More Stable Matchings, Contrapositives

CSE 417 WI24
Lecture 3

Announcements

Wednesday's lecture is aimed at practice with logic and proving things.

If you've taken a course like Math 300, you might decide to skip.

Slides will be posted tonight so you can decide.

Where Were We?

Last time:

Introduced the Gale-Shapley Algorithm

Wrote (and proved) an implication (an “if-then” statement)

Used the implication and some data structures to show Gale-Shapley can run in $O(n^2)$ time.

Today:

Does Gale-Shapley actually work – is the result a stable matching?

What happens if we change who proposes?

Gale-Shapley Algorithm

Initially all r in R and h in H are free
While there is a free r
    Let h be highest on r's list that r has not proposed to
    if h is free, then match (r, h)
    else //h is not free
    suppose (r', h) are matched
        if h prefers r to r'
            unmatch (r', h)
            match (r, h)

Analyzing Gale-Shapley

Efficient?
Halts in $O(n^2)$ steps. ✔

Works?
Need a matching that's:

- Perfect
- Has no blocking pairs

Claim 3: The algorithm identifies a perfect matching.

Why?

We know the algorithm halts. Which means when it halts every rider is matched.

But we have the same number of horses and riders, and we matched them one-to-one.

Hence, the algorithm finds a perfect matching.

Claim 4: The matching has no blocking pairs.

We want to prove a negative

there is no blocking pair.

That's a good sign for proof by contradiction.

What's proof by contradiction?

I want to know p is true.

Imagine, p were false. Study the world that would result.

Realize that world makes no sense (something false is true)

But the real world does make sense! So p must be true.

Claim 4: The matching has no blocking pairs.

We want to prove a negative

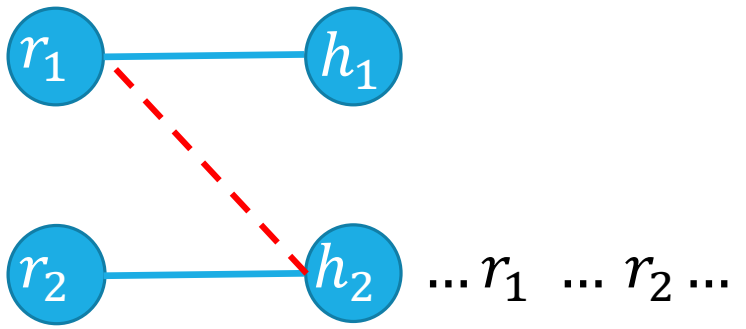
there is no blocking pair.

That's a good sign for proof by contradiction.

Suppose (for contradiction) that (r_1, h_1) and (r_2, h_2) are matched, but

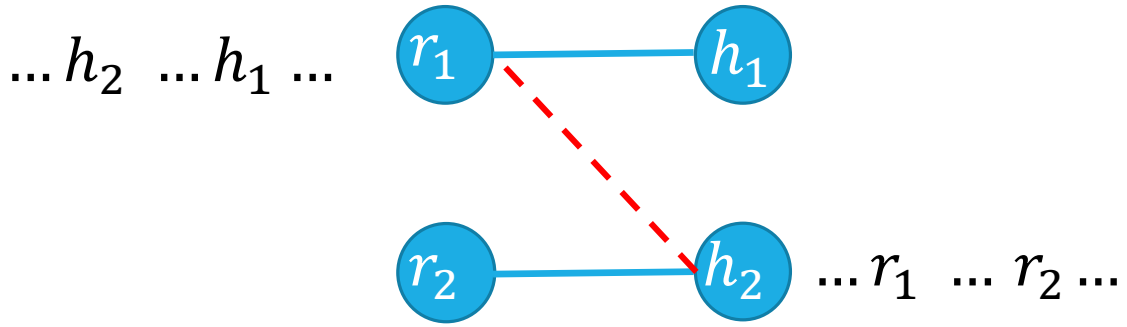
r_1 prefers h_2 to h_1 and

h_2 prefers r_1 to r_2



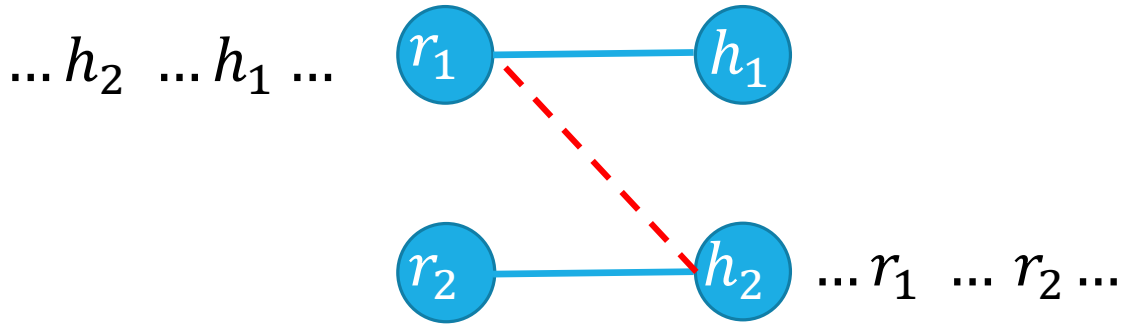
Whenever you use contradiction in a proof, make sure you say you're doing proof by contradiction. Otherwise it's very confusing to the reader.

Claim 4: The matching has no blocking pairs.



How did r_1 end up matched to h_1?

Claim 4: The matching has no blocking pairs.



How did r_1 end up matched to h_1?

They must have proposed to and been rejected by h_2 (since riders propose down their list in order – Observation A).

Why did h_2 reject r_1? It got a better offer from some rider, r'.

If h_2 ever changed matches after that, the match was only better for it, (since horse's partners can only get better for them -- Observation C) so it must prefer r_2 (its final match) to r_1.

But r_1 is before r_2 on h_2's list. That's a contradiction!

Result

Simple, $O(n^2)$ algorithm to compute a stable matching

Corollary
A stable matching always exists.

A "corollary" is a small fact that is easy to see from a big fact.

The corollary isn't obvious!

The "stable roommates problem" doesn't always have a solution:
$2n$ people, rank the other $2n - 1$
Goal is to pair them without any blocking pairs.

Multiple Stable Matchings

Suppose we take our algorithm and let the horses do the "proposing" instead.

h_1 , h_2 r_1 h_1 r_2 , r_1

h_2 , h_1 r_2 h_2 r_1 , r_2

We got a different answer…

What does that mean?

Proposer-Optimality

Some agents might have more than one possible match in a stable matching.

We say that h is a feasible partner for r if there is at least one stable matching where r and h are matched.

When there's more than one stable matching, there is a tremendous benefit to being the proposing side.

Proposer-Optimality

Every member of the proposing side is matched to their favorite of their feasible partners.

Proposer-Optimality

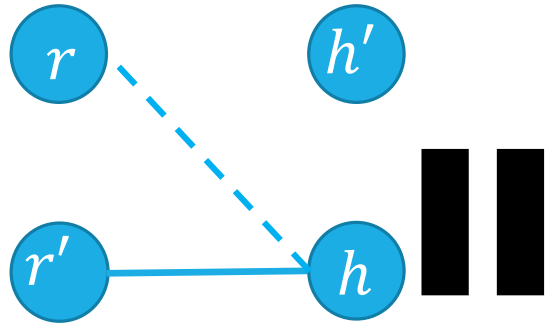
Intuition

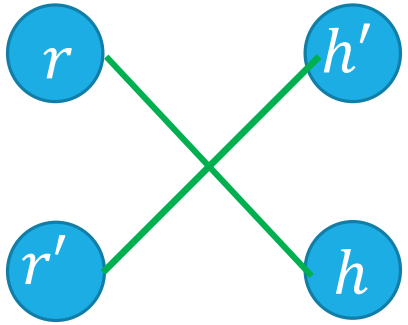
This isn't a full proof (in extra slides)

Suppose r is not matched to their favorite feasible partner, h. It has to be rejected by h during the algorithm (otherwise is matched to h or better). How did that happen? Well h had to have a better offer. But what's the source of that better offer? Call them r'.

There's some stable matching where (r, h) are matched, and r' is matched to h'. For stability h prefers r to r' or r' prefers h' to h.

Must be the second, so r' was also already rejected by a feasible partner.





Implications of Proposer Optimality

Proposer-Optimality

Every member of the proposing side is matched to their favorite of their feasible partners.

We didn't specify which rider proposes when more than one is free

Proposer-optimality says it doesn't matter! You always get the proposer-optimal matching.

So what happens to the other side?

Chooser-Pessimality

A similar argument (it's a quite tricky proof-by-contradiction, but very similar to proposer-optimality), will show that choosing among proposals is a much worse position to be in.

Chooser-Pessimality

Every member of the choosing (non-proposing) side is matched to their least favorite of their feasible partners.

Some More Context and Takeaways

Stable Matching has another common name: “Stable Marriage”

The metaphor used there is “men” and “women” getting married.

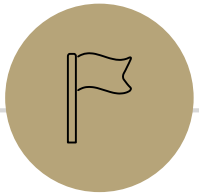
When choosing or analyzing an algorithm, or choosing which parts of a problem to model and which ones to ignore, think about everyone involved, not just the people you’re optimizing for; you might not be able to have it all.

Takeaways

Stable Matchings always exist, and we can find them efficiently.

The GS Algorithm gives proposers their best possible partner

At the expense of those receiving proposals getting their worst possible.



More Implications

When is an implication false?

Computer scientists think of every implication as true or false.

Implications are promises – promises can be broken (or wrong), so they can be false!

The implication $p \rightarrow q$ is false when we can show the promise has been broken. That is when p is true, but q is false.

True or False?

For the purposes of this slide, Alice always carries an umbrella, Bob never carries an umbrella, and it is sunny out right now.

If it is sunny right now, then Alice has her umbrella.

If it is raining right now, then Alice has her umbrella.

If Bob has his umbrella, then it is raining right now.

If it is raining right now, then Bob has his umbrella.

Vacuous Truth

Some of those probably felt weird.

The implication $p \to q$ is vacuously true when p is false.

It's true, but only as a "default" value – it's true precisely because we could not actually use it for anything.

Why is this the rule? See the extra slides.

p	q	$p \to q$
True	True	True
True	False	False
False	True	True
False	False	True

Contrapositives

There are two equivalent ways to write an implication

$p \to q$ and $\neg q \to \neg p$

Contrapositives

To take a contrapositive, switch the “if-part” and “then-part” and negate them both.

If it is raining, then I have my umbrella.

If I do not have my umbrella, then it is not raining

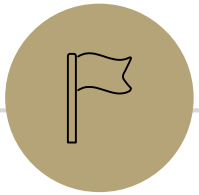
Try it yourself:

If I’m on campus, then I have my Husky card.

Why take contrapositives?

Some implications are easier to prove in their contrapositive form.

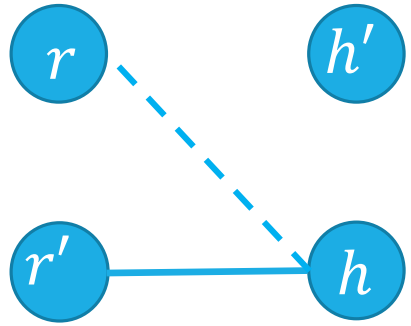
We'll get more practice on Wednesday.



Optional: more context

Proposer-Optimality

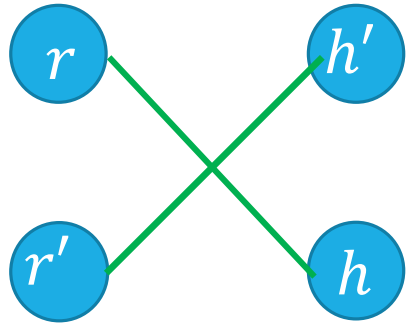
Every member of the proposing side is matched to the favorite of their feasible partners.



Intuition:

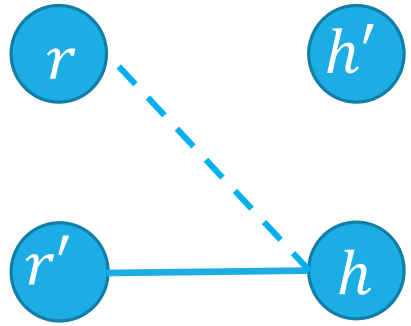
The riders start at the top of their lists. For the claim to be false, some rider r has to be the first to be rejected by their favorite feasible horse, h.

When that happens, h says it prefers some r' (and it does that while r' is still in the "favorite feasible partner" or "too good for you" sections of their list). So r' and h would block any matching



Proposer-Optimality

Every member of the proposing side is matched to the favorite of their feasible partners.



Let's prove it – again by contradiction

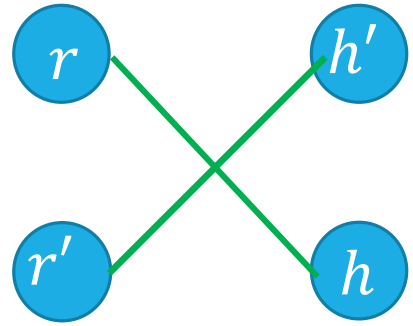
Suppose some rider is not matched to their favorite feasible partner.
Then some r must have been the **first** to be rejected by their favorite feasible partner, h. (Observation A)
And there is an r' that h (temporarily) matched to causing that rejection.

Since r and h are feasible for each other, there is some stable matching (call it M') where (r, h) are matched. The rider r' is matched to some horse h'.

What can we say about r'? They had never been rejected by a feasible partner. So they prefer h to h'.

And h prefers r' to r (by the run of the algorithm).

But then (r', h) are a blocking pair in M'!



Why vacuous truth?

Why do we call vacuous implications true? Why not call them “false”? Or “neither true nor false”?

Answer 1: It’s the convention. Everyone else does it; if you try to call it something else, everyone else will be very confused.

Answer 2: Implications can be general enough to be sometimes vacuous and sometimes not. Consider
“If a number is even and prime, then it is equal to 2”
Depending on what you choose for “number” the implication might be vacuous or not! Among integers greater than 5 it’s vacuously true, among all integers, it’s “regular true”

We’d really rather not think of the implication as “sometimes true, and sometimes neither true nor false” or worse yet “sometimes true and sometimes false” – “true” is the only realistic choice.