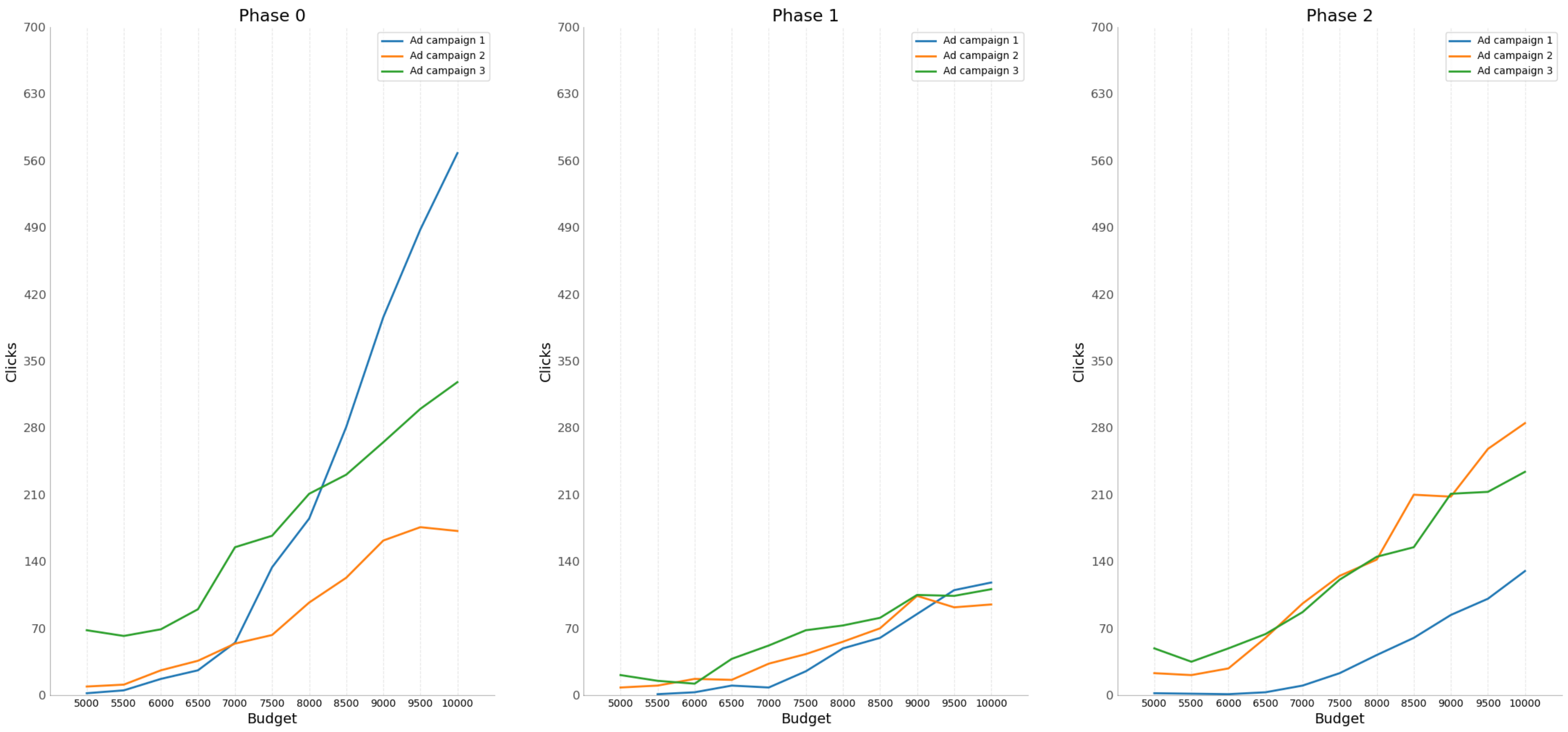
Pricing and Advertising

Gabriele Gavarini

Scenario

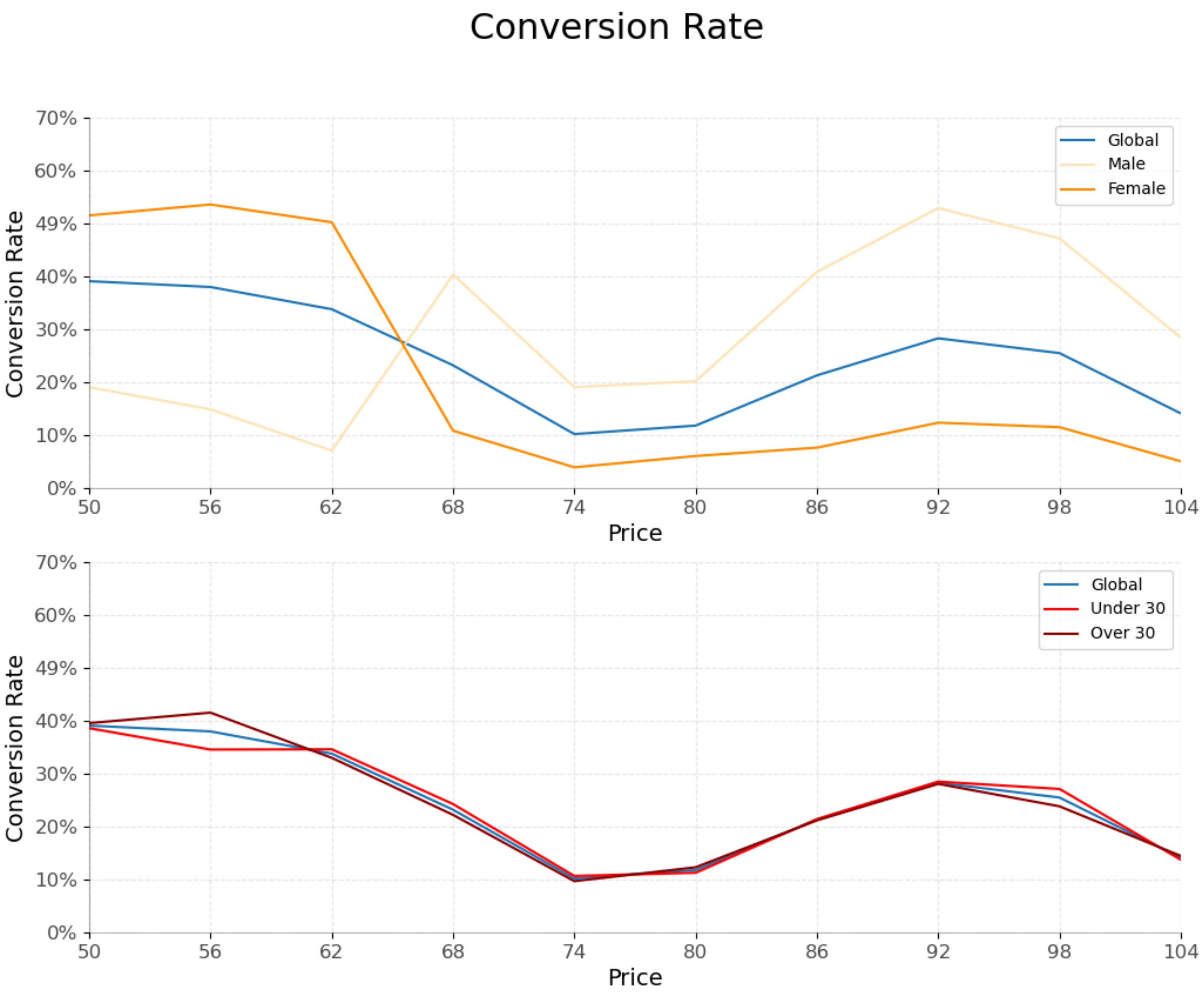
Advertising



- 3 different campaigns with 3 different phases
- Ad campaign 1 is targeted to women older than 30
- Ad campaign 1 is targeted to women younger than 30
- Ad campaign 1 is targeted to men

Scenario

Pricing



- Conversion rate not affected by age
- Conversion rate dependent on sex
- To highlight algorithm performance, we split users based on age but not sex

BUDGET ALLOCATION

Single Phase

Design

The problem can be formulated as:

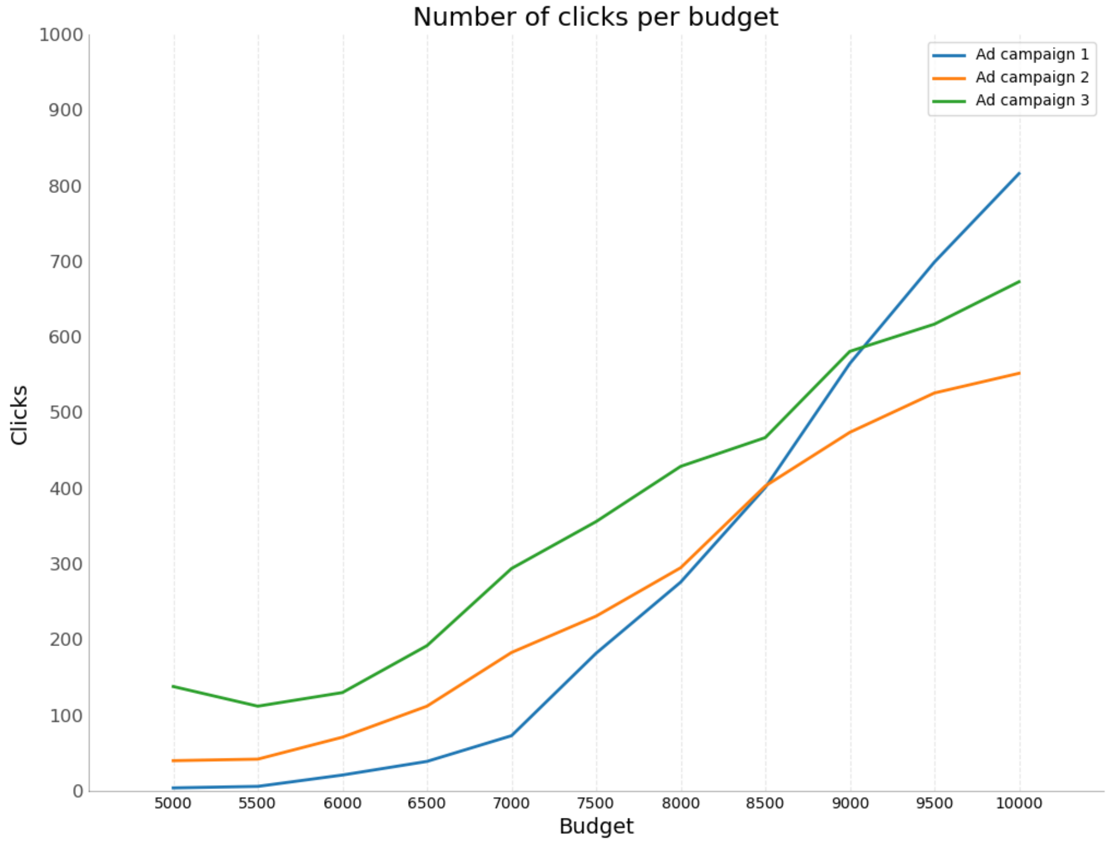
$$\begin{cases} \max\limits_{y_j} \sum\limits_{j}^{N} n_j(y_j) \\ \sum\limits_{j}^{N} y_j < Y \end{cases}$$

For every day t, we run the following loop:

1. Sample the Gaussian Process Thompson Sampler — To obtain the number of clicks n_j
2. Run the Knapsack-like Optimizer — To decide the allocated budget y_j
3. Sample the real-world function — To obtain the number of clicks for the allocated budget y_j
4. Update the Gaussian Process Thompson Sampler — To update the model of the number of clicks for every campaign j.

Budget Allocation – Single Phase

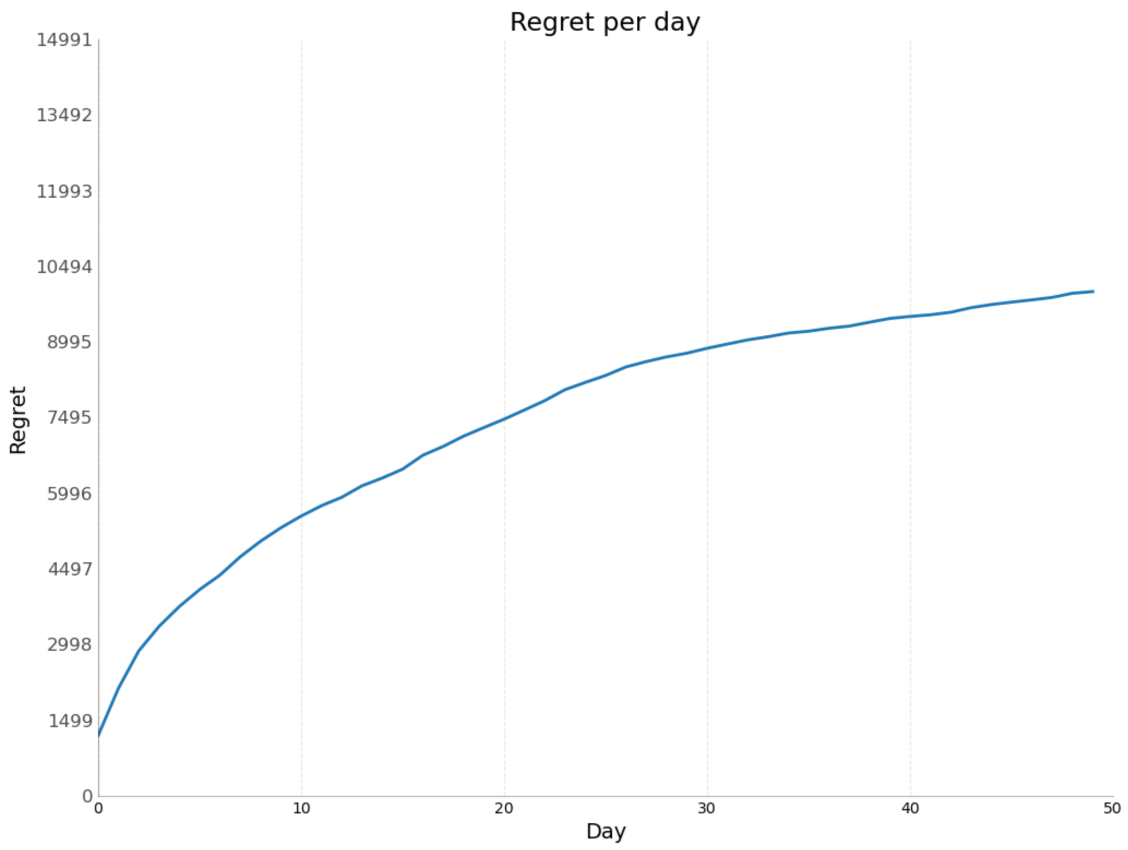
Execution



Setup

- 11 possible budgets, from 5000$ to 10000$ sampled every 500$
- Budget cap of 27000$
- Repeat execution for 50 days

Execution



Setup

- 11 possible budgets, from 5000$ to 10000$ sampled every 500$
- Budget cap of 27000$
- Repeat execution for 50 days

Results

- Rapid increase in first few days
- Stabilization after few days
- Solution found is coherent with ideal solution

BUDGET ALLOCATION

3 Phases

Design

The problem can be formulated as:

$$\begin{cases} \max\limits_{y_j} \sum\limits_{j}^{N} n_j(\boldsymbol{t}, y_j) \\ \sum\limits_{j}^{N} y_j < Y \end{cases}$$

Differently from the one phase model, the optimal solution depends on the time ***t***.

The window size is calculated as:

$$size = c * \sqrt{T}$$

Where *c* is a coefficient and *T* is the time horizon.

For every day t, we run the following loop:

Add a Sliding Window to use only the most recent samples

Sample the **Sliding-window** Gaussian Process Thompson Sampler

To obtain the number of clicks n_j

Run the Knapsack-like Optimizer

To decide the allocated budget y_j

Sample the real-world function

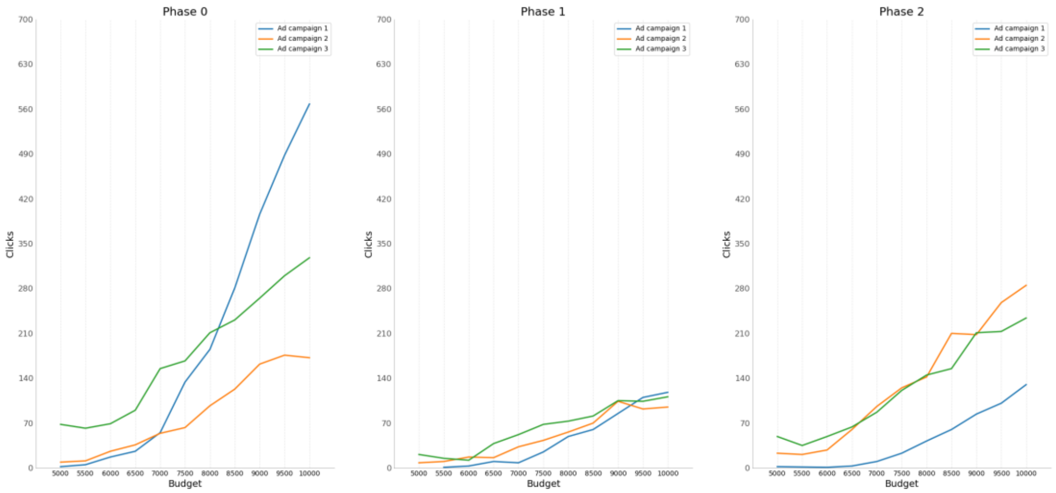
To obtain the number of clicks for the allocated budget y_j

Add a Sliding Window to use only the most recent samples

Update the **Sliding-window** Gaussian Process Thompson Sampler

To update the model of the number of clicks for every campaign j.

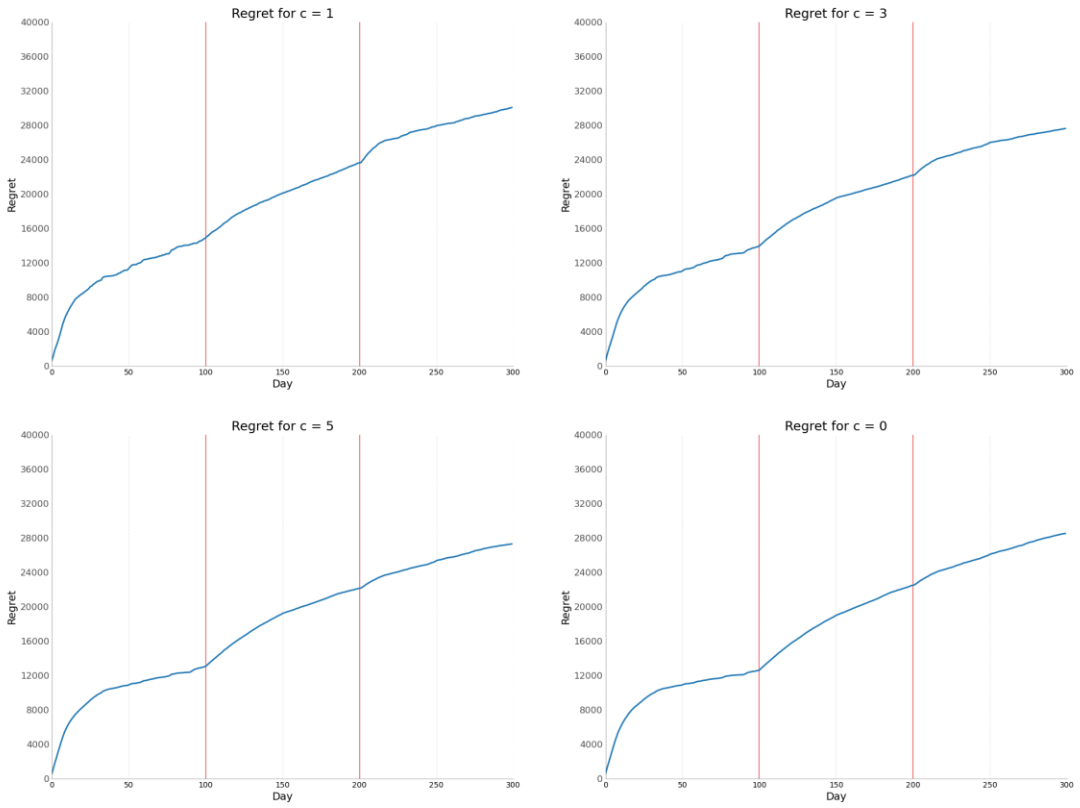
Execution



Setup

- 11 possible budgets, from 5000$ to 10000$ sampled every 500$
- Budget cap of 27000$
- The phase changes every 100 days
- Executed for 300 days
- Repeated for different values of *c*

Execution



Setup

- 11 possible budgets, from 5000$ to 10000$ sampled every 500$
- Budget cap of 27000$
- The phase changes every 100 days
- Executed for 300 days
- Repeated for different values of *c*

Results

- No window ($c = 0$) best in first phase
- $\sim$17 days window best in the end
- No windows is better in static context, windowed better in dynamic ones
- Performances vary with window sizing

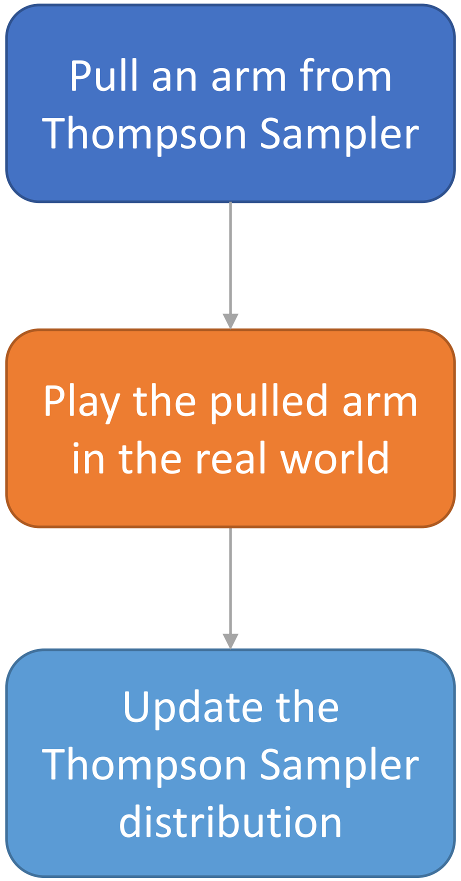
PRICING

Pricing

Design

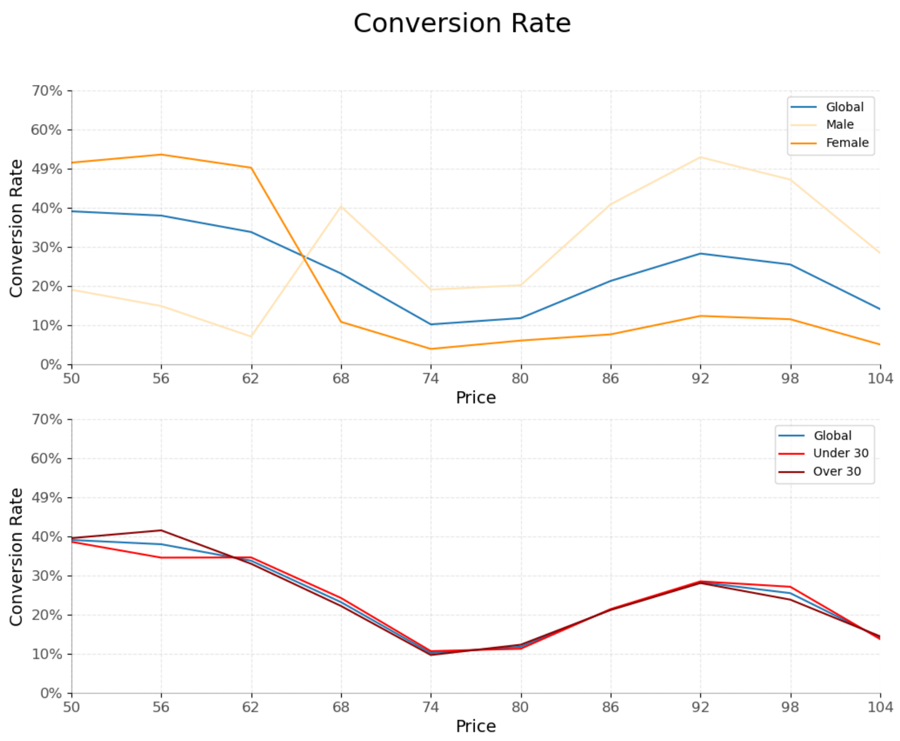
Our objective is to find which price will result in the best conversion rate possible.

For every day t, we run the following loop:



Pricing

Execution

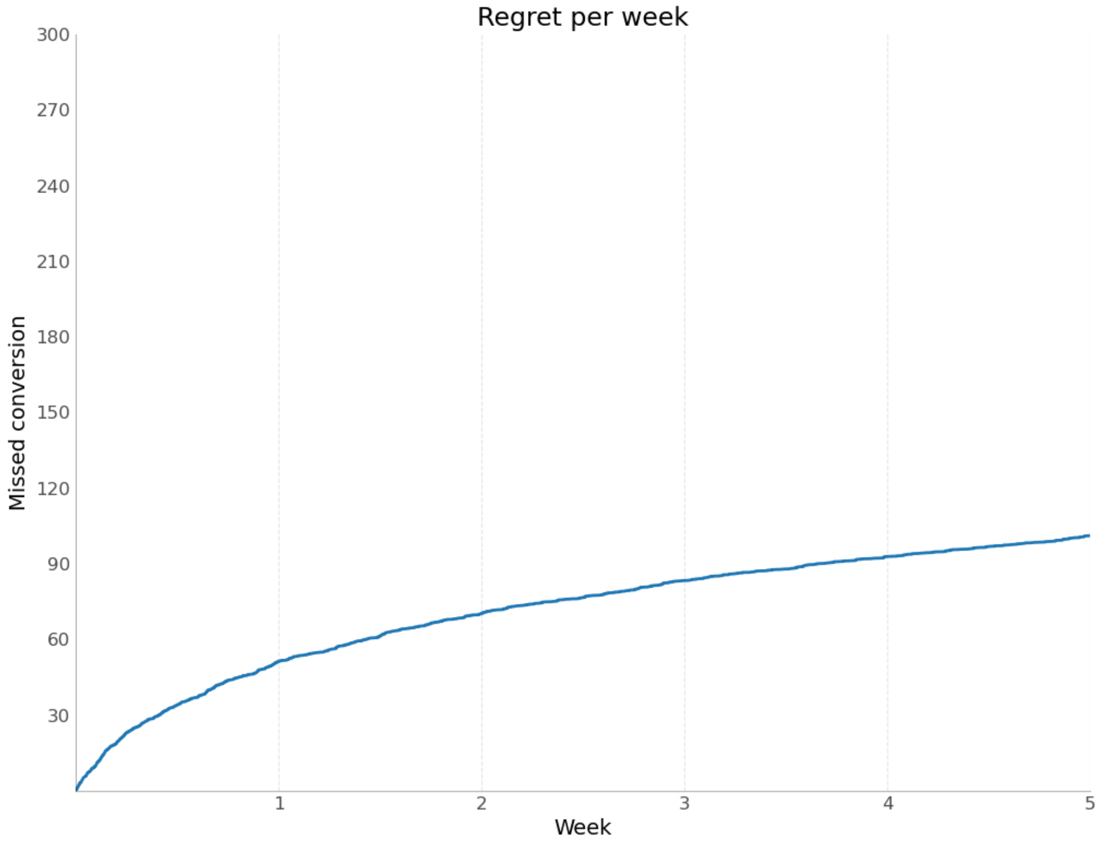


Setup

- 10 possible prices
- Changing price every 10 minutes
- Simulating for 5 weeks

Pricing

Execution



Setup

- 10 possible prices
- Changing price every 10 minutes
- Simulating for 5 weeks

Results

- Rapid increase in first days
- Stabilizes later on
- Small increases due to exploration

PRICING

Context Learning

Design

Our objective is to find which price will result in the best conversion rate possible.

We also want to find a partitioning of users in context. We use a decision tree, splitting when:

$$\mu < p * \mu^*_{a_1} + (1 - p) * \mu^*_{a_0}$$

Where:

- μ is the information before the split
- p is the probability of attribute a being equal to 1
- $\mu^*_{a_1}$ is the information obtained by playing arm a^* when attribute $a = 1$
- $\mu^*_{a_0}$ is the information obtained by playing arm a^* when attribute $a = 0$

For every day t, we run the following loop:

Pull an arm from Thompson Sampler

↓

Play the pulled arm in the real world

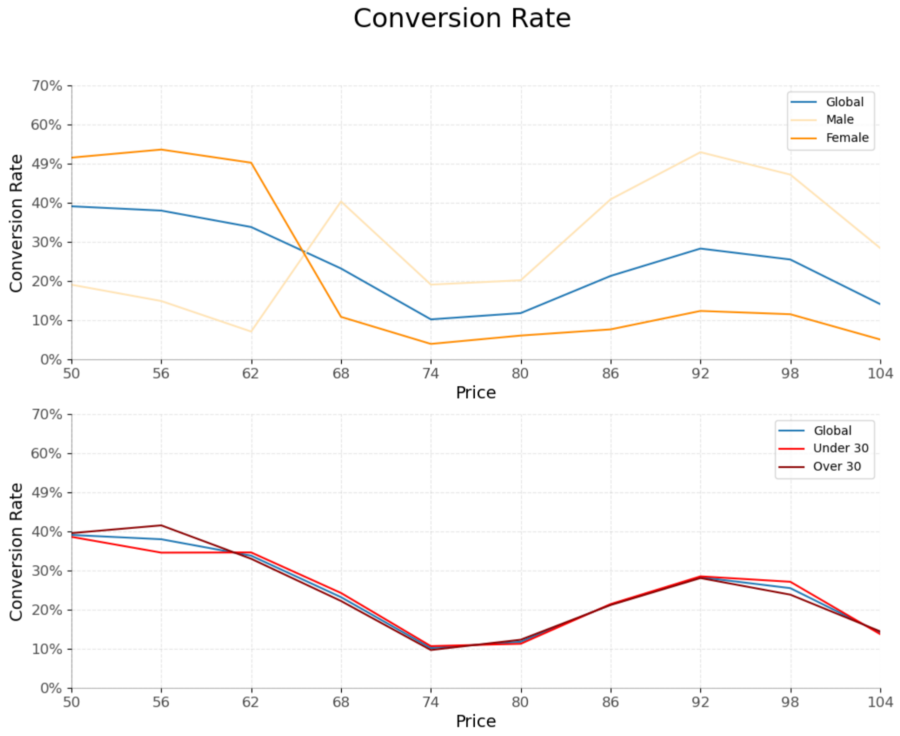
↓

Update the Thompson Sampler distribution

At the end of every week, we generate a new context by:

Run the clustering algorithm

Execution

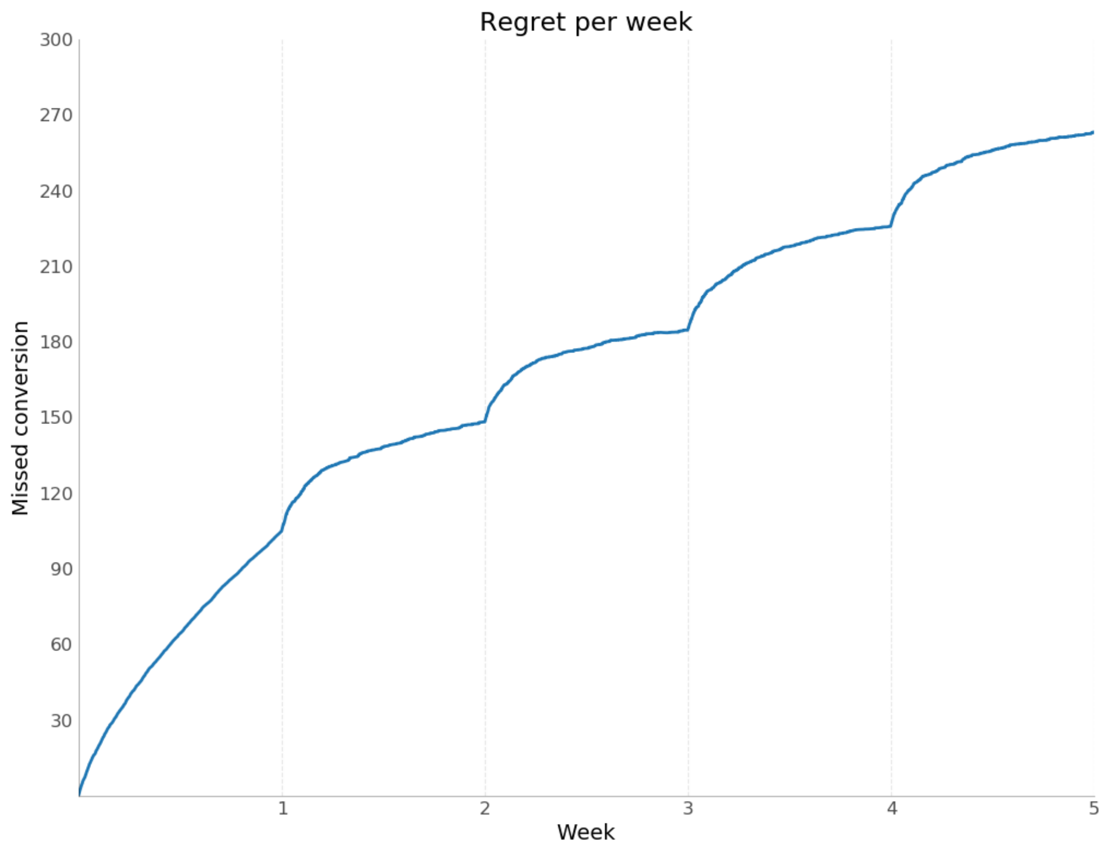


Setup

- 10 possible prices
- Changing price every 10 minutes
- Simulating for 5 weeks
- New context generation at the end of every week
- Initial context based on age and not sex (worst case)

Pricing – Context Learning

Execution



Setup

- 10 possible prices
- Changing price every 10 minutes
- Simulating for 5 weeks
- New context generation at the end of every week
- Initial context based on age and not sex (worst case)

Results

- Worst performance in first week
- Performance get better as soon as new context are generated
- New contexts are based on sex (but sometimes also on age)

SIMULTANUES OPTIMIZATION

Design

The problem can be formulated as:[1]

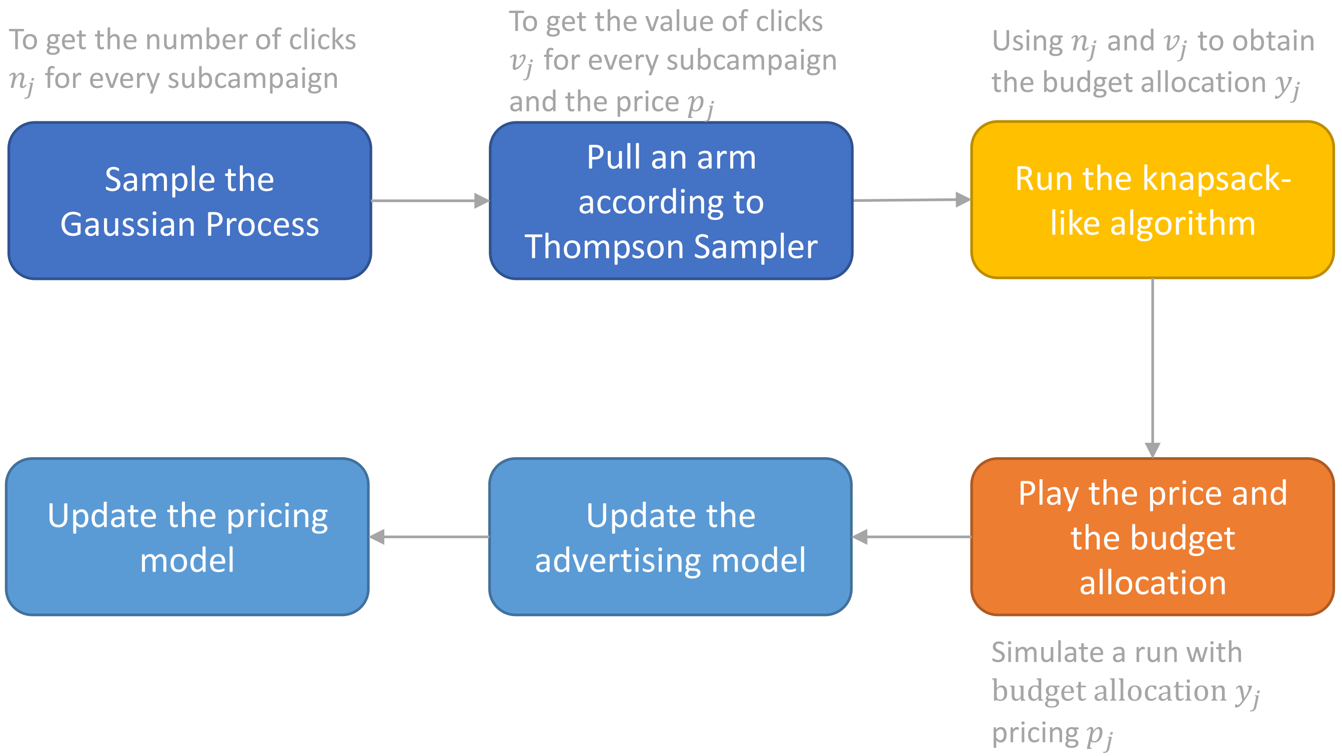
$$\begin{cases} \max\limits_{y_j} \sum\limits_{j}^{N} v_j(p_j)\, n_j(y_j) \\ \sum\limits_{j}^{N} y_j < Y \end{cases}$$

Where we add a value to the clicks.

We solve it using:

- Thompson Sampler for the pricing
- Gaussian Process Thompson Sampler for the budget allocation
- A knapsack-like algorithm to solve the system

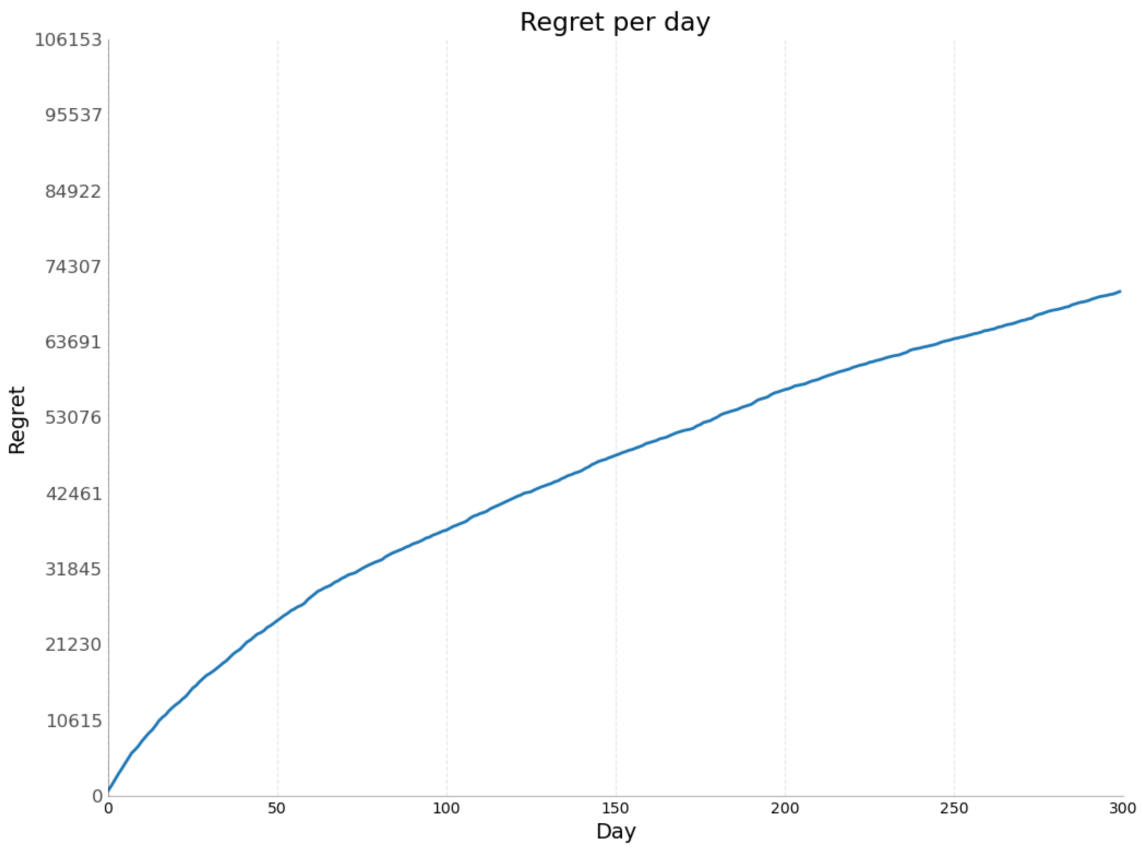
For every day t, we run the following loop:



[1]: *under the assumption that all and only users of context j will buy a product from advertising campaign j.*

Simultaneous Optimization

Execution



Setup

- 10 possible prices
- 11 possible budgets
- Play for 300 days
- One campaign for women under 30, one for women over 30, one for male

Results

- Steep initial increase, slow downs after few days
- Ideal budget allocation found reliably after ~100 days
- Pricing slower convergence, hence, the continuous increase

SIMULTANUES OPTIMIZATION

with a unique price

Design

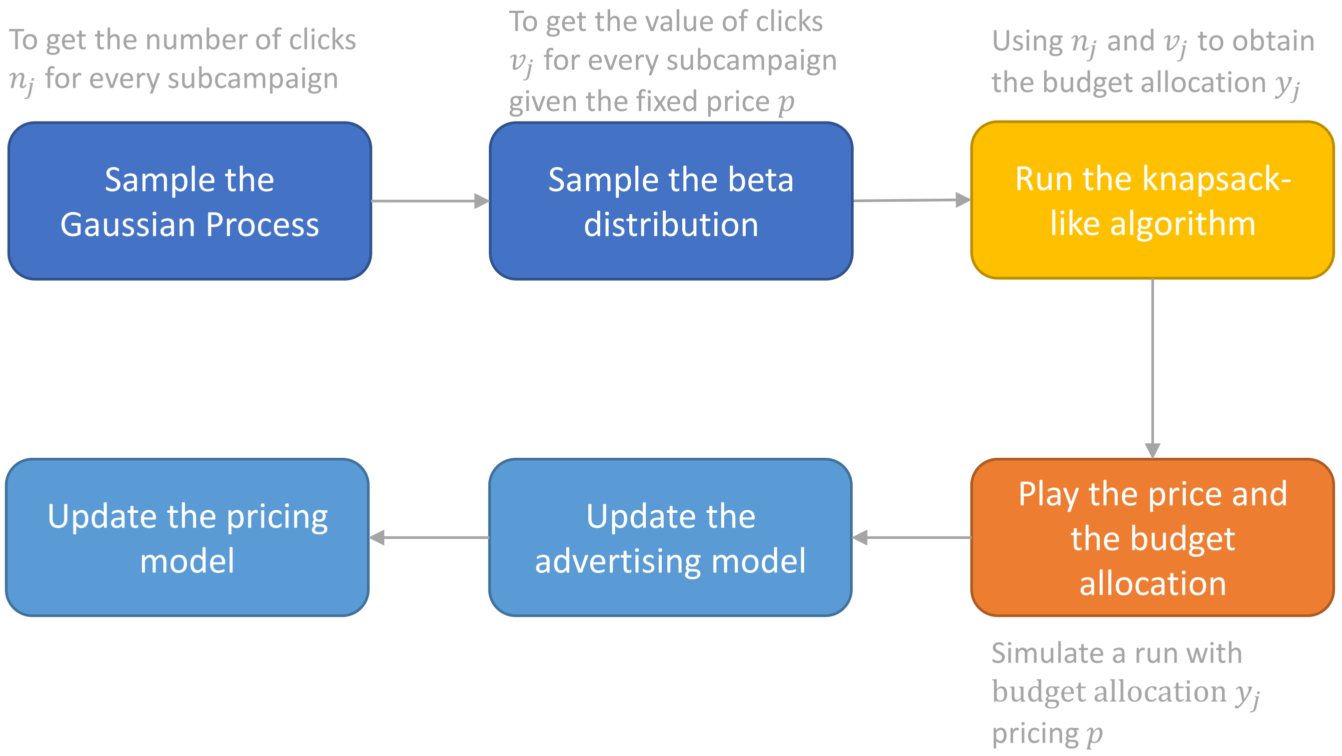
The problem can be formulated as:[1]

$$\begin{cases} \max\limits_{y_j} \sum_{j}^{N} \boldsymbol{v_j(p)} n_j(y_j) \\ \sum_{j}^{N} y_j < Y \end{cases}$$

It's the same problem as before, but the price is **unique** for all the contexts.

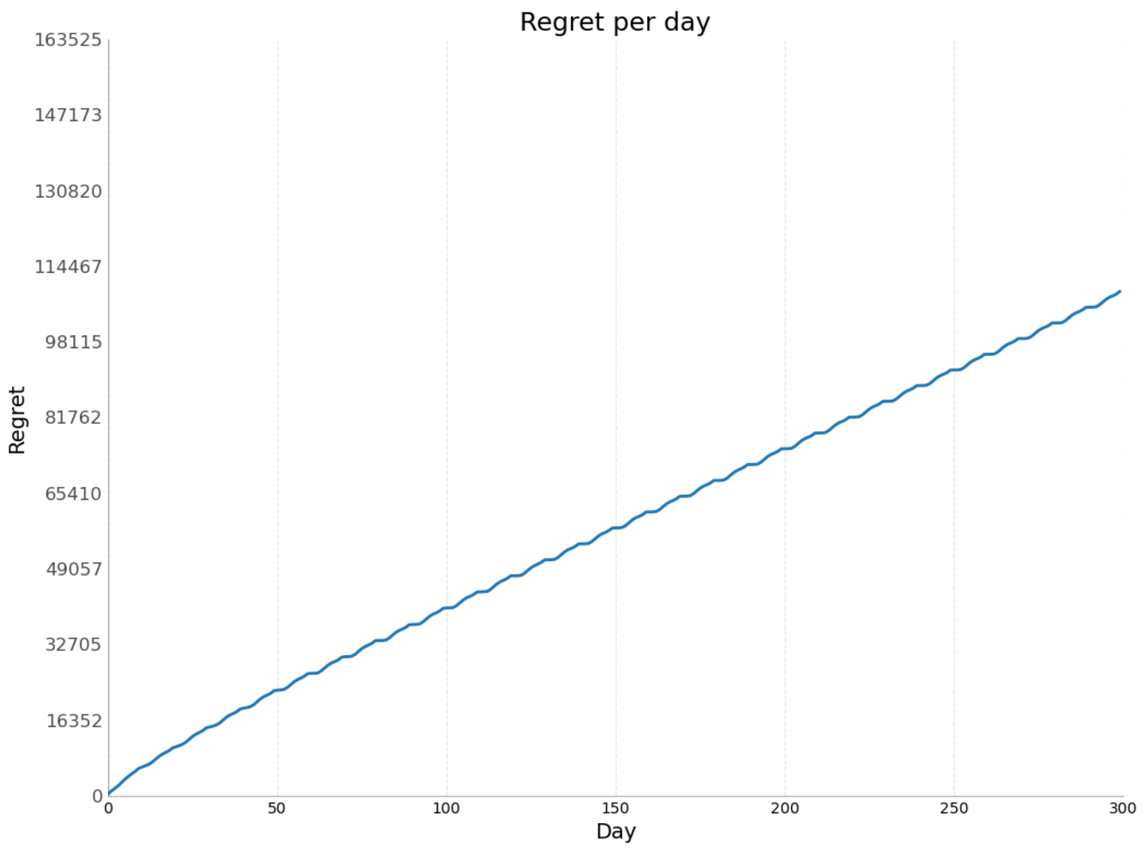
We solve this by cycling all the possible prices, playing a different and unique price every day.

For every day t, we run the following loop:



[1]: *under the assumption that all and only users of context j will buy a product from advertising campaign j.*

Execution



Setup

- 10 possible prices
- 11 possible budgets
- Play for 300 days
- One campaign for women under 30, one for women over 30, one for male
- Play a new price every 10 days

Results

- The regret will always increase
- Convergence to the ideal budget allocation
- Saw-like behavior: regret increases when price is not ideal
- No implicit learning of the pricing