Stable matching problems

A class of matching problems involving a *stability* criterion.

The Stable Marriage problem

Input: n students and n lecturers; each person ranks all n members of the opposite side in strict order of preference
Output: a stable matching

Definitions:
A *matching* is a set of n disjoint (student, lecturer) pairs

A *blocking pair* for a matching M is a (student, lecturer) pair $(s,l) \notin M$ such that:

- s prefers l to their partner in M, and
- l prefers s to their partner in M

A matching is *stable* if it has no blocking pairs

Stable marriage problem: example instance

Students' preferences	Lecturers' preferences
s_1: l_2 l_4 l_1 l_3	l_1: s_2 s_1 s_4 s_3
s_2: l_3 l_1 l_4 l_2	l_2: s_4 s_3 s_1 s_2
s_3: l_2 l_3 l_1 l_4	l_3: s_1 s_4 s_3 s_2
s_4: l_4 l_1 l_3 l_2	l_4: s_2 s_1 s_4 s_3

Matching is unstable as (s_1,l_4) is a blocking pair

Students' preferences	Lecturers' preferences
s_1: l_2 l_4 l_1 l_3	l_1: s_2 s_1 s_4 s_3
s_2: l_3 l_1 l_4 l_2	l_2: s_4 s_3 s_1 s_2
s_3: l_2 l_3 l_1 l_4	l_3: s_1 s_4 s_3 s_2
s_4: l_4 l_1 l_3 l_2	l_4: s_2 s_1 s_4 s_3

Matching is stable – no blocking pairs

Stability checking

Let M(q) denote the partner of q in matching M

- To test whether matching M is stable:

```
stable = true;
for (student s : students)
   for (each lecturer l such that s prefers l to M(s)
      if (l prefers s to M(l))
         stable = false;
```

- $O(n^2)$ complexity with appropriate data structures (see later)

Finding a stable matching

- A stable matching always exists for a given instance of the Stable Marriage problem

- A stable matching may be found in $O(n^2)$ time using the *Gale/Shapley algorithm*

Gale/Shapley algorithm ("student-oriented" version)

```
assign each person to be free;
while ( some student s is free ) {
   l = first lecturer on s's list to whom s hasn't yet applied;
   // s applies to l
   if (l is free)
      assign s and l to be assigned;
   else if (l prefers s to their partner s′) {
      set s and l to be assigned;
      set s′ to be free;
   } else
      l rejects s; // and s remains free
}
output the n assigned pairs;
```

s_1: $\cancel{l_2}$ l_4 l_1 l_3
s_2: l_3 l_1 l_4 l_2
s_3: l_2 l_3 l_1 l_4
s_4: $\cancel{l_4}$ l_1 l_3 l_2

Students' preferences

l_1: s_2 s_1 s_4 s_3
l_2: s_4 s_3 $\cancel{s_1}$ s_2
l_3: s_1 s_4 s_3 s_2
l_4: s_2 s_1 $\cancel{s_4}$ s_3

Lecturers' preferences

Gale/Shapley algorithm ("student-oriented" version)

```
assign each person to be free;
while ( some student s is free ) {
   l = first lecturer on s's list to whom s hasn't yet applied;
   // s applies to l
   if (l is free)
      assign s and l to be assigned;
   else if (l prefers s to their partner s′) {
      set s and l to be assigned;
      set s′ to be free;
   } else
      l rejects s; // and s remains free
}
output the n assigned pairs;
```

s_1: l_2 l_4 l_1 l_3
s_2: l_3 l_1 l_4 l_2
s_3: l_2 l_3 l_1 l_4
s_4: l_4 l_1 l_3 l_2

Students' preferences

l_1: s_2 s_1 s_4 s_3
l_2: s_4 s_3 s_1 s_2
l_3: s_1 s_4 s_3 s_2
l_4: s_2 s_1 s_4 s_3

Lecturers' preferences

Correctness of the Gale/Shapley algorithm

1. The algorithm terminates with everyone assigned

- No student can be rejected by all the lecturers
- Suppose student s was rejected by every lecturer
- Then all lecturers have been assigned at some point
- But if a lecturer becomes assigned they are never again free
- Thus, once s is rejected by the last lecturer on their list, all lecturers are assigned
- But, there are equal numbers of students and lecturers, and no student is assigned to more than one lecturer
- So all students are assigned, contradiction

Correctness of the Gale/Shapley algorithm

2. The algorithm produces a stable matching

- It is immediate that the assigned pairs form a matching M
- Suppose s prefers l to M(s)
- Then s applied to l at some point, and s was rejected by l
- So l was, or became, assigned to a student that they prefer to s, so l prefers M(l) to s
- Thus (s,l) does not block M
- So M cannot have any blocking pairs, and hence is stable
- This establishes the correctness of the algorithm
  – and also proves that a stable matching exists for every instance of the problem

The "student-optimal" property

- **Note that the algorithm, as given, is non-deterministic**
  - **the order in which free students apply is not specified**

- **Theorem 1:**
  (i) All executions of the student-oriented Gale / Shapley algorithm yield the same stable matching M. (So the non-determinism is immaterial.)

  (ii) In this stable matching M, every student has the *best* partner that they can have in any stable matching.

- **Let M be the stable matching given by the student-oriented Gale/Shapley algorithm. We call M the *student-optimal stable matching*.**

- **Theorem 2: In the student-optimal stable matching, each lecturer has the *worst* partner that they can have in any stable matching.**

- **Similarly the lecturer-oriented Gale/Shapley algorithm (roles of students and lecturers reversed) gives rise to the *lecturer-optimal stable matching*.**

Implementation of the Gale/Shapley algorithm

Deciding whether l prefers s to s'

Assume that the input preference lists are represented as follows:

- sp(s, i)=l if lecturer l is at position i of s's list
- lp(l, i)=s if student s is at position i of l's list

Construct lecturers' ranking lists:

- lr(l, s)=i if and only if lp(l, i)=s
- so lr(l, s) < lr(l, s') if and only if l prefers s to s'

Locating free students efficiently

- Use a stack containing the free students
- Initially place all students on the stack
- The while loop iterates as long as the stack is nonempty
- Pop the stack to obtain the next free student to apply
- If a student is rejected, they are free again and are pushed onto the stack

Analysis of the Gale/Shapley algorithm

- **Clearly the lecturers' ranking arrays can be found in $O(n^2)$ time**
- **Easy to keep track of the assigned pairs, and the next lecturer to whom a student will apply**
- **Each iteration of the while loop involves exactly one "apply" operation**
- **No student ever applies twice to the same lecturer**
- **So the total number of iterations is $\leq n^2$**
- **Therefore the complexity of the algorithm is $O(n^2)$**
- **Notice that the Gale/Shapley algorithm runs in *linear* time in the input size**
  - **an instance involves $2n$ preference lists, each of length n, hence $2n^2$ data values**

Applications of Matching Problems / Algorithms

- **Allocating graduating medical students to hospital posts**
  - e.g. NRMP (USA), CaRMS (Canada), SFAS (Scotland), JRMP (Japan)
- **Allocating school pupils to universities**
  - e.g. Spain, Norway, Australia, Turkey, Iran, Hungary, China
- **School placement**
  - e.g. English local authorities, Boston, New York, Singapore
- **University campus housing allocation**
  - e.g. MIT, Michigan, New York
- **Matching trainee teachers to probationary posts**
  - e.g. Scottish Government Teacher Induction Scheme
- **Kidney exchange**
  - e.g. US, The Netherlands, South Korea, UK
- **Student-project allocation**
  - e.g. Glasgow, Southampton, York

Nobel prize in Economic Sciences, 2012

The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 2012

Alvin E. Roth, Lloyd S. Shapley

Photo: © Linda A. Cicero/Stanford

Alvin E. Roth

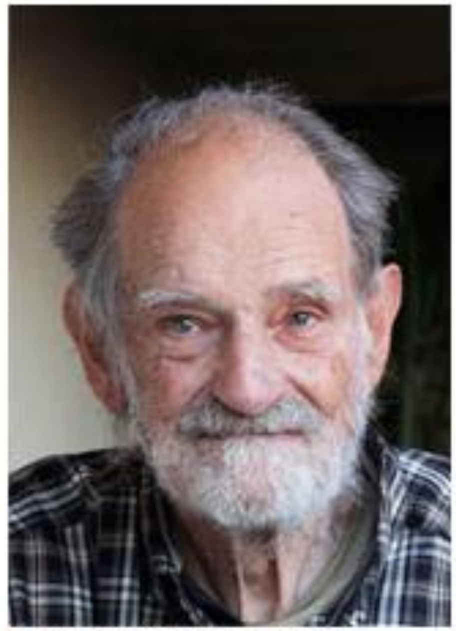

Photo: AP Photo/Reed Saxon

Lloyd S. Shapley

The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 2012 was awarded jointly to Alvin E. Roth and Lloyd S. Shapley *"for the theory of stable allocations and the practice of market design"*

Hospitals / Residents problem (HR)

- n doctors $d_1, d_2, \ldots, d_n$, m hospitals $h_1, h_2, \ldots, h_m$
- Hospital h_i has **capacity** c_i, i.e. has c_i posts available
- Each doctor chooses a subset of the hospitals and ranks them in strict order of preference
- h is **acceptable** to d if h is on d's preference list, otherwise is **unacceptable**
- Each hospital ranks in strict order of preference the doctors for whom it is acceptable
- A **matching** M in an instance of HR is an allocation of doctors to hospitals such that:

1) $(d, h) \in M \Rightarrow d$ and h find each other acceptable
2) Each doctor is matched to at most one hospital
3) No hospital exceeds its capacity

Hospitals / Residents problem: matching

d_1 : h_2 h_1
d_2 : h_1 h_2
d_3 : h_1 h_3
d_4 : h_2 h_3
d_5 : h_2 h_1
d_6 : h_1 h_2

Doctors' preferences

Each hospital has 2 posts

h_1 : d_1 d_3 d_2 d_5 d_6
h_2 : d_2 d_6 d_1 d_4 d_5
h_3 : d_4 d_3

Hospitals' preferences

$M = \{(d_1, h_1), (d_2, h_2), (d_3, h_3), (d_5, h_2), (d_6, h_1)\}$

$(|M| = 5)$

Hospitals / Residents problem: stability

d_1 : h_2 h_1
d_2 : h_1 h_2
d_3 : h_1 h_3
d_4 : h_2 h_3
d_5 : h_2 h_1
d_6 : h_1 h_2

Each hospital has 2 posts

h_1 : d_1 d_3 d_2 d_5 d_6
h_2 : d_2 d_6 d_1 d_4 d_5
h_3 : d_4 d_3

Doctors' preferences **Hospitals' preferences**

- **Matching M is stable if M admits no blocking pair**
  - **(d, h) is a blocking pair of matching M if:**

1. **d, h find each other acceptable and**
2. **either d is unmatched in M or d prefers h to their assigned hospital in M and**
3. **either h is undersubscribed in M or h prefers d to the least preferred doctor assigned to it in M**

Hospitals / Residents problem: unstable matching

d_1 : h_2 h_1
d_2 : h_1 h_2
d_3 : h_1 h_3
d_4 : h_2 h_3
d_5 : h_2 h_1
d_6 : h_1 h_2

Doctors' preferences

Each hospital has 2 posts

h_1 : d_1 d_3 d_2 d_5 d_6
h_2 : d_2 d_6 d_1 d_4 d_5
h_3 : d_4 d_3

Hospitals' preferences

- The matching **M** indicated is not stable
  - **(d_2, h_1)** is a blocking pair

Hospitals / Residents problem: unstable matching

d_1 : h_2 h_1
d_2 : h_1 h_2
d_3 : h_1 h_3
d_4 : h_2 h_3
d_5 : h_2 h_1
d_6 : h_1 h_2

Doctors' preferences

Each hospital has 2 posts

h_1 : d_1 d_3 d_2 d_5 d_6
h_2 : d_2 d_6 d_1 d_4 d_5
h_3 : d_4 d_3

Hospitals' preferences

- The matching **M** indicated is not stable
  - **(d_2, h_1)** is a blocking pair
  - **(d_4, h_2)** is a blocking pair

Hospitals / Residents problem: unstable matching

d_1 : h_2 h_1
d_2 : h_1 h_2
d_3 : h_1 h_3
d_4 : h_2 h_3
d_5 : h_2 h_1
d_6 : h_1 h_2

Each hospital has 2 posts

h_1 : d_1 d_3 d_2 d_5 d_6
h_2 : d_2 d_6 d_1 d_4 d_5
h_3 : d_4 d_3

Doctors' preferences   Hospitals' preferences

- The matching M indicated is not stable
  - (d_2, h_1) is a blocking pair
  - (d_4, h_2) is a blocking pair
  - (d_4, h_3) is a blocking pair

Hospitals / Residents problem: stable matching

d_1 : h_2 h_1
d_2 : h_1 h_2
d_3 : h_1 h_3
d_4 : h_2 h_3
d_5 : h_2 h_1
d_6 : h_1 h_2

Doctors' preferences

Each hospital has 2 posts

h_1 : d_1 d_3 d_2 d_5 d_6
h_2 : d_2 d_6 d_1 d_4 d_5
h_3 : d_4 d_3

Hospitals' preferences

- The alternative matching indicated is stable
  - no blocking pairs
- Example shows that, in a given stable matching,
  - one or more doctors may be unmatched
  - one or more hospitals may be undersubscribed

Algorithmic and structural results

- **A stable matching always exists for an instance of HR**
- **Such a matching may be found in linear time**
  - **by a straightforward extension of the Gale-Shapley algorithm**
  - **doctors apply to hospitals in order of preference**
  - **a hospital accepts a doctor if it is undersubscribed**
  - **if it is full, it compares (*in constant time*) the new doctor to its *least preferred* assignee and rejects one of them accordingly**
  - **a doctor who is rejected by all hospitals remains unmatched**
- **An instance of HR may have many stable matchings**
  - **but all stable matchings for a given instance have the same size and match the same set of doctors**
  - **there are *doctor-optimal* and *hospital-optimal* stable matchings, analogous to student-optimal and lecturer-optimal**