Budgeting and Bidding in Ad Systems: Theory and Practice

Aranyak Mehta
Market Algorithms, Google Research, Mountain View, CA.

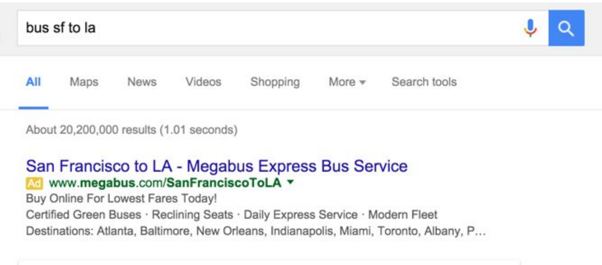
Outline

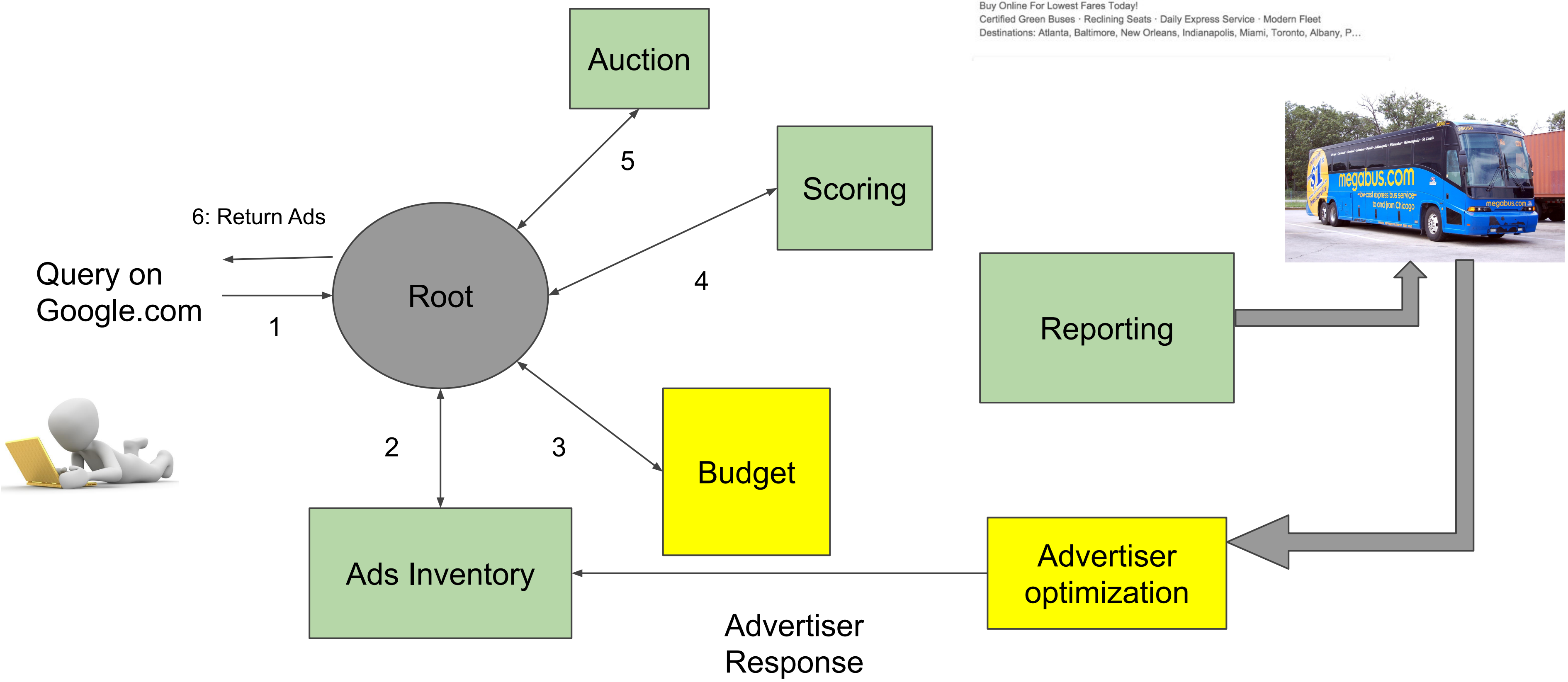
Topics:

1. **Budget Allocation:**
   - Algorithms based on Online Matching
   - Algorithms based on Reinforcement Learning

2. **Auto-Bidding:**
   - Algorithms
   - Equilibrium

Search Ads System Overview

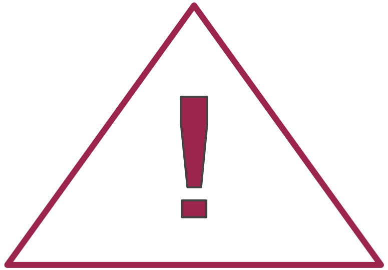




Budget Allocation | Online Matching

Motivation: Demand constraints in Repeated Auctions

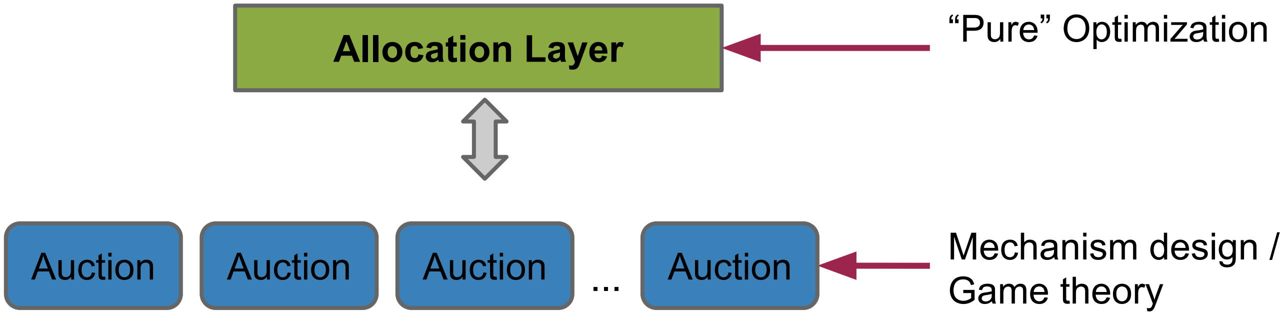
- Auction each arriving ad slot.
- Stateful because of budget constraints.
- Mismatched bidding components.



Traffic = 1000 clicks!

Allocation on top of auction

- Can model it as a repeated online auction with demand constraint.
  - Impossibility results
  - Impractical
- Design: Allocation layer on top of online stateless auction:

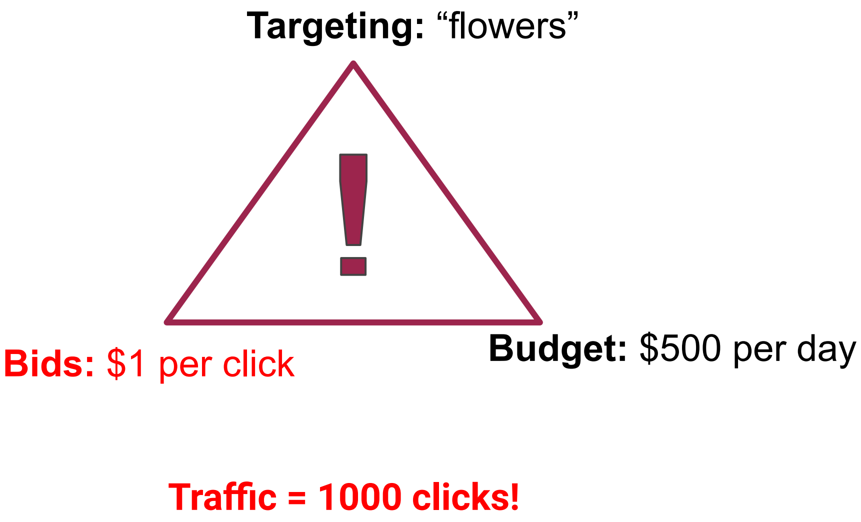


Two Methods

- Bid Lowering
  - “Your bid was too high.”
- Throttling
  - “Your targeting was too broad.”

Two Methods

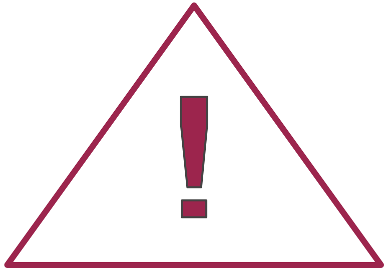
- Bid Lowering
  - "Your bid was too high."
- Throttling
  - "Your targeting was too broad."



Two Methods

- Bid Lowering
  - "Your bid was too high."
- Throttling
  - "Your targeting was too broad."

Targeting: "flowers"



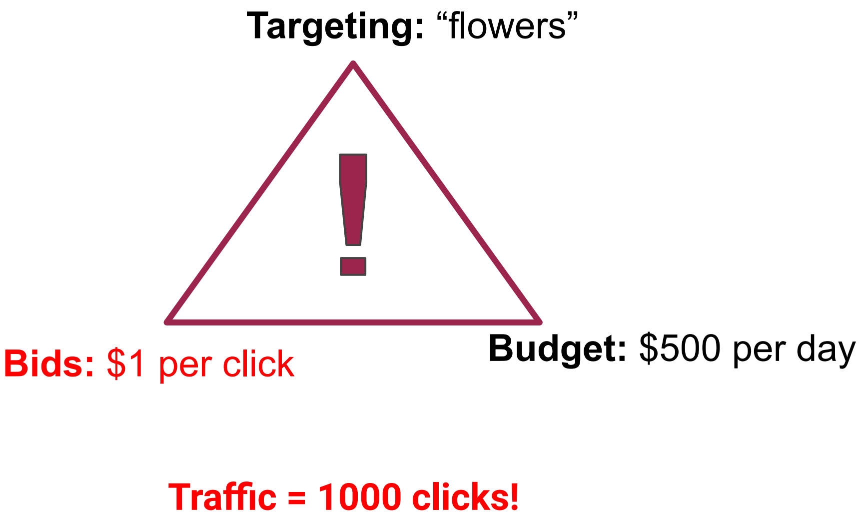
Bids: $1 per click

Budget: $500 per day

Traffic = 1000 clicks!

Two Methods

- Bid Lowering
    - "Your bid was too high."
    - **Heuristic:** reduce bid by some multiplier.
    - **Theoretical abstraction:** How to incorporate the interaction across ads?
- Throttling
    - "Your targeting was too broad."



An abstraction: The “AdWords Problem”

Definition *(M., Saberi, Vazirani, Vazirani, FOCS 2005, JACM 2007)*

- N advertisers, advertiser a has budget $B(a)$
- M search queries that arrive online, advertiser a has bid $bid(a, q)$ for query q

Decision: Algorithm needs to allocate q to one of the advertisers irrevocably (or discard). Allocated advertiser depletes budget by $bid(a, q)$

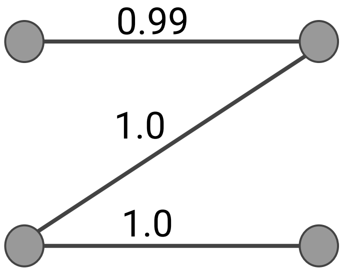
Goal: Maximize sum of values over all queries

Generalizes *online bipartite matching [KVV’90]*

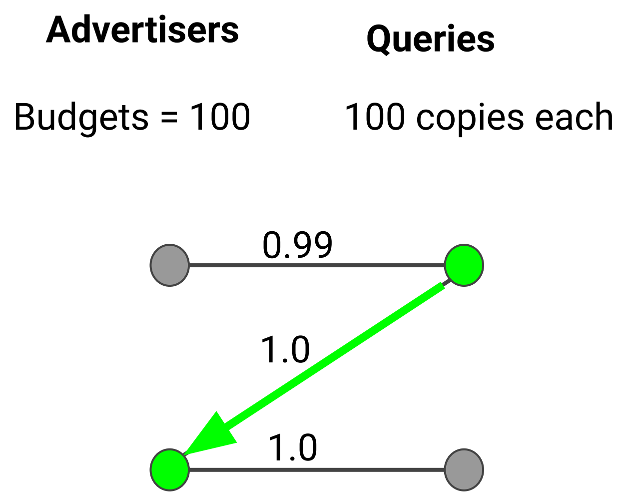
The AdWords Problem

Advertisers **Queries**

Budgets = 100 100 copies each



The AdWords Problem



“Greedy” solution would lead to ½ of the maximum potential.

The MSVV Algorithm

$spent(a)$ = fraction of a's budget already used up.

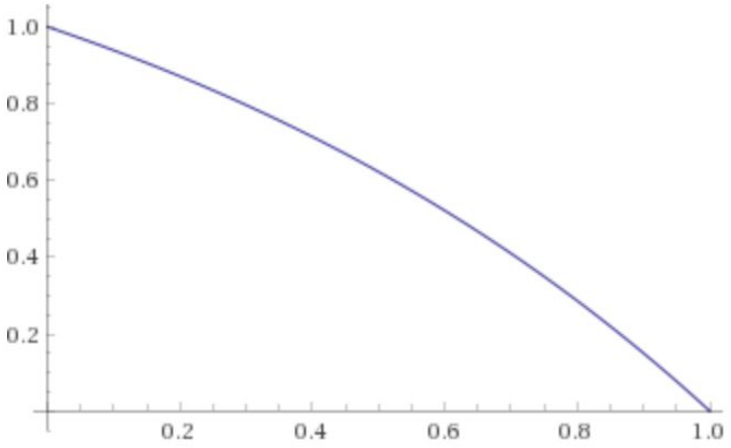
When query q arrives, allocate it to an advertiser that maximizes

$\boldsymbol{bid(a, q) * \Psi(spent(a))}$ where $\Psi(x) \propto 1 - exp(-(1 - x))$.

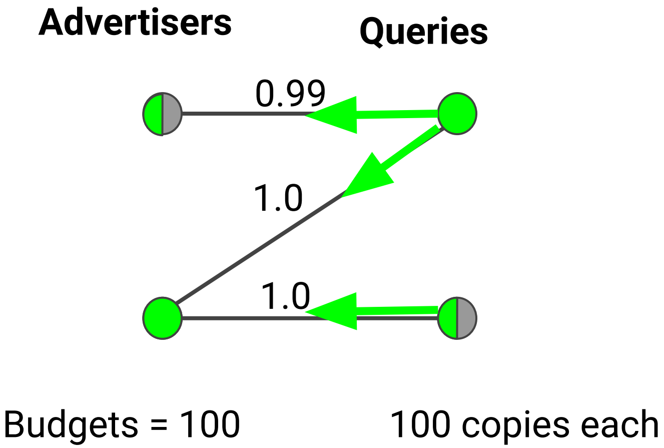
Theorem [MSVV05]

Achieves optimal competitive ratio $1 - 1/e \sim 63\%$

Note: A worst-case guarantee, even if we do not have any estimates.



The AdWords Problem



What about stochastic input?

[Devanur Hayes EC 2009]

- **Intuition:** [MSVV05] proof updates dual variables / bid multipliers as the sequence arrives (explicitly shown in [BJN07]).
  In iid or random order setting, you can sample and estimate duals.

- **Algorithm:**
  - Sample initial segment
  - Solve the LP for the sample
  - Use those duals for the rest of the sequence.

- **Theorem:** 1-epsilon in random order model

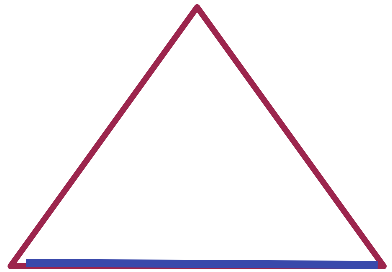
Display ads

[FKMMP WINE 2009]

- Original solution:
  LP / max flow on estimated graph.

- Algorithm 1
  w’ = w - penalty(usage, capacity)

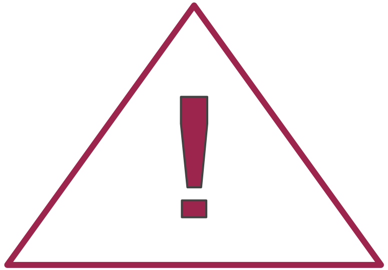
- Algorithm 2: Learning duals a la DH09



Two Methods

- Bid Lowering
  - “Your bid was too high.”
- Throttling
  - “Your targeting was too broad.”

Targeting: “flowers”



Bids: $1 per click

Budget: $500 per day

Traffic = 1000 clicks!

Throttling

- Extreme of bid lowering
  - bid multiplier either 0 or 1.

- “Vanilla” Throttling:

Probability of participation in each auction = Budget / Max-Spend-estimate

Throttling

- Optimized Throttling [Karande, Mehta, Srikant WSDM 2013??]
    - Provide an optimized set of options for the advertiser, rather than random.

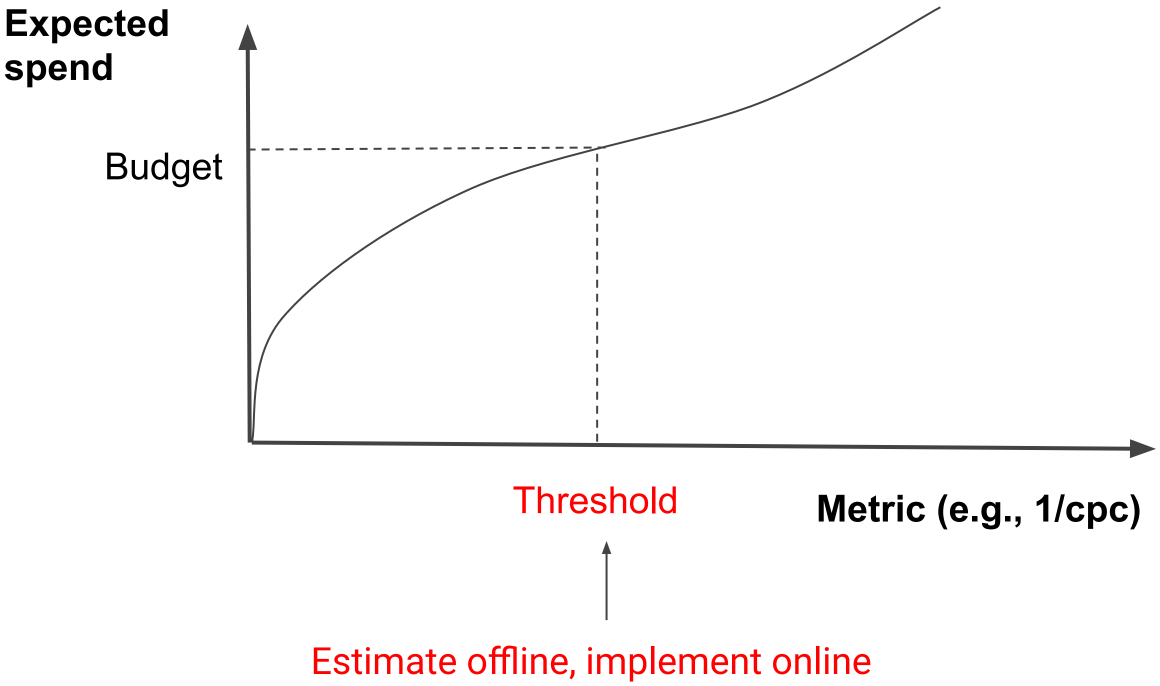
- Knapsack formulation

$$\max_{S} \sum_{i \in S} \mathbf{ctr}_i$$

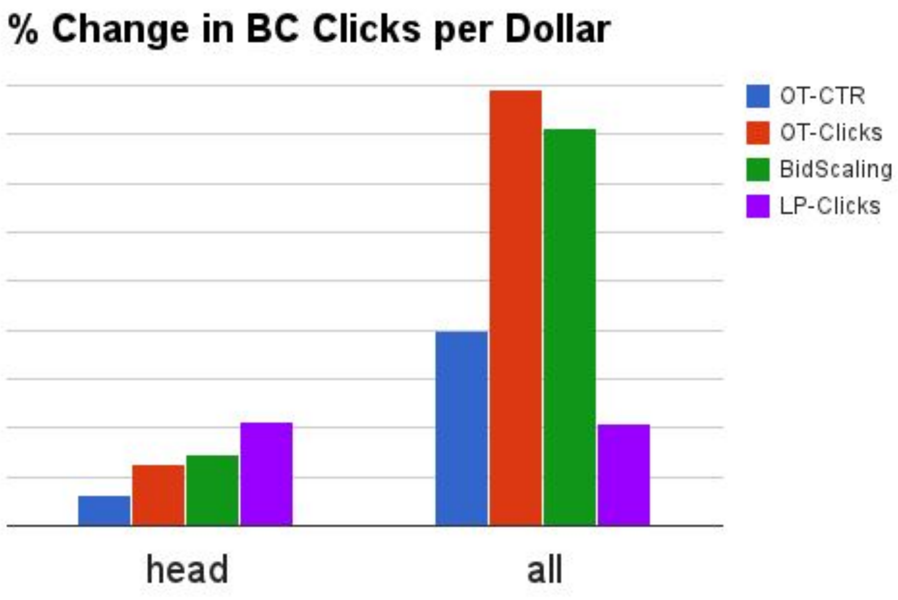
$$\text{s.t.} \quad : \quad \sum_{i \in S} \mathbf{spend}_i \leq B$$

Greedy heuristic: Participate in auctions with best ctr/spend = 1/cpc

Optimized Throttling



Optimized Throttling



A lot more work in this direction.

Survey Book: *Online Matching and Ad Allocation*, M., 2013.

Budget Allocation | Reinforcement Learning

Part of a broader theme

[*A New Dog learns Old Tricks*, Kong, Liaw, M., Sivakumar, ICLR 2019.]

CAN

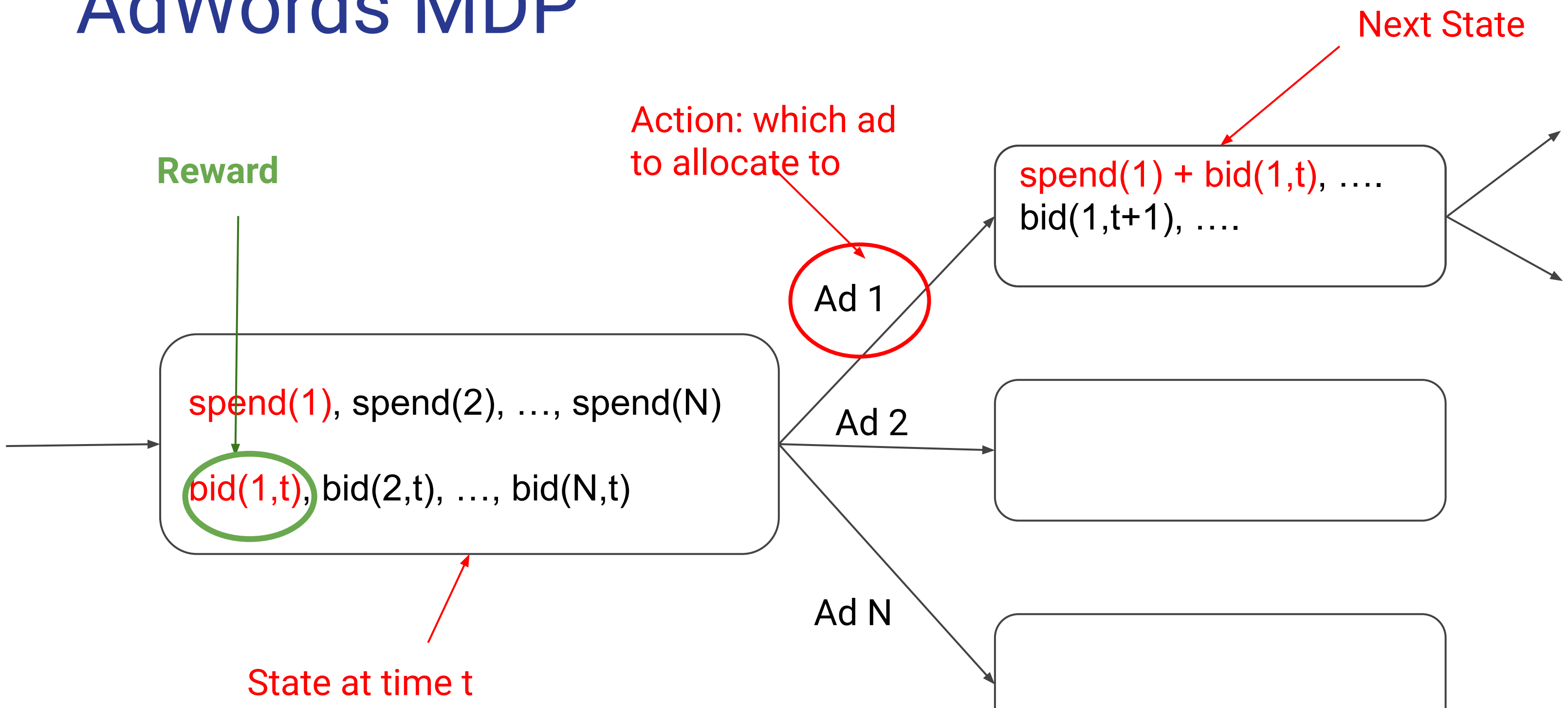
DEEP REINFORCEMENT LEARNING

DESIGN

WORST CASE

ONLINE OPTIMIZATION ALGORITHMS?

“AdWords MDP”



Next State
Action: which ad
to allocate to
Reward
spend(1) + bid(1,t), ….
bid(1,t+1), ….
Ad 1
spend(1), spend(2), …, spend(N)
bid(1,t), bid(2,t), …, bid(N,t)
Ad 2
Ad N
State at time t

Learning an Agent

Goal: Learn agent's policy function that maps state to action.

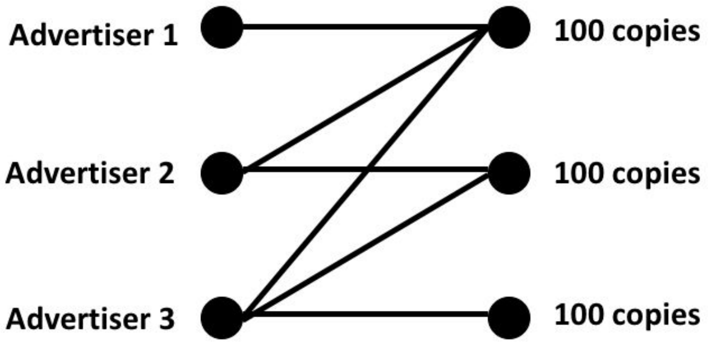
Network: Standard 5-layer 500-neuron-per-layer network with ReLU non-linearity

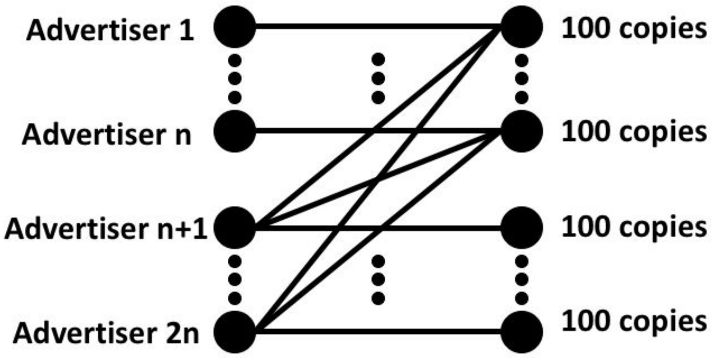
Training: Standard **REINFORCE policy-gradient** learning with learning rate 1e-4, batch size 10.

Takes few hours typically on single-threaded standard Linux desktop

Punch line: It works!

Training Set: Universal Distribution



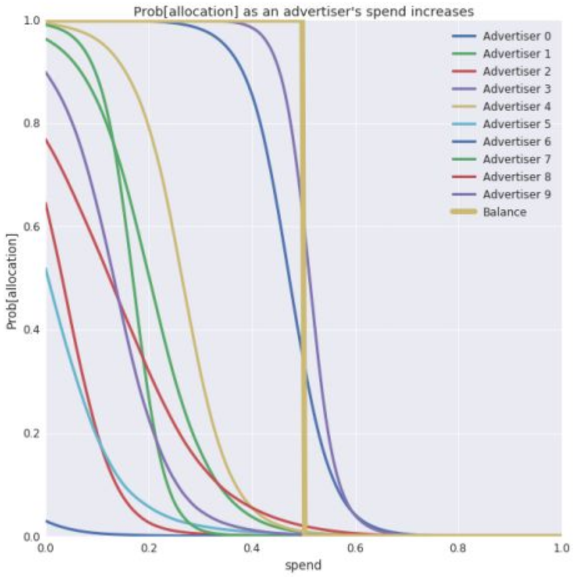


Two expanded versions of the Z-graph

How does the network solve it?

Did it “Find the MSVV Algorithm”? How to evaluate?

Probing the network as a black box.



Warm-up: 0/1 bids

Pretend we’re in the middle of execution for an instance. We’re at an item arrival.

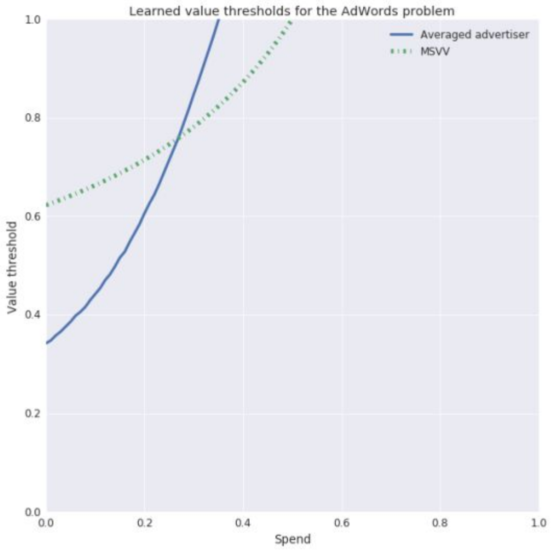
All advertisers have bid=1
All except advertiser i have spend=0.5.

x-axis: spend
y-axis: Probability that advertiser i wins the item

How does the network solve it?

Did it “Find the MSVV Algorithm”? How to evaluate?

1. **Probing the network as a black box.**



(b)

General Case:

All advertisers *except advertiser 0* have bid=1, spend=0.5.

x-axis: spend(0)
y-axis: Minimum bid to win the item.

Blue: Learned Agent
Green: *OPT (MSVV)*

Training small testing big

Table 3: This table compares the performance of the learned algorithm compared the BALANCE in the discretized state space. Here, the agent is trained on the adversarial graph with the ad slots arriving in a permuted order. The agent was only trained on the input instance with 20 advertisers and a common budget of 20 but tested on instances with up to 10^6 ad slots.

Training Regime

No. of advertisers	Budgets (common)	No. of ad slots	Approx. of BALANCE
10	10	100	0.9
20	20	400	0.92
30	30	900	0.88
10	2000	20000	0.85
10	4000	40000	0.85
25	4000	100000	0.84
50	400	20000	0.84
100	100	10000	0.85
100	1000	100000	0.85
200	100	20000	0.85
500	50	25000	0.85
1000	100	10000	0.84
25	40000	1000000	0.84

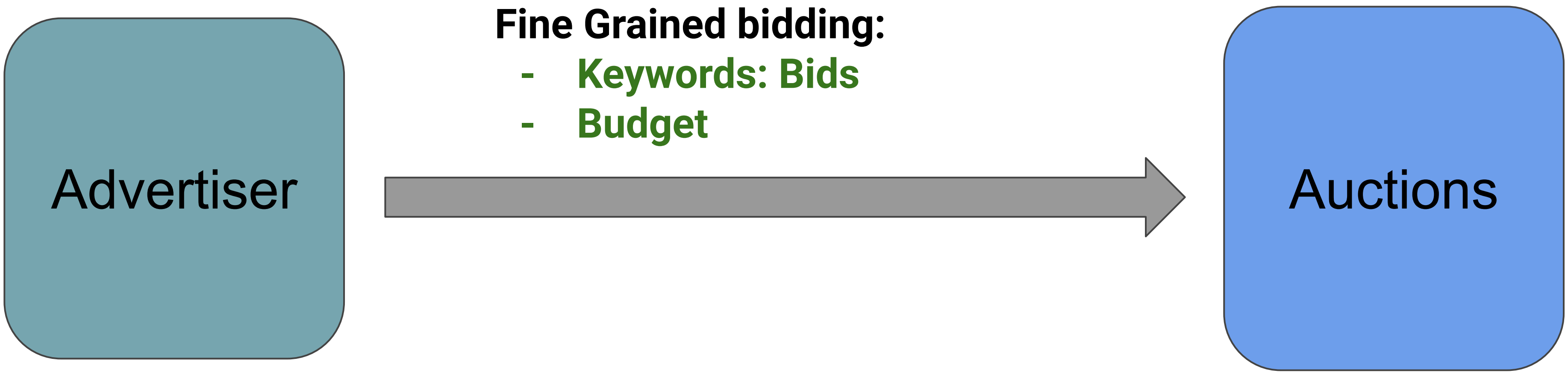
What does this mean for practice?

- RL can potentially find worst case algorithms.
- We know RL can adapt to real distributions / data well.
- Opens up potential to merge ML and Algorithms to work more in tandem.

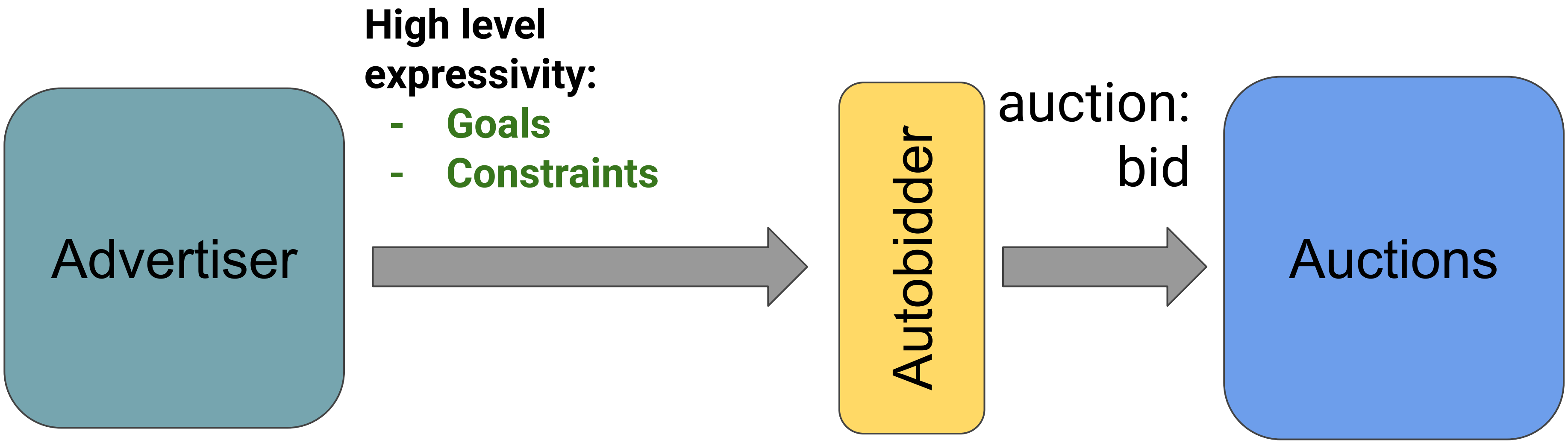
Auto-Bidding: Algorithms and Equilibrium

[Aggarwal, Badanidiyuru, M., 2019]

Performance Auto-Bidding products



Performance Auto-Bidding products



Performance Auto-Bidding products

	Goal	**Constraint**
Budget Optimizer	Clicks	Budget
Target CPA	Conversions	Avg cost-per-conversion
Other potential examples	Post-install-events	Avg cost-per-install
...	…	...

A General Framework

Should you buy the i-th click?

$$Max \sum_i x_i ctr_i v_i$$

The value for the i-th click

$$s.t. \sum_i x_i ctr_i cpc_i \leq B_c + \sum_i x_i ctr_i w_{ic} \quad \forall \; c$$

$$x_i \in \{0, 1\}$$

Constraint specific constants

Expected Spend

A General Framework

- **Budget Optimizer:**
  - v_i = 1, B = budget, w_i = 0
- **Target CPA:**
  - **v_i = pCVR, B = 0**
  - $$\frac{\sum_i x_i ctr_i cpc_i}{\sum_i x_i ctr_i cvr_i} \leq T \;\; \Rightarrow w_i = T \times cvr_i$$
- **Target CPC constraint:**
  - $$\frac{\sum_i x_i ctr_i cpc_i}{\sum_i x_i ctr_i} \leq M \;\; \Rightarrow w_i = M$$

$$Max \sum_i x_i ctr_i v_i$$

$$s.t. \sum_i x_i ctr_i cpc_i \leq B_c + \sum_i x_i ctr_i w_{ic} \quad \forall c$$

$$x_i \in \{0, 1\}$$

Optimal Bidding Algorithm

- Given the LP and all the data, including CPCs, we can solve to say which items you want to pick.
- Can a simple bidding formula lead to the same outcomes?
- Does the answer depend on the underlying auction properties?

Bidding Algorithm

- Complementary slackness conditions say that you want to take all the items with

$$cpc \leq \frac{v_i + \sum_c \alpha_c w_{ic}}{\sum_c \alpha_c}$$

- Can implement it by setting bid:

$$b(i) := \frac{v_i + \sum_c \alpha_c w_{ic}}{\sum_c \alpha_c}$$

Not entirely new, studied in various forms earlier, e.g., [Agrawal-Devanur'15]

Primal Linear Program

$$\max \sum_{i,s} x_{is} ctr_{is} v_i$$

$$\forall c, \sum_{i,s} x_{is} ctr_{is} cpc_{is} \leq B_c + \sum_{i,s} x_{is} ctr_{is} w_{ic}$$

$$\forall i, \sum_{s} x_{is} \leq 1$$

$$\forall i, s, x_{is} \geq 0$$

Dual Linear Program

$$\min \sum_{i} \delta_i + \sum_{c} \alpha_c B_c$$

$$\forall i, s, \ \delta_i + \sum_{c} \alpha_c ctr_{is}(cpc_{is} - w_{ic}) \geq ctr_{is} v_i$$

$$\forall i, \ \delta_i \geq 0$$

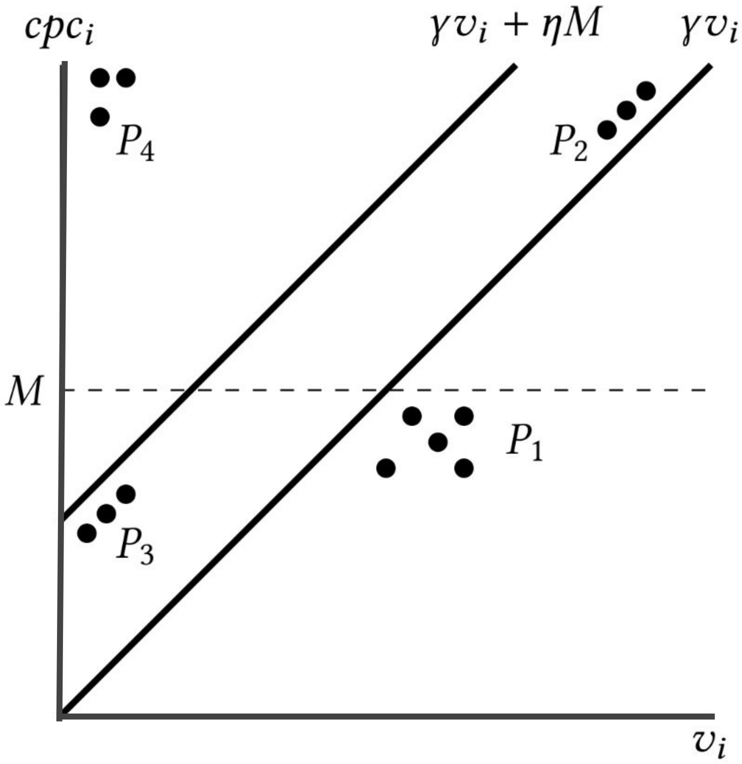
$$\forall c, \ \alpha_c \geq 0$$

Bidding Algorithm

Theorem: With the correct setting of the parameters α_c the bidding formula is optimal iff the auction is truthful.

Note: The parameters can be learned from past data and updated online.

Intuition



Target CPA + Budget

$$b(i) = \frac{v_i + \sum_c \alpha_c w_{ic}}{\sum_c \alpha_c} = \frac{v_i + \alpha \cdot T \cdot v_i}{\alpha} = \gamma v_i$$

Target CPA + Target CPC + Budget

$$b(i) = \frac{v_i + \sum_c \alpha_c w_{ic}}{\sum_c \alpha_c} = \frac{v_i + \alpha_1 \cdot T \cdot v_i + \alpha_2 \cdot M}{\alpha_1 + \alpha_2} = \gamma v_i + \eta M$$

Bidding equilibrium

- What happens when everyone adopts autobidding?
  - Is there an equilibrium?
  - Do we get good overall value in equilibrium, or can it result in bad dynamics leading to low value and revenue?

Does there exist an Equilibrium?

Not Obvious due to interactions.

Theorem: An approx equilibrium exists s.t. each bidder bids almost optimally, given what other bidders are bidding.

Proof: Using Brouwer's fixed point theorem.

$$\phi(\kappa_a) = \kappa_a \cdot (1+\epsilon)^{(\text{Target CPA} - \text{Realized CPA})}$$

Performance in equilibrium: Price of Anarchy

Efficiency == Weighted sum of advertiser goals

E.g., for tCPA: $$Efficiency = \sum_a tCPA(a) \times Conversions(a)$$

(total value of conversions)

GLOBAL OPT: Give q to ad with highest tCPA * pcvr
(and charge first price / for free).

Price of Anarchy

How much value do we lose by allowing one agent per bidder?

$$POA = \underset{instances}{Max} \quad \frac{Efficiency(OPT)}{Efficiency(equilibrium)}$$

Theorem: For the general autobidding problem, POA = 2.

You do not lose more than 50% value in the worst case, and there are instances in which you could lose 50%.

Due to multiple constraints (e.g., budgets), we use the ”Liquid POA” definition.

Proof Idea

A := Queries s.t.
Equilibrium-Ad = OPT-Ad

Equilibrium >= OPT(A)

B := Queries s.t.
Equilibrium-Ad =/= OPT-Ad

Use:
- Second-price auction
- Bid >= tCPA

Equilibrium >= OPT(B)

2 * Equilibrium >= OPT

Questions?