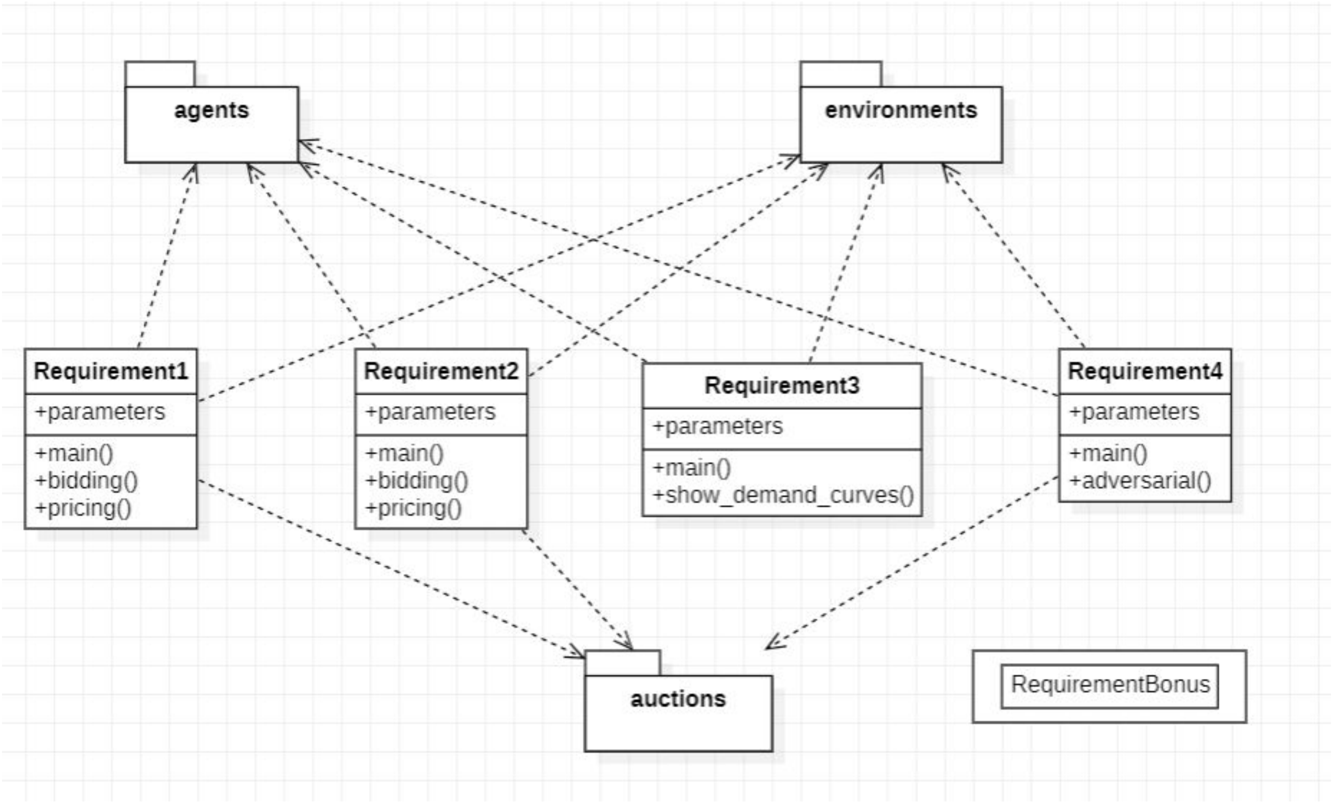
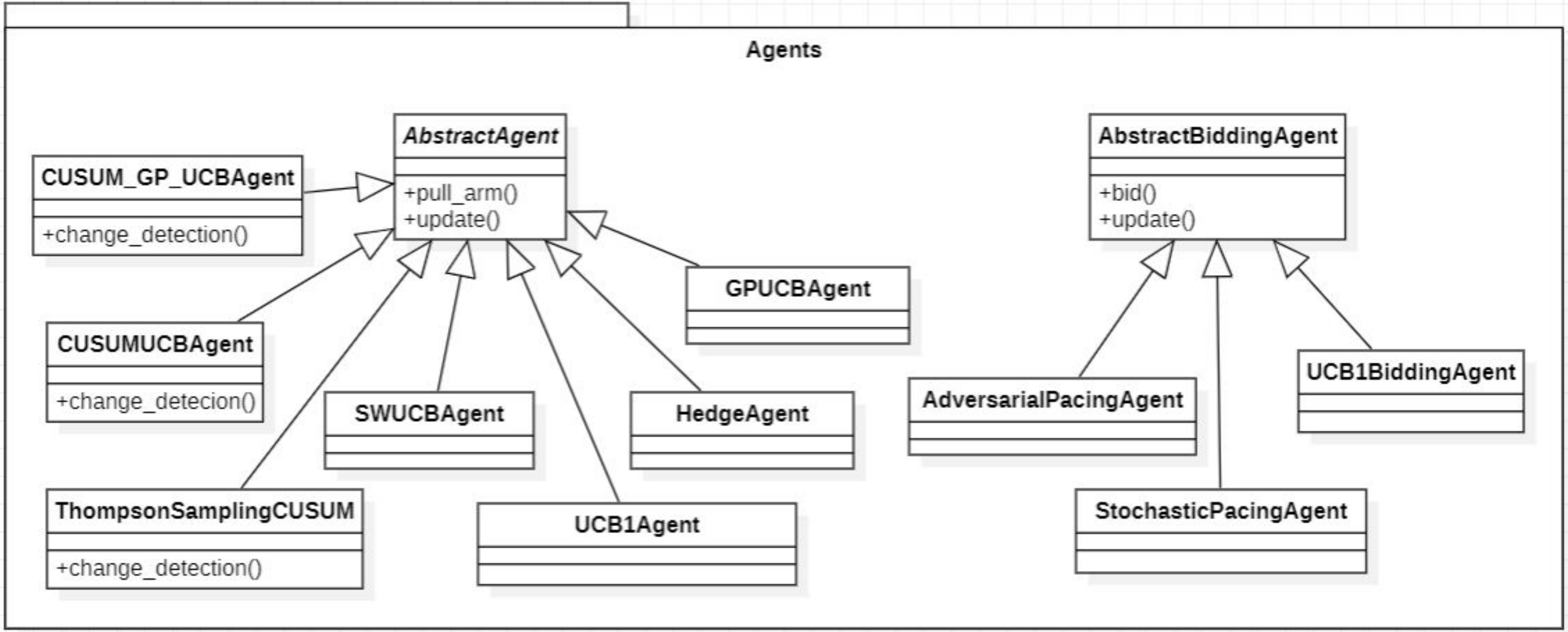
OLA24

Barda Luca
Grillo Niccolò
Franzè Lorenzo

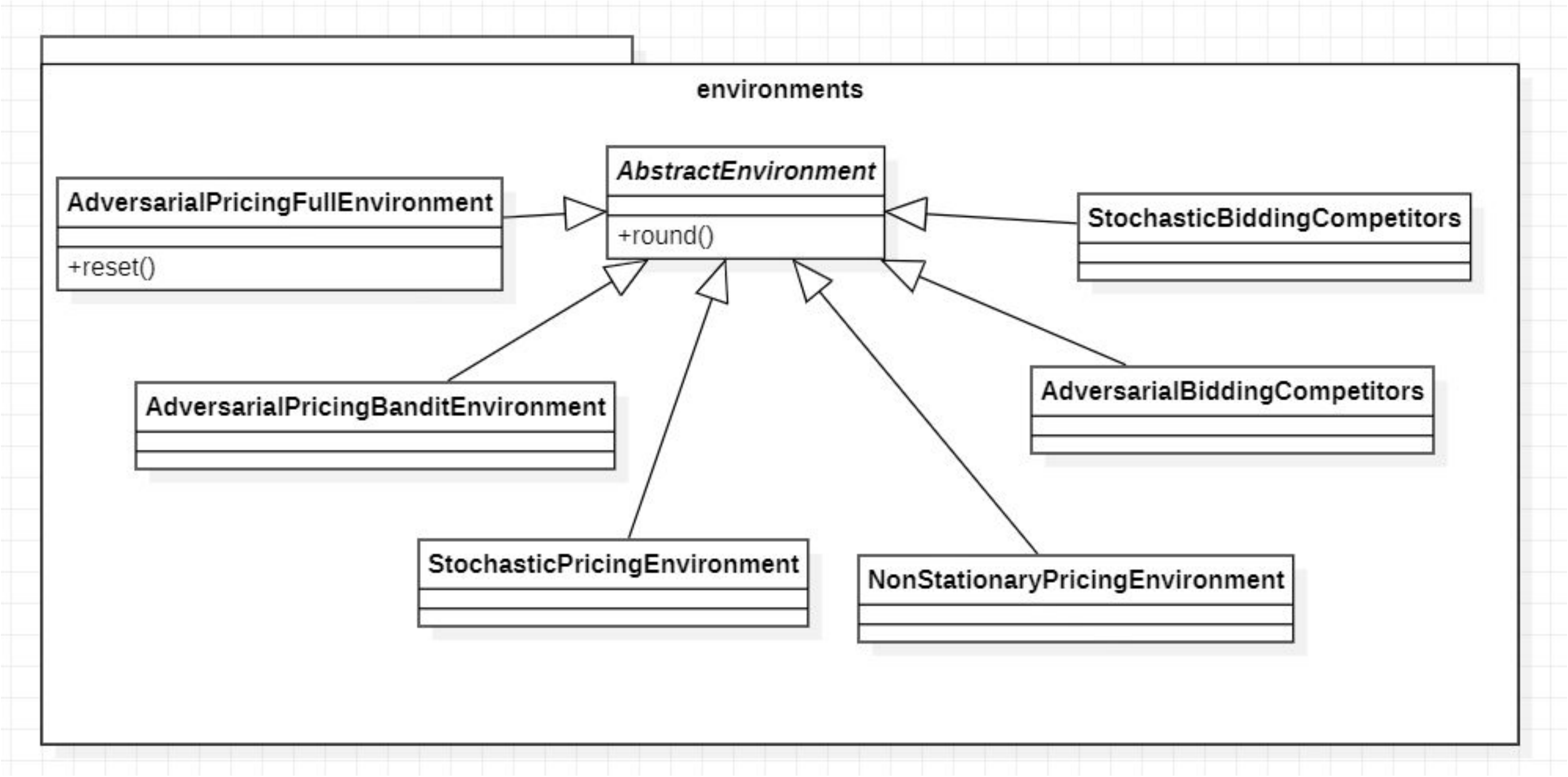
Project structure



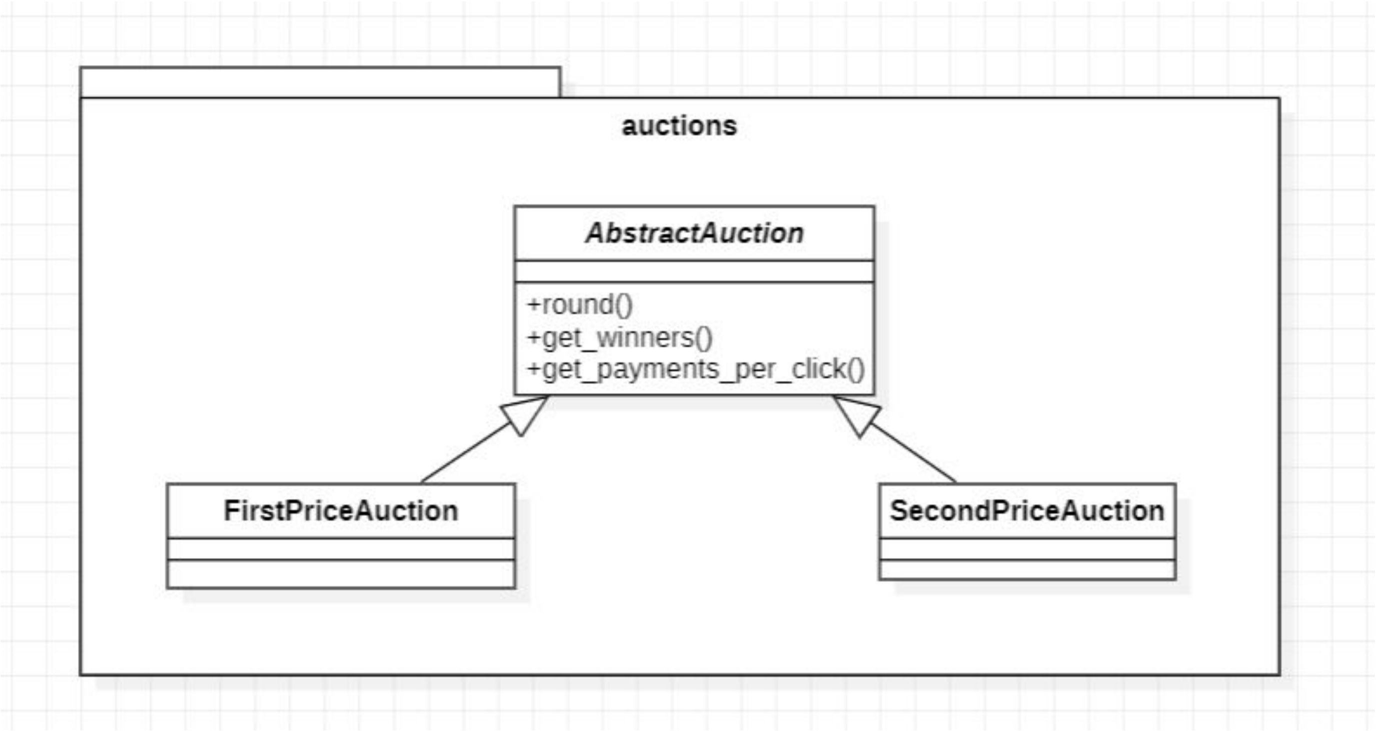
Agents package



Environments package



Auctions package



Requirement 1

```
--run_type ['main', 'bidding', 'pricing']
--bidder_type ['UCB', 'pacing']
--auctions_per_day
--num_days
--n_iters
--num_competitors
--budget
--valuation
--ctr
--theta
--item_cost
--num_buyers
```

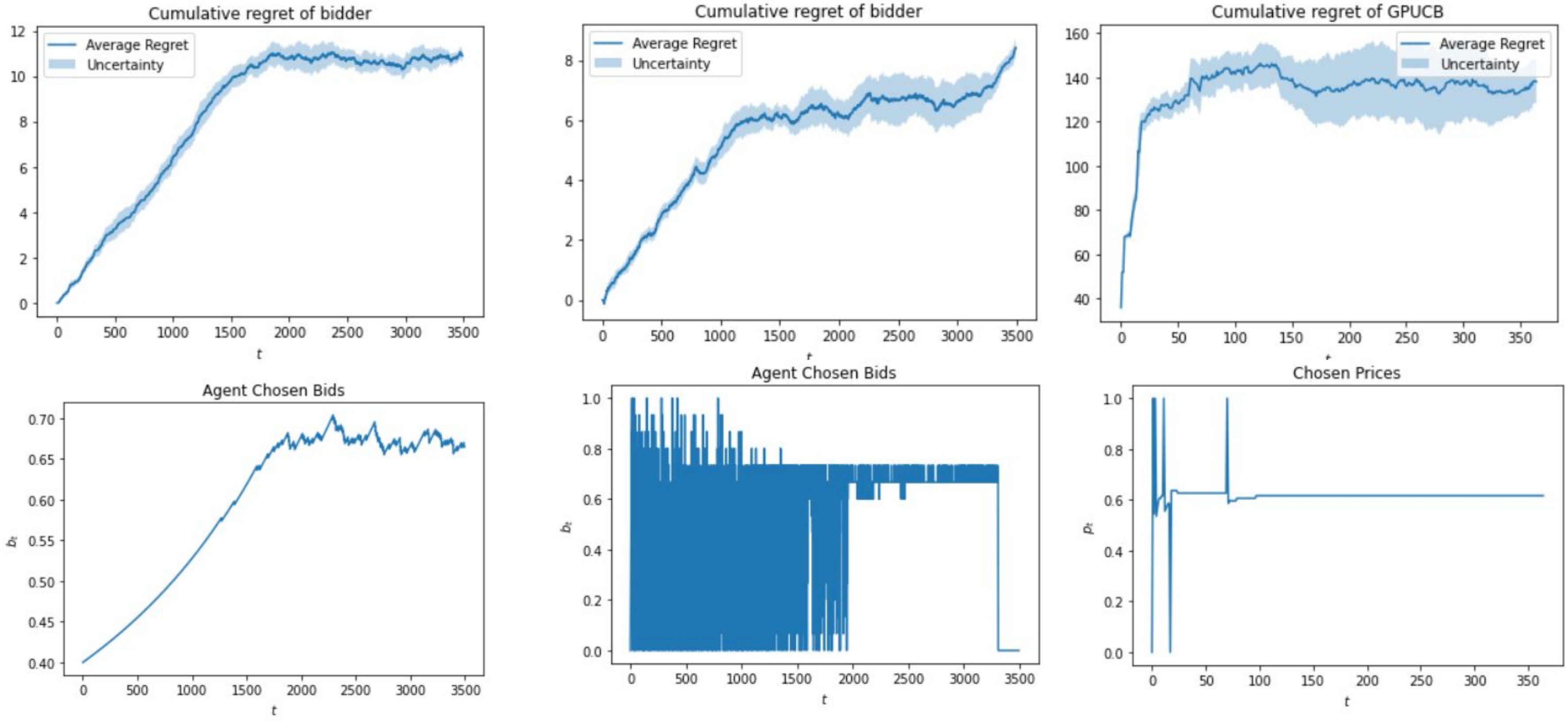
What's left?
- demand curve
- opponent bids
- opponent ctrs

agents:
- UCB1BiddingAgent (bidding)
- StochasticPacingAgent (bidding)
- GPUCBAgent (pricing)

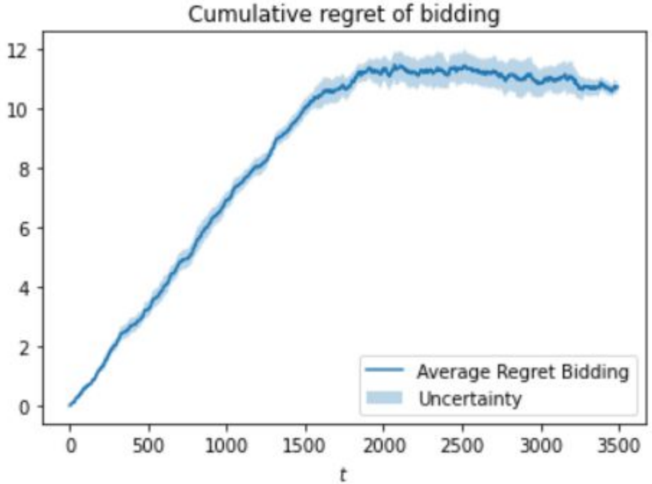
environments:
- StochasticPricingEnvironment (pricing)
- StochasticBiddingCompetitors (bidding)

auctions:
- SecondPriceAuction

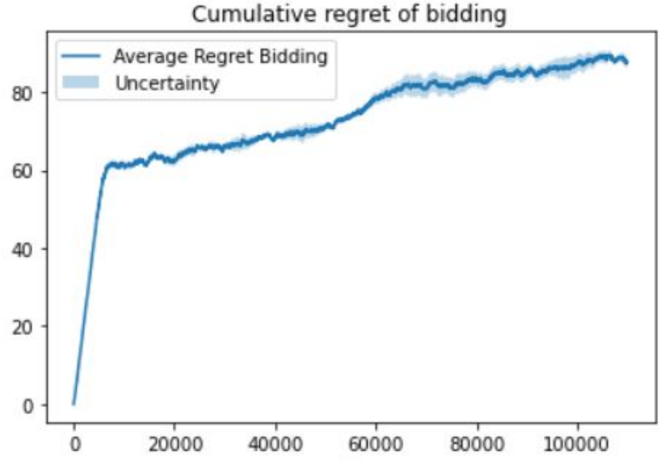
Results bidding and pricing

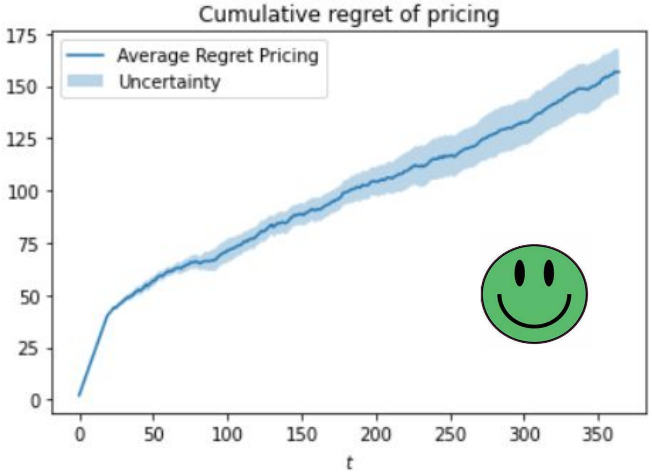


Results main



increase
auctions_per_day





Requirement 2 - Adversarial Full Feedback

```
--run_type ['main', 'bidding', 'pricing']
--auctions_per_day
--num_days
--n_iters
--num_competitors
--budget
--valuation
--ctr
--theta
--item_cost
--num_buyers
```

agents:
- AdversarialPacingAgent (bidding)
- HedgeAgent (pricing)

environments:
- AdversarialBiddingCompetitors (bidding)
- AdversarialPricingEnvironment (pricing)

auctions:
- FirstPriceAuction

Requirement 2 - clairvoyant

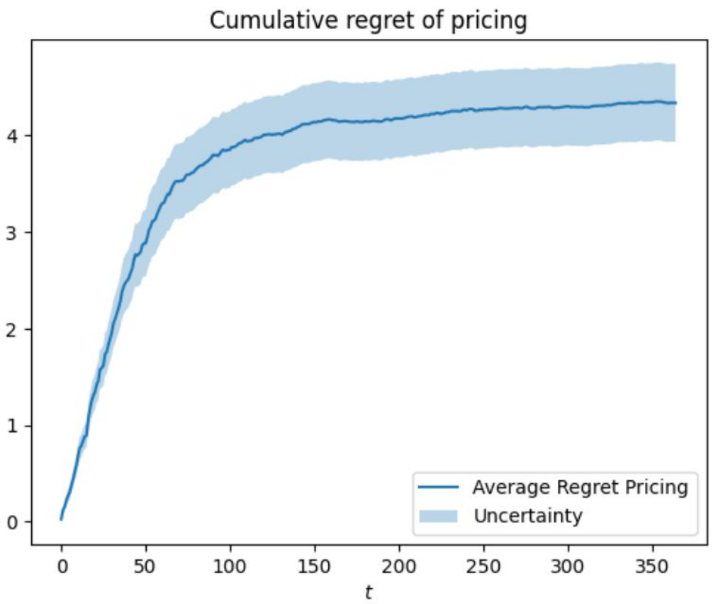
- We used the **adversarial clairvoyant** for both the pricing and bidding:
- For the bidding part is built with the following strategy

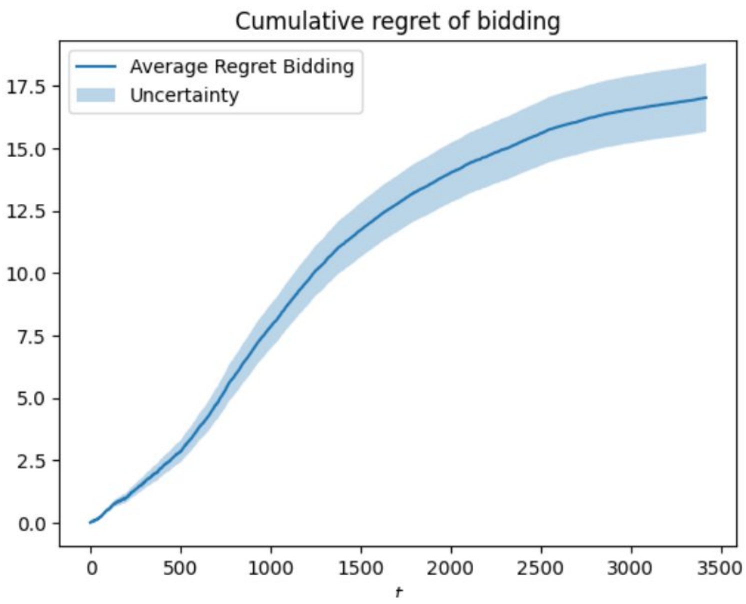
Algorithm 1 Clairvoyant Utility in Adversarial Auctions

Ensure: Maximize utility under budget constraint

```
max_utility ← −∞
for each bid in discr_bids do
    c ← 0                                        ▷ Total money spent
    utility ← 0
    for each auction t = 1 to n_auctions do
        if c < budget then
            all_bids[idx_agent, t] ← bid
            winner, _ ← auction_agent.get_winners(all_bids[:, t])
            if winner == idx_agent then
                utility ← utility + (my_valuation − bid)
                c ← c + bid
            end if
        else
            break
        end if
    end for
    if utility > max_utility then
        max_utility ← utility
    end if
end for
return max_utility
```

Requirement 2 - Results





**(competing bids are generated with rand. unif while the demand, for pricing the conv. prob is parametrized through a theta param randomized randomly.)*

```
%run req2.py --num_days 365 --auctions_per_day 10 --n_iters 100
--num_competitors 10 --my_valuation 0.8 -budget 100 --run_type
'main'
```

**(The competitors clickthrough-rates are randomized with a uniform, changing seed at each iteration.)*

Requirement 3

Non-stationary demand curves

```
num_buyers = 100
T_pricing = 50000
intervals = 5
T_interval = 10000
cost = 10
max_price = 40
K = T_interval**(0.33) = 20
```

Agents:

- **UCB1 agent**
- **Sliding Window UCB agent**
- **GP-UCB CUSUM agent**
- **CUSUM UCB agent**
- **Thompson Sampling CUSUM agent**

D_1:

$$D_1(\text{price}) = \max\left(0, 1 - \frac{\text{price}}{30}\right)$$

D_2:

$$D_2(\text{price}) = \max\left(0, 1 - \frac{\text{price}}{60}\right)$$

D_3:

$$D_3(\text{price}) = \max\left(0, \exp\left(-\frac{(\text{price} - 10)^2}{25}\right)\right)$$

D_4:

$$D_4(\text{price}) = \max\left(0, \frac{1}{2\sqrt{0.05 \cdot \text{price} - 0.3}} - 0.5\right)$$

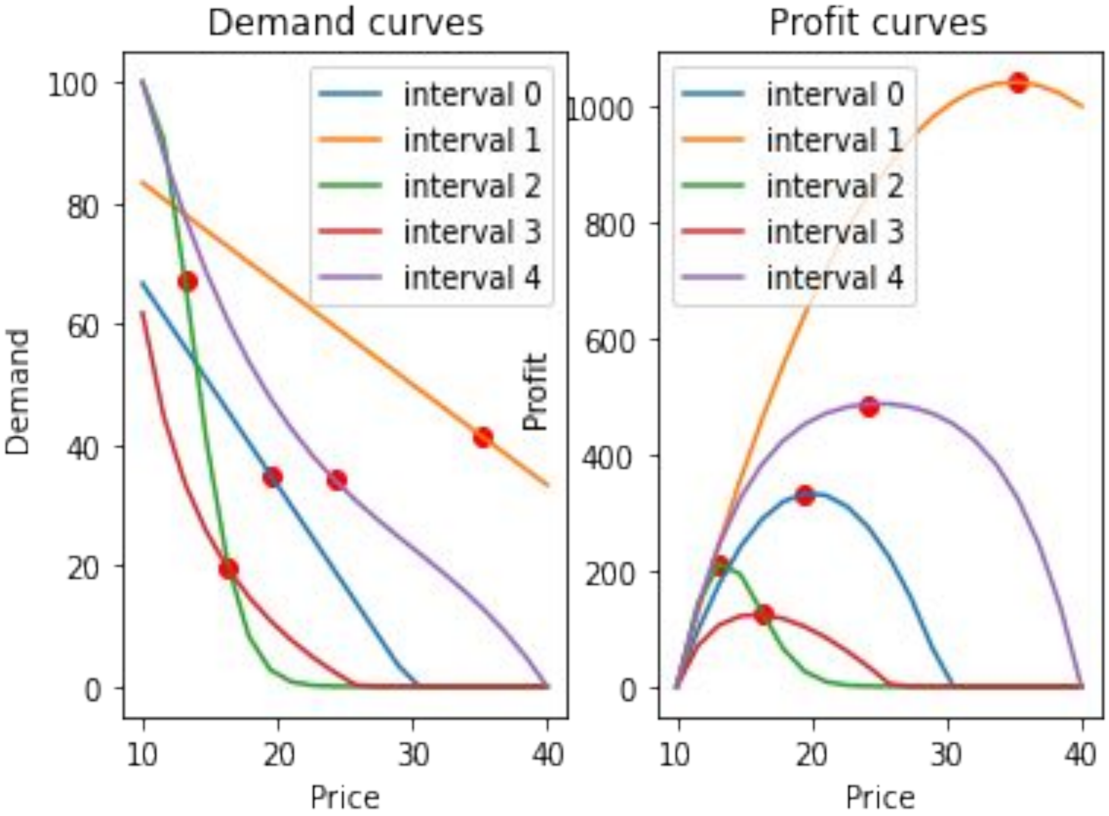
D_5:

$$D_5(\text{price}) = \max\left(0, 1 - \left(2.4 \cdot \frac{\text{price} - 10}{30} - 2.8 \cdot \left(\frac{\text{price} - 10}{30}\right)^2 + 1.4 \cdot \left(\frac{\text{price} - 10}{30}\right)^3\right)\right)$$

Environment:

`NonStationaryPricingEnvironment`

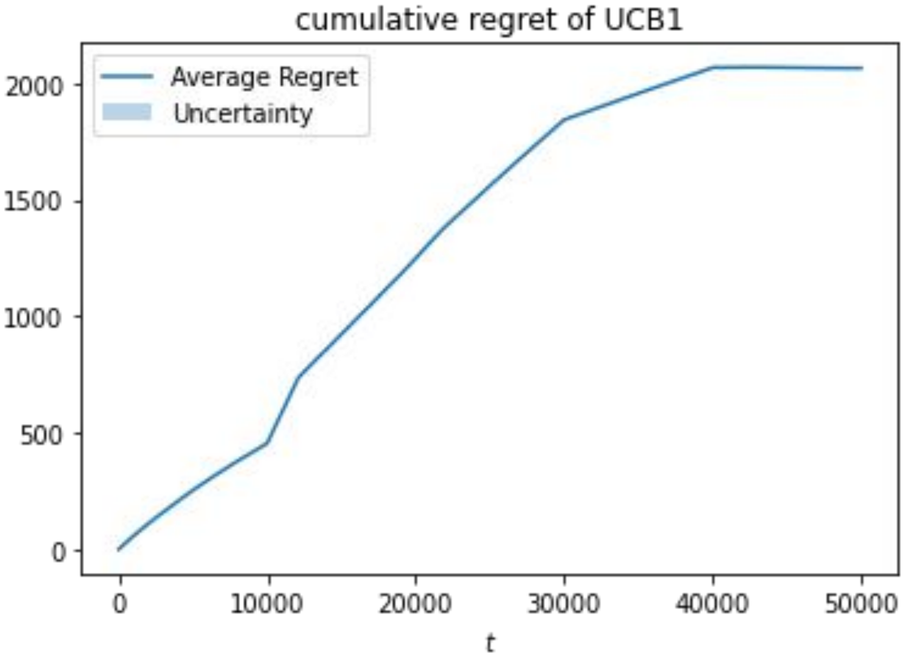
- **Noise**: at each round Binomial distribution to get the numbers of buyers out of all the buyers, according to probability given from the Demand curve
- **Non-stationary:** 4 change points in which the demand curve changes

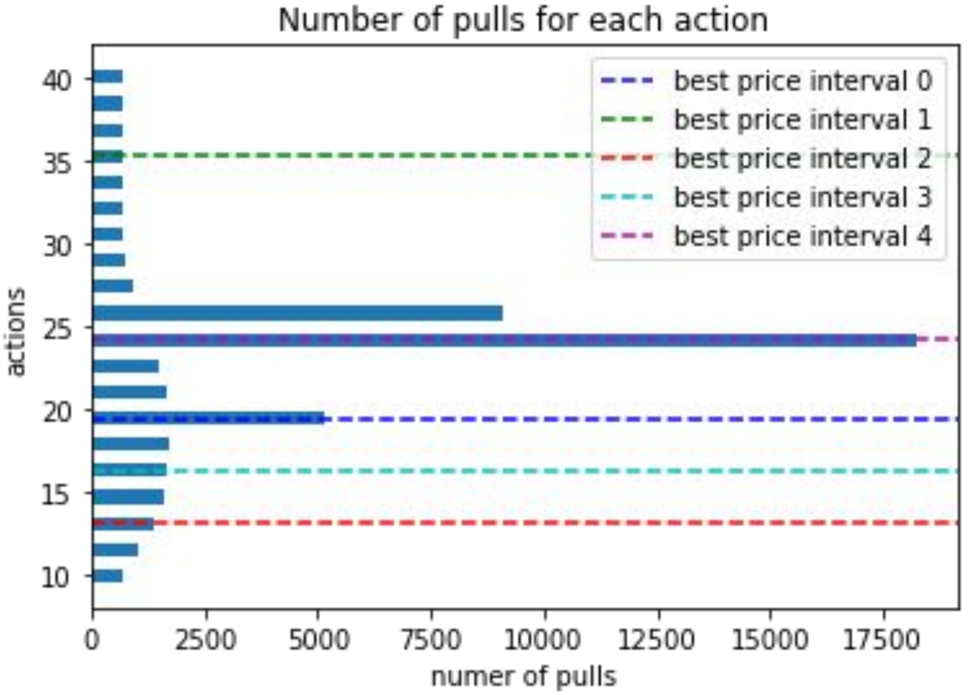


Clairvoyant:

best prices for each interval: [19.473684210526315, 35.26315789473684, 13.157894736842106, 16.315789473684212, 24.210526315789473]
best profit for each interval: [332.4099723 1041.55124654 211.91469302 123.91553435 486.87471704]
maximum profit: 3000.0

UCB 1 Agent

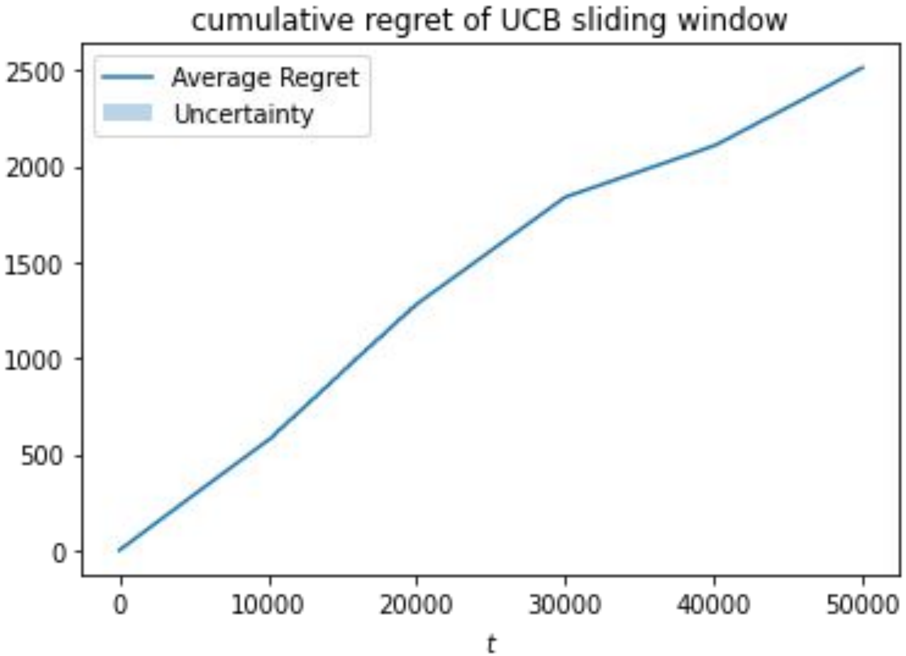

cumulative regret of UCB1
Average Regret
Uncertainty
0
500
1000
1500
2000
0
10000
20000
30000
40000
50000
t


Number of pulls for each action
best price interval 0
best price interval 1
best price interval 2
best price interval 3
best price interval 4
actions
10
15
20
25
30
35
40
0
2500
5000
7500
10000
12500
15000
17500
numer of pulls

Sliding Window UCB Agent

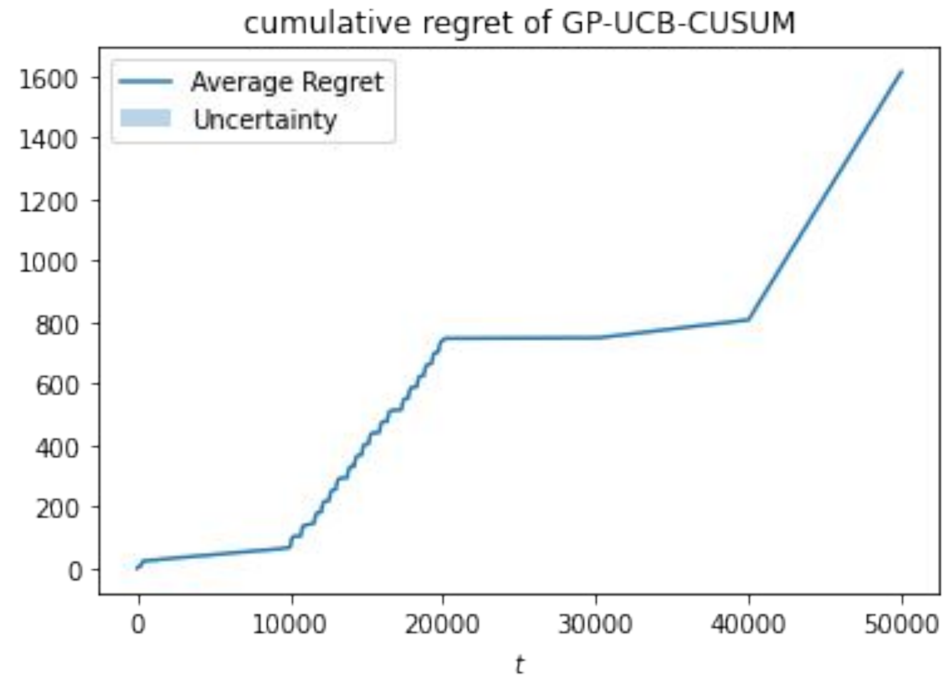
$$W = \left\lfloor 2 \cdot \sqrt{\frac{T_{\text{pricing}} \cdot \log(T_{\text{pricing}})}{\text{intervals} - 1}} \right\rfloor$$

Window size: 735



GP-UCB CUSUM Agent

Used tuned parameters $M = 2 \cdot \left\lfloor \log\left(\frac{T_{\text{pricing}}}{\text{intervals} - 1}\right)\right\rfloor$ $h = 240, \quad \epsilon = 130$

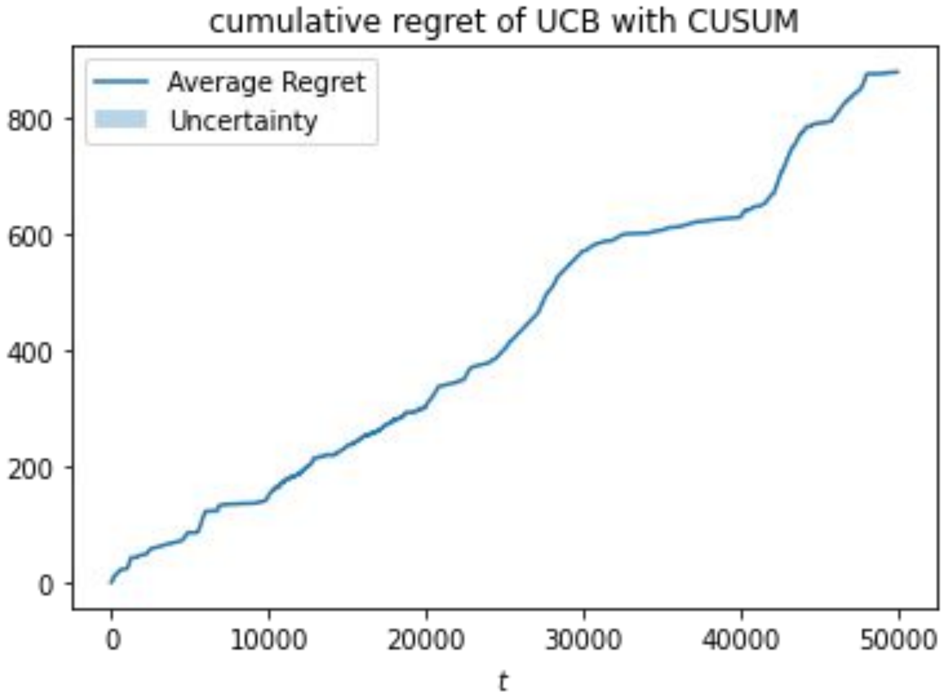


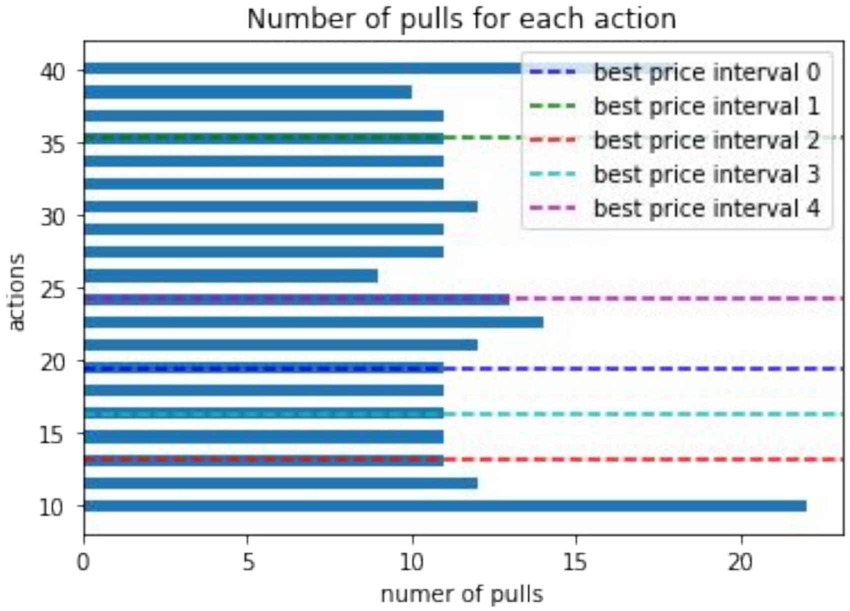
Considerations:

- Parameters tuned in order to get the best results however not all the change points were detected
- Tradeoff between number of changes (ex. in the second interval many wrong changes are detected)
- Variant of the UCB - GP algorithm since in order to reduce the execution time if the same action is performed for more than 40 times with a tolerance of 1e-8 than the GP isn’t updated anymore
- For each change detected all the actions are restored: the whole GP is restarted from scratch

CUSUM UCB Agent

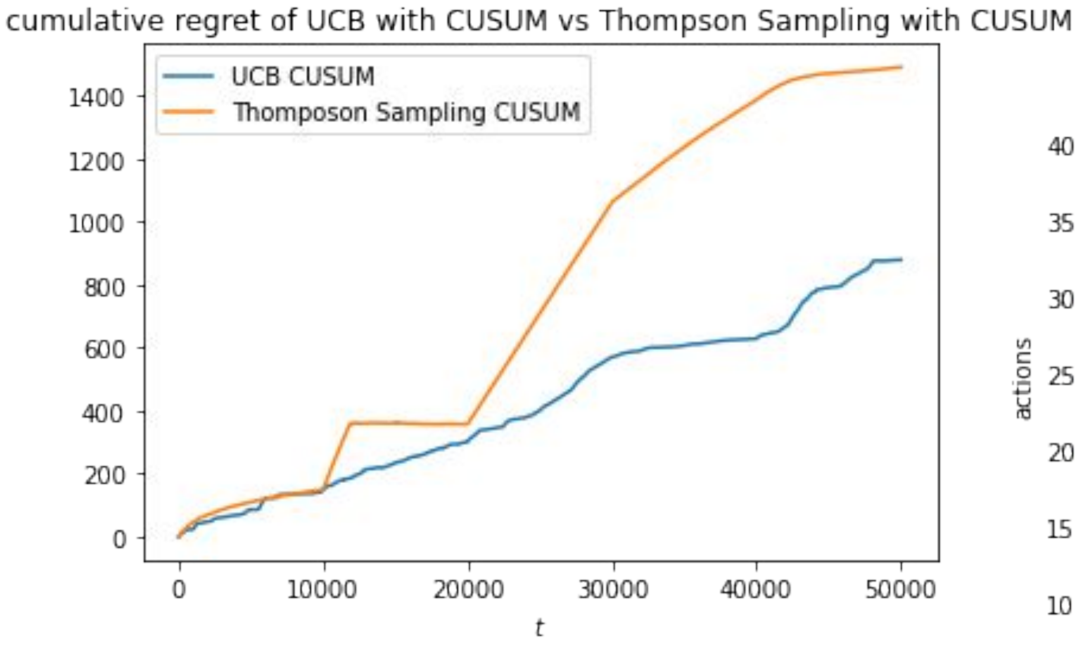
$$h = 2 \cdot \log\left(\frac{T_{\text{pricing}}}{\text{intervals} - 1}\right) \qquad \alpha = \sqrt{\frac{(\text{intervals} - 1) \cdot \log\left(\frac{T_{\text{pricing}}}{\text{intervals}-1}\right)}{T_{\text{pricing}}}} \qquad M = \left\lfloor \log\left(\frac{T_{\text{pricing}}}{\text{intervals}}\right) - 1 \right\rfloor$$





Thompson Sampling CUSUM Agent

h and M as described by theory



Requirement 4 - Three bidders' show-down

```
--scenario ['solo, 'adversarial]
--num_auctions
--n_iters
--num_auctions
--num_participants
--budget
--valuation
--my_ctrs
--theta
--eta
--num_buyers
```

agents:
- StochasticPacingAgent
- AdversarialPacingAgent
- UCB1BiddingAgent

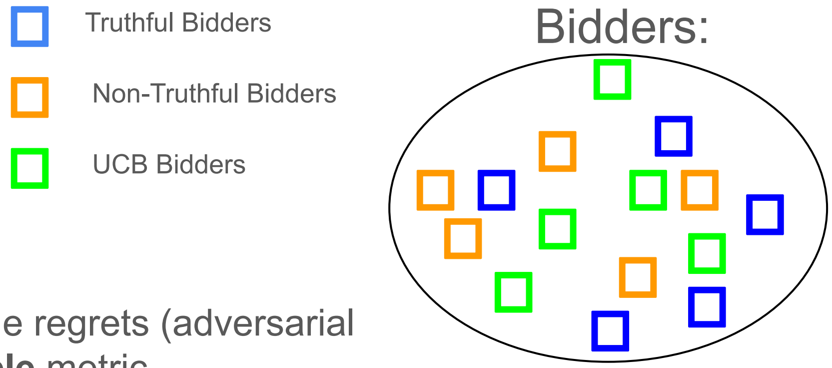
environments:
- None

  or
- AdversarialBiddingCompetitors

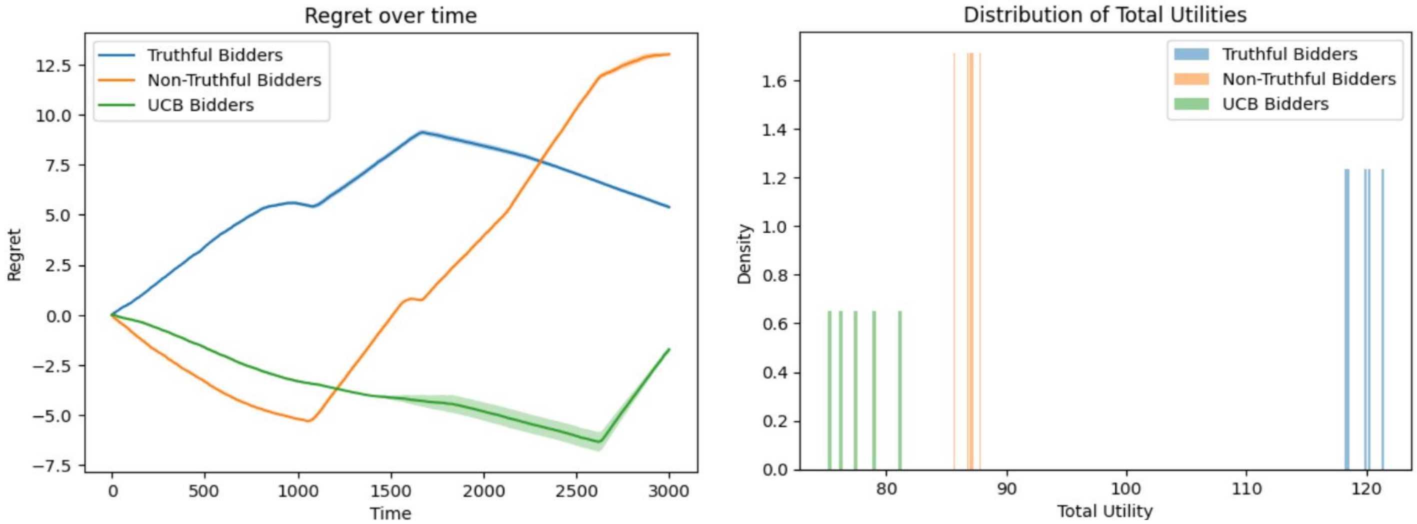
auctions:
- FirstPriceAuction

Two Scenarios:
1. **num_participants is made entirely of bidders of the 3 types each with num_participants//3 members.**
2. **There are 3 agents (1 for each type) and (num_participants - 3) bidders**

Requirement 4.1 - *solo*

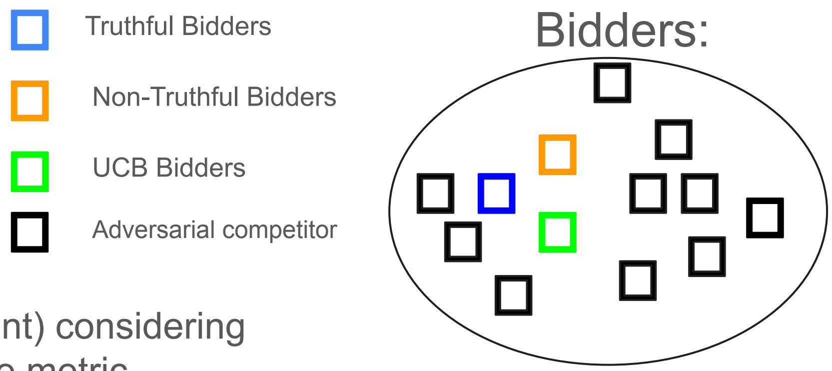


- Here the **Regret** is computed by **averaging** across the regrets (adversarial clairvoyant) of all the bidders of each type. → **Unstable** metric.

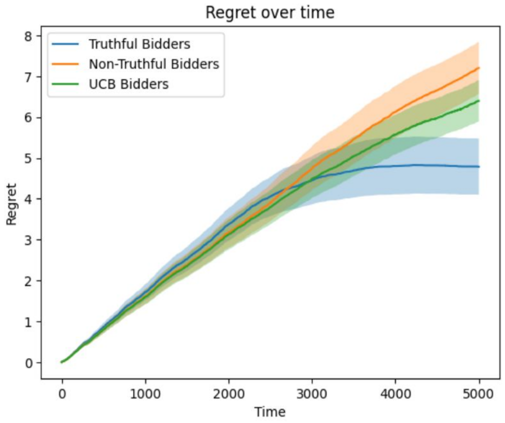


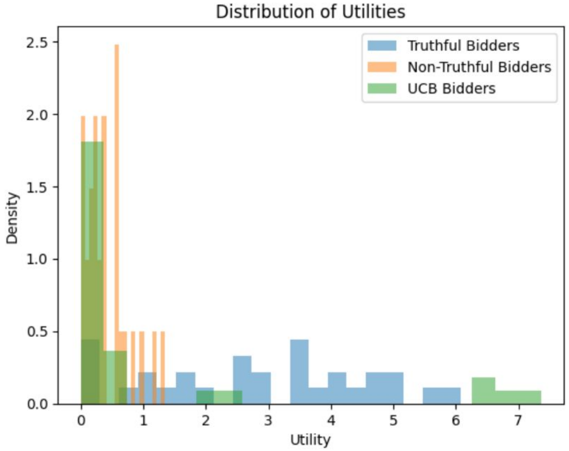

```
%run req4.py --num_auctions 3000 --budget 300 --n_iters 5
--num_participants 30 --eta 0.01 --seed 1 --scenario solo --valuation 0.7
```

Requirement 4.2 - *adversarial*



- Here we consider the 3 Regrets (adversarial clairvoyant) considering only the bids of the adversarial bidders. → More stable metric.

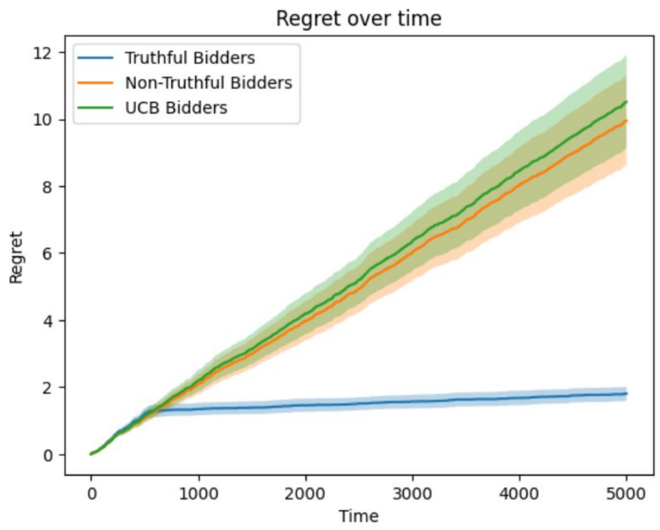




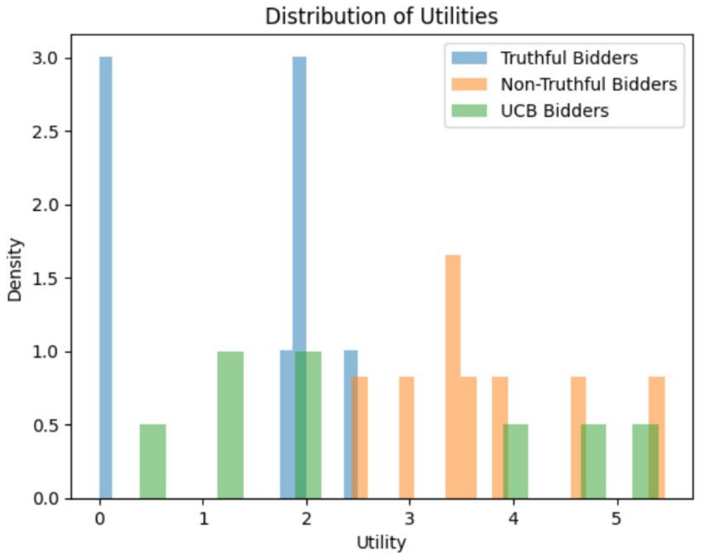

```
%run req4.py --num_auctions 5000 --budget 200 --n_iters 30 --num_participants 100 --eta 0.01 --seed 1 --scenario adversarial --valuation 0.8 --my_ctrs 0.8,0.8,0.8
```

Requirement 4.2 - *adversarial*

- Set eta to the value from theory (higher than before)



- Raise the ctr of the non-truthful bidder from 0.8 to 0.85

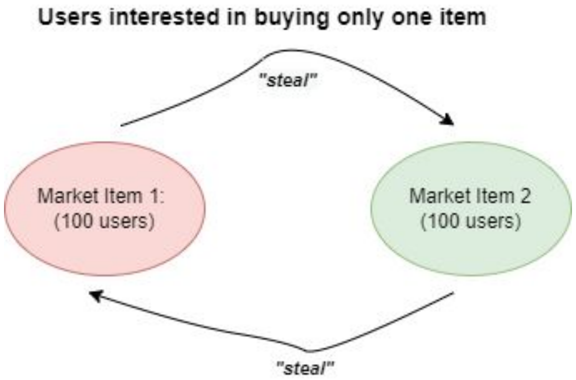


Requirement bonus

Demand curves

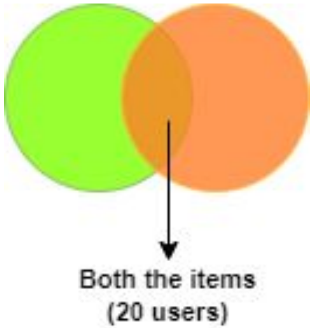
$$D_1(p_1, p_2) = \max\left(0, \left(1 - \beta_1 \times \left(\frac{p_1}{\text{max\_price}}\right) + \gamma_1 \times \left(\frac{p_2}{\text{max\_price}}\right)\right)\right)$$

$$D_2(p_1, p_2) = \max\left(0, \left(1 - \beta_2 \times \left(\frac{p_2}{\text{max\_price}}\right) + \gamma_2 \times \left(\frac{p_1}{\text{max\_price}}\right)\right)\right)$$



$$D_{\text{both}}(p_1, p_2) = \max\left(0, \left(1 - \text{both\_factor} \times \left(\frac{p_1}{\text{max\_price}}\right) - \text{both\_factor} \times \left(\frac{p_2}{\text{max\_price}}\right)\right)\right)$$

$$\begin{aligned} d_{\text{total}}(p_1, p_2) = &\left(\frac{\text{num\_buyers\_market1}}{\text{num\_total\_buyers}}\right) \cdot D_1(p_1, p_2) \\ &+ \left(\frac{\text{num\_buyers\_market2}}{\text{num\_total\_buyers}}\right) \cdot D_2(p_1, p_2) \\ &+ \left(\frac{\text{num\_buyers\_both}}{\text{num\_total\_buyers}}\right) \cdot \text{curve\_both}(p_1, p_2) \end{aligned}$$



Demand curve modelization for two items market:

item 1 cost : 10

item 2 cost : 15

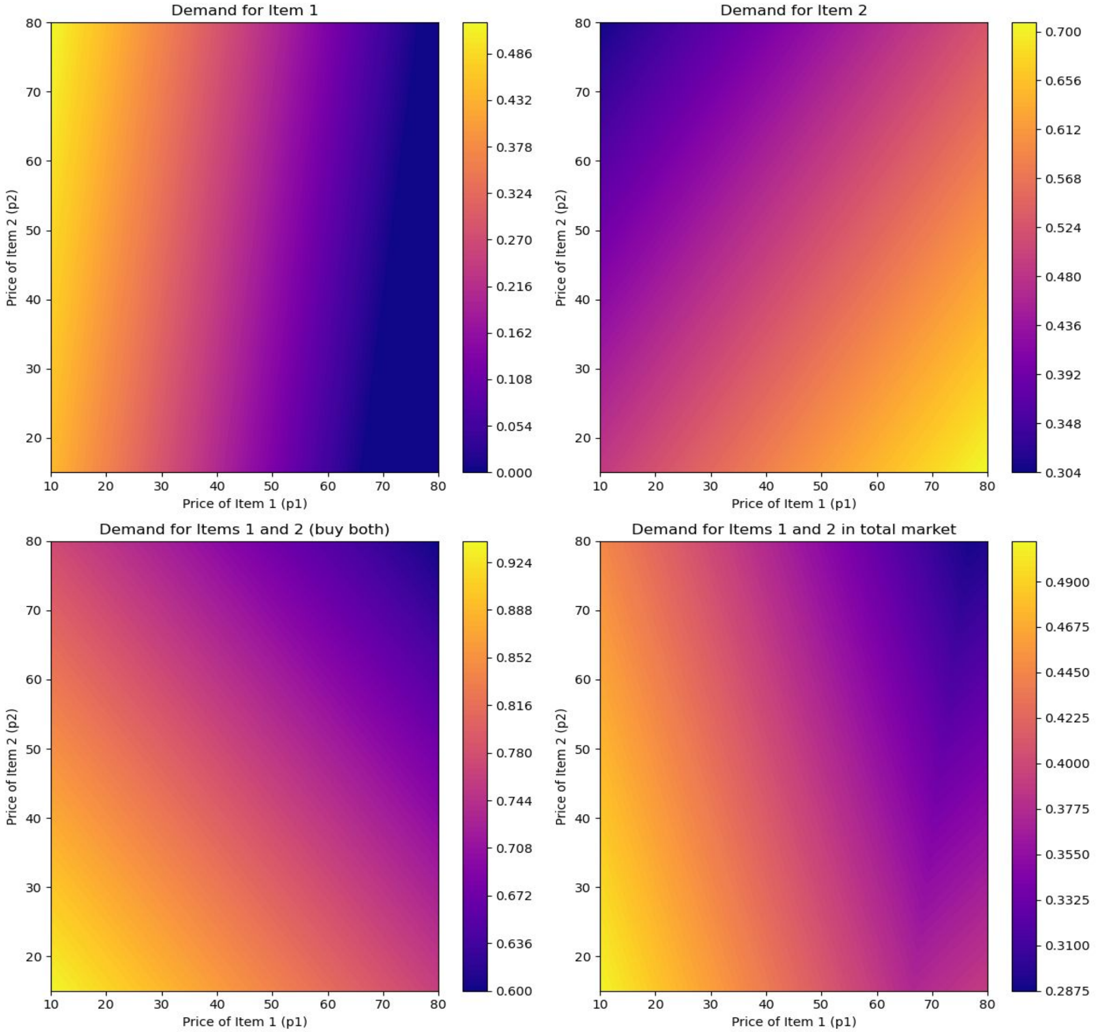
max price : 80

num buyers total : 220

Furthermore we know 20 are interested in buying both the products

Time duration : 1000

Discretization of the price space : 100



Profit curve and Environment

$$\begin{aligned}\text{profit\_curve} = \text{num\_buyers\_market1} \cdot D_1(p_1, p_2) \cdot (p_1 - \text{cost1}) \\ + \text{num\_buyers\_market2} \cdot D_2(p_1, p_2) \cdot (p_2 - \text{cost2}) \\ + \text{num\_buyers\_both} \cdot D_{\text{both}}(p_1, p_2) \cdot (p_1 + p_2 - \text{cost1} - \text{cost2})\end{aligned}$$

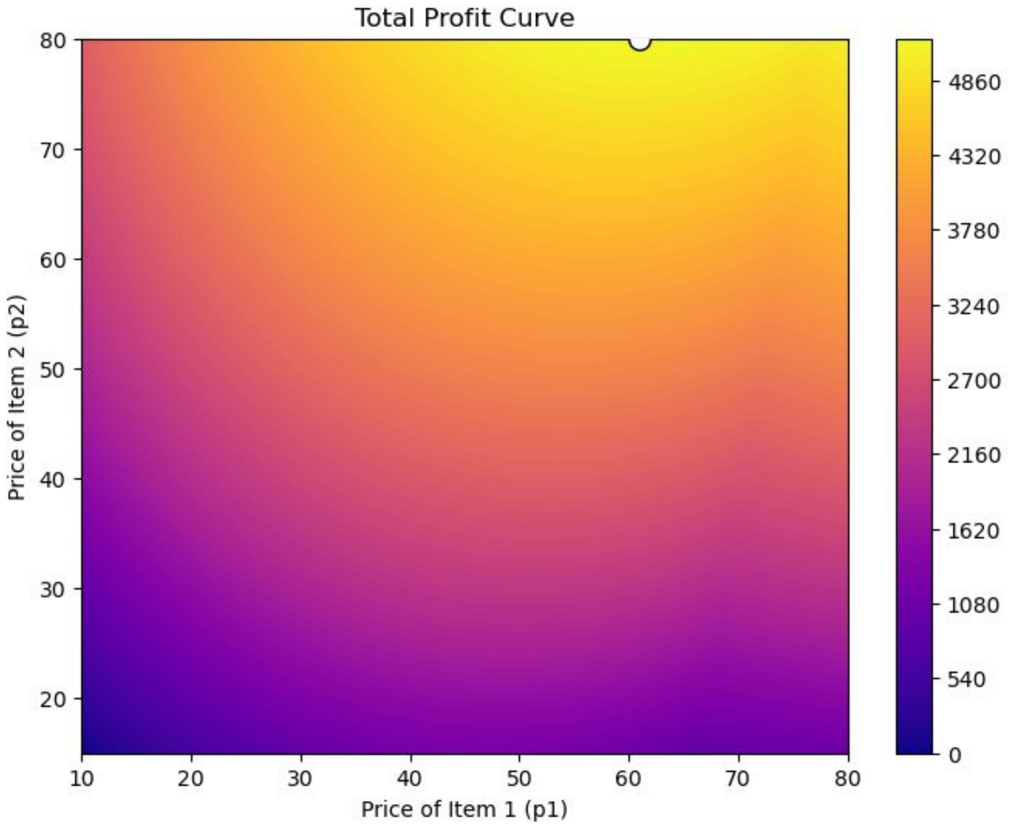
$$d_{t_1} \sim \text{Bin}\left(\text{num\_buyers\_market1}, D_1(p_1, p_2)\right)$$

$$d_{t_2} \sim \text{Bin}\left(\text{num\_buyers\_market2}, D_2(p_1, p_2)\right)$$

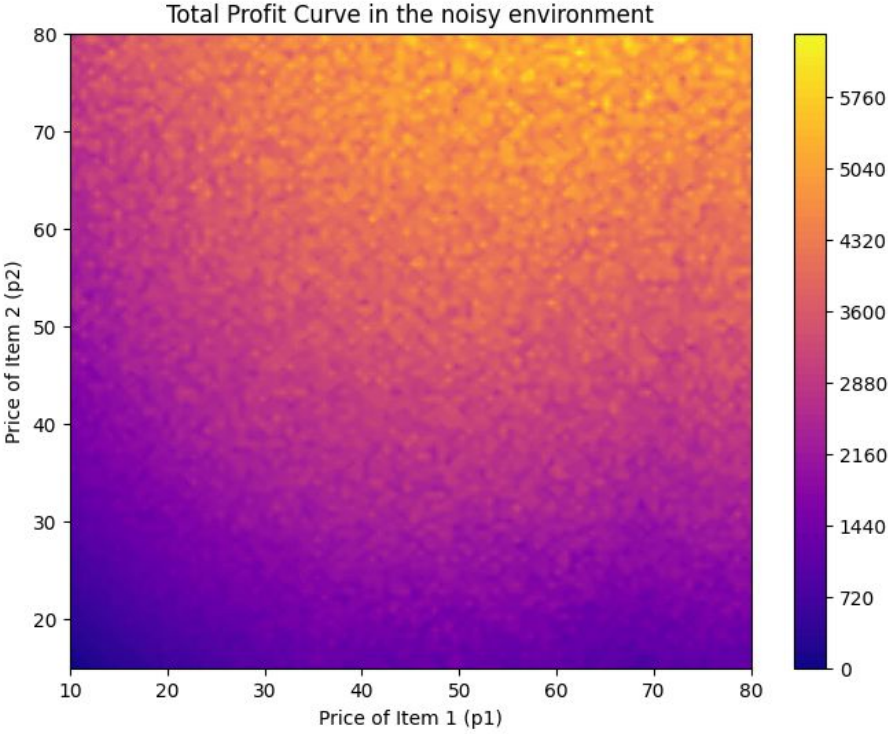
$$d_{t_{\text{both}}} \sim \text{Bin}\left(\text{num\_buyers\_both}, D_{\text{both}}(p_1, p_2)\right)$$

$$d_t = d_{t_1} + d_{t_2} + d_{t_{\text{both}}}$$

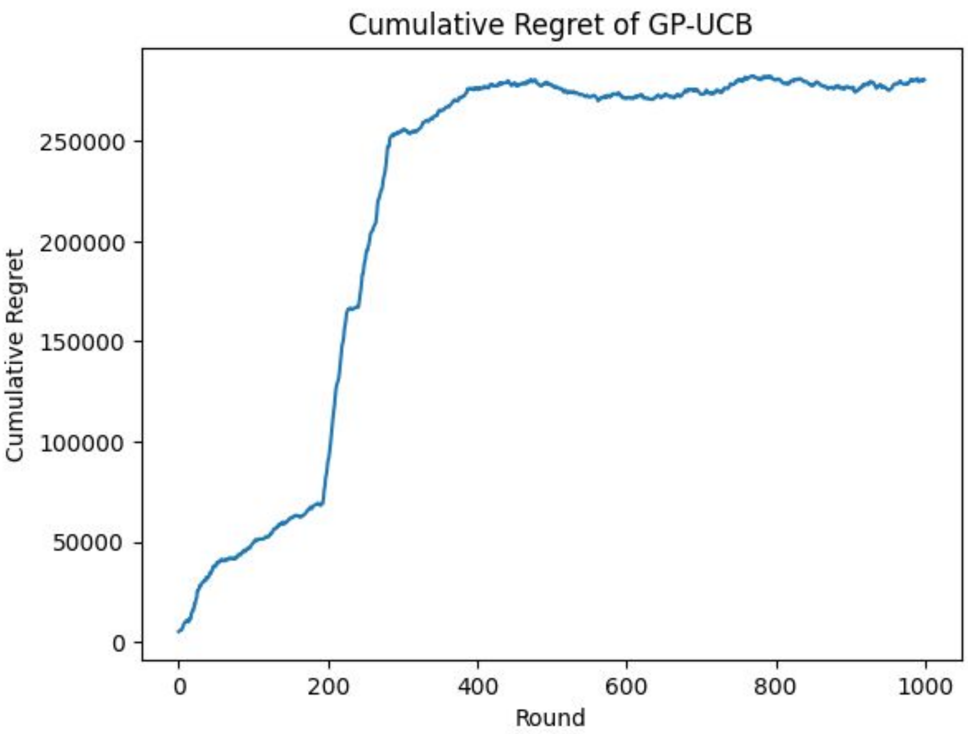
$$r_t = (p_1 - \text{cost1}) \cdot d_{t_1} + (p_2 - \text{cost2}) \cdot d_{t_2} + (p_1 + p_2 - \text{cost1} - \text{cost2}) \cdot d_{t_{\text{both}}}$$

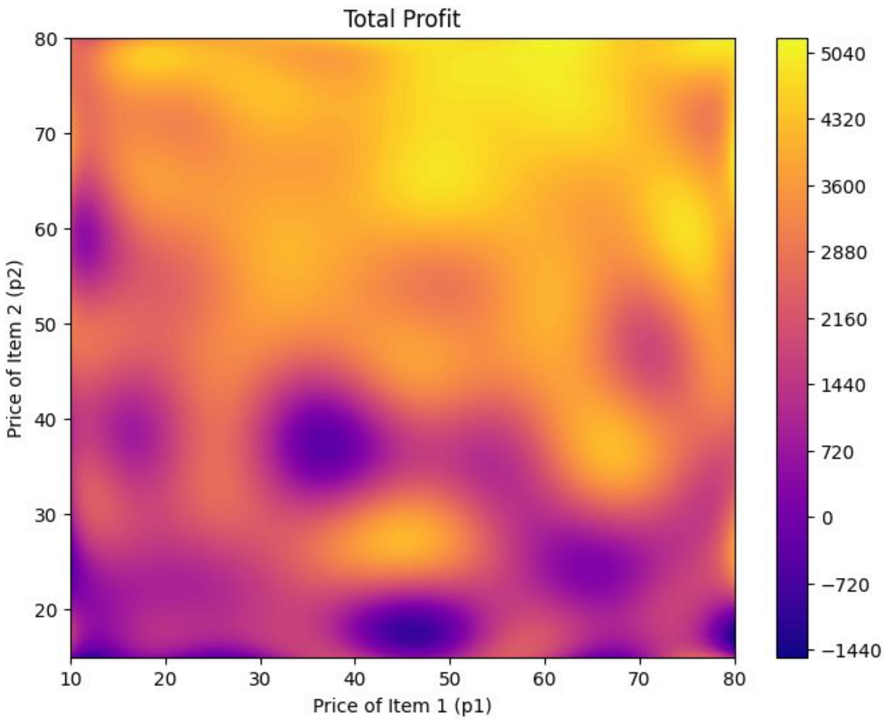



```
best price for item 1:  60.90909090909091
best price for item 2:  80.0
total profit :  5158.290289256198
Expected number of users that would buy for
each population type at the found prices:
      Market 1:  12.414772727272727
      Market 2:  46.53409090909091
      Both markets:  12.954545454545453
```



Agent: GP UCB Multidimensional






```
Best price for item1 found:  61.61616161616162
Best price for item2 found:  80.0
```