CS 331, Fall 2026

Lecture 10 (9/28)

Today: - Stable matching

Stable matching (Part IV, Section 5)

Setup: n applicants $\{$Alice, Bob, ...$\}$

n job openings $\{$Google, Apple, ...$\}$

Input: Preference lists

$a: \alpha \succ \gamma \succ \beta$ $\quad$ $\alpha: b \succ a \succ c$

$b: \gamma \succ \alpha \succ \beta$ $\quad$ $\beta: c \succ a \succ b$

$c: \alpha \succ \beta \succ \gamma$ $\quad$ $\gamma: a \succ b \succ c$

Output: Stable matching

(a, α)
(b, β)
(c, γ)

unstable

(b, α)
(c, β)
(a, γ)

stable

What's the difference?

(b, α) unstable pair:

b prefers $\alpha > \beta$

α prefers $b > a$

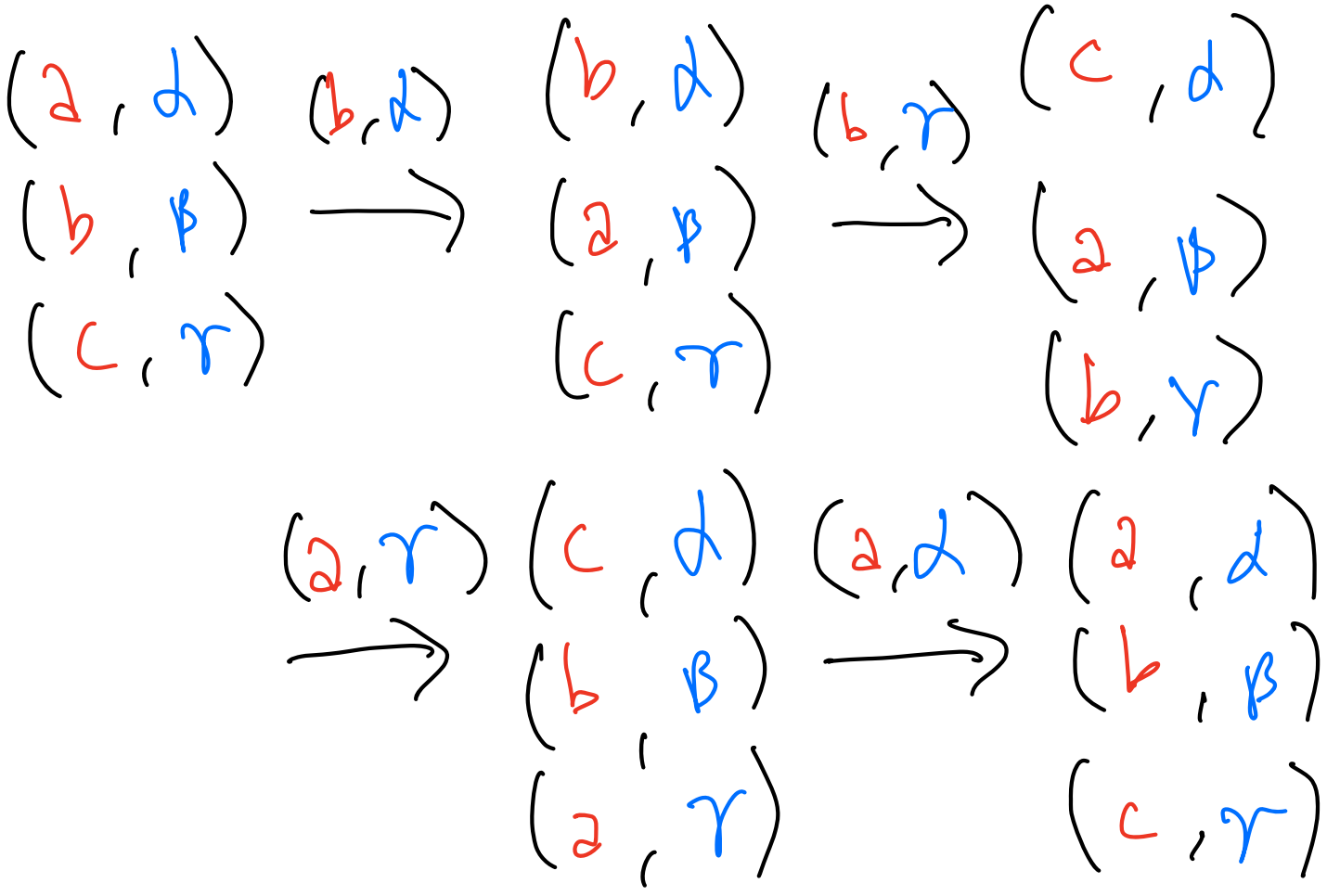
backroom deal...

Stability notion protects against "first-order" deviations, similar concept to Nash equilibrium

How to design a greedy algo?

Idea: fix instability (like inversions)

Issue: cycles



Just because (b, α) unstable doesn't mean we should pair (a, β)!

Key idea: job offers/reneges (unmatch a matched pair)

- Maintain pool of temporary matches
- If (a, d) in the pool, a prefers B to d, and B makes offer to a, can renege and pool $\leftarrow$ pool $\cup (a, B) \setminus (a, d)$

Gale-Shapley 1950

- Hugely influential in practice
  - National Resident Matching Program
  - Faculty recruiting
  - Public schools in NYC, Boston
  - Assignments in US Navy
  - Kidney exchange programs
- Nobel Prize in Economics, 2012

Stable Matching $\left(\{A_a\}_{a \in [n]}, \{J_\alpha\}_{\alpha \in [n]}\right)$:

$M \leftarrow \emptyset$, $i_\alpha \leftarrow 1 \quad \forall \alpha \in [n]$

While $\exists$ unmatched job α:

$a \leftarrow J_\alpha[i_\alpha]$ // favorite applicant who reject

If a unmatched: $M \leftarrow M \cup \{(a, \alpha)\}$

Elif a prefers α to β (current match):

$M \leftarrow M \setminus \{(a, \beta)\} \cup \{(a, \alpha)\}$

i_β ++ // reneged

Else: i_α ++ // rejected

Return M

After $O(n^2)$ preprocessing (inverse lookups, etc.)

Can implement each iter in $O(1)$ time:

Maintain M as Array indexed by a

Runtime: Potential method.

Define function Φ that captures algo progress.

Our potential:

$$\Phi = |M| + \sum_d i_d$$

Algo ends when $|M| = n$, so

$$\Phi \geq n + n^2 \Rightarrow \text{termination}$$

Every iter:
- pointer i_d grows
  OR
- $|M|$ grows

} Φ grows!

Total: $O(n^2)$ linear time!

Correctness: Perfect matching

If $|M| \neq n$, the loop continues!

Stable matching

- Let $\{(a, \alpha), (b, \beta)\} \in M$
- Suppose (a, β) unstable
- If a had offer from β then would not be w/ α

(a, α) $\quad$ $(a, \geq \beta)$

our world $\quad$ if β offered to a

- But β likes $a > b$, so must have offered first.

Hence, no unstable pairs.

Structural fact: Outcome always same, regardless of tiebreaking.

Say a feasible for j and j feasible for a } if $(a, j) \in M$ stable

(for some choice of stable M)

Key claims

1: Every job j gets best feasible a

2: Every applicant a gets worst feasible j

Uniqueness of M follows immediately.

(No ties in preference lists)

Proof of claim 1: Consider <u>first time</u> in G-S where best feasible a for α rejects for some other job β

Because a feasible, $\exists$ stable M' pairing (a, α) and (b, β)

- a prefers β to α
- β prefers a to b ← feasible for β... can't have rejected yet when β makes a offer!

Unstable! ⇒⇐

Proof of claim 2: Suppose G-S pairs (a, α) but some other stable M' pairs (a, β) ← worse feasible job for a (b, α)

- a prefers α to β
- α prefers a to b (proof: Claim 1 says so.)

Unstable! ⇒⇐