Lecture 9
Online Bipartite Matching
Offline bipartite matching
Offline bipartite matching

Given bipartite graph $B = (Y \cup Z, E)$ with $E = \{\{y, z\} : y \in Y, z \in Z\}$. 

Matching $M \subseteq E$ is set of edges where every node is incident to at most one edge from $M$: $\forall v \in Y \cup Z$. \[|\{e \in M : e \cap \{v\}\}| \leq 1\]

Weight of matching $M$ is given by $w(M) = \sum_{e \in M} w_e$.

Goal: Compute maximum weight matching in bipartite graph $B$. 

3 / 26
Offline bipartite matching

Given bipartite graph $B = (Y \cup Z, E)$ with $E = \{\{y, z\} : y \in Y, z \in Z\}$.

- Edge weight function $w : E \to \mathbb{R}_{\geq 0}$. 

Matching $M \subseteq E$ is set of edges where every node is incident to at most one edge from $M$:

$$|\{e \in M : e \cap \{v\}\}| \leq 1 \quad \forall v \in Y \cup Z.$$

Weight of matching $M$ is given by $w(M) = \sum_{e \in M} w_e$. 

Goal: Compute maximum weight matching in bipartite graph $B$. 

Offline bipartite matching

Given bipartite graph $B = (Y \cup Z, E)$ with $E = \{\{y, z\} : y \in Y, z \in Z\}$.
- Edge weight function $w : E \to \mathbb{R}_{\geq 0}$.

Example

![Graph Diagram]

Matching $M \subseteq E$ is set of edges where every node is incident to at most one edge from $M$: $|\{e \in M : e \cap \{v\}\}| \leq 1 \ \forall v \in Y \cup Z$.

Weight of matching $M$ is given by $w(M) = \sum_{e \in M} w(e)$.

Goal: Compute maximum weight matching in bipartite graph $B$. 

3 / 26
Offline bipartite matching

Given bipartite graph $B = (Y \cup Z, E)$ with $E = \{\{y, z\} : y \in Y, z \in Z\}$.
- Edge weight function $w : E \rightarrow \mathbb{R}_{\geq 0}$.

**Example**

Matching $M \subseteq E$ is set of edges where every node is incident to at most one edge from $M$: 
Offline bipartite matching

Given bipartite graph $B = (Y \cup Z, E)$ with $E = \{\{y, z\} : y \in Y, z \in Z\}$.
- Edge weight function $w : E \rightarrow \mathbb{R}_{\geq 0}$.

**Example**

Matching $M \subseteq E$ is set of edges where every node is incident to at most one edge from $M$: $|\{e \in M : e \cap \{v\}\}| \leq 1 \ \forall v \in Y \cup Z$.
Offline bipartite matching

Given bipartite graph $B = (Y \cup Z, E)$ with $E = \{\{y, z\} : y \in Y, z \in Z\}$.
- Edge weight function $w : E \to \mathbb{R}_{\geq 0}$.

**Example**

Matching $M \subseteq E$ is set of edges where every node is incident to at most one edge from $M$: $|\{e \in M : e \cap \{v\}\}| \leq 1 \ \forall v \in Y \cup Z$.
- Weight of matching $M$ is given by

$$w(M) = \sum_{e \in M} w_e.$$
Offline bipartite matching

Given bipartite graph $B = (Y \cup Z, E)$ with $E = \{\{y, z\} : y \in Y, z \in Z\}$.
- Edge weight function $w : E \rightarrow \mathbb{R}_{\geq 0}$.

**Example**

![Graph Image]

- Matching $M \subseteq E$ is set of edges where every node is incident to at most one edge from $M$: $|\{e \in M : e \cap \{v\}\}| \leq 1 \ \forall v \in Y \cup Z$.
- Weight of matching $M$ is given by
  $$w(M) = \sum_{e \in M} w_e.$$  

**Goal:** Compute maximum weight matching in bipartite graph $B$. 
Goal: Compute maximum weight matching in bipartite graph $B$. 
**Goal:** Compute maximum weight matching in bipartite graph $B$.

Many algorithms known for solving this in polynomial time, e.g.: 

- Linear programming.
- Hungarian method.
Goal: Compute maximum weight matching in bipartite graph $B$.

Many algorithms known for solving this in polynomial time, e.g.:

- Linear programming.
Goal: Compute maximum weight matching in bipartite graph $B$.

Many algorithms known for solving this in polynomial time, e.g.:

- Linear programming.
- Hungarian method.
**Goal:** Compute maximum weight matching in bipartite graph $B$.

Many algorithms known for solving this in polynomial time, e.g.:
- Linear programming.
- Hungarian method.

The important thing to remember is the following.
Goal: Compute maximum weight matching in bipartite graph $B$.

Many algorithms known for solving this in polynomial time, e.g.:
- Linear programming.
- Hungarian method.

The important thing to remember is the following.

Theorem (Offline bipartite matching)

There is a poly($n$, $m$)-time algorithm for solving the (offline) maximum weight bipartite matching problem, where $n = |Z|$ and $m = |Y|$. 
Goal: Compute maximum weight matching in bipartite graph $B$.

Many algorithms known for solving this in polynomial time, e.g.:

- Linear programming.
- Hungarian method.

The important thing to remember is the following.

**Theorem (Offline bipartite matching)**

*There is a poly$(n, m)$-time algorithm for solving the (offline) maximum weight bipartite matching problem, where $n = |Z|$ and $m = |Y|$.*

- Parameters $n$ and $m$ are used interchangeably.
**Goal:** Compute maximum weight matching in bipartite graph $B$.

Many algorithms known for solving this in polynomial time, e.g.:

- Linear programming.
- Hungarian method.

The important thing to remember is the following.

**Theorem (Offline bipartite matching)**

*There is a poly($n,m$)-time algorithm for solving the (offline) maximum weight bipartite matching problem, where $n = |Z|$ and $m = |Y|$.*

- Parameters $n$ and $m$ are used interchangeably.
- You may assume that $m = n$ (essentially w.l.o.g.).
Online bipartite matching
Vertex arrival model

We consider the following (semi)-online model:

- Nodes in $Y$ are the offline nodes, which are given.
- Nodes in $Z$ arrive in (unknown) uniform random arrival order $\sigma$.
- When node $z \in Z$ arrives, edge weights $w_{zy}$ for $y \in Y$ are revealed.
- Decide (irrevocably) whether to match up $z$ with some $y \in Y$, or not.

**Goal:** Select maximum weight matching (online).

**Example**

Missing edges have weight $w_{xy} = 0$. Suppose $\sigma = (2, 1, 4, 3)$. 

$Y \quad Z$

$y_1 \quad y_2 \quad y_3 \quad y_4$
Vertex arrival model

We consider the following (semi)-online model:

Nodes in $Y$ are the offline nodes, which are given. Nodes in $Z$ arrive in (unknown) uniform random arrival order $\sigma$. When node $z \in Z$ arrives, edge weights $w_{zy}$ for $y \in Y$ are revealed. Decide (irrevocably) whether to match up $z$ with some $y \in Y$, or not.

Goal: Select maximum weight matching (online).

Example: Missing edges have weight $w_{xy} = 0$. Suppose $\sigma = (2, 1, 4, 3)$.
We consider the following (semi)-online model:

- Nodes in $Y$ are the offline nodes, which are given.

Goal: Select maximum weight matching (online).

Example: Missing edges have weight $w_{xy} = 0$. Suppose $\sigma = (2, 1, 4, 3)$.
Vertex arrival model

We consider the following (semi)-online model:

- Nodes in $Y$ are the offline nodes, which are given.
- Nodes in $Z$ arrive in (unknown) uniform random arrival order $\sigma$.

Example

Suppose $\sigma = (2, 1, 4, 3)$. If $\sigma$ were given beforehand, it could be helpful to know in advance which nodes in $Z$ will arrive next. However, $\sigma$ is not revealed until a node in $Z$ arrives.
Vertex arrival model

We consider the following (semi)-online model:

- Nodes in $Y$ are the offline nodes, which are given.
- Nodes in $Z$ arrive in (unknown) uniform random arrival order $\sigma$.
  - When node $z \in Z$ arrives, edge weights $w_{zy}$ for $y \in Y$ are revealed.
Vertex arrival model

We consider the following (semi)-online model:

- Nodes in $Y$ are the **offline** nodes, which are given.
- Nodes in $Z$ arrive in (unknown) **uniform random arrival order** $\sigma$.
  - When node $z \in Z$ arrives, edge weights $w_{zy}$ for $y \in Y$ are revealed.
  - Decide (irrevocably) whether to match up $z$ with some $y \in Y$, or not.
Vertex arrival model

We consider the following (semi)-online model:

- Nodes in $Y$ are the offline nodes, which are given.
- Nodes in $Z$ arrive in (unknown) uniform random arrival order $\sigma$.
  - When node $z \in Z$ arrives, edge weights $w_{zy}$ for $y \in Y$ are revealed.
  - Decide (irrevocably) whether to match up $z$ with some $y \in Y$, or not.

**Goal:** Select maximum weight matching (online).
We consider the following (semi)-online model:

- Nodes in $Y$ are the offline nodes, which are given.
- Nodes in $Z$ arrive in (unknown) uniform random arrival order $\sigma$.
  - When node $z \in Z$ arrives, edge weights $w_{zy}$ for $y \in Y$ are revealed.
  - Decide (irrevocably) whether to match up $z$ with some $y \in Y$, or not.

**Goal:** Select maximum weight matching (online).

**Example**

Missing edges have weight $w_{xy} = 0$. Suppose $\sigma = (2, 1, 4, 3)$.

```
Y     y1     y2     y3     y4
Z
```
We consider the following (semi)-online model:
- Nodes in $Y$ are the offline nodes, which are given.
- Nodes in $Z$ arrive in (unknown) uniform random arrival order $\sigma$.
  - When node $z \in Z$ arrives, edge weights $w_{zy}$ for $y \in Y$ are revealed.
  - Decide (irrevocably) whether to match up $z$ with some $y \in Y$, or not.

**Goal:** Select maximum weight matching (online).

**Example**

Missing edges have weight $w_{xy} = 0$. Suppose $\sigma = (2, 1, 4, 3)$. 

![Diagram](attachment:vertex_arrival_model_diagram.png)
Vertex arrival model

We consider the following (semi)-online model:

- Nodes in $Y$ are the offline nodes, which are given.
- Nodes in $Z$ arrive in (unknown) uniform random arrival order $\sigma$.
  - When node $z \in Z$ arrives, edge weights $w_{zy}$ for $y \in Y$ are revealed.
  - Decide (irrevocably) whether to match up $z$ with some $y \in Y$, or not.

**Goal:** Select maximum weight matching (online).

**Example**

Missing edges have weight $w_{xy} = 0$. Suppose $\sigma = (2, 1, 4, 3)$.

![Graph Diagram]
Vertex arrival model

We consider the following (semi)-online model:

- Nodes in $Y$ are the offline nodes, which are given.
- Nodes in $Z$ arrive in (unknown) uniform random arrival order $\sigma$.
  - When node $z \in Z$ arrives, edge weights $w_{zy}$ for $y \in Y$ are revealed.
  - Decide (irrevocably) whether to match up $z$ with some $y \in Y$, or not.

**Goal:** Select maximum weight matching (online).

**Example**

Missing edges have weight $w_{xy} = 0$. Suppose $\sigma = (2, 1, 4, 3)$.

![Diagram](image-url)
Vertex arrival model

We consider the following (semi)-online model:

- Nodes in $Y$ are the offline nodes, which are given.
- Nodes in $Z$ arrive in (unknown) uniform random arrival order $\sigma$.
  - When node $z \in Z$ arrives, edge weights $w_{zy}$ for $y \in Y$ are revealed.
  - Decide (irrevocably) whether to match up $z$ with some $y \in Y$, or not.

**Goal:** Select maximum weight matching (online).

**Example**

Missing edges have weight $w_{xy} = 0$. Suppose $\sigma = (2, 1, 4, 3)$. 

\[
\begin{array}{cccc}
\text{Y} & \text{y_1} & \text{y_2} & \text{y_3} & \text{y_4} \\
\text{Z} & \text{z_1} & \text{z_2} & \text{z_3} & \text{z_4} \\
\end{array}
\]

- $w_{y_1z_1} = 4$, $w_{y_2z_1} = 3$, $w_{y_3z_2} = 5$, $w_{y_4z_4} = 1$, $w_{y_1y_2} = 3$.
Vertex arrival model

We consider the following (semi)-online model:
- Nodes in $Y$ are the offline nodes, which are given.
- Nodes in $Z$ arrive in (unknown) uniform random arrival order $\sigma$.
  - When node $z \in Z$ arrives, edge weights $w_{zy}$ for $y \in Y$ are revealed.
  - Decide (irrevocably) whether to match up $z$ with some $y \in Y$, or not.

**Goal:** Select maximum weight matching (online).

**Example**

Missing edges have weight $w_{xy} = 0$. Suppose $\sigma = (2, 1, 4, 3)$. 
Vertex arrival model

We consider the following (semi)-online model:

- Nodes in $Y$ are the offline nodes, which are given.
- Nodes in $Z$ arrive in (unknown) uniform random arrival order $\sigma$.
  - When node $z \in Z$ arrives, edge weights $w_{zy}$ for $y \in Y$ are revealed.
  - Decide (irrevocably) whether to match up $z$ with some $y \in Y$, or not.

**Goal:** Select maximum weight matching (online).

**Example**

Missing edges have weight $w_{xy} = 0$. Suppose $\sigma = (2, 1, 4, 3)$.
Vertex arrival model

We consider the following (semi)-online model:

- Nodes in $Y$ are the offline nodes, which are given.
- Nodes in $Z$ arrive in (unknown) uniform random arrival order $\sigma$.
  - When node $z \in Z$ arrives, edge weights $w_{zy}$ for $y \in Y$ are revealed.
  - Decide (irrevocably) whether to match up $z$ with some $y \in Y$, or not.

**Goal:** Select maximum weight matching (online).

**Example**

Missing edges have weight $w_{xy} = 0$. Suppose $\sigma = (2, 1, 4, 3)$.

Graph showing node connections with edge weights.
Generalization of secretary problem (with uniform random arrivals).

Remark

There exist many other models for online (bipartite) matching:
- Model where all nodes arrive online rather than only one side of the bipartition.
- Model where the edges arrive online instead of the vertices.
Generalization of secretary problem (with uniform random arrivals).

Example

![Diagram](image)

Remark
There exist many other models for online (bipartite) matching:
- Model where all nodes arrive online.
- Rather than only one side of the bipartition.
- Model where the edges arrive online.
  Instead of the vertices.
Generalization of secretary problem (with uniform random arrivals).

Example

Remark
Generalization of secretary problem (with uniform random arrivals).

**Example**

```
Y

Z
Z_1
Z_2
Z_3
Z_4
```

**Remark**

There exist many other models for online (bipartite) matching:
Generalization of secretary problem (with uniform random arrivals).

Example

Remark
There exist many other models for online (bipartite) matching:
- Model where all nodes arrive online.
Generalization of secretary problem (with uniform random arrivals).

Example

```
Y
  / 
Z   y
   /
  /  
Z1  Z2  Z3  Z4
```

Remark

There exist many other models for online (bipartite) matching:
- Model where all nodes arrive online.
- Rather than only one side of the bipartition.
Generalization of secretary problem (with uniform random arrivals).

Example

Remark

There exist many other models for online (bipartite) matching:
- Model where all nodes arrive online.
  - Rather than only one side of the bipartition.
- Model where the edges arrive online.
Generalization of secretary problem (with uniform random arrivals).

Example

![Example Diagram]

Remark

There exist many other models for online (bipartite) matching:

- Model where all nodes arrive online.
  - Rather than only one side of the bipartition.
- Model where the edges arrive online.
  - Instead of the vertices.
Generalization of secretary problem (with uniform random arrivals).

**Example**

![Graph Example](image)

**Remark**

There exist many other models for online (bipartite) matching:
- Model where all nodes arrive online.
  - Rather than only one side of the bipartition.
- Model where the edges arrive online.
  - Instead of the vertices.
Constant-factor approximations

A \( \alpha \)-approximation if \( E_{\sigma} \left[ w(A(\sigma)) \right] \geq \alpha \text{OPT} \)

\text{OPT is weight of an (offline) maximum weight matching.}

\( w(A(\sigma)) \) is (expected) weight of matching selected by \( A \) under \( \sigma \).

Know results:


[Dimitrov-Plaxton, 2008] 1.8-approximation for special case of uniform edge weights.


Will see this algorithm later.

[Reiffenhäuser, 2019].

Strategyproof 1.6-approximation for selling multiple items online.
Deterministic, or randomized, algorithm $\mathcal{A}$ is $\alpha$-approximation if

$$\mathbb{E}_{\sigma}[w(\mathcal{A}(\sigma))] \geq \alpha \text{OPT}$$

$\text{OPT}$ is weight of an (offline) maximum weight matching. $w(\mathcal{A}(\sigma))$ is (expected) weight of matching selected by $\mathcal{A}$ under $\sigma$.

Known results:
- [Babaioff-Immorlica-Kempe-Kleinberg, 2007] $\frac{1}{16}$-approximation for special case of uniform edge weights.
- [Dimitrov-Plaxton, 2008] $\frac{1}{8}$-approximation for special case of uniform edge weights.
- [Korula-Pál, 2009] $\frac{1}{8}$-approximation.
- [Kesselheim-Radke-Tönnis-Vöcking, 2013]. $\left(1 - \frac{1}{e} \right)$-approximation. Best possible!
- [Reiffenhäuser, 2019]. Strategyproof $\frac{1}{e}$-approximation for selling multiple items online.
Deterministic, or randomized, algorithm $\mathcal{A}$ is $\alpha$-approximation if

$$\mathbb{E}_\sigma[w(\mathcal{A}(\sigma))] \geq \alpha \text{OPT}$$

Know results:

- [Babaioff-Immorlica-Kempe-Kleinberg, 2007] $\frac{1}{16}$-approximation for special case of uniform edge weights.
- [Dimitrov-Plaxton, 2008] $\frac{1}{8}$-approximation for special case of uniform edge weights.
- [Korula-Pál, 2009] $\frac{1}{8}$-approximation.
- [Kesselheim-Radke-Tönnis-Vöcking, 2013] $(\frac{1}{e} - \frac{1}{n})$-approximation. Best possible!
- Will see this algorithm later.
- [Reiffenhäuser, 2019].

Strategyproof $\frac{1}{e}$-approximation for selling multiple items online.
Constant-factor approximations

Deterministic, or randomized, algorithm $\mathcal{A}$ is $\alpha$-approximation if

$$E_\sigma[w(\mathcal{A}(\sigma))] \geq \alpha \text{OPT}$$

- OPT is weight of an (offline) maximum weight matching.
Deterministic, or randomized, algorithm $\mathcal{A}$ is $\alpha$-approximation if

$$\mathbb{E}_\sigma[w(\mathcal{A}(\sigma))] \geq \alpha \text{OPT}$$

- OPT is weight of an (offline) maximum weight matching.
- $w(\mathcal{A}(\sigma))$ is (expected) weight of matching selected by $\mathcal{A}$ under $\sigma$. 

Know results:
- [Babaioff-Immorlica-Kempe-Kleinberg, 2007] $\frac{1}{16}$-approximation for special case of uniform edge weights.
- [Dimitrov-Plaxton, 2008] $\frac{1}{8}$-approximation for special case of uniform edge weights.
- [Korula-Pál, 2009] $\frac{1}{8}$-approximation.
- [Kesselheim-Radke-Tönnis-Vöcking, 2013] $(\frac{1}{e} - \frac{1}{n})$-approximation.

Best possible!

Will see this algorithm later.

- [Reiffenhäuser, 2019] Strategyproof $\frac{1}{e}$-approximation for selling multiple items online.
Constant-factor approximations

Deterministic, or randomized, algorithm $\mathcal{A}$ is $\alpha$-approximation if

$$E_\sigma[w(\mathcal{A}(\sigma))] \geq \alpha \text{OPT}$$

- OPT is weight of an (offline) maximum weight matching.
- $w(\mathcal{A}(\sigma))$ is (expected) weight of matching selected by $\mathcal{A}$ under $\sigma$.

Know results:

- [Babaioff-Immorlica-Kempe-Kleinberg, 2007] $\frac{1}{16}$-approximation for special case of uniform edge weights.
- [Dimitrov-Plaxton, 2008] $\frac{1}{8}$-approximation for special case of uniform edge weights.
- [Korula-Pál, 2009] $\frac{1}{8}$-approximation.
- [Kesselheim-Radke-Tönnis-Vöcking, 2013] $(\frac{1}{e} - \frac{1}{n})$-approximation. Best possible!
- [Reiffenhäuser, 2019] Strategyproof $\frac{1}{e}$-approximation for selling multiple items online.
Constant-factor approximations

Deterministic, or randomized, algorithm $\mathcal{A}$ is $\alpha$-approximation if

$$\mathbb{E}_\sigma[w(\mathcal{A}(\sigma))] \geq \alpha \text{OPT}$$

- OPT is weight of an (offline) maximum weight matching.
- $w(\mathcal{A}(\sigma))$ is (expected) weight of matching selected by $\mathcal{A}$ under $\sigma$.

Know results:

- [Babaioff-Immorlica-Kempe-Kleinberg, 2007] $\frac{1}{16}$-approximation for special case of uniform edge weights.
Deterministic, or randomized, algorithm $\mathcal{A}$ is $\alpha$-approximation if

$$\mathbb{E}_\sigma[w(\mathcal{A}(\sigma))] \geq \alpha \text{OPT}$$

- OPT is weight of an (offline) maximum weight matching.
- $w(\mathcal{A}(\sigma))$ is (expected) weight of matching selected by $\mathcal{A}$ under $\sigma$.

**Know results:**

- [Babaioff-Immorlica-Kempe-Kleinberg, 2007]
  - $\frac{1}{16}$-approximation for special case of uniform edge weights.
- [Dimitrov-Plaxton, 2008]
  - $\frac{1}{8}$-approximation for special case of uniform edge weights.
Constant-factor approximations

Deterministic, or randomized, algorithm $\mathcal{A}$ is $\alpha$-approximation if

$$\mathbb{E}_\sigma[w(\mathcal{A}(\sigma))] \geq \alpha \text{OPT}$$

- $\text{OPT}$ is weight of an (offline) maximum weight matching.
- $w(\mathcal{A}(\sigma))$ is (expected) weight of matching selected by $\mathcal{A}$ under $\sigma$.

Know results:
- [Babaioff-Immorlica-Kempe-Kleinberg, 2007] $\frac{1}{16}$-approximation for special case of uniform edge weights.
- [Dimitrov-Plaxton, 2008] $\frac{1}{8}$-approximation for special case of uniform edge weights.
- [Korula-Pál, 2009] $\frac{1}{8}$-approximation
Constant-factor approximations

Deterministic, or randomized, algorithm $\mathcal{A}$ is $\alpha$-approximation if

$$\mathbb{E}_\sigma[w(\mathcal{A}(\sigma))] \geq \alpha \text{OPT}$$

- OPT is weight of an (offline) maximum weight matching.
- $w(\mathcal{A}(\sigma))$ is (expected) weight of matching selected by $\mathcal{A}$ under $\sigma$.

**Know results:**

- [Babaioff-Immorlica-Kempe-Kleinberg, 2007]
  - $\frac{1}{16}$-approximation for special case of uniform edge weights.
- [Dimitrov-Plaxton, 2008]
  - $\frac{1}{8}$-approximation for for special case of uniform edge weights.
- [Korula-Pál, 2009]
  - $\frac{1}{8}$-approximation
- [Kesselheim-Radke-Tönnis-Vöcking, 2013].
  - $(\frac{1}{e} - \frac{1}{n})$-approximation.
Constant-factor approximations

Deterministic, or randomized, algorithm $\mathcal{A}$ is $\alpha$-approximation if

$$\mathbb{E}_\sigma [w(\mathcal{A}(\sigma))] \geq \alpha \text{OPT}$$

- $\text{OPT}$ is weight of an (offline) maximum weight matching.
- $w(\mathcal{A}(\sigma))$ is (expected) weight of matching selected by $\mathcal{A}$ under $\sigma$.

**Know results:**

- [Babaioff-Immorlica-Kempe-Kleinberg, 2007] $\frac{1}{16}$-approximation for special case of uniform edge weights.
- [Dimitrov-Plaxton, 2008] $\frac{1}{8}$-approximation for special case of uniform edge weights.
- [Korula-Pál, 2009] $\frac{1}{8}$-approximation
- [Kesselheim-Radke-Tönnis-Vöcking, 2013].
  - $(\frac{1}{e} - \frac{1}{n})$-approximation.
  - Best possible!
Deterministic, or randomized, algorithm $A$ is $\alpha$-approximation if

$$\mathbb{E}_\sigma[w(A(\sigma))] \geq \alpha \text{OPT}$$

- OPT is weight of an (offline) maximum weight matching.
- $w(A(\sigma))$ is (expected) weight of matching selected by $A$ under $\sigma$.

**Know results:**
- [Babaioff-Immorlica-Kempe-Kleinberg, 2007]
  - $\frac{1}{16}$-approximation for special case of uniform edge weights.
- [Dimitrov-Plaxton, 2008]
  - $\frac{1}{8}$-approximation for special case of uniform edge weights.
- [Korula-Pál, 2009]
  - $\frac{1}{8}$-approximation
- [Kesselheim-Radke-Tönnis-Vöcking, 2013].
  - $(\frac{1}{e} - \frac{1}{n})$-approximation.
  - **Best possible!** Will see this algorithm later.
Deterministic, or randomized, algorithm $\mathcal{A}$ is $\alpha$-approximation if

$$\mathbb{E}_\sigma[w(\mathcal{A}(\sigma))] \geq \alpha \text{OPT}$$

- OPT is weight of an (offline) maximum weight matching.
- $w(\mathcal{A}(\sigma))$ is (expected) weight of matching selected by $\mathcal{A}$ under $\sigma$.

**Know results:**

- [Babaioff-Immorlica-Kempe-Kleinberg, 2007]
  - $\frac{1}{16}$-approximation for special case of uniform edge weights.
- [Dimitrov-Plaxton, 2008]
  - $\frac{1}{8}$-approximation for special case of uniform edge weights.
- [Korula-Pál, 2009]
  - $\frac{1}{8}$-approximation
- [Kesselheim-Radke-Tönnis-Vöcking, 2013].
  - $(\frac{1}{e} - \frac{1}{n})$-approximation.
  - **Best possible!** Will see this algorithm later.
- [Reiffenhäuser, 2019].
  - Strategyproof $\frac{1}{e}$-approximation for selling multiple items online.
Special case of uniform edge weights

Instance has uniform edge weights if for every $z \in \mathbb{Z}$ arriving online, there is a value $v_i > 0$ such that $w_{yz} \in \{0, v_i\}$. If we interpret edges with weight zero as non-existent, then every edge adjacent to $z$ has the same weight.
Special case of uniform edge weights

Instance has **uniform edge weights** if for every \( z \in Z \) arriving online, there is a value \( v_i > 0 \) such that \( w_{yz} \in \{0, v_i\} \).
Special case of uniform edge weights

Instance has **uniform edge weights** if for every $z \in Z$ arriving online, there is a value $v_i > 0$ such that $w_{yz} \in \{0, v_i\}$.

- If we interpret edges with weight zero as non-existent, then every edge adjacent to $z$ has same weight.
Instance has **uniform edge weights** if for every $z \in Z$ arriving online, there is a value $v_i > 0$ such that $w_{yz} \in \{0, v_i\}$.

- If we interpret edges with weight zero as non-existent, then every edge adjacent to $z$ has same weight.
Online bipartite matching

*KRTV-algorithm*
Theorem (Kesselheim-Radke-Tönnis-Vöcking, 2013)

There exists a \( \left( \frac{1}{e} - \frac{1}{m} \right) \)-approximation for the online bipartite matching problem where nodes of one side of the bipartition arrive online in uniform random order.
Theorem (Kesselheim-Radke-Tönnis-Vöcking, 2013)

There exists a \((\frac{1}{e} - \frac{1}{m})\)-approximation for the online bipartite matching problem where nodes of one side of the bipartition arrive online in uniform random order.

- Generalization of (weight-maximization) secretary problem.
### Theorem (Kesselheim-Radke-Tönnis-Vöcking, 2013)

There exists a \( \left( \frac{1}{e} - \frac{1}{m} \right) \)-approximation for the online bipartite matching problem where nodes of one side of the bipartition arrive online in uniform random order.

- Generalization of (weight-maximization) secretary problem.
Theorem (Kesselheim-Radke-Tönnis-Vöcking, 2013)

There exists a \( \left( \frac{1}{e} - \frac{1}{m} \right) \)-approximation for the online bipartite matching problem where nodes of one side of the bipartition arrive online in uniform random order.

- Generalization of (weight-maximization) secretary problem.
  - Corresponding to the case \(|Y| = 1\).
Theorem (Kesselheim-Radke-Tönnis-Vöcking, 2013)

There exists a \((\frac{1}{e} - \frac{1}{m})\)-approximation for the online bipartite matching problem where nodes of one side of the bipartition arrive online in uniform random order.

- Generalization of (weight-maximization) secretary problem.
  - Corresponding to the case \(|Y| = 1\).
- Factor \(\frac{1}{e}\) therefore also best possible.
**Theorem (Kesselheim-Radke-Tönnis-Vöcking, 2013)**

There exists a \((\frac{1}{e} - \frac{1}{m})\)-approximation for the online bipartite matching problem where nodes of one side of the bipartition arrive online in uniform random order.

- Generalization of (weight-maximization) secretary problem.
  - Corresponding to the case \(|Y| = 1\).
- Factor \(\frac{1}{e}\) therefore also best possible.
  - As this is best possible for single secretary problem.
KRTV-algorithm

Theorem (Kesselheim-Radke-Tönnis-Vöcking, 2013)

There exists a \( \left( \frac{1}{e} - \frac{1}{m} \right) \)-approximation for the online bipartite matching problem where nodes of one side of the bipartition arrive online in uniform random order.

- Generalization of (weight-maximization) secretary problem.
  - Corresponding to the case \( |Y| = 1 \).
- Factor \( \frac{1}{e} \) therefore also best possible.
  - As this is best possible for single secretary problem.

Notation:
Theorem (Kesselheim-Radke-Tönnis-Vöcking, 2013)

There exists a \( \left( \frac{1}{e} - \frac{1}{m} \right) \)-approximation for the online bipartite matching problem where nodes of one side of the bipartition arrive online in uniform random order.

- Generalization of (weight-maximization) secretary problem.
  - Corresponding to the case \(|Y| = 1\).
- Factor \( \frac{1}{e} \) therefore also best possible.
  - As this is best possible for single secretary problem.

Notation:

- Assume arrival order is written as \( \sigma = (Z_1, \ldots, Z_m) \).
Theorem (Kesselheim-Radke-Tönnis-Vöcking, 2013)

There exists a \( \left( \frac{1}{e} - \frac{1}{m} \right) \)-approximation for the online bipartite matching problem where nodes of one side of the bipartition arrive online in uniform random order.

- Generalization of (weight-maximization) secretary problem.
  - Corresponding to the case \(|Y| = 1\).
- Factor \( \frac{1}{e} \) therefore also best possible.
  - As this is best possible for single secretary problem.

Notation:
- Assume arrival order is written as \( \sigma = (z_1, \ldots, z_m) \).
- Bipartite graph \( B = (Z \cup Y, E) \) with weights \( w : E \to \mathbb{R}_{\geq 0} \).
Theorem (Kesselheim-Radke-Tönnis-Vöcking, 2013)

There exists a \( \left( \frac{1}{e} - \frac{1}{m} \right) \)-approximation for the online bipartite matching problem where nodes of one side of the bipartition arrive online in uniform random order.

- Generalization of (weight-maximization) secretary problem.
  - Corresponding to the case \( |Y| = 1 \).
- Factor \( \frac{1}{e} \) therefore also best possible.
  - As this is best possible for single secretary problem.

Notation:

- Assume arrival order is written as \( \sigma = (Z_1, \ldots, Z_m) \).
- Bipartite graph \( B = (Z \cup Y, E) \) with weights \( w : E \to \mathbb{R}_{\geq 0} \).
  - Induced subgraph on \( Z' \cup Y' \) is given by bipartite graph \( B' = (Z' \cup Y', E') \) with \( \{y', z'\} \in E' \Leftrightarrow y' \in Y', z' \in Z' \) and \( \{y', z'\} \in E \).
**OPT**($Z', Y'$) := $w(M^*(Z', Y'))$ is weight of max. weight matching $M^*(Z', Y')$ on induced subgraph $B' = (Z' \cup Y', E')$. 
Algorithm constructs an online matching $M$. 

OPT($Z', Y'$) := $w(M^*(Z', Y'))$ is weight of max. weight matching $M^*(Z', Y')$ on induced subgraph $B' = (Z' \cup Y', E')$. 
OPT($Z', Y'$) := $w(M^*(Z', Y'))$ is weight of max. weight matching $M^*(Z', Y')$ on induced subgraph $B' = (Z' \cup Y', E')$.

Algorithm constructs an online matching $M$.

**KRTV-algorithm with arrival order $\sigma = (z_1, \ldots, z_m)$**
Algorithm constructs an online matching $M$. 

**KRTV-algorithm with arrival order $\sigma = (z_1, \ldots, z_m)$**

Set $M = \emptyset$. 

OPT($Z', Y'$) := $w(M^*(Z', Y'))$ is weight of max. weight matching $M^*(Z', Y')$ on induced subgraph $B' = (Z' \cup Y', E')$. 

Phase I (Observation): For $i = 1, \ldots, \lfloor m \rfloor$:
Do not match up $z_i$.

Phase II (Selection): For $i = \lfloor m \rfloor + 1, \ldots, m$:
Compute optimal (offline) matching $M^*($ $\{z_1, \ldots, z_i\} \cup Y)$.
If it holds that $z_i$ is matched up in offline matching $M^*$ to some $y \in Y$ and $y$ is unmatched in online matching $M$, then set $M = M \cup \{z_i, y\}$. 


Algorithm constructs an online matching $M$.

**KRTV-algorithm with arrival order** $\sigma = (z_1, \ldots, z_m)$

Set $M = \emptyset$.

**Phase I (Observation):** For $i = 1, \ldots, \lceil \frac{m}{e} \rceil$:
\[ \text{OPT}(Z', Y') := w(M^*(Z', Y')) \] is weight of max. weight matching 
\[ M^*(Z', Y') \] on induced subgraph \[ B' = (Z' \cup Y', E') \].

Algorithm constructs an online matching \( M \).

**KRTV-algorithm with arrival order** \( \sigma = (z_1, \ldots, z_m) \)

Set \( M = \emptyset \).

**Phase I (Observation):** For \( i = 1, \ldots, \left\lfloor \frac{m}{e} \right\rfloor \):
- Do not match up \( z_i \).
Algorithm constructs an online matching $M$. 

**KRTV-algorithm with arrival order $\sigma = (z_1, \ldots, z_m)$**

Set $M = \emptyset$.

**Phase I (Observation):** For $i = 1, \ldots, \lfloor \frac{m}{e} \rfloor$:
- Do not match up $z_i$.

**Phase II (Selection):** For $i = \lfloor \frac{m}{e} \rfloor + 1, \ldots, m$:

$\text{OPT}(Z', Y') := w(M^*(Z', Y'))$ is weight of max. weight matching $M^*(Z', Y')$ on induced subgraph $B' = (Z' \cup Y', E')$. 

$M^*(Z', Y')$ is weight of max. weight matching $M^*(Z', Y')$ on induced subgraph $B' = (Z' \cup Y', E')$. 

KRTV-algorithm with arrival order $\sigma = (z_1, \ldots, z_m)$
Algorithm constructs an online matching $M$.

**KRTV-algorithm with arrival order $\sigma = (z_1, \ldots, z_m)$**

Set $M = \emptyset$.

**Phase I (Observation):** For $i = 1, \ldots, \lfloor \frac{m}{e} \rfloor$:
- Do not match up $z_i$.

**Phase II (Selection):** For $i = \lfloor \frac{m}{e} \rfloor + 1, \ldots, m$:
- Compute optimal (offline) matching $M^*(\{z_1, \ldots, z_i\} \cup Y)$.

OPT($Z', Y'$) := $w(M^*(Z', Y'))$ is weight of max. weight matching $M^*(Z', Y')$ on induced subgraph $B' = (Z' \cup Y', E')$. 
OPT(Z', Y') := w(M*(Z', Y')) is weight of max. weight matching M*(Z', Y') on induced subgraph B' = (Z' ∪ Y', E').

Algorithm constructs an online matching M.

**KRTV-algorithm with arrival order σ = (z₁, . . . , zₘ)**

Set M = ∅.

**Phase I (Observation):** For i = 1, . . . , \([\frac{m}{e}]\):
- Do not match up zᵢ.

**Phase II (Selection):** For i = \([\frac{m}{e}] + 1, . . . , m\):
- Compute optimal (offline) matching M*(\(\{z₁, . . . , zᵢ\} ∪ Y\)).
- If it holds that
Algorithm constructs an online matching $M$.

KRTV-algorithm with arrival order $\sigma = (z_1, \ldots, z_m)$

Set $M = \emptyset$.

Phase I (Observation): For $i = 1, \ldots, \lfloor \frac{m}{e} \rfloor$:
- Do not match up $z_i$.

Phase II (Selection): For $i = \lfloor \frac{m}{e} \rfloor + 1, \ldots, m$:
- Compute optimal (offline) matching $M^*(\{z_1, \ldots, z_i\} \cup Y)$.
- If it holds that
  - $z_i$ is matched up in offline matching $M^*$ to some $y \in Y$. 

OPT$(Z', Y') := w(M^*(Z', Y'))$ is weight of max. weight matching $M^*(Z', Y')$ on induced subgraph $B' = (Z' \cup Y', E')$. 

OPT($Z', Y'$) := $w(M^*(Z', Y'))$ is weight of max. weight matching $M^*(Z', Y')$ on induced subgraph $B' = (Z' \cup Y', E')$.

Algorithm constructs an online matching $M$.

**KRTV-algorithm with arrival order $\sigma = (z_1, \ldots, z_m)$**

Set $M = \emptyset$.

**Phase I (Observation):** For $i = 1, \ldots, \lfloor \frac{m}{e} \rfloor$:
- Do not match up $z_i$.

**Phase II (Selection):** For $i = \lfloor \frac{m}{e} \rfloor + 1, \ldots, m$:
- Compute optimal (offline) matching $M^*(\{z_1, \ldots, z_i\} \cup Y)$.
- If it holds that
  - $z_i$ is matched up in **offline matching** $M^*$ to some $y \in Y$ and
  - $y$ is unmatched in **online matching** $M$,
\[ \text{OPT}(Z', Y') := w(M^*(Z', Y')) \] is weight of max. weight matching \( M^*(Z', Y') \) on induced subgraph \( B' = (Z' \cup Y', E') \).

Algorithm constructs an online matching \( M \).

**KRTV-algorithm with arrival order \( \sigma = (z_1, \ldots, z_m) \)**

Set \( M = \emptyset \).

**Phase I (Observation):** For \( i = 1, \ldots, \left\lfloor \frac{m}{e} \right\rfloor \):
- Do not match up \( z_i \).

**Phase II (Selection):** For \( i = \left\lfloor \frac{m}{e} \right\rfloor + 1, \ldots, m \):
- Compute optimal (offline) matching \( M^*({\{z_1, \ldots, z_i}\} \cup Y) \).
- If it holds that
  - \( z_i \) is matched up in offline matching \( M^* \) to some \( y \in Y \) and
  - \( y \) is unmatched in online matching \( M \),
  then set \( M = M \cup \{z_i, y\} \).
ALGORITHM 1: KRTV-algorithm for online bipartite matching

Input : Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \rightarrow \mathbb{R}_{\geq 0}$.

Deterministic algorithm $A$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do
  Do nothing
end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do
  Compute optimal matching $M^*_i = M^*(\{z_1, \ldots, z_i\}, Y)$ using $A$
  if $\{z_i, y\} \in M^*_i$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.
  end
end

return $M$

Example (of running Phase II for $i = 1, \ldots, m$)

Matching $M_i^*$

Matching $M$
**Algorithm 2**: KRTV-algorithm for online bipartite matching

**Input**: Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \rightarrow \mathbb{R}_{>0}$. Deterministic algorithm $A$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do
  Do nothing
end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do
  Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $A$
  if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.
  end
end

return $M$

---

**Example (of running Phase II for $i = 1, \ldots, m$)**

Matching $M_i^*$

\[
\begin{array}{cccccc}
Y & y_1 & y_2 & y_3 & y_4 \\
\hline
Z & z_1 & z_2 & z_3 & z_4
\end{array}
\]

Matching $M$

\[
\begin{array}{cccccc}
Y & y_1 & y_2 & y_3 & y_4 \\
\hline
Z & z_1 & z_2 & z_3 & z_4
\end{array}
\]

\[
\begin{array}{ccccccc}
 & 2 & 4 & 3 & 2 & 5 & 5 & 5 \\
\hline
z_1 & y_1 & y_2 & y_3 & y_4 & z_2 & z_3 & z_4
\end{array}
\]
ALGORITHM 3: KRTV-algorithm for online bipartite matching

Input : Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \rightarrow \mathbb{R}_{>0}$.
        Deterministic algorithm $A$ for max. weight bipartite matching.

Set $M = \emptyset$.
for $i = 1, \ldots, \lfloor m/e \rfloor$ do
  Do nothing
end
for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do
  Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $A$
  if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.
  end
end
return $M$

Example (of running Phase II for $i = 1, \ldots, m$)

Matching $M_i^*$

Matching $M$
Algorithm 4: KRTV-algorithm for online bipartite matching

Input: Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \to \mathbb{R}_{>0}$.

Deterministic algorithm $A$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do
| Do nothing
end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do
| Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $A$
| if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ then
| | Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.
end
end

return $M$

Example (of running Phase II for $i = 1, \ldots, m$)
**Algorithm 5**: KRTV-algorithm for online bipartite matching

**Input**: Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \to \mathbb{R}_{>0}$.

Deterministic algorithm $A$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do
  Do nothing
end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do
  Compute optimal matching $M_{i+} = M^*(\{z_1, \ldots, z_i\}, Y)$ using $A$
  if $\{z_i, y\} \in M_{i+}$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$
  end
end
return $M$

**Example (of running Phase II for $i = 1, \ldots, m$)**

Matching $M_{i+}$

Matching $M$
**ALGORITHM 6**: KRTV-algorithm for online bipartite matching

**Input**: Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \to \mathbb{R}_{>0}$.

Deterministic algorithm $A$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do

Do nothing

end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do

Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $A$

if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ then

Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.

end

end

return $M$

---

**Example (of running Phase II for $i = 1, \ldots, m$)**

$$
\begin{align*}
Y & \quad y_1 \quad y_2 \quad y_3 \quad y_4 \\
Z & \quad z_1 \quad z_2 \\
\end{align*}
$$

Matching $M_i^*$

$$
\begin{align*}
Y & \quad y_1 \quad y_2 \quad y_3 \quad y_4 \\
Z & \quad z_1 \quad z_2 \\
\end{align*}
$$

Matching $M$
ALGORITHM 7: KRTV-algorithm for online bipartite matching

Input : Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \to \mathbb{R}_{\geq 0}$.
Deterministic algorithm $A$ for max. weight bipartite matching.

Set $M = \emptyset$.
for $i = 1, \ldots, \lceil m/e \rceil$ do
  Do nothing
end
for $i = \lceil m/e \rceil + 1, \ldots, m$ do
  Compute optimal matching $M_i^* = M^* \left( \{z_1, \ldots, z_i\}, Y \right)$ using $A$
  if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.
  end
end
return $M$

Example (of running Phase II for $i = 1, \ldots, m$)

Matching $M_i^*$

$Y$  \hspace{1cm} $Y$  \hspace{1cm} $Y$  \hspace{1cm} $Y$  \hspace{1cm} $Y$

$Z$  \hspace{1cm} $Z$  \hspace{1cm} $Z$

Matching $M$

$Y$  \hspace{1cm} $Y$  \hspace{1cm} $Y$  \hspace{1cm} $Y$

$Z$  \hspace{1cm} $Z$  \hspace{1cm} $Z$  \hspace{1cm} $Z$
Algorithm 8: KRTV-algorithm for online bipartite matching

**Input:** Bipartite graph \( B = (Z \cup Y, E) \) and weights \( w : E \rightarrow \mathbb{R}_{\geq 0} \).

Deterministic algorithm \( A \) for max. weight bipartite matching.

Set \( M = \emptyset \).

for \( i = 1, \ldots, \lfloor m/e \rfloor \) do
  Do nothing
end

for \( i = \lfloor m/e \rfloor + 1, \ldots, m \) do
  Compute optimal matching \( M_i^* = M^*(\{z_1, \ldots, z_i\}, Y) \) using \( A \)
  if \( \{z_i, y\} \in M_i^* \) for some \( y \in Y \) then
    Set \( M \leftarrow M \cup \{z_i, y\} \) if \( y \) is unmatched in \( M \).
  end
end

return \( M \)

---

**Example (of running Phase II for \( i = 1, \ldots, m \))**

![Diagram](https://via.placeholder.com/150)
ALGORITHM 9: KRTV-algorithm for online bipartite matching

Input: Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \to \mathbb{R}_{>0}$.

Deterministic algorithm $A$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do
  Do nothing
end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do
  Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $A$
  if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.
  end
end

return $M$

Example (of running Phase II for $i = 1, \ldots, m$)
**ALGORITHM 10:** KRTV-algorithm for online bipartite matching

**Input:** Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \rightarrow \mathbb{R}_{\geq 0}$.

Deterministic algorithm $A$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do
  Do nothing
end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do
  Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $A$
  if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.
  end
end

return $M$

---

**Example (of running Phase II for $i = 1, \ldots, m$)**

- Matching $M_i^*$
  - $Z_1$ matched to $y_2$
  - $z_2$ matched to $y_3$
  - $z_3$ matched to $y_4$

- Matching $M$
  - $z_1$ matched to $y_1$
  - $z_2$ matched to $y_2$
  - $z_3$ matched to $y_3$
  - $z_4$ matched to $y_4$
**ALGORITHM 11:** KRTV-algorithm for online bipartite matching

**Input:** Bipartite graph \( B = (Z \cup Y, E) \) and weights \( w : E \rightarrow \mathbb{R}_{>0} \). Determine algorithm \( A \) for max. weight bipartite matching.

Set \( M = \emptyset \).

for \( i = 1, \ldots, \lfloor m/e \rfloor \) do

Do nothing

end

for \( i = \lfloor m/e \rfloor + 1, \ldots, m \) do

Compute optimal matching \( M_i^* = M^*(\{z_1, \ldots, z_i\}, Y) \) using \( A \)

if \( \{z_i, y\} \in M_i^* \) for some \( y \in Y \) then

Set \( M \leftarrow M \cup \{z_i, y\} \) if \( y \) is unmatched in \( M \).

end

end

return \( M \)

---

**Example (of running Phase II for \( i = 1, \ldots, m \))**

- **Matching \( M_i^* \):**
  - \( Z \):
    - \( Z_1 \): 2
    - \( Z_2 \): 4
    - \( Z_3 \):
      - 2
      - \( Z_4 \):
        - 5
        - 5
        - 5
  - \( Y \):
    - \( y_1 \): 3
    - \( y_2 \):
      - 2
      - \( y_3 \):
        - 5
        - 5
        - 5
    - \( y_4 \):
      - 5

- **Matching \( M \):**
  - \( Z \):
    - \( Z_1 \):
      - 2
      - \( Z_2 \):
        - 4
        - \( Z_3 \):
          - 3
        - \( Z_4 \):
          - 2
          - 5
          - 5
          - 5
ALGORITHM 12: KRTV-algorithm for online bipartite matching

Input: Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \rightarrow \mathbb{R}_{>0}$.

Deterministic algorithm $A$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do
  Do nothing
end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do
  Compute optimal matching $M^*_i = \mathcal{M}^*(\{z_1, \ldots, z_i\}, Y)$ using $A$
  if $\{z_i, y\} \in M^*_i$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.
  end
end

return $M$

Example (of running Phase II for $i = 1, \ldots, m$)
ALGORITHM 13: KRTV-algorithm for online bipartite matching

Input: Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \to \mathbb{R}_{>0}$.
Deterministic algorithm $A$ for max. weight bipartite matching.

Set $M = \emptyset$.
for $i = 1, \ldots, \lfloor m/e \rfloor$ do
  Do nothing
end
for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do
  Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $A$
  if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.
end
end
return $M$

Example (of running Phase II for $i = 1, \ldots, m$)
Online bipartite matching

*KRTV-algorithm: Sketch of analysis*
ALGORITHM 14: KRTV-algorithm for online bipartite matching

Input : Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \rightarrow \mathbb{R}_{\geq 0}$. Deterministic algorithm $A$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do
  Do nothing
end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do
  Compute optimal matching $M^*_i = M^*(\{z_1, \ldots, z_i\}, Y)$ using $A$
  if $\{z_i, y\} \in M^*_i$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.
  end
end

return $M$
Analysis (sketch)

**Algorithm 15:** KRTV-algorithm for online bipartite matching

**Input**: Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \to \mathbb{R}_{\geq 0}$.

- Deterministic algorithm $A$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do

Do nothing

end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do

Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $A$

if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ then

Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.

end

end

return $M$

We will bound contribution $A_i$ of (random) node $i$ arriving in step $i \geq \lceil \frac{m}{e} \rceil$: 

**Algorithm 16:** KRTV-algorithm for online bipartite matching

**Input:** Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \rightarrow \mathbb{R}_{\geq 0}$.

Deterministic algorithm $A$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lceil m/e \rceil$ do
  Do nothing
end

for $i = \lceil m/e \rceil + 1, \ldots, m$ do
  Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $A$
  if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.
  end
end

return $M$

We will bound contribution $A_i$ of (random) node $i$ arriving in step $i \geq \lceil m/e \rceil$:

*(Notation $i$ is used for multiple things to keep everything readable.)*
Analysis (sketch)

**Algorithm 17:** KRTV-algorithm for online bipartite matching

**Input:** Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \to \mathbb{R}_{\geq 0}$, Deterministic algorithm $A$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do
  Do nothing
end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do
  Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $A$
  if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.
  end
end

return $M$

We will bound contribution $A_i$ of (random) node $i$ arriving in step $i \geq \lceil \frac{m}{e} \rceil$:

(Notation $i$ is used for multiple things to keep everything readable.)

- For arrival order $\sigma$, we have
**ALGORITHM 18: KRTV-algorithm for online bipartite matching**

**Input**: Bipartite graph \( B = (Z \cup Y, E) \) and weights \( w : E \to \mathbb{R}_{\geq 0} \).

Deterministic algorithm \( A \) for max. weight bipartite matching.

Set \( M = \emptyset \).

for \( i = 1, \ldots, \lfloor m/e \rfloor \) do
  Do nothing
end

for \( i = \lfloor m/e \rfloor + 1, \ldots, m \) do
  Compute optimal matching \( M_i^* = M^*(\{z_1, \ldots, z_i\}, Y) \) using \( A \)
  if \( \{z_i, y\} \in M_i^* \) for some \( y \in Y \) then
    Set \( M \leftarrow M \cup \{z_i, y\} \) if \( y \) is unmatched in \( M \).
  end
end

return \( M \)

We will bound contribution \( A_i \) of (random) node \( i \) arriving in step \( i \geq \lceil \frac{m}{e} \rceil \):

(Notation \( i \) is used for multiple things to keep everything readable.)

- For arrival order \( \sigma \), we have

\[
A_i = \begin{cases} 
    w_{ir} & \text{if } i \text{ gets matched up with } r \text{ under } \sigma, \\
    0 & \text{otherwise.}
\end{cases}
\]
ALGORITHM 19: KRTV-algorithm for online bipartite matching

Input : Bipartite graph \( B = (Z \cup Y, E) \) and weights \( w : E \to \mathbb{R}_{\geq 0} \).
Deterministic algorithm \( A \) for max. weight bipartite matching.

Set \( M = \emptyset \).

for \( i = 1, \ldots, \lfloor m/e \rfloor \) do
    Do nothing
end

for \( i = \lfloor m/e \rfloor + 1, \ldots, m \) do
    Compute optimal matching \( M_i^* = M^*(\{z_1, \ldots, z_i\}, Y) \) using \( A \)
    if \( \{z_i, y\} \in M_i^* \) for some \( y \in Y \) then
        Set \( M \leftarrow M \cup \{z_i, y\} \) if \( y \) is unmatched in \( M \).
    end
end

return \( M \)

We will bound contribution \( A_i \) of (random) node \( i \) arriving in step \( i \geq \lceil m/e \rceil \):
(Notation \( i \) is used for multiple things to keep everything readable.)

- For arrival order \( \sigma \), we have
  \[
  A_i = \begin{cases} 
    w_{ir} & \text{if } i \text{ gets matched up with } r \text{ under } \sigma, \\
    0 & \text{otherwise.}
  \end{cases}
  \]

- Then
  \[
  \mathbb{E}_\sigma[A_i] = \mathbb{E}_\sigma[\text{Weight of edge } e^{(i)} = \{i, r\} \text{ assigned to } i \text{ in } M_i^*] \\
  \times \mathbb{P}_\sigma[\text{Node } i \text{ can be added to the online matching } M].
  \]
Two claims:

\[ E_\sigma [\text{Weight of edge } e^{(i)} = \{i, r\} \text{ assigned to } i \text{ in } M_i^*] \geq \frac{\text{OPT}}{n} \]

\[ P_\sigma [\text{Node } i \text{ can be added to the online matching } M] \geq \frac{\lfloor n/e \rfloor}{i - 1} \]

where OPT is the offline optimum (on the whole instance).
Two claims:

\[ \mathbb{E}_\sigma [\text{Weight of edge } e^{(i)} = \{i, r\} \text{ assigned to } i \text{ in } M_i^*] \geq \frac{\text{OPT}}{n} \]

\[ \mathbb{P}_\sigma [\text{Node } i \text{ can be added to the online matching } M] \geq \frac{\lfloor n/e \rfloor}{i - 1} \]

where OPT is the offline optimum (on the whole instance).

Exercise: Prove these claims.
Two claims:

\[ \mathbb{E}_\sigma [\text{Weight of edge } e^{(i)} = \{i, r\} \text{ assigned to } i \text{ in } M_i^*] \geq \frac{\text{OPT}}{n} \]

\[ \mathbb{P}_\sigma [\text{Node } i \text{ can be added to the online matching } M] \geq \frac{\lfloor n/e \rfloor}{i - 1} \]

where \( \text{OPT} \) is the offline optimum (on the whole instance).

Exercise: Prove these claims.

The \( \left( \frac{1}{e} - \frac{1}{m} \right) \)-approximation then follows, because
Two claims:

\[ \mathbb{E}_\sigma [\text{Weight of edge } e^{(i)} = \{i, r\} \text{ assigned to } i \text{ in } M_i^*] \geq \frac{\text{OPT}}{n} \]

\[ \mathbb{P}_\sigma [\text{Node } i \text{ can be added to the online matching } M] \geq \frac{\lfloor n/e \rfloor}{i - 1} \]

where OPT is the offline optimum (on the whole instance).

Exercise: Prove these claims.

The \((\frac{1}{e} - \frac{1}{m})\)-approximation then follows, because

\[ \mathbb{E}_\sigma [w(M)] \]
Two claims:

\[ E_\sigma [\text{Weight of edge } e^{(i)} = \{i, r\} \text{ assigned to } i \text{ in } M_i^*] \geq \frac{\text{OPT}}{n} \]

\[ P_\sigma [\text{Node } i \text{ can be added to the online matching } M] \geq \frac{\lfloor n/e \rfloor}{i - 1} \]

where OPT is the offline optimum (on the whole instance).

Exercise: Prove these claims.

The \( \left( \frac{1}{e} - \frac{1}{m} \right) \)-approximation then follows, because

\[ E_\sigma [w(M)] = \sum_{i=\lfloor m/e \rfloor+1}^{m} E_\sigma [A_i] \]
Two claims:

\[ \mathbb{E}_{\sigma}[\text{Weight of edge } e^{(i)} = \{i, r\} \text{ assigned to } i \text{ in } M_i^*] \geq \frac{\text{OPT}}{n} \]

\[ \mathbb{P}_{\sigma}[\text{Node } i \text{ can be added to the online matching } M] \geq \frac{\lfloor n/e \rfloor}{i - 1} \]

where OPT is the offline optimum (on the whole instance).

Exercise: Prove these claims.

The \( \left( \frac{1}{e} - \frac{1}{m} \right) \)-approximation then follows, because

\[ \mathbb{E}_{\sigma}[w(M)] = \sum_{i=\lceil m/e \rceil + 1}^{m} \mathbb{E}_{\sigma}[A_i] \geq \sum_{i=\lceil m/e \rceil + 1}^{m} \frac{\text{OPT}}{m} \frac{\lfloor m/e \rfloor}{i - 1} \]
Two claims:

\[ \mathbb{E}_\sigma[\text{Weight of edge } e^{(i)} = \{i, r\} \text{ assigned to } i \text{ in } M^*_i] \geq \frac{OPT}{n} \]

\[ \mathbb{P}_\sigma[\text{Node } i \text{ can be added to the online matching } M] \geq \frac{\lfloor n/e \rfloor}{i - 1} \]

where OPT is the offline optimum (on the whole instance).

Exercise: Prove these claims.

The \((\frac{1}{e} - \frac{1}{m})\)-approximation then follows, because

\[
\mathbb{E}_\sigma[w(M)] = \sum_{i=\lfloor m/e \rfloor + 1}^{m} \mathbb{E}_\sigma[A_i] \geq \sum_{i=\lfloor m/e \rfloor + 1}^{m} \frac{OPT}{m} \frac{\lfloor m/e \rfloor}{i - 1} \\
= \frac{\lfloor m/e \rfloor}{m} \cdot OPT \cdot \sum_{i=\lfloor m/e \rfloor + 1}^{m} \frac{1}{i - 1}
\]
Two claims:
\[
\mathbb{E}_\sigma[\text{Weight of edge } e^{(i)} = \{i, r\} \text{ assigned to } i \text{ in } M_i^*] \geq \frac{\text{OPT}}{n}
\]
\[
\mathbb{P}_\sigma[\text{Node } i \text{ can be added to the online matching } M] \geq \frac{\lfloor n/e \rfloor}{i - 1}
\]
where OPT is the offline optimum (on the whole instance).

Exercise: Prove these claims.

The \((1/e - 1/m)\)-approximation then follows, because
\[
\mathbb{E}_\sigma[w(M)] = \sum_{i=\lceil m/e \rceil + 1}^{m} \mathbb{E}_\sigma[A_i] \geq \sum_{i=\lceil m/e \rceil + 1}^{m} \frac{\text{OPT} \cdot \lfloor m/e \rfloor}{m} \cdot \frac{1}{i - 1}
\]
\[
= \frac{\lfloor m/e \rfloor}{m} \cdot \text{OPT} \cdot \sum_{i=\lceil m/e \rceil + 1}^{m} \frac{1}{i - 1}
\]
\[
\geq \left(1/e - 1/m\right) \cdot \text{OPT} \cdot 1
\]
Offline mechanism design (recap)
Recap offline setting

Unit-demand setting:

- A set of items $M = \{1, \ldots, m\}$
- A set of bidders $N = \{1, \ldots, n\}$
- For every $i \in N$, a private valuation function $v_i: M \to \mathbb{R} \geq 0$.
  - Value $v_{ij} = v_i(j)$ is value of bidder $i$ for item $j$.
- For every $i \in N$, a bid function $b_i: M \to \mathbb{R} \geq 0$.
  - Bid $b_{ij} = b_i(j)$ is maximum amount $i$ is willing to pay for item $j$.

The goal is to assign (at most) one item to every bidder.

Example: Non-existing edges have $b_{ij} = 0$.

<table>
<thead>
<tr>
<th>Items</th>
<th>Bidders</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
Recap offline setting

Unit-demand setting:
- Set of items $M = \{1, \ldots, m\}$
Recap offline setting

Unit-demand setting:
- Set of items $M = \{1, \ldots, m\}$
- Set of bidders $N = \{1, \ldots, n\}$

Value $v_{ij}$ is value of bidder $i$ for item $j$.

Bid $b_{ij}$ is maximum amount $i$ is willing to pay for item $j$.

The goal is to assign (at most) one item to every bidder.

Example
Non-existing edges have $b_{ij} = 0$. 

<table>
<thead>
<tr>
<th>Items</th>
<th>Bidders</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
Recap offline setting

Unit-demand setting:
- Set of items $M = \{1, \ldots, m\}$
- Set of bidders $N = \{1, \ldots, n\}$
- For every $i \in N$ a private valuation function $\nu_i : M \to \mathbb{R}_{\geq 0}$.

Value $\nu_{ij} = \nu_i(j)$ is value of bidder $i$ for item $j$.

Bid $b_{ij} = b_i(j)$ is maximum amount $i$ is willing to pay for item $j$. 

Example: Non-existing edges have $b_{ij} = 0$. 

<table>
<thead>
<tr>
<th>Items</th>
<th>Bidders</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1, 5</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

$\nu_{ij} = 18/26$
Recap offline setting

Unit-demand setting:

- Set of items $M = \{1, \ldots, m\}$
- Set of bidders $N = \{1, \ldots, n\}$
- For every $i \in N$ a private valuation function $v_i : M \to \mathbb{R}_{\geq 0}$.
  - Value $v_{ij} = v_i(j)$ is value of bidder $i$ for item $j$.

Example:

<table>
<thead>
<tr>
<th>Items</th>
<th>Bidders</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

\[ b_{ij} = 1 \]
Recap offline setting

Unit-demand setting:
- Set of items $M = \{1, \ldots, m\}$
- Set of bidders $N = \{1, \ldots, n\}$
- For every $i \in N$ a private valuation function $v_i : M \rightarrow \mathbb{R}_{\geq 0}$.
  - Value $v_{ij} = v_i(j)$ is value of bidder $i$ for item $j$.
- For every $i \in N$ a bid function $b_i : M \rightarrow \mathbb{R}_{\geq 0}$. 
Recap offline setting

Unit-demand setting:

- Set of items $M = \{1, \ldots, m\}$
- Set of bidders $N = \{1, \ldots, n\}$
- For every $i \in N$ a private valuation function $v_i : M \rightarrow \mathbb{R}_{\geq 0}$.
  - Value $v_{ij} = v_i(j)$ is value of bidder $i$ for item $j$.
- For every $i \in N$ a bid function $b_i : M \rightarrow \mathbb{R}_{\geq 0}$.
  - Bid $b_{ij} = b_i(j)$ is maximum amount $i$ is willing to pay for item $j$. 
Recap offline setting

Unit-demand setting:
- Set of items $M = \{1, \ldots, m\}$
- Set of bidders $N = \{1, \ldots, n\}$
- For every $i \in N$ a private valuation function $v_i : M \to \mathbb{R}_{\geq 0}$.
  - Value $v_{ij} = v_i(j)$ is value of bidder $i$ for item $j$.
- For every $i \in N$ a bid function $b_i : M \to \mathbb{R}_{\geq 0}$.
  - Bid $b_{ij} = b_i(j)$ is maximum amount $i$ is willing to pay for item $j$.

The goal is to assign (at most) one item to every bidder.
Recap offline setting

Unit-demand setting:

- Set of items $M = \{1, \ldots, m\}$
- Set of bidders $N = \{1, \ldots, n\}$
- For every $i \in N$ a private valuation function $v_i : M \rightarrow \mathbb{R}_{\geq 0}$.
  - Value $v_{ij} = v_i(j)$ is value of bidder $i$ for item $j$.
- For every $i \in N$ a bid function $b_i : M \rightarrow \mathbb{R}_{\geq 0}$.
  - Bid $b_{ij} = b_i(j)$ is maximum amount $i$ is willing to pay for item $j$.

The goal is to assign (at most) one item to every bidder.

Example

Non-existing edges have $b_{ij} = 0$. 

![Diagram showing items and bidders with bids and valuations]
Definition (Mechanism)

An (offline) mechanism \((x, p)\) is given by an allocation rule

\[ x : \mathbb{R}^{n \times m}_{\geq 0} \to \{0, 1\}^{n \times m}, \]

with \(\sum_i x_{ij} \leq 1\) and \(\sum_j x_{ij} \leq 1\), and pricing rule \(p : \mathbb{R}^{n \times m}_{\geq 0} \to \mathbb{R}_{\geq 0}\).
Definition (Mechanism)

An (offline) mechanism \((x, p)\) is given by an allocation rule
\[
x : \mathbb{R}^{n \times m}_{\geq 0} \rightarrow \{0, 1\}^{n \times m},
\]
with \(\sum_i x_{ij} \leq 1\) and \(\sum_j x_{ij} \leq 1\), and pricing rule \(p : \mathbb{R}^{n \times m}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}\).

- For bidder \(i\), we have bid vector \(b_i = (b_{i1}, \ldots, b_{im})\).
Definition (Mechanism)

An (offline) mechanism \((x, p)\) is given by an allocation rule
\[ x : \mathbb{R}_{\geq 0}^{n \times m} \rightarrow \{0, 1\}^{n \times m}, \]
with \(\sum_i x_{ij} \leq 1\) and \(\sum_j x_{ij} \leq 1\), and pricing rule \(p : \mathbb{R}_{\geq 0}^{n \times m} \rightarrow \mathbb{R}_{\geq 0}\).

- For bidder \(i\), we have bid vector \(b_i = (b_{i1}, \ldots, b_{im})\).
- With \(b = (b_1, \ldots, b_n)\), we have \(x = x(b)\) and \(p = p(b)\).
An (offline) mechanism \((x, p)\) is given by an allocation rule
\[
x : \mathbb{R}^{n \times m}_{\geq 0} \to \{0, 1\}^{n \times m},
\]
with \(\sum_i x_{ij} \leq 1\) and \(\sum_j x_{ij} \leq 1\), and pricing rule \(p : \mathbb{R}^{n \times m}_{\geq 0} \to \mathbb{R}^m_{\geq 0}\).

For bidder \(i\), we have bid vector \(b_i = (b_{i1}, \ldots, b_{im})\).
- With \(b = (b_1, \ldots, b_n)\), we have \(x = x(b)\) and \(p = p(b)\).
- Utility of bidder \(i\) is
\[
    u_i(b) = \begin{cases} 
        v_{ij} - p_j(b) & \text{if } j \text{ is the item } i \text{ receives,} \\
        0 & \text{if } i \text{ does not get an item.}
    \end{cases}
\]
An (offline) mechanism \((x, p)\) is given by an allocation rule
\[
x : \mathbb{R}_{\geq 0}^{n \times m} \to \{0, 1\}^{n \times m},
\]
with \(\sum_i x_{ij} \leq 1\) and \(\sum_j x_{ij} \leq 1\), and pricing rule \(p : \mathbb{R}_{\geq 0}^{n \times m} \to \mathbb{R}_{\geq 0}\).

- For bidder \(i\), we have bid vector \(b_i = (b_{i1}, \ldots, b_{im})\).
  - With \(b = (b_1, \ldots, b_n)\), we have \(x = x(b)\) and \(p = p(b)\).
- Utility of bidder \(i\) is
  \[
u_i(b) = \begin{cases} 
    v_{ij} - p_j(b) & \text{if } j \text{ is the item } i \text{ receives,} \\
    0 & \text{if } i \text{ does not get an item.}
  \end{cases}
\]

Desired properties:
Definition (Mechanism)

An (offline) mechanism \((x, p)\) is given by an allocation rule \(x : \mathbb{R}^{n \times m}_{\geq 0} \rightarrow \{0, 1\}^{n \times m}\), with \(\sum_i x_{ij} \leq 1\) and \(\sum_j x_{ij} \leq 1\), and pricing rule \(p : \mathbb{R}^{n \times m}_{\geq 0} \rightarrow \mathbb{R}^m_{\geq 0}\).

- For bidder \(i\), we have bid vector \(b_i = (b_{i1}, \ldots, b_{im})\).
  - With \(b = (b_1, \ldots, b_n)\), we have \(x = x(b)\) and \(p = p(b)\).
- Utility of bidder \(i\) is
  \[
  u_i(b) = \begin{cases} 
  v_{ij} - p_j(b) & \text{if } j \text{ is the item } i \text{ receives,} \\
  0 & \text{if } i \text{ does not get an item.}
  \end{cases}
  \]

Desired properties:

- \textit{Strategyproof:}
Definition (Mechanism)

An (offline) mechanism \((x, p)\) is given by an allocation rule

\[ x : \mathbb{R}_{\geq 0}^{n \times m} \rightarrow \{0, 1\}^{n \times m}, \]

with \(\sum_i x_{ij} \leq 1\) and \(\sum_j x_{ij} \leq 1\), and pricing rule \(p : \mathbb{R}_{\geq 0}^{n \times m} \rightarrow \mathbb{R}_{\geq 0}\).

- For bidder \(i\), we have **bid vector** \(b_i = (b_{i1}, \ldots, b_{im})\).
  - With \(b = (b_1, \ldots, b_n)\), we have \(x = x(b)\) and \(p = p(b)\).
- **Utility** of bidder \(i\) is
  \[ u_i(b) = \begin{cases} 
  v_{ij} - p_j(b) & \text{if } j \text{ is the item } i \text{ receives,} \\
  0 & \text{if } i \text{ does not get an item.} 
  \end{cases} \]

**Desired properties:**

- **Strategyproof:** For every \(i \in N\), bidding true valuations \(v_i = (v_{i1}, \ldots, v_{im})\) is dominant strategy.
Definition (Mechanism)

An (offline) mechanism \((x, p)\) is given by an allocation rule

\[
x : \mathbb{R}^{n \times m}_{\geq 0} \to \{0, 1\}^{n \times m},
\]

with \(\sum_i x_{ij} \leq 1\) and \(\sum_j x_{ij} \leq 1\), and pricing rule \(p : \mathbb{R}^{n \times m}_{\geq 0} \to \mathbb{R}_{\geq 0}^m\).

- For bidder \(i\), we have bid vector \(b_i = (b_{i1}, \ldots, b_{im})\).
  - With \(b = (b_1, \ldots, b_n)\), we have \(x = x(b)\) and \(p = p(b)\).
- **Utility** of bidder \(i\) is

\[
u_i(b) = \begin{cases} v_{ij} - p_j(b) & \text{if } j \text{ is the item } i \text{ receives}, \\ 0 & \text{if } i \text{ does not get an item}. \end{cases}
\]

**Desired properties:**
- **Strategyproof:** For every \(i \in N\), bidding true valuations \(v_i = (v_{i1}, \ldots, v_{im})\) is dominant strategy.
  - It should hold that

\[
u_i(b_{-i}, v_i) \geq u_i(b_{-i}, b'_i)
\]

for all \(b_{-i} = (b_1, \ldots, b_{i-1}, b_{i+1}, \ldots, b_n)\) and other bid vector \(b'_i\).
Definition (Mechanism)

An (offline) mechanism \((x, p)\) is given by an allocation rule
\[
x : \mathbb{R}^{n \times m}_{\geq 0} \to \{0, 1\}^{n \times m},
\]
with \(\sum_i x_{ij} \leq 1\) and \(\sum_j x_{ij} \leq 1\), and pricing rule \(p : \mathbb{R}^{n \times m}_{\geq 0} \to \mathbb{R}_{\geq 0}^{m}\).

- For bidder \(i\), we have bid vector \(b_i = (b_{i1}, \ldots, b_{im})\).
  - With \(b = (b_1, \ldots, b_n)\), we have \(x = x(b)\) and \(p = p(b)\).
- Utility of bidder \(i\) is
  \[
u_i(b) = \begin{cases} v_{ij} - p_j(b) & \text{if } j \text{ is the item } i \text{ receives}, \\ 0 & \text{if } i \text{ does not get an item}. \end{cases}
\]

Desired properties:
- **Strategyproof**: For every \(i \in N\), bidding true valuations \(v_i = (v_{i1}, \ldots, v_{im})\) is dominant strategy.
  - It should hold that
    \[
u_i(b_{-i}, v_i) \geq u_i(b_{-i}, b'_i)
    \]
    for all \(b_{-i} = (b_1, \ldots, b_{i-1}, b_{i+1}, \ldots, b_n)\) and other bid vector \(b'_i\).
- Also would like to have **individual rationality**, welfare maximization, and computational tractability.
Vickrey-Clarke-Groves (VCG) mechanism

Bipartite graph $B = (X \cup Y, E)$ with edge-weights $w: E \rightarrow \mathbb{R}_{\geq 0}$.

$OPT(X', Y')$ is sum of edge weights of max. weight bipartite matching on induced subgraph $B' = (X' \cup Y', E)$ where $X' \subseteq X, Y' \subseteq Y$.

VCG mechanism

Collect bid vectors $b_1, \ldots, b_n$ from bidders.

Compute maximum weight bipartite matching $L^\ast$ (the allocation $x$)

If bidder $i$ gets item $j$, i.e., $\{i, j\} \in L^\ast(N, M)$, then charge her $p_{ij}(b) = OPT(N \{i\}, M) - OPT(N \{i\}, M \{j\})$, and otherwise nothing.

$OPT(N \{i\}, M) - OPT(N \{i\}, M \{j\})$ is welfare loss for other players by assigning $j$ to $i$. 
Vickrey-Clarke-Groves (VCG) mechanism

Notation:

Bipartite graph $B = (X \cup Y, E)$ with edge-weights $w: E \rightarrow \mathbb{R} \geq 0$.

$OPT(X', Y')$ is sum of edge weights of max. weight bipartite matching on induced subgraph $B' = (X' \cup Y', E)$ where $X' \subseteq X, Y' \subseteq Y$.

VCG mechanism

Collect bid vectors $b_1, \ldots, b_n$ from bidders.

Compute maximum weight bipartite matching $L^*$ (the allocation $x$)

If bidder $i$ gets item $j$, i.e., $\{i, j\} \in L^*(N, M)$, then charge her $p_{ij}(b) = OPT(N \{i\}, M) - OPT(N \{i\}, M \{j\})$, and otherwise nothing.

$OPT(N \{i\}, M) - OPT(N \{i\}, M \{j\})$ is welfare loss for other players by assigning $j$ to $i$. 
Vickrey-Clarke-Groves (VCG) mechanism

Notation:
- Bipartite graph $B = (X \cup Y, E)$ with edge-weights $w : E \to \mathbb{R}_{\geq 0}$. 
Notation:

- Bipartite graph $B = (X \cup Y, E)$ with edge-weights $w : E \to \mathbb{R}_{\geq 0}$.
- $\text{OPT}(X', Y')$ is sum of edge weights of max. weight bipartite matching on induced subgraph $B' = (X' \cup Y', E)$ where $X' \subseteq X$, $Y' \subseteq Y$. 

Vickrey-Clarke-Groves (VCG) mechanism
Vickrey-Clarke-Groves (VCG) mechanism

Notation:
- Bipartite graph $B = (X \cup Y, E)$ with edge-weights $w : E \to \mathbb{R}_{\geq 0}$.
  - $\text{OPT}(X', Y')$ is sum of edge weights of max. weight bipartite matching on induced subgraph $B' = (X' \cup Y', E)$ where $X' \subseteq X$, $Y' \subseteq Y$. 

VCG mechanism

Collect bid vectors $b_1, \ldots, b_n$ from bidders.

Compute maximum weight bipartite matching $L^*$ (the allocation $x$)

If bidder $i$ gets item $j$, i.e., $\{i, j\} \in L^*(N, M)$,
then charge her $p_{ij}(b) = \text{OPT}(N \{i\}, M) - \text{OPT}(N \{i\}, M \{j\})$,
and otherwise nothing.

$\text{OPT}(N \{i\}, M) - \text{OPT}(N \{i\}, M \{j\})$ is welfare loss for other players by assigning $j$ to $i$. 

Vickrey-Clarke-Groves (VCG) mechanism

Notation:
- Bipartite graph $B = (X \cup Y, E)$ with edge-weights $w : E \rightarrow \mathbb{R}_{\geq 0}$.
- $\text{OPT}(X', Y')$ is sum of edge weights of max. weight bipartite matching on induced subgraph $B' = (X' \cup Y', E)$ where $X' \subseteq X$, $Y' \subseteq Y$.

VCG mechanism
- Collect bid vectors $b_1, \ldots, b_n$ from bidders.
Notation:
- Bipartite graph \( B = (X \cup Y, E) \) with edge-weights \( w : E \rightarrow \mathbb{R}_{\geq 0} \).
- \( \text{OPT}(X', Y') \) is sum of edge weights of max. weight bipartite matching on induced subgraph \( B' = (X' \cup Y', E) \) where \( X' \subseteq X, Y' \subseteq Y \).

VCG mechanism
- Collect bid vectors \( b_1, \ldots, b_n \) from bidders.
- Compute maximum weight bipartite matching \( L^* \) (the allocation \( x \))
Vickrey-Clarke-Groves (VCG) mechanism

Notation:
- Bipartite graph $B = (X \cup Y, E)$ with edge-weights $w : E \to \mathbb{R}_{\geq 0}$.
  - $\text{OPT}(X', Y')$ is sum of edge weights of max. weight bipartite matching on induced subgraph $B' = (X' \cup Y', E)$ where $X' \subseteq X$, $Y' \subseteq Y$.

VCG mechanism
- Collect bid vectors $b_1, \ldots, b_n$ from bidders.
- Compute maximum weight bipartite matching $L^*$ (the allocation $x$)
- If bidder $i$ gets item $j$, i.e., $\{i, j\} \in L^*(N, M)$,
Vickrey-Clarke-Groves (VCG) mechanism

Notation:
- Bipartite graph $B = (X \cup Y, E)$ with edge-weights $w : E \rightarrow \mathbb{R}_{\geq 0}$.
- $\text{OPT}(X', Y')$ is sum of edge weights of max. weight bipartite matching on induced subgraph $B' = (X' \cup Y', E)$ where $X' \subseteq X, Y' \subseteq Y$.

VCG mechanism
- Collect bid vectors $b_1, \ldots, b_n$ from bidders.
- Compute maximum weight bipartite matching $L^*$ (the allocation $x$)
- If bidder $i$ gets item $j$, i.e., $\{i, j\} \in L^*(N, M)$, then charge her
  $$p_{ij}(b) = \text{OPT}(N \setminus \{i\}, M) - \text{OPT}(N \setminus \{i\}, M \setminus \{j\}),$$
Vickrey-Clarke-Groves (VCG) mechanism

Notation:
- Bipartite graph $B = (X \cup Y, E)$ with edge-weights $w : E \rightarrow \mathbb{R}_{\geq 0}$.
- $\text{OPT}(X', Y')$ is sum of edge weights of max. weight bipartite matching on induced subgraph $B' = (X' \cup Y', E)$ where $X' \subseteq X, Y' \subseteq Y$.

VCG mechanism
- Collect bid vectors $b_1, \ldots, b_n$ from bidders.
- Compute maximum weight bipartite matching $L^*$ (the allocation $x$)
- If bidder $i$ gets item $j$, i.e., $\{i, j\} \in L^*(N, M)$, then charge her
  $$p_{ij}(b) = \text{OPT}(N \setminus \{i\}, M) - \text{OPT}(N \setminus \{i\}, M \setminus \{j\}),$$
  and otherwise nothing.
Vickrey-Clarke-Groves (VCG) mechanism

Notation:
- Bipartite graph $B = (X \cup Y, E)$ with edge-weights $w : E \rightarrow \mathbb{R}_{\geq 0}$.
- $\text{OPT}(X', Y')$ is sum of edge weights of max. weight bipartite matching on induced subgraph $B' = (X' \cup Y', E)$ where $X' \subseteq X$, $Y' \subseteq Y$.

VCG mechanism
- Collect bid vectors $b_1, \ldots, b_n$ from bidders.
- Compute maximum weight bipartite matching $L^*$ (the allocation $x$)
- If bidder $i$ gets item $j$, i.e., $\{i, j\} \in L^*(N, M)$, then charge her
  \[ p_{ij}(b) = \text{OPT}(N \setminus \{i\}, M) - \text{OPT}(N \setminus \{i\}, M \setminus \{j\}), \]
  and otherwise nothing.

$\text{OPT}(N \setminus \{i\}, M) - \text{OPT}(N \setminus \{i\}, M \setminus \{j\})$ is welfare loss for other players by assigning $j$ to $i$. 
Online bipartite matching

Strategyproof online mechanism
Selling multiple items online

Setting:
Bidder has valuation vector \( v_i \) for items in \( M \).
Whenever bidder arrives online, it submits bid vector \( b_i \).

Bidders arrive one by one in unknown order \( \sigma = (\sigma(1), \ldots, \sigma(n)) \).

Online mechanism (informal)
For \( k = 1, \ldots, n \), upon arrival of bidder \( \sigma(k) \):

- Bid vector \( b_k \) is revealed.
- Decide (irrevocably) whether to assign an item to \( \sigma(k) \).
- If yes, charge price \( p(b_{\sigma(1)}, \ldots, b_{\sigma(k)}) \).

Utility of bidder \( i \), when \( \sigma(k) = i \), is given by:
\[
u_i(k(b_{\sigma(1)}, \ldots, b_{\sigma(k)})) = \begin{cases} v_{ij} - p(b_{\sigma(1)}, \ldots, b_{\sigma(k)}) & \text{if } i \text{ gets item } j, \\ 0 & \text{otherwise}. \end{cases}\]
Selling multiple items online

Setting:

Bidder has valuation vector $v_i$ for items in $M$. Whenever bidder arrives online, it submits bid vector $b_i$. Bidders arrive one by one in unknown order $\sigma = (\sigma(1), ..., \sigma(n))$. Online mechanism (informal)

For $k = 1, ..., n$, upon arrival of bidder $\sigma(k)$:

- Bid vector $b_k$ is revealed.
- Decide (irrevocably) whether to assign an item to $\sigma(k)$.
- If yes, charge price $p(b_{\sigma(1)}, ..., b_{\sigma(k)})$.

Utility of bidder $i$, when $\sigma(k) = i$, is given by:

$$u_i, k(b_{\sigma(1)}, ..., b_{\sigma(k)}) = \begin{cases} v_{ij} - p(b_{\sigma(1)}, ..., b_{\sigma(k)}) & \text{if } i \text{ gets item } j, \\ 0 & \text{otherwise.} \end{cases}$$
Selling multiple items online

Setting:
- Bidder has valuation vector $v_i$ for items in $M$. 
Selling multiple items online

Setting:
- Bidder has valuation vector $v_i$ for items in $M$.
- Whenever bidder arrives online, it submits bid vector $b_i$. 
Selling multiple items online

Setting:
- Bidder has valuation vector $v_i$ for items in $M$.
- Whenever bidder arrives online, it submits bid vector $b_i$.

Bidders arrive one by one in unknown order $\sigma = (\sigma(1), \ldots, \sigma(n))$. 
Selling multiple items online

Setting:
- Bidder has valuation vector $v_i$ for items in $M$.
- Whenever bidder arrives online, it submits bid vector $b_i$.

Bidder arrive one by one in unknown order $\sigma = (\sigma(1), \ldots, \sigma(n))$.

Online mechanism (informal)
Selling multiple items online

Setting:
- Bidder has valuation vector $v_i$ for items in $M$.
- Whenever bidder arrives online, it submits bid vector $b_i$.

Bidders arrive one by one in unknown order $\sigma = (\sigma(1), \ldots, \sigma(n))$.

Online mechanism (informal)

For $k = 1, \ldots, n$, upon arrival of bidder $\sigma(k)$:
Selling multiple items online

Setting:
- Bidder has valuation vector $v_i$ for items in $M$.
- Whenever bidder arrives online, it submits bid vector $b_i$.

*Bidders arrive one by one in unknown order $\sigma = (\sigma(1), \ldots, \sigma(n))$.*

**Online mechanism (informal)**
For $k = 1, \ldots, n$, upon arrival of bidder $\sigma(k)$:
- Bid vector $b_k$ is revealed.
Selling multiple items online

Setting:
- Bidder has valuation vector $v_i$ for items in $M$.
- Whenever bidder arrives online, it submits bid vector $b_i$.

Bidders arrive one by one in unknown order $\sigma = (\sigma(1), \ldots, \sigma(n))$.

Online mechanism (informal)

For $k = 1, \ldots, n$, upon arrival of bidder $\sigma(k)$:
- Bid vector $b_k$ is revealed.
- Decide (irrevocably) whether to assign an item to $\sigma(k)$. 

Utility of bidder $i$, when $\sigma(k) = i$, is given by $u_i(k) = \begin{cases} v_{ij} - p(b_{\sigma(1)}, \ldots, b_{\sigma(k)}) & \text{if } i \text{ gets item } j, \\ 0 & \text{otherwise.} \end{cases}$
Selling multiple items online

Setting:
- Bidder has valuation vector \( v_i \) for items in \( M \).
- Whenever bidder arrives online, it submits bid vector \( b_i \).

*Bidder* arrive **one by one in unknown order** \( \sigma = (\sigma(1), \ldots, \sigma(n)) \).

**Online mechanism (informal)**

For \( k = 1, \ldots, n \), upon arrival of bidder \( \sigma(k) \):
- Bid vector \( b_k \) is revealed.
- Decide (irrevocably) whether to assign an item to \( \sigma(k) \).
  - If yes, charge price \( p(b_{\sigma(1)}, \ldots, b_{\sigma(k)}) \).

**Utility of bidder** \( i \), when \( \sigma(k) = i \), is given by

\[
\begin{align*}
    u_i(b_{\sigma(1)}, \ldots, b_{\sigma(k)}) &= \begin{cases}
    v_{ij} - p(b_{\sigma(1)}, \ldots, b_{\sigma(k)}) & \text{if } i \text{ gets item } j, \\
    0 & \text{otherwise}.
    \end{cases}
\end{align*}
\]
Selling multiple items online

Setting:
- Bidder has valuation vector $v_i$ for items in $M$.
- Whenever bidder arrives online, it submits bid vector $b_i$.

_Bidders arrive one by one in unknown order $\sigma = (\sigma(1), \ldots, \sigma(n))$._

**Online mechanism (informal)**

For $k = 1, \ldots, n$, upon arrival of bidder $\sigma(k)$:
- Bid vector $b_k$ is revealed.
- Decide (irrevocably) whether to assign an item to $\sigma(k)$.
  - If yes, charge price $p(b_{\sigma(1)}, \ldots, b_{\sigma(k)})$.

Utility of bidder $i$, when $\sigma(k) = i$, is given by

$$u_{i,k}(b_{\sigma(1)}, \ldots, b_{\sigma(k)}) = \begin{cases} v_{ij} - p(b_{\sigma(1)}, \ldots, b_{\sigma(k)}) & \text{if } i \text{ gets item } j, \\ 0 & \text{otherwise.} \end{cases}$$
Requirements for (online) deterministic mechanism \((x, p)\):
Requirements for (online) deterministic mechanism \((x, p)\):
Takes as input deterministic ordering \((y_1, \ldots, y_n)\) and bid vectors \(b_1, \ldots, b_n\) for the item.
Requirements for (online) deterministic mechanism $((x, p))$:
Takes as input deterministic ordering $(y_1, \ldots, y_n)$ and bid vectors $b_1, \ldots, b_n$ for the item.
- Specifies for every $k = 1, \ldots, n$ whether to allocate an item to $y_k$. 
Requirements for (online) deterministic mechanism \((x, p)\):
Takes as input deterministic ordering \((y_1, \ldots, y_n)\) and bid vectors \(b_1, \ldots, b_n\) for the item.

- Specifies for every \(k = 1, \ldots, n\) whether to allocate an item to \(y_k\).
- The \(\{0, 1\}\)-variable \(x_{k\ell}\) for whether or not to allocate item \(\ell\) to bidder \(y_k\) (and price \(p_k\), if yes) is function of:
Requirements for (online) deterministic mechanism \((x, p)\):
Takes as input deterministic ordering \((y_1, \ldots, y_n)\) and bid vectors \(b_1, \ldots, b_n\) for the item.
- Specifies for every \(k = 1, \ldots, n\) whether to allocate an item to \(y_k\).
- The \(\{0, 1\}\)-variable \(x_{k\ell}\) for whether or not to allocate item \(\ell\) to bidder \(y_k\) (and price \(p_k\), if yes) is function of:
  - Total number of bidders \(n\).
Requirements for (online) deterministic mechanism \((x, p)\):
Takes as input deterministic ordering \((y_1, \ldots, y_n)\) and bid vectors \(b_1, \ldots, b_n\) for the item.

- Specifies for every \(k = 1, \ldots, n\) whether to allocate an item to \(y_k\).
- The \(\{0, 1\}\)-variable \(x_{k\ell}\) for whether or not to allocate item \(\ell\) to bidder \(y_k\) (and price \(p_k\), if yes) is function of:
  - Total number of bidders \(n\).
  - Bidders \(y_1, \ldots, y_k\).
Requirements for (online) deterministic mechanism $(x, p)$:
Takes as input deterministic ordering $(y_1, \ldots, y_n)$ and bid vectors $b_1, \ldots, b_n$ for the item.

- Specifies for every $k = 1, \ldots, n$ whether to allocate an item to $y_k$.
- The $\{0, 1\}$-variable $x_{k\ell}$ for whether or not to allocate item $\ell$ to bidder $y_k$ (and price $p_k$, if yes) is function of:
  - Total number of bidders $n$.
  - Bidders $y_1, \ldots, y_k$.
  - Bids $b_1, \ldots, b_k$. 

As before, $\sum_k x_{k\ell} \leq 1$ and $\sum_\ell x_{k\ell} \leq 1$.

Mechanism is truthful, if, upon arrival, reporting truthful bids is optimal (assuming bidders have full knowledge about $(x, p)$ and bidders arrived so far), for every possible arrival order $\sigma$. 

Requirements for (online) deterministic mechanism \((x, p)\):
Takes as input deterministic ordering \((y_1, \ldots, y_n)\) and bid vectors \(b_1, \ldots, b_n\) for the item.

- Specifies for every \(k = 1, \ldots, n\) whether to allocate an item to \(y_k\).
- The \(\{0, 1\}\)-variable \(x_{k\ell}\) for whether or not to allocate item \(\ell\) to bidder \(y_k\) (and price \(p_k\), if yes) is function of:
  - Total number of bidders \(n\).
  - Bidders \(y_1, \ldots, y_k\).
  - Bids \(b_1, \ldots, b_k\).
  - The order \((y_1, \ldots, y_k)\).
Requirements for (online) deterministic mechanism \((x, p)\):

Takes as input deterministic ordering \((y_1, \ldots, y_n)\) and bid vectors \(b_1, \ldots, b_n\) for the item.

- Specifies for every \(k = 1, \ldots, n\) whether to allocate an item to \(y_k\).
- The \(\{0, 1\}\)-variable \(x_{k\ell}\) for whether or not to allocate item \(\ell\) to bidder \(y_k\) (and price \(p_k\), if yes) is function of:
  - Total number of bidders \(n\).
  - Bidders \(y_1, \ldots, y_k\).
  - Bids \(b_1, \ldots, b_k\).
  - The order \((y_1, \ldots, y_k)\).

As before, \(\sum_k x_{k\ell} \leq 1\) and \(\sum_{\ell} x_{k\ell} \leq 1\).
Requirements for (online) deterministic mechanism \((x, p)\):
Takes as input deterministic ordering \((y_1, \ldots, y_n)\) and bid vectors \(b_1, \ldots, b_n\) for the item.

- Specifies for every \(k = 1, \ldots, n\) whether to allocate an item to \(y_k\).
- The \(\{0, 1\}\)-variable \(x_{k\ell}\) for whether or not to allocate item \(\ell\) to bidder \(y_k\) (and price \(p_k\), if yes) is function of:
  - Total number of bidders \(n\).
  - Bidders \(y_1, \ldots, y_k\).
  - Bids \(b_1, \ldots, b_k\).
  - The order \((y_1, \ldots, y_k)\).

As before, \(\sum_k x_{k\ell} \leq 1\) and \(\sum_{\ell} x_{k\ell} \leq 1\).

Mechanism is truthful, if, upon arrival, reporting truthful bids is optimal (assuming bidders have full knowledge about \((x, p)\) and bidders arrived so far), for every possible arrival order \(\sigma\).
ALGORITHM 20: KRTV-algorithm for online bipartite matching

**Input**: Bipartite graph \( B = (Z \cup Y, E) \) and weights \( w : E \to \mathbb{R}_{\geq 0} \).

Deterministic algorithm \( A \) for max. weight bipartite matching.

Set \( M = \emptyset \).

for \( i = 1, \ldots, \lfloor m/e \rfloor \) do
  Do nothing
end

for \( i = \lfloor m/e \rfloor + 1, \ldots, m \) do
  Compute optimal matching \( M^*_i = M^*(\{z_1, \ldots, z_i\}, Y) \) using \( A \)
  if \( \{z, y\} \in M^*_i \) for some \( y \in Y \) then
    Set \( M \leftarrow M \cup \{z, y\} \) if \( y \) is unmatched in \( M \).
  end
end

return \( M \)

Example (of running Phase II for \( i = 1, \ldots, m \))

\[
Y \quad y_1 \quad y_2 \quad y_3
\]

Online process

\[
Z
\]
An observation regarding the KRTV-algorithm

ALGORITHM 21: KRTV-algorithm for online bipartite matching

Input: Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \to \mathbb{R}_{\geq 0}$. Deterministic algorithm $\mathcal{A}$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do
  Do nothing
end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do
  Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $\mathcal{A}$
  if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.
  end
end

return $M$

Example (of running Phase II for $i = 1, \ldots, m$)

Online process

$Y$  $y_1$  $y_2$  $y_3$

$Z$
**ALGORITHM 22: KRTV-algorithm for online bipartite matching**

**Input**: Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \to \mathbb{R}_{\geq 0}$.

Deterministic algorithm $\mathcal{A}$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do
  Do nothing
end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do
  Compute optimal matching $M^*_i = M^*(\{z_1, \ldots, z_i\}, Y)$ using $\mathcal{A}$
  if $\{z_i, y\} \in M^*_i$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.
  end
end

return $M$

---

**Example (of running Phase II for $i = 1, \ldots, m$)**

Online process

- $Z$ with $Z_1$
- $Y$ with $y_1$, $y_2$, $y_3$

Connections:
- $Z_1$ to $y_1$ with weight 2
- $Z_1$ to $y_2$ with weight 4
An observation regarding the KRTV-algorithm

ALGORITHM 23: KRTV-algorithm for online bipartite matching

**Input** : Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \to \mathbb{R}_{\geq 0}$.

Deterministic algorithm $\mathcal{A}$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do
  Do nothing
end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do
  Compute optimal matching $M^*_i = M^*(\{z_1, \ldots, z_i\}, Y)$ using $\mathcal{A}$
  if $\{z, y\} \in M^*_i$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.
  end
end

return $M$

---

Example (of running Phase II for $i = 1, \ldots, m$)

Online process

Z

\[ \begin{array}{c}
Z_1 \\
2 \\
\end{array} \]

Y

\[ \begin{array}{c}
y_1 \\
y_2 \\
y_3 \\
\end{array} \]

\[ 2 \quad 4 \]
An observation regarding the KRTV-algorithm

**ALGORITHM 24: KRTV-algorithm for online bipartite matching**

**Input**: Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \to \mathbb{R}_{\geq 0}$.

Deterministic algorithm $A$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do

| Do nothing

end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do

Compute optimal matching $M^*_i = M^*(\{z_1, \ldots, z_i\}, Y)$ using $A$

if $\{z_i, y\} \in M^*_i$ for some $y \in Y$ then

| Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.

end

end

return $M$

---

**Example (of running Phase II for $i = 1, \ldots, m$)**

Online process

- $Y$
  - $y_1$
  - $y_2$
  - $y_3$
- $Z$
  - $z_1$

```
Y  y1       y2       y3
  |         |         |
  |         |         |
  2       4

Z  z1
  |
  2
```
An observation regarding the KRTV-algorithm

**ALGORITHM 25: KRTV-algorithm for online bipartite matching**

**Input**: Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \to \mathbb{R}_{\geq 0}$.

- Deterministic algorithm $\mathcal{A}$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do
  Do nothing
end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do
  Compute optimal matching $M^*_i = \mathcal{A}(\{z_1, \ldots, z_i\}, Y)$ using $\mathcal{A}$
  if $\{z_i, y\} \in M^*_i$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z_i, y\}$ if $y$ is unmatched in $M$.
  end
end

return $M$

---

**Example (of running Phase II for $i = 1, \ldots, m$)**

Online process

- $Z \rightarrow Y$
  - $Z_1 \rightarrow y_1$
  - $Z_2 \rightarrow y_2$
  - $Z_2 \rightarrow y_3$

Weights:
- $e_{Z_1Y_1} = 2$
- $e_{Z_2Y_2} = 4$
- $e_{Z_2Y_3} = 3$
- $e_{Z_1Y_3} = 1$
An observation regarding the KRTV-algorithm

**Algorithm 26:** KRTV-algorithm for online bipartite matching

**Input:** Bipartite graph \( B = (Z \cup Y, E) \) and weights \( w : E \rightarrow \mathbb{R}_{\geq 0} \).

Deterministic algorithm \( A \) for max. weight bipartite matching.

Set \( M = \emptyset \).

for \( i = 1, \ldots, \lfloor m/e \rfloor \) do

Do nothing

end

for \( i = \lfloor m/e \rfloor + 1, \ldots, m \) do

Compute optimal matching \( M^*_i = M^*(\{z_1, \ldots, z_i\}, Y) \) using \( A \)

if \( \{z_i, y\} \in M^*_i \) for some \( y \in Y \) then

Set \( M \leftarrow M \cup \{z_i, y\} \) if \( y \) is unmatched in \( M \).

end

end

return \( M \)

---

**Example (of running Phase II for \( i = 1, \ldots, m \))**

![Example graph](#)
An observation regarding the KRTV-algorithm

**Algorithm 27:** KRTV-algorithm for online bipartite matching

**Input:** Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \rightarrow \mathbb{R}_{\geq 0}$. Deterministic algorithm $\mathcal{A}$ for max. weight bipartite matching.

Set $M = \emptyset$.

for $i = 1, \ldots, \lfloor m/e \rfloor$ do
  Do nothing
end

for $i = \lfloor m/e \rfloor + 1, \ldots, m$ do
  Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $\mathcal{A}$
  if $\{z, y\} \in M_i^*$ for some $y \in Y$ then
    Set $M \leftarrow M \cup \{z, y\}$ if $y$ is unmatched in $M$.
  end
end

return $M$

---

**Example (of running Phase II for $i = 1, \ldots, m$)**

Bidder might have incentive to misreport true valuations, as, in the offline matching $M_i^*$ she is matched up with item already assigned to an earlier bidder.
Strategyproof online mechanism

Theorem (Reiffenhäuser, 2019)
There exists a strategyproof $e$-approximation for the online bipartite matching problem with uniform random arrivals of the bidders. The mechanism keeps track of items $J \subseteq M$ not yet allocated.

Upon arrival of bidder $z_i$, it computes VCG-price for every unallocated item in $J$:

$$p_j(k) = \text{OPT}(\{z_1, \ldots, z_{i-1}\}, J) - \text{OPT}(\{z_1, \ldots, z_{i-1}\}, J\{j\}).$$

If there exists at least one item $j \in J$ for which $b_{ij} \geq p_j(k)$, then we assign an item $j^* = \arg\max\{b_{ij} - p_j(k) : j \in J\}$ to bidder $i$, and set $J = J\{j^*\}$.

We charge price $p_j^*(k)$ to bidder $i$. 


Theorem (Reiffenhäuser, 2019)

There exists a strategyproof $\frac{1}{e}$-approximation for the online bipartite matching problem with uniform random arrivals of the bidders.
Strategyproof online mechanism

**Theorem (Reiffenhäuser, 2019)**

There exists a strategyproof $\frac{1}{e}$-approximation for the online bipartite matching problem with uniform random arrivals of the bidders.

Mechanism keeps track of items $J \subseteq M$ not yet allocated.
There exists a strategyproof $\frac{1}{e}$-approximation for the online bipartite matching problem with uniform random arrivals of the bidders.

Mechanism keeps track of items $J \subseteq M$ not yet allocated.

- Upon arrival of bidder $z_i$, it computes VCG-price for every unallocated item in $J$:
  \[ p_j(k) = \text{OPT}(\{z_1, \ldots, z_{i-1}\}, J) - \text{OPT}(\{z_1, \ldots, z_{i-1}\}, J \setminus \{j\}). \]
Strategyproof online mechanism

Theorem (Reiffenhäuser, 2019)

There exists a strategyproof $\frac{1}{e}$-approximation for the online bipartite matching problem with uniform random arrivals of the bidders.

Mechanism keeps track of items $J \subseteq M$ not yet allocated.

- Upon arrival of bidder $z_i$, it computes VCG-price for every unallocated item in $J$:
  $$p_j(k) = \text{OPT}(\{z_1, \ldots, z_{i-1}\}, J) - \text{OPT}(\{z_1, \ldots, z_{i-1}\}, J \setminus \{j\}).$$

- If there exists at least one item $j \in J$ for which $b_{ij} \geq p_j(k)$, then we assign an item
  $$j^* = \arg\max\{b_{ij} - p_j(k) : j \in J\}$$
  to bidder $i$, and set $J = J \setminus \{j^*\}$. 
Strategyproof online mechanism

**Theorem (Reiffenhäuser, 2019)**

There exists a strategyproof $\frac{1}{e}$-approximation for the online bipartite matching problem with uniform random arrivals of the bidders.

Mechanism keeps track of items $J \subseteq M$ not yet allocated.

- Upon arrival of bidder $z_i$, it computes VCG-price for every unallocated item in $J$:
  
  $$p_j(k) = \text{OPT}(\{z_1, \ldots, z_{i-1}\}, J) - \text{OPT}(\{z_1, \ldots, z_{i-1}\}, J \setminus \{j\}).$$

- If there exists at least one item $j \in J$ for which $b_{ij} \geq p_j(k)$, then we assign an item
  
  $$j^* = \arg\max\{b_{ij} - p_j(k) : j \in J\}$$

  to bidder $i$, and set $J = J \setminus \{j^*\}$.

- We charge price $p_{j^*}(k)$ to bidder $i$. 
Theorem (Reiffenhäuser, 2019)

There exists a strategyproof $\epsilon$-approximation for the online bipartite matching problem with uniform random arrivals of the bidders. Although the algorithm is still relatively simple to describe, analysis is much harder.
Theorem (Reiffenhäuser, 2019)

There exists a strategyproof $\frac{1}{e}$-approximation for the online bipartite matching problem with uniform random arrivals of the bidders.
There exists a strategyproof $\frac{1}{e}$-approximation for the online bipartite matching problem with uniform random arrivals of the bidders.

Although the algorithm is still relatively simple to describe, analysis is much harder.