Stable Matchings as a Distributed Problem

COSC 373
Spring 2022

Overview

1. Stable Matchings + Gale Shapley Algorithm
2. SMP in the Distributed Setting
3. The Port-Ordering Model

Last Time

Stable matchings (Stable marriage problem, SMP)

1. Set S of students
2. Set U of internships

"agents"

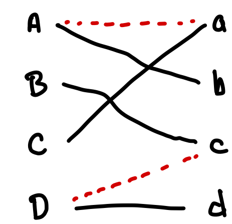
3. Preference list for each agent
   - ranked list of acceptable partners

1-1 matching between students & internships

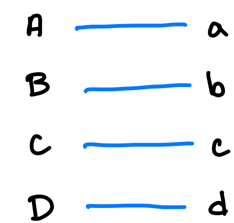
	1	2	3	4
Anna:	a	b	c	
Beck:	c	b	a	
Cameron:	a	d	c	
Daniel:	c	d	b	

	1	2	3	4
a:	A	C	B	
b:	A	B	D	
c:	A	C	D	B
d:	D	C		

Not Stable



Stable



Questions

1. Do stable matchings always exist? → Yes!

2. How can we find a stable matching?

→ algorithm: · apply to 1 internship @ a time

· internships

– defer or reject

At termination deferred applicants form stable matching.

– No ties, "complete prefs" rank <u>all</u> potential partners

Gale-Shapley Algorithm

Ideas.

1. Each student applies to one internship at a time
   - can only apply to another if rejected
   - apply sequentially in decreasing order of preference

2. Each internship can defer at most one applicant at a time
   - reject all but best applicant to date

Terminology. Call a student *active* if

(1) not currently matched

(2) has not been rejected by all acceptable internships

Gale Shapley Algorithm

While some student is active

1. All active students apply to next most favored internship
2. Each internship defers best app so far, rejects others

Return set of deferred pairs

Theorem. (GS, 1962). On all inputs, the GS algorithm terminates and returns a stable matching.

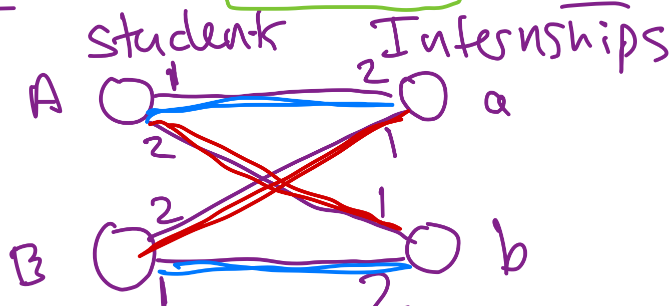
Example.

Students

	1	2	3	4
A	a	b	c	
B	~~c~~	b	a	
C	~~a~~	d	c	
D	c	d	b	

Internships

	1	2	3	4
a :	$\boxed{A}$	~~C~~	B	
b :	~~A~~	B	D	
c :	~~A~~	C	D	~~B~~
d :	D	C		



Pareto optimality

Analysis of Gale-Shapley Algorithm.

Terminology. Call a single iteration of the while-loop a *round*.

Observations:

- **O1** students apply to internships sequentially in decreasing order of preference
- **O2** If student s applies to u in round r, s was rejected by all preferred internships in previous rounds
- **O3** If internship u is matched in round r, then u remains matched in all subseq. rounds
- **O4.** If u rejects s in round r, then u prefers its match to s in all subsequent rounds.

consequence of 2

Gale Shapley Algorithm

While some student is active

1. All active students apply to next most favored internship
2. Each internship defers best app so far, rejects others

Return set of deferred pairs

Termination. Suppose $S = \{s_1, s_2, \dots, s_n\}$ and s_i has m_i acceptable partners. Then the GS algorithm terminates after at most $m = m_1 + m_2 + \dots + m_n = \sum_{i=1}^{n} m_i$ rounds.

$\underbrace{n + n + \cdots + n}_{n}$

$\boxed{m_i = n \Downarrow m = n^2}$

Proof.

Note: if someone is active in round r, they were rejected in round $r-1$

$\longrightarrow$ each round w/ no termination witnesses a rejection.

How many rejections can s_i get (total)? $\leq m_i$

Total rejections before termination: $m_1 + m_2 + m_3 + \dots + m_n$

Stability. Suppose M is the matching of deferred applicants. Then M is a stable matching.

Proof. Consider pair (s, u) that are not matched w/ each other in M

- if s doesn't prefer u to partner in M, then not blocking
- if s does prefer u, then s applied to u and was rejected (02)
- then u prefers match to s, so (s, u) not blocking (Of) //

can this bound be achieved?

Interlude. Why start w/ stable matchings?

- Algorithmic problem that has nothing to do w/ computers
- math problem w/out numbers
- economic problem w/out money
- applications
  - matching medical residents to hospitals
  - kidney exchange
- academic significance
  - GS paper has 6,000+ citations
  - 2012 Nobel prize in economics to Shapley & Roth for work on SM
- endless variations

Personal history:

- subject of my PhD dissertation
- entry to distributed computing
- started from offhand comment

Finally. Stable matching problem is naturally a distributed problem

Stable Matchings as a Distributed Problem

Distributed Features

1. Each agent (student or internship) has own local input (pref. list)
   - preferences are only initially known to self
2. finding solution requires coordination and communication
3. Communication is (naturally) restricted
   - only communicate w/ acceptable partners
   - locality

Questions

1. How could we implement GS in practice in decentralized manner?
   - communication via email

2. What complications/subtleties arise in the decentralized implementation?

Gale Shapley Algorithm

While some student is active
- 1. All active students apply to next most favored internship
- 2. Each internship defers best app so far, rejects others

Return set of deferred pairs

→ Scheduling + synchronization
- broadcast all-to-all comm?
-

→ detecting termination

Complicating Factors

1. synchrony vs asynchrony

   asynchrony → send emails when I get around to it

   synchrony → respond to all of yesterday's emails today

2. Termination detection

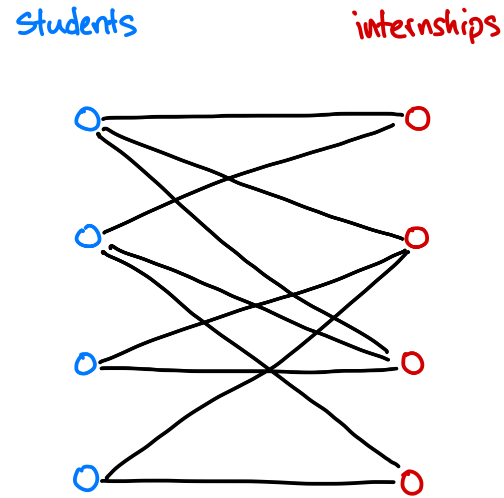
   How do we know when computation is complete?

Port-Ordering (PO) Model

<u>Network</u>

Modeled as a <u>graph</u> $G = (V, E)$

- V = set of <u>nodes</u> a.k.a. <u>vertices</u>
- E = set of <u>edges</u>
  - each edge is a pair of nodes
  - edges = connections between nodes



<u>Variation</u> nodes may have distinct roles

- e.g. students & internships

<u>Terminology</u> vertices $u, v \in V$ are <u>neighbors</u> if (u,v) is an edge in E

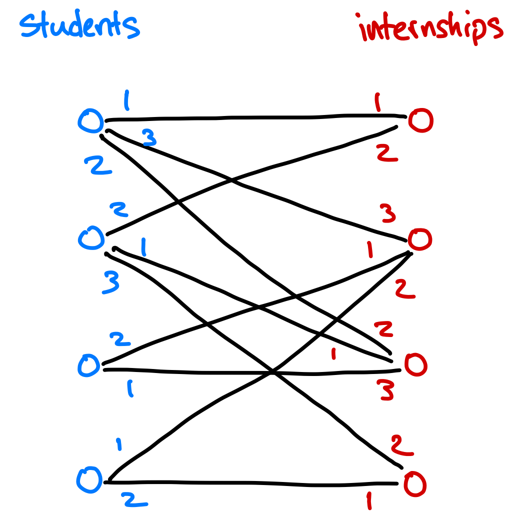
- u and v are <u>incident</u> to (u,v)

The <u>degree</u> of v, denoted $\deg(v)$ is

$$\deg(v) = \#\text{ of neighbors of } v$$
$$= \#\text{ incident edges}$$

Port - Ordering

Each vertex v has an ordering of its neighbors : $1, 2, 3, \ldots, \deg(v)$



Crucial Vertices do not have "unique identifiers"

- "anonymous network"
- nodes distinguish neighbors only by port number

Note : Stable matching instances $\Longrightarrow$ PO Networks

students $\rightsquigarrow$ blue nodes

internships $\rightsquigarrow$ red nodes

} blue nodes only have red neighbors and vice versa $\rightarrow$ graph is bipartite

preferences $\rightsquigarrow$ port ordering

$\rightarrow$ top ranked partner gets port 1, etc.

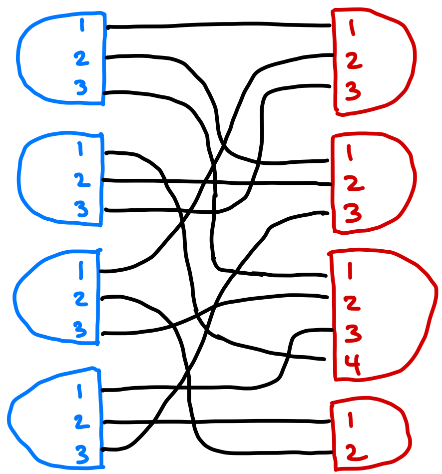
SM Instance

	1	2	3	4
Anna:	a	b	c	
Beck:	c	b	a	
Cameron:	a	d	c	
Daniel:	c	d	b	

	1	2	3	4
a:	A	C	B	
b:	A	B	D	
c:	A	C	D	B
d:	D	C		

$\Downarrow$

PO Network



Computation in PO model

Execution proceeds in <u>synchronous rounds</u>

Each round, each node does:

1. receive messages sent by neighbors in prev. rounds
2. perform local computations
   → no restrictions on complexity
3. sends messages to neighboring nodes, or halts and produces output

<u>More formally</u> a

- Each node has initial state S consisting of local input and ports
- throughout execution, updates state to include contents of received messages each round
- sent messages in round i determined by state in round i (or choice to halt)

} a <u>PO protocol</u> is a specification of messages/halt from states

Gale Shapley as a PO protocol

Procedure for students:

```
Initialize:
  cur = 1
each round i, do
  if i = 1, send "apply" to cur
  if received "reject" from cur in round i-1
    if cur = my degree, return ⊥ and halt
    else, cur ← cur+1, send "apply" to cur
return cur and halt
```

Procedure for internships

```
Initialize:
  cur = ∞
each round i, do
  set reject ← ∅
  for each j from which received "apply" in round i-1
    if j < cur
      add cur to reject
      set cur ← j
    else
      add j to reject
  for each j in reject
    send "reject" to j
return cur
```

Does this work?

Issue with Gale Shapley in PO model:

TERMINATION

When is the procedure complete?

Previously: if there are m total acceptable partners (i.e. edges in graph), then GS returns after $\leq m$ iterations.

- Note m iterations of GS $= 2m$ rounds of PO model

Could we just detect when $2m$ rounds have elapsed and halt/return then?

Gale Shapley Algorithm

While some student is active

1. All active students apply to next most favored internship
2. Each internship defers best app so far, rejects others

Return set of deferred pairs

Coming Up

- if we do not give nodes some "global" information w/ initial state solving SMP in PO model is impossible

- if graph has diameter D, no distributed algorithm can solve SMP in fewer than D-1 rounds

- if nodes have unique IDs, then SMP can be solved w/ out additional "global" information, and termination can be detected after $O(D)$ rounds

- the simpler problem of finding a "maximal matching" can be solved w/ only local information