10 Feb 2021 — Stable Matching

Announcements

1. Most of you are now invited to join 4820 on Canvas. We'll do another enrollment sync this afternoon.

2. Recall CIS partner finding event Thurs (tomorrow) 9–10:30pm for 4000-level. URL can be found in Monday's lecture notes. ("Lectures" section of 4820 course website.)

3. Meet Caroline Lui. She can answer chat questions.

Can job markets function more efficiently if they are centralized?

Gale & Shapley (1962),

Imagine there are n firms and n workers.

Each firm has a ranking of all workers in order of preference.

Each worker has a ranking of all firms in order of preference.

What does it mean to "do a good job matching workers to firms"?

<u>Definition</u>. A matching is <u>stable</u> if there are no two pairs (f_1, w_1) and (f_2, w_2) such that

- f_1 prefers w_2 to w_1
- w_2 prefers f_1 to f_2.

(Such a configuration is called a <u>blocking pair</u>.)

A matching with a blocking pair is not "self-enforcing."

Examples.

	Top Choice	2nd choice
f_1	w_1	w_2
f_2	w_1	w_2

	Top	2nd
w_1	f_1	f_2
w_2	f_1	f_2

The red & blue matching is the only stable one.

	Top	2nd
f_1	w_1	w_2
f_2	w_2	w_1

	Top	2nd
w_1	f_2	f_1
w_2	f_1	w_2

The red and blue matchings are both stable.

Both are asymmetric: one favors firms, the other favors workers.

The Proposal Algorithm (Gale-Shapley, 1962)

Initially $match(f) = \perp$ and $match(w) = \perp$ $\forall f, w$

// "$\perp$" represents unmatched.

An implementation would specify a rule for choosing f if more than one meets the condition.

while $\boxed{\exists \text{ unmatched firm } f}$ that didn't yet make an offer to every worker:

f picks w, the highest ranked worker it didn't yet make an offer to

"pseudocode"

f makes offer to w

if $match(w) = \perp$

$match(f) = w$

$match(w) = f$

else

let $f' = match(w)$

if w prefers f to f'

$match(w) = f$

$match(f) = w$

$match(f') = \perp$

endif

endwhile

output the set of all pairs $(f, match(f))$.

Analyzing the algorithm

1. Does it always terminate?
2. Does it always output a stable matching? We will show something better: it outputs a stable perfect matching.
   every worker and firm are matched.
3. Does it run efficiently?
   "Efficient" in 4820 will always mean, "Running time is bounded above by $O(p(n))$ where n denotes input length, and $p(n)$ is a polynomial function of n."
   "The algorithm runs in polynomial time."

Termination: Number of offers increases by 1 in each while loop iteration.
No firm makes an offer to same worker twice $\Longrightarrow$ at most n^2 offers are made $\Longrightarrow$ the while loop iterates $\leq n^2$ times.

Running time: $\leq n^2$ loop iterations.
How fast can we do one loop

iteration?

Maintain a FIFO queue of unmatched firms. Initially all firms are in the queue.

At initialization time every firm makes a linked list of workers from most to least preferred. Next offer goes to next worker in linked list. $O(n^2)$

At initialization time every worker makes an array mapping firms to their ranks. $O(n^2)$

With $O(n^2)$ preprocessing time to initialize data structures, the proposal alg can be implemented using $O(1)$ operations per loop iter.

$\Longrightarrow O(n^2)$ running time.