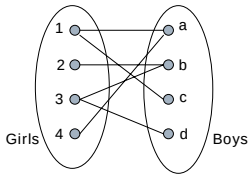
CS 345
Data Mining

Online algorithms
Search advertising

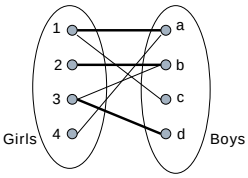
Online algorithms

- Classic model of algorithms
  - You get to see the entire input, then compute some function of it
  - In this context, "offline algorithm"
- Online algorithm
  - You get to see the input one piece at a time, and need to make irrevocable decisions along the way
- Similar to data stream models

Example: Bipartite matching

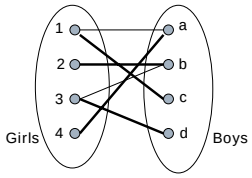


Example: Bipartite matching



M = {(1,a),(2,b),(3,d)} is a matching
Cardinality of matching = |M| = 3

Example: Bipartite matching



M = {(1,c),(2,b),(3,d),(4,a)} is a perfect matching

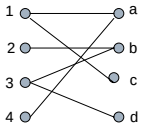
Matching Algorithm

- Problem: Find a maximum-cardinality matching for a given bipartite graph
  - A perfect one if it exists
- There is a polynomial-time offline algorithm (Hopcroft and Karp 1973)
- But what if we don't have the entire graph upfront?

Online problem

- Initially, we are given the set Boys
- In each round, one girl's choices are revealed
- At that time, we have to decide to either:
  - Pair the girl with a boy
  - Don't pair the girl with any boy
- Example of application: assigning tasks to servers

Online problem



(1,a)
(2,b)
(3,d)

Greedy algorithm

- Pair the new girl with any eligible boy
  - If there is none, don't pair girl
- How good is the algorithm?

Competitive Ratio

- For input I, suppose greedy produces matching M_{greedy} while an optimal matching is M_{opt}

Competitive ratio =

$$\min_{\text{all possible inputs I}} (|M_{greedy}|/|M_{opt}|)$$

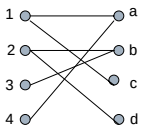
Analyzing the greedy algorithm

- Consider the set G of girls matched in M_{opt} but not in M_{greedy}
- Then it must be the case that every boy adjacent to girls in G is already matched in M_{greedy}
- There must be at least |G| such boys
  - Otherwise the optimal algorithm could not have matched all the G girls
- Therefore

$|M_{greedy}| \geq |G| = |M_{opt} - M_{greedy}|$

$|M_{greedy}|/|M_{opt}| \geq 1/2$

Worst-case scenario



(1,a)
(2,b)

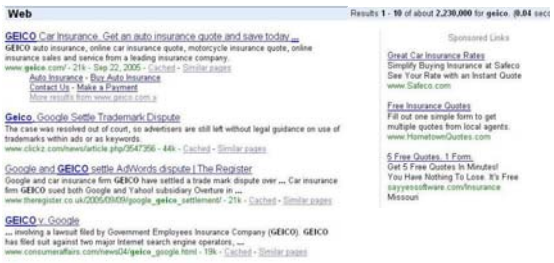
History of web advertising

- Banner ads (1995-2001)
  - Initial form of web advertising
  - Popular websites charged X$ for every 1000 “impressions” of ad
    - Called “CPM” rate
    - Modeled similar to TV, magazine ads
  - Untargeted to demographically tageted
  - Low clickthrough rates
    - low ROI for advertisers

Performance-based advertising

- Introduced by Overture around 2000
  - Advertisers “bid” on search keywords
  - When someone searches for that keyword, the highest bidder’s ad is shown
  - Advertiser is charged only if the ad is clicked on
- Similar model adopted by Google with some changes around 2002
  - Called “Adwords”

Ads vs. search results



Web 2.0

- Performance-based advertising works!
  - Multi-billion-dollar industry
- Interesting problems
  - What ads to show for a search?
  - If I’m an advertiser, which search terms should I bid on and how much to bid?

Adwords problem

- A stream of queries arrives at the search engine
  - q1, q2,…
- Several advertisers bid on each query
- When query q_i arrives, search engine must pick a subset of advertisers whose ads are shown
- Goal: maximize search engine’s revenues
- Clearly we need an online algorithm!

Greedy algorithm

- Simplest algorithm is greedy
- It’s easy to see that the greedy algorithm is actually optimal!

Complications (1)

- Each ad has a different likelihood of being clicked
  - Advertiser 1 bids $2, click probability = 0.1
  - Advertiser 2 bids $1, click probability = 0.5
  - Clickthrough rate measured historically
- Simple solution
  - Instead of raw bids, use the "expected revenue per click"

Complications (2)

- Each advertiser has a limited budget
  - Search engine guarantees that the advertiser will not be charged more than their daily budget

Simplified model (for now)

- Assume all bids are 0 or 1
- Each advertiser has the same budget B
- One advertiser per query
- Let's try the greedy algorithm
  - Arbitrarily pick an eligible advertiser for each keyword

Bad scenario for greedy

- Two advertisers A and B
- A bids on query x, B bids on x and y
- Both have budgets of $4
- Query stream: xxxxyyyy
  - Worst case greedy choice: BBBB____
  - Optimal: AAAABBBB
  - Competitive ratio = ½
- Simple analysis shows this is the worst case

BALANCE algorithm [MSVV]

- [Mehta, Saberi, Vazirani, and Vazirani]
- For each query, pick the advertiser with the largest unspent budget
  - Break ties arbitrarily

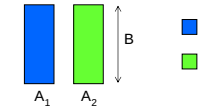
Example: BALANCE

- Two advertisers A and B
- A bids on query x, B bids on x and y
- Both have budgets of $4
- Query stream: xxxxyyyy
- BALANCE choice: ABABBB__
  - Optimal: AAAABBBB
- Competitive ratio = ¾

Analyzing BALANCE

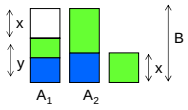
- Consider simple case: two advertisers, A_1 and A_2, each with budget B (assume $B \gg 1$)
- Assume optimal solution exhausts both advertisers' budgets
- BALANCE must exhaust at least one advertiser's budget
  - If not, we can allocate more queries
  - Assume BALANCE exhausts A_2's budget

Analyzing Balance



Queries allocated to A_1 in optimal solution

Queries allocated to A_2 in optimal solution



Opt revenue = 2B
Balance revenue = 2B-x = B+y
We have $y \geq x$
Balance revenue is minimum for x=y=B/2
Minimum Balance revenue = 3B/2
Competitive Ratio = 3/4

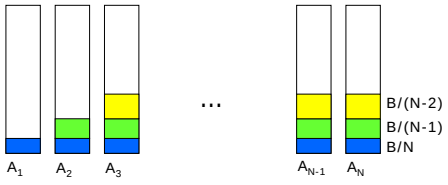
General Result

- In the general case, worst competitive ratio of BALANCE is 1–1/e = approx. 0.63
- Interestingly, no online algorithm has a better competitive ratio
- Won't go through the details here, but let's see the worst case that gives this ratio

Worst case for BALANCE

- N advertisers, each with budget $B \gg N \gg 1$
- NB queries appear in N rounds of B queries each
- Round 1 queries: bidders A_1, A_2, …, A_N
- Round 2 queries: bidders A_2, A_3, …, A_N
- Round i queries: bidders A_i, …, A_N
- Optimum allocation: allocate round i queries to A_i
  - Optimum revenue NB

BALANCE allocation



After k rounds, sum of allocations to each of bins $A_k,\ldots,A_N$ is $S_k = S_{k+1} = \ldots = S_N = \sum_{1..1..k} B/(N-i+1)$

If we find the smallest k such that $S_k \geq B$, then after k rounds we cannot allocate any queries to any advertiser

BALANCE analysis

B/1 B/2 B/3 … B/(N-k+1) … B/(N-1) B/N

S_1

S_2

$S_k = B$

1/1 1/2 1/3 … 1/(N-k+1) … 1/(N-1) 1/N

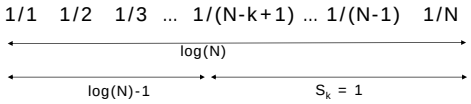
S_1

S_2

$S_k = 1$

BALANCE analysis

- Fact: $H_n = \sum_{1 \cdot i \cdot n} 1/i$ = approx. log(n) for large n
  - Result due to Euler



$S_k = 1$ implies $H_{N-k} = \log(N)-1 = \log(N/e)$
N-k = N/e
k = N(1-1/e)

BALANCE analysis

- So after the first N(1-1/e) rounds, we cannot allocate a query to any advertiser
- Revenue = BN(1-1/e)
- Competitive ratio = 1-1/e

General version of problem

- Arbitrary bids, budgets
- Consider query q, advertiser i
  - Bid = x_i
  - Budget = b_i
- BALANCE can be terrible
  - Consider two advertisers A_1 and A_2
  - A_1: $x_1 = 1$, $b_1 = 110$
  - A_2: $x_2 = 10$, $b_2 = 100$

Generalized BALANCE

- Arbitrary bids; consider query q, bidder i
  - Bid = x_i
  - Budget = b_i
  - Amount spent so far = m_i
  - Fraction of budget left over $f_i = 1-m_i/b_i$
  - Define $\psi_i(q) = x_i(1-e^{-f_i})$
- Allocate query q to bidder i with largest value of $\psi_i(q)$
- Same competitive ratio (1-1/e)