Critical Relaxed Stable Matchings with Two-Sided Ties

Keshav Ranjan

IIT Madras, India

joint work with Meghana Nasre (IITM) and Prajakta Nimbhorkar(CMI)

CS Theory Seminar

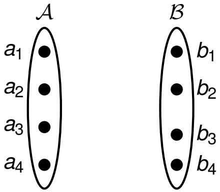
(Chennai Mathematical Institute, Chennai)

Sept 22, 2023

Problem Setup

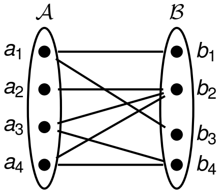
Stable Marriage Problem with Ties and Critical Agents

- A bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$
  - **Vertex set $\mathcal{A} \cup \mathcal{B}$:** $\mathcal{A} = \{a_1, \ldots, a_{n_1}\}$ $\mathcal{B} = \{b_1, \ldots, b_{n_2}\}$



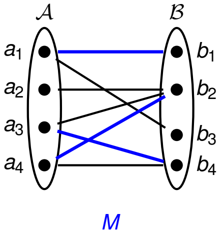
Stable Marriage Problem with Ties and Critical Agents

- A bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$
  - **Vertex set $\mathcal{A} \cup \mathcal{B}$:** $\mathcal{A} = \{a_1, \ldots, a_{n_1}\}$ $\mathcal{B} = \{b_1, \ldots, b_{n_2}\}$
  - **Edge set $E \subseteq \mathcal{A} \times \mathcal{B}$:** Mutually acceptable agent-pairs



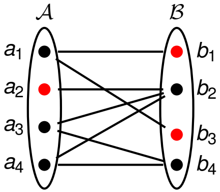
Stable Marriage Problem with Ties and Critical Agents

- A bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$
  - **Vertex set $\mathcal{A} \cup \mathcal{B}$:** $\mathcal{A} = \{a_1, \ldots, a_{n_1}\}$ $\mathcal{B} = \{b_1, \ldots, b_{n_2}\}$
  - **Edge set $E \subseteq \mathcal{A} \times \mathcal{B}$:** Mutually acceptable agent-pairs
- **Matching M:** Set of *independent* edges



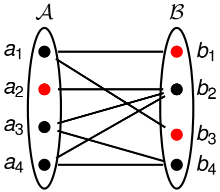
Stable Marriage Problem with Ties and Critical Agents

- A bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$
  - **Vertex set $\mathcal{A} \cup \mathcal{B}$:** $\mathcal{A} = \{a_1, \ldots, a_{n_1}\}$ $\mathcal{B} = \{b_1, \ldots, b_{n_2}\}$
  - **Edge set $E \subseteq \mathcal{A} \times \mathcal{B}$:** Mutually acceptable agent-pairs
- **Matching M:** Set of *independent* edges
- **Critical agents**: $C \subseteq \mathcal{A} \cup \mathcal{B}$



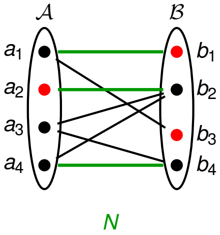
Stable Marriage Problem with Ties and Critical Agents

- A bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$
  - **Vertex set $\mathcal{A} \cup \mathcal{B}$:** $\mathcal{A} = \{a_1, \ldots, a_{n_1}\}$ $\mathcal{B} = \{b_1, \ldots, b_{n_2}\}$
  - **Edge set $E \subseteq \mathcal{A} \times \mathcal{B}$:** Mutually acceptable agent-pairs
- **Matching M:** Set of *independent* edges
- **Critical agents**: $C \subseteq \mathcal{A} \cup \mathcal{B}$
- **Feasible Matching**: Matches all critical agents



Stable Marriage Problem with Ties and Critical Agents

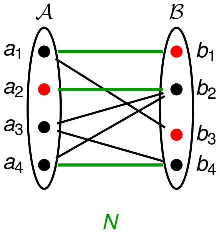
- A bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$
  - **Vertex set $\mathcal{A} \cup \mathcal{B}$:** $\mathcal{A} = \{a_1, \ldots, a_{n_1}\}$ $\quad \mathcal{B} = \{b_1, \ldots, b_{n_2}\}$
  - **Edge set $E \subseteq \mathcal{A} \times \mathcal{B}$:** Mutually acceptable agent-pairs
- **Matching M:** Set of *independent* edges
- **Critical agents**: $C \subseteq \mathcal{A} \cup \mathcal{B}$
- **Feasible Matching**: Matches all critical agents
- **Critical matching N**: No other matching matches *more* critical agents than N



N matches two critical nodes a_2 and b_1

Stable Marriage Problem with Ties and Critical Agents

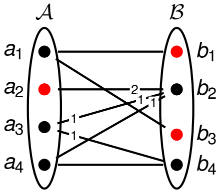
- A bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$
  - **Vertex set $\mathcal{A} \cup \mathcal{B}$:** $\mathcal{A} = \{a_1, \ldots, a_{n_1}\}$ $\mathcal{B} = \{b_1, \ldots, b_{n_2}\}$
  - **Edge set $E \subseteq \mathcal{A} \times \mathcal{B}$:** Mutually acceptable agent-pairs
- **Matching M:** Set of *independent* edges
- **Critical agents**: $C \subseteq \mathcal{A} \cup \mathcal{B}$
- **Feasible Matching**: Matches all critical agents
- **Critical matching N**: No other matching matches *more* critical agents than N



Critical matching

Stable Marriage Problem with Ties and Critical Agents

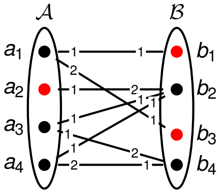
- A bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$
  - **Vertex set $\mathcal{A} \cup \mathcal{B}$:** $\mathcal{A} = \{a_1, \ldots, a_{n_1}\}$ $\mathcal{B} = \{b_1, \ldots, b_{n_2}\}$
  - **Edge set $E \subseteq \mathcal{A} \times \mathcal{B}$:** Mutually acceptable agent-pairs
- **Matching M:** Set of *independent* edges
- **Critical agents**: $C \subseteq \mathcal{A} \cup \mathcal{B}$
- **Feasible Matching**: Matches all critical agents
- **Critical matching N**: No other matching matches *more* critical agents than N
- **Preference list**: *Ranking* over the acceptable agents (**ties** are allowed)



a_3 and a_4 are in a **tie** at **rank 1** for b_2

Stable Marriage Problem with Ties and Critical Agents

- A bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$
  - **Vertex set $\mathcal{A} \cup \mathcal{B}$:** $\mathcal{A} = \{a_1, \ldots, a_{n_1}\}$ $\mathcal{B} = \{b_1, \ldots, b_{n_2}\}$
  - **Edge set $E \subseteq \mathcal{A} \times \mathcal{B}$:** Mutually acceptable agent-pairs
- **Matching M:** Set of *independent* edges
- **Critical agents**: $C \subseteq \mathcal{A} \cup \mathcal{B}$
- **Feasible Matching**: Matches all critical agents
- **Critical matching N**: No other matching matches *more* critical agents than N
- **Preference list**: *Ranking* over the acceptable agents (**ties** are allowed)
- **Goal**: Compute a *critical* matching that is *optimal* w.r.t. preferences



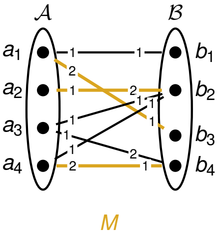
Optimality Notions

Optimality Notion

- Stability (without critical nodes) [Gale and Shapley, 1962]

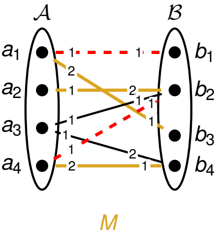
Optimality Notion

- Stability (without critical nodes) [Gale and Shapley, 1962]
  - A pair $(a, b) \notin M$ **blocks** a matching M if both a and b have *incentive* to deviate from M



Optimality Notion

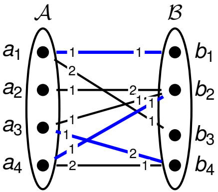
- Stability (without critical nodes) [Gale and Shapley, 1962]
  - A pair $(a, b) \notin M$ blocks a matching M if both a and b have *incentive* to deviate from M



(a_1, b_1) and (a_4, b_2) are blocking pairs

Optimality Notion

- Stability (without critical nodes) [Gale and Shapley, 1962]
  - A pair $(a, b) \notin M$ **blocks** a matching M if both a and b have *incentive* to deviate from M
  - A matching M is **stable** if no vertex-pair blocks it



M_s

Stable matching

Optimality Notion

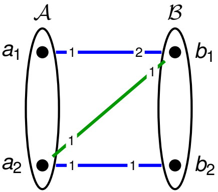
- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists

Optimality Notion

- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists
  - For **strict** list instance, all stable matchings have the **same size**

Optimality Notion

- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists
  - For **strict** list instance, all stable matchings have the **same size**
  - For **tied** list instance, stable matchings may have **different sizes**



Optimality Notion

- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists
  - For **strict** list instance, all stable matchings have the **same size**
  - For **tied** list instance, stable matchings may have **different sizes**
  - Computing a **maximum-size** stable matching is **NP-hard** [Manlove et al., 2002]

Optimality Notion

- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists
  - For **strict** list instance, all stable matchings have the **same size**
  - For **tied** list instance, stable matchings may have **different sizes**
  - Computing a **maximum-size** stable matching is **NP-hard** [Manlove et al., 2002]
  - $\frac{3}{2}$**-approximation** of maximum size stable matching [Király, 2013]

Optimality Notion

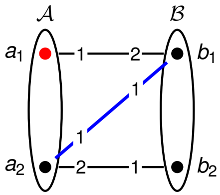
- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists
  - For **strict** list instance, all stable matchings have the **same size**
  - For **tied** list instance, stable matchings may have **different sizes**
  - Computing a **maximum-size** stable matching is **NP-hard** [Manlove et al., 2002]
  - $\frac{3}{2}$**-approximation** of maximum size stable matching [Király, 2013]
- Stability (in the presence of critical nodes)

Optimality Notion

- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists
  - For **strict** list instance, all stable matchings have the **same size**
  - For **tied** list instance, stable matchings may have **different sizes**
  - Computing a **maximum-size** stable matching is **NP-hard** [Manlove et al., 2002]
  - $\frac{3}{2}$**-approximation** of maximum size stable matching [Király, 2013]
- Stability (in the presence of critical nodes)
  - Stable **and** critical matching is **not guaranteed** to exist

Optimality Notion

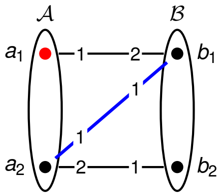
- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists
  - For **strict** list instance, all stable matchings have the **same size**
  - For **tied** list instance, stable matchings may have **different sizes**
  - Computing a **maximum-size** stable matching is **NP-hard** [Manlove et al., 2002]
  - $\frac{3}{2}$**-approximation** of maximum size stable matching [Király, 2013]
- Stability (in the presence of critical nodes)
  - Stable **and** critical matching is **not guaranteed** to exist



Only stable matching $\{(a_2, b_1)\}$ is not critical

Optimality Notion

- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists
  - For **strict** list instance, all stable matchings have the **same size**
  - For **tied** list instance, stable matchings may have **different sizes**
  - Computing a **maximum-size** stable matching is **NP-hard** [Manlove et al., 2002]
  - $\frac{3}{2}$**-approximation** of maximum size stable matching [Király, 2013]
- Stability (in the presence of critical nodes)
  - Stable **and** critical matching is **not guaranteed** to exist
  - In a **tied list** instance, computing a critical and stable matching is **NP-hard**



Only stable matching $\{(a_2, b_1)\}$ is not critical

Optimality Notion

- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists
  - For **strict** list instance, all stable matchings have the **same size**
  - For **tied** list instance, stable matchings may have **different sizes**
  - Computing a **maximum-size** stable matching is **NP-hard** [Manlove et al., 2002]
  - $\frac{3}{2}$**-approximation** of maximum size stable matching [Király, 2013]
- Stability (in the presence of critical nodes)
  - Stable **and** critical matching is **not guaranteed** to exist
  - In a **tied list** instance, computing a critical and stable matching is **NP-hard**

- Maximally Satisfying Lower-quotas (MSLQ) [Goko et al., 2022, Makino et al., 2022]

Optimality Notion

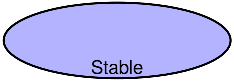
- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists
  - For **strict** list instance, all stable matchings have the **same size**
  - For **tied** list instance, stable matchings may have **different sizes**
  - Computing a **maximum-size** stable matching is **NP-hard** [Manlove et al., 2002]
  - $\frac{3}{2}$**-approximation** of maximum size stable matching [Király, 2013]
- Stability (in the presence of critical nodes)
  - Stable **and** critical matching is **not guaranteed** to exist
  - In a **tied list** instance, computing a critical and stable matching is **NP-hard**

- Maximally Satisfying Lower-quotas (MSLQ) [Goko et al., 2022, Makino et al., 2022]
  - Find a **near-critical** matching M in the set of all stable matchings

Optimality Notion

- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists
  - For **strict** list instance, all stable matchings have the **same size**
  - For **tied** list instance, stable matchings may have **different sizes**
  - Computing a **maximum-size** stable matching is **NP-hard** [Manlove et al., 2002]
  - $\frac{3}{2}$**-approximation** of maximum size stable matching [Király, 2013]
- Stability (in the presence of critical nodes)
  - Stable **and** critical matching is **not guaranteed** to exist
  - In a **tied list** instance, computing a critical and stable matching is **NP-hard**

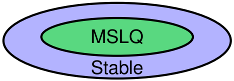
- Maximally Satisfying Lower-quotas (MSLQ) [Goko et al., 2022, Makino et al., 2022]
  - Find a **near-critical** matching *M* in the set of all stable matchings



Optimality Notion

- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists
  - For **strict** list instance, all stable matchings have the **same size**
  - For **tied** list instance, stable matchings may have **different sizes**
  - Computing a **maximum-size** stable matching is **NP-hard** [Manlove et al., 2002]
  - $\frac{3}{2}$**-approximation** of maximum size stable matching [Király, 2013]
- Stability (in the presence of critical nodes)
  - Stable **and** critical matching is **not guaranteed** to exist
  - In a **tied list** instance, computing a critical and stable matching is **NP-hard**

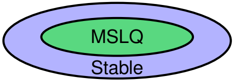
- Maximally Satisfying Lower-quotas (MSLQ) [Goko et al., 2022, Makino et al., 2022]
  - Find a **near-critical** matching M in the set of all stable matchings



Optimality Notion

- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists
  - For **strict** list instance, all stable matchings have the **same size**
  - For **tied** list instance, stable matchings may have **different sizes**
  - Computing a **maximum-size** stable matching is **NP-hard** [Manlove et al., 2002]
  - $\frac{3}{2}$**-approximation** of maximum size stable matching [Király, 2013]
- Stability (in the presence of critical nodes)
  - Stable **and** critical matching is **not guaranteed** to exist
  - In a **tied list** instance, computing a critical and stable matching is **NP-hard**

- Maximally Satisfying Lower-quotas (MSLQ) [Goko et al., 2022, Makino et al., 2022]
  - Find a **near-critical** matching M in the set of all stable matchings



  - Computing MSLQ stable matching is **NP-hard** even with one-sided ties

Optimality Notion

- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists
  - For **strict** list instance, all stable matchings have the **same size**
  - For **tied** list instance, stable matchings may have **different sizes**
  - Computing a **maximum-size** stable matching is **NP-hard** [Manlove et al., 2002]
  - $\frac{3}{2}$**-approximation** of maximum size stable matching [Király, 2013]
- Stability (in the presence of critical nodes)
  - Stable **and** critical matching is **not guaranteed** to exist
  - In a **tied list** instance, computing a critical and stable matching is **NP-hard**

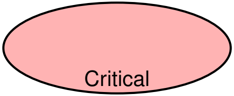
- Maximally Satisfying Lower-quotas (MSLQ) [Goko et al., 2022, Makino et al., 2022]
  - Find a **near-critical** matching M in the set of all stable matchings

  - Computing MSLQ stable matching is **NP-hard** even with one-sided ties

Optimality Notion

- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists
  - For **strict** list instance, all stable matchings have the **same size**
  - For **tied** list instance, stable matchings may have **different sizes**
  - Computing a **maximum-size** stable matching is **NP-hard** [Manlove et al., 2002]
  - $\frac{3}{2}$**-approximation** of maximum size stable matching [Király, 2013]
- Stability (in the presence of critical nodes)
  - Stable **and** critical matching is **not guaranteed** to exist
  - In a **tied list** instance, computing a critical and stable matching is **NP-hard**
- Maximally Satisfying Lower-quotas (MSLQ) [Goko et al., 2022, Makino et al., 2022]
  - Find a **near-critical** matching M in the set of all stable matchings

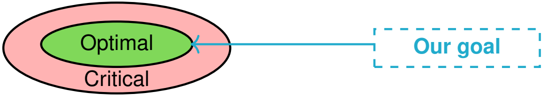
  - Computing MSLQ stable matching is **NP-hard** even with one-sided ties
- Our Goal
  - Find an **optimal** matching M in the set of all critical matchings



Optimality Notion

- Stability (without critical nodes) [Gale and Shapley, 1962]
  - Stable matching **always** exists
  - For **strict** list instance, all stable matchings have the **same size**
  - For **tied** list instance, stable matchings may have **different sizes**
  - Computing a **maximum-size** stable matching is **NP-hard** [Manlove et al., 2002]
  - $\frac{3}{2}$**-approximation** of maximum size stable matching [Király, 2013]
- Stability (in the presence of critical nodes)
  - Stable **and** critical matching is **not guaranteed** to exist
  - In a **tied list** instance, computing a critical and stable matching is **NP-hard**
- Maximally Satisfying Lower-quotas (MSLQ) [Goko et al., 2022, Makino et al., 2022]
  - Find a **near-critical** matching M in the set of all stable matchings

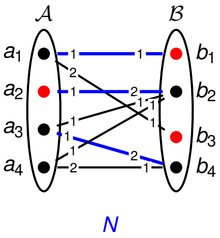
  - Computing MSLQ stable matching is **NP-hard** even with one-sided ties
- Our Goal
  - Find an **optimal** matching M in the set of all critical matchings



Relaxed Stability [Krishnaa et al., 2020]

- A matching M is **Relaxed Stable Matching** (RSM) if for every blocking pair (a, b)
  - a is matched to a critical node **or**
  - b is matched to a critical node

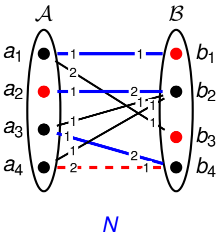
- A matching M is **Relaxed Stable Matching** (RSM) if for every blocking pair (a, b)
  - a is matched to a critical node **or**
  - b is matched to a critical node



Relaxed Stability

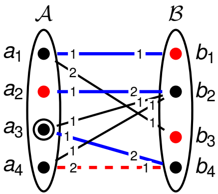
[Krishnaa et al., 2020]

- A matching M is **Relaxed Stable Matching** (RSM) if for every blocking pair (a, b)
  - a is matched to a critical node **or**
  - b is matched to a critical node



(a_4, b_4) is a blocking pair w.r.t. N

- A matching M is **Relaxed Stable Matching** (RSM) if for every blocking pair (a, b)
  - a is matched to a critical node **or**
  - b is matched to a critical node

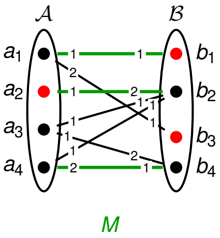


N is not an RSM

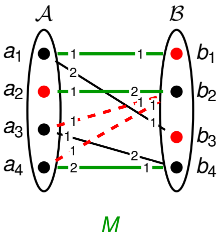
(a_4, b_4) is not justified

Relaxed Stability

- A matching M is **Relaxed Stable Matching** (RSM) if for every blocking pair (a, b)
  - a is matched to a critical node **or**
  - b is matched to a critical node



- A matching M is **Relaxed Stable Matching** (RSM) if for every blocking pair (a, b)
  - a is matched to a critical node **or**
  - b is matched to a critical node

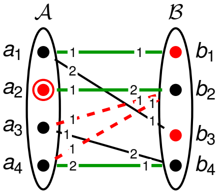


(a_3, b_2) and (a_4, b_2) are blocking pairs

Relaxed Stability

[Krishnaa et al., 2020]

- A matching M is **Relaxed Stable Matching** (RSM) if for every blocking pair (a, b)
  - a is matched to a critical node **or**
  - b is matched to a critical node



M is an RSM

Both blocking pairs are **justified** by b_2

Relaxed Stability

[Krishnaa et al., 2020]

Our goal

- Compute a matching that is **critical** and **relaxed stable**

Relaxed Stability

[Krishnaa et al., 2020]

Our goal

- Compute a matching that is **critical** and **relaxed stable**

Observation

- Computing max-size critical Relaxed Stable Matching (RSM) is **NP-hard**

Relaxed Stability

[Krishnaa et al., 2020]

Our goal

- Compute a matching that is **critical** and **relaxed stable**

Observation

- Computing max-size critical Relaxed Stable Matching (RSM) is **NP-hard**
  - **Ignore** critical nodes

Our goal

- Compute a matching that is **critical** and **relaxed stable**

Observation

- Computing max-size critical Relaxed Stable Matching (RSM) is **NP-hard**
  - **Ignore** critical nodes
  - Problem is **same** as computing max-size stable matching

Relaxed Stability
[Krishnaa et al., 2020]

Our goal

- Compute a matching that is **critical** and **relaxed stable**

Observation

- Computing max-size critical Relaxed Stable Matching (RSM) is **NP-hard**
  - **Ignore** critical nodes
  - Problem is **same** as computing max-size stable matching

- Does a critical relaxed stable matching always exist?

Relaxed Stability

[Krishnaa et al., 2020]

Our goal

- Compute a matching that is **critical** and **relaxed stable**

Observation

- Computing max-size critical Relaxed Stable Matching (RSM) is **NP-hard**
  - **Ignore** critical nodes
  - Problem is **same** as computing max-size stable matching

- Does a critical relaxed stable matching always exist?
- Can it be efficiently computed?

Relaxed Stability

[Krishnaa et al., 2020]

Our goal

- Compute a matching that is **critical** and **relaxed stable**

Observation

- Computing max-size critical Relaxed Stable Matching (RSM) is **NP-hard**
  - **Ignore** critical nodes
  - Problem is **same** as computing max-size stable matching

- Does a critical relaxed stable matching always exist?
- Can it be efficiently computed?
- The **best-known** approximation ratio for **max-size stable** matching is $\frac{3}{2}$

Relaxed Stability

[Krishnaa et al., 2020]

Our goal

- Compute a matching that is **critical** and **relaxed stable**

Observation

- Computing max-size critical Relaxed Stable Matching (RSM) is **NP-hard**
  - **Ignore** critical nodes
  - Problem is **same** as computing max-size stable matching

- Does a critical relaxed stable matching always exist?
- Can it be efficiently computed?
- The **best-known** approximation ratio for **max-size stable** matching is $\frac{3}{2}$
  - Can we achieve $\frac{3}{2}$-approximation for max-size critical RSM?

Relaxed Stability

[Krishnaa et al., 2020]

Our goal

- Compute a matching that is **critical** and **relaxed stable**

Observation

- Computing max-size critical Relaxed Stable Matching (RSM) is **NP-hard**
  - **Ignore** critical nodes
  - Problem is **same** as computing max-size stable matching

- Does a critical relaxed stable matching always exist? ✓
- Can it be efficiently computed? ✓
- The **best-known** approximation ratio for **max-size stable** matching is $\frac{3}{2}$
  - Can we achieve $\frac{3}{2}$-approximation for max-size critical RSM? ✓

Background

Gale-Shapley algorithm [Gale and Shapley, 1962]

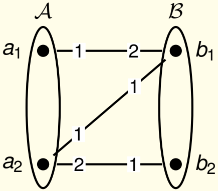
- No critical nodes and no ties
- Well-known **linear-time** algorithm for stable matching

Gale-Shapley algorithm [Gale and Shapley, 1962]

- No critical nodes and no ties
- Well-known **linear-time** algorithm for stable matching
- Vertices in $\mathcal{A}$ propose and vertices in $\mathcal{B}$ accept/reject

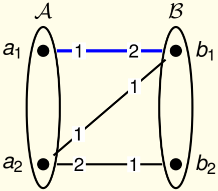
Gale-Shapley algorithm [Gale and Shapley, 1962]

- No critical nodes and no ties
- Well-known **linear-time** algorithm for stable matching
- Vertices in $\mathcal{A}$ propose and vertices in $\mathcal{B}$ accept/reject



Gale-Shapley algorithm [Gale and Shapley, 1962]

- No critical nodes and no ties
- Well-known **linear-time** algorithm for stable matching
- Vertices in $\mathcal{A}$ propose and vertices in $\mathcal{B}$ accept/reject



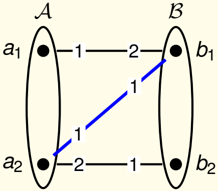
$a_1 \rightarrow b_1$ ✓

b_1 was **unmatched** and hence **accepts** a_2's proposal

Gale-Shapley algorithm

[Gale and Shapley, 1962]

- No critical nodes and no ties
- Well-known **linear-time** algorithm for stable matching
- Vertices in $\mathcal{A}$ propose and vertices in $\mathcal{B}$ accept/reject



$a_1 \rightarrow b_1$ X

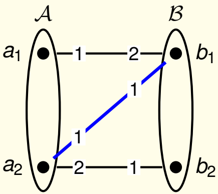
$a_2 \rightarrow b_1$ ✓

b_1 **prefers** a_2 over a_1

Gale-Shapley algorithm

[Gale and Shapley, 1962]

- No critical nodes and no ties
- Well-known **linear-time** algorithm for stable matching
- Vertices in $\mathcal{A}$ propose and vertices in $\mathcal{B}$ accept/reject



$a_1 \rightarrow b_1$ X

$a_2 \rightarrow b_1$ ✓

All $a \in \mathcal{A}$ are either **matched** or **exhausted** their preference list

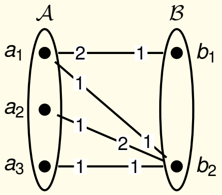
Gale-Shapley algorithm

[Gale and Shapley, 1962]

- No critical nodes and no ties
- Well-known **linear-time** algorithm for stable matching
- Vertices in $\mathcal{A}$ propose and vertices in $\mathcal{B}$ accept/reject
- Algorithm outputs a stable matching M

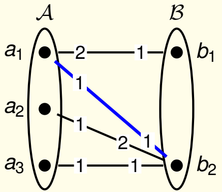
Ties in Preference Lists

- Tied lists on the receiving ($\mathcal{B}$) side and strict list on the proposing ($\mathcal{A}$) side



Király's algorithm: Version I [Király, 2011]

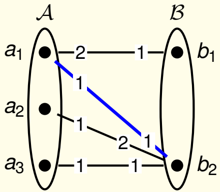
- Tied lists on the receiving ($\mathcal{B}$) side and strict list on the proposing ($\mathcal{A}$) side
- Execute Gale-Shapley algorithm



$a_1 \rightarrow b_2$ ✓

b_2 was **unmatched** and hence **accepts** a_1's proposal

- Tied lists on the receiving ($\mathcal{B}$) side and strict list on the proposing ($\mathcal{A}$) side
- Execute Gale-Shapley algorithm



$a_1 \rightarrow b_2$ ✓

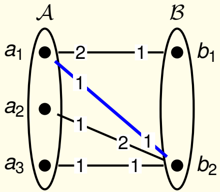
$a_2 \rightarrow b_2$ X

b_2 **rejects** a_2's proposal as a_2 is **not better** than a_1 for b_2

Király's algorithm: Version I

[Király, 2011]

- Tied lists on the receiving ($\mathcal{B}$) side and strict list on the proposing ($\mathcal{A}$) side
- Execute Gale-Shapley algorithm



$a_1 \rightarrow b_2$ ✓

$a_2 \rightarrow b_2$ X

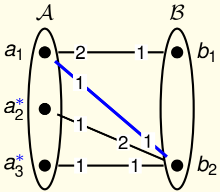
$a_3 \rightarrow b_2$ X

b_2 **rejects** a_3's proposal as a_3 is **not better** than a_1 for b_2

Király's algorithm: Version I

[Király, 2011]

- Tied lists on the receiving ($\mathcal{B}$) side and strict list on the proposing ($\mathcal{A}$) side
- Execute Gale-Shapley algorithm
- Unmatched vertices on $\mathcal{A}$-side gets one more chance to propose with a '$*$' status



$a_1 \rightarrow b_2$ ✓

$a_2 \rightarrow b_2$ X

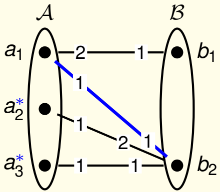
$a_3 \rightarrow b_2$ X

a_2 and a_3 get '$*$' **status** as they exhausted their preference lists

Király's algorithm: Version I

[Király, 2011]

- Tied lists on the receiving ($\mathcal{B}$) side and strict list on the proposing ($\mathcal{A}$) side
- Execute Gale-Shapley algorithm
- Unmatched vertices on $\mathcal{A}$-side gets one more chance to propose with a '$*$' status
- $*$ status of a improves its rank by ϵ in all its neighbours' preference lists ($1 > \epsilon > 0$)



$a_1 \rightarrow b_2$ ✓

$a_2 \rightarrow b_2$ X

$a_3 \rightarrow b_2$ X

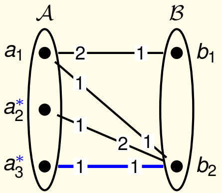
$a_2^* \rightarrow b_2$ X

b_2 **rejects** a_2^*'s proposal as a_2^* is **not better** than a_1 for b_2

Király's algorithm: Version I

[Király, 2011]

- Tied lists on the receiving ($\mathcal{B}$) side and strict list on the proposing ($\mathcal{A}$) side
- Execute Gale-Shapley algorithm
- Unmatched vertices on $\mathcal{A}$-side gets one more chance to propose with a '$*$' status
- $*$ status of a improves its rank by ϵ in all its neighbours' preference lists ($1 > \epsilon > 0$)



$a_1 \rightarrow b_2$ X

$a_2 \rightarrow b_2$ X

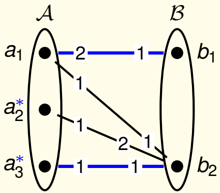
$a_3 \rightarrow b_2$ X

$a_2^* \rightarrow b_2$ X

$a_3^* \rightarrow b_2$ ✓

b_2 **accepts** a_3^*'s proposal and rejects a_1 as a_3^* is better than a_1 for b_2

- Tied lists on the receiving ($\mathcal{B}$) side and strict list on the proposing ($\mathcal{A}$) side
- Execute Gale-Shapley algorithm
- Unmatched vertices on $\mathcal{A}$-side gets one more chance to propose with a '$*$' status
- $*$ status of a improves its rank by ϵ in all its neighbours' preference lists ($1 > \epsilon > 0$)



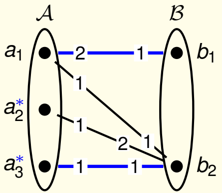
$a_1 \rightarrow b_2$ X
$a_2 \rightarrow b_2$ X
$a_3 \rightarrow b_2$ X
$a_2^* \rightarrow b_2$ X
$a_3^* \rightarrow b_2$ ✓
$a_1 \rightarrow b_1$ ✓

b_1 was **unmatched** and hence **accepts** a_1's proposal

Király's algorithm: Version I

[Király, 2011]

- Tied lists on the receiving ($\mathcal{B}$) side and strict list on the proposing ($\mathcal{A}$) side
- Execute Gale-Shapley algorithm
- Unmatched vertices on $\mathcal{A}$-side gets one more chance to propose with a '$*$' status
- $*$ status of a improves its rank by ϵ in all its neighbours' preference lists ($1 > \epsilon > 0$)
- $\frac{3}{2}$-approximation algorithm of the max-size stable matching



$a_1 \rightarrow b_2$ X

$a_2 \rightarrow b_2$ X

$a_3 \rightarrow b_2$ X

$a_2^* \rightarrow b_2$ X

$a_3^* \rightarrow b_2$ ✓

$a_1 \rightarrow b_1$ ✓

b_1 was **unmatched** and hence **accepts** a_1's proposal

Király's algorithm: Version II

- Tied lists on the proposing ($\mathcal{A}$) side and strict lists on the receiving ($\mathcal{B}$) side

Király's algorithm: Version II

- Tied lists on the proposing ($\mathcal{A}$) side and strict lists on the receiving ($\mathcal{B}$) side
  - **Uncertain proposal** (a, b)**:** b is k^{th}-ranked nbr of a, $\exists b' \neq b$ at rank k, and b' is unmatched

Király's algorithm: Version II

- Tied lists on the proposing ($\mathcal{A}$) side and strict lists on the receiving ($\mathcal{B}$) side
  - **Uncertain proposal** (a, b)**:** b is k^{th}-ranked nbr of a, $\exists b' \neq b$ at rank k, and b' is unmatched
  - The uncertain proposal (a, b) remains uncertain until b rejects a
  - b **rejects uncertain** a as soon as it gets **any proposal**

Király's algorithm: Version II

- Tied lists on the proposing ($\mathcal{A}$) side and strict lists on the receiving ($\mathcal{B}$) side
  - **Uncertain proposal** (a, b)**:** b is k^{th}-ranked nbr of a, $\exists b' \neq b$ at rank k, and b' is unmatched
  - The uncertain proposal (a, b) remains uncertain until b rejects a
  - b **rejects uncertain** a as soon as it gets **any proposal**
  - When b **rejects** an uncertain a then a **marks** b to propose “once again in future"

Király's algorithm: Version II

- Tied lists on the proposing ($\mathcal{A}$) side and strict lists on the receiving ($\mathcal{B}$) side
  - **Uncertain proposal** (a, b)**:** b is k^{th}-ranked nbr of a, $\exists b' \neq b$ at rank k, and b' is unmatched
  - The uncertain proposal (a, b) remains uncertain until b rejects a
  - b **rejects uncertain** a as soon as it gets **any proposal**
  - When b **rejects** an uncertain a then a **marks** b to propose “once again in future"
  - **Favourite nbr** b **of** a**:** $k =$ min rank for a at which marked or unmatched nbrs exist

Király's algorithm: Version II

- Tied lists on the proposing ($\mathcal{A}$) side and strict lists on the receiving ($\mathcal{B}$) side
  - **Uncertain proposal** (a, b)**:** b is k^{th}-ranked nbr of a, $\exists b' \neq b$ at rank k, and b' is unmatched
  - The uncertain proposal (a, b) remains uncertain until b rejects a
  - b **rejects uncertain** a as soon as it gets **any proposal**
  - When b **rejects** an uncertain a then a **marks** b to propose “once again in future"
  - **Favourite nbr b of a:** $k =$ min rank for a at which marked or unmatched nbrs exist
    - (i) b is an **unmatched** vertex at rank k, or

Király's algorithm: Version II

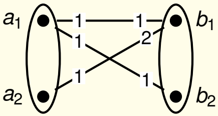
- Tied lists on the proposing ($\mathcal{A}$) side and strict lists on the receiving ($\mathcal{B}$) side
  - **Uncertain proposal** (a, b)**:** b is k^{th}-ranked nbr of a, $\exists b' \neq b$ at rank k, and b' is unmatched
  - The uncertain proposal (a, b) remains uncertain until b rejects a
  - b **rejects uncertain** a as soon as it gets **any proposal**
  - When b **rejects** an uncertain a then a **marks** b to propose "once again in future"
  - **Favourite nbr** b **of** a**:** $k =$ min rank for a at which marked or unmatched nbrs exist
    - (i) b is an **unmatched** vertex at rank k, or
    - (ii) if all vertices at rank k are matched, then b is **unproposed** vertex by a or

Király's algorithm: Version II

- Tied lists on the proposing ($\mathcal{A}$) side and strict lists on the receiving ($\mathcal{B}$) side
  - **Uncertain proposal** (a, b)**:** b is k^{th}-ranked nbr of a, $\exists b' \neq b$ at rank k, and b' is unmatched
  - The uncertain proposal (a, b) remains uncertain until b rejects a
  - b **rejects uncertain** a as soon as it gets **any proposal**
  - When b **rejects** an uncertain a then a **marks** b to propose “once again in future"
  - **Favourite nbr** b **of** a**:** $k =$ min rank for a at which marked or unmatched nbrs exist
    (i) b is an **unmatched** vertex at rank k, or
    (ii) if all vertices at rank k are matched, then b is **unproposed** vertex by a or
    (iii) if all vertices at rank k are already proposed by a, then b is a **marked** vertex by a

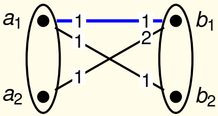
Király's algorithm: Version II

- Tied lists on the proposing ($\mathcal{A}$) side and strict lists on the receiving ($\mathcal{B}$) side
  - **Uncertain proposal** (a, b)**:** b is k^{th}-ranked nbr of a, $\exists b' \neq b$ at rank k, and b' is unmatched
  - The uncertain proposal (a, b) remains uncertain until b rejects a
  - b **rejects uncertain** a as soon as it gets **any proposal**
  - When b **rejects** an uncertain a then a **marks** b to propose "once again in future"
  - **Favourite nbr** b **of** a**:** $k =$ min rank for a at which marked or unmatched nbrs exist
    - (i) b is an **unmatched** vertex at rank k, or
    - (ii) if all vertices at rank k are matched, then b is **unproposed** vertex by a or
    - (iii) if all vertices at rank k are already proposed by a, then b is a **marked** vertex by a



Király's algorithm: Version II

- Tied lists on the proposing ($\mathcal{A}$) side and strict lists on the receiving ($\mathcal{B}$) side
  - **Uncertain proposal** (a, b)**:** b is k^{th}-ranked nbr of a, $\exists b' \neq b$ at rank k, and b' is unmatched
  - The uncertain proposal (a, b) remains uncertain until b rejects a
  - b **rejects uncertain** a as soon as it gets **any proposal**
  - When b **rejects** an uncertain a then a **marks** b to propose “once again in future"
  - **Favourite nbr** b **of** a**:** $k =$ min rank for a at which marked or unmatched nbrs exist
    1. (i) b is an **unmatched** vertex at rank k, or
    2. (ii) if all vertices at rank k are matched, then b is **unproposed** vertex by a or
    3. (iii) if all vertices at rank k are already proposed by a, then b is a **marked** vertex by a

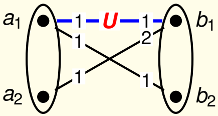


$a_1 \rightarrow b_1$ ✓

b_1 was **unmatched** and hence **accepts** a_1's proposal

Király's algorithm: Version II

- Tied lists on the proposing ($\mathcal{A}$) side and strict lists on the receiving ($\mathcal{B}$) side
  - **Uncertain proposal** (a, b)**:** b is k^{th}-ranked nbr of a, $\exists b' \neq b$ at rank k, and b' is unmatched
  - The uncertain proposal (a, b) remains uncertain until b rejects a
  - b **rejects uncertain** a as soon as it gets **any proposal**
  - When b **rejects** an uncertain a then a **marks** b to propose “once again in future"
  - **Favourite nbr** b **of** a**:** $k =$ min rank for a at which marked or unmatched nbrs exist
    - (i) b is an **unmatched** vertex at rank k, or
    - (ii) if all vertices at rank k are matched, then b is **unproposed** vertex by a or
    - (iii) if all vertices at rank k are already proposed by a, then b is a **marked** vertex by a



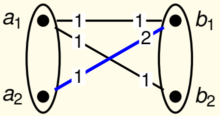
$a_1 \rightarrow b_1$ ✓

(a_1, b_1) is **uncertain**

Király's algorithm: Version II

- Tied lists on the proposing ($\mathcal{A}$) side and strict lists on the receiving ($\mathcal{B}$) side
  - **Uncertain proposal** (a, b)**:** b is k^{th}-ranked nbr of a, $\exists b' \neq b$ at rank k, and b' is unmatched
  - The uncertain proposal (a, b) remains uncertain until b rejects a
  - b **rejects uncertain** a as soon as it gets **any proposal**
  - When b **rejects** an uncertain a then a **marks** b to propose "once again in future"
  - **Favourite nbr** b **of** a**:** $k =$ min rank for a at which marked or unmatched nbrs exist
    - (i) b is an **unmatched** vertex at rank k, or
    - (ii) if all vertices at rank k are matched, then b is **unproposed** vertex by a or
    - (iii) if all vertices at rank k are already proposed by a, then b is a **marked** vertex by a

a_1 marks b_1



$a_1 \rightarrow b_1$ X

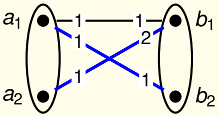
$a_2 \rightarrow b_1$ ✓

b_1 **rejects** a_1 and accepts a_2's proposal as (a_1, b_1) was **uncertain**

Király's algorithm: Version II

- Tied lists on the proposing ($\mathcal{A}$) side and strict lists on the receiving ($\mathcal{B}$) side
  - **Uncertain proposal (a, b):** b is k^{th}-ranked nbr of a, $\exists b' \neq b$ at rank k, and b' is unmatched
  - The uncertain proposal (a, b) remains uncertain until b rejects a
  - b **rejects uncertain** a as soon as it gets **any proposal**
  - When b **rejects** an uncertain a then a **marks** b to propose "once again in future"
  - **Favourite nbr b of a:** $k =$ min rank for a at which marked or unmatched nbrs exist
    - (i) b is an **unmatched** vertex at rank k, or
    - (ii) if all vertices at rank k are matched, then b is **unproposed** vertex by a or
    - (iii) if all vertices at rank k are already proposed by a, then b is a **marked** vertex by a

a_1 marks b_1



$a_1 \rightarrow b_1$ X

$a_2 \rightarrow b_1$ ✓

$a_1 \rightarrow b_2$ ✓

b_2 was **unmatched** and hence **accepts** a_1's proposal

Király's algorithm [Király, 2013]

- Tied lists on both sides

Király's algorithm [Király, 2013]

- Tied lists on both sides
  - **Combine** the two ideas: **Version I** and **Version II**

Király's algorithm [Király, 2013]

- Tied lists on both sides
  - **Combine** the two ideas: **Version I** and **Version II**
  - Run the algorithm for tied lists on the proposing ($\mathcal{A}$) side

Király's algorithm

[Király, 2013]

- Tied lists on both sides
  - **Combine** the two ideas: **Version I** and **Version II**
  - Run the algorithm for tied lists on the proposing ($\mathcal{A}$) side
  - If any $a \in \mathcal{A}$ remained unmatched then it gets a $*$ status

Király's algorithm

[Király, 2013]

- Tied lists on both sides
  - **Combine** the two ideas: **Version I** and **Version II**
  - Run the algorithm for tied lists on the proposing ($\mathcal{A}$) side
  - If any $a \in \mathcal{A}$ remained unmatched then it gets a $*$ status
  - Stop when each $a \in \mathcal{A}$ is either matched or exhausted its pref list as a $*$ status vertex

Király's algorithm

[Király, 2013]

- Tied lists on both sides
  - **Combine** the two ideas: **Version I** and **Version II**
  - Run the algorithm for tied lists on the proposing ($\mathcal{A}$) side
  - If any $a \in \mathcal{A}$ remained unmatched then it gets a $*$ status
  - Stop when each $a \in \mathcal{A}$ is either matched or exhausted its pref list as a $*$ status vertex
  - Output matching M is a **stable** matching

Király's algorithm

[Király, 2013]

- Tied lists on both sides
  - **Combine** the two ideas: **Version I** and **Version II**
  - Run the algorithm for tied lists on the proposing ($\mathcal{A}$) side
  - If any $a \in \mathcal{A}$ remained unmatched then it gets a $*$ status
  - Stop when each $a \in \mathcal{A}$ is either matched or exhausted its pref list as a $*$ status vertex
  - Output matching M is a **stable** matching

- Suppose $\exists$ a blocking pair (a, b) w.r.t. the output matching M

Király's algorithm

[Király, 2013]

- Tied lists on both sides
  - **Combine** the two ideas: **Version I** and **Version II**
  - Run the algorithm for tied lists on the proposing ($\mathcal{A}$) side
  - If any $a \in \mathcal{A}$ remained unmatched then it gets a $*$ status
  - Stop when each $a \in \mathcal{A}$ is either matched or exhausted its pref list as a $*$ status vertex
  - Output matching M is a **stable** matching

- Suppose $\exists$ a blocking pair (a, b) w.r.t. the output matching M
- b must be matched

Király's algorithm [Király, 2013]

- Tied lists on both sides
  - **Combine** the two ideas: **Version I** and **Version II**
  - Run the algorithm for tied lists on the proposing ($\mathcal{A}$) side
  - If any $a \in \mathcal{A}$ remained unmatched then it gets a $*$ status
  - Stop when each $a \in \mathcal{A}$ is either matched or exhausted its pref list as a $*$ status vertex
  - Output matching M is a **stable** matching

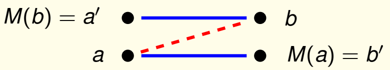
- Suppose $\exists$ a blocking pair (a, b) w.r.t. the output matching M
- b must be matched
- a must be matched: If not, a must have proposed to b at least twice
  - $b \in \mathcal{B}$ cannot be in an uncertain proposal after receiving its second proposal
  - $M(b)$ is not worse than a – contradiction

Király's algorithm

[Király, 2013]

- Tied lists on both sides
  - **Combine** the two ideas: **Version I** and **Version II**
  - Run the algorithm for tied lists on the proposing ($\mathcal{A}$) side
  - If any $a \in \mathcal{A}$ remained unmatched then it gets a $*$ status
  - Stop when each $a \in \mathcal{A}$ is either matched or exhausted its pref list as a $*$ status vertex
  - Output matching M is a **stable** matching

- Suppose $\exists$ a blocking pair (a, b) w.r.t. the output matching M
- b must be matched
- a must be matched: If not, a must have proposed to b at least twice
  - $b \in \mathcal{B}$ cannot be in an uncertain proposal after receiving its second proposal
  - $M(b)$ is not worse than a – contradiction
- a must have proposed to b and b rejected it

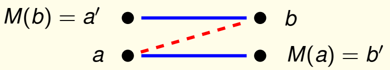


Király's algorithm

[Király, 2013]

- Tied lists on both sides
  - **Combine** the two ideas: **Version I** and **Version II**
  - Run the algorithm for tied lists on the proposing ($\mathcal{A}$) side
  - If any $a \in \mathcal{A}$ remained unmatched then it gets a $*$ status
  - Stop when each $a \in \mathcal{A}$ is either matched or exhausted its pref list as a $*$ status vertex
  - Output matching M is a **stable** matching

- Suppose $\exists$ a blocking pair (a, b) w.r.t. the output matching M
- b must be matched
- a must be matched: If not, a must have proposed to b at least twice
  - $b \in \mathcal{B}$ cannot be in an uncertain proposal after receiving its second proposal
  - $M(b)$ is not worse than a – contradiction
- a must have proposed to b and b rejected it
  - (a, b) was not uncertain: $a' = M(b)$ is not worse than a for b

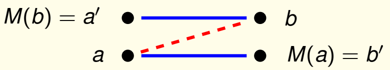


Király's algorithm

[Király, 2013]

- Tied lists on both sides
  - **Combine** the two ideas: **Version I** and **Version II**
  - Run the algorithm for tied lists on the proposing ($\mathcal{A}$) side
  - If any $a \in \mathcal{A}$ remained unmatched then it gets a $*$ status
  - Stop when each $a \in \mathcal{A}$ is either matched or exhausted its pref list as a $*$ status vertex
  - Output matching M is a **stable** matching

- Suppose $\exists$ a blocking pair (a, b) w.r.t. the output matching M
- b must be matched
- a must be matched: If not, a must have proposed to b at least twice
  - $b \in \mathcal{B}$ cannot be in an uncertain proposal after receiving its second proposal
  - $M(b)$ is not worse than a – contradiction
- a must have proposed to b and b rejected it
  - (a, b) was not uncertain: $a' = M(b)$ is not worse than a for b
  - (a, b) was uncertain: a proposed b again before proposing to $M(a)$



Király's algorithm

[Király, 2013]

- Tied lists on both sides
  - **Combine** the two ideas: **Version I** and **Version II**
  - Run the algorithm for tied lists on the proposing ($\mathcal{A}$) side
  - If any $a \in \mathcal{A}$ remained unmatched then it gets a $*$ status
  - Stop when each $a \in \mathcal{A}$ is either matched or exhausted its pref list as a $*$ status vertex
  - Output matching M is a **stable** matching
  - $\frac{3}{2}$**-approximation** of maximum size stable matching

Király's algorithm

[Király, 2013]

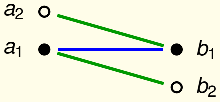
$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

- Suppose such 3-length augmenting path exists



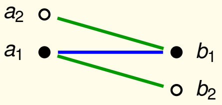
Király's algorithm [Király, 2013]

$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

- Suppose such 3-length augmenting path exists

b_2 is unmatched in M



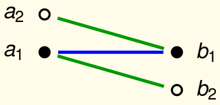
Király's algorithm

[Király, 2013]

$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

- Suppose such 3-length augmenting path exists



b_2 is unmatched in M

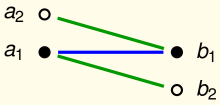
No proposal sent to b_2

a_1 without $*$ status

$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

- Suppose such 3-length augmenting path exists
- **Claim 1:** a_1 prefers b_1 over b_2



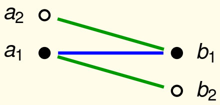
$\boxed{\text{No proposal sent to } b_2}$

$\boxed{a_1 \text{ without } * \text{ status}}$

$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

- Suppose such 3-length augmenting path exists
- **Claim 1:** a_1 prefers b_1 over b_2
  - Suppose not then b_1 and b_2 are tied



No proposal sent to b_2

a_1 without $*$ status

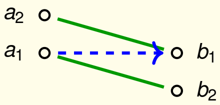
$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

- Suppose such 3-length augmenting path exists
- **Claim 1:** a_1 prefers b_1 over b_2
  - Suppose not then b_1 and b_2 are tied

No proposal sent to b_2

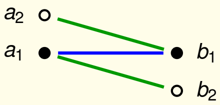
a_1 without $*$ status



$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

- Suppose such 3-length augmenting path exists
- **Claim 1:** a_1 prefers b_1 over b_2
  - Suppose not then b_1 and b_2 are tied



No proposal sent to b_2

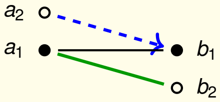
a_1 without $*$ status

(a_1, b_1) is uncertain

$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

- Suppose such 3-length augmenting path exists
- **Claim 1:** a_1 prefers b_1 over b_2
  - Suppose not then b_1 and b_2 are tied
  - b_1 received other proposals



No proposal sent to b_2

a_1 without $*$ status

(a_1, b_1) is uncertain

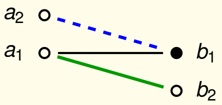
Király's algorithm

[Király, 2013]

$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

- Suppose such 3-length augmenting path exists
- **Claim 1:** a_1 prefers b_1 over b_2
  - Suppose not then b_1 and b_2 are tied
  - b_1 received other proposals



No proposal sent to b_2

a_1 without $*$ status

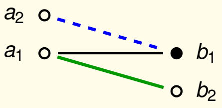
(a_1, b_1) is uncertain

b_1 rejected a_1

$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

- Suppose such 3-length augmenting path exists
- **Claim 1:** a_1 prefers b_1 over b_2
  - Suppose not then b_1 and b_2 are tied
  - b_1 received other proposals
  - a_1 must propose b_2 before proposing to b_1



No proposal sent to b_2

a_1 without $*$ status

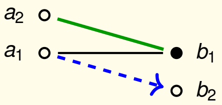
(a_1, b_1) is uncertain

b_1 rejected a_1

$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

- Suppose such 3-length augmenting path exists
- **Claim 1:** a_1 prefers b_1 over b_2
  - Suppose not then b_1 and b_2 are tied
  - b_1 received other proposals
  - a_1 must propose b_2 before proposing to b_1



No proposal sent to b_2

a_1 without $*$ status

(a_1, b_1) is uncertain

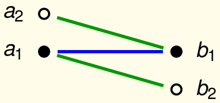
b_1 rejected a_1

a_1 proposed b_2

$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

- Suppose such 3-length augmenting path exists
- **Claim 1:** a_1 prefers b_1 over b_2
- **Claim 2**: b_1 prefers a_1 over a_2



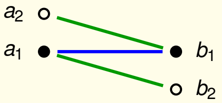
Király's algorithm

[Király, 2013]

$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

- Suppose such 3-length augmenting path exists
- **Claim 1:** a_1 prefers b_1 over b_2
- **Claim 2**: b_1 prefers a_1 over a_2
  - **Observation:** $b \in \mathcal{B}$ is not part of an uncertain proposal after receiving its second proposal



Király's algorithm

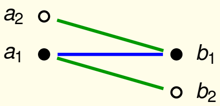
[Király, 2013]

$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

- Suppose such 3-length augmenting path exists
- **Claim 1:** a_1 prefers b_1 over b_2
- **Claim 2**: b_1 prefers a_1 over a_2
  - **Observation:** $b \in \mathcal{B}$ is not part of an uncertain proposal after receiving its second proposal

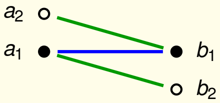
$\boxed{b_1 \text{ received} \geq 3 \text{ proposals}}$



$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

- Suppose such 3-length augmenting path exists
- **Claim 1:** a_1 prefers b_1 over b_2
- **Claim 2**: b_1 prefers a_1 over a_2
  - **Observation:** $b \in \mathcal{B}$ is not part of an uncertain proposal after receiving its second proposal



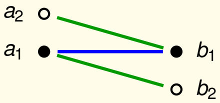
b_1 received ≥ 3 proposals

a_1 did not get $*$ status

$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

- Suppose such 3-length augmenting path exists
- **Claim 1:** a_1 prefers b_1 over b_2
- **Claim 2**: b_1 prefers a_1 over a_2
  - **Observation:** $b \in \mathcal{B}$ is not part of an uncertain proposal after receiving its second proposal



$\boxed{b_1 \text{ received} \geq 3 \text{ proposals}}$

$\boxed{a_1 \text{ did not get } * \text{ status}}$

$\boxed{b_1 \text{ rejected } a_2^* \text{ but not } a_1}$

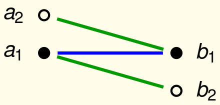
Király's algorithm

[Király, 2013]

$\frac{3}{2}$-approximation of maximum size stable matching M^*

Proof idea: No 1 or 3-length aug-path w.r.t. M in $(M \oplus M^*)$

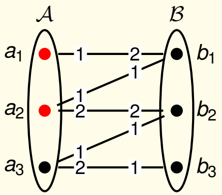
- Suppose such 3-length augmenting path exists
- **Claim 1:** a_1 prefers b_1 over b_2
- **Claim 2**: b_1 prefers a_1 over a_2
- M^* is not stable – a contradiction



Critical Nodes

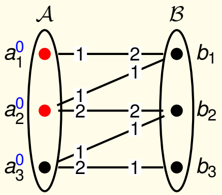
Feasible RSM: Multi-level Gale-Shapley algorithm

- **Assumptions:** (i) Strict lists and (ii) $C \subseteq \mathcal{A}$



Feasible RSM: Multi-level Gale-Shapley algorithm

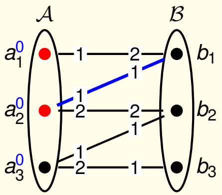
- **Assumptions:** (i) Strict lists and (ii) $C \subseteq \mathcal{A}$
- All $a \in \mathcal{A}$ are at level 0 to begin with



All $a \in \mathcal{A}$ are **initially** at level **0**

Feasible RSM: Multi-level Gale-Shapley algorithm

- **Assumptions:** (i) Strict lists and (ii) $C \subseteq \mathcal{A}$
- All $a \in \mathcal{A}$ are at level 0 to begin with
- Execute Gale-Shapley algorithm

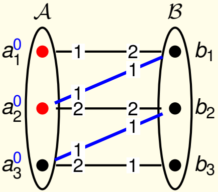


$a_2^0 \rightarrow b_1$ ✓

b_1 accepts a_2^0's proposal as it was unmatched

Feasible RSM: Multi-level Gale-Shapley algorithm

- **Assumptions:** (i) Strict lists and (ii) $C \subseteq \mathcal{A}$
- All $a \in \mathcal{A}$ are at level 0 to begin with
- Execute Gale-Shapley algorithm



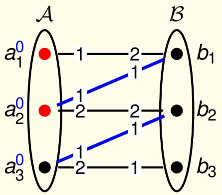
$a_2^0 \rightarrow b_1$ ✓

$a_3^0 \rightarrow b_2$ ✓

b_2 accepts a_3^0's proposal as it was unmatched

Feasible RSM: Multi-level Gale-Shapley algorithm

- **Assumptions:** (i) Strict lists and (ii) $C \subseteq \mathcal{A}$
- All $a \in \mathcal{A}$ are at level 0 to begin with
- Execute Gale-Shapley algorithm



$a_2^0 \rightarrow b_1$ ✓

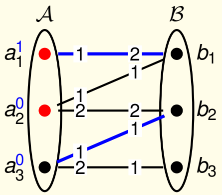
$a_3^0 \rightarrow b_2$ ✓

$a_1^0 \rightarrow b_1$ X

b_1 rejects a_1^0's proposal as $M(b_1) = a_2^0$ is better preferred

Feasible RSM: Multi-level Gale-Shapley algorithm

- **Assumptions:** (i) Strict lists and (ii) $C \subseteq \mathcal{A}$
- All $a \in \mathcal{A}$ are at level 0 to begin with
- Execute Gale-Shapley algorithm
- Unmatched critical vertices raise their level and propose again



$a_2^0 \rightarrow b_1$ X

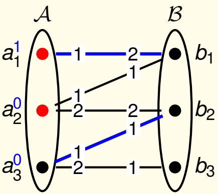
$a_3^0 \rightarrow b_2$ ✓

$a_1^1 \rightarrow b_1$ ✓

b_1 prefers a_1^1 as it is at a higher level

Feasible RSM: Multi-level Gale-Shapley algorithm

- **Assumptions:** (i) Strict lists and (ii) $C \subseteq \mathcal{A}$
- All $a \in \mathcal{A}$ are at level 0 to begin with
- Execute Gale-Shapley algorithm
- Unmatched critical vertices raise their level and propose again
- Vertices at higher level are preferred more by $b \in \mathcal{B}$ than those at lower level



$a_2^0 \rightarrow b_1$ X

$a_3^0 \rightarrow b_2$ ✓

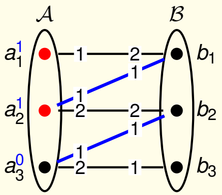
$a_1^1 \rightarrow b_1$ ✓

$a_2^0 \rightarrow b_2$ X

b_2 rejects a_2^0's proposal as $M(b_2) = a_3^0$ is better preferred

Feasible RSM: Multi-level Gale-Shapley algorithm

- **Assumptions:** (i) Strict lists and (ii) $C \subseteq \mathcal{A}$
- All $a \in \mathcal{A}$ are at level 0 to begin with
- Execute Gale-Shapley algorithm
- Unmatched critical vertices raise their level and propose again
- Vertices at higher level are preferred more by $b \in \mathcal{B}$ than those at lower level



$a_2^0 \rightarrow b_1$ X

$a_3^0 \rightarrow b_2$ ✓

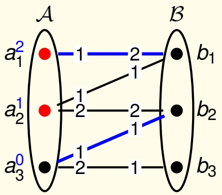
$a_1^1 \rightarrow b_1$ X

$a_2^1 \rightarrow b_1$ ✓

b_1 prefers a_2^1 better than a_1^1

Feasible RSM: Multi-level Gale-Shapley algorithm

- **Assumptions:** (i) Strict lists and (ii) $C \subseteq \mathcal{A}$
- All $a \in \mathcal{A}$ are at level 0 to begin with
- Execute Gale-Shapley algorithm
- Unmatched critical vertices raise their level and propose again
- Vertices at higher level are preferred more by $b \in \mathcal{B}$ than those at lower level



$a_2^0 \rightarrow b_1$ X

$a_3^0 \rightarrow b_2$ ✓

$a_1^1 \rightarrow b_1$ X

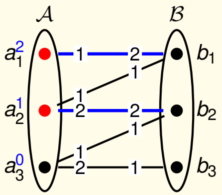
$a_2^1 \rightarrow b_1$ X

$a_1^2 \rightarrow b_1$ ✓

b_1 prefers a_1^2 as it is at a higher level

Feasible RSM: Multi-level Gale-Shapley algorithm

- **Assumptions:** (i) Strict lists and (ii) $C \subseteq \mathcal{A}$
- All $a \in \mathcal{A}$ are at level 0 to begin with
- Execute Gale-Shapley algorithm
- Unmatched critical vertices raise their level and propose again
- Vertices at higher level are preferred more by $b \in \mathcal{B}$ than those at lower level



$a_2^0 \rightarrow b_1$ X

$a_3^0 \rightarrow b_2$ X

$a_1^1 \rightarrow b_1$ X

$a_2^1 \rightarrow b_1$ X

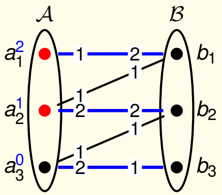
$a_1^2 \rightarrow b_1$ ✓

$a_2^1 \rightarrow b_2$ ✓

b_2 prefers a higher level neighbour a_2^1

Feasible RSM: Multi-level Gale-Shapley algorithm

- **Assumptions:** (i) Strict lists and (ii) $C \subseteq \mathcal{A}$
- All $a \in \mathcal{A}$ are at level 0 to begin with
- Execute Gale-Shapley algorithm
- Unmatched critical vertices raise their level and propose again
- Vertices at higher level are preferred more by $b \in \mathcal{B}$ than those at lower level



$a_2^0 \rightarrow b_1$ X

$a_3^0 \rightarrow b_2$ X

$a_1^1 \rightarrow b_1$ X

$a_2^1 \rightarrow b_1$ X

$a_1^2 \rightarrow b_1$ ✓

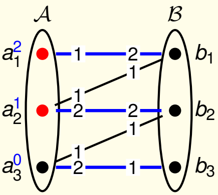
$a_2^1 \rightarrow b_2$ ✓

$a_3^0 \rightarrow b_3$ ✓

b_3 accepts a_3^0's proposal as it was unmatched

Feasible RSM: Multi-level Gale-Shapley algorithm

- **Assumptions:** (i) Strict lists and (ii) $C \subseteq \mathcal{A}$
- All $a \in \mathcal{A}$ are at level 0 to begin with
- Execute Gale-Shapley algorithm
- Unmatched critical vertices raise their level and propose again
- Vertices at higher level are preferred more by $b \in \mathcal{B}$ than those at lower level



$a_2^0 \rightarrow b_1$ X

$a_3^0 \rightarrow b_2$ X

$a_1^1 \rightarrow b_1$ X

$a_2^1 \rightarrow b_1$ X

$a_1^2 \rightarrow b_1$ ✓

$a_2^1 \rightarrow b_2$ ✓

$a_3^0 \rightarrow b_3$ ✓

The highest level achieved $= 2 = |C|$

Feasible RSM: Multi-level Gale-Shapley algorithm

- **Assumptions:** (i) Strict lists and (ii) $C \subseteq \mathcal{A}$
- All $a \in \mathcal{A}$ are at level 0 to begin with
- Execute Gale-Shapley algorithm
- Unmatched critical vertices raise their **level up to** $|C|$ and propose again
- Vertices at higher level are preferred more by $b \in \mathcal{B}$ than those at lower level

Correctness

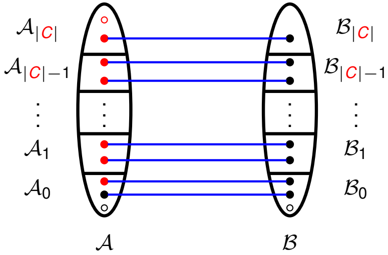
Assuming G admits a feasible matching:
Claim 1: Output matching M is feasible.
Claim 2: Output matching M is relaxed stable.

Properties of the output matching

- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm

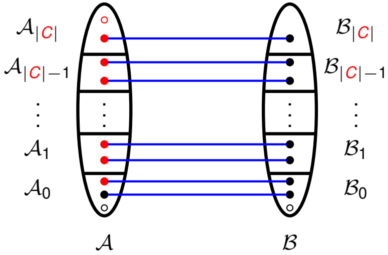
Properties of the output matching

- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm
- Partition the vertices based on levels to give a level structure for G



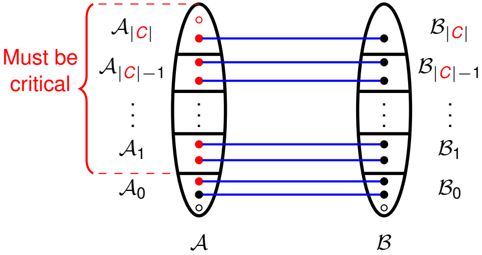
Properties of the output matching

- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm
- Partition the vertices based on levels to give a level structure for G
- All the matched edges are horizontal



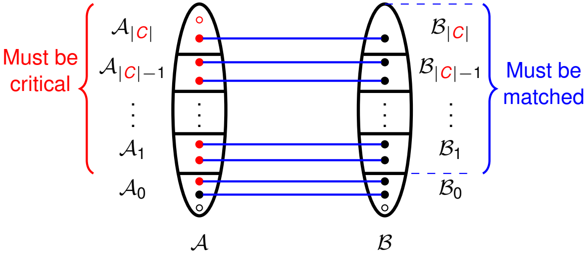
Properties of the output matching

- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm
- Partition the vertices based on levels to give a level structure for G
- All the matched edges are horizontal



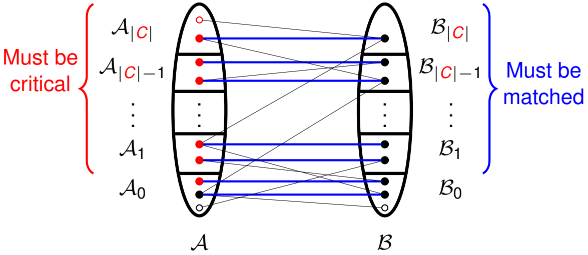
Properties of the output matching

- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm
- Partition the vertices based on levels to give a level structure for G
- All the matched edges are horizontal



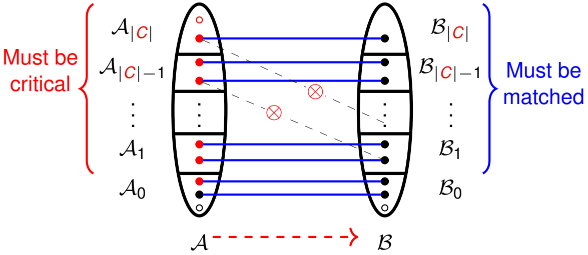
Properties of the output matching

- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm
- Partition the vertices based on levels to give a level structure for G
- All the matched edges are horizontal



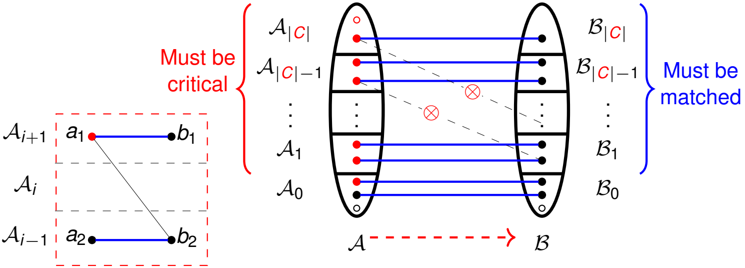
Properties of the output matching

- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm
- Partition the vertices based on levels to give a level structure for G
- All the matched edges are horizontal
- No steep downward (at least two levels down) edges



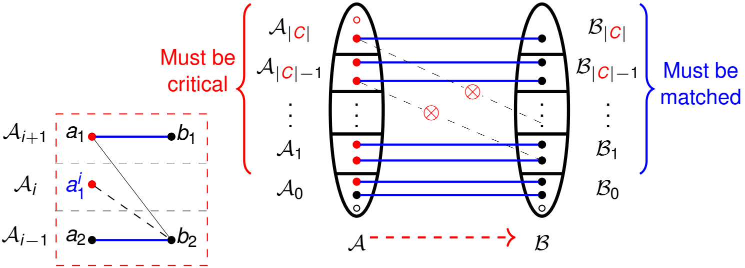
Properties of the output matching

- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm
- Partition the vertices based on levels to give a level structure for G
- All the matched edges are horizontal
- No steep downward (at least two levels down) edges



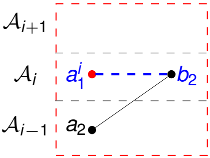
Properties of the output matching

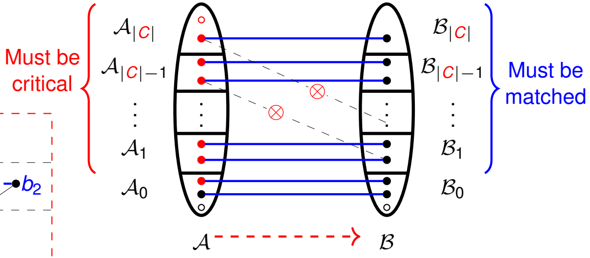
- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm
- Partition the vertices based on levels to give a level structure for G
- All the matched edges are horizontal
- No steep downward (at least two levels down) edges



Properties of the output matching

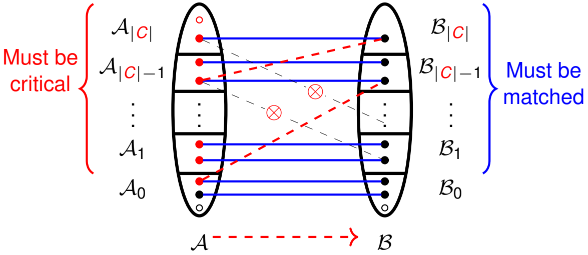
- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm
- Partition the vertices based on levels to give a level structure for G
- All the matched edges are horizontal
- No steep downward (at least two levels down) edges





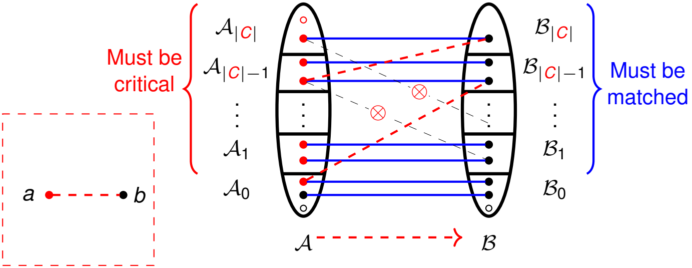
Properties of the output matching

- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm
- Partition the vertices based on levels to give a level structure for G
- All the matched edges are horizontal
- No steep downward (at least two levels down) edges
- All blocking edges are upwards (maybe steep)



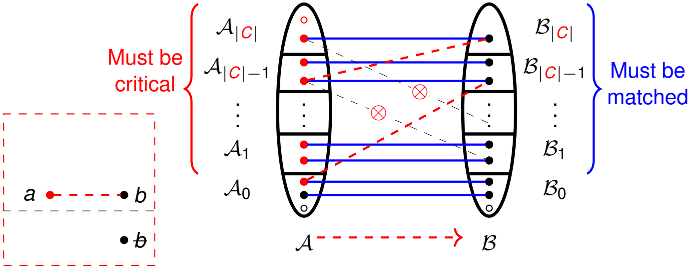
Properties of the output matching

- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm
- Partition the vertices based on levels to give a level structure for G
- All the matched edges are horizontal
- No steep downward (at least two levels down) edges
- All blocking edges are upwards (maybe steep)



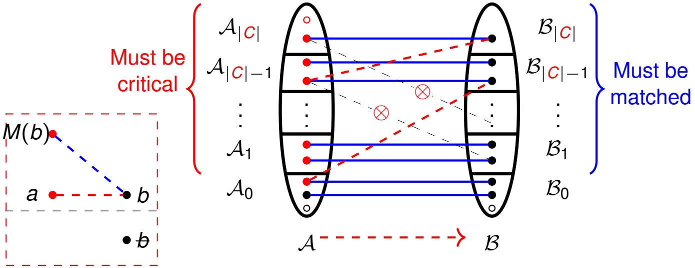
Properties of the output matching

- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm
- Partition the vertices based on levels to give a level structure for G
- All the matched edges are horizontal
- No steep downward (at least two levels down) edges
- All blocking edges are upwards (maybe steep)



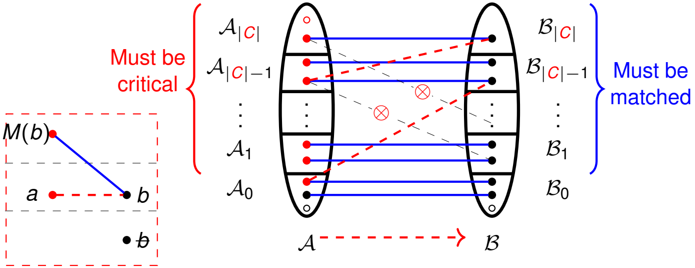
Properties of the output matching

- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm
- Partition the vertices based on levels to give a level structure for G
- All the matched edges are horizontal
- No steep downward (at least two levels down) edges
- All blocking edges are upwards (maybe steep)



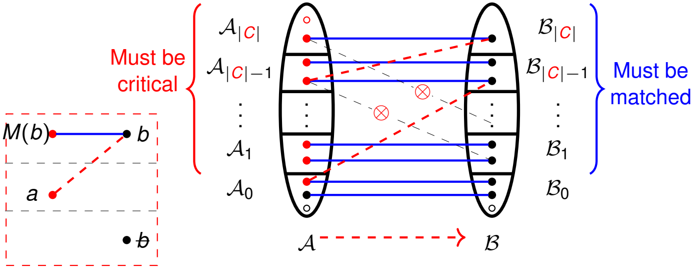
Properties of the output matching

- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm
- Partition the vertices based on levels to give a level structure for G
- All the matched edges are horizontal
- No steep downward (at least two levels down) edges
- All blocking edges are upwards (maybe steep)



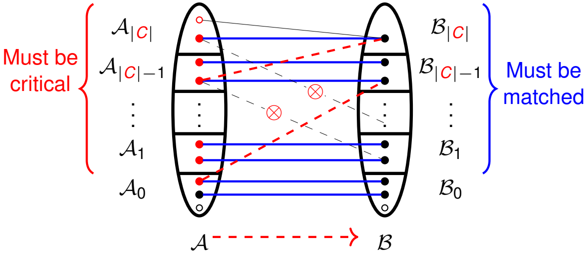
Properties of the output matching

- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm
- Partition the vertices based on levels to give a level structure for G
- All the matched edges are horizontal
- No steep downward (at least two levels down) edges
- All blocking edges are upwards (maybe steep)



Properties of the output matching

- Each $a \in \mathcal{A}$ is assigned a level at the end of the algorithm
- Partition the vertices based on levels to give a level structure for G
- All the matched edges are horizontal
- No steep downward (at least two levels down) edges
- All blocking edges are upwards (maybe steep)
- All neighbours of unmatched critical a are in $\mathcal{B}_{|C|}$



Correctness

Output matching is feasible

Proof Sketch:

- By contradiction

Correctness

Output matching is feasible

Proof Sketch:

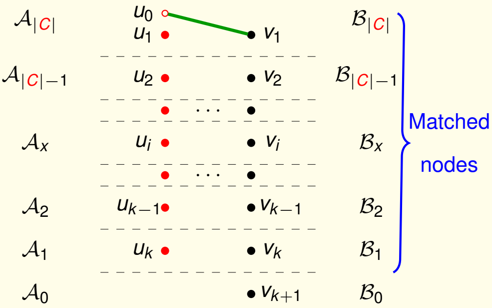
- By contradiction
  - Suppose M is not feasible. By assumption, a feasible matching N exists.

Correctness

Output matching is feasible

Proof Sketch:

- By contradiction
  - Suppose M is not feasible. By assumption, a feasible matching N exists.



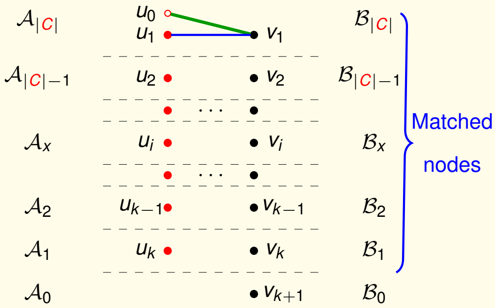
All neighbours of u_0 are in $\mathcal{B}_{|C|}$

Correctness

Output matching is feasible

Proof Sketch:

- By contradiction
  - Suppose M is not feasible. By assumption, a feasible matching N exists.



All neighbours of u_0 are in $\mathcal{B}_{|C|}$

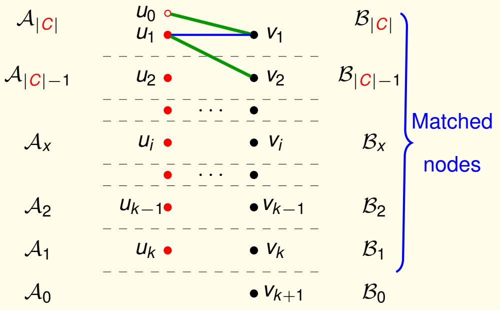
All matched edges in M are horizontal

Correctness

Output matching is feasible

Proof Sketch:

- By contradiction
  - Suppose M is not feasible. By assumption, a feasible matching N exists.



All neighbours of u_0 are in $\mathcal{B}_{|C|}$

All matched edges in M are horizontal

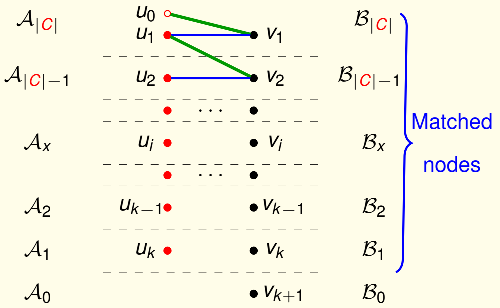
No steep downward edge

Correctness

Output matching is feasible

Proof Sketch:

- By contradiction
  - Suppose M is not feasible. By assumption, a feasible matching N exists.



All neighbours of u_0 are in $\mathcal{B}_{|C|}$

All matched edges in M are horizontal

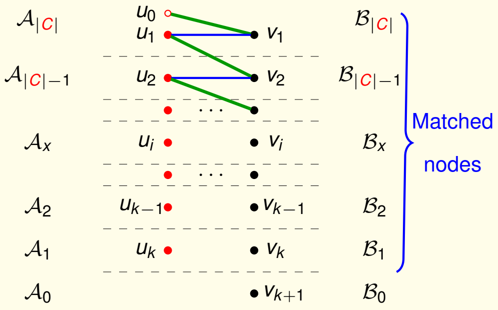
No steep downward edge

Correctness

Output matching is feasible

Proof Sketch:

- By contradiction
  - Suppose M is not feasible. By assumption, a feasible matching N exists.



All neighbours of u_0 are in $\mathcal{B}_{|C|}$

All matched edges in M are horizontal

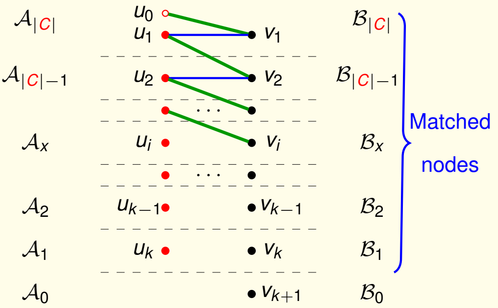
No steep downward edge

Correctness

Output matching is feasible

Proof Sketch:

- By contradiction
  - Suppose M is not feasible. By assumption, a feasible matching N exists.



All neighbours of u_0 are in $\mathcal{B}_{|C|}$

All matched edges in M are horizontal

No steep downward edge

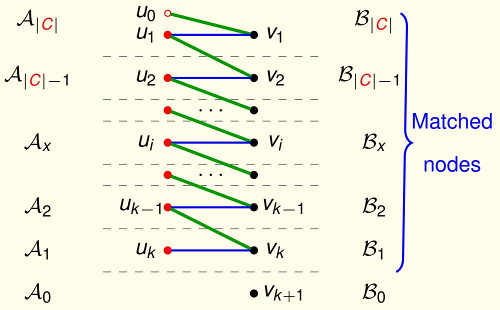
Path must end in $\mathcal{A}_0 \cup \mathcal{B}_0$

Correctness

Output matching is feasible

Proof Sketch:

- By contradiction
  - Suppose M is not feasible. By assumption, a feasible matching N exists.



All neighbours of u_0 are in $\mathcal{B}_{|C|}$

All matched edges in M are horizontal

No steep downward edge

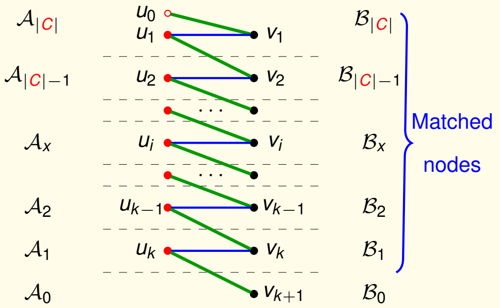
Path must end in $\mathcal{A}_0 \cup \mathcal{B}_0$

Correctness

Output matching is feasible

Proof Sketch:

- By contradiction
  - Suppose M is not feasible. By assumption, a feasible matching N exists.



All neighbours of u_0 are in $\mathcal{B}_{|C|}$

All matched edges in M are horizontal

No steep downward edge

Path must end in $\mathcal{A}_0 \cup \mathcal{B}_0$

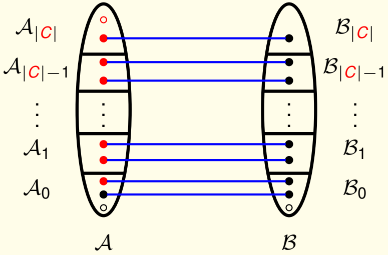
$\#$critical vertices $> |C|$

Correctness

Output matching M of our algorithm is Relaxed Stable Matching

Correctness

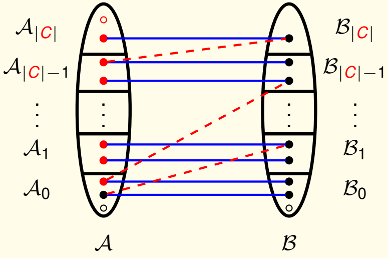
Output matching M of our algorithm is Relaxed Stable Matching



Correctness

Output matching M of our algorithm is Relaxed Stable Matching

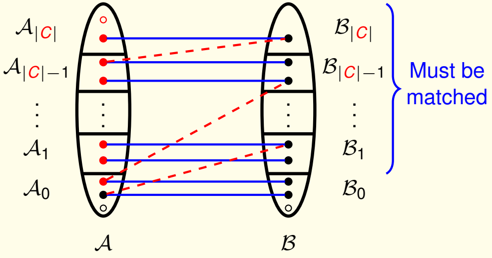
All blocking edges w.r.t. M are upwards



Correctness

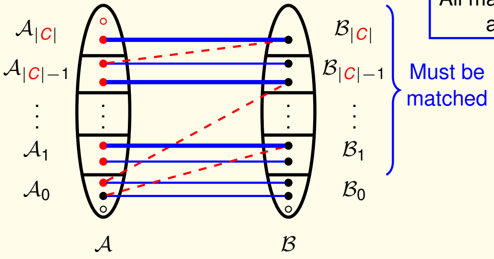
Output matching M of our algorithm is Relaxed Stable Matching

All blocking edges w.r.t. M are upwards



Correctness

Output matching M of our algorithm is Relaxed Stable Matching

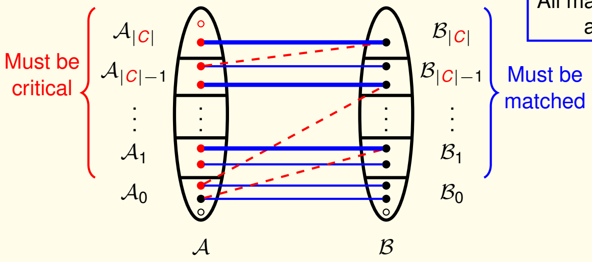


All blocking edges w.r.t. M are upwards

All matched edges in M are horizontal

Correctness

Output matching M of our algorithm is Relaxed Stable Matching

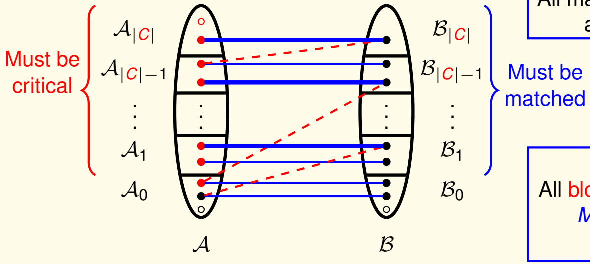


All blocking edges w.r.t. M are upwards

All matched edges in M are horizontal

Correctness

Output matching M of our algorithm is Relaxed Stable Matching



All blocking edges w.r.t. M are upwards

All matched edges in M are horizontal

All blocking edges w.r.t. M are **justified**

Algorithm's Outline

An Evolving Perspective

Summary of our algorithm

- **Assumptions:** (i) Strict lists and (ii) No critical nodes

Gale-Shapley Level

No critical nodes

All vertices in $\mathcal{A}$ propose to **all** neighbours in $\mathcal{B}$

Summary of our algorithm

- **Assumptions:** (i) Strict lists and (ii) $C \subseteq \mathcal{A}$

Higher level (Level $1, 2, \ldots, |C|$)

Critical nodes on $\mathcal{A}$-side

Critical vertices in $\mathcal{A}$ propose to **all** neighbours in $\mathcal{B}$

Gale-Shapley level (Level 0)

No critical nodes

All vertices in $\mathcal{A}$ propose to **all** neighbours in $\mathcal{B}$

Summary of our algorithm

- **Assumptions:** (i) **Tied lists** and (ii) $C \subseteq \mathcal{A}$

Higher level (Level $1, 2, \ldots, |C|$)

Critical nodes on $\mathcal{A}$-side

Critical vertices in $\mathcal{A}$ propose to **all** neighbours in $\mathcal{B}$

Király's algorithm (Level 0 and 0^*)

All vertices in $\mathcal{A}$ propose to all neighbours in $\mathcal{B}$

Summary of our algorithm

- **Assumptions:** (i) Tied lists, (ii) $C \subseteq \mathcal{A} \cup \mathcal{B}$ and (iii) $|\mathcal{A} \cap C| = s$ and $|\mathcal{B} \cap C| = t$

Higher level (Level $t, \ldots, s + t$)

Critical nodes on $\mathcal{A}$-side

Critical vertices in $\mathcal{A}$ propose to **all** neighbours in $\mathcal{B}$

Király's algorithm (Level t and t^*)

All vertices in $\mathcal{A}$ propose to all neighbours in $\mathcal{B}$

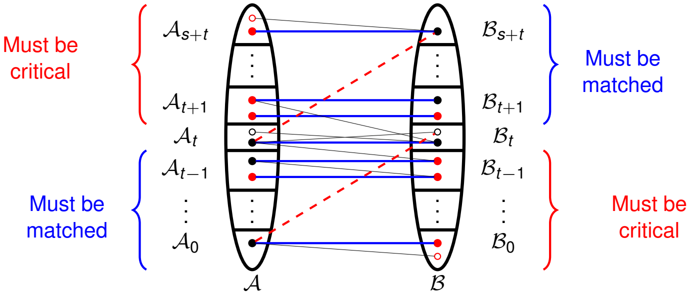
Lower level (Level $0, 1, \ldots, t - 1$)

Critical nodes on $\mathcal{B}$-side

All vertices in $\mathcal{A}$ propose to **critical** neighbours in $\mathcal{B}$

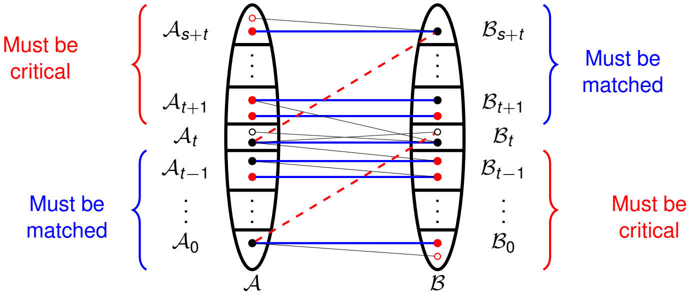
Summary of our algorithm

- M is critical



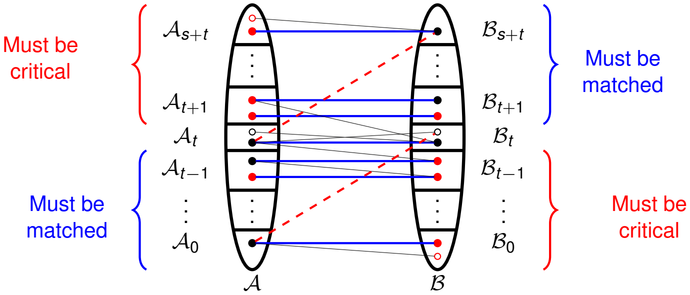
Correctness

- M is critical
- M is Relaxed Stable Matching (RSM)



Correctness

- M is critical
- M is Relaxed Stable Matching (RSM)
- $|M| \geq \frac{3}{2} \cdot |M^*|$ for any Max-size Critical Relaxed Stable Matching M^*



Conclusion

- For an instance $G = (\mathcal{A} \cup \mathcal{B}, E, C)$ with ties on both sides and $C \subseteq \mathcal{A} \cup \mathcal{B}$

Conclusion

- For an instance $G = (\mathcal{A} \cup \mathcal{B}, E, C)$ with ties on both sides and $C \subseteq \mathcal{A} \cup \mathcal{B}$
- Critical Relaxed Stable Matching (RSM) **always** exists

Conclusion

- For an instance $G = (\mathcal{A} \cup \mathcal{B}, E, C)$ with ties on both sides and $C \subseteq \mathcal{A} \cup \mathcal{B}$
- Critical Relaxed Stable Matching (RSM) **always** exists
- Computing maximum size critical RSM is **NP-Hard**

Conclusion

- For an instance $G = (\mathcal{A} \cup \mathcal{B}, E, C)$ with ties on both sides and $C \subseteq \mathcal{A} \cup \mathcal{B}$
- Critical Relaxed Stable Matching (RSM) **always** exists
- Computing maximum size critical RSM is **NP-Hard**
- $\frac{3}{2}$**-approximation** of the max-size critical RSM

Conclusion

- For an instance $G = (\mathcal{A} \cup \mathcal{B}, E, C)$ with ties on both sides and $C \subseteq \mathcal{A} \cup \mathcal{B}$
- Critical Relaxed Stable Matching (RSM) **always** exists
- Computing maximum size critical RSM is **NP-Hard**
- $\frac{3}{2}$**-approximation** of the max-size critical RSM
- Natural extension is to the many-to-many setting

Conclusion

- For an instance $G = (\mathcal{A} \cup \mathcal{B}, E, C)$ with ties on both sides and $C \subseteq \mathcal{A} \cup \mathcal{B}$
- Critical Relaxed Stable Matching (RSM) **always** exists
- Computing maximum size critical RSM is **NP-Hard**
- $\frac{3}{2}$**-approximation** of the max-size critical RSM
- Natural extension is to the many-to-many setting

Thank You!

keshav@cse.iitm.ac.in

Reference I

Gale, D. and Shapley, L. S. (1962).
College admissions and the stability of marriage.
The American Mathematical Monthly, 69(1):9–15.

Goko, H., Makino, K., Miyazaki, S., and Yokoi, Y. (2022).
Maximally satisfying lower quotas in the hospitals/residents problem with ties.
In 39th International Symposium on Theoretical Aspects of Computer Science.

Király, Z. (2011).
Better and simpler approximation algorithms for the stable marriage problem.
Algorithmica, 60(1):3–20.

Király, Z. (2013).
Linear time local approximation algorithm for maximum stable marriage.
Algorithms, 6(3):471–484.

Krishnaa, P., Limaye, G., Nasre, M., and Nimbhorkar, P. (2020).
Envy-freeness and relaxed stability: Hardness and approximation algorithms.
In International Symposium on Algorithmic Game Theory, pages 193–208.
Springer.

Reference II

Makino, K., Miyazaki, S., and Yokoi, Y. (2022).
Incomplete list setting of the hospitals/residents problem with maximally satisfying lower quotas.
In Kanellopoulos, P., Kyropoulou, M., and Voudouris, A. A., editors, Algorithmic Game Theory - 15th International Symposium, SAGT 2022, Colchester, UK, September 12-15, 2022, Proceedings, volume 13584 of Lecture Notes in Computer Science, pages 544–561. Springer.

Manlove, D. F., Irving, R. W., Iwama, K., Miyazaki, S., and Morita, Y. (2002).
Hard variants of stable marriage.
Theoretical Computer Science, 276(1-2):261–279.