Stable Matching Intro

Note 4

Goal: We have n candidates and n jobs; each candidate and job has a preference list. We want to pair up candidates and jobs such that there is a stable matching.

Definitions:

- *Stable Matching Instance*: A set of jobs and candidates and their respective preference lists
- *Matching*: Disjoint set of (c_i, j_i) pairs that are matched together
- *Rogue Couple*: a pair (c, j) *not* in the matching that prefers each other over their current matchings (c, j') and (c', j)
- *Stable Matching*: matching with no rogue couples
- *Optimal candidate for job*: highest ranked candidate for a job in any *stable* matching
- *Optimal job for candidate*: highest ranked job for a candidate in any *stable* matching
- *Job optimal*: All jobs get their optimal candidates; a candidate optimal matching is defined similarly
- *Candidate pessimal*: All candidates get their pessimal jobs (i.e. lowest ranked job for each candidate in any *stable* matching); a job pessimal matching is defined similarly

Propose and reject algorithm: Each day,

1. *Morning*: Jobs propose to the top candidate who have not rejected them; note that a job will propose to the same candidate as the previous day if they were not rejected
2. *Afternoon*: Candidates say "maybe" to the best job offer so far, keeping them *in hand* or *on a string*, and say "no" to every other offer
3. *Evening*: Jobs cross off the candidates that have rejected them

The process halts when no rejections happen; all candidates then accept their current offer. Note that the algorithm only produces one stable matching, so there can be other stable matchings not produced by the algorithm.

Improvement lemma: every candidate will only say maybe to better job offers as time goes on. Similarly, every job will only propose to worse candidates as time goes on.

As a result, the propose and reject algorithm always produces a stable matching that is job optimal and candidate pessimal.

Remember, a stable matching *instance* is only defined as the set of jobs, candidates, and preference lists. It does not include the matching itself, nor any algorithm.

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 they receive some proposal on every day thereafter until termination.

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

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

(d) There does not exist 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.

3 Be a Judge

Note 4

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) 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.

(b) 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.

(c) 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.

4 Stable Matching III

Note 4

(a) True or False?

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

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

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

(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 they have on their 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 they would have gotten under the jobs-proposing Propose-and-Reject Algorithm.