CS 70 Discrete Mathematics and Probability Theory
Spring 2024 Seshia, Sinclair

DIS 1B

1 Stable Matching

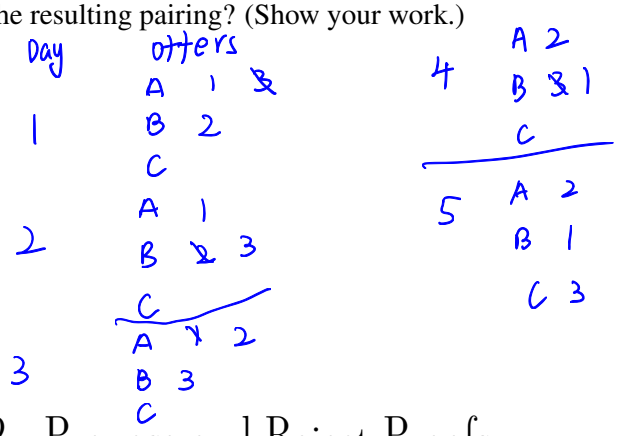
Note 4

Consider the set of jobs $J = \{1, 2, 3\}$ and the set of candidates $C = \{A, B, C\}$ with the following preferences.

Jobs	Candidates
1	A > B > C
2	B > A > C
3	A > B > C

Candidates	Jobs
A	2 > 1 > 3
B	1 > 3 > 2
C	1 > 2 > 3

Run the traditional propose-and-reject algorithm on this example. How many days does it take and what is the resulting pairing? (Show your work.)



2 Propose-and-Reject Proofs

Note 4

Prove the following statements about the traditional propose-and-reject algorithm.

(a) In any execution of the algorithm, if a candidate receives a proposal on day i, then she receives some proposal on every day thereafter until termination.

once a candidate has a proposal on day i, she will always has one job J after day i, this job J will send a proposal to she tommorw

at least

day j

Inductive step

day j+1

(b) In any execution of the algorithm, if a candidate receives no proposal on day i, then she receives no proposal on any previous day j, $1 \leq j < i$.

proof by contraposition → if she receives proposal on any previous day j, $1 \leq j < i$, she will receives a proposal on day i, according to (a)

(c) In any execution of the algorithm, there is at least one candidate who only receives a single proposal. (Hint: use the parts above!)

If every one receive a proposal, then the algorithm will halt.

at least

according to (a), there will be a people receives a proposal at the last day.

3 Be a Judge

Note 4

By stable matching instance, we mean a set of jobs and candidates and their preference lists. For each of the following statements, indicate whether the statement is True or False and justify your answer with a short 2-3 line explanation:

(a) There is a stable matching instance for n jobs and n candidates for $n > 1$, such that in a stable matching algorithm with jobs proposing, every job ends up with its least preferred candidate.

False. every job will be rejected by n-1 People.

but according to 2(c), at least one people will only receive one proposal, he can't say no to others.

(b) In a stable matching instance, if job J and candidate C each put each other at the top of their respective preference lists, then J must be paired with C in every stable pairing.

True (J, C') (J', C) ⇒ J prefer C and C prefer J.

(J, C) will be a rouge couple

(c) In a stable matching instance with at least two jobs and two candidates, if job J and candidate C each put each other at the bottom of their respective preference lists, then J cannot be paired with C in any stable pairing.

False, (J, C)

J1 C1 C2 C1 J1 J2

J2 C1 C2 C2 J1 J2

⇒ (J1, C1)

(J2, C2)

(d) For every $n > 1$, there is a stable matching instance for n jobs and n candidates which has an **unstable** pairing where **every** unmatched job-candidate pair is a rogue couple or pairing.

False ??? true → by the proof of B, only one stable situation

Jobs			
1	A	C	B
2	B	A	C
3	C	B	A

C			
A	1	2	3
B	2	3	1
C	3	1	2

Stable match (A, 1) (B, 2) (C, 3)

unstable (A, 2) [(B, 1) (B, 3) / (C, 1) (C, 3)] → all unstable.

4 Stable Matching III

Note 4

(a) True or False?

(i) If a candidate accidentally rejects a job she prefers on a given day, then the algorithm still always ends with a matching.

False, the last man get the offer, if he reject, the algorithm can't ends.

(ii) The Propose-and-Reject Algorithm never produces a candidate-optimal matching.

False, as the example in (3.c).

(iii) If the same job is last on the preference list of every candidate, the job must end up with its least preferred candidate.

↓ first of others

False

J1	C2	C1
J2	C1	C2

C1	J1	J2
C2	J1	J2

[J1, C2] [J2 C1]

(b) As you've seen from lecture, the jobs-proposing Propose-and-Reject Algorithm produces an employer-optimal stable matching. Let's see if the candidate have any way of improving their standing. Suppose exactly one of the candidates has the power to arbitrarily reject one proposal, regardless of which job she has on her string (if any). Construct an example that illustrates the following: for any $n \geq 2$, there exists a stable matching instance for which using this power helps **every** candidate, i.e. every candidate gets a better job than she would have gotten under the jobs-proposing Propose-and-Reject Algorithm.

watch out any

J1 c3 c2 c1
J2 c1 c3 c2
J3 c2 c1 c3

c1 J1 J3 J2
c2 J2 J1 J3
c3 J3 J2 J1

↓ original.

[J1, c3] → say no → c_3:
[J2, c1]
[J3, c2]

[c2 : J1] say no to J3
[c1, J2]

→ c_3:
[c2, J1]
[c1, J3] say no to j2

→ [c3, J2]
[c2, J1]
[c1, J3]
→ every one happy now.

off-by-one error

Job	preferences
J1	c1 c2 .. cn
J2	c2 c3 .. cn 1
:	
	c3 c4 ... 1 c2
Jn	:
	cn c1 c2 .. cn-1

Codinate	preference
c1	Jn Jn-1 .. J1
c2	J1 Jn . . . J2
c3	J2 J1 . . J3
:	
cn	Jn-1 . . . . J1 Jn

every one have the worst

c1 reject at first day, every one will have the best choice