Online Learning and Collusion in Multi-unit Auctions

Simina Branzei (Purdue)

SLMath Randomization, Neutrality, and Fairness Workshop

Joint with

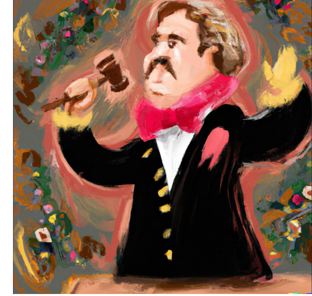
Mahsa Derakhshan (Northeastern), **Negin Golrezaei** (MIT),

and **Yanjun Han** (New York University)

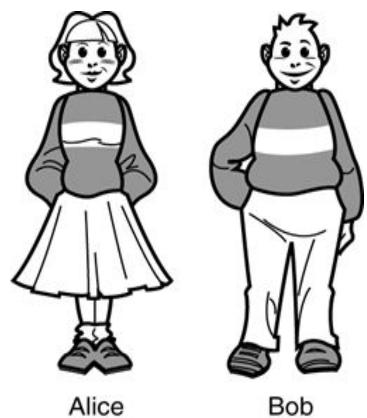
Multi-unit Auction

Seller brings multiple identical units of a good (e.g. chairs)

Set of buyers have money and may be interested in purchasing the goods.



Buyers



Multi-unit Auction – Model

- K units of a good for sale;

Multi-unit Auction – Model

- K units of a good for sale; Set $[n] = \{1, \dots, n\}$ of players with money.

Multi-unit Auction – Model

- K units of a good for sale; Set $[n] = \{1, \dots, n\}$ of players with money. Each player i has private value $v_{i,j}$ for getting a j-th unit in their bundle.

Multi-unit Auction – Model

- K units of a good for sale; Set $[n] = \{1, \dots, n\}$ of players with money. Each player i has private value $v_{i,j}$ for getting a j-th unit in their bundle. Value of player i for j units is $V_i(j) = \sum_{\ell=1}^{j} v_{i,\ell}$.

Multi-unit Auction – Model

- K units of a good for sale; Set $[n] = \{1, \dots, n\}$ of players with money. Each player i has private value $v_{i,j}$ for getting a j-th unit in their bundle. Value of player i for j units is $V_i(j) = \sum_{\ell=1}^{j} v_{i,\ell}$.

Diminishing marginal returns: $v_{i,1} \geq \cdots \geq v_{i,K}$.

Multi-unit Auction – Model

- K units of a good for sale; Set $[n] = \{1, \ldots, n\}$ of players with money. Each player i has private value $v_{i,j}$ for getting a j-th unit in their bundle. Value of player i for j units is $V_i(j) = \sum_{\ell=1}^{j} v_{i,\ell}$. Diminishing marginal returns: $v_{i,1} \geq \cdots \geq v_{i,K}$.

Multi-unit Auction – Model

Auction format: the Walrasian mechanism.

Multi-unit Auction – Model

Reported values (Declared willingness to pay)

- **Auction format:** the Walrasian mechanism.

Each player i submits bids $b_i = \left(b_{i,1}, \ldots, b_{i,K}\right)$.

Multi-unit Auction – Model

Reported values (Declared willingness to pay)

- **Auction format:** the Walrasian mechanism.

Each player i submits bids $b_i = \left(b_{i,1}, \dots, b_{i,K}\right)$.

Auctioneer sorts the bids from highest to lowest,

Multi-unit Auction – Model

Reported values (Declared willingness to pay)

- **Auction format:** the Walrasian mechanism.

Each player i submits bids $b_i = \left(b_{i,1}, \dots, b_{i,K}\right)$.

Auctioneer sorts the bids from highest to lowest, sets a price p

per unit

Multi-unit Auction – Model

Reported values (Declared willingness to pay)

- **Auction format:** the Walrasian mechanism.

Each player i submits bids $b_i = \left(b_{i,1}, \dots, b_{i,K}\right)$.

Auctioneer sorts the bids from highest to lowest, sets a price p per unit, and allocates the j-th unit to the player that submitted the j-th highest bid charging them p for that unit

Multi-unit Auction – Model

Reported values (Declared willingness to pay)

- **Auction format:** the Walrasian mechanism.

Each player i submits bids $b_i = \left(b_{i,1}, \dots, b_{i,K}\right)$.

Auctioneer sorts the bids from highest to lowest, sets a price p per unit, and allocates the j-th unit to the player that submitted the j-th highest bid charging them p for that unit (lexicographic tie breaking).

Multi-unit Auction – Model

Reported values (Declared willingness to pay)

- **Auction format:** the Walrasian mechanism.

Each player i submits bids $b_i = (b_{i,1}, \ldots, b_{i,K})$.

Auctioneer sorts the bids from highest to lowest, sets a price p per unit, and allocates the j-th unit to the player that submitted the j-th highest bid charging them p for that unit (lexicographic tie breaking). Player i's utility = value – price (for bundle received)

Multi-unit Auction – Model

Reported values (Declared willingness to pay)

- **Auction format:** the Walrasian mechanism.

Each player i submits bids $b_i = \left(b_{i,1}, \ldots, b_{i,K}\right)$.

Auctioneer sorts the bids from highest to lowest, sets a price p per unit, and allocates the j-th unit to the player that submitted the j-th highest bid charging them p for that unit (lexicographic tie breaking). Player i's utility = value – price (for bundle received)

Two versions of the auction:

Multi-unit Auction – Model

Reported values (Declared willingness to pay)

- **Auction format:** the Walrasian mechanism.

Each player i submits bids $b_i = (b_{i,1}, \dots, b_{i,K})$.

Auctioneer sorts the bids from highest to lowest, sets a price p per unit, and allocates the j-th unit to the player that submitted the j-th highest bid charging them p for that unit (lexicographic tie breaking). Player i's utility = value – price (for bundle received)

Max Walrasian price

Two versions of the auction:

Multi-unit Auction – Model

Reported values (Declared willingness to pay)

- **Auction format:** the Walrasian mechanism.

Each player i submits bids $b_i = (b_{i,1}, \ldots, b_{i,K})$.

Auctioneer sorts the bids from highest to lowest, sets a price p per unit, and allocates the j-th unit to the player that submitted the j-th highest bid charging them p for that unit (lexicographic tie breaking). Player i's utility = value – price (for bundle received)

Two versions of the auction:

Multi-unit Auction – Model

Reported values (Declared willingness to pay)

- **Auction format:** the Walrasian mechanism.

Each player i submits bids $b_i = (b_{i,1}, \ldots, b_{i,K})$.

Auctioneer sorts the bids from highest to lowest, sets a price p per unit, and allocates the j-th unit to the player that submitted the j-th highest bid charging them p for that unit (lexicographic tie breaking). Player i's utility = value – price (for bundle received)

Two versions of the auction:

Min Walrasian price

Multi-unit Auction – Example

- **Example:** $K = 2$ units and $n = 2$ players (**Alice** and **Bob**).

Valuations: $v_{Alice} = (5, 2)$ and $v_{Bob} = (4, 1)$.

Multi-unit Auction – Example

- **Example:** $K = 2$ units and $n = 2$ players (**Alice** and **Bob**).

Valuations: $v_{Alice} = (5, 2)$ and $v_{Bob} = (4, 1)$.

Bids: Alice submits bids $b_{Alice} = (3, 1)$ and Bob submits

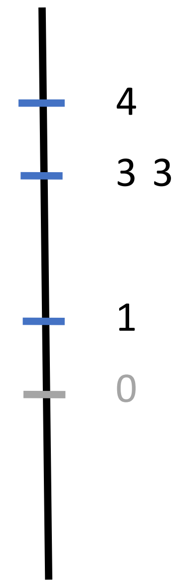
$b_{Bob} = (4, 3)$.

Multi-unit Auction – Example

- **Example:** $K = 2$ units and $n = 2$ players (**Alice** and **Bob**).

Valuations: $v_{Alice} = (5, 2)$ and $v_{Bob} = (4, 1)$.

Bids: Alice submits bids $b_{Alice} = (3, 1)$ and Bob submits $b_{Bob} = (4, 3)$. The sorted bids are $(4, 3, 3, 1)$.



Multi-unit Auction – Example

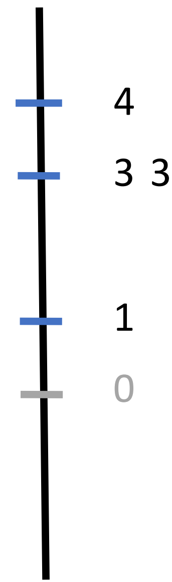
- **Example:** $K = 2$ units and $n = 2$ players (**Alice** and **Bob**).

Valuations: $v_{Alice} = (5, 2)$ and $v_{Bob} = (4, 1)$.

Bids: Alice submits bids $b_{Alice} = (3, 1)$ and Bob submits $b_{Bob} = (4, 3)$. The sorted bids are $(4, 3, 3, 1)$.

Allocation: the first unit is allocated to Bob and the second to Alice.

- K-th price: Price per unit is $p = 3$.



Multi-unit Auction – Example

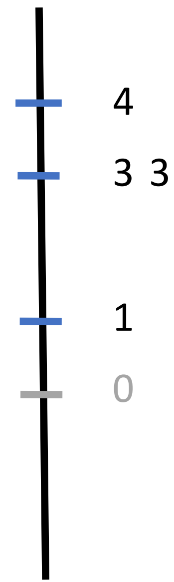
- **Example:** $K = 2$ units and $n = 2$ players (**Alice** and **Bob**).

Valuations: $v_{Alice} = (5, 2)$ and $v_{Bob} = (4, 1)$.

Bids: Alice submits bids $b_{Alice} = (3, 1)$ and Bob submits $b_{Bob} = (4, 3)$. The sorted bids are $(4, 3, 3, 1)$.

Allocation: the first unit is allocated to Bob and the second to Alice.

- K-th price: Price per unit is $p = 3$.

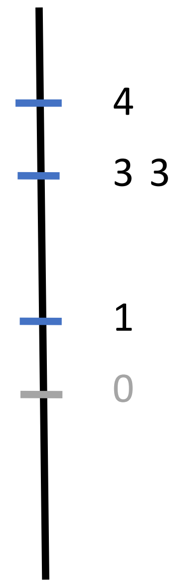


Multi-unit Auction – Example

- **Example:** $K = 2$ units and $n = 2$ players (**Alice** and **Bob**).

Valuations: $v_{Alice} = (5, 2)$ and $v_{Bob} = (4, 1)$.

Bids: Alice submits bids $b_{Alice} = (3, 1)$ and Bob submits $b_{Bob} = (4, 3)$. The sorted bids are $(4, 3, 3, 1)$.



Allocation: the first unit is allocated to Bob and the second to Alice.

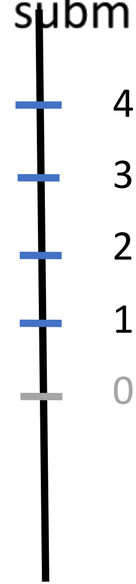
- $(K + 1)$-st price: Price per unit is $p = 3$.

Multi-unit Auction – Example

- **Example:** $K = 2$ units and $n = 2$ players (**Alice** and **Bob**).

Valuations: $v_{Alice} = (5, 2)$ and $v_{Bob} = (4, 1)$.

Bids: If instead Alice submits bids $b_{Alice} = (3, 1)$ and Bob submits $b_{Bob} = (4, 2)$. The sorted bids are $(4, 3, 2, 1)$.



Multi-unit Auction – Example

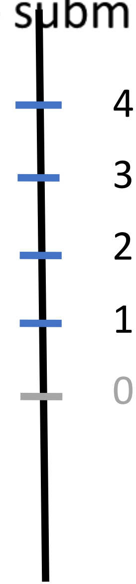
- **Example:** $K = 2$ units and $n = 2$ players (**Alice** and **Bob**).

Valuations: $v_{Alice} = (5, 2)$ and $v_{Bob} = (4, 1)$.

Bids: If instead Alice submits bids $b_{Alice} = (3, 1)$ and Bob submits $b_{Bob} = (4, 2)$. The sorted bids are $(4, 3, 2, 1)$.

Allocation: the first unit is allocated to Bob and the second to Alice.

- K-th price: Price per unit is $p = 3$.

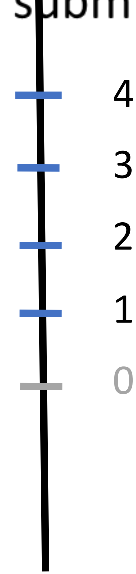


Multi-unit Auction – Example

- **Example:** $K = 2$ units and $n = 2$ players (**Alice** and **Bob**).

Valuations: $v_{Alice} = (5, 2)$ and $v_{Bob} = (4, 1)$.

Bids: If instead Alice submits bids $b_{Alice} = (3, 1)$ and Bob submits $b_{Bob} = (4, 2)$. The sorted bids are $(4, 3, 2, 1)$.



Allocation: the first unit is allocated to Bob and the second to Alice.

- $(K + 1)$-st price: Price per unit is $p = 2$.

This work: Repeated Auction

This work: Repeated Auction

- In each round $t = 1, 2, \ldots,$ the next steps take place:

This work: Repeated Auction

- In each round $t = 1, 2, \ldots$, the next steps take place:

  The auctioneer announces K units for sale.

This work: Repeated Auction

- In each round $t = 1, 2, \ldots,$ the next steps take place:

The auctioneer announces K units for sale.

Each player i privately submits bids $b_i^t = \left(b_{i,1}^t, \ldots, b_{i,K}^t\right)$, where

$b_{i,j}^t$ is player i's bid for a j-th unit at time t.

This work: Repeated Auction

- In each round $t = 1, 2, \dots,$ the next steps take place:

  The auctioneer announces K units for sale.

  Each player i privately submits bids $b_i^t = \left(b_{i,1}^t, \dots, b_{i,K}^t\right)$, where $b_{i,j}^t$ is player i's bid for a j-th unit at time t.

  Auctioneer runs the auction with bids $b^t = \left(b_1^t, \dots, b_n^t\right)$ and reveals information (feedback) about the outcome to the players.

This work: Repeated Auction

- In each round $t = 1, 2, \ldots,$ the next steps take place:

  The auctioneer announces K units for sale.

  Each player i privately submits bids $b_i^t = \left(b_{i,1}^t, \ldots, b_{i,K}^t\right)$, where $b_{i,j}^t$ is player i's bid for a j-th unit at time t.

  Auctioneer runs the auction with bids $b^t = \left(b_1^t, \ldots, b_n^t\right)$ and reveals information (feedback) about the outcome to the players.

Auctioneer can decide to hide some information from a player

Example

- Each quarter the government sells a number K of licenses, each for the right to emit 1 ton of $CO2$.

Example

- Each quarter the government sells a number K of licenses, each for the right to emit 1 ton of $CO2$.
- The players submit bids, then the auction is run and the licenses are allocated.

Example

- Each quarter the government sells a number K of licenses, each for the right to emit 1 ton of $CO2$.
- The players submit bids, then the auction is run and the licenses are allocated.
- The format used in practice is based on the

Related Work

Large body of literature on multi-unit auctions (e.g. Maskin-Riley-Hahn ’89; Engelbrecht-Kahn ‘98; Borgs-Chayes-Immorlica-Mahdian-Saberi ’05; J. Hartline, A. Karlin, D. Kempe, C. Kenyon, and F. McSherry ‘05; Dobzinski-Nisan ’07; Dobzinski-Lavi-Nisan ’12; Feldman-Fiat-Leonardi-Sankowski ’12; Goel-Mirrokni-Leme ‘13; Ausubel-Cramton-Pycia-Rostek-Weretka ’14).

Repeated multi-unit auctions: allocating carbon licenses (Cramton-Kerr ’02; ads in online settings (Balseiro-Besbes-Weintraub 15); US treasury notes (Markakis-Telelis 15).

Related Work

Repeated auctions and quality of outcomes reached (Lucier-Borodin ’10; Bhawalkar and Roughgarden ’11; Daskalakis-Syrganis 16; Nedelec-Calauzènes-Karoui-Perchet ‘22)

Outline

- Offline Setting
- Online Setting
- Collusion
- Discussion

Outline

- Offline Setting
- Online Setting
- Collusion
- Discussion

The Offline Problem

Input:

- a player i with valuation v_i and
- a history $H_{-i} = (b^1_{-i}, b^2_{-i}, \ldots, b^T_{-i})$ containing the bids of the other players in rounds 1 through T.

The offline problem is: What is the best response (bid vector) for player i given the historical data? That is:

The Offline Problem: (K+1)-st price example

- Suppose $K = 2$ and Alice's valuation is $v_{Alice} = (3.1, 2.99)$. Suppose $T = 2$ and $n = 3$. The history of bids of the other players (say Bob and Carol) are depicted below.

4

3

2

1.5, 1.5

1, 1

1

0

0

The Offline Problem: (K+1)-st price example

- Suppose $K = 2$ and Alice's valuation is $v_{Alice} = (3.1, 2.99)$. Suppose $T = 2$ and $n = 3$. The history of bids of Bob and Carol are depicted below. If Alice bids $b = v_{Alice} = (3.1, 2.99)$:

4

3.1

3.1

3

2.99

2.99

2

1.5, 1.5

1, 1

1

0

0

The Offline Problem: (K+1)-st price example

- Suppose $K = 2$ and Alice's valuation is $v_{Alice} = (3.1, 2.99)$. Suppose $T = 2$ and $n = 3$. The history of bids of Bob and Carol are depicted below. If Alice bids $b = v_{Alice} = (3.1, 2.99)$: she gets 1 unit and pays 2.99 in each round, with utility $3.1 - 2.99 = 0.11$.

4

3.1

3.1

3

2.99

2.99

2

1.5, 1.5

1, 1

1

0

0

The Offline Problem: (K+1)-st price example

- Suppose $K = 2$ and Alice's valuation is $v_{Alice} = (3.1, 2.99)$. Suppose $T = 2$ and $n = 3$. The history of bids of Bob and Carol are depicted below. If Alice bid $b = (3.1, 2)$:

4

3.1 3.1

3

2 2 2

1.5, 1.5

1, 1 1

0 0

The Offline Problem: (K+1)-st price example

- Suppose $K = 2$ and Alice's valuation is $v_{Alice} = (3.1, 2.99)$. Suppose $T = 2$ and $n = 3$. The history of bids of Bob and Carol are depicted below. If Alice bid $b = (3.1, 2)$: she still gets 1 unit each round but pays only 2 each round, with utility $3.1 - 2 = 1.1 > 0.11$ each round (better than before)

4

3.1

3.1

3

2

2

2

1.5, 1.5

1, 1

1

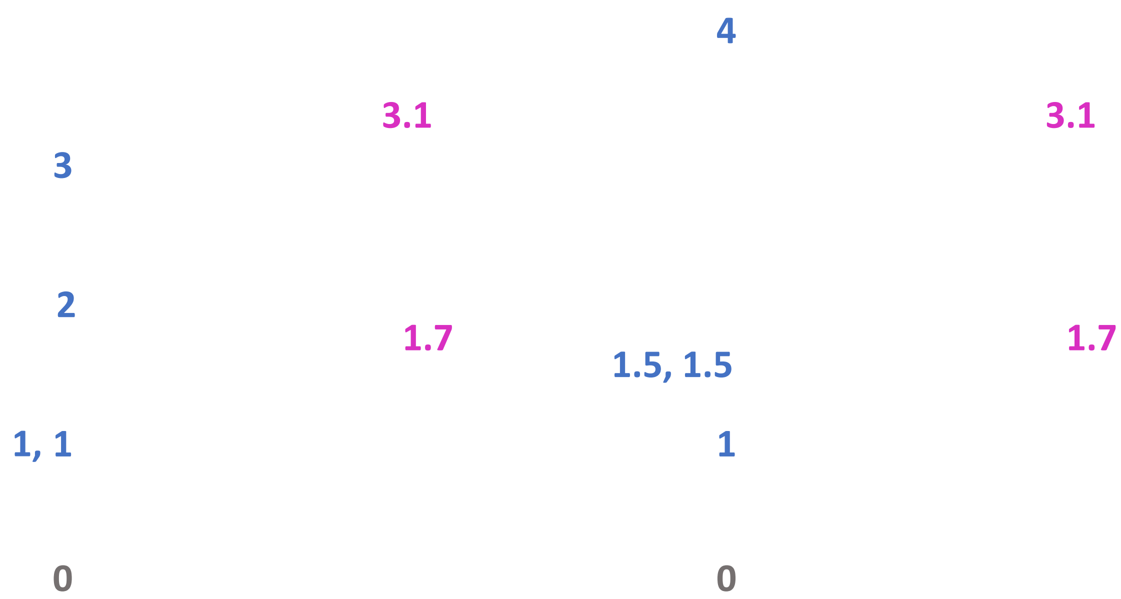
0

0

The Offline Problem: (K+1)-st price example

- Suppose $K = 2$ and Alice's valuation is $v_{Alice} = (3.1, 2.99)$. Suppose $T = 2$ and $n = 3$. The history of bids of Bob and Carol are depicted below.

Another possible bid vector of Alice is $b = (3.1, 1.7)$:



The Offline Problem: (K+1)-st price example

- Suppose $K = 2$ and Alice's valuation is $v_{Alice} = (3.1, 2.99)$. Suppose $T = 2$ and $n = 3$. The history of bids of Bob and Carol are depicted below.

Another possible bid vector of Alice is $b = (3.1, 1.7)$: then in round 1 Alice gets 0 units and pays 0; in round 2, Alice gets 1 unit and pays 1.7.

4

3.1 3.1

3

2

1.7 1.7

1.5, 1.5

1, 1 1

0 0

The Offline Problem

- For now we can assume the bids are restricted to discrete domain

$$D = \{\ell \cdot \epsilon \mid \ell \in N\}.$$

Observation: There is an optimum bid vector for player i in the following set of “candidate” bids:

$$S_i = \{0\} \cup \{b_{j,k}^{t} \mid j \in [n] \setminus \{i\}, k \in [K]\} \cup$$

Algorithm

- **Theorem:** There is a polynomial time algorithm for the offline problem.

Input: Player i with valuation v_i and history $H_{-i} = (b^1_{-i}, b^2_{-i}, \dots, b^T_{-i})$ with the bids of other players in rounds 1 through T.

Output: Bid vector b_i^* with $b_i^* = \arg\max_{\beta} \sum_{t=1}^{T} u_i(\beta, b^t_{-i})$.

Algorithm

- **Theorem:** There is a polynomial time algorithm for the offline problem.

Input: Player i with valuation v_i and history $H_{-i} = \left(b^1_{-i}, b^2_{-i}, \ldots, b^T_{-i}\right)$ with the bids of other players in rounds 1 through T.

Output: Bid vector b_i^* with $b_i^* = \arg\max_{\beta} \sum_{t=1}^{T} u_i\left(\beta, b^t_{-i}\right)$.

Algorithm

Theorem: There is a polynomial time algorithm for the offline problem.

Proof sketch:

Algorithm

- **Theorem:** There is a polynomial time algorithm for the offline problem.

Proof sketch: For any bid vector β, write the cumulative utility of the player when playing β while the others play according to H_{-i}.

Algorithm

- **Theorem:** There is a polynomial time algorithm for the offline problem.

Proof sketch: For any bid vector β, write the cumulative utility of the player when playing β while the others play according to H_{-i}. Then observe it can be rewritten as a sum of K terms, so that each term j only depends on j, β_j and β_{j+1}.

Algorithm

- **Theorem:** There is a polynomial time algorithm for the offline problem.

Proof sketch: For any bid vector β, write the cumulative utility of the player when playing β while the others play according to H_{-i}. Then observe it can be rewritten as a sum of K terms, so that each term j only depends on j, β_j and β_{j+1}. Create layered DAG where vertices are tuples of the form (β_j, j).

Algorithm

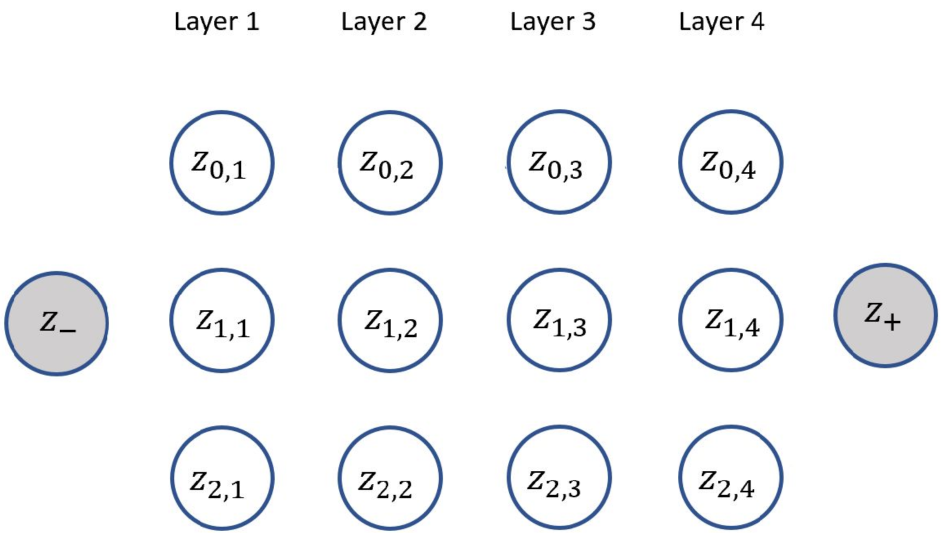
- **Theorem:** There is a polynomial time algorithm for the offline problem.

Proof sketch: For any bid vector β, write the cumulative utility of the player when playing β while the others play according to H_{-i}. Then observe it can be rewritten as a sum of K terms, so that each term j only depends on j, β_j and β_{j+1}. Create layered DAG where vertices are tuples of the form (β_j, j). Make term j

DAG G_i for the offline problem of player i

- **Vertices:** Create a vertex $z_{s,j}$ for each $s \in S_i$ and $j \in [K]$. We say vertex $z_{s,j}$ is in layer j. Add source z_- and sink z_+.

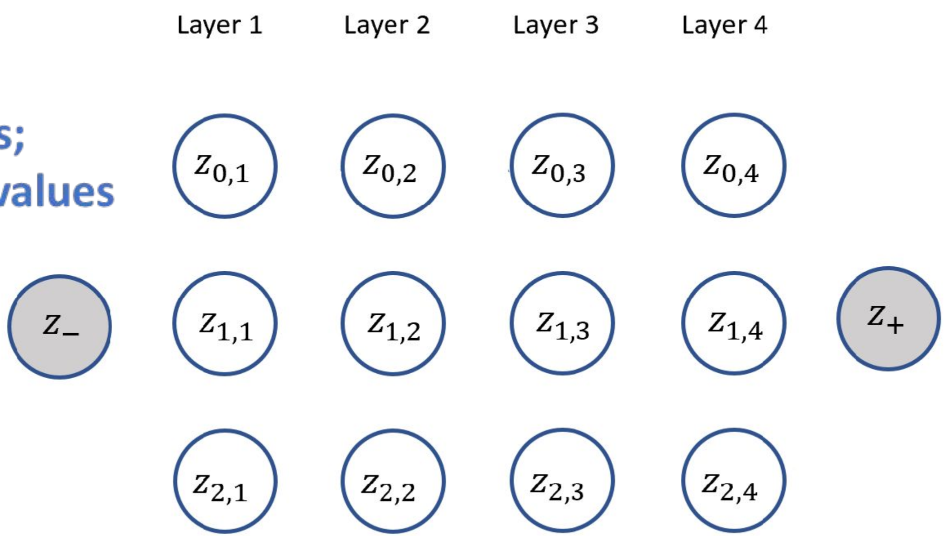
Example: $K = 4$ units;
Set of possible bid values ("candidate" bids) $S_i = \{0, 1, 2\}$.



DAG G_i for the offline problem of player i

- **Vertices:** Create a vertex $z_{s,j}$ for each $s \in S_i$ and $j \in [K]$. We say vertex $z_{s,j}$ is in layer j. Add source z_- and sink z_+.

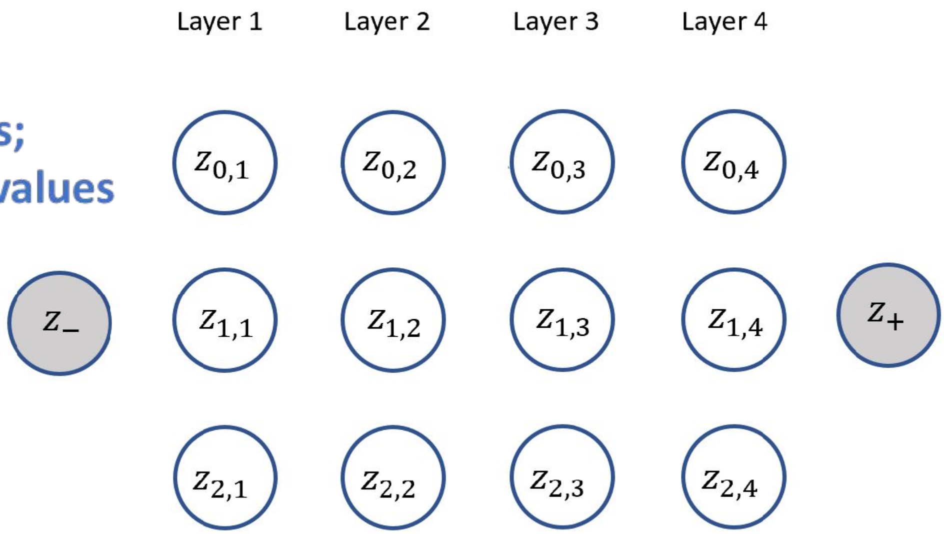
Example: $K = 4$ units; Set of candidate bid values $S_i = \{0, 1, 2\}$.



DAG G_i for the offline problem of player i

- **Vertices:** Create a vertex $z_{s,j}$ for each $s \in S_i$ and $j \in [K]$. We say vertex $z_{s,j}$ is in layer j. Add source z_- and sink z_+.

Example: $K = 4$ units; Set of candidate bid values $S_i = \{0, 1, 2\}$.

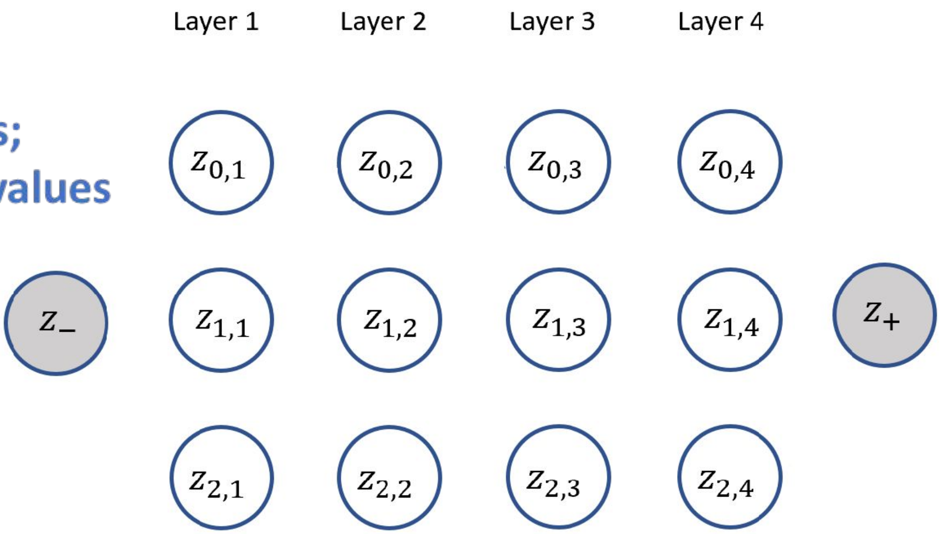


DAG G_i for the offline problem of player i

- **Vertices:** Create a vertex $z_{s,j}$ for each $s \in S_i$ and $j \in [K]$. We say vertex $z_{s,j}$ is in layer j. Add source z_- and sink z_+.

Example: $K = 4$ units; Set of candidate bid values $S_i = \{0, 1, 2\}$.

One vertex for each possible bid value for first unit

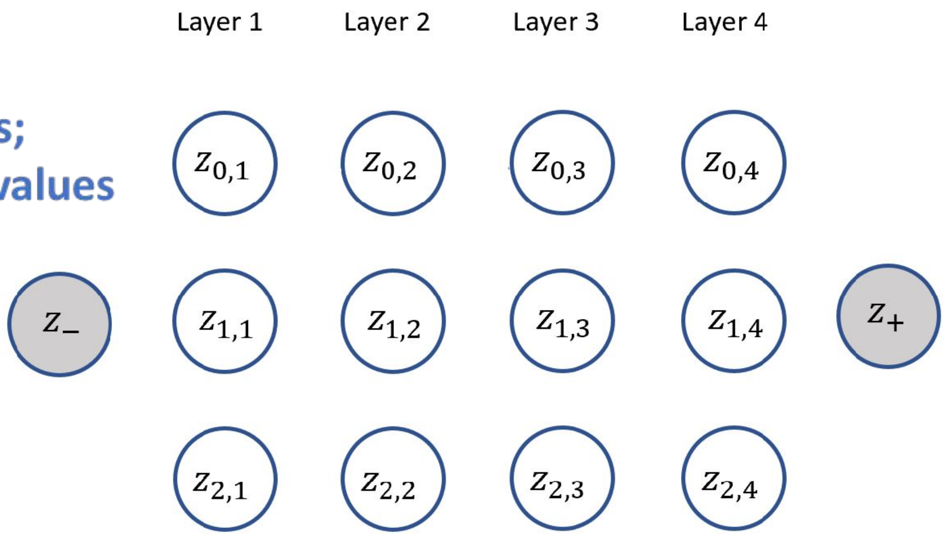


DAG G_i for the offline problem of player i

- **Vertices:** Create a vertex $z_{s,j}$ for each $s \in S_i$ and $j \in [K]$. We say vertex $z_{s,j}$ is in layer j. Add source z_- and sink z_+.

Example: $K = 4$ units; Set of candidate bid values $S_i = \{0, 1, 2\}$.

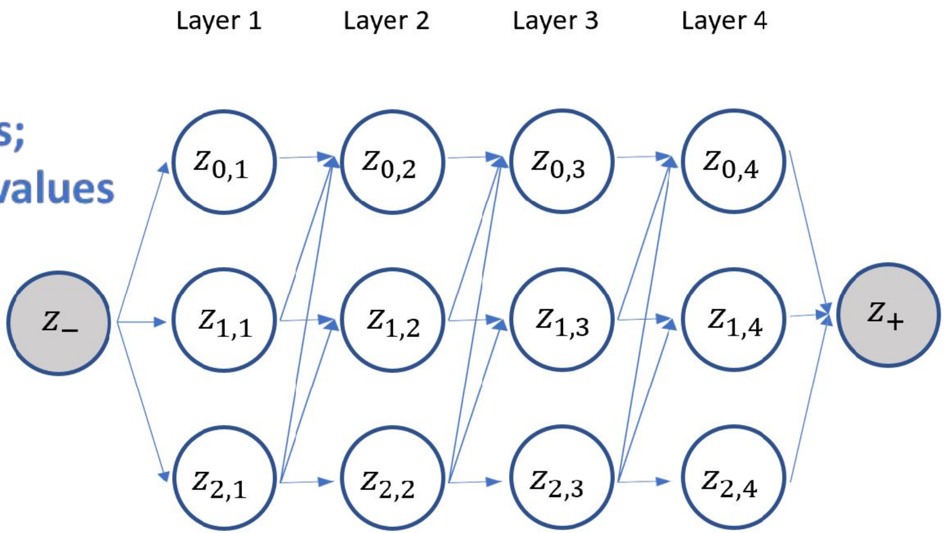
One vertex for each possible bid value for second unit



DAG G_i for the offline problem of player i

- **Edges:** For each $j \in [K-1]$ and pair of bids $r, s \in S_i$ with $r \geq s$, create directed edge from vertex $z_{r,j}$ to $z_{s,j+1}$.

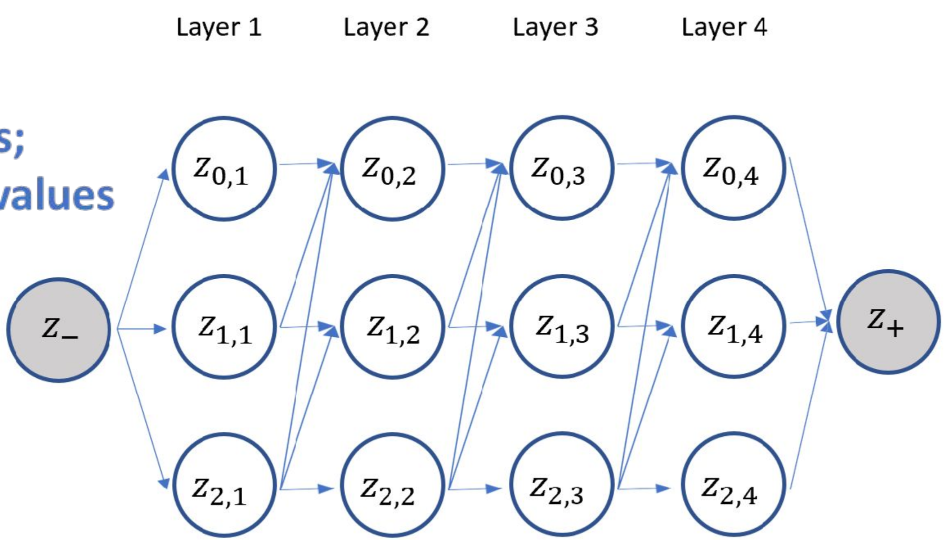
Example: $K = 4$ units;
Set of candidate bid values
$S_i = \{0, 1, 2\}$.



DAG G_i for the offline problem of player i

- **Edges:** For each $j \in [K-1]$ and pair of bids $r, s \in S_i$ with $r \geq s$, create directed edge from vertex $z_{r,j}$ to $z_{s,j+1}$.

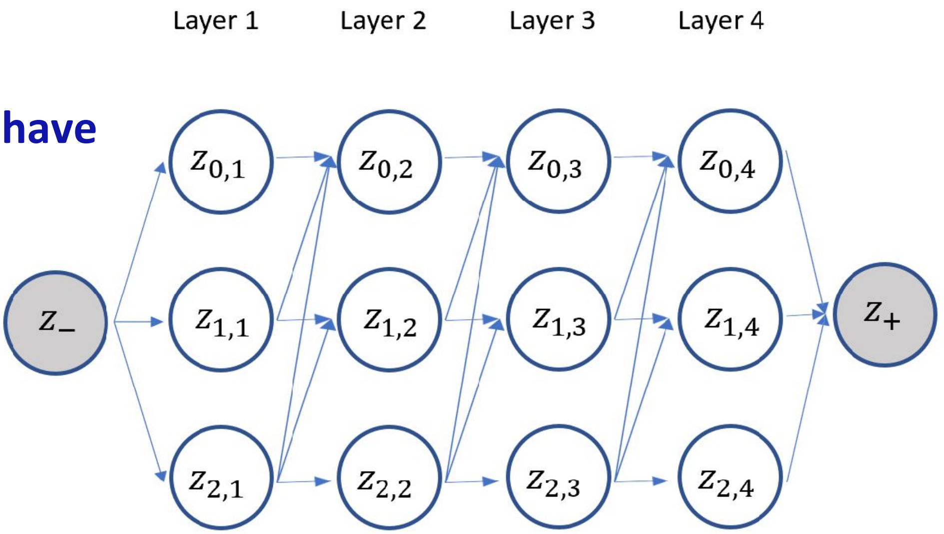
Example: $K = 4$ units; Set of candidate bid values $S_i = \{0, 1, 2\}$.



DAG G_i for the offline problem of player i

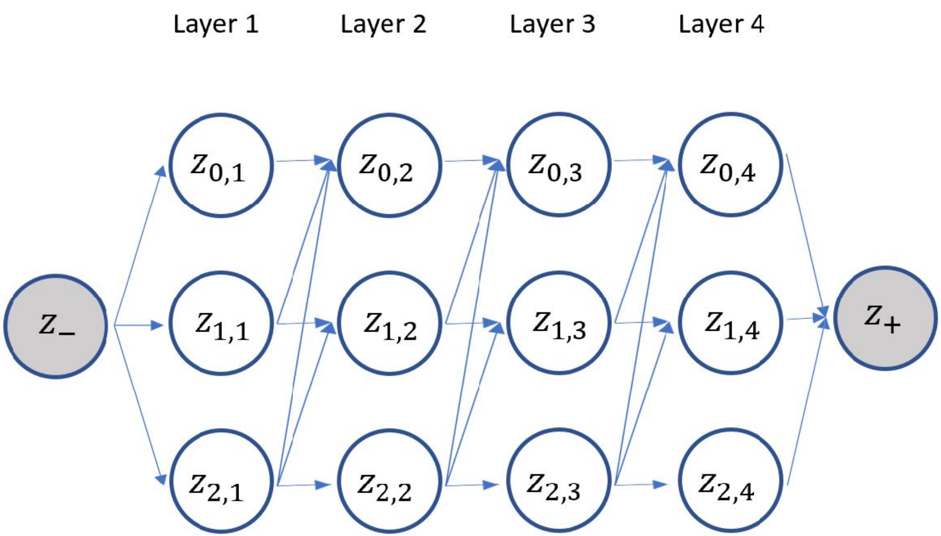
- **Edges:** For each $j \in [K-1]$ and pair of bids $r, s \in S_i$ with $r \geq s$, create directed edge from vertex $z_{r,j}$ to $z_{s,j+1}$. Also add edges from source z_- to each vertex in layer 1 and from each vertex in layer K to the sink z_+.

Note: the nodes do not have equal degrees.



DAG G_i for the offline problem of player i

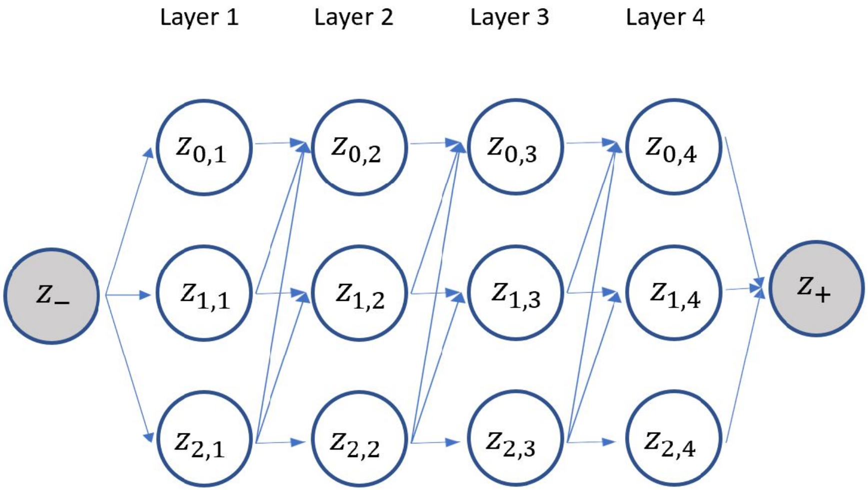
- **Weights:** For each edge $\mathrm{e} = (\mathrm{z}_{\mathrm{r,j}}, \mathrm{z}_{\mathrm{s,j+1}})$ or $\mathrm{e} = \left(\mathrm{z}_{\mathrm{r},K}, \mathrm{z}_{+}\right)$, let $\beta = (\beta_1, \dots, \beta_K)$ be a bid vector with $\beta_j = r$ and $\beta_{j+1} = s$.



DAG G_i for the offline problem of player i

- **Weights:** For each edge $e = (z_{r,j}, z_{s,j+1})$ or $e = (z_{r,K}, z_+)$, let $\beta = (\beta_1, \ldots, \beta_K)$ be a bid vector with $\beta_j = r$ and $\beta_{j+1} = s$.

Define weight of edge e as the j-th term obtained when rewriting the cumulative utility of the player with bid vector β



DAG G_i for the offline problem of player i

- **Weights:** For each edge $\mathrm{e} = (\mathrm{z}_{\mathrm{r,j}}, \mathrm{z}_{\mathrm{s,j+1}})$ or $\mathrm{e} = \left(\mathrm{z}_{\mathrm{r},K}, \mathrm{z}_{+}\right)$, let

$\beta = (\beta_1, \ldots, \beta_K)$ be a bid vector with $\beta_j = r$ and $\beta_{j+1} = s$.

Define weight of edge e as the j-th term obtained when rewriting the cumulative utility of the player with bid vector β

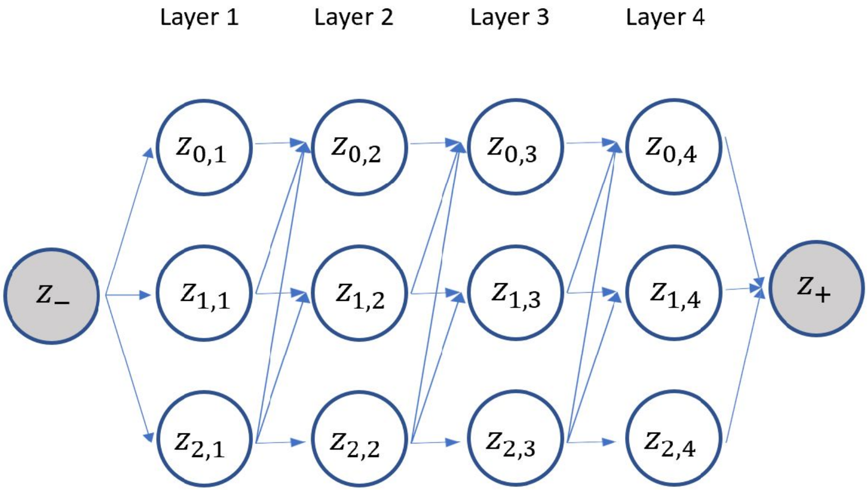
Formally

$$w_e = \sum_{t=1}^{T} 1\{x_i(\beta, b_{-i}^t) \geq j\}(v_{i,j} - r) +$$

[illegible]

DAG G_i for the offline problem of player i

- **Weights:** For each edge $\mathrm{e} = (\mathrm{z}_{\mathrm{r,j}}, \mathrm{z}_{\mathrm{s,j+1}})$ or $\mathrm{e} = (\mathrm{z}_{\mathrm{r},K}, \mathrm{z}_{+})$, let $\beta = (\beta_1, \ldots, \beta_K)$ be a bid vector with $\beta_j = r$ and $\beta_{j+1} = s$.

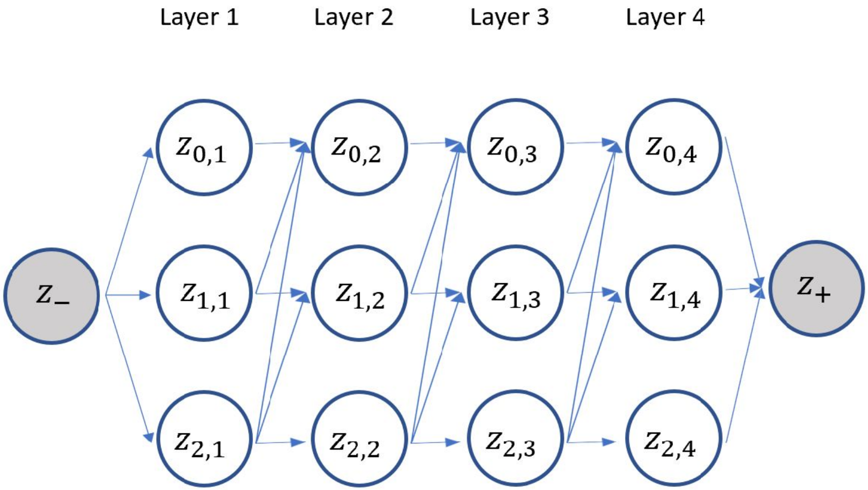


DAG G_i for the offline problem of player i

- **Weights:** For each edge $\mathrm{e} = (\mathrm{z}_{\mathrm{r,j}}, \mathrm{z}_{\mathrm{s,j+1}})$ or $\mathrm{e} = \left(\mathrm{z}_{\mathrm{r},K}, \mathrm{z}_{+}\right)$, let

$\beta = (\beta_1, \ldots, \beta_K)$ be a bid vector with $\beta_j = r$ and $\beta_{j+1} = s$.

Define weight of edge e as the j-th term obtained when rewriting the cumulative utility of the player with bid vector β.

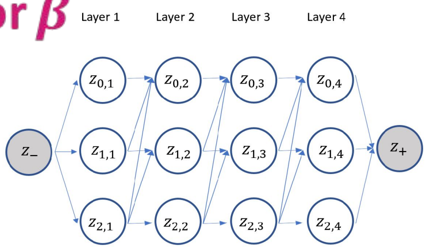


DAG G_i for the offline problem of player i

- **Weights:** For each edge $\mathrm{e} = (\mathrm{z}_{\mathrm{r,j}}, \mathrm{z}_{\mathrm{s,j+1}})$ or $\mathrm{e} = \left(\mathrm{z}_{\mathrm{r},K}, \mathrm{z}_{+}\right)$, let $\beta = (\beta_1, \ldots, \beta_K)$ be a bid vector with $\beta_j = r$ and $\beta_{j+1} = s$.

Define weight of edge e as the j-th term obtained when rewriting the cumulative utility of the player with bid vector β.

The edge weight depends on entire bid vector β



DAG G_i for the offline problem of player i

- **Weights:** For each edge $\mathrm{e} = (\mathrm{z}_{\mathrm{r,j}}, \mathrm{z}_{\mathrm{s,j+1}})$ or $\mathrm{e} = \left(\mathrm{z}_{\mathrm{r},K}, \mathrm{z}_{+}\right)$, let

$\beta = (\beta_1, \dots, \beta_K)$ be a bid vector with $\beta_j = r$ and $\beta_{j+1} = s$.

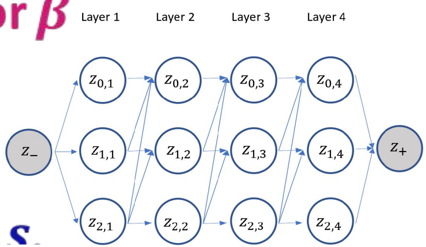
Define weight of edge e as the j-th term obtained when rewriting

the cumulative utility of the player with bid vector β.

The edge weight depends on entire bid vector β

However, the same edge weight is obtained

for any bid vector β with $\beta_j = r$ and $\beta_{j+1} = s$.

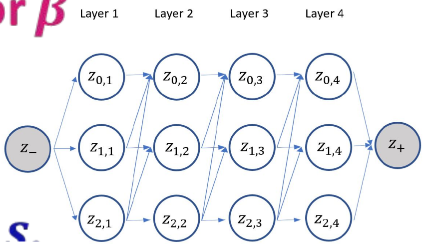


DAG G_i for the offline problem of player i

- **Weights:** For each edge $e = (z_{r,j}, z_{s,j+1})$ or $e = \left(z_{r,K}, z_{+}\right)$, let $\beta = (\beta_1, \ldots, \beta_K)$ be a bid vector with $\beta_j = r$ and $\beta_{j+1} = s$.

  Define weight of edge e as the j-th term obtained when rewriting the cumulative utility of the player with bid vector β.

The edge weight depends on entire bid vector $\boldsymbol{\beta}$



However, the same edge weight is obtained for any bid vector $\boldsymbol{\beta}$ with $\boldsymbol{\beta_j = r}$ and $\boldsymbol{\beta_{j+1} = s}$.

Algorithm

- **Algorithm:** Construct set S_i of candidate bids and the DAG G_i.

Algorithm

- **Algorithm:** Construct set S_i of candidate bids and the DAG G_i.

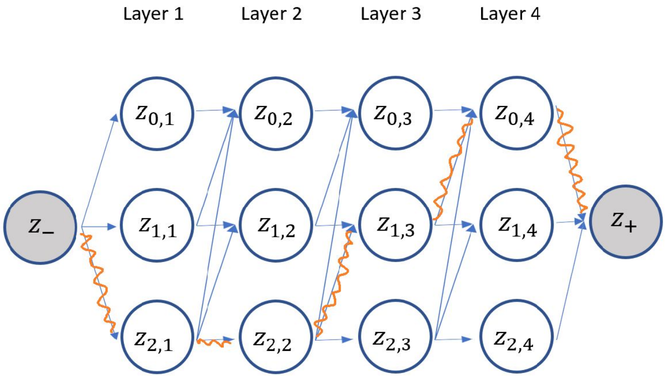
There is a **bijective map between paths from the source to the sink in the DAG and bid vectors $\boldsymbol{\beta} \in S_i$ of player $\boldsymbol{i}$**.

Algorithm

- **Algorithm:** Construct set S_i of candidate bids and the DAG G_i.

There is a **bijective map between paths from the source to the sink in the DAG and bid vectors $\beta \in S_i$ of player i**.

Compute a max weight path in G_i and output the corresponding bid vector.



Outline

- Offline Setting
- Online Setting
- Collusion
- Discussion

Recall the Online Setting

In each round $t = 1, 2, \ldots,$ the next steps take place:

The auctioneer announces K units for sale.

Each player i privately submits bids $b_i^t = \left(b_{i,1}^t, \ldots, b_{i,K}^t\right)$, where $b_{i,j}^t$ is player i's bid for a j-th unit at time t.

Auctioneer runs the auction with bids $b^t = \left(b_1^t, \ldots, b_n^t\right)$ and reveals information (feedback) about the outcome to the players.

Recall the Online Setting

In each round $t = 1, 2, \ldots,$ the next steps take place:

The auctioneer announces K units for sale.

Each player i privately submits bids $b_i^t = \left(b_{i,1}^t, \ldots, b_{i,K}^t\right)$, where $b_{i,j}^t$ is player i's bid for a j-th unit at time t.

Auctioneer runs the auction with bids $b^t = \left(b_1^t, \ldots, b_n^t\right)$ and reveals information (feedback) about the outcome to the players.

Feedback: Full information – All the bids b^t are public knowledge

The Online Setting – Strategies

- A **pure strategy** for player i at time t is a function π_i^t that maps the historical data available to the player to the next action to play (for round t).

The Online Setting – Strategies

- A **pure strategy** for player i at time t is a function π_i^t that maps the historical data available to the player to the next action to play (for round t).

Mixed strategy: probability distribution over pure strategy. The bid vector b_i^t submitted by player i at time t is the realization of the mixed strategy.

The Online Setting – Strategies

- A **pure strategy** for player i at time t is a function π_i^t that maps the historical data available to the player to the next action to play (for round t).

Mixed strategy: probability distribution over pure strategy. The bid vector b_i^t submitted by player i at time t is the realization of the mixed strategy.

$\boldsymbol{\pi_i} = (\boldsymbol{\pi_i^1}, \dots, \boldsymbol{\pi_i^T})$: the overall strategy of player i over the time

The Online Setting – Regret

- Given a bidding strategy $\pi_i = \left(\pi_i^1, \ldots, \pi_i^T\right)$ of player i, the **regret** of the player is defined with respect to a history H_{-i}^t of bids by other players:

The Online Setting – Regret

- Given a bidding strategy $\pi_i = \left(\pi_i^1, \ldots, \pi_i^T\right)$ of player i, the **regret** of the player is defined with respect to a history H_{-i}^t of bids by other players:

$$\mathrm{Reg}_i(\pi_i, H_{-i}^T) = \max_{\boldsymbol{\beta} \in \mathcal{S}_i^K} \sum_{t=1}^{T} \sum_{j=1}^{K} \left(v_{i,j} - p(\boldsymbol{\beta}, \mathbf{b}_{-i}^t)\right) \cdot \mathbb{1}_{\{x_i(\boldsymbol{\beta}, \mathbf{b}_{-i}^t) \geq j\}}$$
$$- \sum_{t=1}^{T} \mathbb{E}_{\mathbf{b}_i^t \sim \pi_i^t(H_i^{t-1})} \left[\sum_{j=1}^{K} \left(v_{i,j} - p(\mathbf{b}^t)\right) \cdot \mathbb{1}_{\{x_i(\mathbf{b}^t) \geq j\}} \right]$$

The Online Setting – Regret

- Given a bidding strategy $\pi_i = (\pi_i^1, \ldots, \pi_i^T)$ of player i, the **regret** of the player is defined with respect to a history H_{-i}^t of bids by other players:

Cumulative utility of player i when using bid vector β in each round while the others have bid profile b_{-i}^t in each round t

$$\mathrm{Reg}_i(\pi_i, H_{-i}^T) = \max_{\boldsymbol{\beta} \in \mathcal{S}_i^K} \sum_{t=1}^{T} \sum_{j=1}^{K} \left(v_{i,j} - p(\boldsymbol{\beta}, \mathbf{b}_{-i}^t)\right) \cdot \mathbb{1}_{\{x_i(\boldsymbol{\beta}, \mathbf{b}_{-i}^t) \geq j\}}$$

$$- \sum_{t=1}^{T} \mathbb{E}_{\mathbf{b}_i^t \sim \pi_i^t(H_i^{t-1})} \left[\sum_{j=1}^{K} \left(v_{i,j} - p(\mathbf{b}^t)\right) \cdot \mathbb{1}_{\{x_i(\mathbf{b}^t) \geq j\}} \right]$$

The Online Setting – Regret

- Given a bidding strategy $\pi_i = (\pi_i^1, \ldots, \pi_i^T)$ of player i, the **regret** of the player is defined with respect to a history H_{-i}^t of bids by other players:

$$\mathrm{Reg}_i(\pi_i, H_{-i}^T) = \max_{\boldsymbol{\beta} \in \mathcal{S}_i^K} \sum_{t=1}^{T} \sum_{j=1}^{K} \left(v_{i,j} - p(\boldsymbol{\beta}, \mathbf{b}_{-i}^t)\right) \cdot \mathbb{1}_{\{x_i(\boldsymbol{\beta}, \mathbf{b}_{-i}^t) \geq j\}}$$
$$- \sum_{t=1}^{T} \mathbb{E}_{\mathbf{b}_i^t \sim \pi_i^t(H_i^{t-1})} \left[\sum_{j=1}^{K} \left(v_{i,j} - p(\mathbf{b}^t)\right) \cdot \mathbb{1}_{\{x_i(\mathbf{b}^t) \geq j\}} \right]$$

Expected cumulative utility of player i when using mixed strategy π_i in each round (which is only allowed to access the history up to end of the previous round when prescribing what to play)**; the other players have bid profile b_{-i}^t in each round t.**

The Online Setting – Regret

- Given a bidding strategy $\pi_i = (\pi_i^1, \dots, \pi_i^T)$ of player i, the **regret** of the player is defined with respect to a history H_{-i}^t of bids by other players:

$$\mathrm{Reg}_i(\pi_i, H_{-i}^T) = \max_{\boldsymbol{\beta} \in \mathcal{S}_i^K} \sum_{t=1}^{T} \sum_{j=1}^{K} \left(v_{i,j} - p(\boldsymbol{\beta}, \mathbf{b}_{-i}^t)\right) \cdot \mathbb{1}_{\{x_i(\boldsymbol{\beta}, \mathbf{b}_{-i}^t) \geq j\}}$$
$$- \sum_{t=1}^{T} \mathbb{E}_{\mathbf{b}_i^t \sim \pi_i^t(H_i^{t-1})} \left[\sum_{j=1}^{K} \left(v_{i,j} - p(\mathbf{b}^t)\right) \cdot \mathbb{1}_{\{x_i(\mathbf{b}^t) \geq j\}} \right]$$

For the purpose of giving player i a bidding algorithm, we think of the other players as adversarial, and aim to achieve small regret **regardless of H_{-i}^t.**

The Online Setting – Full Information Feedback

- The bids b^t are revealed at the end of each round t.

The Online Setting – Full Information Feedback

- The bids b^t are revealed at the end of each round t. Fix a player i.

The Online Setting – Full Information Feedback

- The bids b^t are revealed at the end of each round t. Fix a player i.

Main idea: We construct a DAG G^t, which is the same as the one from the offline setting, **except the edge weights are based on the current round** (rather than the sum over all rounds as it was in the offline setting).

The Online Setting – Full Information Feedback

- The bids b^t are revealed at the end of each round t. Fix a player i.

Main idea: We construct a DAG G^t, which is the same as the one from the offline setting, **except the edge weights are based on the current round** (rather than the sum over all rounds as it was in the offline setting).

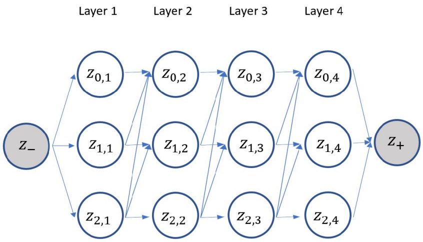
That is, for each edge $e = (z_{r,j}, z_{s,j+1})$ or $e = (z_{r,K}, z_+)$, let $\beta = (\beta_1, \dots, \beta_K)$ be a bid vector with $\beta_j = r$ and $\beta_{j+1} = r$ (where

$$w^t(e) = \mathbb{1}_{\{x_i(\boldsymbol{\beta},\mathbf{b}^t_{-i}) \geq j\}} \left(v_{i,j} - r\right) + j\left[\mathbb{1}_{\{x_i(\boldsymbol{\beta},\mathbf{b}^t_{-i}) > j\}} \left(r - s\right) + \mathbb{1}_{\{x_i(\mathbf{h}^t) = j\}} \left(r - p(\boldsymbol{\beta}, \mathbf{b}^t_{-i})\right)\right]$$

The Online Setting – Full Information Feedback

- The bids b^t are revealed at the end of each round t. Fix a player i.

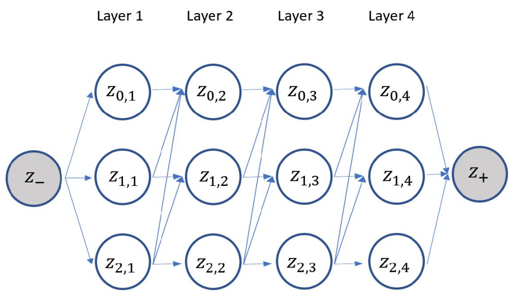
Main idea: We construct a DAG G^t, which is the same as the one from the offline setting, **except the edge weights are based on the current round** (rather than the sum over all rounds as it was in the offline setting).



Create a set of **experts**, each corresponding to a path from source to sink in the DAG G^t and run MWU.

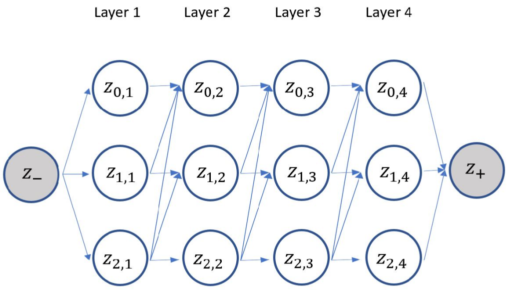
The Online Setting – Full Information Feedback

- **Bidding algorithm.** At each time $t \in [T]$, maintain a probability distribution P^t over all the possible paths from the source to the sink in G^t and then sample a path b_i^t



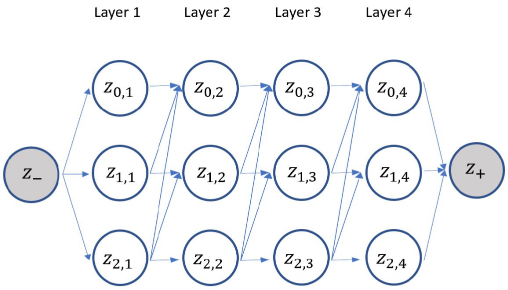
The Online Setting – Full Information Feedback

- **Bidding algorithm.** At each time $t \in [T]$, maintain a probability distribution P^t over all the possible paths from the source to the sink in G^t, and then sample a path b_i^t, where P^t is defined as:



The Online Setting – Full Information Feedback

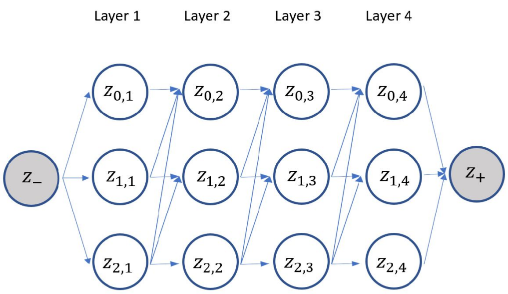
- **Bidding algorithm.** At each time $t \in [T]$, maintain a probability distribution P^t over all the possible paths from the source to the sink in G^t, and then sample a path b_i^t, where P^t is defined as:



- For $t = 1$, consider the random walk where the player i starts from the source z_-, chooses a *uniformly* random neighbor

The Online Setting – Full Information Feedback

- **Bidding algorithm.** At each time $t \in [T]$, maintain a probability distribution P^t over all the possible paths from the source to the sink in G^t, and then sample a path b_i^t, where P^t is defined as:



- For $t = 1$, consider the random walk where the player i starts from the source z_-, chooses a *uniformly* random neighbor

The Online Setting – Full Information Feedback

- **Bidding algorithm.** At each time $t \in [T]$, maintain a probability distribution P^t over all the possible paths from the source to the sink in G^t, and then sample a path b_i^t, where P^t is defined as:
- For $t \geq 2$, recursively define for all paths $\mathfrak{p}$

$$P^t(\mathfrak{p}) = \frac{P^{t-1}(\mathfrak{p}) \exp(\eta \sum_{e \in \mathfrak{p}} w^{t-1}(e))}{\sum_{\mathfrak{q}} P^{t-1}(\mathfrak{q}) \exp(\eta \sum_{e \in \mathfrak{q}} w^{t-1}(e))}$$

The Online Setting – Full Information Feedback

- **Note:** The bidding algorithm described is also known as Hedge:

- N experts, each expert $p \in [N]$ is a path from source to sink
- learning rate η, time horizon T, max reward $L = K\, v_{i,1}$
- initial distribution σ over the experts: $\sigma_p = P^1(p) \geq 1/\left\lceil \frac{v_{i,1}}{\epsilon} \right\rceil^K$

Forecaster starts with initial distribution σ, then at each step predicts according to an expert drawn from distribution, observes the utility (i.e. how good it was to listen to each expert) and updates probability of choosing each expert next time (less likely to listen to experts that gave bad advice so far).

The Online Setting – Full Information Feedback

- The regret of the learner is at most $\frac{1}{\eta} \cdot \max_{p \in [N]} \log\left(\frac{1}{\sigma_p}\right) + \frac{T L^2 \eta}{8}$.

Setting **discretization level** to $\epsilon = v_{i,1}\sqrt{K/T}$ and **learning rate** to

$\eta = \frac{\sqrt{\log T}}{v_{i,1}\sqrt{KT}}$ gives **regret upper bound** of $O\left(v_{i,1} \cdot \sqrt{TK^3 \log(T)}\right)$

for bidder i.

The Online Setting – Full Information Feedback

- The regret of the learner is at most $\frac{1}{\eta} \cdot \max_{p \in [N]} \log\left(\frac{1}{\sigma_p}\right) + \frac{TL^2\eta}{8}$.

Setting **discretization level** to $\epsilon = v_{i,1}\sqrt{K/T}$ and **learning rate** to

$\eta = \frac{\sqrt{\log T}}{v_{i,1}\sqrt{KT}}$ gives **regret upper bound** of $O\left(v_{i,1} \cdot \sqrt{TK^3 \log(T)}\right)$

for bidder i.

The Online Setting – Full Information Feedback

Polynomial time implementation.

For the efficient implementation, we follow the work of Takimoto-Warmuth '03 on path kernels. Simulate MWU using an "indirect" algorithm that maintains only probability per edge rather than per each path.

The Online Setting – Full Information Feedback

- **Polynomial time implementation.** We decompose the probability $P^t(p)$ of choosing path p in the form $P^t(p) = \prod_{e \in p} \phi^t(e)$, for an appropriate choice of ϕ^t. For each vertex u, $\phi^t(u, \cdot)$ is a probability distribution over the outneighbors of u.

The Online Setting – Full Information Feedback

- **Polynomial time implementation.** We decompose the probability $P^t(p)$ of choosing path p in the form $P^t(p) = \prod_{e \in p} \phi^t(e)$, for an appropriate choice of ϕ^t. For each vertex u, $\phi^t(u, \cdot)$ is a probability distribution over the outneighbors of u.

Then a random path $p^t \sim P^t$ could be drawn as follows:

The Online Setting – Full Information Feedback

- **Polynomial time implementation.** We decompose the probability $P^t(p)$ of choosing path p in the form $P^t(p) = \prod_{e \in p} \phi^t(e)$, for an appropriate choice of ϕ^t. For each vertex u, $\phi^t(u, \cdot)$ is a probability distribution over the outneighbors of u.

Then a random path $p^t \sim P^t$ could be drawn as follows:

Starting from the source z the learner first draws a random

The Online Setting – Full Information Feedback

- **Polynomial time implementation.** We decompose the probability $P^t(p)$ of choosing path p in the form $P^t(p) = \prod_{e \in p} \phi^t(e)$, for an appropriate choice of ϕ^t. For each vertex u, $\phi^t(u, \cdot)$ is a probability distribution over the outneighbors of u.

Then a random path $p^t \sim P^t$ could be drawn as follows:

Starting from the source z the learner first draws a random

The Online Setting – Full Information Feedback

- **Polynomial time implementation.** We decompose the probability $P^t(p)$ of choosing path p in the form $P^t(p) = \prod_{e \in p} \phi^t(e)$, for an appropriate choice of ϕ^t. For each vertex u, $\phi^t(u, \cdot)$ is a probability distribution over the outneighbors of u.

Then a random path $p^t \sim P^t$ could be drawn as follows:

Starting from the source z the learner first draws a random

The Online Setting – Full Information Feedback

- **Polynomial time implementation.** We decompose the probability $P^t(p)$ of choosing path p in the form $P^t(p) = \prod_{e \in p} \phi^t(e)$, for an appropriate choice of ϕ^t. For each vertex u, $\phi^t(u, \cdot)$ is a probability distribution over the outneighbors of u.

Then a random path $p^t \sim P^t$ could be drawn as follows:

Starting from the source z the learner first draws a random

The Online Setting – Full Information Feedback

- **Polynomial time implementation.** We decompose the probability $P^t(p)$ of choosing path p in the form $P^t(p) = \prod_{e \in p} \phi^t(e)$, for an appropriate choice of ϕ^t. For each vertex u, $\phi^t(u, \cdot)$ is a probability distribution over the outneighbors of u.

Then a random path $p^t \sim P^t$ could be drawn as follows:

Starting from the source z the learner first draws a random

Conclusion for full information feedback

- **Theorem.** Each player i has a bidding algorithm in the repeated

Conclusion for full information feedback

- **Theorem.** Each player i has a bidding algorithm in the repeated

Dependence on T is sub-linear,
but on K is super-linear

Conclusion for bandit feedback

- **Bandit feedback.** Each player i submits the bids privately in

Conclusion for bandit feedback

- **Bandit feedback.** Each player i submits the bids privately in

Conclusion for bandit feedback

- **Bandit feedback.** Each player i submits the bids privately in

Dependence on T is sub-linear,
but on K is super-linear

Regret lower bound

- **For both full information and bandit feedback.**

Regret lower bound

- **Sketch of construction:** Let $K = 2k$ and $v_{ij} = 1$ for all $j \in [K]$.

Regret lower bound

- **Sketch of construction:** Let $K = 2k$ and $v_{ij} = 1$ for all $j \in [K]$.

Regret lower bound

- **Sketch of construction:** Let $K = 2k$ and $v_{ij} = 1$ for all $j \in [K]$.

***k* zero entries**

Regret lower bound

- **Sketch of construction:** Let $K = 2k$ and $v_{ij} = 1$ for all $j \in [K]$.

***k* zero entries**

Regret lower bound

- **Sketch (cont.):** $K = 2k$ and $v_{ij} = 1$ for all $j \in [K]$.

Regret lower bound

- **Sketch (cont.):** $K = 2k$ and $v_{ij} = 1$ for all $j \in [K]$.

Regret lower bound

- **Sketch (cont.):** $K = 2k$ and $v_{ij} = 1$ for all $j \in [K]$.

Regret lower bound

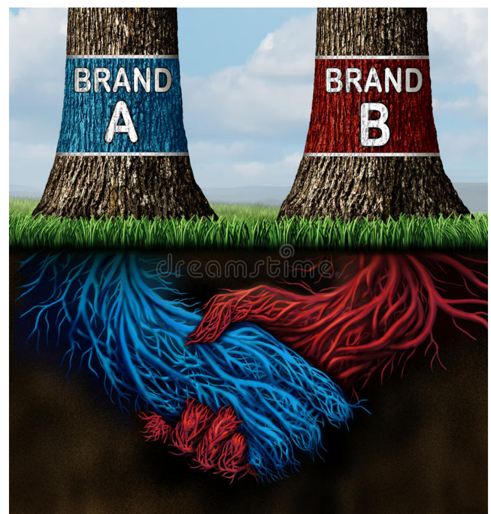
- **Sketch (cont.):** $K = 2k$ and $v_{ij} = 1$ for all $j \in [K]$.

Outline

- Offline Setting
- Online Setting
- **Collusion**
- Discussion

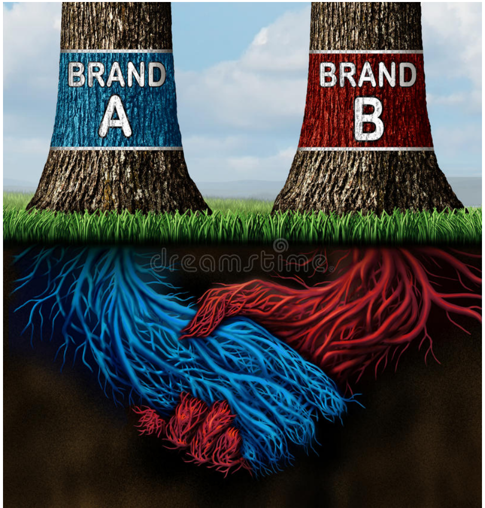
A few words about quality of equilibria

Bidders in an auction can collude to improve their utilities together (e.g. bid low to keep the price small), pay each other for favors. Repeated auctions allow bidders to use bid rotation schemes as the history of bids can serve as a communication device (e.g. Aoyagi ’03).



A few words about quality of equilibria

Collusion can emerge naturally even when the agents are not trying to conspire but use q-learning algorithms to update their strategies (e.g. Calvano-Calzolari-Denicolo-Pastorello ’21, on algorithmic collusion in markets such as Cournot oligopolies).



A few words about quality of equilibria

- Consider the game between the players (bidders).

A few words about quality of equilibria

- **From prior work** - see e.g. Kremer-Nyborg ’04; Ausubel-

A few words about quality of equilibria

- **From prior work** - see e.g. Kremer-Nyborg '04; Ausubel-

H

K bids of value $H \gg max_{i,j} v_{ij}$
(some very large value)
All other bids zero.

K bids

True valuations,
in (0, H)

Remaining
bids

0

A few words about quality of equilibria

- **From prior work** - see e.g. Kremer-Nyborg '04; Ausubel-

K bids

True valuations, in (0, H)

Remaining bids

H

0

K bids of value $H \gg max_{i,j} v_{ij}$
(some very large value)
All other bids zero.
Nobody can deviate and improve:

A few words about quality of equilibria

- **From prior work** - see e.g. Kremer-Nyborg '04; Ausubel-

K **bids**

True valuations, in (0, H)

Remaining bids

H

0

K bids of value $H \gg max_{i,j} v_{ij}$
(some very large value)
All other bids zero.
Nobody can deviate and improve:

- The winners pay zero.

A few words about quality of equilibria

- **From prior work** - see e.g. Kremer-Nyborg '04; Ausubel-

K bids	H	K bids of value $H \gg max_{i,j} v_{ij}$ (some very large value) All other bids zero.
True valuations, in (0, H)		**Nobody can deviate and improve:**
Remaining bids	0	

- The winners pay zero.
- The losers would have to bid $\geq H$

But then they would pay

$$H \gg max_{i,j} v_{ij}$$

A few words about quality of equilibria

- **From prior work** - see e.g. Kremer-Nyborg ’04; Ausubel-

A few words about quality of equilibria

- **From prior work** - see e.g. Kremer-Nyborg '04; Ausubel-

H

In contrast, no such equilibrium exists in the K-th price auction.

E.g. this profile is not a Nash equilibrium in the K-th price auction

0

A few words about quality of equilibria

- **From prior work** - see e.g. Kremer-Nyborg '04; Ausubel-

In contrast, no such equilibrium exists in the K-th price auction. **Why?** If the price is 0, it means the K-th highest bid is 0 and all losing bids are 0.

H

K-1 bids

All other bids

0

A few words about quality of equilibria

- **From prior work** - see e.g. Kremer-Nyborg ’04; Ausubel-

In contrast, no such equilibrium exists in the K-th price auction. **Why?** If the price is 0, it means the K-th highest bid is 0 and all losing bids are 0.

When players are hungry there is a player i with value $v_{i1} \in (0, H)$ who can bid a small $\epsilon > 0$ and get a unit => its utility will be $v_{i1} - \epsilon > 0$.

Core of the game between the bidders

Consider the game between the players (bidders) using the core solution concept.

Core of the game between the bidders

Consider the game between the players (bidders) using the core solution concept.

The core of auctions has been studied extensively, usually allowing the auctioneer to collude together with the players (see, e.g., Ausubel-Milgrom ’02, Krishna-Maenner ’01).

Core of the game between the bidders

Consider the game between the players (bidders) using the core solution concept.

The core of auctions has been studied extensively, usually allowing the auctioneer to collude together with the players (see, e.g., Ausubel-Milgrom ’02, Krishna-Maenner ’01).

We will assume the auctioneer chooses the auction format and then does not take further actions except to run it. The bidders may coordinate their actions.

Core of the game between the bidders

Core of the game between the bidders

- A strategy profile $b = (b_1, \ldots, b_n)$ is (core) stable if no group of

Core of the game between the bidders

- A strategy profile $b = (b_1, \ldots, b_n)$ is (core) stable if no group of

Core of the game between the bidders

- A strategy profile $b = (b_1, \dots, b_n)$ is (core) stable if no group of

Core of the game between the bidders

- **Theorem.** Consider K units and $n > K$ hungry players. The core

Core of the game between the bidders

- **Theorem.** Consider K units and $n > K$ hungry players. The core

Recall hungry players means $v_{ij} > 0 \ \forall \ i, j$

Core of the game between the bidders

- **Theorem.** Consider K units and $n > K$ hungry players. The core

Core of the game between the bidders

- **Theorem.** Consider K units and $n > K$ hungry players. The core

Core of the game between the bidders

- **Theorem.** Consider K units and $n > K$ hungry players. The core

Same construction as for Nash equilibrium of the $(K+1)$-st auction.

$$H \gg \max_{ij} v_{ij}$$

This bid profile is stable (no coalition can deviate and improve).

H

K bids

All other bids

0

Core with transfers

- Consider strategy profiles of the form (b, t), where $b\ =$

Core with transfers

- Consider strategy profiles of the form (b, t), where $b \ =$

Core with transfers

- **Theorem.** Consider K units and $n > K$ hungry players. The core

Core with transfers

- **Theorem.** Consider K units and $n > K$ hungry players. The core

Core with transfers

- **Theorem.** Consider K units and $n > K$ hungry players. The core

Core with transfers

- **Theorem.** Consider K units and $n > K$ hungry players. The core

Core with transfers

- **Theorem.** Consider K units and $n > K$ hungry players. The core

Core with transfers

- **Theorem.** Consider K units and $n > K$ hungry players. The core

Welfare = sum of valuations
It really means the players
with highest valuations win.
In this setting these can be seen
as the players with most money.

Core with transfers

- **Theorem.** Consider K units and $n > K$ hungry players. The core

Core with transfers

- **Theorem.** Consider K units and $n > K$ hungry players. The core

Outline

- Offline Setting
- Online Setting
- Collusion
- **Discussion**

Discussion

- The $(K+1)$-st price auction is used for selling carbon licenses

Discussion

- The $(K+1)$-st price auction is used for selling carbon licenses

Discussion

- The $(K+1)$-st price auction is used for selling carbon licenses

Discussion

- The $(K + 1)$-st price auction is used for selling carbon licenses

Discussion

- The $(K+1)$-st price auction is used for selling carbon licenses

Recall the Online Setting

In each round $t = 1, 2, \ldots,$ the next steps take place:

The auctioneer announces K units for sale.

Each player i privately submits bids $b_i^t = \left(b_{i,1}^t, \ldots, b_{i,K}^t\right)$, where $b_{i,j}^t$ is player i's bid for a j-th unit at time t.

Auctioneer runs the auction with bids $b^t = \left(b_1^t, \ldots, b_n^t\right)$ and reveals information (feedback) about the outcome to the players.

Recall the Online Setting

In each round $t = 1, 2, \ldots,$ the next steps take place:

The auctioneer announces K units for sale.

Each player i privately submits bids $b_i^t = \left(b_{i,1}^t, \ldots, b_{i,K}^t\right)$, where $b_{i,j}^t$ is player i's bid for a j-th unit at time t.

Auctioneer runs the auction with bids $b^t = \left(b_1^t, \ldots, b_n^t\right)$ and reveals information (feedback) about the outcome to the players.

Feedback:

Recall the Online Setting

In each round $t = 1, 2, \ldots,$ the next steps take place:

The auctioneer announces K units for sale.

Each player i privately submits bids $b_i^t = \left(b_{i,1}^t, \ldots, b_{i,K}^t\right)$, where $b_{i,j}^t$ is player i's bid for a j-th unit at time t.

Auctioneer runs the auction with bids $b^t = \left(b_1^t, \ldots, b_n^t\right)$ and reveals information (feedback) about the outcome to the players.

Feedback: Full information – All the bids b^t are public knowledge