CS 70 Discussion 2A

September 11, 2024

Stable Matching

Problem: We have n jobs and n candidates. Each job contains an ordered preference list of all n candidates, and each candidate has an ordered preference list of all n jobs.

Ex:

Job	Preferences
1	$A > B > C$
2	$B > A > C$
3	$A > B > C$

Candidate	Preferences
A	$2 > 1 > 3$
B	$1 > 3 > 2$
C	$1 > 2 > 3$

We want to create a matching of jobs and candidates such that:

1. There are no unmatched jobs or candidates and each job/candidate participates in only one pair.
2. We have no **rogue couples** (i.e. a job and a candidate who are not paired together but would both prefer to be with each other instead of who they are currently paired with).

An example of a stable matching of the above preferences is $(A, 2), (B, 1), (C, 3)$.

Propose-and-Reject Algorithm

Problem: How do we generate a stable matching given a set of preferences?

Algorithm: The **Propose-and-Reject Algorithm** produces a stable matching from a given set of preferences. The algorithm consists of running the following steps once each "day" until termination:

1. Each job sends an offer to its top candidate that hasn't rejected them yet
2. Each candidate rejects all but their favorite job among the offers it got today
3. If there were no rejections today, then we have constructed a stable matching and our algorithm is done running

Propose-and-Reject (Example)

Job	Preferences
1	$A > B > C$
2	$B > A > C$
3	$A > B > C$

Candidate	Preferences
A	$2 > 1 > 3$
B	$1 > 3 > 2$
C	$1 > 2 > 3$

Propose-and-Reject (Example)

Day 1 (Jobs Propose)

Job	Preferences
1	$A > B > C$
2	$B > A > C$
3	$A > B > C$

Candidate	Preferences
A	$2 > 1 > 3$
B	$1 > 3 > 2$
C	$1 > 2 > 3$

Candidate	Offers
A	$1, 3$
B	2
C	

Propose-and-Reject (Example)

Day 1 (Candidates Reject)

Job	Preferences
1	$A > B > C$
2	$B > A > C$
3	~~A~~ $> B > C$

Candidate	Preferences
A	$2 > 1 > 3$
B	$1 > 3 > 2$
C	$1 > 2 > 3$

Candidate	Offers
A	$1, \cancel{3}$
B	2
C	

Propose-and-Reject (Example)

Day 2 (Jobs Propose)

Job	Preferences
1	$A > B > C$
2	$B > A > C$
3	~~A~~ $> B > C$

Candidate	Preferences
A	$2 > 1 > 3$
B	$1 > 3 > 2$
C	$1 > 2 > 3$

Candidate	Offers
A	1
B	$2, 3$
C	

Propose-and-Reject (Example)

Day 2 (Candidates Reject)

Job	Preferences
1	$A > B > C$
2	$\cancel{B} > A > C$
3	$\cancel{A} > B > C$

Candidate	Preferences
A	$2 > 1 > 3$
B	$1 > 3 > 2$
C	$1 > 2 > 3$

Candidate	Offers
A	1
B	$\cancel{2}, 3$
C	

Propose-and-Reject (Example)

Day 3 (Jobs Propose)

Job	Preferences
1	$A > B > C$
2	$\cancel{B} > A > C$
3	$\cancel{A} > B > C$

Candidate	Preferences
A	$2 > 1 > 3$
B	$1 > 3 > 2$
C	$1 > 2 > 3$

Candidate	Offers
A	$1, 2$
B	3
C	

Propose-and-Reject (Example)

Day 3 (Candidates Reject)

Job	Preferences
1	~~$A > B > C$~~
2	~~$B > A > C$~~
3	~~$A > B > C$~~

Candidate	Preferences
A	$2 > 1 > 3$
B	$1 > 3 > 2$
C	$1 > 2 > 3$

Candidate	Offers
A	~~1~~, 2
B	3
C	

Propose-and-Reject (Example)

Day 4 (Jobs Propose)

Job	Preferences
1	~~A~~ $> B > C$
2	~~B~~ $> A > C$
3	~~A~~ $> B > C$

Candidate	Preferences
A	$2 > 1 > 3$
B	$1 > 3 > 2$
C	$1 > 2 > 3$

Candidate	Offers
A	2
B	$3, 1$
C	

Propose-and-Reject (Example)

Day 4 (Candidates Reject)

Job	Preferences
1	~~$A > B > C$~~
2	~~$B > A > C$~~
3	~~$A > B > C$~~

Candidate	Preferences
A	$2 > 1 > 3$
B	$1 > 3 > 2$
C	$1 > 2 > 3$

Candidate	Offers
A	2
B	~~3~~, 1
C	

Propose-and-Reject (Example)

Day 5 (Jobs Propose)

Job	Preferences
1	~~$A > B > C$~~
2	~~$B > A > C$~~
3	~~$A > B > C$~~

Candidate	Preferences
A	$2 > 1 > 3$
B	$1 > 3 > 2$
C	$1 > 2 > 3$

Candidate	Offers
A	2
B	1
C	3

Propose-and-Reject (Example)

Day 5 (Candidates Reject)

Job	Preferences
1	~~$A > B > C$~~
2	~~$B > A > C$~~
3	~~$A > B > C$~~

Candidate	Preferences
A	$2 > 1 > 3$
B	$1 > 3 > 2$
C	$1 > 2 > 3$

Candidate	Offers
A	2
B	1
C	3

No Rejections! Our algorithm is done running after 5 days.

Additional Notes

- Propose-and-Reject just produces one of the possibly several stable matchings (i.e. there can be more stable matchings than what the algorithm may output).
- Propose-and-Reject produces the **job-optimal** (each job is paired with the best possible candidate it can be paired with among all stable pairings) and **candidate-pessimal** (each candidate is paired with the worst possible job it can be paired with among all stable pairings) matching.
- If you have candidates propose instead of the jobs, you get the **candidate-optimal** and **job-pessimal** matching.