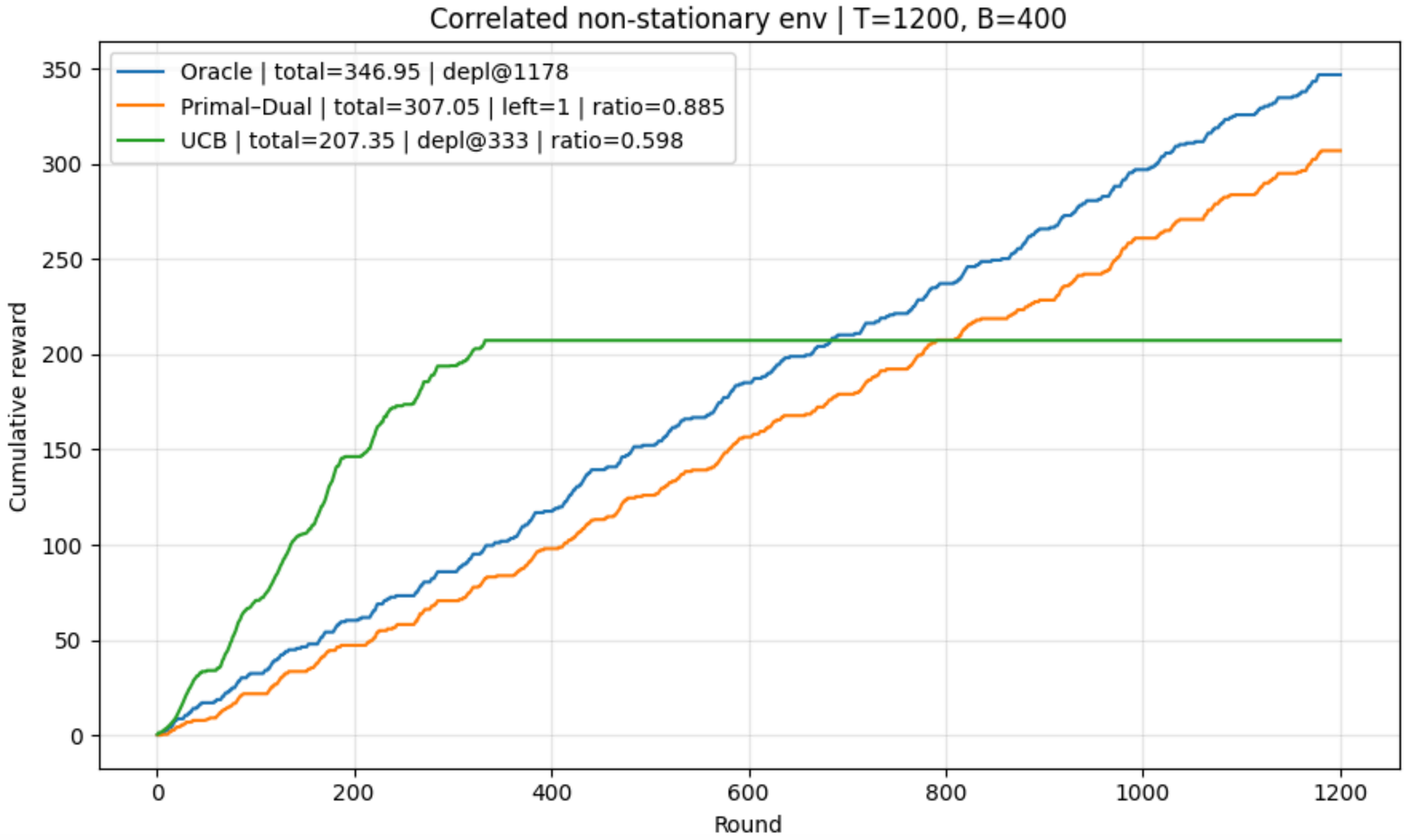


As expected the UCB agents outperforms the primal dual which wastes rounds overexploring a stationary environment, while the UCB capitalizes on the stationarity and once identified the optimal arm exploits it

4: Stochastic environment Arm selection

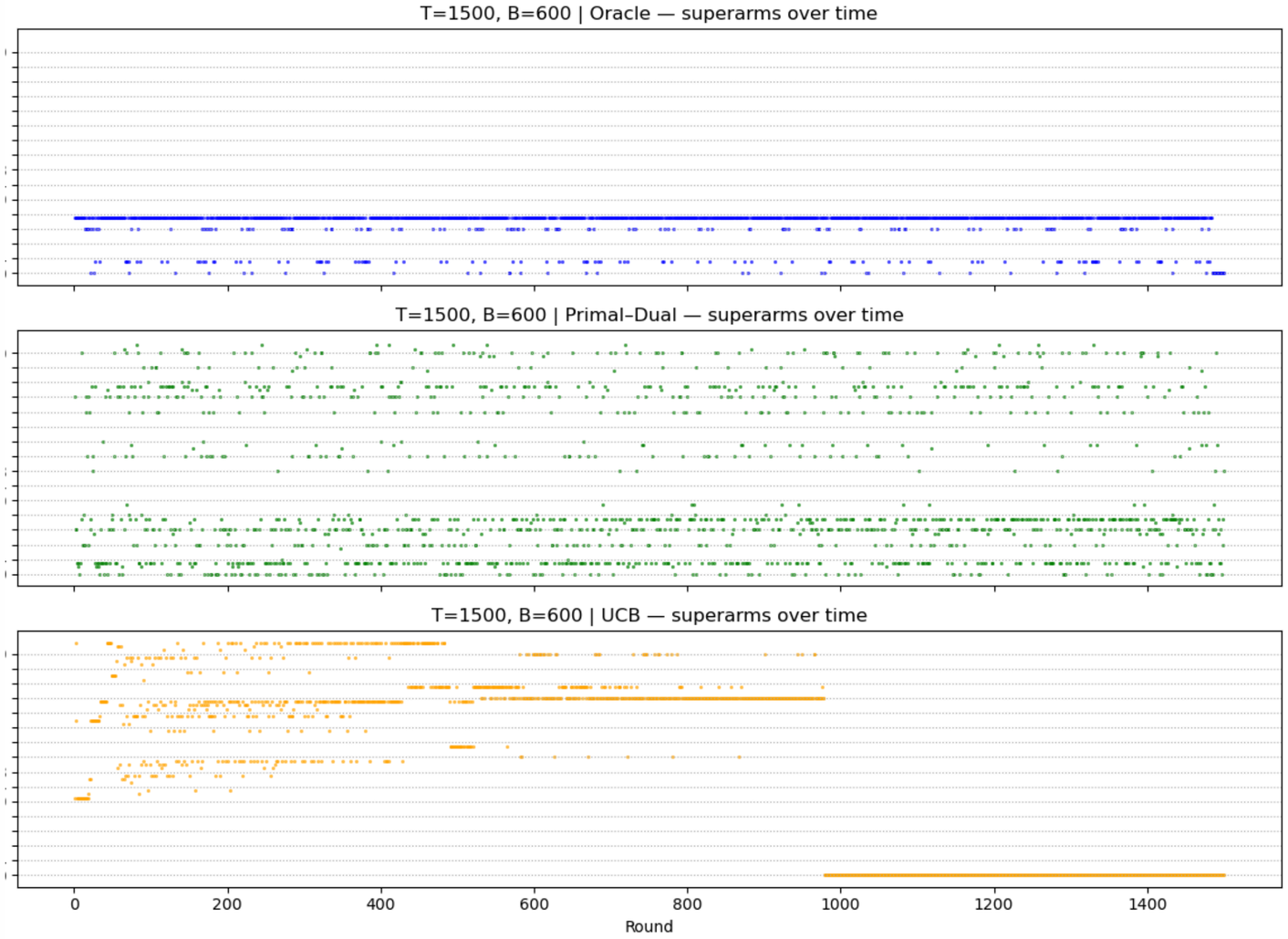


Performance on Highly non-Stationary Environment



On a highly non-stationary environment the constant exploration of the Primal Dual agent alloes it to obtain a much higher reward compared to the UCB which tries to leverage stationarity when it's not present

4: Highly non-Stationary Environment Arm selection



5: Multi-Product non-Stationary Environment

Environment: Piecewise-Stationary Multi-Product
Valuation Model: K intervals with different $\mathrm{Beta}(a_{i,j}, b_{i,j})$ per product per interval

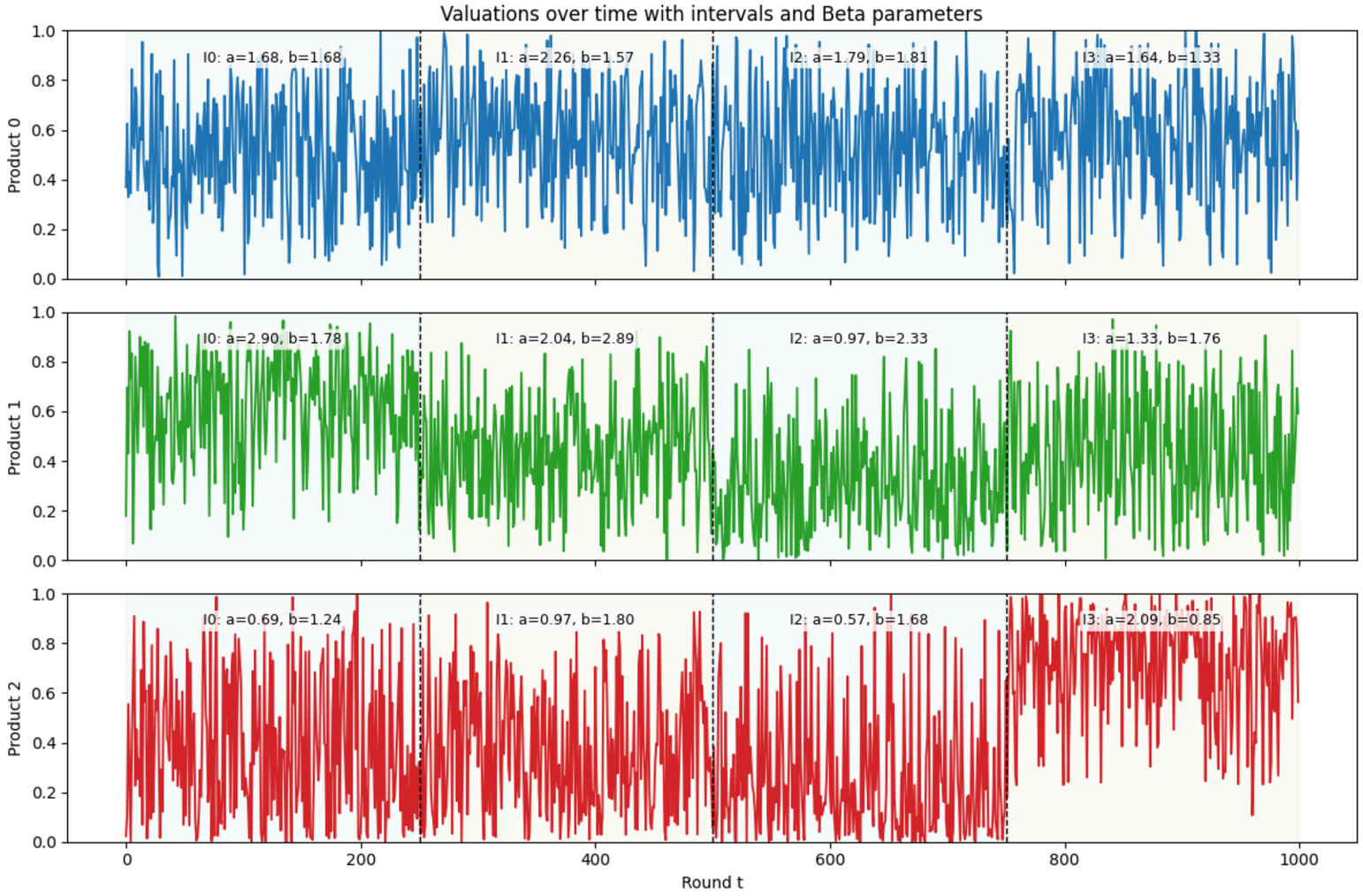
Algorithm: Sliding Window UCB
Windowed Estimates (window size W):

$$\hat{\mu}_{i,t}^{(W)} = \frac{1}{|W_t|} \sum_{s \in W_t} r_{i,s}$$

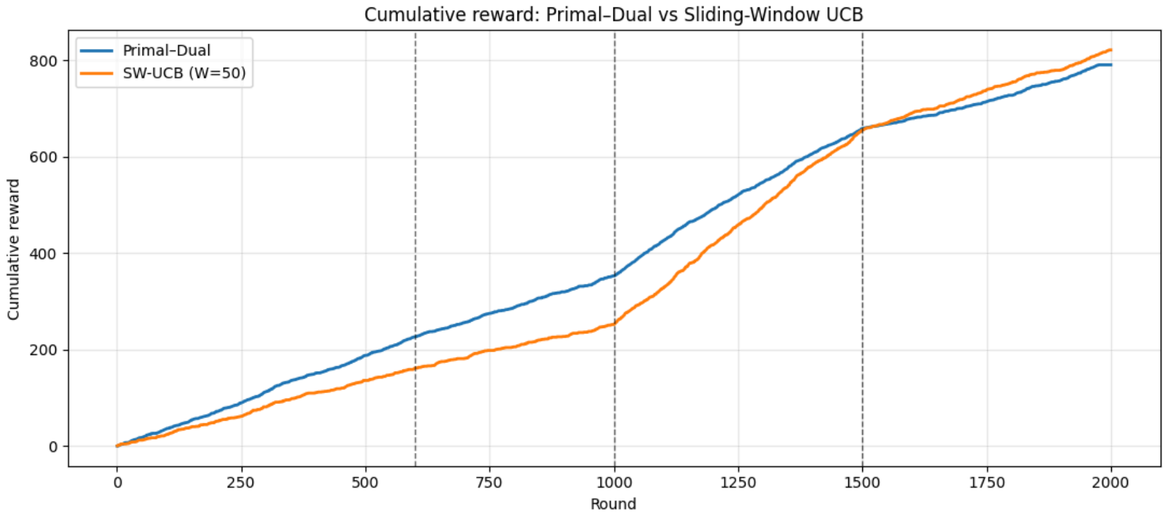
UCB with Sliding Window:

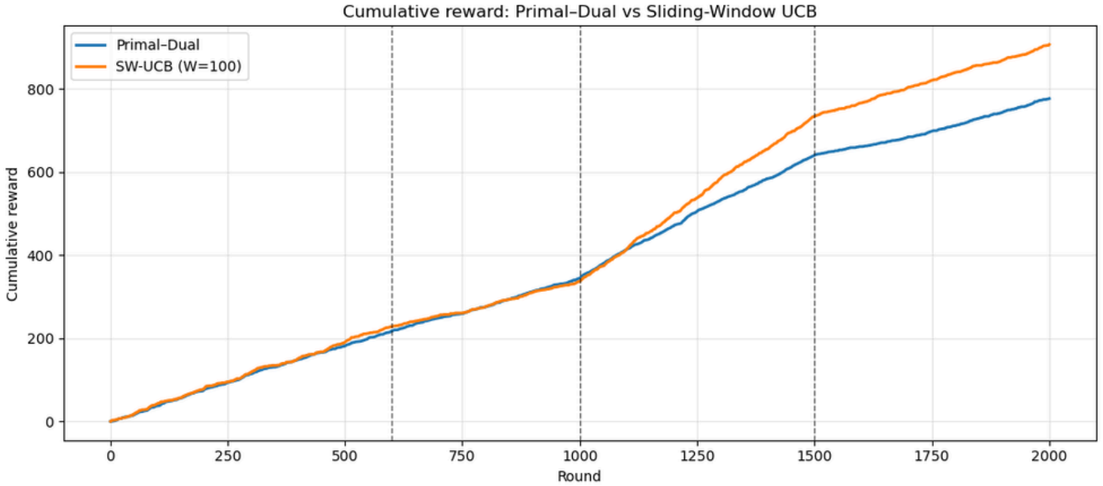
$$\mathrm{arm}_t = \arg\max_i \left[\hat{\mu}_{i,t}^{(W)} + \sqrt{\frac{2 \log T}{|W_t|}} \right]$$

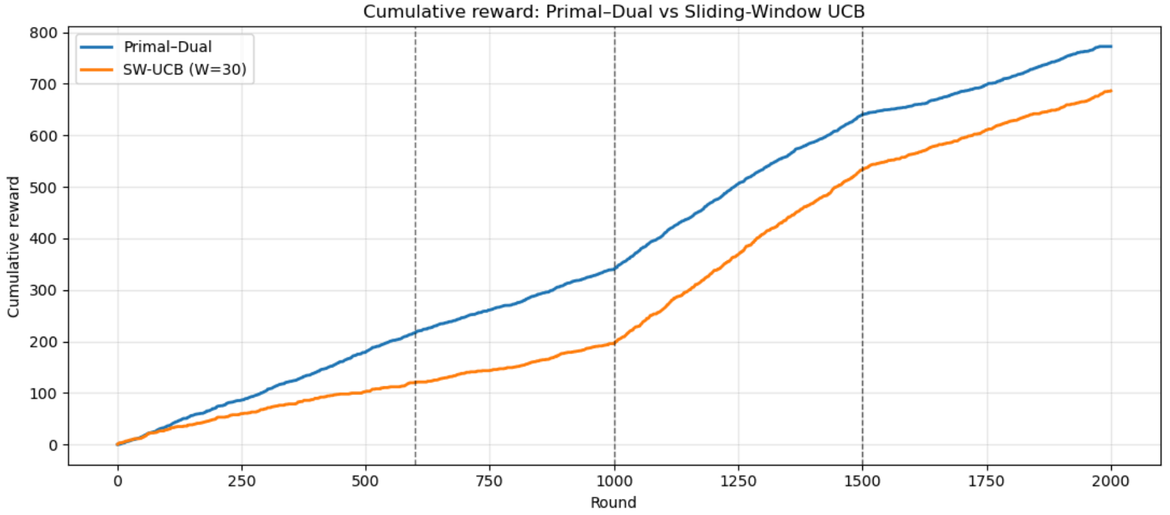
Slightly Non-Stationary Environment

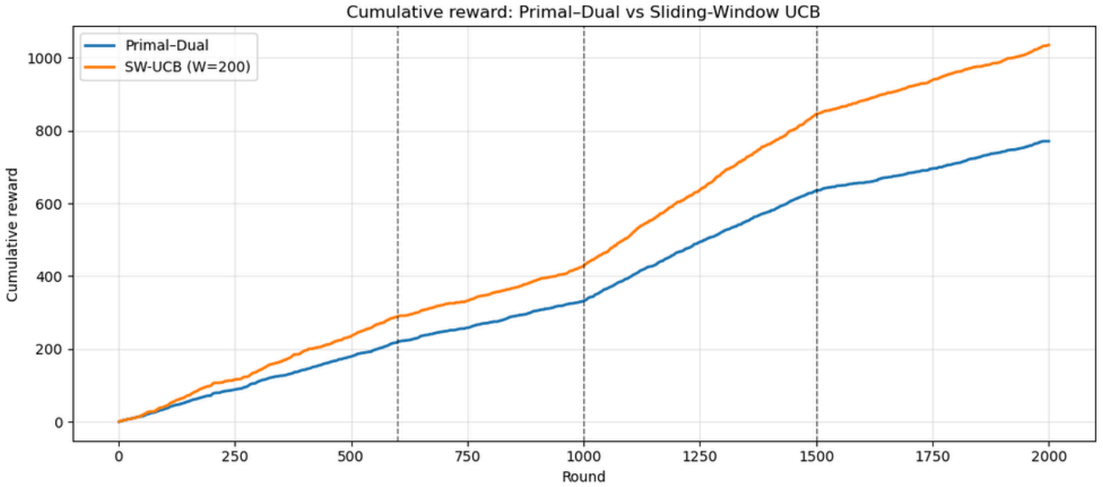


Performance









Final Remarks

- In the single product setting, UCB1 has enough room to spend the initial rounds exploring, and then start exploiting in the second part of the game. Even though a budget constraint is imposed, the agent is still able to “find” the optimal plan and keep regret sublinear, staying close to the Oracle player.
- In requirment 2 it is important to give enough time and budget to learn since the high amount of possibilities requires time to identify the optimal one.
- Once compared, a clear difference can be seen in the algorithms:
  - UCB is a very “switch on / switch off” agent: it soon sets its pick on one arm, and after it switches it does not usually come back to the previous one.
  - Primal-Dual has a more gradual approach. The reasoning resembles more uninformed search, and it usually tries out a lot of combination. It slowly proceeds to be more firm towards a set of possible optimal arms.
- The highly nonstationary environment (Requirement 4), in a multiple products setting, sees the Primal-Dual Algorithm “learn” over time as the environment without falling behind as the UCB does (as we observed in requirment 3)
- In Requirement 5, the effectiveness of the SW-UCB can be seen through the correspondance with the increase in W. However the optimal W is instance dependent and there is identify it beforehand.

Thank you

Jan Bartolini
Lorenzo Biraghi
Francesco Inzerillo
Pratik Rameshwar Potdukhe
Matteo Zanetti