# COMP6207
# Algorithmic Game Theory

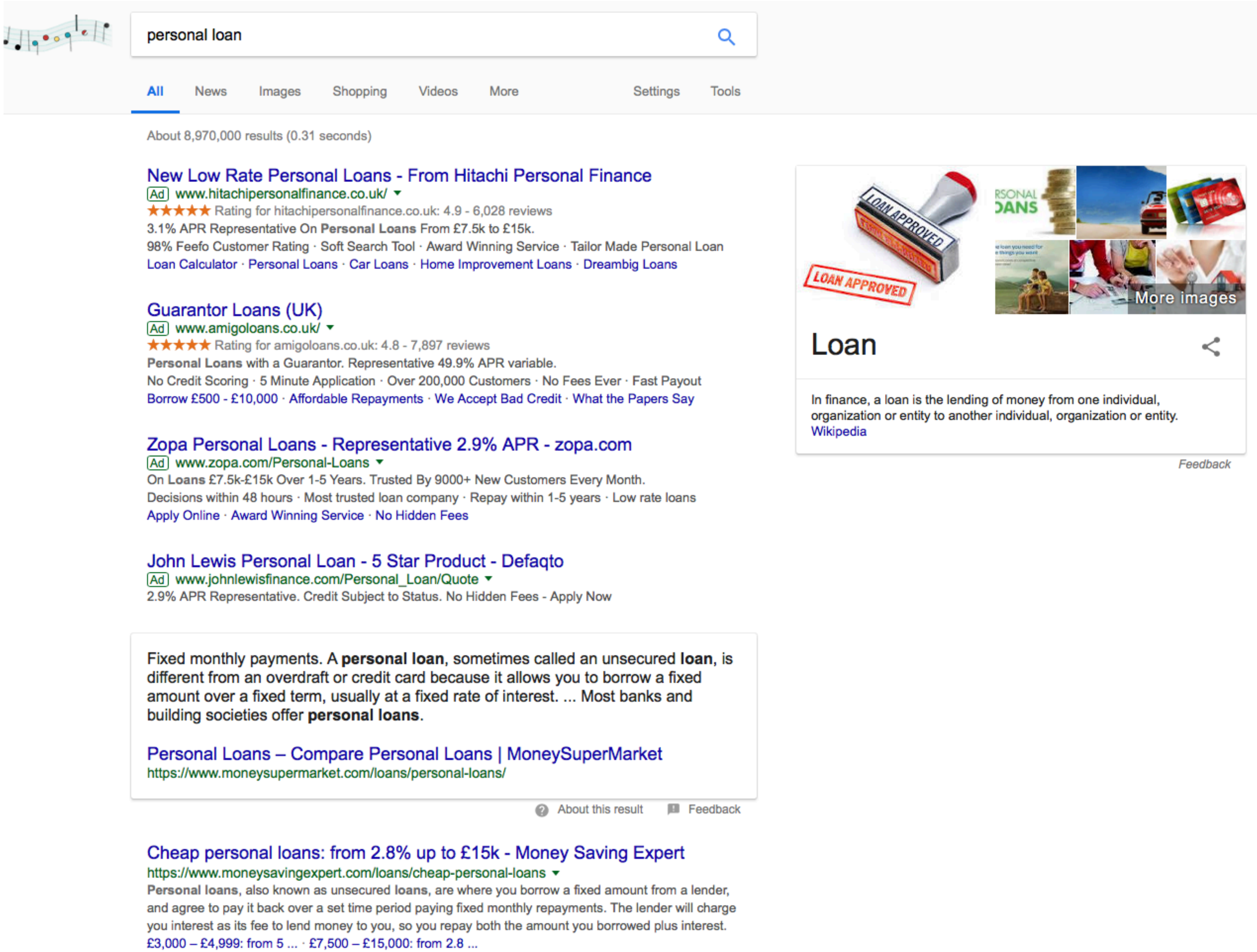
Lecture 17
Sponsored Search Auctions

Pavel Naumov
p.naumov@soton.ac.uk
Electronics and Computer Science
University of Southampton

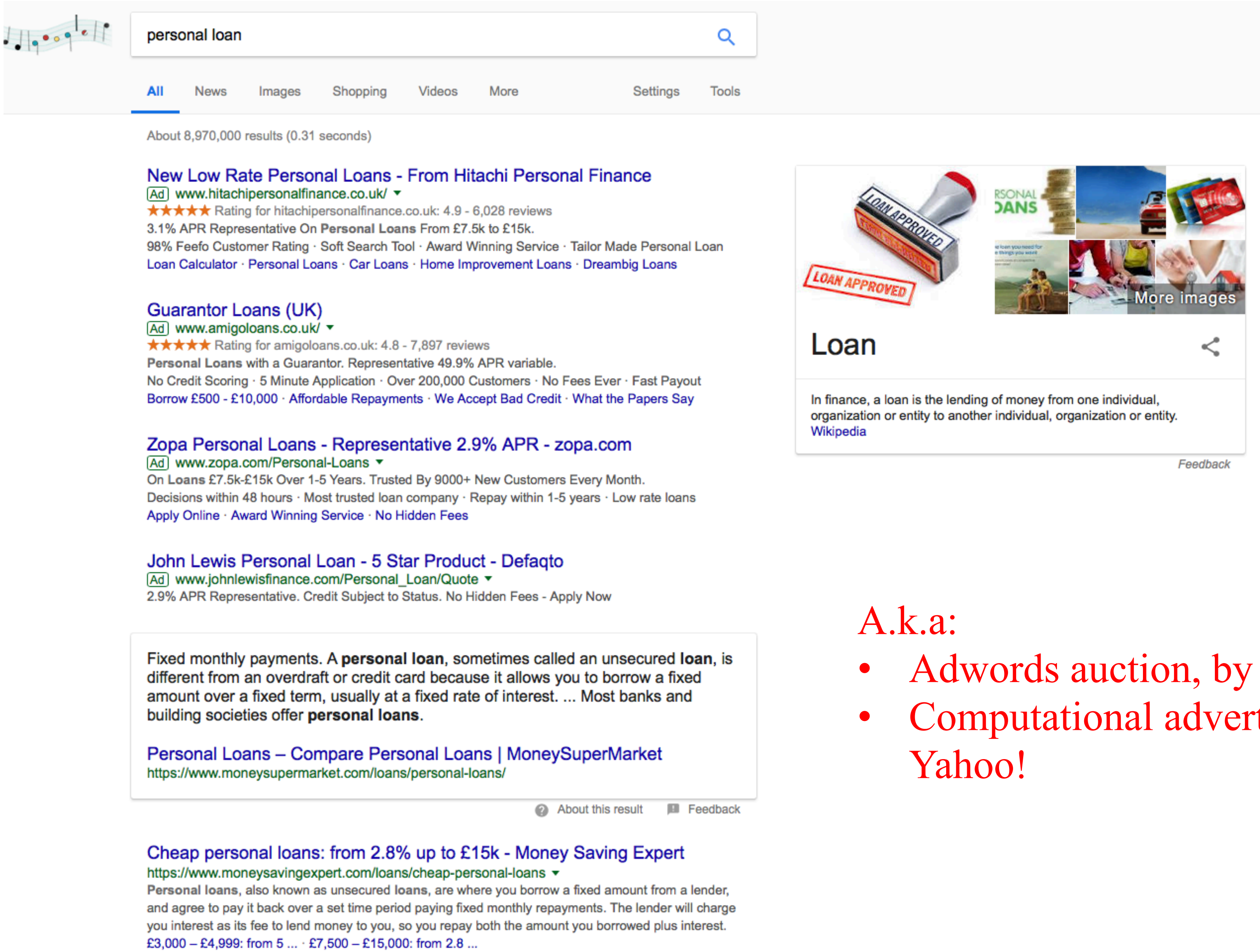
# Learning Outcomes

- By the end of this session, the students should be able to
  - ***Describe*** the search engine's ad slot auction model
  - ***Compare*** the practical model with several theoretical models covered in previous sessions
  - ***Evaluate*** the best matching algorithm for search engines.

# Sponsored Search Auctions

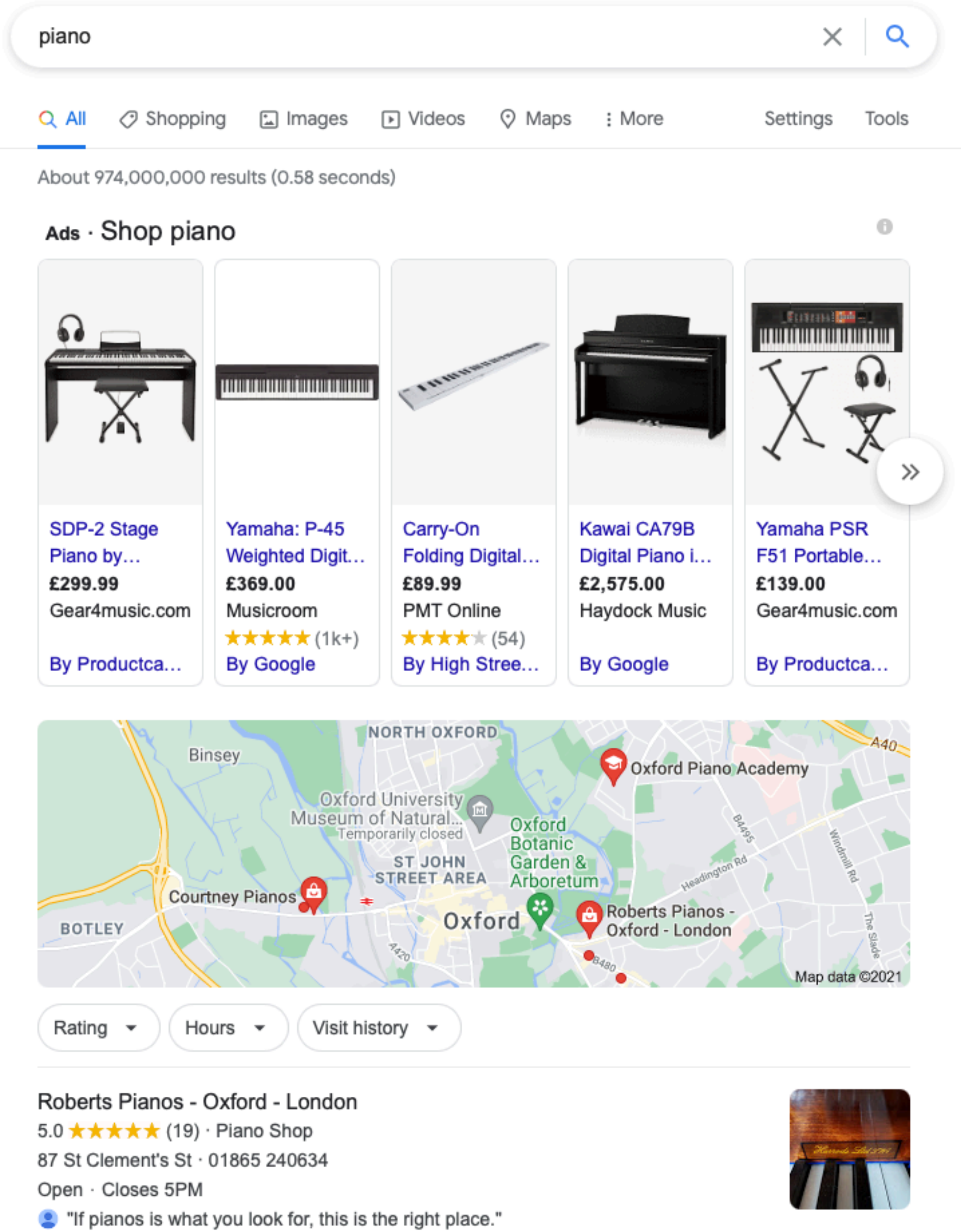


# Sponsored Search Auctions




A.k.a:

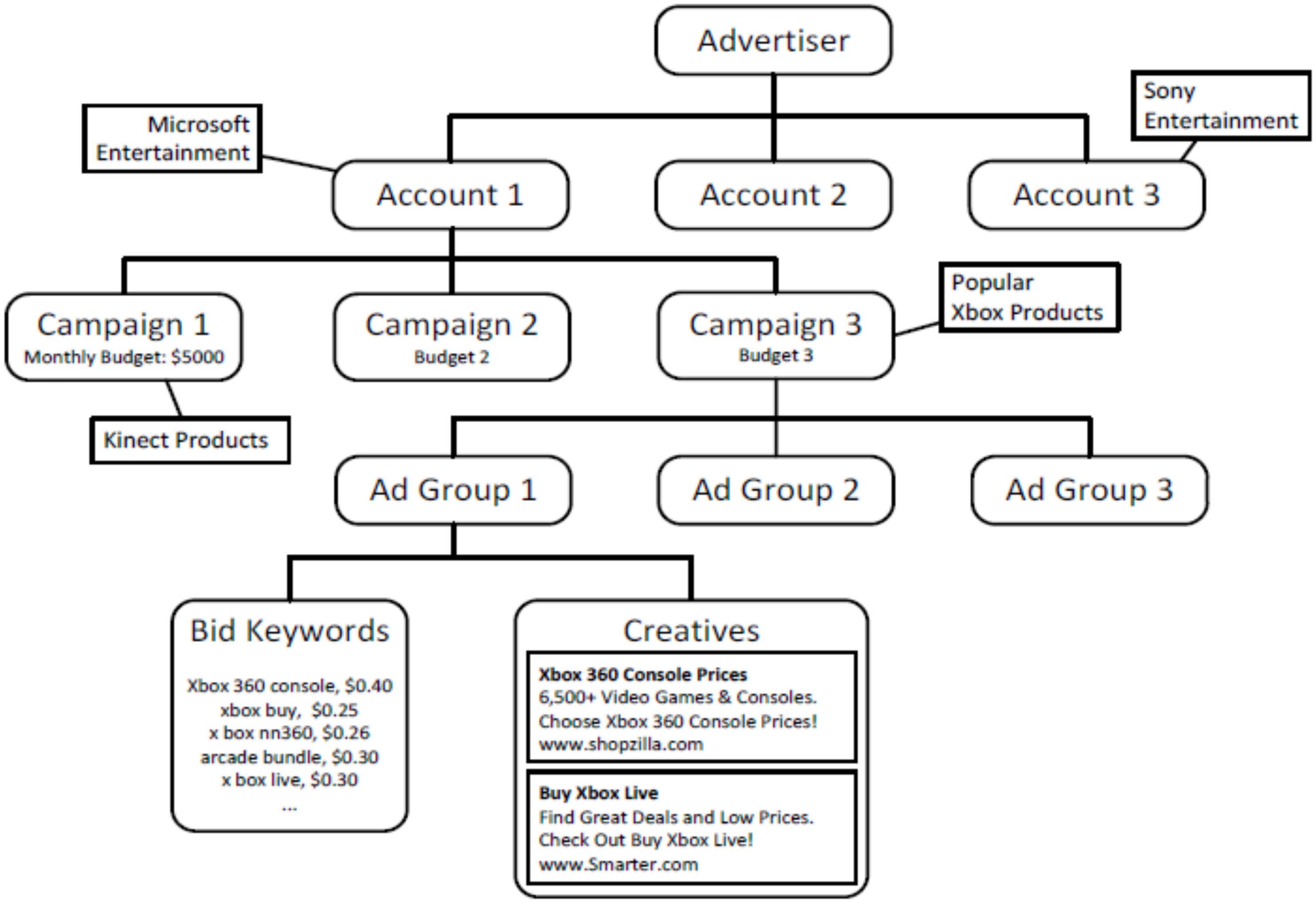
- Adwords auction, by Google
- Computational advertising, by Yahoo!


piano
All
Shopping
Images
Videos
Maps
More
Settings
Tools
About 974,000,000 results (0.58 seconds)
Ads · Shop piano
SDP-2 Stage Piano by…
£299.99
Gear4music.com
By Productca…
Yamaha: P-45 Weighted Digit…
£369.00
Musicroom
(1k+)
By Google
Carry-On Folding Digital…
£89.99
PMT Online
(54)
By High Stree…
Kawai CA79B Digital Piano i…
£2,575.00
Haydock Music
By Google
Yamaha PSR F51 Portable…
£139.00
Gear4music.com
By Productca…
NORTH OXFORD
Binsey
Oxford Piano Academy
A40
Oxford University Museum of Natural…
Temporarily closed
Oxford Botanic Garden & Arboretum
ST JOHN STREET AREA
Courtney Pianos
BOTLEY
Oxford
Roberts Pianos - Oxford - London
Headington Rd
Windmill Rd
The Slade
B4495
A420
B480
Map data ©2021
Rating
Hours
Visit history
Roberts Pianos - Oxford - London
5.0 (19) · Piano Shop
87 St Clement's St · 01865 240634
Open · Closes 5PM
"If pianos is what you look for, this is the right place."

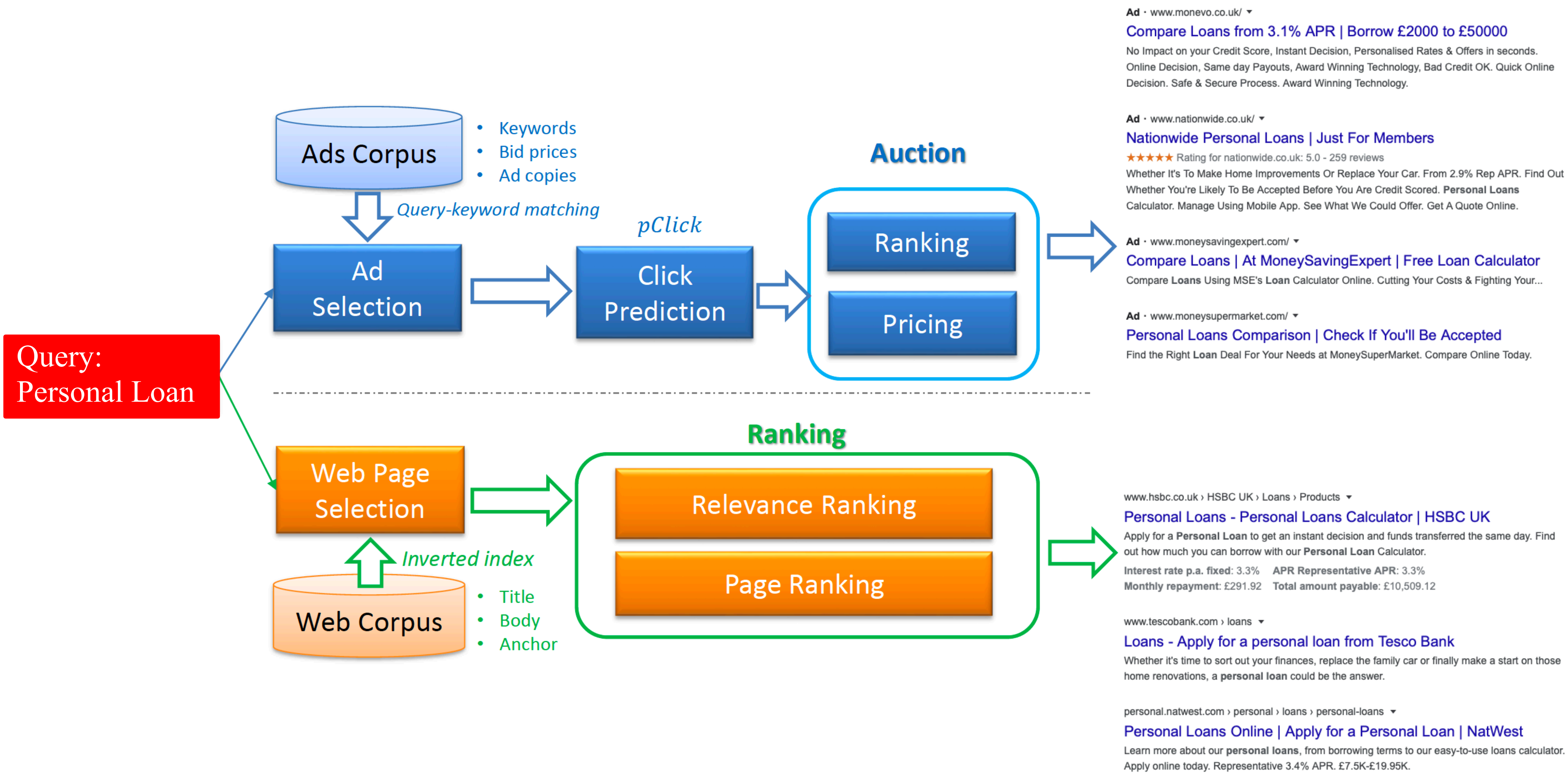
# Advertiser/bidder's Perspective

- Has a set of interested keywords
- Has a private value for each of the keywords
- Has a private daily/weekly/monthly budget
- Use their budget to buy clicks
- Obj: how to maximize utility, subject to budget?

# Advertiser's Perspective



# Search Engine's Perspective



# Search Engine

- Products of Search Engines
  - $k$ slots to sell
  - Each slot can generate a certain number of clicks

- How to match advertisers with slots?
- How to charge advertisers?
- Obj: how to maximize revenue?

# Payment Models

- Per-per-1000 impressions (per-per-mille or PPM/ PPI)
  - An "impression" is when the ad is displayed
  - Based on traditional advertising in printed media
  - Used by search engines in early days (90s)
- **Pay-per-click**
  - Only pay when clicked
  - Nowadays used by all search engines
- Pay-per-conversion
  - Only pay when actual purchase is made
  - Need to track, and attribution problem

# Why auctions?

# Why auctions?

- In online advertising
  - A large number of advertisers
  - Rich user information and context
  - Value of an impression/click varies largely across different advertisers

# Why auctions?

- In online advertising
  - A large number of advertisers
  - Rich user information and context
  - Value of an impression/click varies largely across different advertisers
- Pricing is based on the value to the clients/advertisers, not the cost to SE

# Why auctions?

- In online advertising
  - A large number of advertisers
  - Rich user information and context
  - Value of an impression/click varies largely across different advertisers
- Pricing is based on the value to the clients/advertisers, not the cost to SE
- Price differentiation is the key enabler of profit

# Why auctions?

- In online advertising
  - A large number of advertisers
  - Rich user information and context
  - Value of an impression/click varies largely across different advertisers
- Pricing is based on the value to the clients/advertisers, not the cost to SE
- Price differentiation is the key enabler of profit
- Essentially a *price discovery mechanism*

# Why auctions?

- In online advertising
  - A large number of advertisers
  - Rich user information and context
  - Value of an impression/click varies largely across different advertisers
- Pricing is based on the value to the clients/advertisers, not the cost to SE
- Price differentiation is the key enabler of profit
- Essentially a *price discovery mechanism*
- Auctions provide efficiency and revenue benefits over fixed prices, for markets where goods are hard for a seller to value.

# How to maximise revenue?

- Objective: design online matching algorithm with good competitive ratio

# How to maximise revenue?

- Objective: design online matching algorithm with good competitive ratio

- For a moment let's consider a **simplified version**:
  - All advertisers have the same budget
  - There is only one ad slot

# How to maximise revenue?

- Objective: design online matching algorithm with good competitive ratio

- For a moment let's consider a **simplified version**:
  - All advertisers have the same budget
  - There is only one ad slot
  - Once a query comes in and the search engine matches it to a bidder, a payment is collected
  - When a bidder is matched with a query, she pays what she bids

# How to maximise revenue?

- Objective: design online matching algorithm with good competitive ratio

- For a moment let's consider a **simplified version**:
  - All advertisers have the same budget
  - There is only one ad slot
  - Once a query comes in and the search engine matches it to a bidder, a payment is collected
  - When a bidder is matched with a query, she pays what she bids
  - When a bidders' budget is exhausted, she retreats from the market

# How to maximise revenue?

- Objective: design online matching algorithm with good competitive ratio

- For a moment let's consider a **simplified version**:
  - All advertisers have the same budget
  - There is only one ad slot
  - Once a query comes in and the search engine matches it to a bidder, a payment is collected
  - When a bidder is matched with a query, she pays what she bids
  - When a bidders' budget is exhausted, she retreats from the market
  - No 2nd price; no concern about DS truthfulness; integral matching; once matched, payment is collected

# Online Matching

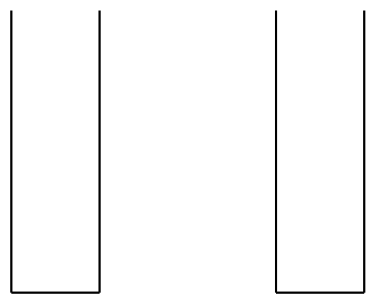
Matching advertisers

| Queries | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £0.99 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = B_2 = £100$

Queries:

100 Pepsi 100 Fanta



Bidder 1 Bidder 2

# Online Matching

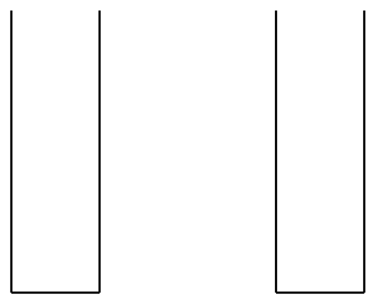
- Greedy: Always assign queries to the highest bidder

Matching advertisers

| Queries | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £0.99 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = B_2 = £100$

Queries:
100 Pepsi 100 Fanta



Bidder 1 Bidder 2

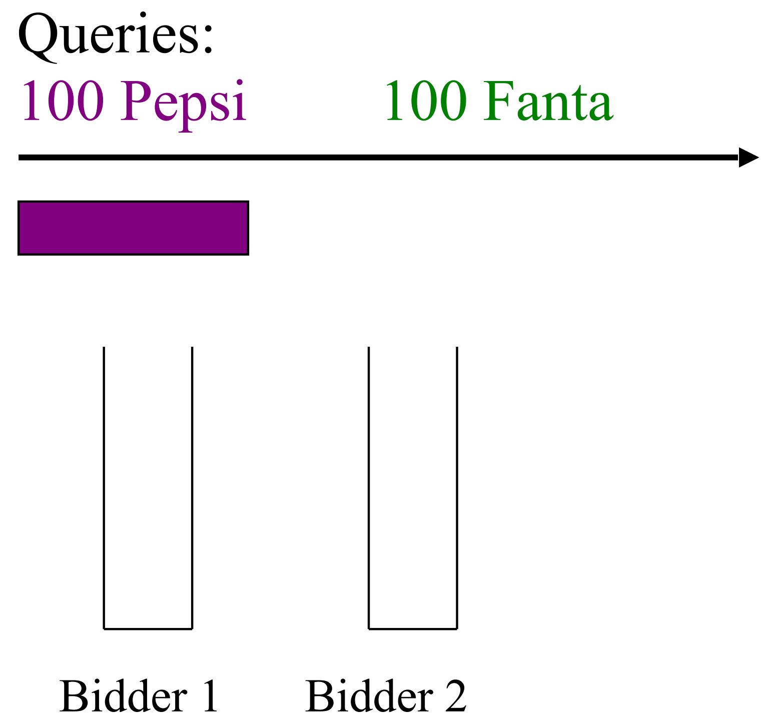
# Online Matching

- Greedy: Always assign queries to the highest bidder

Matching advertisers

| Queries | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £0.99 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = B_2 = £100$



# Online Matching

- Greedy: Always assign queries to the highest bidder

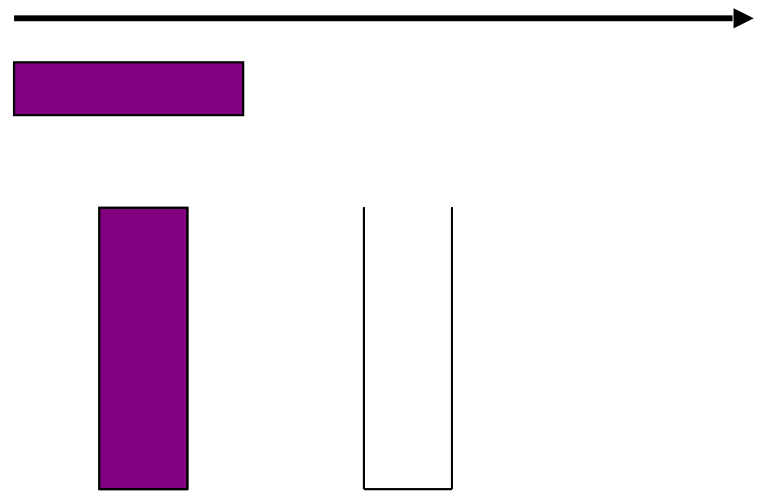
Matching advertisers

| Queries | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £0.99 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = B_2 = £100$

Queries:

100 Pepsi 100 Fanta



Bidder 1 Bidder 2

# Online Matching

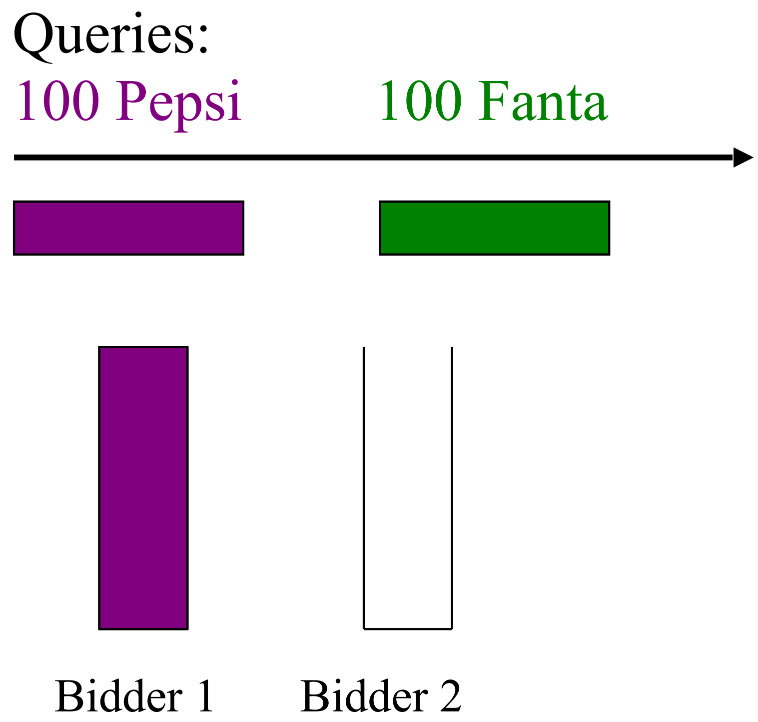
- Greedy: Always assign queries to the highest bidder

Matching advertisers

Queries

| | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £0.99 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = B_2 = £100$



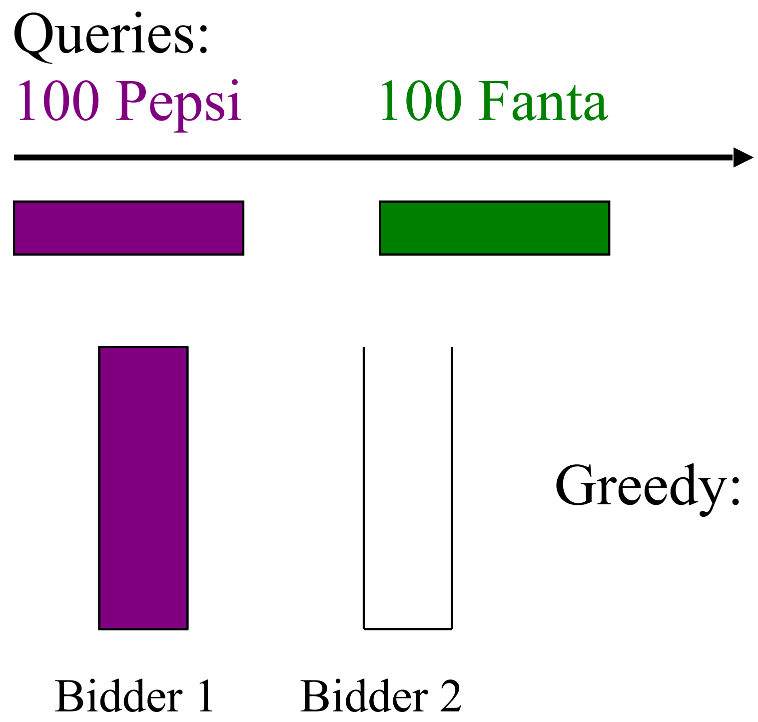
# Online Matching

- Greedy: Always assign queries to the highest bidder

Matching advertisers

| Queries | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £0.99 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = B_2 = £100$




Greedy: £100

# Competitive ratio of Greedy algorithm

Matching advertisers

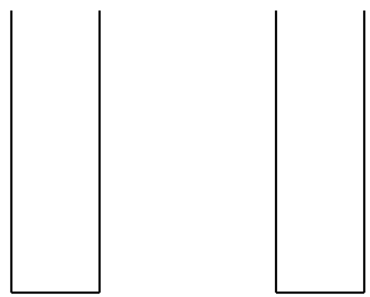
Queries

| | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £0.99 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = B_2 = £100$

Queries:

100 Pepsi 100 Fanta



Bidder 1 Bidder 2

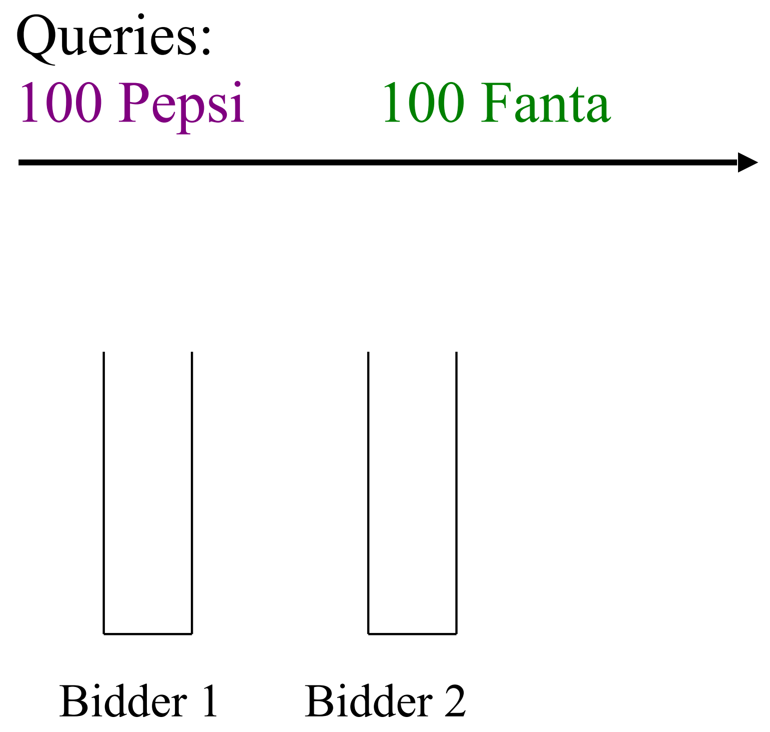
# Competitive ratio of Greedy algorithm

## Offline Optimal: hindsight matching decisions

Matching advertisers

| Queries | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £0.99 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = B_2 = £100$



# Competitive ratio of Greedy algorithm

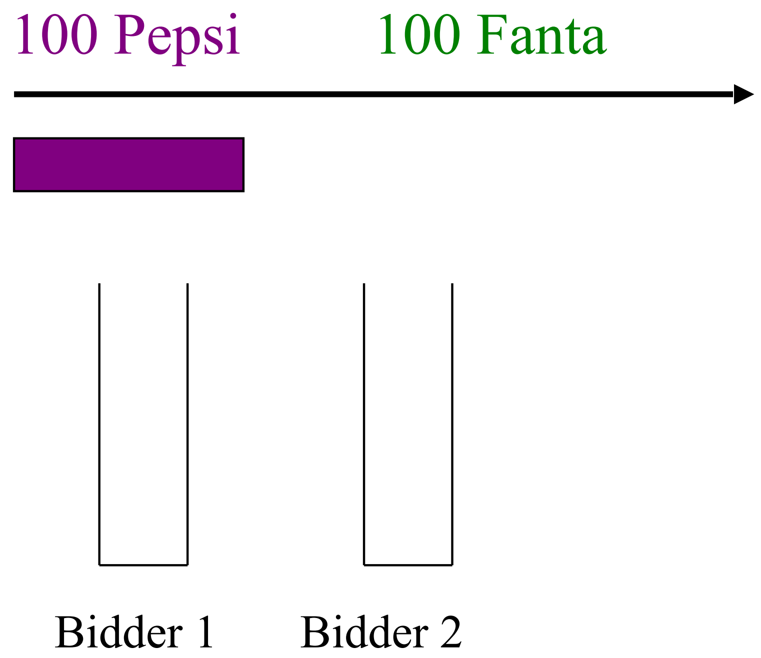
## Offline Optimal: hindsight matching decisions

Matching advertisers

| Queries | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £0.99 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = B_2 = £100$

Queries:



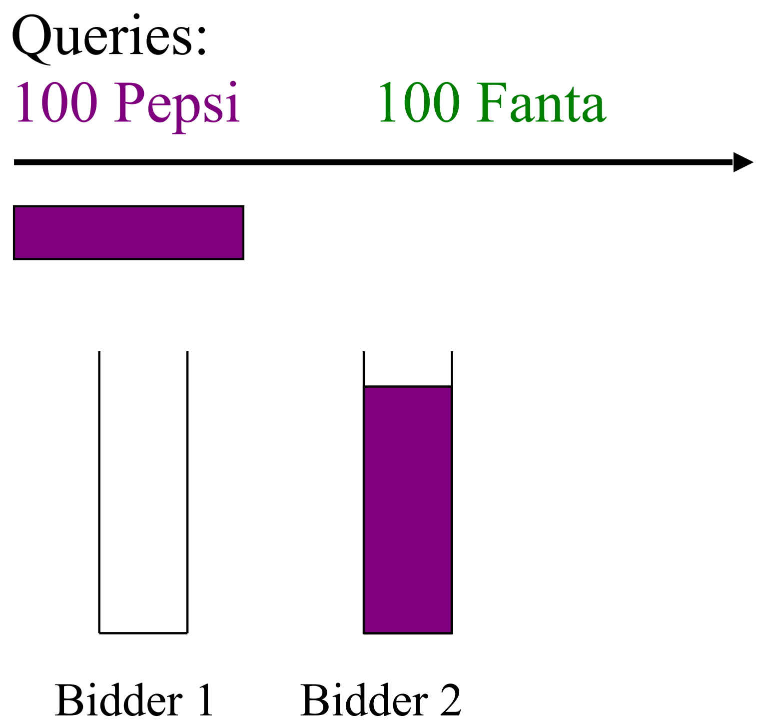
# Competitive ratio of Greedy algorithm

## Offline Optimal: hindsight matching decisions

Matching advertisers

| Queries | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £0.99 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = B_2 = £100$



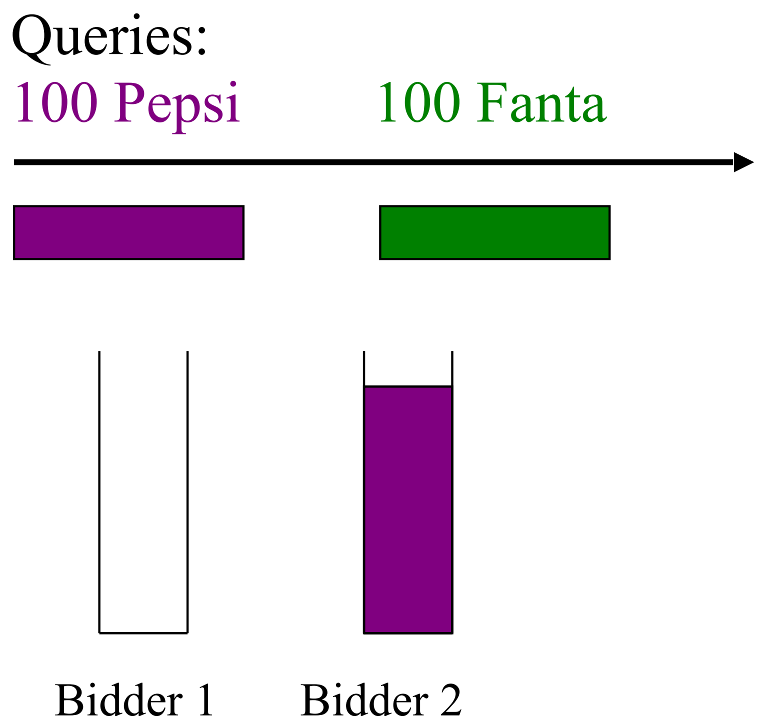
# Competitive ratio of Greedy algorithm

## Offline Optimal: hindsight matching decisions

Matching advertisers

| Queries | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £0.99 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = B_2 = £100$



# Competitive ratio of Greedy algorithm

## Offline Optimal: hindsight matching decisions

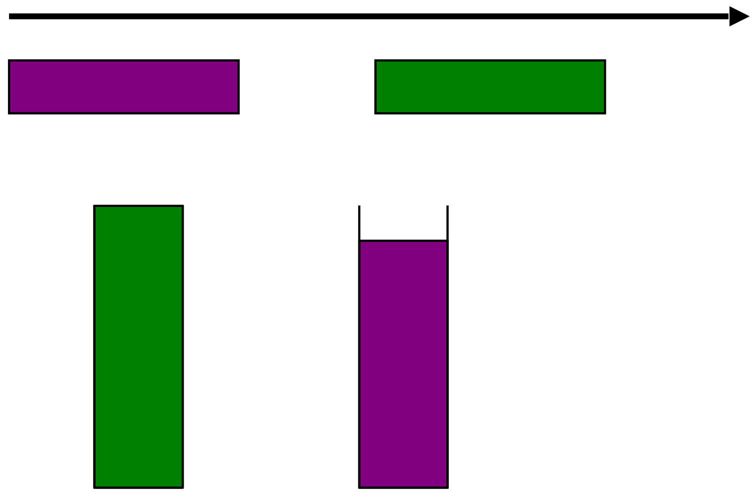
Matching advertisers

| Queries | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £0.99 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = B_2 = £100$

Queries:

100 Pepsi 100 Fanta



Bidder 1 Bidder 2

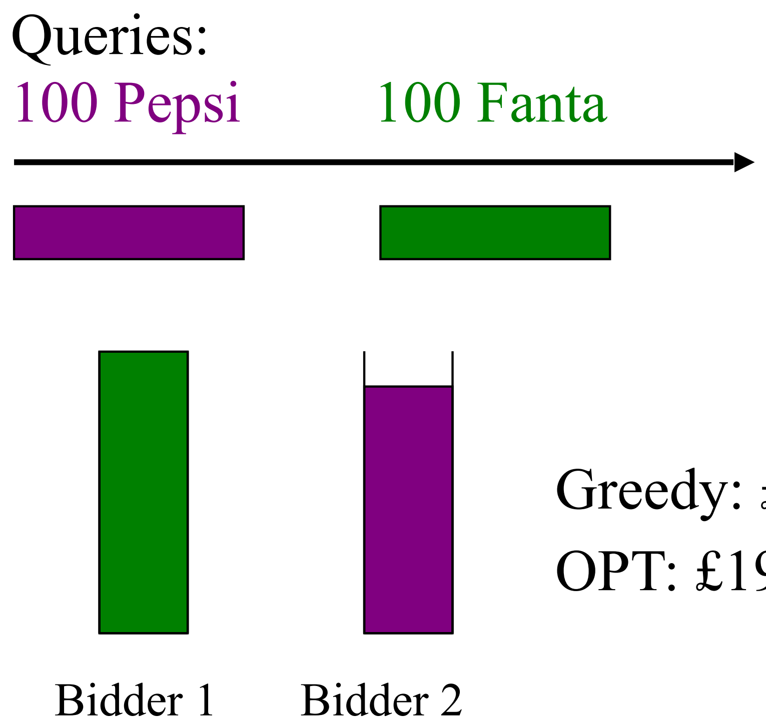
# Competitive ratio of Greedy algorithm

## Offline Optimal: hindsight matching decisions

Matching advertisers

| Queries | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £0.99 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = B_2 = £100$




Greedy: £100

OPT: £199

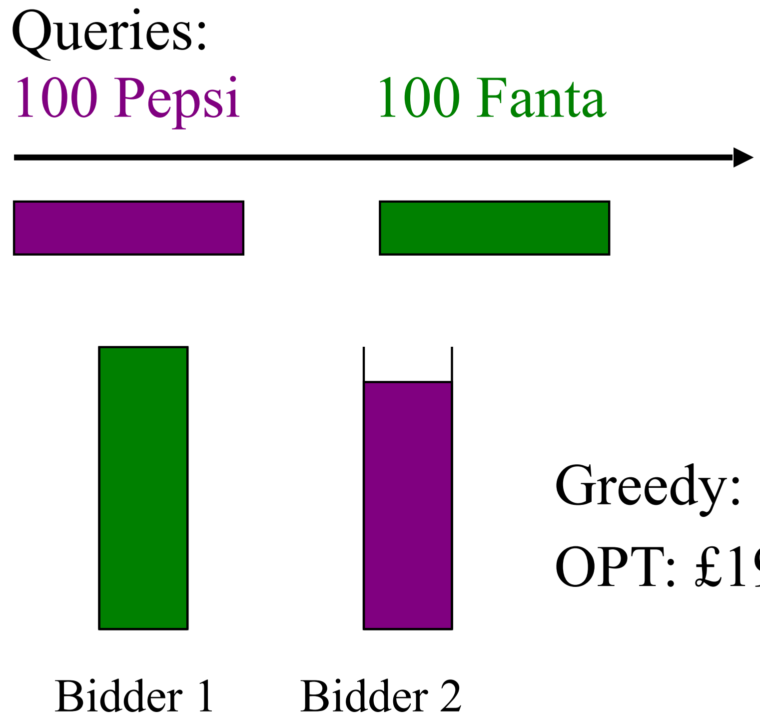
# Competitive ratio of Greedy algorithm

## Offline Optimal: hindsight matching decisions

Matching advertisers

| Queries | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £0.99 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = B_2 = £100$




Greedy: £100

OPT: £199

Greedy achieves nearly ½ of the OPT

# Competitive ratio of Greedy algorithm

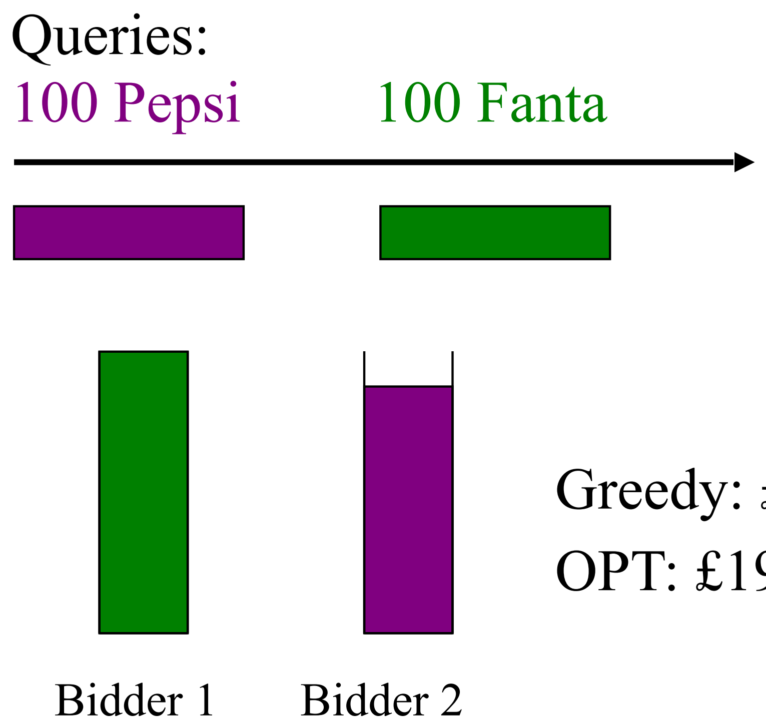
## Offline Optimal: hindsight matching decisions

Matching advertisers

Queries

| | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £0.99 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = B_2 = £100$




Greedy: £100

OPT: £199

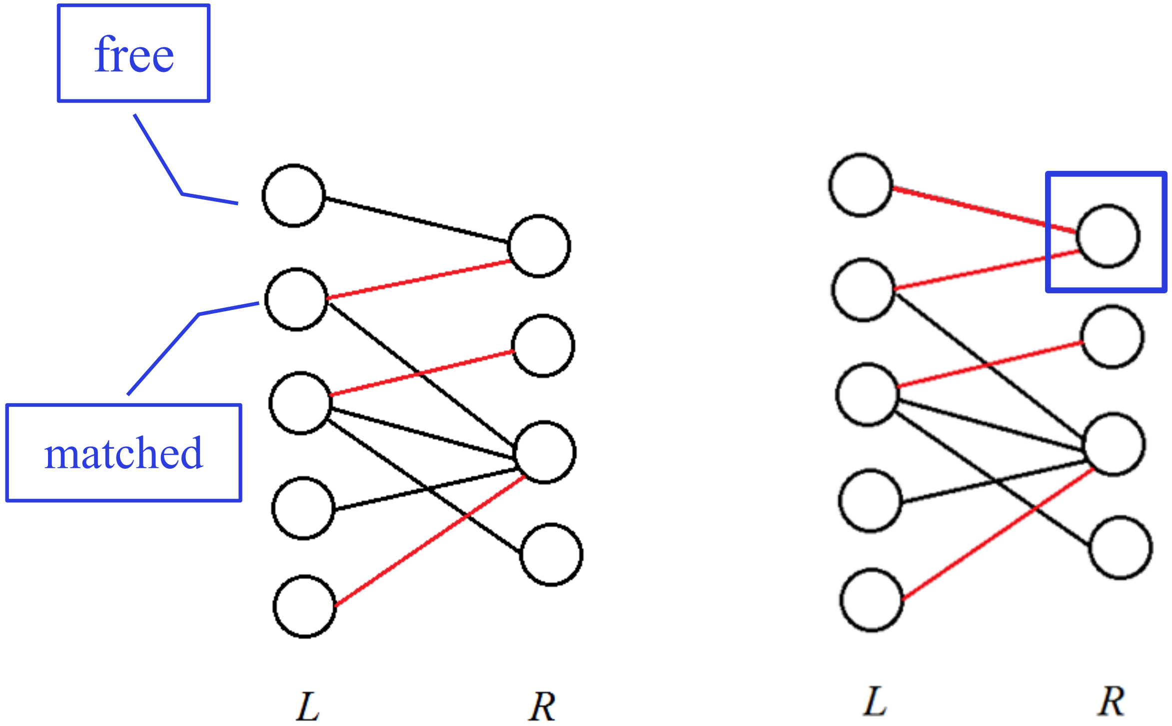
Greedy achieves nearly ½ of the OPT

Can we do better?

# Online Bipartite Matching

# Bipartite matching

- Two sets **L** and **R**, each has **n** nodes
- Each node in **L** is compatible with some nodes in **R**
- A matching is a subset of the edges s.t. every node is connected to at most one edge.

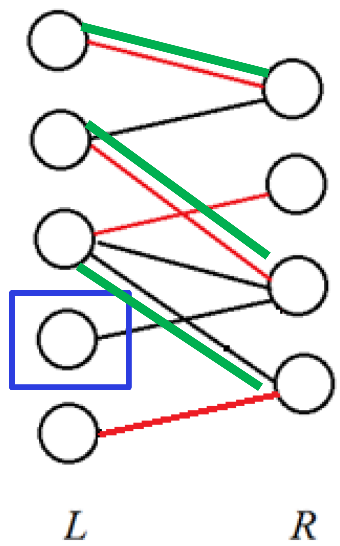



A Matching

Not a Matching

# Bipartite matching

- Maximal matching:
  - A matching that can not be made larger by adding another edge
- Maximum matching:
  - A matching that contains the largest possible number of edges
- Perfect matching:
  - A matching that matches all vertices
  - Even number of vertices
  - A.k.a., complete matching or 1-factor




- Red: a maximum matching (not perfect)
- Green: a maximal matching

A maximum matching => a maximal matching
A maximum matching <≠ a maximal matching

# Maximum Matching Algorithm

- Problem: Find a maximum matching for a given bipartite graph
  - A poly-time algorithm (Hopcroft and Karp 1973) using the concept of augmenting path
  - $O(|E|\sqrt{|V|})$

# Online Matching

- What if we don't have the entire graph upfront?

# Online Matching

- What if we don't have the entire graph upfront?
- **L** is known. In each round, one node $v$ in **R** and its neighbours **N($v$)** in **L** are revealed.

# Online Matching

- What if we don't have the entire graph upfront?
- **L** is known. In each round, one node ***v*** in **R** and its neighbours **N(*v*)** in **L** are revealed.

- At that time, we have to decide to either:
  - Pair ***v*** with a node in **N(*v*)**
  - Don't pair ***v*** with any node in **N(*v*)**
  - Again, decision cannot be canceled/revoked later

# Online Matching

- What if we don't have the entire graph upfront?
- **L** is known. In each round, one node ***v*** in **R** and its neighbours **N(*v*)** in **L** are revealed.

- At that time, we have to decide to either:
  - Pair ***v*** with a node in **N(*v*)**
  - Don't pair ***v*** with any node in **N(*v*)**
  - Again, decision cannot be canceled/revoked later

- The goal is to construct as large a matching as possible
  - Large: number of pairs matched
  - Start thinking pairing keywords **R** with advertisers **L**

# Few words on history

- Online bipartite matching was first studied in 1990, when online algorithms first became popular

- In 21st century, Web Advertising became popular and proved to be a huge revenue generator
  - **match** advertisers with ads slots

- Interests in online bipartite matching was rekindled
  - A lot of results were discovered and published within the last decade or so

# Comparison Benchmark

- If it was offline, i.e., if we could make decisions until all nodes in **R** are revealed, we could make the best decision.
  - Hopcroft and Karp's algorithm

# Comparison Benchmark

- If it was offline, i.e., if we could make decisions until all nodes in **R** are revealed, we could make the best decision.
  - Hopcroft and Karp's algorithm
- We care about the competitive ratio
  - Informally speaking, to what extend an online algorithm could have achieved with full hindsight.

# Comparison Benchmark

- If it was offline, i.e., if we could make decisions until all nodes in **R** are revealed, we could make the best decision.
  - Hopcroft and Karp's algorithm
- We care about the competitive ratio
  - Informally speaking, to what extend an online algorithm could have achieved with full hindsight.
  - Formally, an algorithm ALG is ***c***-competitive if

$$\boldsymbol{ALG(I) \leq c * OPT(I)}, \forall \ inputs \ \boldsymbol{I}$$

# Comparison Benchmark

- If it was offline, i.e., if we could make decisions until all nodes in **R** are revealed, we could make the best decision.
  - Hopcroft and Karp's algorithm
- We care about the competitive ratio
  - Informally speaking, to what extend an online algorithm could have achieved with full hindsight.
  - Formally, an algorithm ALG is ***c***-competitive if

$$\boldsymbol{ALG(I) \le c * OPT(I), \forall\ inputs\ I}$$

  - Worst-case analysis.

# Example: A Greedy Algorithm

- When $\boldsymbol{v} \in \boldsymbol{R}$ appears, pair it with any compatible node in $\boldsymbol{L}$
- How good is the algorithm?

# Example: A Greedy Algorithm

- When $\boldsymbol{v} \in \boldsymbol{R}$ appears, pair it with any compatible node in $\boldsymbol{L}$
- How good is the algorithm?
  - At most **½**, why?

# Example: A Greedy Algorithm

- When $v \in R$ appears, pair it with any compatible node in $L$
- How good is the algorithm?
  - At most ½, why?
  - Any deterministic algorithm is at least ½

# Example: A Greedy Algorithm

- When $\boldsymbol{v} \in \boldsymbol{R}$ appears, pair it with any compatible node in $\boldsymbol{L}$
- How good is the algorithm?
  - At most **½**, why?
  - Any deterministic algorithm is at least **½**
  - So the competitive ratio is tight **½**.

# Online Fractional Bipartite Matching

- Fractional matching
  - When a node $\mathbf{v} \in \mathbf{R}$ arrives, not necessarily need to fully match it to a node in $\mathbf{N(v)}$; could partially match $\mathbf{v}$ with some nodes in $\mathbf{N(v)}$
  - In other words, decisions are not binary anymore

# Water Level (WL) Algorithm

- a.k.a., water-filling algorithm

# Water Level (WL) Algorithm

- a.k.a., water-filling algorithm

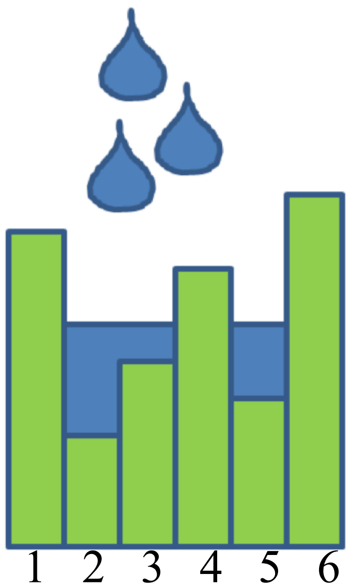
- Physical metaphor:
  - each vertex $\mathbf{w} \in \mathbf{L}$ is a container with capacity $\mathbf{1}$
  - each vertex $\mathbf{v} \in \mathbf{R}$ is a source of one unit of water

# Water Level (WL) Algorithm

- a.k.a., water-filling algorithm

- Physical metaphor:
  - each vertex **w** $\in$ **L** is a container with capacity **1**
  - each vertex **v** $\in$ **R** is a source of one unit of water

- High level description:
  - When a node **v** $\in$ **R** arrives, drain water from **v** to **N(v)** , always preferring the neighbour with the lowest current water level, until either
    - all **N(v)** is full, or
    - **v** is empty



# Water Level Algorithm

- The WL algorithm is deterministic and is $\left(\mathbf{1} - \frac{\mathbf{1}}{\boldsymbol{e}}\right)$-competitive for the online *fractional* bipartite matching problem

- The analysis (that we didn't discuss here) is tight
  - There is an instance/example for which WL returns a fractional matching of size $\mathbf{1} - \frac{\mathbf{1}}{\boldsymbol{e}}$ times the maximum possible

* If you are interested in the analysis/proof, you can find it here.

# Online Randomised Bipartite Matching

- But what about our original problem of *integral* online bipartite matching? (Note that fractional $\neq$ randomised)

# Online Randomised Bipartite Matching

- But what about our original problem of *integral* online bipartite matching? (Note that fractional $\neq$ randomised)
- The same idea as in WL can be used to design a randomised online algorithm for the original integral online bipartite matching problem

# Online Randomised Bipartite Matching

- But what about our original problem of *integral* online bipartite matching? (Note that fractional $\neq$ randomised)
- The same idea as in WL can be used to design a randomised online algorithm for the original integral online bipartite matching problem
- RANKING algorithm by [Karp, Vazirani, Vazirani, 1990]
  - $\left(1 - \frac{1}{e}\right)$ - competitive
  - **High-level idea**: a special transformation from WL such that the probability that a given edge is included in the matching plays the same role as its fractional value in the WL algorithm

# Online Randomised Bipartite Matching

- But what about our original problem of *integral* online bipartite matching? (Note that fractional $\neq$ randomised)
- The same idea as in WL can be used to design a randomised online algorithm for the original integral online bipartite matching problem
- RANKING algorithm by [Karp, Vazirani, Vazirani, 1990]
  - $(1 - \frac{1}{e})$ - competitive
  - **High-level idea**: a special transformation from WL such that the probability that a given edge is included in the matching plays the same role as its fractional value in the WL algorithm

**The competitive ratio of $\left(1 - \frac{1}{e}\right)$ is tight!**

No online algorithm, deterministic or randomised, has a competitive ratio better than $1 - \frac{1}{e}$ for maximum bipartite matching.

# Online b-matching

- b-matching: every node in **L** can be matched for at most **b** times
  - think of **b** as the capacity of each node in **L**

# Online b-matching

- b-matching: every node in **L** can be matched for at most **b** times
  - think of **b** as the capacity of each node in **L**

- A deterministic algorithm BALANCE [Kalyanasundaram and Pruhs, 2000] achieves the optimal competitive ratio $\mathbf{1-\frac{1}{\boldsymbol{e}}}$ for this problem

# Online b-matching

- b-matching: every node in **L** can be matched for at most **b** times
  - think of **b** as the capacity of each node in **L**

- A deterministic algorithm BALANCE [Kalyanasundaram and Pruhs, 2000] achieves the optimal competitive ratio $\mathbf{1-\frac{1}{e}}$ for this problem
  - The competitive ratio is $\mathbf{1-\frac{1}{(1+\frac{1}{b})^b}}$ to be precise, which approaches $\mathbf{1-\frac{1}{e}}$ for large b

# Online b-matching

- b-matching: every node in **L** can be matched for at most **b** times
  - think of **b** as the capacity of each node in **L**

- A deterministic algorithm BALANCE [Kalyanasundaram and Pruhs, 2000] achieves the optimal competitive ratio $\mathbf{1-\frac{1}{e}}$ for this problem
  - The competitive ratio is $\mathbf{1-\frac{1}{(1+\frac{1}{b})^{b}}}$ to be precise, which approaches $\mathbf{1-\frac{1}{e}}$ for large b
  - **Idea**: when a node **v** in **R** arrives, match it to a node **N(v)** in **L** with highest remaining match “allowance”

# Online b-matching

- b-matching: every node in **L** can be matched for at most **b** times
  - think of **b** as the capacity of each node in **L**

- A deterministic algorithm BALANCE [Kalyanasundaram and Pruhs, 2000] achieves the optimal competitive ratio $\mathbf{1 - \frac{1}{e}}$ for this problem
  - The competitive ratio is $\mathbf{1 - \frac{1}{(1+\frac{1}{b})^{b}}}$ to be precise, which approaches $\mathbf{1 - \frac{1}{e}}$ for large b
  - **Idea**: when a node **v** in **R** arrives, match it to a node **N(v)** in **L** with highest remaining match "allowance"
    - If there is more than one such node, pick one arbitrarily

# The BALANCE algorithm for *b*-matching

- Special Case: All budgets are £100; bids are either £0 or £1 (all non-zero bids are the same)
- BALANCE: awards the query to an interested bidder who has the highest unspent budget

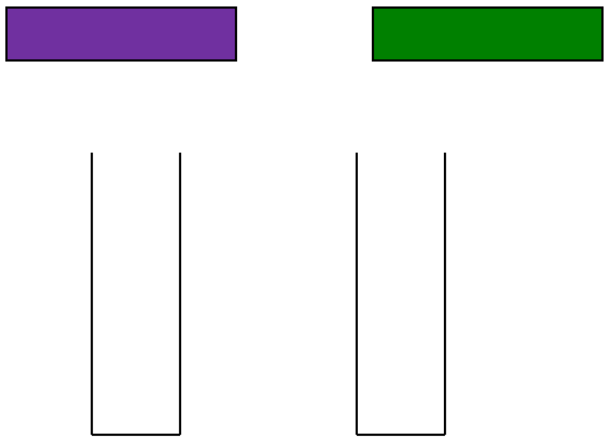
# The BALANCE algorithm for *b*-matching

- Special Case: All budgets are £100; bids are either £0 or £1 (all non-zero bids are the same)
- BALANCE: awards the query to an interested bidder who has the highest unspent budget

| | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £1 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = \text{Budget}_2 = £100$

Queries:
100 Pepsi then 100 Fanta



Bidder 1 Bidder 2

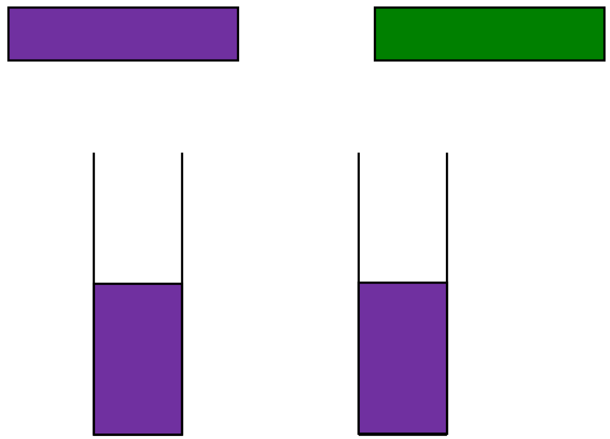
# The BALANCE algorithm for *b*-matching

- Special Case: All budgets are £100; bids are either £0 or £1 (all non-zero bids are the same)
- BALANCE: awards the query to an interested bidder who has the highest unspent budget

| | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £1 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = \text{Budget}_2 = £100$

Queries:
100 Pepsi then 100 Fanta



Bidder 1 Bidder 2

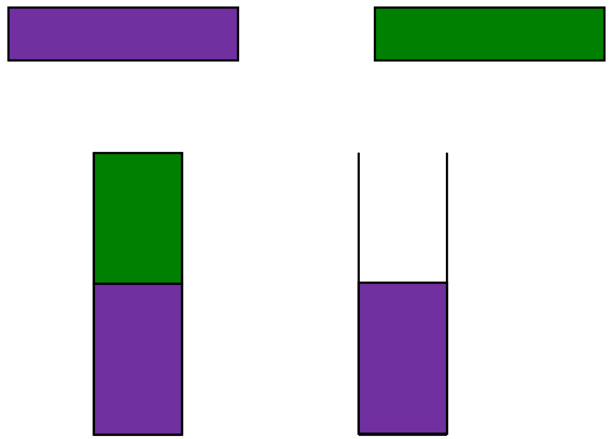
# The BALANCE algorithm for *b*-matching

- Special Case: All budgets are £100; bids are either £0 or £1 (all non-zero bids are the same)
- BALANCE: awards the query to an interested bidder who has the highest unspent budget

| | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £1 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = \text{Budget}_2 = £100$

Queries:
100 Pepsi then 100 Fanta



Bidder 1 Bidder 2

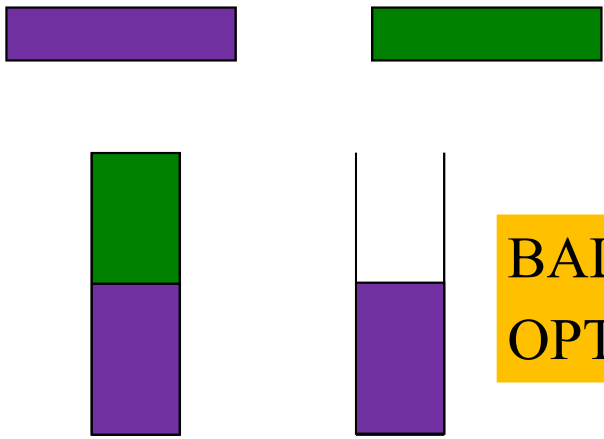
# The BALANCE algorithm for *b*-matching

- Special Case: All budgets are £100; bids are either £0 or £1 (all non-zero bids are the same)
- BALANCE: awards the query to an interested bidder who has the highest unspent budget

| | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £1 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = \text{Budget}_2 = £100$

Queries:
100 Pepsi then 100 Fanta



Bidder 1 Bidder 2

BALANCE: £150
OPTIMAL: £200

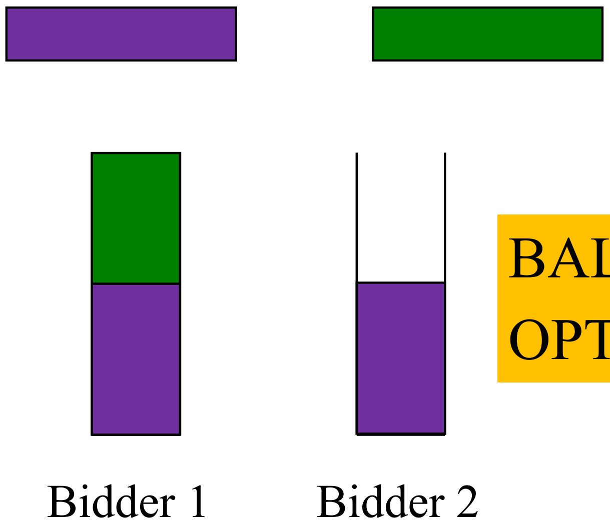
# The BALANCE algorithm for *b*-matching

- Special Case: All budgets are £100; bids are either £0 or £1 (all non-zero bids are the same)
- BALANCE: awards the query to an interested bidder who has the highest unspent budget

| | Bidder 1 | Bidder 2 |
|---|---|---|
| Pepsi | £1 | £1 |
| Fanta | £1 | £0 |

$\text{Budget}_1 = \text{Budget}_2 = £100$

Queries:
100 Pepsi then 100 Fanta




BALANCE: £150
OPTIMAL: £200

Competitive ratio on this instance: ¾
However, the algorithm's competitive ratio is 1-1/*e*

# Extension from *b*-matching: bids can be different

- We have learnt
  - By considering the remaining budget of the bidders, we can possibly improve the competitive ratio

# Extension from *b*-matching: bids can be different

- We have learnt
  - By considering the remaining budget of the bidders, we can possibly improve the competitive ratio
  - So, maybe we should consider both the advertisers' bid and their unspent budget?

# Extension from *b*-matching: bids can be different

- We have learnt
  - By considering the remaining budget of the bidders, we can possibly improve the competitive ratio

  - So, maybe we should consider both the advertisers' bid and their unspent budget?

  - But, what is the best tradeoff between this two, in order to obtain the optimal competitive ratio?

# Factor-revealing LP algorithm (achieving optimal competitive ratio)

- When a query arrives, allocate it to the bidder **i** maximizing $\mathbf{b_i \times \varphi(x)}$

# Factor-revealing LP algorithm (achieving optimal competitive ratio)

- When a query arrives, allocate it to the bidder **i** maximizing $\mathbf{b_i \times \varphi(x)}$
  - $\mathbf{b_i}$ is bidder **i**'s bid
  - $\boldsymbol{\varphi(x) = 1 - e^{x-1}}$

# Factor-revealing LP algorithm (achieving optimal competitive ratio)

- When a query arrives, allocate it to the bidder **i** maximizing $\mathbf{b_i \times \varphi(x)}$
  - $\mathbf{b_i}$ is bidder **i**'s bid
  - $\boldsymbol{\varphi(x) = 1 - e^{x-1}}$
  - **x** is the fraction of **i**'s budget that has been spent so far.
  - In particular, $\mathbf{x = \frac{m_i}{B}}$, where $\mathbf{m_i}$ is the amount of money spent by **i** when the query arrives, and **B** is **i**'s total budget.

# Factor-revealing LP algorithm (achieving optimal competitive ratio)

- When a query arrives, allocate it to the bidder **i** maximizing $\mathbf{b_i \times \varphi(x)}$
  - $\mathbf{b_i}$ is bidder **i**'s bid
  - $\mathbf{\varphi(x) = 1 - e^{x-1}}$
  - **x** is the fraction of **i**'s budget that has been spent so far.
  - In particular, $\mathbf{x = \frac{m_i}{B}}$, where $\mathbf{m_i}$ is the amount of money spent by **i** when the query arrives, and **B** is **i**'s total budget.

**Theorem** [Mehta, Saberi, Vazirani, Vazirani , FOCS 2005]
The competitive ratio of the Factor-revealing LP algorithm is
1-1/*e* = 0.632 and it is optimal.

# Back to the original model

- We started from a simplified model; to what extent the result can be migrated back to the realistic model?

# Back to the original model

- We started from a simplified model; to what extent the result can be migrated back to the realistic model?

- The same competitive ratio can be achieved when
  - Bidders have different budgets
  - Multiple ads can appear with the results of a query
  - A bidder pays only if the user clicks on its ad
  - Instead of charging a bidder its actual bid, the search engine charges it the next highest bid (GSP)

# Advertiser/bidder

- Each has a set of interested keywords
- Each has a private value for each of the keyword
- Each has a private daily/weekly/monthly budget
- Use their budget to buy clicks
- Obj: how to maximise utility, subject to budget?

# Search Engine

- Products of Search Engines
  - $k$ slots to sell
  - Each slot can generate a certain number of clicks

- How to match slots to advertisers?
- How to charge advertisers?
- Obj: how to maximise revenue?

- Pay-per-impression
- Pay-per-click
- Pay-per-action

# Review & Preview

- Auction models
  - Classical auction models
  - Generalisation to sponsored search
- Online bipartite matching
  - Worst-case analysis
- Sponsored search auctions
  - Revenue maximisation
  - Strategic behaviours
- Extra reading


Sponsored search auctions: an overview of research with emphasis on game theoretic aspects

Patrick Maillé, Evangelos Markakis, Maurizio Naldi, George D. Stamoulis & Bruno Tuffin

*Electronic Commerce Research* **12**, 265–300(2012) | Cite this article