Topics in Algorithmic Game Theory and Economics

Pieter Kleer

Max Planck Institute for Informatics (D1)
Saarland Informatics Campus

January 20, 2020

Lecture 9
Online Bipartite Matching

Offline bipartite matching

Offline bipartite matching

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

Offline bipartite matching

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

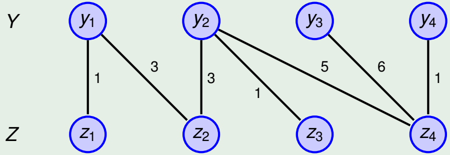
- Edge weight function $w : E \to \mathbb{R}_{\geq 0}$.

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

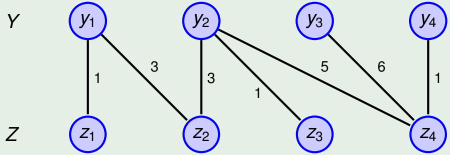


Offline bipartite matching

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



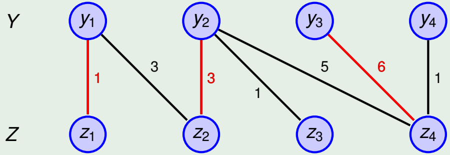
- Matching $M \subseteq E$ is set of edges where every node is incident to at most one edge from M:

Offline bipartite matching

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

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

Example



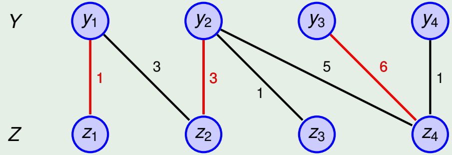
- Matching $M \subseteq E$ is set of edges where every node is incident to at most one edge from M: $|\{e \in M : e \cap \{v\}\}| \leq 1 \ \forall v \in Y \cup Z$.

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

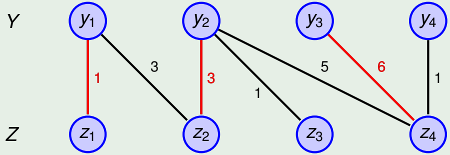
$$w(M) = \sum_{e \in M} w_e.$$

Offline bipartite matching

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

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

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

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

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.:

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

Vertex arrival model

We consider the following (semi)-online model:

Vertex arrival model

We consider the following (semi)-online model:

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

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 σ.

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 σ.
  - 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 σ.
  - 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 σ.
    - 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).

Vertex arrival model

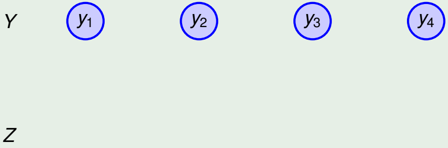
We consider the following (semi)-online model:

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

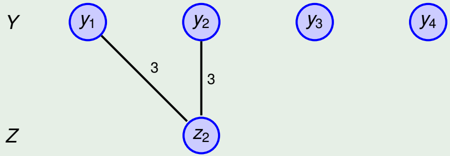
We consider the following (semi)-online model:

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

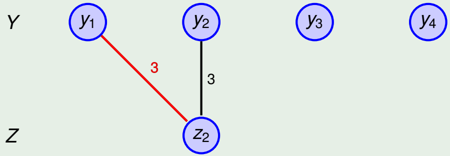
We consider the following (semi)-online model:

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

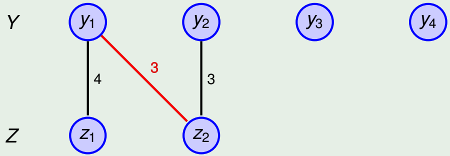
We consider the following (semi)-online model:

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

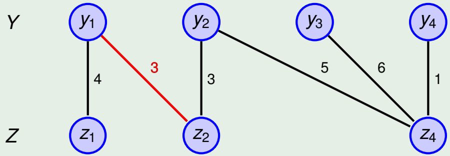
We consider the following (semi)-online model:

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

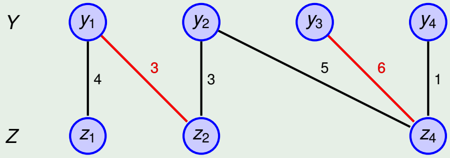
We consider the following (semi)-online model:

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

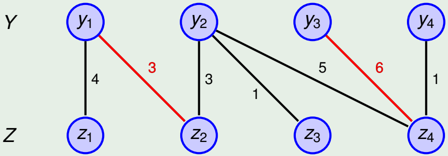
We consider the following (semi)-online model:

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

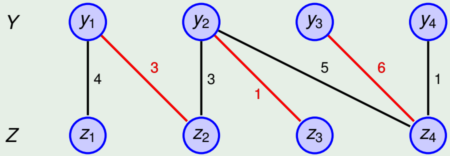
We consider the following (semi)-online model:

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

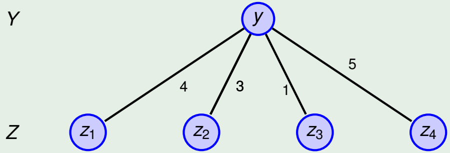
Missing edges have weight $w_{xy} = 0$. Suppose $\sigma = (2, 1, 4, 3)$.



Generalization of secretary problem (with uniform random arrivals).

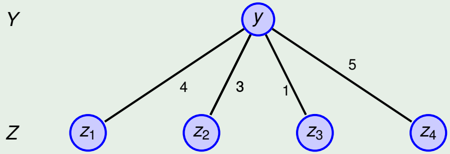
Generalization of secretary problem (with uniform random arrivals).

Example



Generalization of secretary problem (with uniform random arrivals).

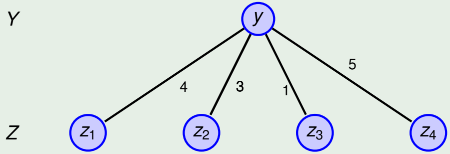
Example



Remark

Generalization of secretary problem (with uniform random arrivals).

Example

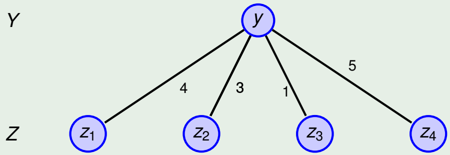


Remark

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

Generalization of secretary problem (with uniform random arrivals).

Example



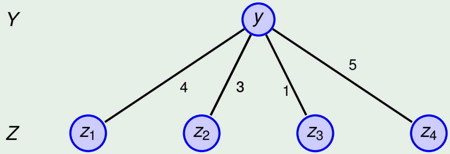
Remark

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

- Model where all nodes arrive online.

Generalization of secretary problem (with uniform random arrivals).

Example



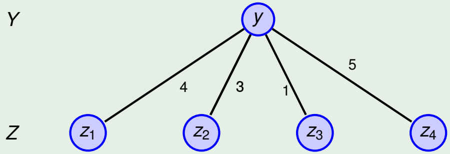
Remark

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



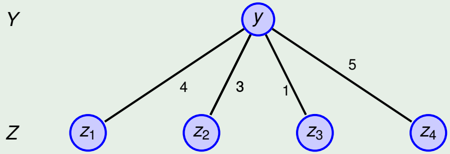
Remark

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



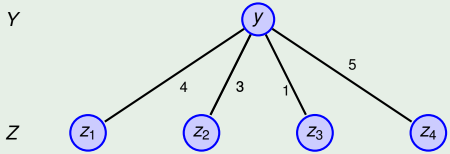
Remark

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

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

Constant-factor approximations

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

Constant-factor approximations

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

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

Constant-factor approximations

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

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

- OPT is weight of an (offline) maximum weight matching.

Constant-factor approximations

Deterministic, or randomized, algorithm $\mathcal{A}$ is α-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 σ.

Constant-factor approximations

Deterministic, or randomized, algorithm $\mathcal{A}$ is α-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 σ.

Know results:

Constant-factor approximations

Deterministic, or randomized, algorithm $\mathcal{A}$ is α-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 σ.

Know results:

- [Babaioff-Immorlica-Kempe-Kleinberg, 2007]
    - $\frac{1}{16}$-approximation for special case of uniform edge weights.

Constant-factor approximations

Deterministic, or randomized, algorithm $\mathcal{A}$ is α-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 σ.

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.

Constant-factor approximations

Deterministic, or randomized, algorithm $\mathcal{A}$ is α-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 σ.

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

Constant-factor approximations

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

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

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

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].
  - $\left(\frac{1}{e} - \frac{1}{n}\right)$-approximation.

Constant-factor approximations

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

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

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

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].
  - $\left(\frac{1}{e} - \frac{1}{n}\right)$-approximation.
  - Best possible!

Constant-factor approximations

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

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

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

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].
  - $\left(\frac{1}{e} - \frac{1}{n}\right)$-approximation.
  - Best possible! Will see this algorithm later.

Constant-factor approximations

Deterministic, or randomized, algorithm $\mathcal{A}$ is α-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 σ.

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].
  - $\left(\frac{1}{e} - \frac{1}{n}\right)$-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

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 $w_{yz} \in \{0, v_i\}$.

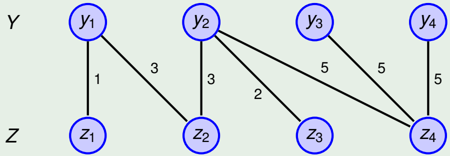
- If we interpret edges with weight zero as non-existent, then every edge adjacent to z has same weight.

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.

Example



Online bipartite matching

KRTV-algorithm

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.*

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.

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.

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 $|Y| = 1$.

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.

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.

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:

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:

- Assume arrival order is written as $\sigma = (z_1, \ldots, z_m)$.

KRTV-algorithm

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.

Notation:

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

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:

- 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 $\{y', z'\} \in E$.

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

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

Algorithm constructs an online matching M.

- $\mathrm{OPT}(Z', Y') := w(M^*(Z', Y'))$ 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 $\sigma = (z_1, \ldots, z_m)$

- $\text{OPT}(Z', Y') := w(M^*(Z', Y'))$ 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 $M = \emptyset$.

- $\text{OPT}(Z', Y') := w(M^*(Z', Y'))$ 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 $i = 1, \ldots, \lfloor \frac{m}{e} \rfloor$:

- $\mathrm{OPT}(Z', Y') := w(M^*(Z', Y'))$ 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.

- $\text{OPT}(Z', Y') := w(M^*(Z', Y'))$ 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 $i = 1, \ldots, \lfloor \frac{m}{e} \rfloor$:

- Do not match up z_i.

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

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

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 $M^*(\{z_1, \ldots, z_i\} \cup Y)$.

- $\mathrm{OPT}(Z', Y') := w(M^*(Z', Y'))$ 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

- $\text{OPT}(Z', Y') := w(M^*(Z', Y'))$ 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 $i = 1, \ldots, \lfloor \frac{m}{e} \rfloor$:

- 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 $y \in Y$

- $\text{OPT}(Z', Y') := w(M^*(Z', Y'))$ 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 $i = 1, \ldots, \lfloor \frac{m}{e} \rfloor$:

- 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,

- 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 $\sigma = (z_1, \dots, z_m)$

Set $M = \emptyset$.

Phase I (Observation): For $i = 1, \dots, \lfloor \frac{m}{e} \rfloor$:

- Do not match up z_i.

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

- Compute optimal (offline) matching $M^*(\{z_1, \dots, z_i\} \cup Y)$.
- 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,

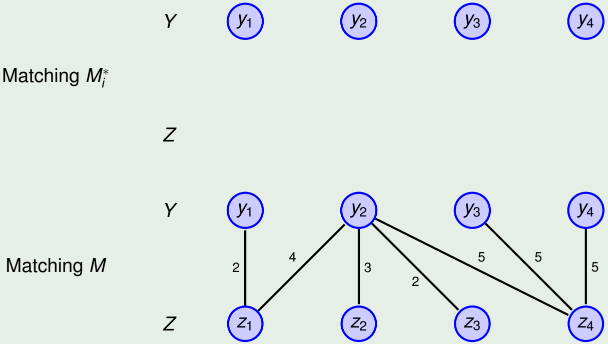
  then set $M = M \cup \{z_i, y\}$.

ALGORITHM 1: KRTV-algorithm for online bipartite matching

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

Set $M = \emptyset$.
for $i = 1, \dots, \lfloor m/e \rfloor$ **do**
  Do nothing
end
for $i = \lfloor m/e \rfloor + 1, \dots, m$ **do**
  Compute optimal matching $M_i^* = M^*(\{z_1, \dots, z_i\}, Y)$ 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 $i = 1, \dots, m$)



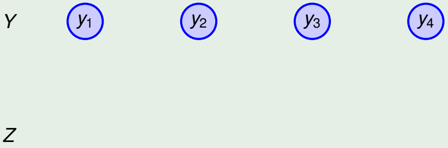
ALGORITHM 2: KRTV-algorithm for online bipartite matching

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.

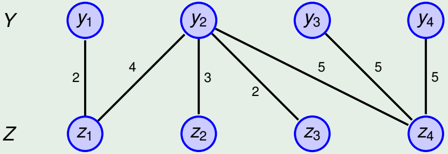
Set $M = \emptyset$.
for $i = 1, \ldots, \lfloor m/e \rfloor$ **do**
  Do nothing
end
for $i = \lfloor m/e \rfloor + 1, \ldots, m$ **do**
  Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $\mathcal{A}$
  if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ **then**
    Set $M \leftarrow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

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

Matching M_i^*



Matching M



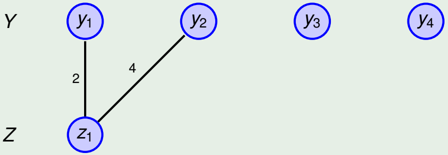
ALGORITHM 3: KRTV-algorithm for online bipartite matching

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.

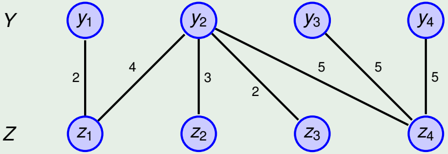
Set $M = \emptyset$.
for $i = 1, \ldots, \lfloor m/e \rfloor$ **do**
    Do nothing
end
for $i = \lfloor m/e \rfloor + 1, \ldots, m$ **do**
    Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $\mathcal{A}$
    if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ **then**
        Set $M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

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

Matching M_i^*



Matching M



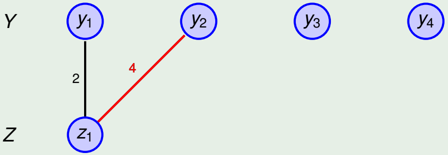
ALGORITHM 4: KRTV-algorithm for online bipartite matching

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

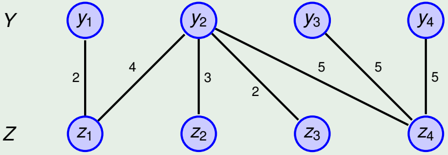
Set $M = \emptyset$.
for $i = 1, \ldots, \lfloor m/e \rfloor$ **do**
  Do nothing
end
for $i = \lfloor m/e \rfloor + 1, \ldots, m$ **do**
  Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $\mathcal{A}$
  if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ **then**
    Set $M \leftarrow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

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

Matching M_i^*



Matching M

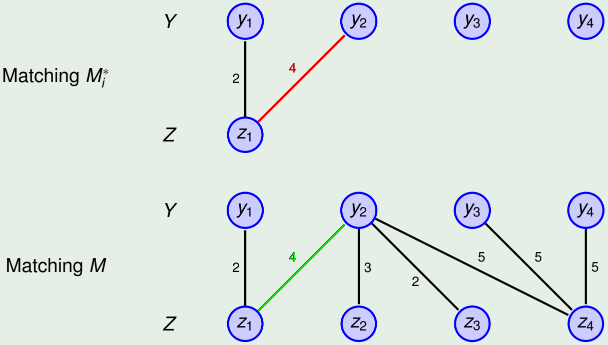


ALGORITHM 5: KRTV-algorithm for online bipartite matching

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 $i = 1, \ldots, m$)

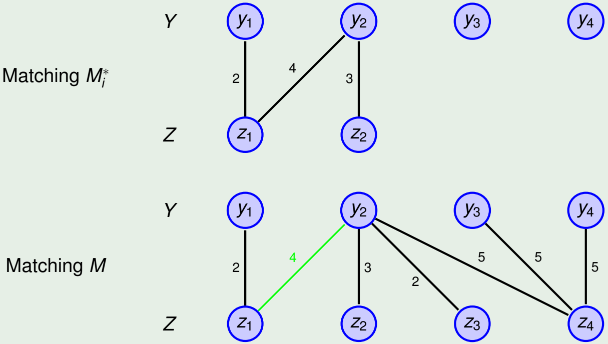


ALGORITHM 6: KRTV-algorithm for online bipartite matching

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 $i = 1, \ldots, m$)

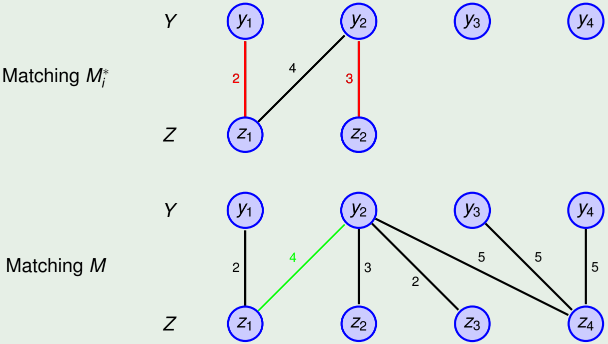


ALGORITHM 7: KRTV-algorithm for online bipartite matching

Input : Bipartite graph $B = (Z \cup Y, E)$ and weights $w : E \to \mathbb{R}_{\geq 0}$.
Deterministic algorithm $\mathcal{A}$ 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 $i = 1, \ldots, m$)

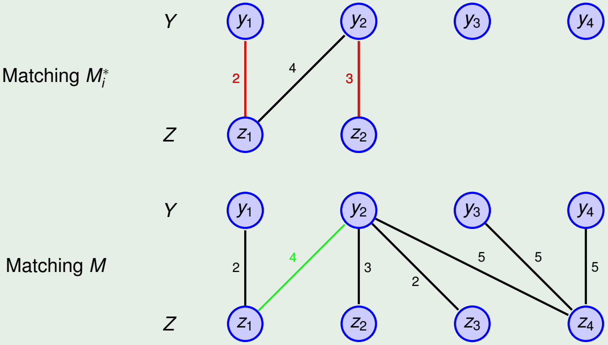


ALGORITHM 8: KRTV-algorithm for online bipartite matching

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 $i = 1, \ldots, m$)



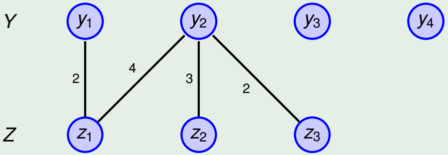
ALGORITHM 9: KRTV-algorithm for online bipartite matching

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.

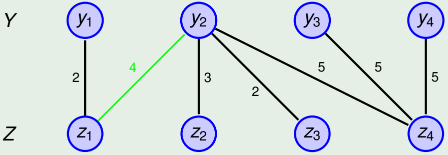
Set $M = \emptyset$.
for $i = 1, \ldots, \lfloor m/e \rfloor$ **do**
  Do nothing
end
for $i = \lfloor m/e \rfloor + 1, \ldots, m$ **do**
  Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $\mathcal{A}$
  if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ **then**
    Set $M \leftarrow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

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

Matching M_i^*



Matching M



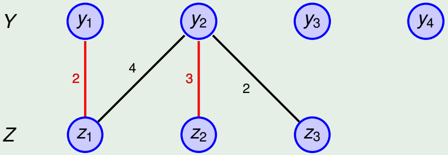
ALGORITHM 10: KRTV-algorithm for online bipartite matching

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.

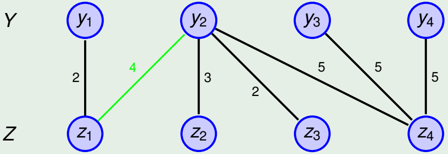
Set $M = \emptyset$.
for $i = 1, \ldots, \lfloor m/e \rfloor$ **do**
  Do nothing
end
for $i = \lfloor m/e \rfloor + 1, \ldots, m$ **do**
  Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $\mathcal{A}$
  if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ **then**
    Set $M \leftarrow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

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

Matching M_i^*



Matching M



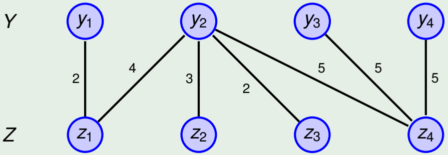
ALGORITHM 11: KRTV-algorithm for online bipartite matching

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.

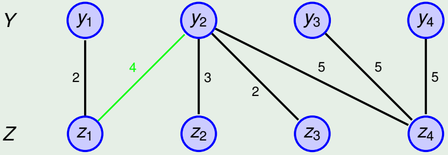
Set $M = \emptyset$.
for $i = 1, \ldots, \lfloor m/e \rfloor$ **do**
  Do nothing
end
for $i = \lfloor m/e \rfloor + 1, \ldots, m$ **do**
  Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $\mathcal{A}$
  if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ **then**
    Set $M \leftarrow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

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

Matching M_i^*



Matching M

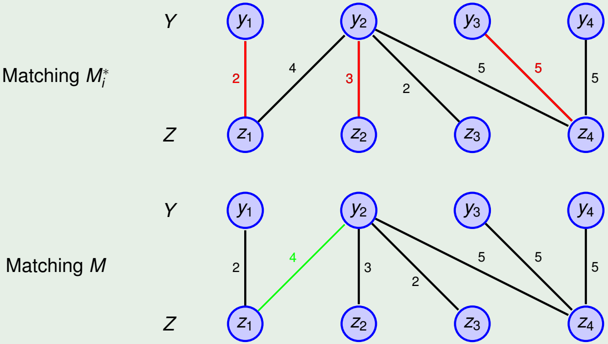


ALGORITHM 12: KRTV-algorithm for online bipartite matching

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 $i = 1, \ldots, m$)



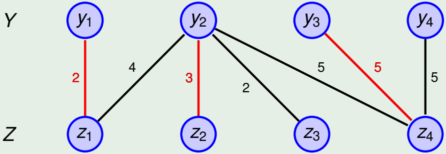
ALGORITHM 13: KRTV-algorithm for online bipartite matching

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.

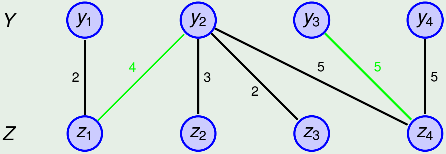
Set $M = \emptyset$.
for $i = 1, \ldots, \lfloor m/e \rfloor$ **do**
  Do nothing
end
for $i = \lfloor m/e \rfloor + 1, \ldots, m$ **do**
  Compute optimal matching $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ using $\mathcal{A}$
  if $\{z_i, y\} \in M_i^*$ for some $y \in Y$ **then**
    Set $M \leftarrow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

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

Matching M_i^*



Matching M



Online bipartite matching

KRTV-algorithm: Sketch of analysis

Analysis (sketch)

ALGORITHM 14: 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 = ∅.
for i = 1, ..., ⌊m/e⌋ do
|   Do nothing
end
for i = ⌊m/e⌋ + 1, ..., m do
|   Compute optimal matching M*_i = M*({z_1, ..., z_i}, Y) using A
|   if {z_i, y} ∈ M*_i for some y ∈ Y then
|   |   Set M ← M ∪ {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 $\mathcal{A}$ 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

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

Analysis (sketch)

ALGORITHM 16: 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 = ∅.
for i = 1, ..., ⌊m/e⌋ do
  | Do nothing
end
for i = ⌊m/e⌋ + 1, ..., m do
  | Compute optimal matching M*_i = M*({z_1, ..., z_i}, Y) using 𝒜
  | if {z_i, y} ∈ M*_i for some y ∈ Y then
  |   | Set M ← M ∪ {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.)

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

```
Set M = ∅.
for i = 1, ..., ⌊m/e⌋ do
  | Do nothing
end
for i = ⌊m/e⌋ + 1, ..., m do
  | Compute optimal matching M*_i = M*({z_1, ..., z_i}, Y) using A
  | if {z_i, y} ∈ M*_i for some y ∈ Y then
  |   | Set M ← M ∪ {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 σ, we have

Analysis (sketch)

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

```
Set M = ∅.
for i = 1, ..., ⌊m/e⌋ do
  | Do nothing
end
for i = ⌊m/e⌋ + 1, ..., m do
  | Compute optimal matching M*_i = M*({z_1, ..., z_i}, Y) using A
  | if {z_i, y} ∈ M*_i for some y ∈ Y then
  |   | Set M ← M ∪ {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 σ, we have

$$A_i = \begin{cases} w_{ir} & \text{if } i \text{ gets matched up with } r \text{ under } \sigma, \\ 0 & \text{otherwise.} \end{cases}$$

Analysis (sketch)

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

```
Set M = ∅.
for i = 1, ..., ⌊m/e⌋ do
  | Do nothing
end
for i = ⌊m/e⌋ + 1, ..., m do
  | Compute optimal matching M*_i = M*({z_1, ..., z_i}, Y) using A
  | if {z_i, y} ∈ M*_i for some y ∈ Y then
  |   | Set M ← M ∪ {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 σ, 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:

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

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

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

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

$$\mathbb{E}_\sigma[w(M)]$$

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

$$\mathbb{E}_\sigma[w(M)] = \sum_{i=\lfloor m/e \rfloor + 1}^{m} \mathbb{E}_\sigma[A_i]$$

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

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

Two claims:

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

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

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

$$\begin{aligned}
\mathbb{E}_\sigma[w(M)] &= \sum_{i=\lfloor m/e \rfloor+1}^{m} \mathbb{E}_\sigma[A_i] \geq \sum_{i=\lfloor m/e \rfloor+1}^{m} \frac{\text{OPT}}{m} \frac{\lfloor m/e \rfloor}{i-1} \\
&= \frac{\lfloor m/e \rfloor}{m} \cdot \text{OPT} \cdot \sum_{i=\lfloor m/e \rfloor+1}^{m} \frac{1}{i-1}
\end{aligned}$$

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

$$\begin{aligned}
\mathbb{E}_\sigma[w(M)] &= \sum_{i=\lfloor m/e \rfloor+1}^{m} \mathbb{E}_\sigma[A_i] \geq \sum_{i=\lfloor m/e \rfloor+1}^{m} \frac{\text{OPT}}{m} \frac{\lfloor m/e \rfloor}{i-1} \\
&= \frac{\lfloor m/e \rfloor}{m} \cdot \text{OPT} \cdot \sum_{i=\lfloor m/e \rfloor+1}^{m} \frac{1}{i-1} \\
&\geq \left(\frac{1}{e} - \frac{1}{m}\right) \cdot \text{OPT} \cdot 1
\end{aligned}$$

Offline mechanism design (recap)

Recap offline setting

Unit-demand setting:

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 $N = \{1, \ldots, n\}$

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 $v_i : M \to \mathbb{R}_{\geq 0}$.

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.

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 $b_i : M \to \mathbb{R}_{\geq 0}$.

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.

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.

Recap offline setting

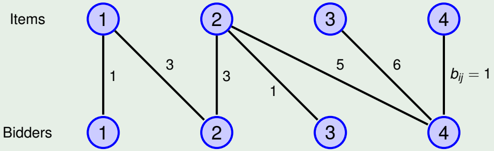
Unit-demand setting:

- 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.

Example

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



Definition (Mechanism)

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 $p : \mathbb{R}_{\geq 0}^{n \times m} \to \mathbb{R}_{\geq 0}^{m}$.

Definition (Mechanism)

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 $p : \mathbb{R}_{\geq 0}^{n \times m} \to \mathbb{R}_{\geq 0}^{m}$.

- 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

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

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

- 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 $p = p(b)$.

Definition (Mechanism)

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 $p : \mathbb{R}_{\geq 0}^{n \times m} \to \mathbb{R}_{\geq 0}^{m}$.

- 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

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

Definition (Mechanism)

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 $p : \mathbb{R}_{\geq 0}^{n \times m} \to \mathbb{R}_{\geq 0}^{m}$.

- 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

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

Desired properties:

Definition (Mechanism)

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

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

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

- 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

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

Desired properties:

- *Strategyproof:*

Definition (Mechanism)

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 $p : \mathbb{R}_{\geq 0}^{n \times m} \to \mathbb{R}_{\geq 0}^{m}$.

- 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

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

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

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

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

- 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

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

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

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

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

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

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

Desired properties:

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

$$u_i(b_{-i}, v_i) \geq u_i(b_{-i}, b_i')$$

    for all $b_{-i} = (b_1, \dots, b_{i-1}, b_{i+1}, \dots, b_n)$ and other bid vector b_i'.
- Also would like to have *individual rationality*, *welfare maximization*, and *computational tractability*.

Vickrey-Clarke-Groves (VCG) mechanism

Vickrey-Clarke-Groves (VCG) mechanism

Notation:

Vickrey-Clarke-Groves (VCG) mechanism

Notation:

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

Vickrey-Clarke-Groves (VCG) mechanism

Notation:

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

Vickrey-Clarke-Groves (VCG) mechanism

Notation:

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

Vickrey-Clarke-Groves (VCG) mechanism

Notation:

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

Vickrey-Clarke-Groves (VCG) mechanism

Notation:

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

$$p_{ij}(b) = \text{OPT}(N \setminus \{i\}, M) - \text{OPT}(N \setminus \{i\}, M \setminus \{j\}),$$

Vickrey-Clarke-Groves (VCG) mechanism

Notation:

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

$$p_{ij}(b) = \text{OPT}(N \setminus \{i\}, M) - \text{OPT}(N \setminus \{i\}, M \setminus \{j\}),$$

  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

Selling multiple items online

Setting:

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.

Bidders 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 $\sigma(k)$.

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 $p(b_{\sigma(1)}, \ldots, b_{\sigma(k)})$.

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 ℓ 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 ℓ 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 ℓ 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 $(y_1, \dots, y_n)$ and bid vectors $b_1, \dots, b_n$ for the item.

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

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 ℓ 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 ℓ 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 ℓ 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 σ.

An observation regarding the KRTV-algorithm

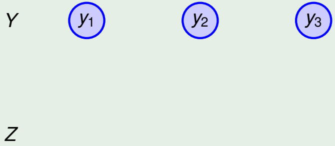
ALGORITHM 20: KRTV-algorithm for online bipartite matching

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

Set $M = \emptyset$.
for $i = 1, \dots, \lfloor m/e \rfloor$ **do**
  Do nothing
end
for $i = \lfloor m/e \rfloor + 1, \dots, m$ **do**
  Compute optimal matching $M_i^* = M^*(\{z_1, \dots, z_i\}, Y)$ 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 $i = 1, \dots, m$)

Online process



An observation regarding the KRTV-algorithm

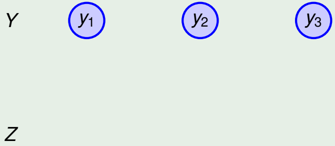
ALGORITHM 21: KRTV-algorithm for online bipartite matching

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 = ∅.
for i = 1, ..., ⌊m/e⌋ do
|   Do nothing
end
for i = ⌊m/e⌋ + 1, ..., m do
|   Compute optimal matching M_i^* = M^*({z_1, ..., z_i}, Y) using A
|   if {z_i, y} ∈ M_i^* for some y ∈ Y then
|   |   Set M ← M ∪ {z_i, y} if y is unmatched in M.
|   end
end
return M
```

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

Online process



An observation regarding the KRTV-algorithm

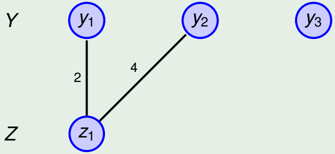
ALGORITHM 22: KRTV-algorithm for online bipartite matching

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 $\{z_i, y\} \in M_i^*$ 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



An observation regarding the KRTV-algorithm

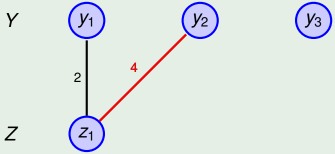
ALGORITHM 23: KRTV-algorithm for online bipartite matching

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 $i = 1, \dots, \lfloor m/e \rfloor$ **do**
  Do nothing
end
for $i = \lfloor m/e \rfloor + 1, \dots, m$ **do**
  Compute optimal matching $M_i^* = M^*(\{z_1, \dots, z_i\}, Y)$ 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 $i = 1, \dots, m$)

Online process



An observation regarding the KRTV-algorithm

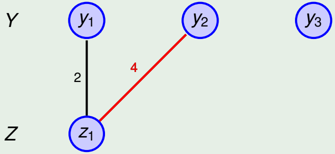
ALGORITHM 24: KRTV-algorithm for online bipartite matching

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



An observation regarding the KRTV-algorithm

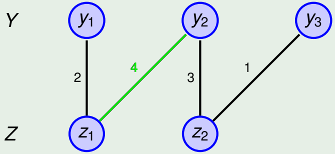
ALGORITHM 25: KRTV-algorithm for online bipartite matching

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 $M_i^* = M^*(\{z_1, \ldots, z_i\}, Y)$ 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



An observation regarding the KRTV-algorithm

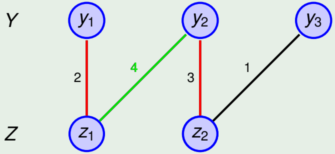
ALGORITHM 26: KRTV-algorithm for online bipartite matching

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



An observation regarding the KRTV-algorithm

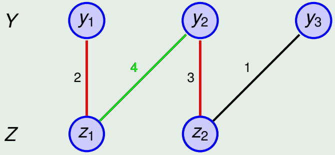
ALGORITHM 27: KRTV-algorithm for online bipartite matching

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

Set $M = \emptyset$.
for $i = 1, \dots, \lfloor m/e \rfloor$ **do**
  Do nothing
end
for $i = \lfloor m/e \rfloor + 1, \dots, m$ **do**
  Compute optimal matching $M_i^* = M^*(\{z_1, \dots, z_i\}, Y)$ 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 $i = 1, \dots, m$)

Online process



- 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

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.

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.

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:

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

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

$$j^* = \text{argmax}\{b_{ij} - p_j(k) : j \in J\}$$

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:

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

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

$$j^* = \mathrm{argmax}\{b_{ij} - p_j(k) : j \in J\}$$

to bidder i, and set $J = J \setminus \{j^*\}$.
- We charge price $p_{j^*}(k)$ to bidder i.

Strategyproof online mechanism

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.

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.

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