Bidding Algorithm Overview

Yuanlong Chen

Brand Auction

July 11, 2024

Agenda

1. **Bidding Products**
2. **Auto-Bidding Algorithm**
3. **Cost Cap Bidding Algorithm**

Bidding Products

Three Major Bidding Products

1. **Max Delivery (a.k.a no bid, lowest cost)**
   Advertiser's input: budget, no ROI requirements
2. **Cost Cap**
   Advertiser's input: budget + cost cap (max CPX)
3. **Manual bidding(other companies)**
   Advertiser's input: budget + fixed bid price

Auto Bidding Algorithm

Formulation(use CPC ads as example)

$$\max_{x_t} \sum_{t=1}^{T} x_t \cdot r_t \quad \text{s.t.} \quad \sum_{t=1}^{T} x_t \cdot c_t \leq B$$

where

t – t-th auction opportunity

T – total auction opportunity

r_t – pctr at t

c_t – cost at t

B – total budget

x_t – indicator whether t-th auction win or lose, $x_t \in \{0, 1\}$

Auto Bidding Algorithm

Optimal Bidding Formula

Auction Mechanism: assume we use second price auction, then

$$x_t = 1_{\{\mathrm{ecpm}_t > c_t\}} \quad \text{where} \quad \mathrm{ecpm}_t = b_t * r_t$$

Idea: use primal-dual + online gradient descent to derive an optimal bidding formula for auto-bidding campaigns

Auto Bidding Algorithm

Optimal Bidding Formula(cont'd)

STEP 1: Write Lagrangian

$$\mathcal{L}(x_t, \lambda) = \sum_{t=1}^{T} x_t r_t + \lambda \left(B - \sum_{t=1}^{T} x_t c_t \right)$$

$$= \sum_{t=1}^{T} x_t (r_t - \lambda c_t) + \lambda B$$

Optimal Bidding Formula(cont'd)

STEP 2: Derive dual problem

$$\mathcal{L}^*(\lambda) = \max_{x_t} \mathcal{L}(x_t, \lambda)$$

$$= \sum_{t=1}^{T} (r_t - \lambda c_t)_+ + \lambda B$$

where $(z)_+ = \max(0, z)$ (a.k.a ReLU)

Auto Bidding Algorithm

Optimal Bidding Formula(cont'd)

STEP 3: Solve dual problem

$$\lambda^* = \min_{\lambda \geq 0} \mathcal{L}^*(\lambda)$$

$$\Rightarrow \text{ecpm}_t^* = \frac{r_t}{\lambda^*}$$

i.e.

$$b_t^* \cdot r_t = \frac{r_t}{\lambda^*} \Rightarrow b_t^* = \frac{1}{\lambda^*}$$

Note: this is true for all incentive compatible auctions

Conclusion: optimal bid is constant bid

Auto Bidding Algorithm

How To Find Optimal Bid b^*

(From KKT condition) b^* is the bid level that exactly depletes the budget, i.e.,

$$\sum_{t=1}^{T} x_t c_t \leq B \implies \sum_{t=1}^{T} x_t c_t = B$$

Assume r_t, c_t are subject to i.i.d. at $(\tau, \tau + d\tau)$, the cost

$$\sum_{\tau \leq t \leq \tau + d\tau} x_t c_t \sim \# \text{ of auction opportunities in } (\tau, \tau + d\tau)$$

Auto Bidding Algorithm

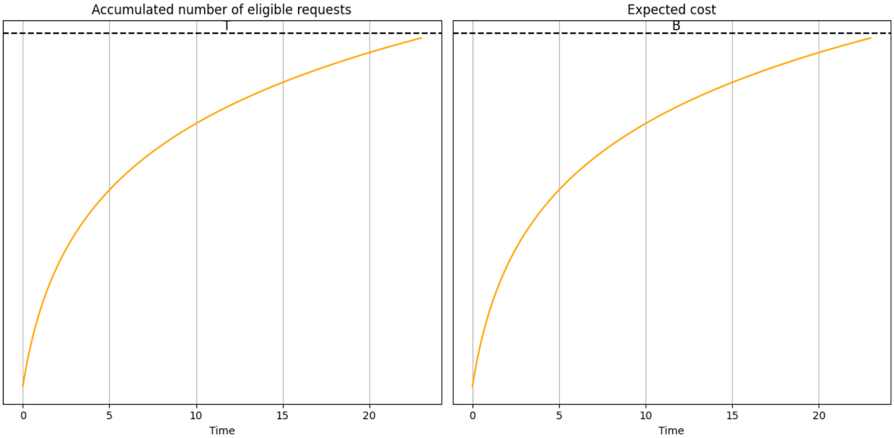

Figure: Left: Accumulated number of eligible requests. Right:Accumulated expected cost.

Auto Bidding Algorithm

In practice, we construct an expected cost curve based on flow ratio map (e.g. 15 minute granularity):

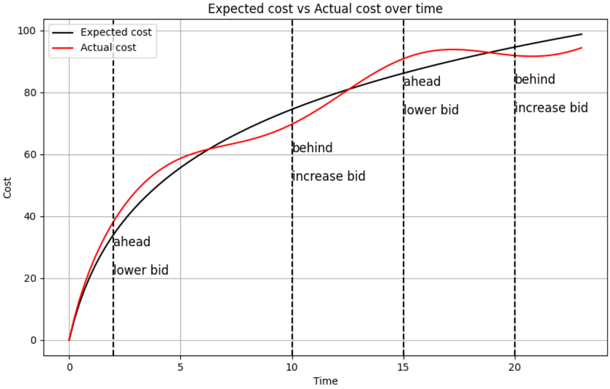

Figure: Expected cost vs Actual cost over time

Auto Bidding Algorithm

How To Find Optimal Bid b^* (cont'd)

Intuitively, do pacing by comparing actual cost to expected cost:

- Ahead schedule $\rightarrow$ lower bid
- Behind schedule $\rightarrow$ increase bid
- Ahead schedule $\rightarrow$ lower bid
- Behind schedule $\rightarrow$ increase bid

By following the curve: $b_t \rightarrow b^*$

Auto Bidding Algorithm

Controller-based Algrorithm

Approach 1: PID Controller

1. $e(t_k) = C_e(t_k) - C_a(t_k)$
2. $\phi(t_{k+1}) \leftarrow \lambda_P e(t_k) + \lambda_I \sum_{j=1}^{k} e(t_j)\Delta t_j + \lambda_D \frac{\Delta e(t_k)}{\Delta t_k}$
3. $b(t_{k+1}) \leftarrow b(t_k) \cdot \exp(\phi(t_{k+1}))$

Auto Bidding Algorithm

Controller-based Algrorithm(cont'd)

Approach 2: MPC Controller

MPC optimizes the entire future spend, at time t, remaining budget $B_{t,r}$

$\Rightarrow$ expected cost speed cs_t

$\Rightarrow$ expected bid price b_t

Goal: find $cs_t = f(b_t)$ where f is monotonically non-decreasing

Auto Bidding Algorithm

iMPC: Longest Increasing Subsequence of historical (bid, cost) pairs + interpolation

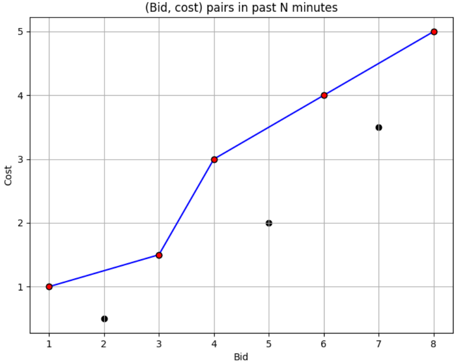

Figure: (Bid, cost) pair in past N minutes. Blue line represents the longest increasing subsequence used for interpolation.

Auto Bidding Algorithm

iMPC Controller(cont'd)

At time, t, for a given cost speed cs_t, find corresponding bid price b_t

iMPC algorithm:

1. fetch historical bid-cost pairs in past N minutes
2. Choose longest increasing subsequence(LIS), then interpolate to get f
3. $b_t = f^{-1}(cs_t)$

Auto Bidding Algorithm

MPC Controller

LIS drawbacks: potentially throws away data points.

Q: How to utilize all data to reduce noise while maintain monotonicity property of f?

Idea: Use Isotonic Regression

Auto Bidding Algorithm

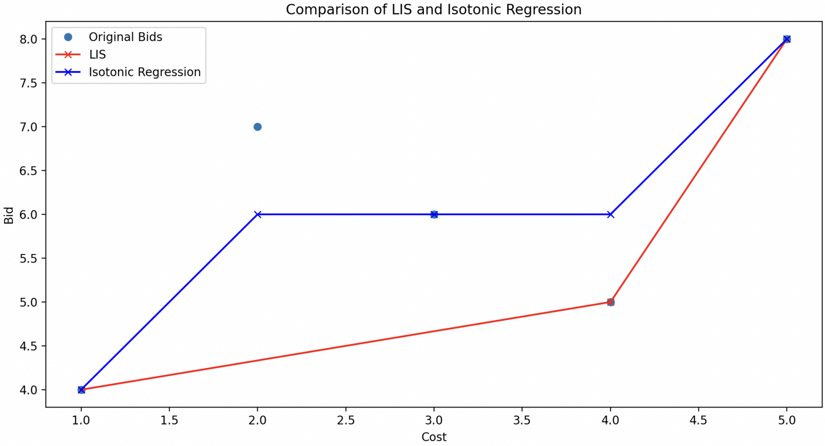

Figure: Isotonic Regression vs LIS.

Cost Cap Algorithm

Formulation(use CPC ads as example)

$$\max_{x_t} \sum_{t=1}^{T} x_t r_t$$

subject to

$$\sum_{t=1}^{T} x_t c_t \leq B$$

$$\sum_{t=1}^{T} x_t c_t \leq C \left(\sum_{t=1}^{T} x_t r_t \right)$$

where C is cost cap (advertiser bid)

Cost Cap Algorithm

Optimal Bidding Formula

Use primal-dual method, similarly we get optimal bid:

$$b^* = \frac{1 + \mu^* C}{\lambda^* + \mu^*} = \frac{\lambda^*}{\lambda^* + \mu^*} \cdot \frac{1}{\lambda^*} + \frac{\mu^*}{\lambda^* + \mu^*} \cdot C$$

Note: if $\mu^* = 0$, $b^* = \frac{1}{\lambda^*}$ which is equivalent to auto bidding
In practice, we use the "cost-min" algorithm:

$$b_t = \min(\text{flow\_bid}, \text{risk\_bid})$$

where

$$\text{risk\_bid} = \frac{\text{remaining budget}}{\text{remaining event goal}}$$

flow_bid is the same as in auto-bidding.