

CS 331, Fall 2025  
Lecture 10 (9/29)

Today: — Stable  
matching

# Stable matching (Part IV, Section 5)

Setup:  $n$  applicants  $\{Alice, Bob, \dots\}$   
 $n$  job openings  $\{Goose, Apple, \dots\}$

Input: Preference lists

$a: d > r > b$

$b: r > d > b$

$c: d > b > r$

$d: b > a > c$

$b: c > a > b$

$r: a > b > c$

Output: Stable matching

(a, d)

(b, β)

(c, γ)

unstable

(b, d)

(c, β)

(a, γ)

stable

What's the difference?

(b, d) unstable pair:

b prefers d > β

d prefers b > a

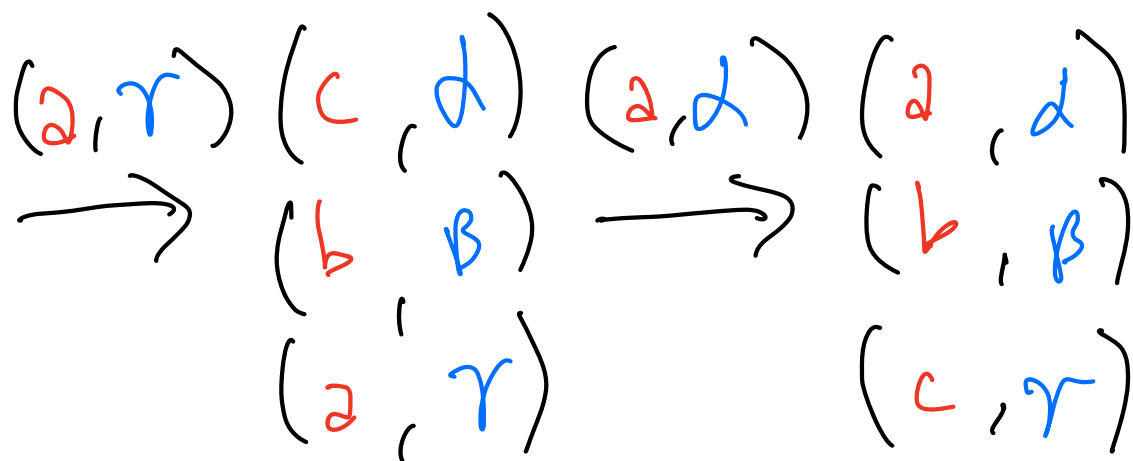
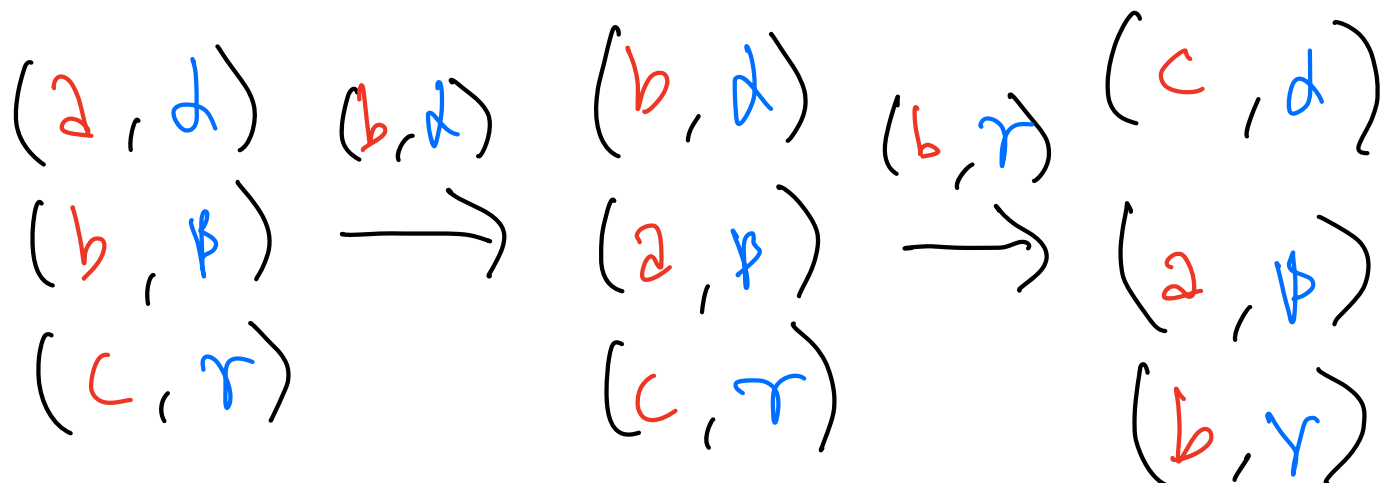
backroom  
deal...

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  $(b, d)$  unstable doesn't mean we should pair  $(a, \beta)$ !

Key idea: job offers/renege (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  $\text{pool} \leftarrow \text{pool} \cup (a, b) \setminus (a, d)$

## Gale-Shapley algo

- 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  $(\{A_a\}_{a \in G}, \{J_d\}_{d \in G})$ :

$M \leftarrow \emptyset, i_d \leftarrow 1 \quad \forall d \in G$

While  $\exists$  unmatched job  $d$ :

$a \leftarrow J_d[i_d]$  // favorite applicant who reject

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

Else if  $a$  prefers  $d$  to  $b$  (current match):

$M \leftarrow M \setminus \{(a, b)\} \cup \{(a, d)\}$

$i_b++$  // rejected

Else:  
 $i_d++$  // rejected

Return  $M$

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

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

Maintain  $M$  as Array indexed by  $a$

Routine: Potential method.

Define function  $\Phi$  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:  $\left. \begin{array}{l} \bullet \text{ pointer } i_d \text{ grows} \\ \text{OR} \\ \bullet |M| \text{ grows} \end{array} \right\} \Phi \text{ grows!}$

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

Correctness: Perfect matchings

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

Stable matching

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

$(a, d)$

our world

$(a, \geq B)$

if  $B$  offered to  $a$

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

Hence, no unstable pairs.

Structural fact: Outcome always same,  
regardless of tiebreaking.

Say  $\alpha$  feasible for  $\beta$  } if  $(\alpha, \beta) \in M$   
 $\beta$  feasible for  $\alpha$  } stable  
(for some choice of stable  $M$ )

Key claims

1: Every job  $\beta$  gets best feasible  $\alpha$

2: Every applicant  $\alpha$  gets worst feasible  $\beta$

Uniqueness of  $M$  follows immediately.

(No ties in preference lists)



Proof of claim 1: Consider first time  
in G-S where best feasible  $a$  for  $d$   
rejects for some other job  $B$

Because  $a$  feasible,  $\exists$  stable  $M'$  pairing  
 $(a, d)$  and  $(b, B)$

- $a$  prefers  $B$  to  $d$
  - $B$  prefers  $a$  to  $b$   $\leftarrow$  feasible for  $B$ ...  
Can't have rejected yet  
when  $B$  makes  $a$  offer!
- Unstable!  $\Rightarrow \Leftarrow$

Proof of claim 2: Suppose G-S pairs  $(a, d)$   
but some other stable  $M'$  pairs  $(a, B)$   $\leftarrow$  more feasible  
job for  $a$   
 $(b, d)$

- $a$  prefers  $d$  to  $B$
- $d$  prefers  $a$  to  $b$  (proof: claim 1 says so.)

Unstable!  $\Rightarrow \Leftarrow$