Adwords: Online Matching in Advertising

Seminar: Selected Topics in Efficient Algorithms

Mitja Krebs

Chair of Algorithms and Complexity
(Prof. Dr. Susanne Albers)
Department of Computer Science
Technical University of Munich

December 6, 2018

Outline

Outline

Outline

Google Ads

Google Ads

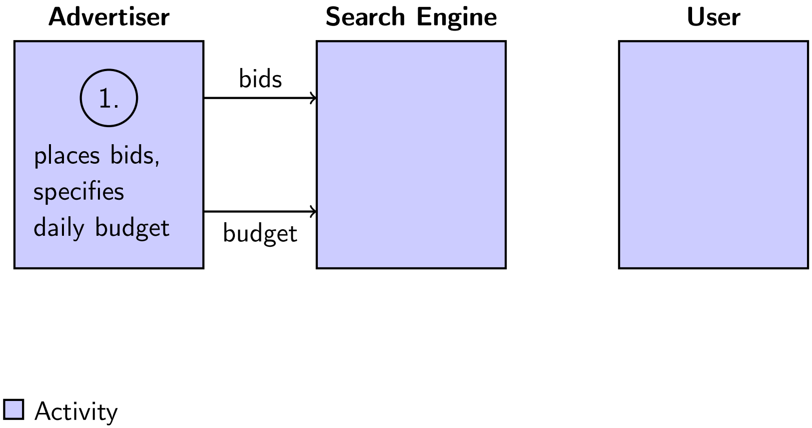
__Google Ads__ (previously __Google AdWords__ [...]) is an online advertising platform developed by Google, where advertisers pay to display brief advertisements [...] within the Google ad network to web users. Google Ads' system is based [...] on keywords determined by advertisers. Google uses [this characteristic] to place advertising copy on pages where they think it might be relevant.

(Wikipedia contributors "Google Ads")

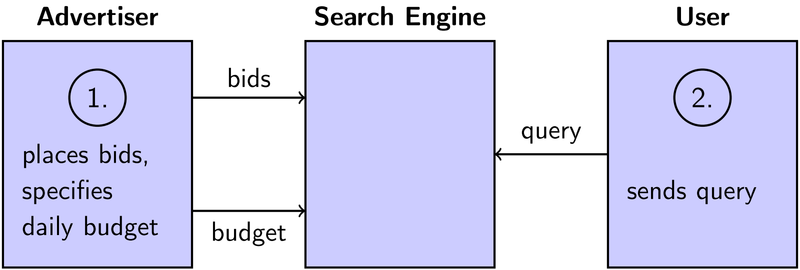
Search Advertising

Advertiser **Search Engine** **User**

Search Advertising

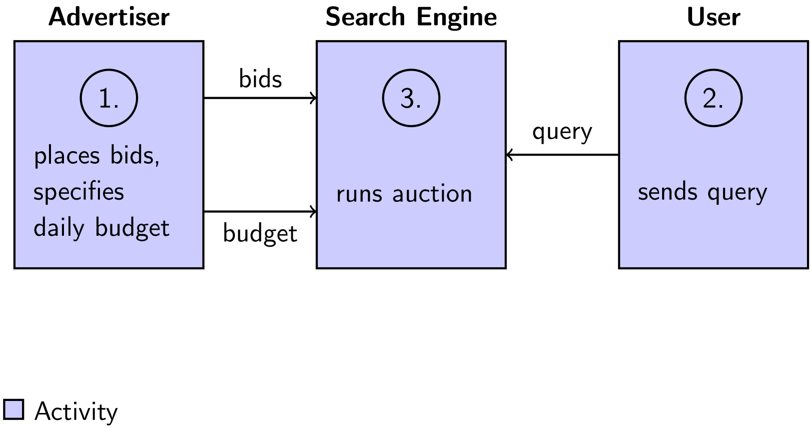


Search Advertising

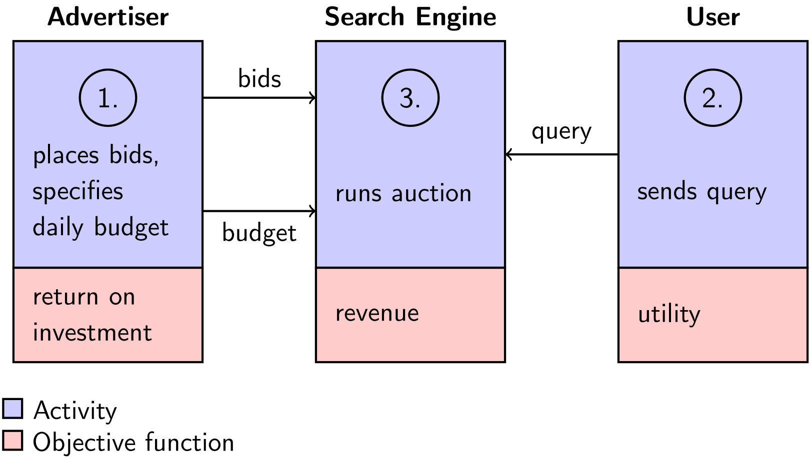


Activity

Search Advertising



Search Advertising



Outline

Matching

Definition

Let $G = (V, E)$ be a graph. Then $M \subseteq E$ is called matching if every vertex $v \in V$ is incident to at most one edge $e \in M$.

Let M be a matching in G. Then:

- M is called bipartite matching if G is bipartite.
- M is called maximum matching if $|M| \geq |M'|$ for all matchings $M' \subseteq E$.

Online vs. Offline Algorithm

Definition

1. An online algorithm ALG is presented with a request sequence $\sigma = \sigma_1, \ldots, \sigma_m$. The requests $\sigma_t, 1 \leq t \leq m$, must be served in their order of occurence. More specifically, when serving request σ_t, ALG does not know any request $\sigma_{t'}$ with $t' > t$.

   Serving requests affects the objective function value attained by ALG, and the goal is to optimize the value of the objective function attained on the entire request sequence.

Online vs. Offline Algorithm

Definition

1. An online algorithm ALG is presented with a request sequence $\sigma = \sigma_1, \ldots, \sigma_m$. The requests $\sigma_t, 1 \leq t \leq m$, must be served in their order of occurence. More specifically, when serving request σ_t, ALG does not know any request $\sigma_{t'}$ with $t' > t$.

   Serving requests affects the objective function value attained by ALG, and the goal is to optimize the value of the objective function attained on the entire request sequence.
2. An offline algorithm, on the other hand, is an omniscient algorithm that knows the entire request sequence in advance and can compute an optimum output.

Online Bipartite Matching Problem

Online Bipartite Matching Problem

Given: A bipartite graph $G = (U, V, E)$, where:

- U arrives offline
- V arrives online
- When a vertex in V arrives, its neighbors in U are revealed

Online Bipartite Matching Problem

Online Bipartite Matching Problem

Given: A bipartite graph $G = (U, V, E)$, where:

- U arrives offline
- V arrives online
- When a vertex in V arrives, its neighbors in U are revealed

Task:

- An arriving vertex needs to be matched to an available neighbor (if any)
- A match once made cannot be revoked

Online Bipartite Matching Problem

Online Bipartite Matching Problem

Given: A bipartite graph $G = (U, V, E)$, where:

- U arrives offline
- V arrives online
- When a vertex in V arrives, its neighbors in U are revealed

Task:

- An arriving vertex needs to be matched to an available neighbor (if any)
- A match once made cannot be revoked

Goal: Maximize the size of the matching

Adwords Problem

Adwords Problem

Given: A bipartite graph $G = (U, V, E)$, where:

- Each vertex $u \in U$ has a budget b_u
- Each edge $(u, v) \in E$ has a bid $c_{u,v}$

Adwords Problem

Adwords Problem

Given: A bipartite graph $G = (U, V, E)$, where:

- Each vertex $u \in U$ has a budget b_u
- Each edge $(u, v) \in E$ has a bid $c_{u,v}$

Task:

- When an arriving vertex $v \in V$ is matched to a neighbor $u \in U$, then u depletes $c_{u,v}$ amount of its budget
- When a vertex depletes its entire budget, then it becomes unavailable

Adwords Problem

Adwords Problem

Given: A bipartite graph $G = (U, V, E)$, where:

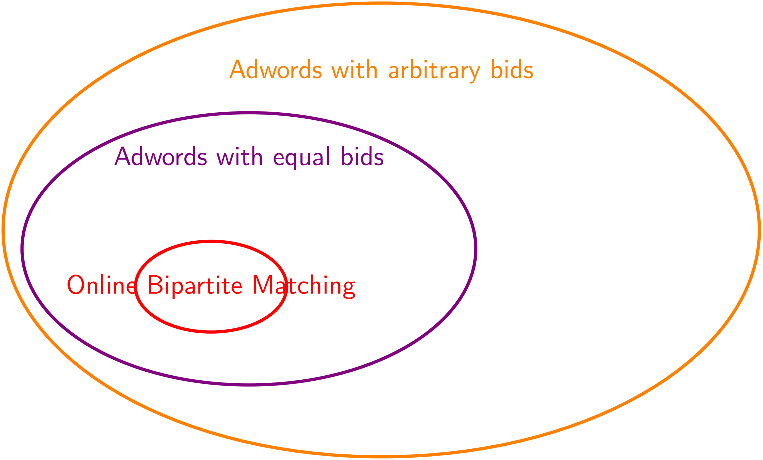
- Each vertex $u \in U$ has a budget b_u
- Each edge $(u, v) \in E$ has a bid $c_{u,v}$

Task:

- When an arriving vertex $v \in V$ is matched to a neighbor $u \in U$, then u depletes $c_{u,v}$ amount of its budget
- When a vertex depletes its entire budget, then it becomes unavailable

Goal: Maximize the total money spent

Landscape of Problems

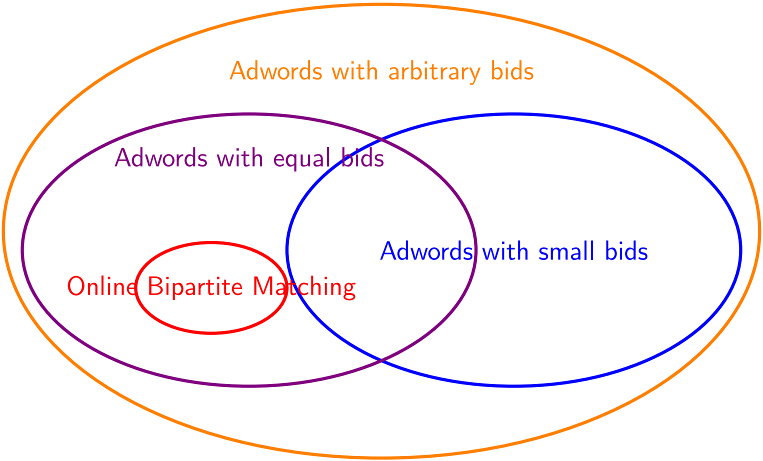


Outline

Assumptions

1. Each bid is small compared to the corresponding budget (i.e., $\forall (u, v) \in E : c_{u,v} \ll b_u$).

Landscape of Problems



GREEDY

Algorithm 1: GREEDY

when *the next vertex* $v \in V$ *arrives*:
  if *all neighbors of* v *are unavailable* **then**
    continue
  else
    match v to that available neighbor $u \in U$ which maximizes $c_{u,v}$

BALANCE

Algorithm 2: BALANCE

when *the next vertex* $v \in V$ *arrives*:

  if *all neighbors of* v *are unavailable* **then**

    continue

  else

    match v to that available neighbor $u \in U$ which has spent the least fraction of its budget so far

Competitive Ratio

Definition

In a competitive analysis, an online algorithm ALG is compared to an optimum offline algorithm OPT.

Given the entire graph $G = (U, V, E)$ and the input order of V, σ, let $\mathrm{ALG}(G, \sigma)$ denote the value of the objective function attained by ALG, and let $\mathrm{OPT}(G, \sigma)$ denote the maximum objective function value attained offline.

The algorithm ALG is called α-competitive if $\frac{\mathrm{ALG}(G,\sigma)}{\mathrm{OPT}(G,\sigma)} \geq \alpha$ for all graphs G and all input orders σ. The factor α is also called the competitive ratio of ALG.

Two Extreme Examples for the Adwords Problem

Example

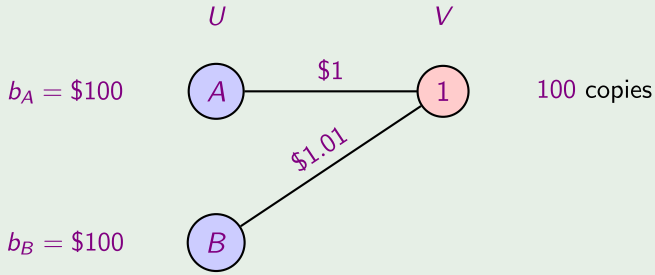
U

$b_A = \$100$ A

$b_B = \$100$ B

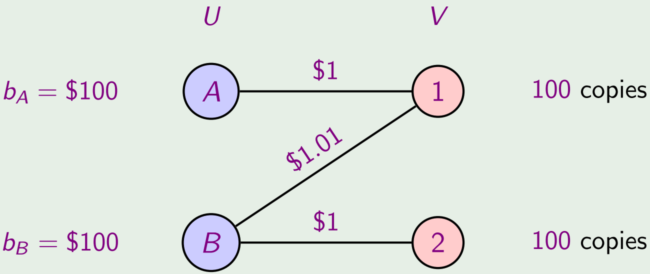
Two Extreme Examples for the Adwords Problem

Example



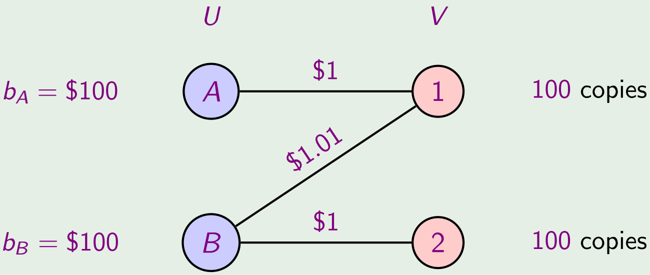
Two Extreme Examples for the Adwords Problem

Example



Two Extreme Examples for the Adwords Problem

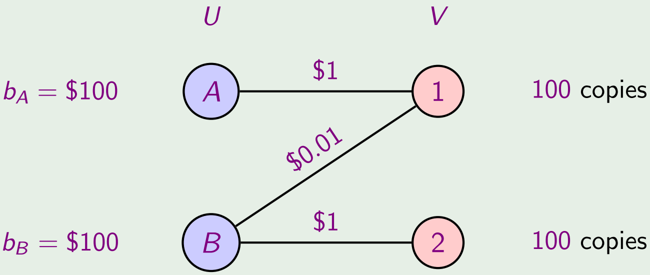
Example



GREEDY achieves a ratio of $\frac{1}{2}$.
BALANCE achieves a ratio of $\frac{3}{4}$.

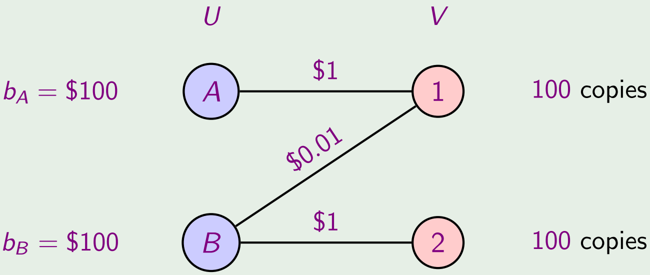
Two Extreme Examples for the Adwords Problem

Example



Two Extreme Examples for the Adwords Problem

Example



GREEDY achieves a ratio of 1.
BALANCE achieves a ratio of $\frac{1}{2}$.

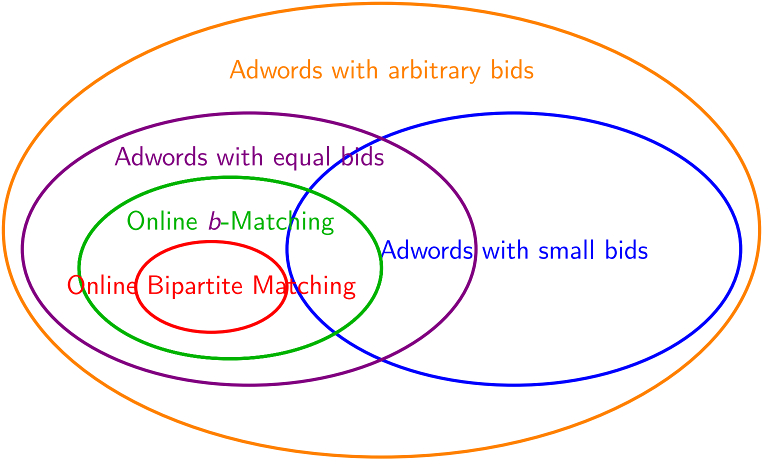
Online b-Matching Problem

Online b-Matching Problem

The Online b-Matching problem is the special case of the Adwords problem in which

- each budget is equal to $b \in \mathbb{N}$
- each non-zero bid is equal to 1

Landscape of Problems



MSVV

Algorithm 3: MSVV

when *the next vertex* $v \in V$ *arrives*:
  if *all neighbors of* v *are unavailable* **then**
    continue
  else
    match v to that available neighbor $u \in U$ which maximizes $c_{u,v}\psi\left(\frac{s_u}{b_u}\right)$,
    where s_u is the amount of u's budget spent so far, and $\psi(x) = 1 - e^{x-1}$

If the budgets are all infinity, then MSVV becomes GREEDY.
If the bids are all equal, then MSVV becomes BALANCE.

Outline

Outline

Assumptions

1. Each bid is small compared to the corresponding budget (i.e., $\forall(u, v) \in E : c_{u,v} \ll b_u$).
2. Each bidder has a budget of 1.

Slab

The budget of each bidder is discretized into k equal parts, where k is a large integer.

Definition

Let $i \in \{1, \ldots, k\}$. Slab i is the $\left[\frac{i-1}{k}, \frac{i}{k}\right)$ portion of each bidder's budget.

Slab

The budget of each bidder is discretized into k equal parts, where k is a large integer.

Definition

Let $i \in \{1, \ldots, k\}$. Slab i is the $\left[\frac{i-1}{k}, \frac{i}{k}\right)$ portion of each bidder's budget.

Notation

Let $u \in U, v \in V$. Then $\text{slab}(u, v)$ denotes the currently active slab for u at the arrival of v. If v is apparent from the context, we write $\text{slab}(u)$ instead.

Let $i \in \{1, \ldots, k\}$. Then β_i denotes the total money spent by the bidders from slab i in the run of MSVV.

Assumptions

1. $\forall (u, v) \in E : c_{u,v} \leq \frac{1}{k^2}$.
2. Each bidder has a budget of 1.

Notation

$n := |U|$

Notation

$n := |U|$

Definition

Let $i \in \{1, \ldots, k\}$. A bidder is of type i if the money spent by it at the end of MSVV lies in the range $\left(\frac{i-1}{k}, \frac{i}{k}\right]$.

By convention a bidder who spends none of its budget is assigned type 1.

Notation

$n := |U|$

Definition

Let $i \in \{1, \ldots, k\}$. A bidder is of type i if the money spent by it at the end of MSVV lies in the range $\left(\frac{i-1}{k}, \frac{i}{k}\right]$.

By convention a bidder who spends none of its budget is assigned type 1.

Notation

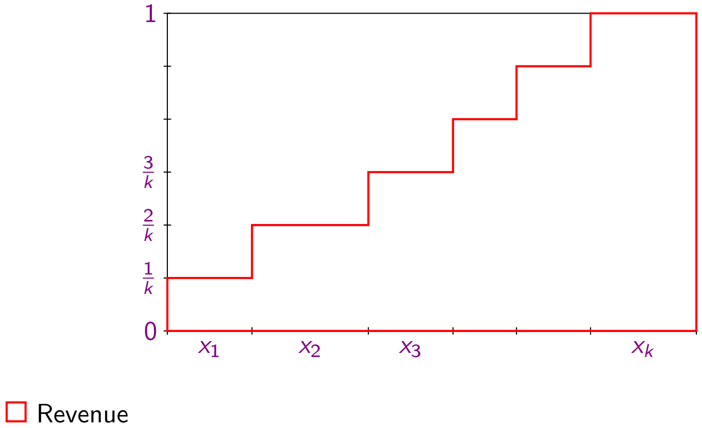
Let $u \in U$. Then $\text{type}(u)$ denotes the type of bidder u.

Let $i \in \{1, \ldots, k\}$. Then x_i denotes the number of bidders of type i (so, $\sum_{i=1}^{k} x_i = n$).

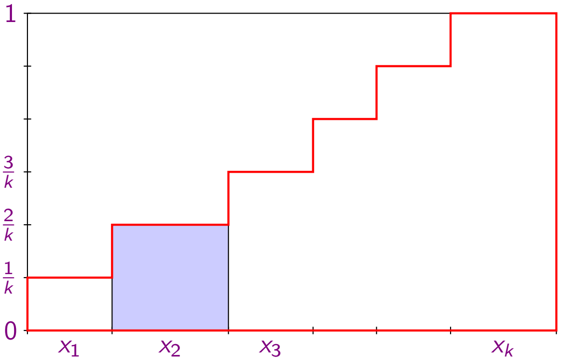
Assumptions

1. $\forall (u, v) \in E : c_{u,v} \leq \frac{1}{k^2}$.
2. Each bidder has a budget of 1.
3. OPT exhausts the budget of each bidder.

Revenue Structure at the End of MSVV

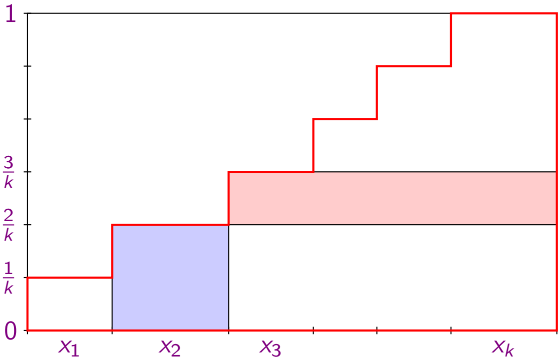


Revenue Structure at the End of MSVV



- Revenue
- Total money spent by bidders of type 2

Revenue Structure at the End of MSVV



- Revenue
- Total money spent by bidders of type 2
- Total money spent by bidders from slab 3

Assumptions

1. $\forall (u, v) \in E : c_{u,v} \leq \frac{1}{k^2}$.
2. Each bidder has a budget of 1.
3. OPT exhausts the budget of each bidder.
4. Each bidder of type $i \in \{1, ..., k\}$ spends exactly $\frac{i}{k}$.

Assumptions

Each bidder of type $i \in \{1, \ldots, k\}$ spends exactly $\frac{i}{k}$.

The total error resulting from this simplification is at most $\frac{n}{k}$.

Assumptions

Each bidder of type $i \in \{1, \ldots, k\}$ spends exactly $\frac{i}{k}$.

The total error resulting from this simplification is at most $\frac{n}{k}$.

$\frac{i}{k}$

$\frac{i-1}{l}$

Assumptions

Each bidder of type $i \in \{1, ..., k\}$ spends exactly $\frac{i}{k}$.

The total error resulting from this simplification is at most $\frac{n}{k}$.

Assumptions

Each bidder of type $i \in \{1, ..., k\}$ spends exactly $\frac{i}{k}$.

The total error resulting from this simplification is at most $\frac{n}{k}$.

$\frac{i}{k} \longleftarrow$ **simplification:** $\frac{i}{k}$

$\frac{i-1}{l} \longleftarrow$ **worst case:** $\frac{i-1}{k} + \varepsilon$

Assumptions

Each bidder of type $i \in \{1, \dots, k\}$ spends exactly $\frac{i}{k}$.

The total error resulting from this simplification is at most $\frac{n}{k}$.

$$\frac{i}{k} \longleftarrow \textbf{simplification: } \frac{i}{k}$$

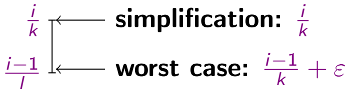
$$\frac{i-1}{l} \longleftarrow \textbf{worst case: } \frac{i-1}{k} + \varepsilon$$

Error for a bidder of type i is at most: $\frac{i}{k} - \left(\frac{i-1}{k} + \varepsilon\right) = \frac{1}{k} - \varepsilon \leq \frac{1}{k}$

Assumptions

Each bidder of type $i \in \{1, \dots, k\}$ spends exactly $\frac{i}{k}$.

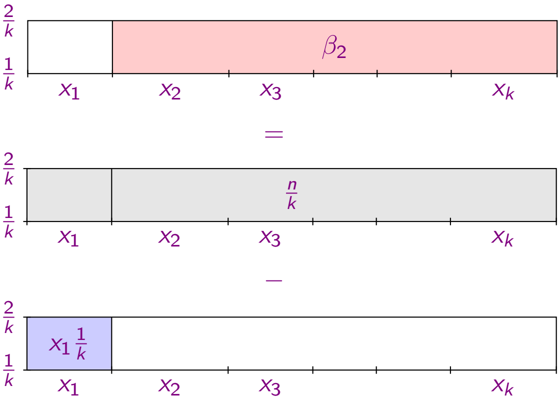
The total error resulting from this simplification is at most $\frac{n}{k}$.



Error for a bidder of type i is at most: $\frac{i}{k} - \left(\frac{i-1}{k} + \varepsilon\right) = \frac{1}{k} - \varepsilon \leq \frac{1}{k}$

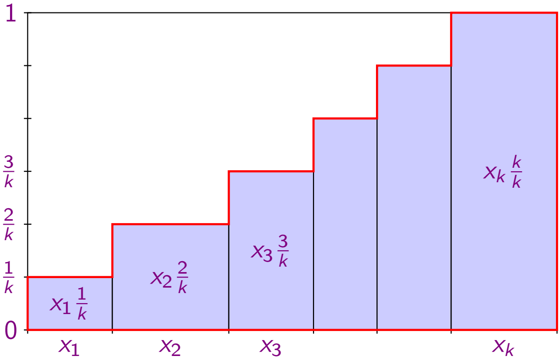
Total error is at most: $n\frac{1}{k} = \frac{n}{k}$

Relation Between the β_i and x_i

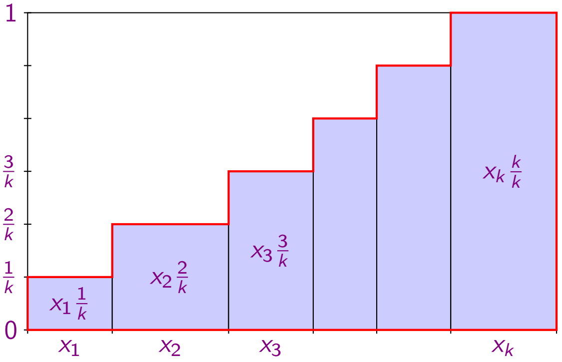


$$\forall i \in \{1, \ldots, k\} : \beta_i = \frac{n - \sum_{j=1}^{i-1} x_j}{k}$$

Revenue Calculation for MSVV via Types

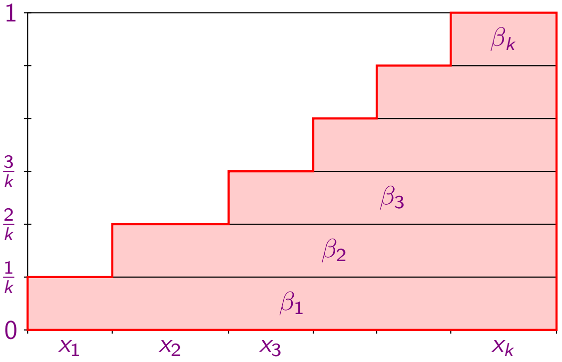


Revenue Calculation for MSVV via Types

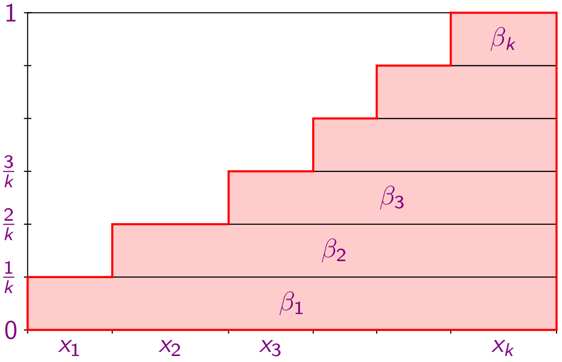


$$\pi_{\text{MSVV}} \geq \sum_{i=1}^{k} x_i \frac{i}{k} - \frac{n}{k}$$

Revenue Calculation for MSVV via Slabs



Revenue Calculation for MSVV via Slabs



$$\pi_{\text{MSVV}} \geq \sum_{i=1}^{k} \beta_i - \frac{n}{k}$$

Optimization Problem

Definition

An instance of an optimization problem is specified by the data:

$n \in \mathbb{N}$

Optimization Problem

Definition

An instance of an optimization problem is specified by the data:

$$n \in \mathbb{N}, \qquad F \subseteq \mathbb{R}^n$$

Optimization Problem

Definition

An instance of an optimization problem is specified by the data:

$$n \in \mathbb{N}, \qquad F \subseteq \mathbb{R}^n, \qquad \varphi : \mathbb{R}^n \to \mathbb{R}$$

Optimization Problem

Definition

An instance of an optimization problem is specified by the data:

$$n \in \mathbb{N}, \qquad F \subseteq \mathbb{R}^n, \qquad \varphi : \mathbb{R}^n \to \mathbb{R}, \qquad \text{opt} \in \{\min, \max\}$$

Optimization Problem

Definition

An instance of an optimization problem is specified by the data:

$$n \in \mathbb{N}, \qquad F \subseteq \mathbb{R}^n, \qquad \varphi : \mathbb{R}^n \to \mathbb{R}, \qquad \text{opt} \in \{\min, \max\}$$

A point $x^* \in F$ is optimal if

$$\begin{aligned} x \in F \quad &\Rightarrow \quad \varphi(x^*) \leq \varphi(x) \qquad \text{for opt} = \min \\ x \in F \quad &\Rightarrow \quad \varphi(x^*) \geq \varphi(x) \qquad \text{for opt} = \max \end{aligned}$$

Optimization Problem

Definition

An instance of an optimization problem is specified by the data:

$$n \in \mathbb{N}, \qquad F \subseteq \mathbb{R}^n, \qquad \varphi : \mathbb{R}^n \to \mathbb{R}, \qquad \text{opt} \in \{\min, \max\}$$

A point $x^* \in F$ is optimal if

$$\begin{aligned} x \in F \quad &\Rightarrow \quad \varphi(x^*) \leq \varphi(x) \qquad \text{for opt} = \min \\ x \in F \quad &\Rightarrow \quad \varphi(x^*) \geq \varphi(x) \qquad \text{for opt} = \max \end{aligned}$$

Goal: Optimize φ over F, i.e., decide if $F = \emptyset$, or find an optimal point x^*, or decide that φ is not bounded from below ($\text{opt} = \min$) or above ($\text{opt} = \max$).

Linear Optimization Problem

Definition

An instance of a linear optimization problem or a linear program (LP) is specified by the data:

$m, n \in \mathbb{N}$

Linear Optimization Problem

Definition

An instance of a linear optimization problem or a linear program (LP) is specified by the data:

$$m, n \in \mathbb{N}, \quad a_1, \ldots, a_m \in \mathbb{R}^n$$

Linear Optimization Problem

Definition

An instance of a linear optimization problem or a linear program (LP) is specified by the data:

$$m, n \in \mathbb{N}, \quad a_1, \ldots, a_m \in \mathbb{R}^n, \quad \beta_1, \ldots, \beta_m \in \mathbb{R}$$

Linear Optimization Problem

Definition

An instance of a linear optimization problem or a linear program (LP) is specified by the data:

$$m, n \in \mathbb{N}, \quad a_1, \ldots, a_m \in \mathbb{R}^n, \quad \beta_1, \ldots, \beta_m \in \mathbb{R}, \quad \gamma_1, \ldots, \gamma_n \in \mathbb{R}$$

Linear Optimization Problem

Definition

An instance of a linear optimization problem or a linear program (LP) is specified by the data:

$$m, n \in \mathbb{N}, \quad a_1, \ldots, a_m \in \mathbb{R}^n, \quad \beta_1, \ldots, \beta_m \in \mathbb{R}, \quad \gamma_1, \ldots, \gamma_n \in \mathbb{R}$$

The linear functional $\varphi : \mathbb{R}^n \to \mathbb{R}$ for $x := (\xi_1, \ldots, \xi_n)^T \in \mathbb{R}$ is defined by

$$\varphi(x) := \sum_{i=1}^{n} \gamma_i \xi_i$$

and let

$$F := \{x \in \mathbb{R}^n : a_1^T x \le \beta_1 \ \wedge \ \ldots \ \wedge \ a_m^T x \le \beta_m\}$$

Linear Optimization Problem

Definition

An instance of a linear optimization problem or a linear program (LP) is specified by the data:

$$m, n \in \mathbb{N}, \quad a_1, \ldots, a_m \in \mathbb{R}^n, \quad \beta_1, \ldots, \beta_m \in \mathbb{R}, \quad \gamma_1, \ldots, \gamma_n \in \mathbb{R}$$

The linear functional $\varphi : \mathbb{R}^n \to \mathbb{R}$ for $x := (\xi_1, \ldots, \xi_n)^T \in \mathbb{R}$ is defined by

$$\varphi(x) := \sum_{i=1}^{n} \gamma_i \xi_i$$

and let

$$F := \{x \in \mathbb{R}^n : a_1^T x \leq \beta_1 \ \wedge \ \ldots \ \wedge \ a_m^T x \leq \beta_m\}$$

Goal: Maximize φ over F.

Linear Optimization Problem

Notation

Let $(m, n, a_1, \ldots, a_m, \beta_1, \ldots, \beta_m, \gamma_1, \ldots, \gamma_n)$ be an instance of a linear optimization problem. Set

$$A := (a_1, \ldots, a_m)^T \in \mathbb{R}^{m \times n}$$
$$b := (\beta_1, \ldots, \beta_m)^T \in \mathbb{R}^m$$
$$c := (\gamma_1, \ldots, \gamma_n)^T \in \mathbb{R}^n$$

Then we often write

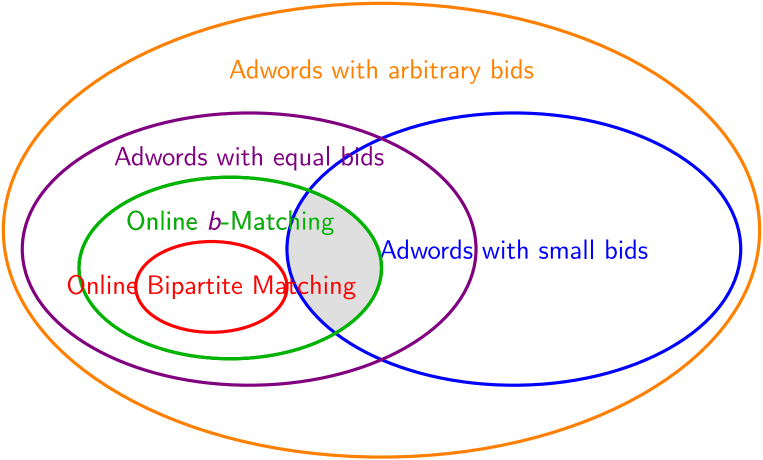
$$\max \ c^T x$$
$$Ax \leq b$$

Outline

Assumptions

1. $\forall (u, v) \in E : c_{u,v} \leq \frac{1}{k^2}$.
2. Each bidder has a budget of 1.
3. OPT exhausts the budget of each bidder.
4. Each bidder of type $i \in \{1, ..., k\}$ spends exactly $\frac{i}{k}$.
5. All non-zero bids are equal.

Landscape of Problems



A Linear Program for BALANCE

Lemma

Let $v \in V$. If OPT assigns query v to a bidder of type $i \in \{1, \ldots, k-1\}$, then Balance gets the money for v from some slab j such that $1 \leq j \leq i$.

A Linear Program for BALANCE

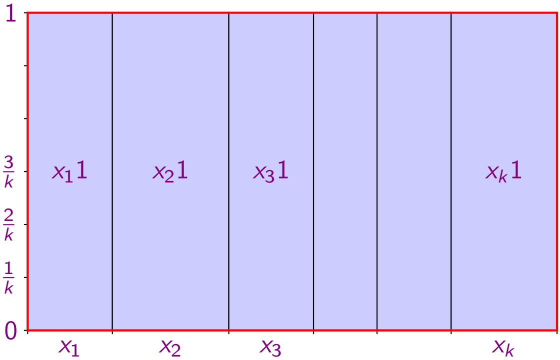
Lemma

Let $v \in V$. If OPT assigns query v to a bidder of type $i \in \{1, \ldots, k-1\}$, then Balance gets the money for v from some slab j such that $1 \le j \le i$.

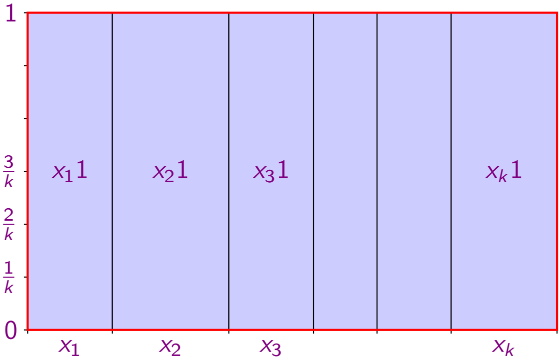
Lemma

$$\forall i \in \{1, \ldots, k-1\} : \sum_{j=1}^{i} x_j \le \sum_{j=1}^{i} \beta_i$$

Revenue Calculation for OPT



Revenue Calculation for OPT



$$\pi_{\text{OPT}} = \sum_{i=1}^{k} x_i = n$$

A Linear Program for BALANCE

Lemma

$$\forall i \in \{1, \ldots, k-1\} : \sum_{j=1}^{i} \left(1 + \frac{i-j}{k}\right) x_j \leq \frac{i}{k} n$$

A Linear Program for BALANCE

$$\pi_{\text{BALANCE}} \geq \sum_{i=1}^{k} x_i \frac{i}{k} - \frac{n}{k}$$

A Linear Program for BALANCE

$$\pi_{\text{BALANCE}} \geq \sum_{i=1}^{k} x_i \frac{i}{k} - \frac{n}{k}$$

$$= \sum_{i=1}^{k-1} \frac{i}{k} x_i + x_k - \frac{n}{k}$$

A Linear Program for BALANCE

$$\begin{aligned}
\pi_{\text{BALANCE}} &\geq \sum_{i=1}^{k} x_i \frac{i}{k} - \frac{n}{k} \\
&= \sum_{i=1}^{k-1} \frac{i}{k} x_i + x_k - \frac{n}{k} \\
&= \sum_{i=1}^{k-1} \frac{i}{k} x_i + \left(n - \sum_{i=1}^{k-1} x_i \right) - \frac{n}{k}
\end{aligned}$$

A Linear Program for BALANCE

$$\begin{aligned}
\pi_{\text{BALANCE}} &\geq \sum_{i=1}^{k} x_i \frac{i}{k} - \frac{n}{k} \\
&= \sum_{i=1}^{k-1} \frac{i}{k} x_i + x_k - \frac{n}{k} \\
&= \sum_{i=1}^{k-1} \frac{i}{k} x_i + \left(n - \sum_{i=1}^{k-1} x_i \right) - \frac{n}{k} \\
&= n - \sum_{i=1}^{k-1} \left(1 - \frac{i}{k} \right) x_i - \frac{n}{k}
\end{aligned}$$

A Linear Program for BALANCE

$$
\begin{aligned}
\pi_{\text{BALANCE}} &\geq \sum_{i=1}^{k} x_i \frac{i}{k} - \frac{n}{k} \\
&= \sum_{i=1}^{k-1} \frac{i}{k} x_i + x_k - \frac{n}{k} \\
&= \sum_{i=1}^{k-1} \frac{i}{k} x_i + \left(n - \sum_{i=1}^{k-1} x_i \right) - \frac{n}{k} \\
&= n - \sum_{i=1}^{k-1} \left(1 - \frac{i}{k} \right) x_i - \frac{n}{k} \\
&= n - \sum_{i=1}^{k-1} \frac{k-i}{k} x_i - \frac{n}{k}
\end{aligned}
$$

A Linear Program for BALANCE

Notation

The linear program (P) is defined by

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{k-1} \frac{k-i}{k} x_i \\ \text{subject to} \quad & \sum_{j=1}^{i} \left(1 + \frac{i-j}{k}\right) x_j \leq \frac{i}{k} n \quad \forall i \in \{1, \ldots, k-1\} \\ & x_i \geq 0 \quad \forall i \in \{1, \ldots, k-1\} \end{aligned}$$

Duality

Notation

The two linear programs (I) and (II) are defined by

$$
\text{(I)} \quad \begin{aligned} \max\ & c^T x \\ Ax &\le b \\ x &\ge 0 \end{aligned}
\qquad\qquad
\text{(II)} \quad \begin{aligned} \min\ & b^T y \\ A^T y &\ge c \\ y &\ge 0 \end{aligned}
$$

Let

$$
\begin{aligned}
P &:= \{x \in \mathbb{R}^n : Ax \le b \ \wedge\ x \ge 0\} \\
Q &:= \{y \in \mathbb{R}^m : A^T y \ge c \ \wedge\ y \ge 0\}
\end{aligned}
$$

and $\zeta_{(\mathrm{I})}$, $\zeta_{(\mathrm{II})}$ be the optimum values of (I), (II) respectively.

Duality

Definition

The linear programs (I) and (II) are called dual of each other. Often one is called primal, the other dual.

Duality

Theorem

The following statements hold:

Duality

Theorem

The following statements hold:

1. $\zeta_{(I)} \leq \zeta_{(II)}$

Duality

Theorem

The following statements hold:

1. $\zeta_{(I)} \leq \zeta_{(II)}$
2. $P \neq \emptyset \lor Q \neq \emptyset \quad \Rightarrow \quad \zeta_{(I)} = \zeta_{(II)}$

Duality

Theorem

The following statements hold:

1. $\zeta_{(I)} \leq \zeta_{(II)}$
2. $P \neq \emptyset \vee Q \neq \emptyset \quad \Rightarrow \quad \zeta_{(I)} = \zeta_{(II)}$
3. *Let* $x^* \in P, y^* \in Q$. *Then*

$$\begin{aligned} & c^T x^* = b^T y^* \\ \Longleftrightarrow\ & \zeta_{(I)} = c^T x^* \ \wedge \ \zeta_{(II)} = b^T y^* \\ \Longleftrightarrow\ & (y^*)^T (b - Ax^*) = 0 \ \wedge \ (x^*)^T \left(c - A^T y^*\right) = 0 \end{aligned}$$

A Solution to the Linear Program

Notation

The dual linear program of (P), (D), is defined by

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{i=1}^{k-1} \frac{i}{k} n y_i & \\ \text{subject to} & \displaystyle\sum_{j=i}^{k-1} \left(1 + \frac{j-i}{k}\right) y_j \geq \frac{k-i}{k} & \forall i \in \{1, \ldots, k-1\} \\ & y_i \geq 0 & \forall i \in \{1, \ldots, k-1\} \end{array}$$

A Solution to the Linear Program

Lemma

$$x_i^* = \frac{n}{k}\left(1 - \frac{1}{k}\right)^{i-1} \quad \text{for } i = 1, \ldots, k-1$$

is a solution to the system

$$\sum_{j=1}^{i}\left(1 + \frac{i-j}{k}\right) x_j = \frac{i}{k} n \quad \forall i \in \{1, \ldots, k-1\}$$

A Solution to the Linear Program

Lemma

$$y_i^* = \frac{1}{k}\left(1 - \frac{1}{k}\right)^{k-i-1} \quad \text{for } i = 1, \ldots, k-1$$

is a solution to the system

$$\sum_{j=i}^{k-1} \left(1 + \frac{j-i}{k}\right) y_j = \frac{k-i}{k} \quad \forall i \in \{1, \ldots, k-1\}$$

A Solution to the Linear Program

Lemma

The optimum objective function value of the LP (P) is $n\left(1-\frac{1}{k}\right)^k$.

A Solution to the Linear Program

Lemma

The optimum objective function value of the LP (P) is $n\left(1-\frac{1}{k}\right)^k$.

Proof.

$$x_i^* = \frac{n}{k}\left(1-\frac{1}{k}\right)^{i-1} \quad \text{for } i = 1, \ldots, k-1$$

and

$$y_i^* = \frac{1}{k}\left(1-\frac{1}{k}\right)^{k-i-1} \quad \text{for } i = 1, \ldots, k-1$$

are feasible solutions of the primal and dual programs.

A Solution to the Linear Program

Proof (continued).

By construction, they clearly satisfy

$$(y^*)^T(b - Ax^*) = 0 \ \wedge \ (x^*)^T\left(c - A^T y^*\right) = 0$$

A Solution to the Linear Program

Proof (continued).

By construction, they clearly satisfy

$$(y^*)^T (b - Ax^*) = 0 \wedge (x^*)^T \left(c - A^T y^*\right) = 0$$

Hence, they are also optimum solutions of the primal and dual programs.

A Solution to the Linear Program

Proof (continued).

By construction, they clearly satisfy

$$\left(y^{*}\right)^{T}\left(b-Ax^{*}\right)=0 \wedge \left(x^{*}\right)^{T}\left(c-A^{T}y^{*}\right)=0$$

Hence, they are also optimum solutions of the primal and dual programs.

This gives an optimum objective function value of

$$c^{T}x^{*}=\sum_{i=1}^{k-1}\frac{k-i}{k}\frac{n}{k}\left(1-\frac{1}{k}\right)^{i-1}$$

A Solution to the Linear Program

Proof (continued).

By construction, they clearly satisfy

$$(y^*)^T (b - Ax^*) = 0 \;\wedge\; (x^*)^T \left(c - A^T y^*\right) = 0$$

Hence, they are also optimum solutions of the primal and dual programs.

This gives an optimum objective function value of

$$c^T x^* = \sum_{i=1}^{k-1} \frac{k-i}{k} \frac{n}{k} \left(1 - \frac{1}{k}\right)^{i-1} = n \left(1 - \frac{1}{k}\right)^k$$

$\square$

Competitive Ratio of BALANCE for b-Matching

No randomized online algorithm can have a competitive ratio better than $1 - \frac{1}{e}$ for the b-Matching problem, for large b.

Theorem

BALANCE achieves a competitive ratio of $1 - \frac{1}{e}$ for the b-Matching problem, for large b.

Competitive Ratio of BALANCE for b-Matching

No randomized online algorithm can have a competitive ratio better than $1 - \frac{1}{e}$ for the b-Matching problem, for large b.

Theorem

BALANCE achieves a competitive ratio of $1 - \frac{1}{e}$ for the b-Matching problem, for large b.

Theorem

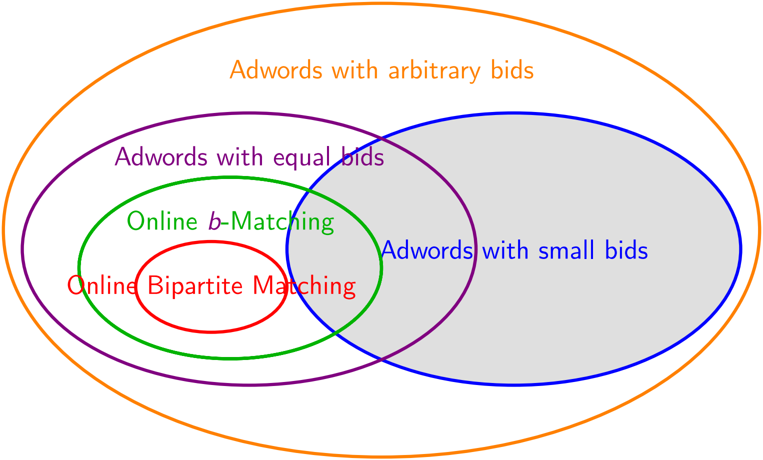
No randomized online algorithm can have a competitive ratio better than $1 - \frac{1}{e}$ for the b-Matching problem, for large b.

Outline

Assumptions

1. Each bid is small compared to the corresponding budget (i.e., $\forall (u, v) \in E : c_{u,v} \ll b_u$).
2. Each bidder has a budget of 1.
3. OPT exhausts the budget of each bidder.
4. Each bidder of type $i \in \{1, ..., k\}$ spends exactly $\frac{i}{k}$.
5. ~~All non-zero bids are equal.~~

Landscape of Problems



An Inequality Constraint for MSVV

Notation

Let ALG $\in \{\text{OPT}, \text{MSVV}\}, v \in V$. Then $u_{\text{ALG}}(v)$ denotes the bidder ALG assigns query v to.

An Inequality Constraint for MSVV

Notation

Let $\text{ALG} \in \{\text{OPT}, \text{MSVV}\}, v \in V$. Then $u_{\text{ALG}}(v)$ denotes the bidder ALG assigns query v to.

Lemma

For all queries $v \in V$ *such that* $type(u_{OPT}(v)) \in \{1, \ldots, k-1\}$:

$$c_{u_{OPT}(v),v}\psi_k\left(type(u_{OPT}(v)\right) \leq c_{u_{MSVV}(v),v}\psi_k\left(slab(u_{MSVV}(v)\right)$$

An Inequality Constraint for MSVV

Lemma

$$\sum_{i=1}^{k-1} \psi_k(i) x_i \leq \sum_{i=1}^{k-1} \psi_k(i) \beta_i + \frac{n}{k}$$

Competitive Ratio for MSVV for Adwords with Small Bids

Theorem

MSVV achieves a competitive ratio of $1 - \frac{1}{e}$ *for the Adwords problem with small bids.*

Competitive Ratio for MSVV for Adwords with Small Bids

Proof.

We can use

$$\forall i \in \{1, \dots, k\} : \beta_i = \frac{n - \sum_{j=1}^{i-1} x_j}{k}$$

and the explicit form of the perturbation function $\varphi(x) = 1 - e^{x-1}$ in

$$\sum_{i=1}^{k-1} \psi_k(i) x_i \leq \sum_{i=1}^{k-1} \psi_k(i) \beta_i + \frac{n}{k}$$

Competitive Ratio for MSVV for Adwords with Small Bids

Proof (continued).

to get

$$\sum_{i=1}^{k} x_i \frac{i}{k} - \frac{n}{k} \geq n\left(1 - \frac{1}{e}\right), \quad \text{as } k \to \infty$$

But the left hand side is a lower bound on π_{MSVV}, thus proving the theorem. $\square$

Outline

Assumptions

1. Each bid is small compared to the corresponding budget (i.e., $\forall(u, v) \in E : c_{u,v} \ll b_u$).
2. Each bidder has a budget of 1.
3. OPT exhausts the budget of each bidder.
4. Each bidder of type $i \in \{1, ..., k\}$ spends exactly $\frac{i}{k}$.
5. ~~All non-zero bids are equal.~~

Assumptions

1. Each bid is small compared to the corresponding budget (i.e., $\forall(u, v) \in E : c_{u,v} \ll b_u$).
2. ~~Each bidder has a budget of 1.~~
3. OPT exhausts the budget of each bidder.
4. Each bidder of type $i \in \{1, ..., k\}$ spends exactly $\frac{i}{k}$.
5. ~~All non-zero bids are equal.~~

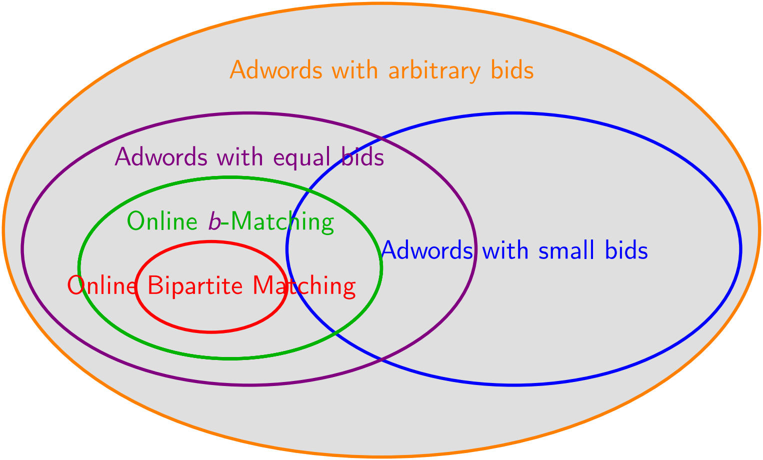
Assumptions

1. Each bid is small compared to the corresponding budget (i.e., $\forall (u, v) \in E : c_{u,v} \ll b_u$).
2. ~~Each bidder has a budget of 1.~~
3. ~~OPT exhausts the budget of each bidder.~~
4. Each bidder of type $i \in \{1, ..., k\}$ spends exactly $\frac{i}{k}$.
5. ~~All non-zero bids are equal.~~

Assumptions

1. ~~Each bid is small compared to the corresponding budget (i.e., $\forall (u, v) \in E : c_{u,v} \ll b_u$).~~
2. ~~Each bidder has a budget of 1.~~
3. ~~OPT exhausts the budget of each bidder.~~
4. Each bidder of type $i \in \{1, ..., k\}$ spends exactly $\frac{i}{k}$.
5. ~~All non-zero bids are equal.~~

Landscape of Problems



Adwords with Arbitrary Bids

Theorem

GREEDY achieves a competitive ratio of $\frac{1}{2}$ for the Adwords problem.

Adwords with Arbitrary Bids

Theorem

GREEDY achieves a competitive ratio of $\frac{1}{2}$ for the Adwords problem.

Open Question

Find an algorithm which beats the competitive ratio of $\frac{1}{2}$ for the Adwords problem with arbitrary bids.