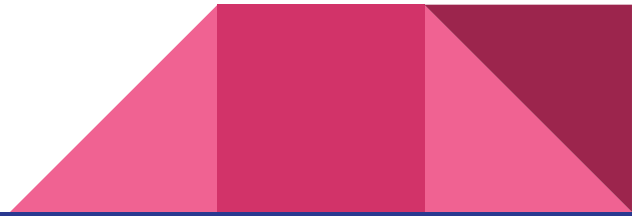
Online Learning Applications Project

Social Influence & Advertising

Pablo Giaccaglia
Gianmario Careddu
Marco Bonalumi
Sara Zoccheddu
Riccardo Ambrosini Barzaghi

Scenario

- E-commerce grocery website
- **Unlimited number** of units of 5 different items can be sold.
- Each product is associated to an advertising campaign.
- **Daily budget B** to spend for advertising products.



Simulator working principles

- Number of users $\mathbf{N_{users}}$ as reference (expected number of users buying on the e-commerce without any advertising campaign)

- The **alpha value** determines what fraction of $\mathbf{N_{users}}$ will land on the e-commerce

- The total alpha values can be greater than 1.

- All the transactions are made with a “generic currency” that takes as a **reference price.**

Campaigns and alpha functions

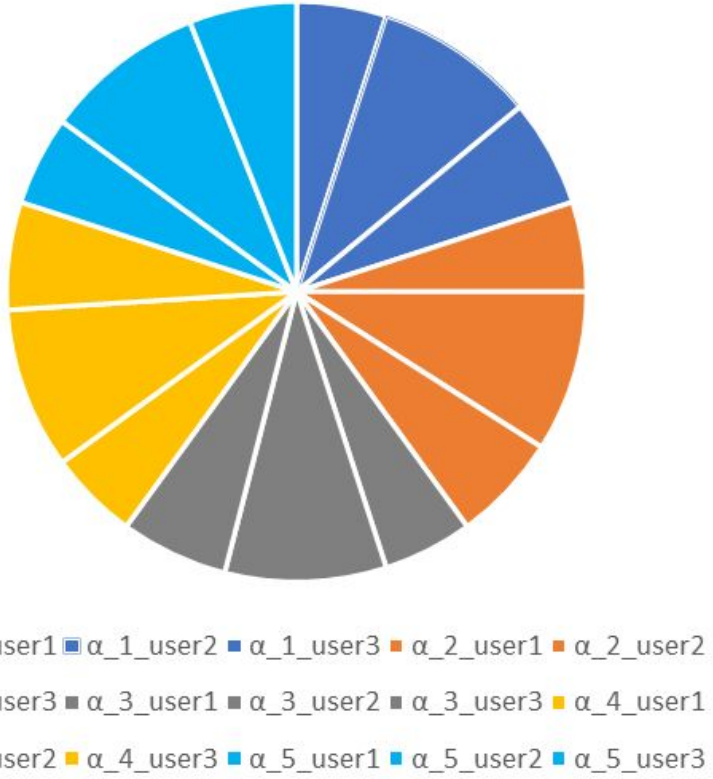
Every campaign receives a budget [0, +∞), and can be seen as a function c: budget → α value

$$\alpha_{\max} = \sum_{c \in C} \alpha_{\max}^{c}$$

The α value produced by a campaign **c** is the sum of the α functions, associated to that campaign, of each user class, weighted by the **class probability**

$$\alpha_c = \sum_{j \in U} p_j * f_{j,c}\left(b * p_j\right)$$

α values: complete picture



Graphs - User navigation

- The navigation of each user class is modelled using a **fully connected graph** and a **secondary list** for each product

- Is it a general solution?

Revenue mathematical formulation

$$\text{Revenue}_{\text{user } i} = p_i * N_{\text{users}} * R_{\text{price}} \sum_{(c,b)\in A} f_{i,c}\,(b * p_i) * E_{\text{profit} \sim \text{user } i}\,(\text{Primary}_c)$$

Expected profit computed via **DFS (depth-first search)** algorithm.

Users

3 classes divided into **4 types**

Described by **2 binary features**: *student/worker* and *living alone/with family*.

Types	Type Probability	Classes	Class probability
FAMILY + STUDENT	0.125	User 1	0.25
FAMILY + WORKER	0.125		
ALONE + STUDENT	0.45	User 2	0.45
ALONE + WORKER	0.3	User 3	0.3

	STUDENT	WORKER
FAMILY	User1	User 1
ALONE	User 2	User 3

User classes: details

User1 (family)

- Segment size: **MEDIUM**
- Resistance to ads: **HIGH**
  (it depends also on other members)
- Activation budget: **MEDIUM**
- Reservation price: **MEDIUM**
- Expected n purchase: **HIGH**
  (many people to satisfy)

User2 (student)

- Segment size: **HIGH**
- Resistance to ads: **LOW**
  (but more spam is needed)
- Activation budget: **HIGH**
- Reservation price: **LOW**
  (little money)
- Expected n purchase: **LOW**

User3 (worker)

- Segment size: **MEDIUM**
- Resistance to ads: **MEDIUM**
- Activation budget: **LOW**
  (they need little exposition to start buying)
- Reservation price: **HIGH**
- Expected n purchase: **LOW**

α functions behavior

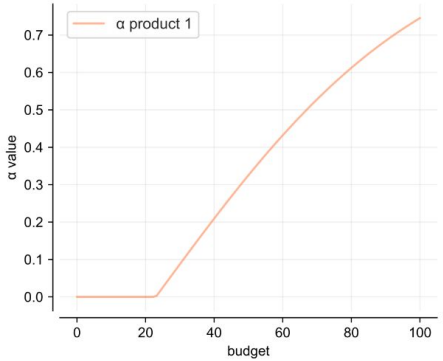
User classes behavior encoded into **sigmoid-like** function:

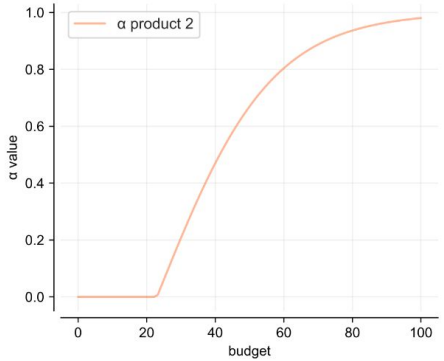
$$f(x) = \max\left(0, -1 + \frac{2}{1 + e^{-\frac{v}{x-a}}}\right)$$

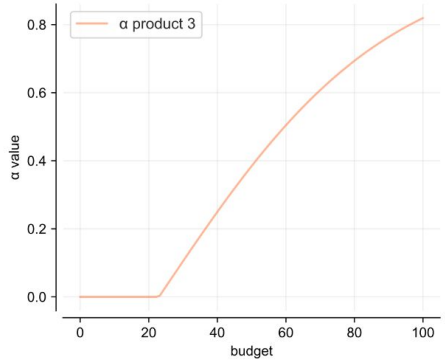
v	Saturation speed
a	Activation point

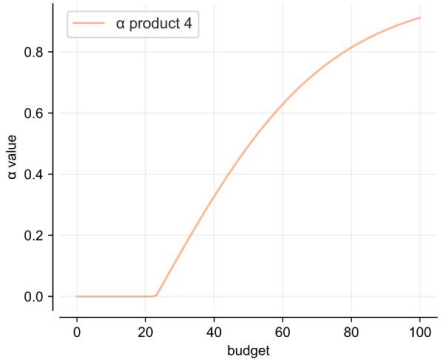
- **Saturation** at value 1
- Scaled by $\alpha_{\substack{\text{max} \\ \text{campaign}}}$

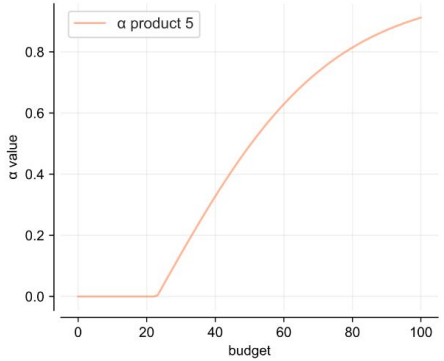
α function example - user 1











Problem details

Allocation of budget over campaigns to maximize the total profit

What to learn:

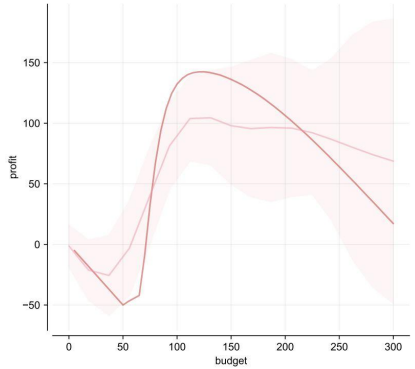
- Function $\mathbf{p_c(b)}$: profit given budget to spend for each campaign

How to employ the knowledge:

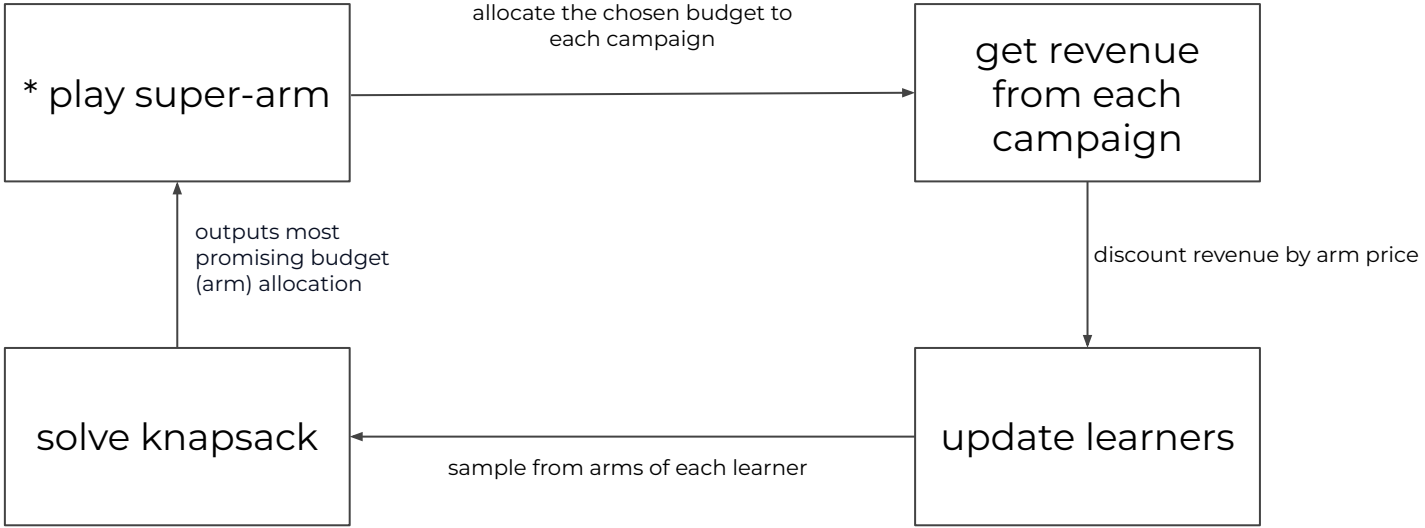
- Perfect estimation of all $\mathbf{p_c(b)}$ means knowing the exact combination of budgets maximizing the profit

In practice:

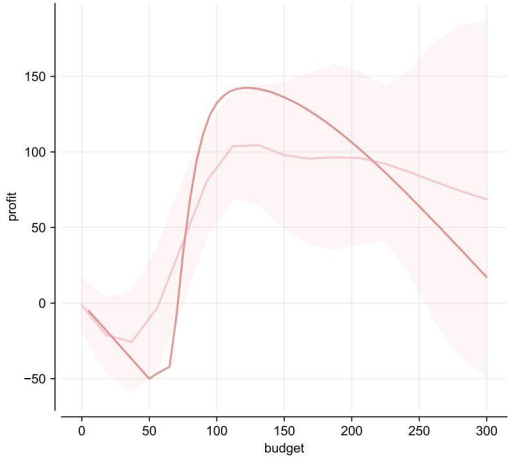
- We can only estimate the functions $\mathbf{p_c(b)}$
- Finding the best combination is an **NP hard problem**



Combinatorial Online Learning Framework



Regularities: profit functions of campaigns

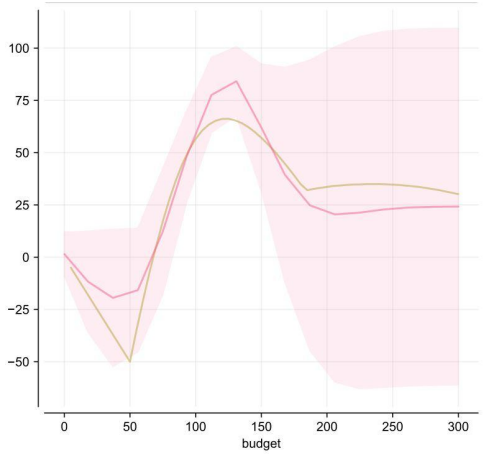


The $p_c(b)$ functions characterized by:

- **loss part**
- **positive slope**
- **saturation**
- **over allocation**

Regularities:

- Correlation among budgets
- Continuity



Experiments & Results

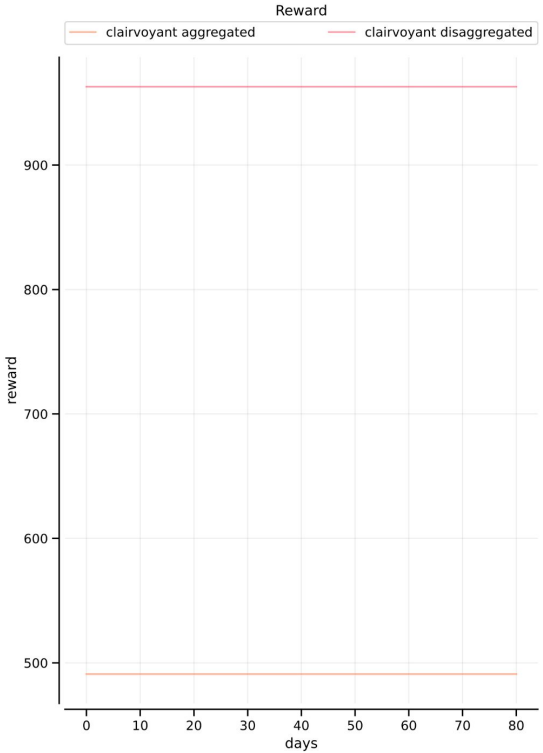
Learners available to wrapper

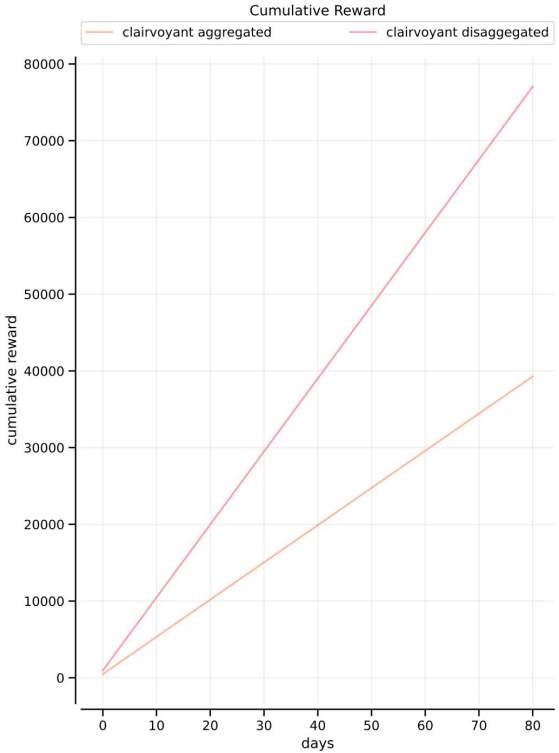
	Full Bandit Name
G-TS	Gaussian Thompson Sampling
GP-TS	Gaussian Process Thompson Sampling
GP-UCB1	Gaussian Process UCB1
CUSUM-GP-UCB1	Cumulative Sum Change Detection Gaussian Process UCB1
CUSUM-GTS	Cumulative Sum Change Detection Gaussian Thompson Sampling
SW-GTS	Sliding Window Gaussian Thompson Sampling
SW-GP-UCB1	Sliding Window Gaussian Process UCB1

Simulations setup

Experiments	100
Days	80
N_{users}	350
Reference price	4 €
Daily budget	300 €
Budget step	5 (step of knapsack solver)
Number of arms	$3\left\lceil (T\log(T))^{1/4} \right\rceil + 1$ = 13

Step 2 - Optimization algorithm: disaggregated vs aggregated case





	Average daily reward	Average cumulative reward
Clairvoyant disaggregated	77040	491
Clairvoyant aggregated	39280	963

Step 3 - Optimization with uncertain α functions

- Noisy α functions
- Aggregated profit over the users (5 profit curves)
- Gaussian Processes to calculate means and variances for arms of Thompson Sampling and UCB
- Thompson Sampling with gaussian priors added for additional reference
- Arms as tuples (campaign, budget).
- Knapsack called to solve combinatorial problem and choose super-arm to play

Algorithms

Combinatorial Thompson Sampling using Gaussian priors (C-GTS)

for each t = 1,2, ...,*T*, do

1) For each arm i = 1, ... , N sample θ_i from distribution $\mathcal{N}(\mu, \sigma^2)$

2) Play *super-arm* $\boldsymbol{a}$ = arg max $\{\boldsymbol{\Sigma}(\theta_i\text{-}cost_i)\}$, subject to combinatorial Knapsack constraints observe r_t

3) update the corresponding μ, σ for each sub-arm composing $\boldsymbol{a}$

Combinatorial Gaussian Process Thompson Sampling (C-GPTS)

for each t = 1,2, ...,*T*, do

1) For each arm i = 1, ... , N sample θ_i from distribution $\mathcal{N}(\mu, \sigma^2)$

2) Play *super-arm* $\boldsymbol{a}$ = arg max $\{\boldsymbol{\Sigma}(\theta_i\text{-}cost_i)\}$, subject to combinatorial Knapsack constraints observe r_t

3) for each sub-arm composing $\boldsymbol{a}$, update μ, σ for all candidates through the GP

Combinatorial Gaussian Process Upper Confidence Bound (C-GPUCB1)

for each t = 1,2, ...,*T*, do

1) Compute □ = 2 log ($t^2\boldsymbol{\pi}^2|A|$ /6□)

2) Play *super-arm* $\boldsymbol{a}$ = arg max ($\boldsymbol{\Sigma}\mu_i + \sigma_i\sqrt{□}$ -*cost*$_i$) subject to combinatorial Knapsack constraints observe r_t

3) for each sub-arm composing $\boldsymbol{a}$, update μ, σ for all candidates through the GP

Noise generation

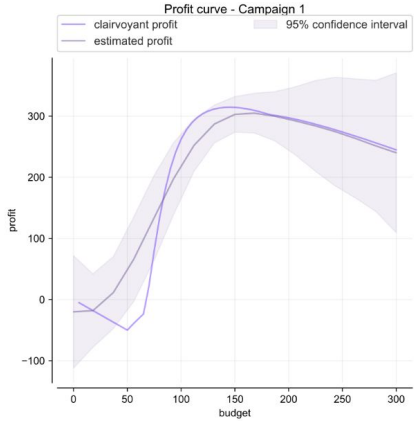
Generated as a **scale factor** for both α functions and expected number of purchased items.

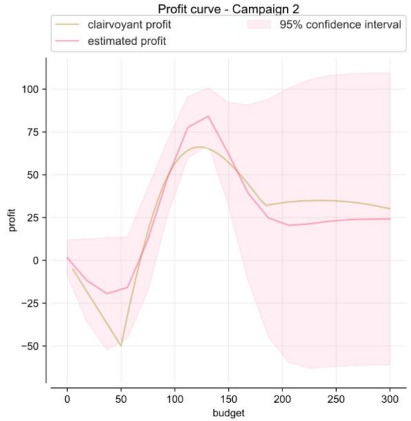
$$\text{noise factor} = 1 + \mathcal{U}(-0.1, 0.1) + \mathcal{G}(0, 0.13)$$

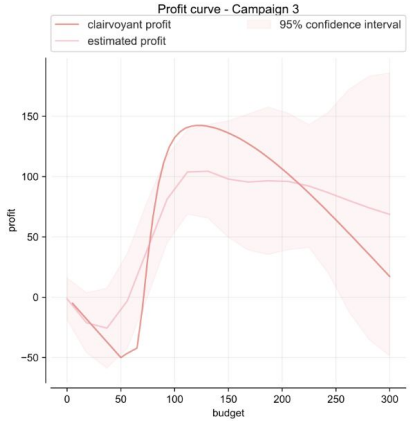
Impact on α functions: reaction of the user to campaign is noisy.

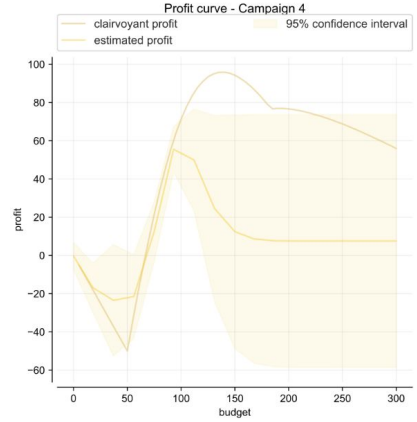
Impact on expected purchases : noise affects the user expected profit.

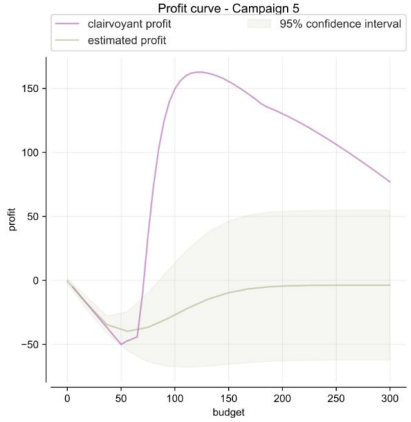
Step 3 - Estimated Profit Curve > GP-TS



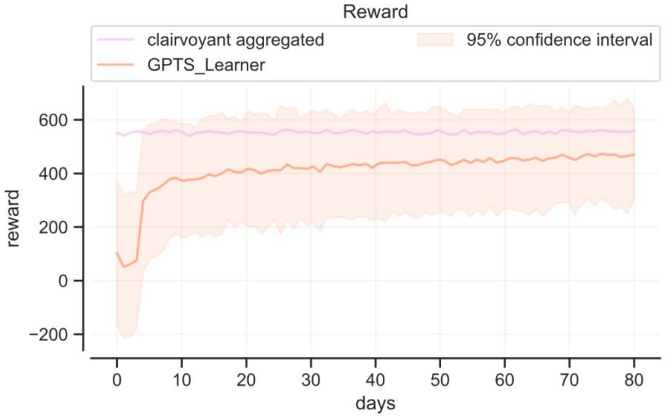


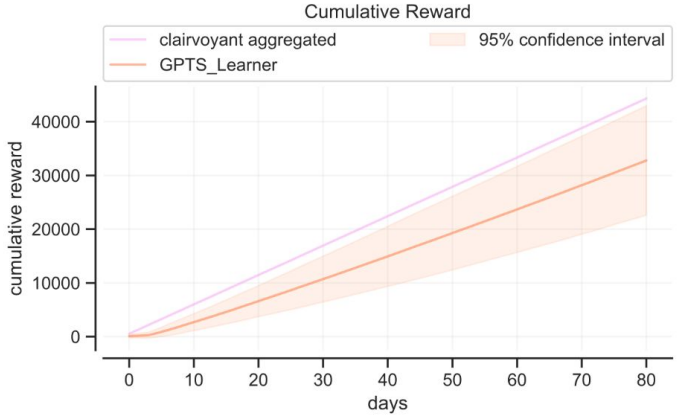






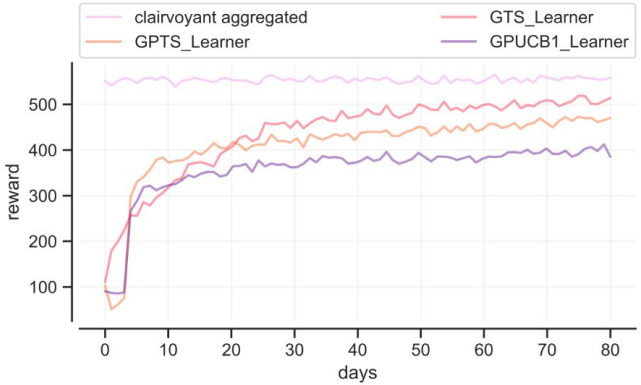
Step 3 - Confidence Intervals > GP-TS



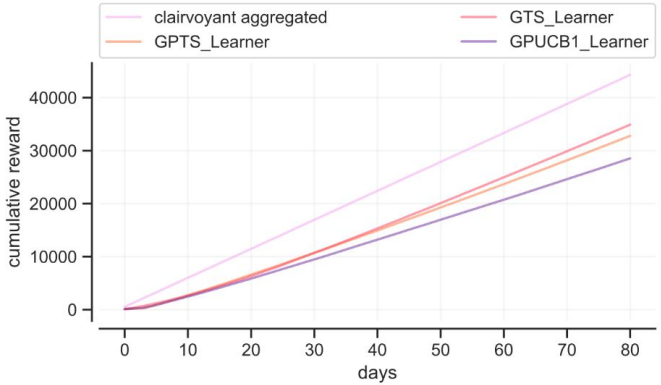


Step 3 - Graphical results

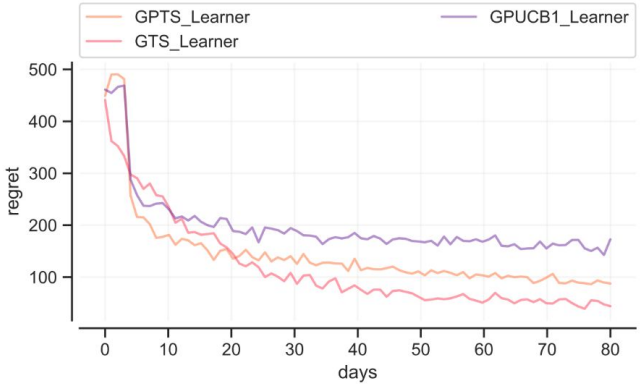
Reward



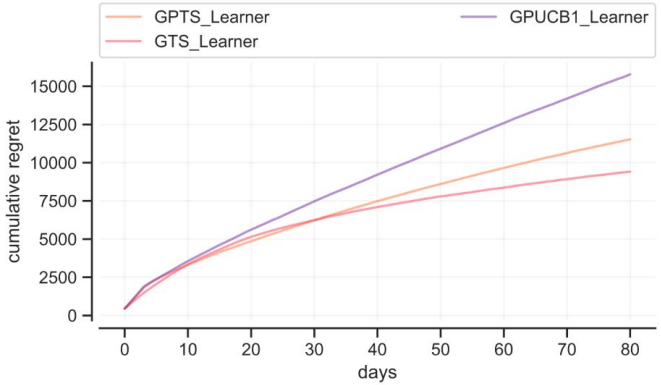
Cumulative Reward



Regret



Cumulative Regret



Gaussian Process Thompson Sampling Regret Bound

CGP-TS regret at time **T**:

- ***γ*** : information gain at time **T**.
- **M** : number of arms
- **σ** : maximum variance of the **GP**
- **Λ > 0** : Lipschitz constant (max regret slope)
- **C** : number of subsets (campaigns)
- **δ**: confidence for the regret bound.

$$R_T \leq \sqrt{\frac{2\Lambda^2}{\log\left(1+\frac{1}{\sigma^2}\right)} CTB \sum_{k=1}^{C} \gamma_{k,T}} \quad \text{w.p. } 1-\delta$$

$$B = 8\log\left(2\frac{T^2 MC}{\delta}\right)$$

GP-TS & GP-UCB1 Regret Bounds

Since the kernel of the Gaussian Process Bandits are squared exponential, the term $\sum_{i=1}^{C} \gamma_{i,NM_i}$ can be bounded by $\mathcal{O}\left(C \log(NM)^{(d+1)}\right)$, thus the following upper bound is provided:

$$\mathcal{R}_N(\mathfrak{U}) = \mathcal{O}\left(C\sqrt{N \log\left(CN^3M^3\right)(\log\left(NM\right))^{(d+1)}}\right)^{*}$$

Bound scales:

- **linearly** in the number of subsets **C**,
- **logarithmically** in the number of arms **M**.

* When Gaussian Processes Meet Combinatorial Bandits: GCB

Combinatorial Gaussian Thompson Sampling Regret Bound

$$\mathscr{R}_T(\mathrm{CGTS}) = O\left(C|M|\frac{\log(T)}{\Delta_{C,\min}}\right)$$

C-GTS regret at time **T**:

- **M**: number of arms
- **C**: number of subsets (campaigns)
- $\Delta_{\mathbf{C,\ min}}$: $\Delta_{\min}$ when **considering all the campaigns**

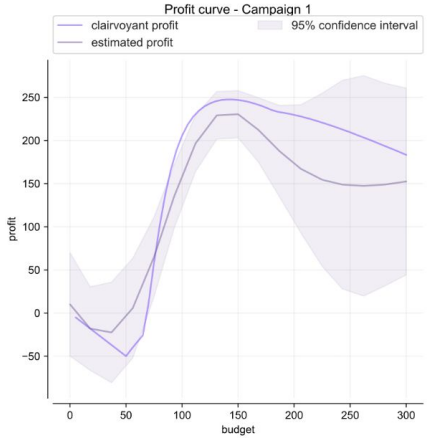
Step 4 - Optimization with uncertain α functions and number of items sold

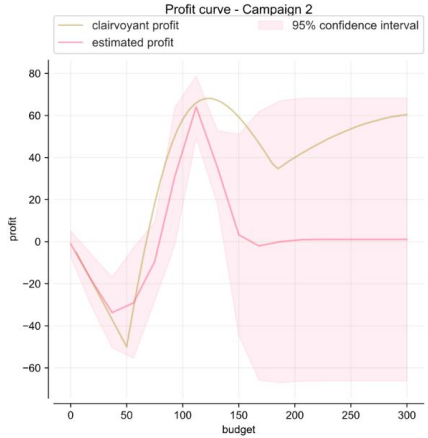
- The algorithms still work because **the learning is bound to the reward** received by the environment, no direct impact of additional noise on learners.

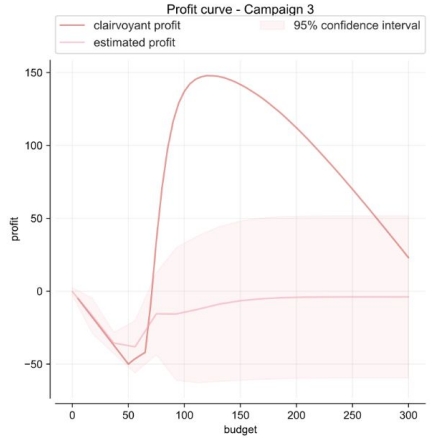
- Bandit actions are the **same as before** and so is the **update phase**

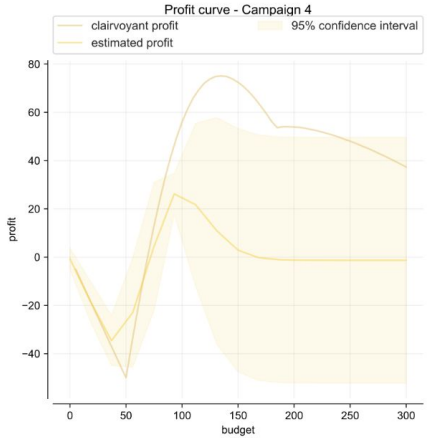
- The reward received will be even **more noisy**, so the regret will probably be larger.

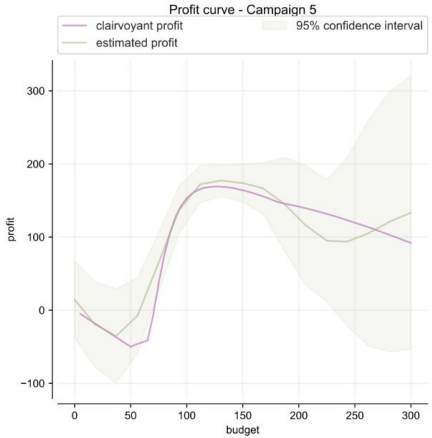
Step 4 - Estimated Profit Curve > GP-TS



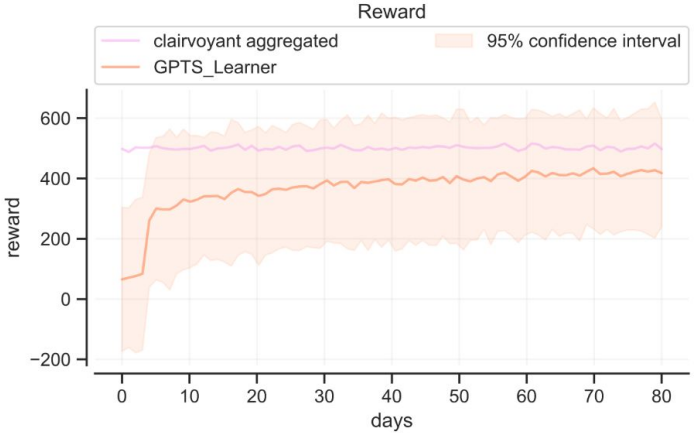


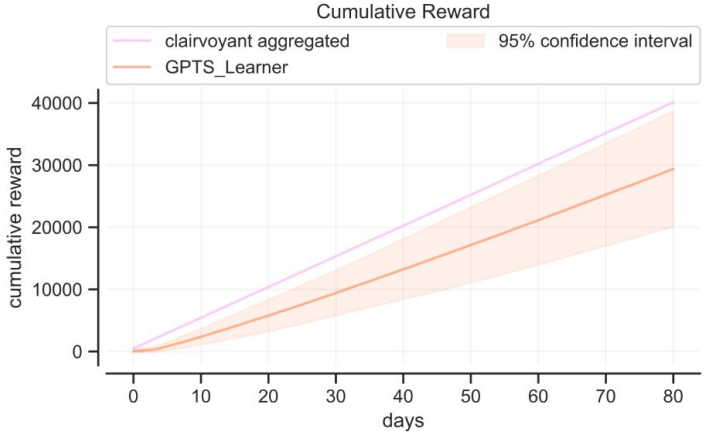


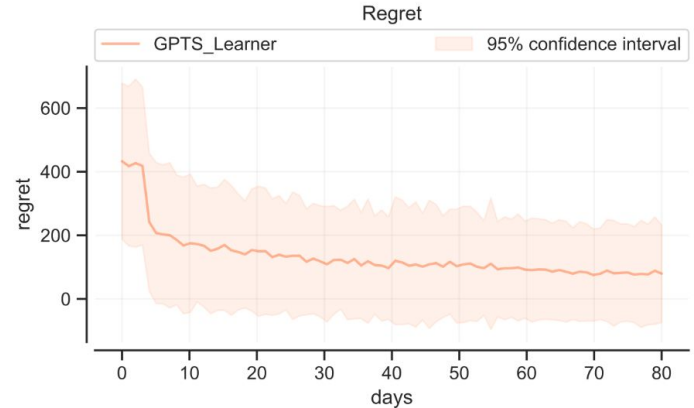




Step 4 - Confidence Intervals > GP-TS

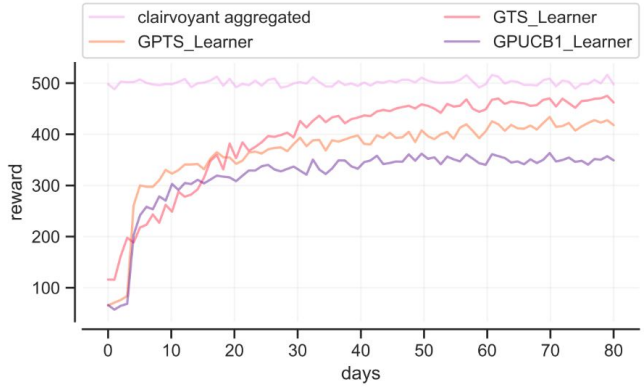




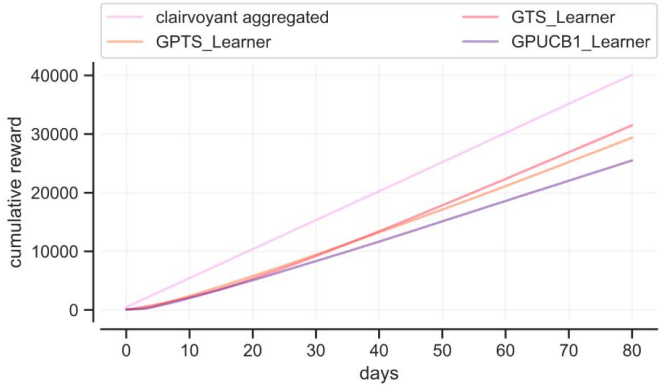


Step 4 - Graphical results

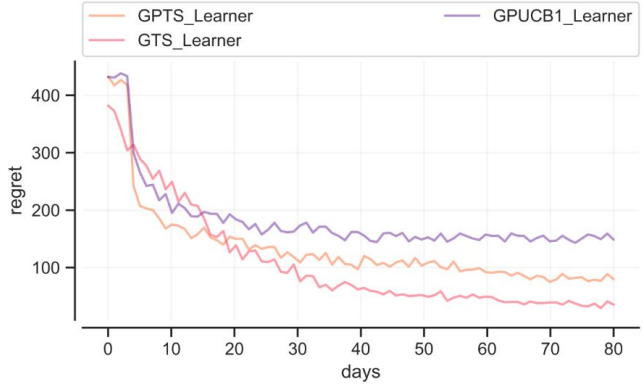
Reward



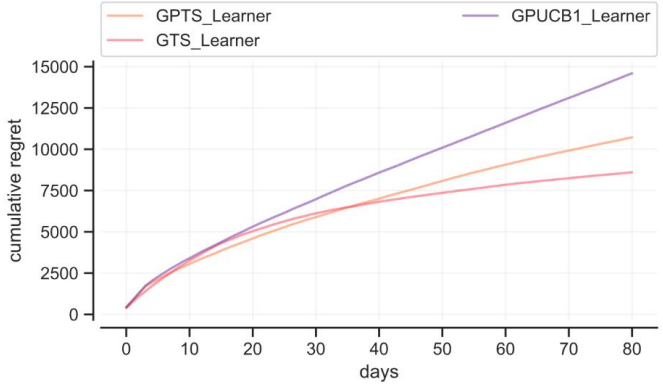
Cumulative Reward



Regret



Cumulative Regret



Step 3 & 4 : Conclusions

- GTS and GP-TS that always outplayed GP-UCB.
- GPTS worked better than GTS when considering a larger number of arms
- GPTS gave the best results when fine-tuning the kernel values of the GP.

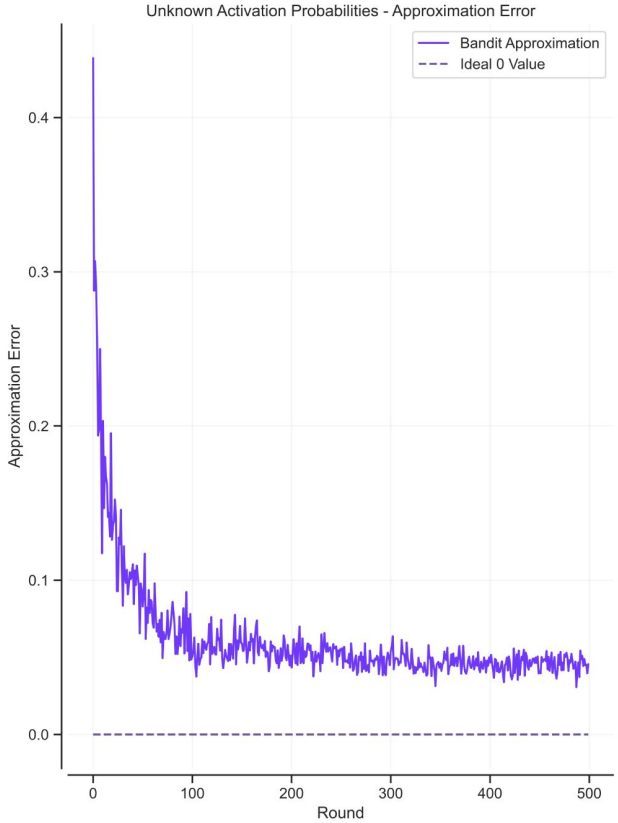
Step 5: Optimization with uncertain graph weights

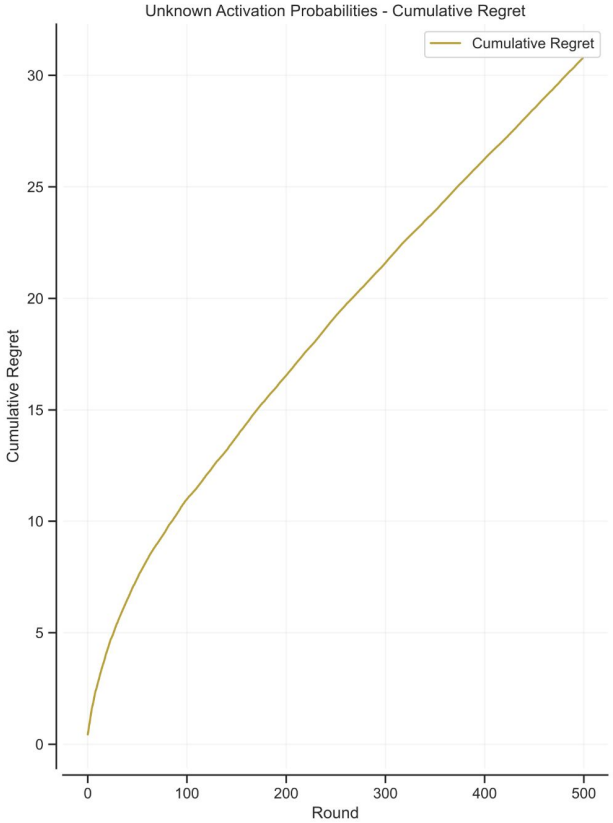
- **Influence episodes** & **Monte Carlo Sampling** employed to tackle the problem.
- Monte Carlo Sampling was employed to find the best seed for observing a large number of edges activating in an influence episode.
- The collected data about activations updates **Beta parameters** associated with each edge.
- **Beta distributions** samples are used for Monte Carlo and provide the final estimated weights.

Step 5: Parameters

Simulations	500
Delta	0.2
Epsilon	0.1
Seeds (S)	1
Monte Carlo Repetitions	$R = \frac{1}{\epsilon^2}\log(\lvert S\rvert + 1)\log\left(\frac{1}{\delta}\right)$ ≈ 30

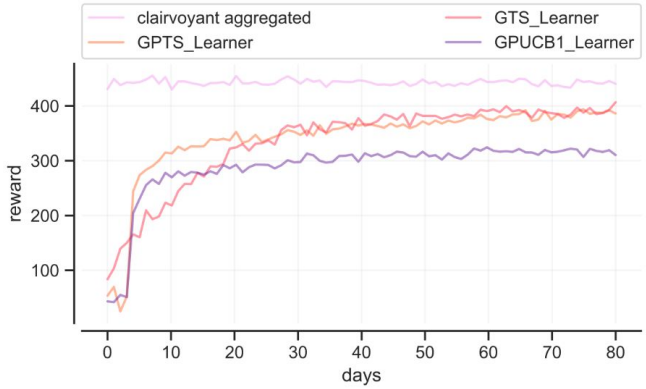
Weights estimation



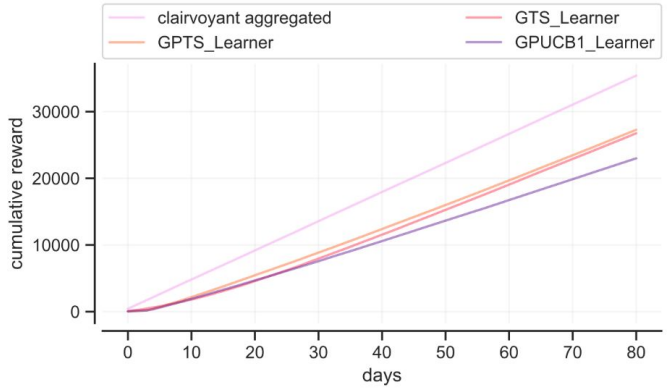


Step 5 - Graphical results

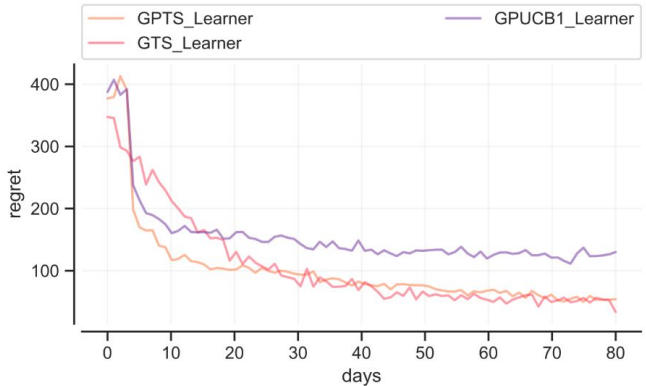
Reward



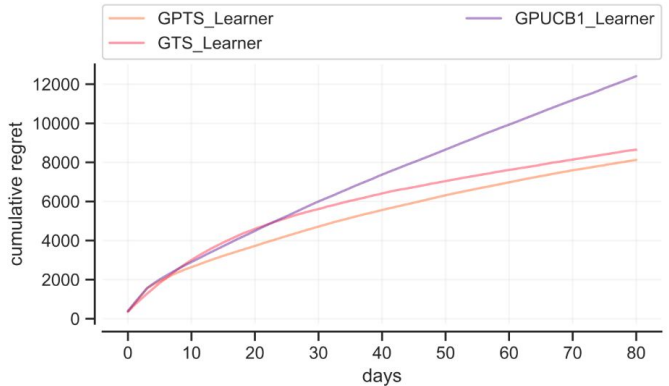
Cumulative Reward



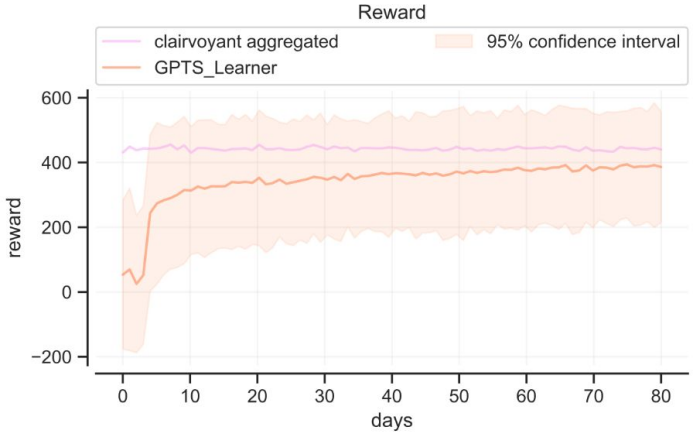
Regret

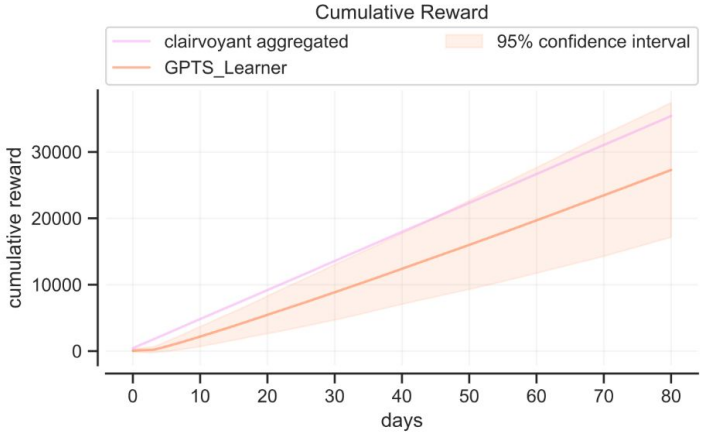


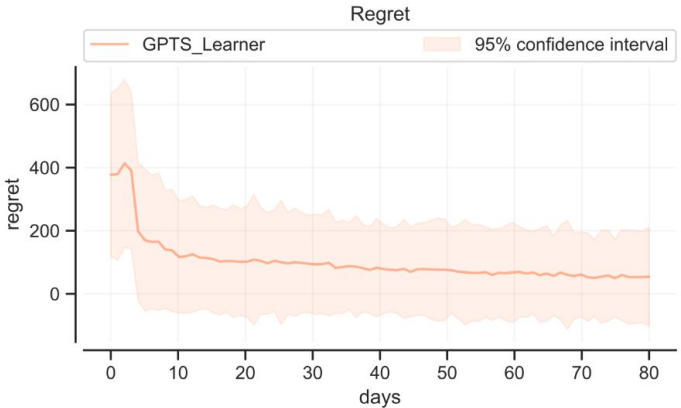
Cumulative Regret



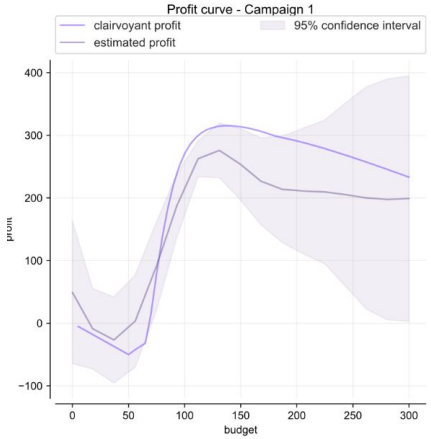
Step 5 - Confidence Intervals > GP-TS

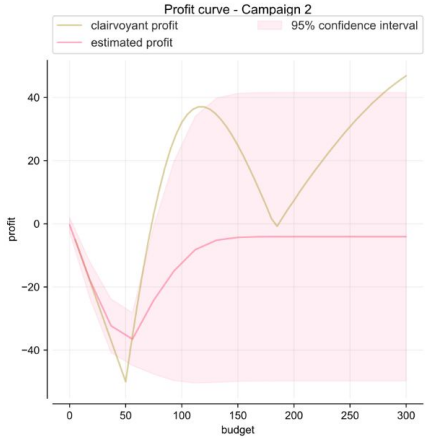


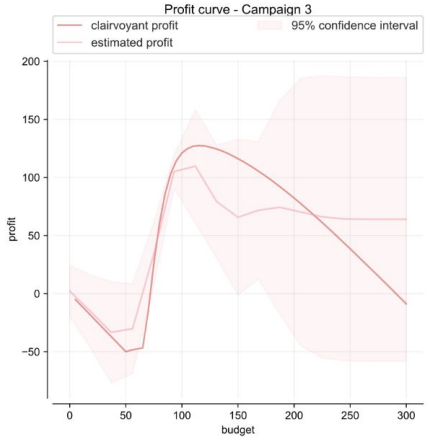


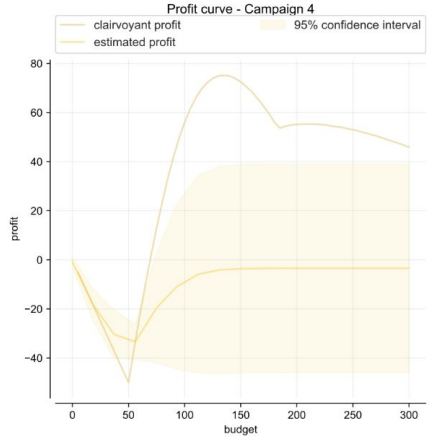


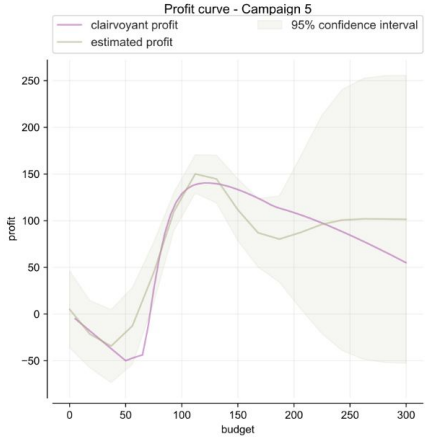
Step 5 - Estimated Profit Curve > GP-TS











Step 6 - Non-stationary demand curve

- Demand curve subject to **abrupt changes**
- Each phase has **different user probabilities** and **e-commerce traffic**
- **Change Detection** and a **Sliding-Window** algorithms employed

Step 6 - Abrupt changes

Context	Season	Number of users
Average e-commerce traffic	1 april - 30 april	350
New competitor enters the marker	1 may - 28 may	175
Competitor turns out to be of lower quality & more expensive	29 may - 20 june	700

Algorithms - Sliding Window

Sliding Window Combinatorial Thompson Sampling using Gaussian priors (C-GTS)

for each t = 1,2, ...,*T*, do

1) For each arm i = 1, ... , N sample θ_i from distribution $\mathcal{N}(\mu, \sigma^2)$

2) Play *super-arm* $\boldsymbol{a}$ = arg max $\{\boldsymbol{\Sigma}(\theta_i\text{-}cost_i)\}$, subject to combinatorial Knapsack constraints observe r_t

3) update the corresponding μ, σ for each sub-arm composing $\boldsymbol{a}$, considering only samples in the current window (t-w, t)

Sliding Window Combinatorial Gaussian Process Upper Confidence Bound (C-GPUCB1)

for each t = 1,2, ...,*T*, do

1) Compute $\square = 2 \log (t^2\boldsymbol{\pi}^2|A| /6\square)$

2) Play *super-arm* $\boldsymbol{a}$ = arg max $(\boldsymbol{\Sigma}\mu_i + \sigma_i\sqrt{\square} - cost_i)$ subject to combinatorial Knapsack constraints observe r_t

3) for each sub-arm composing $\boldsymbol{a}$, update μ, σ for all candidates through the GP, considering only samples in the current window (t-w, t)

Algorithms - CUSUM

Cumulative Sum Combinatorial Thompson Sampling using Gaussian priors (C-GTS)

for each t = 1,2, ...,*T*, do

1) For each arm i = 1, ... , N sample θ_i from distribution $\mathcal{N}(\mu, \sigma^2)$

2) Play *super-arm (*subject to combinatorial Knapsack constraints***)***,* $\boldsymbol{a}$ = arg max $\{\boldsymbol{\Sigma}(\theta_i\text{-}cost_i)\}$ with probability 1-$\boldsymbol{\alpha}$, play random s*uper-arm w*ith probability $\boldsymbol{\alpha}$, observe r_t

3) update the corresponding μ, σ for each sub-arm composing ***a,*** considering only samples after the last detection.

Cumulative Sum Combinatorial Gaussian Process Upper Confidence Bound (C-GPUCB1)

for each t = 1,2, ...,*T*, do

1) Compute □ = 2 log ($t^2\boldsymbol{\pi}^2|A|$ /6□)

2) Play *super-arm (*subject to combinatorial Knapsack constraints***)***,* $\boldsymbol{a}$ = arg max ($\boldsymbol{\Sigma}\mu_i + \sigma_i\sqrt{□} - cost_i$) with probability 1-$\boldsymbol{\alpha}$, play random s*uper-arm w*ith probability $\boldsymbol{\alpha}$, observe r_t

3) for each sub-arm composing ***a,*** update μ, σ for all candidates through the GP, considering only samples after the last detection

Step 6 - CUSUM

The first **M** valid samples are used to produce the **reference point**

Empirical mean of arm a over the first **M** valid samples $\bar{X}_a^0$

From the M+I-th valid sample on, we check whether there is a change

- **Positive deviation** from the reference point at t : $s_a^+(t) = \left(x_a(t) - \bar{X}_a^0(t)\right) - \epsilon$
- **Negative deviation** from the reference point at t : $s_a^-(t) = -\left(x_a(t) - \bar{X}_a^0(t)\right) - \epsilon$

Cumulative positive deviation from the reference point at t : $g_a^+(t) = \max\{0, g_a^+(t-1) + s_a^+(t)\}$

Cumulative negative deviation from the reference point at t : $g_a^-(t) = \max\{0, g_a^-(t-1) + s_a^-(t)\}$

We have a change if : $g_a^-(t) > h \quad \text{or} \quad g_a^+(t) > h$

Then, **CUSUM reset**

Step 6 - SW & CUSUM Parameters

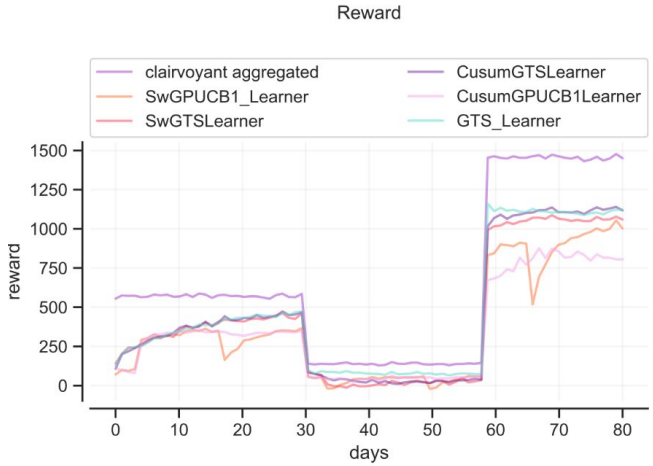
Sliding Window

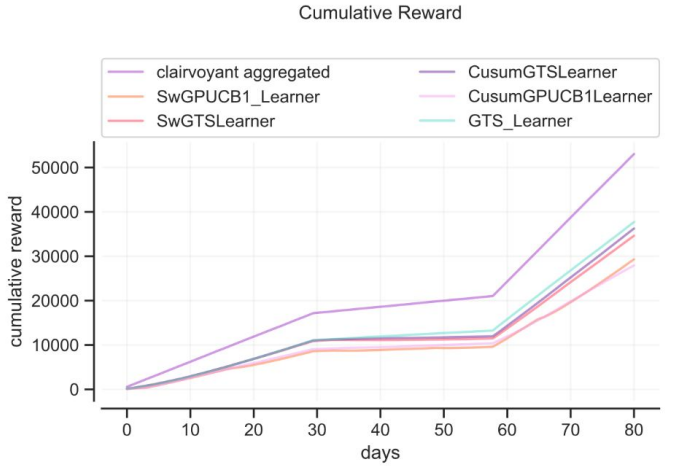
Window Size = $\lceil (\sqrt{T} + 0.1T) \rceil$

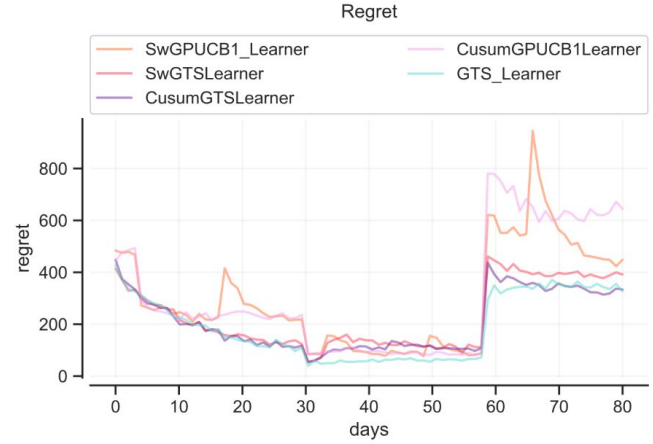
Change Detection CUSUM

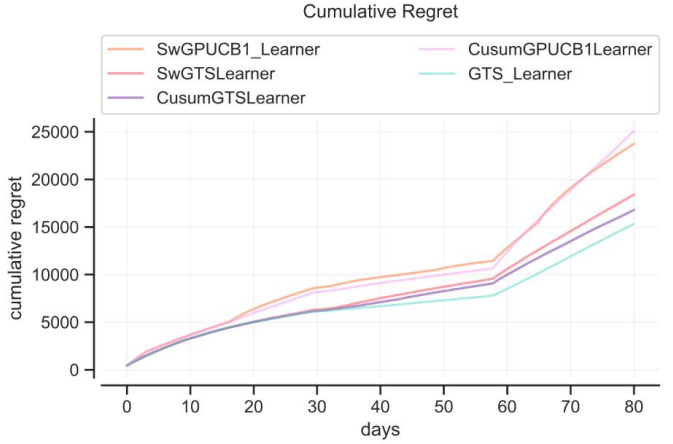
Samples for reference point	10
Epsilon	0.05
Detection threshold	200
Exploration factor	0.01

Step 6 - Graphical results

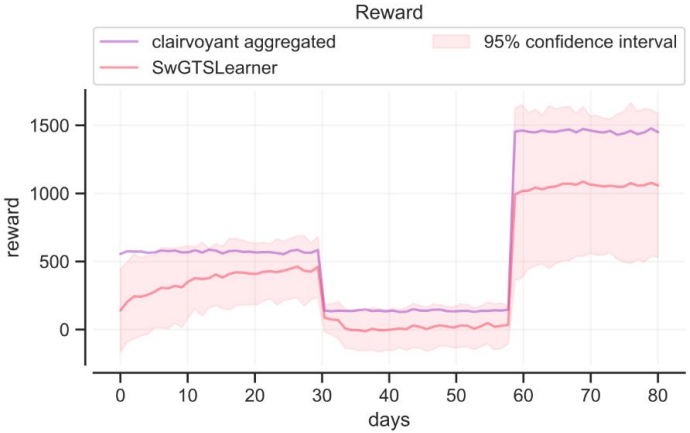


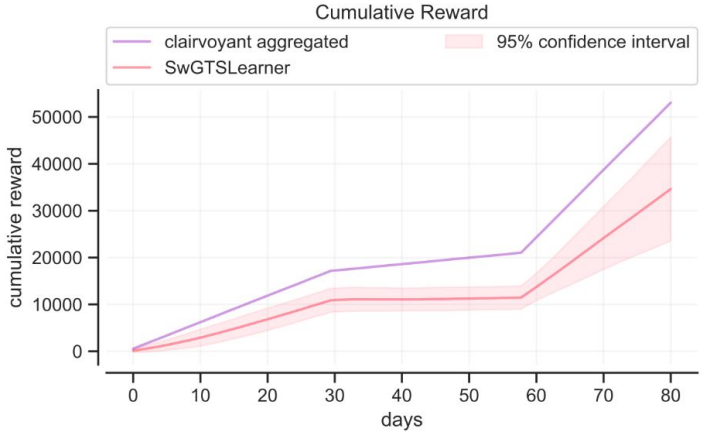


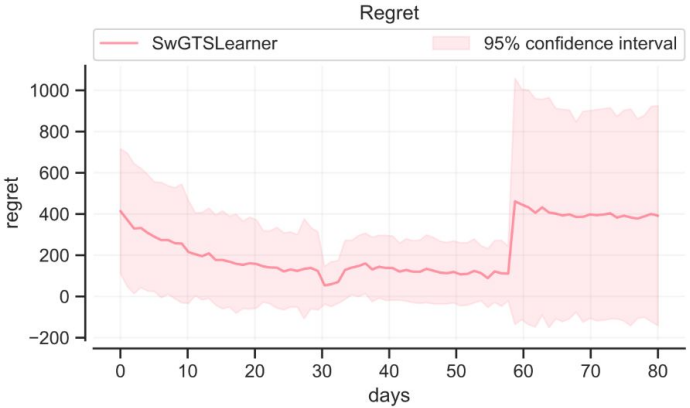


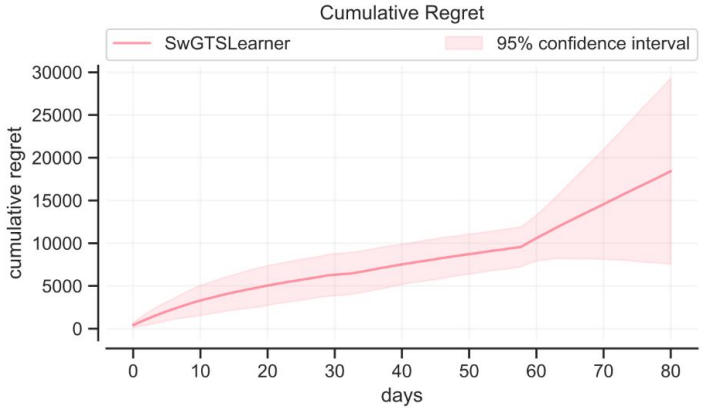


Step 6 - Confidence Intervals > SW-GTS

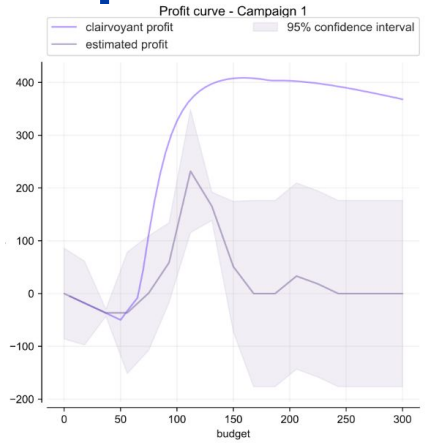


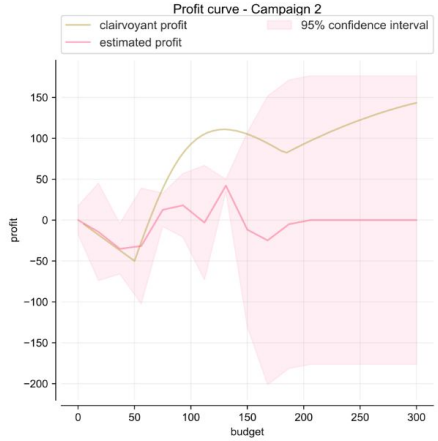


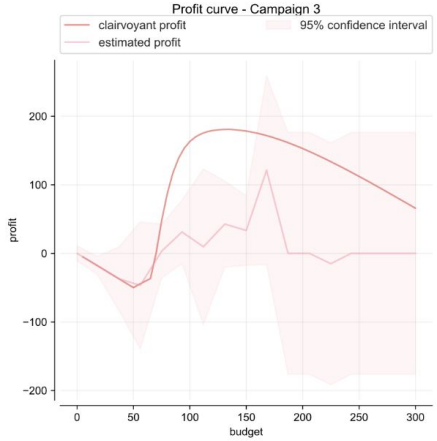


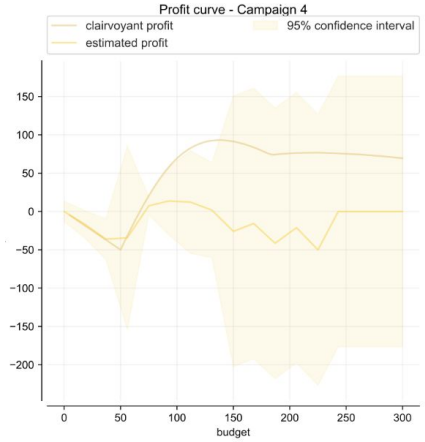


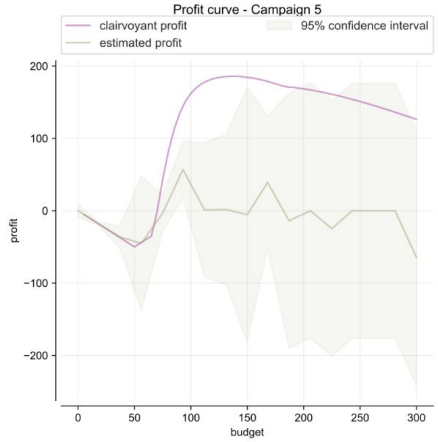
Step 6 - Estimated Profit Curve > SW-GTS











Step 7 - Context generation

The same structure of the combinatorial learner has been kept, the only difference was in the number of learners employed:

- 5 learners (previous steps learners)
- 10 learners (1 split)
- 20 learners (2 split)

The number of learners grows exponentially as $\mathbf{5^{(number\ of\ splits)}}$

In the simulation the breakpoints activating the splits can be setted

The main advantage of performing the split is that the budget can directed with more accuracy over the best users, the drawback is that convergence is slower since the problem is harder

Step 7 - Context generation

2 Split techniques developed:

- Copying the learners instances of the previous contexts
- Doubling and restarting the learning process (better)

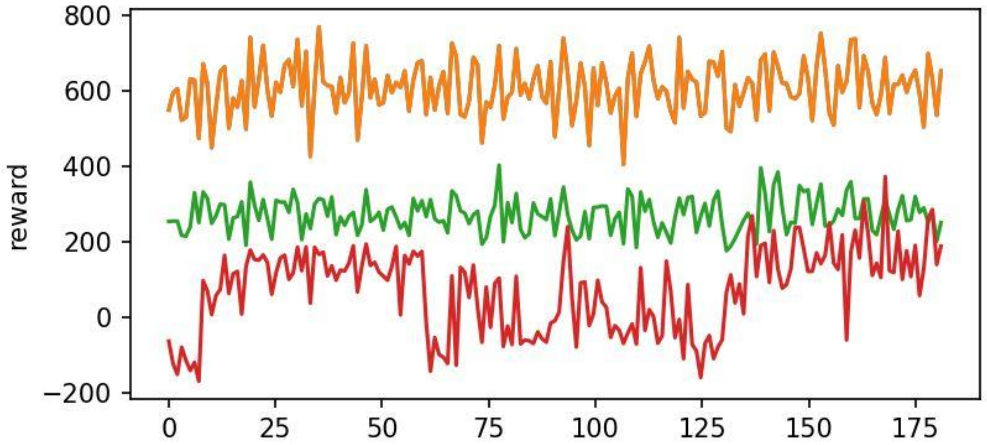
The decision of the split yes/no has been performed using the Hoeffding lower bound of the 2 cases, where x is the mean profit value

$$\bar{x} - \sqrt{-\frac{\log(\delta)}{2|Z|}}$$

where **δ** is the confidence and Z is the set of data. ***δ** = 0.8 used*

Step 7 - Context generation

The spotted pattern was that the disaggregated case was the best, the convergence slower but better on the long run, however in real scenarios we may not be able to afford such long periods of training.



Average behavior observed using GPTS with context generation length 60 days with breakpoints at 60 and 120.

The second split passed the **Hoeffding lower bound** test with 125.41 > 92.51