Algorithms for Data Science
Web Advertising

Silviu Maniu

October 16th, 2020

Université Paris-Saclay

Table of contents

First iteration: **banner ads** (around 1995)

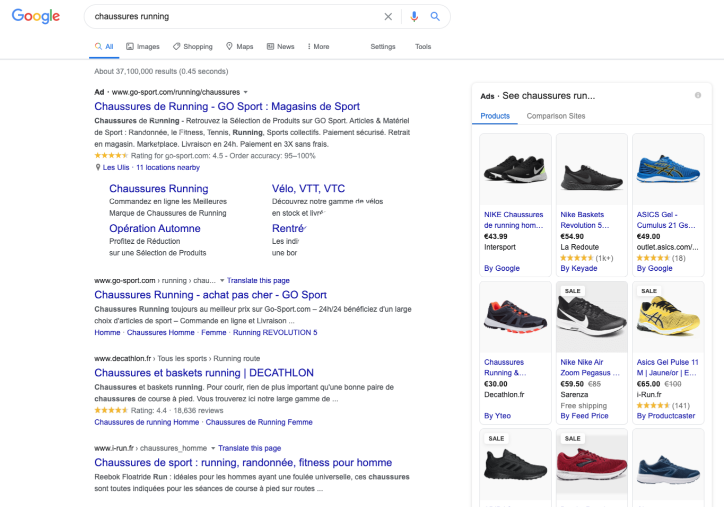
Banner Ads

First iteration: **banner ads** (around 1995)

- charging per $1,000$ "impressions" (**clicks**)
- **CPM** – cost per thousand impressions (as in TV, print media)
- **untargeted** vs. **demographically targeted**
- low **click through rates** – low return on investment

Performance-Based Advertising

Second iteration: ads on search results (around 2001)



Performance-Based Advertising

Second iteration: **ads on search results** (around 2001)

- advertisers **bid** on **search keywords**
- on **click** – highest bidder ad is shown
- charging only if add is clicked
- adopted by Google around 2002 – **Adwords**

Performance-Based Advertising

Part of Web 2.0 – huge industry (several billion $)

Problem: what ads to show for a given query

- another related problem: which search terms should an advertiser bid on, and for how much
- part of computational game theory

Table of contents

Online Algorithms

Data Streams: limited resources to process data as it comes

Online algorithms

- decision must be made **immediately** as data comes
- vs. **offline** – data is processed in its entirety

Greedy Algorithm for Online Optimization Problems

Optimization problem: maximizing or minimizing an objective function on the data

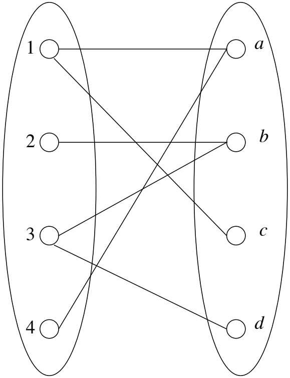
Greedy algorithm: take decision locally, by optimizing only based on the current element and the past

Not always optimal vs. offline algorithsm:

- **competitive ratio**: the ratio between the offline solution and the online solution over all inputs $c = \min_G \frac{|M_g|}{|M_o|}$

Matching Problem

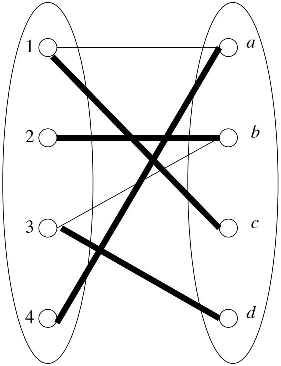
Bipartite Graph: a graph $G(V_1 \cup V_2, E)$ having two disjoint sets of nodes V_1 and V_2 and edges only havins one endpoint in V_1 and one in V_2, i.e., $E \subseteq V_1 \times V_2$



Matching Problem

Matching: choosing a subset of the edges in the bipartite graph s.t. no node has more than two edges in the matching

- perfect – every node is in the matching
- maximal – has the largest number of edges possible



Greedy Algorithm for Matching

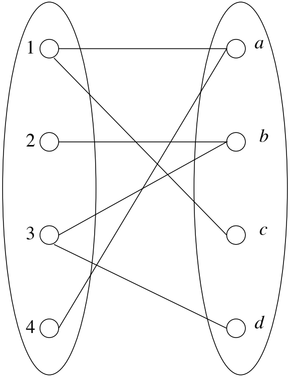
Offline case: algorithms for finding maximal matchings are $\mathcal{O}(n^2)$, where $n = |E|$

Online case: can use the **greedy** algorithm:

1. consider the edges in the order they arrive
2. add edge (x, y) only if neither x nor y are endpoints

Example of Greedy Matching

Edges arrive in the order: $(1, a)$, $(1, c)$, $(2, b)$, $(3, b)$, $(3, d)$, $(4, a)$



Result of greedy matching: $(1, a)$, $(2, b)$, $(3, d)$ – not maximal

Competitive Ratio of Greedy Matching

M_o – maximal matching, M_g – greedy matching

L – left nodes matched in M_o but not in M_g

R – right nodes connected to any node in L

Claim: every node in R is matched in M_g

- prove by contradiction: assume it is not the case
- then there will exist edge (l, r), $l \in L$
- then, it should be matched (neither is added to the matching)
- contradiction!

Competitive Ratio of Greedy Matching

Claim: every node in R is matched in M_g

- $|M_o| \leq |M_g| + |L|$ – only nodes in L can be matched in M_o
- $|L| \leq |R|$ – in M_o, all nodes in L are matched
- $|R| \leq |M_g|$ – every node in R in mateched in M_g

– this gives us $|M_g| \geq \frac{|M_o|}{2}$ – **lower bound** on the competitive ratio

But $1/2$ is also an **upper bound** – can find a counter example

Competitive ratio is then exactly $1/2$

Table of contents

Adwords Problem

Problem: match queries in a search engines with advertisers

We have:

- a set of **bids** by advertisers for search queries
- **click-through rate** for each advertiser-query pair
- **budget** for each advertiser (time, money, etc.)
- **limit on the number of ads** to be displayed

Adwords Problem

Problem: match queries in a search engines with advertisers

Restrictions on the set of advertisers:

- the size is under the limit of number of ads
- each advertiser in the set has bid on the query
- each advertiser has enough budget left over

Adwords Setting

1. **stream of queries** arrives at search engines $q_1, q_2, \ldots$
2. advertisers **bid** on each query
3. when q_i arrives search engine picks a subset of advertisers

Objective: **maximize search engine revenue**

If we consider queries as being the “left” side and advertisers the “right” side in a bipartite graph – **online bipartite matching**

- **weighted case**: the matching depends on the CTR and the budget

Adwords in Practice

In practice: combine CTR and bid – **expected revenue**

- **value** of an ad – expected revenue
- **revenue** to the search engine – sum of values of matched ads

Advertiser	CTR	Bid	CTR $\times$ Bid
A	0.02	7.5	0.15
B	0.05	5.0	0.25
C	0.01	1.0	0.01

Measuring CTR

Value of an ad is directly linked to the CTR rate

- high bid is useless if the CTR is very low

Click-through rate is measured historically – **difficult problem**

- **explore**: do we try an ad to measure the CTR rate for future campaigns?
- **exploit**: do we use the current known CTR rate, even if they could be outdated?

Greedy Algorithm

Setting:

- there is one ad shown for each query
- all advertisers have the same budget B
- all ads have same CTR
- value is then the same

Greedy algorithm:

- pick any advertiser who has bid for that query
- same competitive ratio as in online matching – $1/2$

Worst-case Greedy

Advertiser A: bids on query x, budget 4 Advertiser B: bids on queries x and y, budget 4

Stream: $x\ x\ x\ x\ y\ y\ y\ y$

Greedy choice:

- worst case: $B\ B\ B\ B\ .\ .\ .\ .$
- optimal: $A\ A\ A\ A\ B\ B\ B\ B$
- competitive ratio: $1/2$

BALANCE algorithm [Mehta et al., 2007]

BALANCE Algorithm:

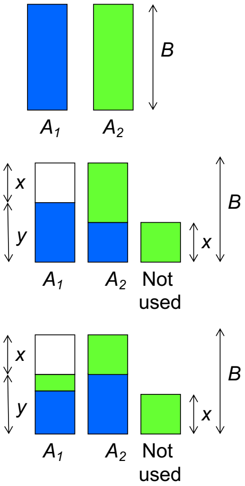
- assign query to the bidder having the most budget left
- **competitive ratio** $3/4$
- **tie breaking**: must be deterministic

Previous example:

- if A is preferred to B: $A\ B\ A\ B\ B\ B\ \ldots$
- establishes an **upper bound** on competitive ratio for 2 bidders

BALANCE – Lower Bound for 2 Bidders

Assumption: advertisers A_1, A_2 budget B (consumed by the optimal algo), revenue $2B$



BALANCE must **exhaust the budget of at least one bidder**, e.g., A_2
Case of assigned bids ($x + y = B$):

- **at least half of the queries are assigned to A_1**: $y \geq B/2$, so $y \geq x$
- **more than half of the queries are assigned to A_2**: remaining budget of A_2 is less than $B/2$, so $x \leq B/2$, so $y \geq x$

Minimal BALANCE revenue at $x = y = B/2$, revenue $3B/2$ **competitive ratio** $\frac{3B/2}{2B} = 3/4$

BALANCE – Multiple Bidders

In the general case, BALANCE competitive ratio is not much lower than the simple case:

- **competitive ratio**: $1 - 1/e = 0.63\ldots$
- **no online algorithm has a better competitve ratio**

BALANCE – Worst Case for Multiple Bidders

Advertisers: N – $A_1, \ldots, A_N$, each having budget $B > N$

Queries: N rounds of B queries

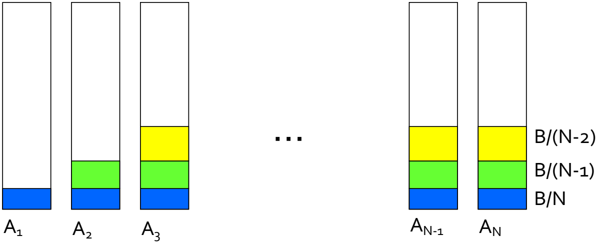
Bids: round i - bidders $A_i, \ldots, A_n$

Optimum allocation: allocate round i queries to A_i

- revenue $N \cdot B$

BALANCE – Worst Case for Multiple Bidders

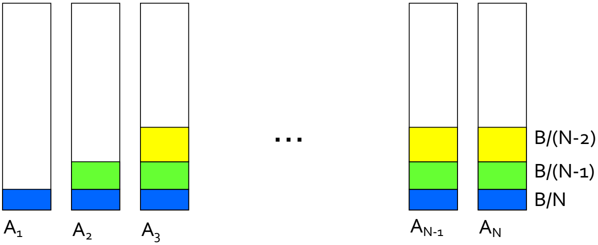
BALANCE allocation



- BALANCE assigns queries in round k to $N-(k-1)$ advertisers
- sum of allocation to each advertiser $A_k, \ldots, A_N$: $S_k = \sum_{i=1}^{k-1} \frac{B}{N-(i-1)}$
- the smallest k at which $S_k \geq B$ is the point after which no advertisers can be allocated $k = N(1-1/e)$

BALANCE – Worst Case for Multiple Bidders

BALANCE allocation



- after $k = N(1 - 1/e)$ we cannot get any revenue
- **total revenue**: $B \cdot N(1 - 1/e)$
- upper bound on **competitive ratio**: $1 - 1/e$

BALANCE with Arbitrary Bids

BALANCE works well when bids are 0 or 1

- if arbitrary bids, it can fail and have competitive ratio 0

Example:

- advertisers A_1, A_2, one query q arriving 10 times
- A_1: bids 1, budget 110
- A_2: bids 10, budget 100
- optimal: assign all queries to A_2, revenue 100
- BALANCE: assigns all queries to A_1, revenue 10

Generalized BALANCE

BALANCE can be generalized to arbitrary bids:

- bid x_i, budget b_i, amount spend so far m_i
- fraction of leftover budget $f_i = 1 - m_i/b_i$
- for a query q, use $\psi_i(q) = x_i(1 - e^{-f_i})$

Decision:

- allocate query q to bidder i having largest value of $\psi_i(q)$

Same competitive ratio: $1 - 1/e$

Adwords Implementation

In practice

- advertisers **bid of sets of words**
- if a search query contains exactly those words – the advertiser becomes a bidder
- can use **distributed hash tables**
- queries can be distributed on **several machines** also – multiple streams

Another applications:

- Google also matches **ads to emails** – much harder problem (mails are much larger)

Acknowledgments

The contents follows Chapter 8 of [Leskovec et al., 2020]. Figures in slides 11, 12, 14, 26, 29, and 30 are taken from `https://www.mmds.org/`

References i

Leskovec, J., Rajaraman, A., and Ullman, J. (2020).
Mining of Massive Datasets.
Cambridge University Press.

Mehta, A., Saberi, A., Vazirani, U., and Vazirani, V. (2007).
Adwords and generalized online matching.
J. ACM, 54(5):22–es.