

CS70 — SPRING 2026

LECTURE 4: JAN. 29

Summary

- Proof techniques

- Direct Proof
 - Proof by Contraposition
 - Proof by Contradiction
 - Proof by Cases
 - Proof by Induction
-

- This lecture : Application to Stable Matching

Stable Matching Problem

Input: n jobs, n candidates

for each job, a ranked list of all candidates
- - - candidate, - - - - - jobs } "preference lists"

Goal: match up jobs & candidates in a "good" way

Motivation for CS70

"Killer app" for proofs!

Example

Databricks : A C B

Ericsson : C A B

FedEx : A C B

Annie : E F D

Bertha : F E D

Caro : F E D

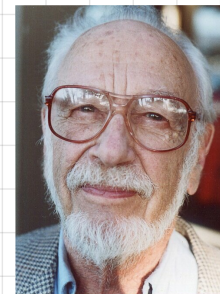
Possible matching : $\{ (\underbrace{D, C}_{\text{mm}}) (\underbrace{E, B}_{\text{mm}}) (F, A) \}$

Defn : A rogue couple in a matching is a job & candidate who prefer each other to their current partners

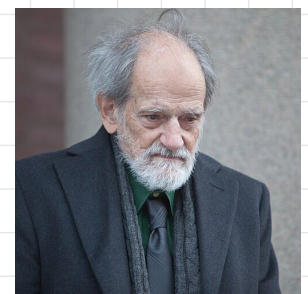
Defn : A matching is stable if it has no rogue couples

National Residency Matching Program (NRMP)

- matches residency applicants with hospitals
- chaotic system until 1950
- early 1950s: switch to centralized algorithm, then to stable matching
- formalized by Gale & Shapley in 1962
- crowning achievement in combinatorial algorithms
- CS70: "killer app" for proofs



David Gale



Lloyd Shapley

Do stable matchings always exist?

Roommates Problem:

Input: $2n$ people, each with a pref. list for all $2n-1$ others

Goal: find a stable matching

Example:

Annie : B C D

Bertha : C A D

Caro : A B D

Daphne : <whatever>

Claim: There is no stable matching for this input!

Proof:

Possible algorithm :

while a rogue couple (J, C) exists
switch the partners of J & C

Problem : may create new rogue couples*

Note : If above algorithm works for stable matching
then it should also work for roommates !

So : Look for an algorithm that exploits asymmetry

Note : Roth/Vande Vate : \exists carefully chosen seq. of switches
that works for stable matching
(but complicated & hard to describe)

The Propose-and-Reject Algorithm

REPEAT

1 "DAY"

1. Each job makes an offer to the highest ranked candidate on its list who hasn't rejected it
2. Each candidate says "maybe" to the job she likes best among her offers, and rejects the others
3. If a job was rejected, it crosses that candidate off its list

puts offer "on hold"

UNTIL no rejections occurred

OUTPUT resulting matching

Q1: Does this algorithm terminate?

Q2: Does it always output a matching?

Q3: Is that matching always stable?

Example

Databricks : A C B

Ericsson : C A B

FedEx : A C B

Annie : E F D

Bertha : F E D

Caro : F E D

	Day1	Day2	Day3
Annie			
Bertha			
Caro			

Another Example

Jobs

1 : A B C D
2 : B D A C
3 : B A D C
4 : A B C D

Candidates

A : 2 1 3 4
B : 1 4 2 3
C : 2 3 1 4
D : 4 1 2 3

Day 1

Day 2

Day 3

Day 4

A
B
C
D

Analysis of P-&-R Algorithm

Theorem : The algorithm always terminates

Proof :

Improvement Lemma

Suppose job J offers to candidate C on day k .
Then on every day $i \geq k$, C has on hold an offer from a job she likes at least as much as J .

Proof: Induction on i

Base case: $i = k$ - true by defn. of algorithm

Ind. step: Assume for some $i \geq k$ & prove for $i+1$

On day i , C has on hold an offer from job $J' \geq J$

On day $i+1$, J' again offers to C (along, possibly, with others)

So after day $i+1$ C has offer from $J'' \geq J' \geq J$



Theorem : The algorithm always outputs a matching

Proof :

Theorem: The output matching is always stable

Proof:

Example

Databricks : A C B

Ericsson : C A B

FedEx : A C B

Annie : E F D

Bertha : F E D

Caro : F E D

Matching output by alg :
 $\{(A, F), (B, D), (C, E)\}$

Only other stable matching:

$\{(A, E), (B, D), (C, F)\}$

What's special about the output matching ?

Definition : For a job J , the optimal candidate for J is the best candidate J has in any stable matching

Definition : The job-optimal matching is the one in which each job is matched with its optimal candidate

ALERT : Is this even a valid matching ?

(How do we know that two jobs don't have the same optimal candidate ?)

NOTE : In previous example, output matching is job optimal.

Theorem: The matching output by the algorithm is job-optimal

Proof: Enough to prove $\forall k \geq 0 \ P(k)$, where

$P(k)$: On day k , no job gets rejected by its opt. candidate

Proof by induction on k

Base case: $k=0$ ✓ (nothing to prove)

Inductive step: Sp no such rejections through day k

Day $k+1$: Sp. for \exists that job J gets rejected by its opt. candidate C^* in favor of some job J^*

Since C^* is opt. for J , \exists stable matching

$$\mathcal{M} = \{ \dots (J, C^*) \dots (J^*, C') \dots \}$$

Claim: J^*, C^* are a rogue couple

Inductive step: Sp no such rejections through day k

Day $k+1$: Sp. for \ast that job J gets rejected by its opt.

candidate C^* in favor of some job J^*

Since C^* is opt. for J , \exists stable matching

$$\mathcal{M} = \{ \dots (J, C^*) \dots (J^*, C') \dots \}$$

Claim: J^*, C^* are a rogue couple

Proof of Claim: Certainly C^* prefers J^* to J

What about J^* ?

By ind. hypothesis, J^* has not yet been rejected by its optimal candidate

Hence, since J^* is offering to C^* now, J^* must like C^* at least as much as its opt. candidate, hence at least as much as C (partner in stable \mathcal{M})

Thus J^*, C^* are a rogue couple \square

Theorem: The job-optimal matching is candidate-pessimal

Proof: By contradiction

Let $M = \{ \dots (J, C) \dots \}$ be job-optimal

Sp. for \times that M is not candidate-pessimal

Then \exists stable matching

$$M' = \{ \dots (J, C') \dots (J^*, C) \dots \}$$

s.t. C prefers J to J^*

Claim: J, C is a rogue couple in M'

Proof of Claim: C prefers J to J^* by assⁿ.

Also, J prefers C to C' since M is job-optimal

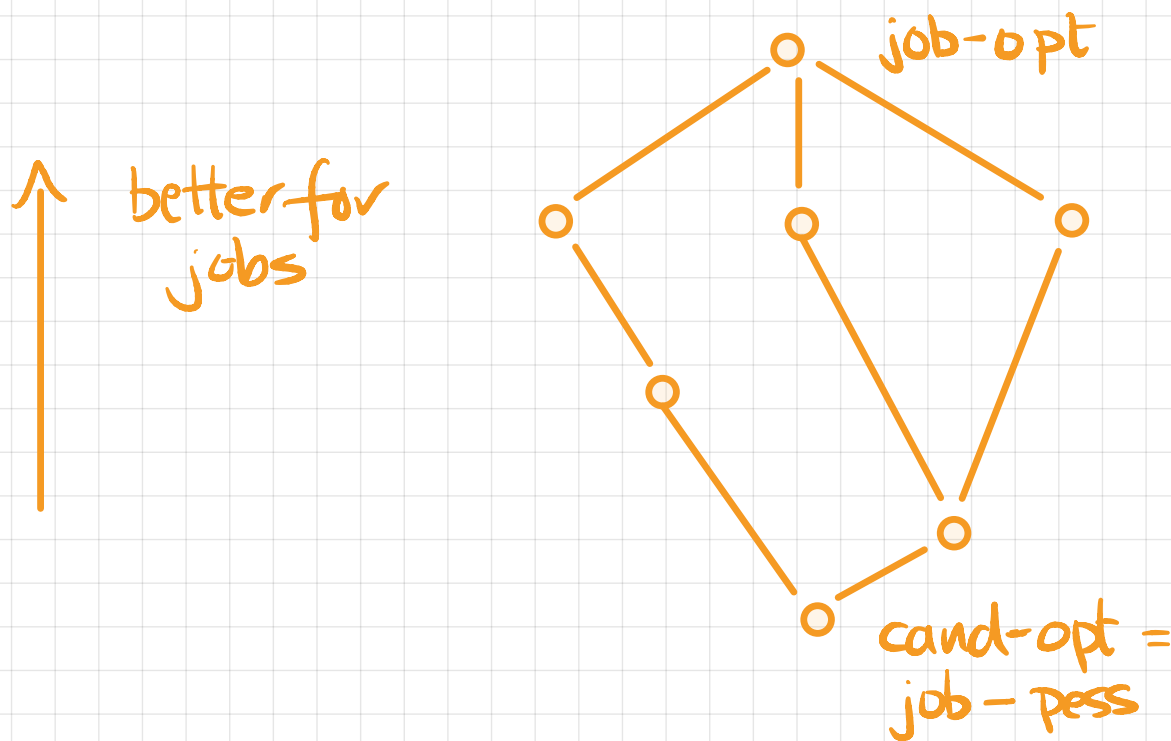
Hence M' is not stable \times



Q: How many stable matchings can there be?

A: Between 1 and exponentially many (in n)

Fact: Stable matchings form a lattice (partial order)



$M \leq M'$ iff
every job does at
least as well in
 M' as in M

Another Example

Jobs

1 :	A	C	D	B
2 :	B	A	C	D
3 :	C	D	B	A
4 :	D	B	A	C

Candidates

A :	3	4	2	1
B :	1	3	4	2
C :	4	2	1	3
D :	2	1	3	4

Job-optimal: $\{(1, A), (2, B), (3, C), (4, D)\}$

Cand.-optimal: $\{(1, B), (2, D), (3, A), (4, C)\}$

Also stable : $\{(1, C), (2, A), (3, D), (4, B)\}$

Ex:

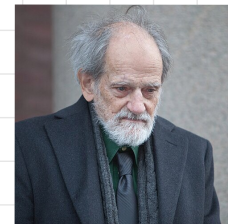
Check this is stable

Find another stable matching

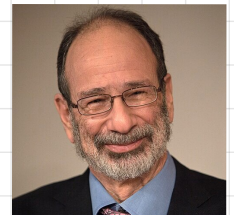
How do these four matchings fit into the lattice?

N.R.M.P. Revisited

- Originally (1950s) hospital-optimal
- switched to candidate-optimal in 1990s
- various bells-and-whistles, e.g.
 - allows incomplete pref. lists
 - allows couples to apply jointly
- used in many other contexts, e.g.
 - organ donor matching
 - school districts
 - internet content delivery
- Nobel Prize in Economics 2012 (Shapley-Roth)



Lloyd Shapley



Alvin Roth

Summary

- Stable matching: an important real-world problem
 - Not obvious that a stable matching always exists !
 - We proved that a stable matching exists by designing & analyzing a clever algorithm
 - Also: notions of optimality among stable matchings
-

- Next week : Graphs !