EE360C: Algorithms

Stable Marriage

Pedro Santacruz

Department of Electrical and Computer Engineering
University of Texas at Austin

Fall 2014

The Informal Problem

Consider the problem of optimally matching a set of applicants to a set of open positions.

- Applicants to summer internships
- Applicants to graduate school
- Medical school graduate applicants to residency programs
- Eligible males wanting to marry eligible females

Actually, it seems like it should be easy. Why is it a hard problem, practically?

Let's think about how to solve the problem if we have perfect information...

Stablity and Instability

Given a set of preferences among hospitals and medical students, define a self-enforcing admissions process.

Unstable Pair

Applicant x and hospital y are unstable if:

- x prefers y to its assigned hospital
- y prefers x to one of its admitted students

Stable Assignment

A stable assignment is one with no unstable pairs.

- This is a natural and desirable condition.
- Individual self-interest will prevent any applicant/hospital deal being made.

Formulating the Problem

Consider a set $M = \{m_1, \ldots, m_n\}$ of n men and a set $W = \{w_1, \ldots, w_n\}$ of n women.

- A matching S is a set of ordered pairs, each from $M \times W$, s.t. each member of M and each member of W appears in at most one pair in S.
- A perfect matching S' is a matching s.t. each member of M and each member of W appears in exactly one pair in S'.
- Each man $m \in M$ ranks all of the women; m prefers w to w' if m ranks w higher than w'. We refer to the ordered ranking of m as his preference list.
- Each woman ranks all of the men in the same way.
- An instability results when a perfect matching S contains two pairs (m, w) and (m', w') s.t. m prefers w' to w and w' prefers m to m'.

GOAL: A perfect set of marriages with no instabilities.

An Example

Is the assignment X-C, Y-B, Z-A stable?

favorite ↓ least favorite ↓

	1st	2nd	3rd
Xavier	Amy	Bertha	Clare
Yancey	Bertha	Amy	Clare
Zeus	Amy	Bertha	Clare

Men's Preference Profile

favorite ↓ least favorite ↓

	1st	2nd	3rd
Amy	Yancey	Xavier	Zeus
Bertha	Xavier	Yancey	Zeus
Clare	Xavier	Yancey	Zeus

Women's Preference Profile

No. Bertha and Xavier would hook up.

Questions About Stable Marriage

1. Does there exist a stable matching for every set of preference lists?
2. Given a set of preference lists, can we efficiently construct a stable matching if there is one?

The Gale-Shapley Algorithm

```
Gale-Shapley()
 1  Initially all $m \in M$ and $w \in W$ are free
 2  while $\exists m$ who is free and hasn't proposed to every $w \in W$
 3        do Choose such a man $m$
 4           Let $w$ be the highest ranked in $m$'s preference list
               to whom $m$ has not yet proposed
 5           if $w$ is free
 6              then $(m, w)$ become engaged
 7              else  $w$ is currently engaged to $m'$
 8                    if $w$ prefers $m'$ to $m$
 9                       then $m$ remains free
10                       else  $w$ prefers $m$ to $m'$
11                             $(m, w)$ become engaged
12                             $m'$ becomes free
13  return the set $S$ of engaged pairs
```

But Does it Work?

It’s never a bad idea to convince yourself informally on a simple example...

Some Axioms

- w remains engaged from the point at which she receives her first proposal
- the sequence of partners with which w is engaged gets increasingly better (in terms of her preference list)
- the sequence of women to whom m proposes get increasingly worse (in terms of his preference list)

Observations

Men propose to women in decreasing order of preference (they’re “optimistic”).

Once a woman is matched, she never becomes unmatched (she only “trades up”).

Termination

Theorem

The G-S algorithm terminates after at most n^2 iterations of the while loop.

What is a good measure of progress?

- the number of free men?
- the number of engaged couples?
- the number of proposals made?

Each iteration consists of one man proposing to a woman he has never proposed to before. So we count the number of proposals. After each iteration of the while loop, the number of proposals increases by one; the total number of proposals is upper bounded by n^2.

Towards a Perfect Matching

Theorem

If m is free at some point in the execution of the algorithm, then there is a woman to whom he has not yet proposed.

Proof

If, at some point, m is free but has already proposed to every woman, then every woman must be engaged (because once engaged, they stay engaged, and they would have said yes to m if they weren't engaged when he proposed). Since all n women are engaged there must be n engaged men. This contradicts the claim that m is free.

Towards a Perfect Matching (cont.)

Theorem

The set S returned at termination is a perfect matching.

Proof

Suppose the algorithm terminates with a free man m. Then m must have proposed to every woman (otherwise the while loop would still be active, and we wouldn't be at termination). But this contradicts the previous theorem, which stated that there cannot be a free man that has proposed to every woman.

Correctness

Theorem

Consider an execution of the G-S algorithm that returns a set of pairs S. The set S is a stable matching.

Proof

Assume there is an instability. Then there exist two pairs (m, w) and (m', w') in S s.t. m prefers w' to w and w' prefers m to m'. During execution, m's last proposal must have been to w. Had m proposed to w' at some earlier time? If not, w must be higher than w' on m's preference list (which is a contradition to our assumption that m prefers w' to w). If so, then he was rejected by w' in favor of some other guy m''. Either $m'' = m'$ or w' prefers m' to m'' (since the quality of her match only goes up). Either way, this is a contradiction to our founding assumption.

SUMMARY

The Gale-Shapley algorithm guarantees to find a stable matching for any problem instance.

- How do we implement the Gale-Shapley algorithm efficiently?
- If there are multiple stable matchings, which one does the algorithm find?

Implementation

We can describe a $O(n^2)$ implementation.

Representing Men and Women

- Assume men are named $1 \dots n$
- Assume women are named $1' \dots n'$

Engagements

- maintain a list of free men in a queue
- maintain two arrays of length n, $wife[m]$ and $husband[w]$
  - set entry to 0 if unmatched
  - if m matched to w, then $wife[m] = w$ and $husband[w] = m$

Proposals

- For each man, maintain a list of women, ordered by preference
- Maintain an array $count[m]$ that counts the number of proposals made by man m

Implementation (cont.)

Women Rejecting/Accepting

- For each woman, create inverse of preference list.
- Allows constant time queries:
    - A woman prefers m to m' if $inverse[m] < inverse[m']$

Proposal Process

- The first free man, m, in the queue proposes the woman at the front of his preference list, w
- He increments $count[m]$ and removes w from his preference list
- w accepts the proposal if she is unengaged or prefers m to her current match
- if w accepts, her former match goes back on the queue of men; otherwise m proposes to his next favorite

Understanding the Solution

For a given problem instance, there may be several stable matchings. Do all executions of Gale-Shapley yield the same stable matching? If so, which one?

An instance with two stable matchings:
A-X, B-Y, C-Z
A-Y, B-X, C-Z

	1st	2nd	3rd
Xavier	A	B	C
Yancey	B	A	C
Zeus	A	B	C

	1st	2nd	3rd
Amy	Y	X	Z
Bertha	X	Y	Z
Clare	X	Y	Z

Understanding the Solution (cont.)

For a given problem instance, there may be several stable matchings. Do all executions of Gale-Shapley yield the same stable matching? If so, which one?

- Man m is a valid partner for woman w if there exists some stable matching in which they are matched
- A man-optimal assignment is one in which every man receives the best valid partner

- Claim 1: All executions of GS yield man-optimal assignment, which is a stable matching.
- Claim 2: All executions of GS yield woman-pessimal assignment, which is a stable matching (i.e., each woman receives the worst possible valid partner).

WHAT ABOUT SIMILAR PROBLEMS?

Consider the stable roommate problem. $2n$ people each rank the others from 1 to $2n - 1$. The goal is to assign roommate pairs so that none are unstable.

	1st	2nd	3rd
Adam	B	C	D
Bob	C	A	D
Chris	A	B	D
Doofus	A	B	C

A-B, C-D	$\Rightarrow$	B-C unstable
A-C, B-D	$\Rightarrow$	A-B unstable
A-D, B-C	$\Rightarrow$	A-C unstable

Observation: a stable matching for the stable roommate problem doesn't always exist.

Final Thoughts

Steps in Algorithm Design

- Formulate the problem precisely.
- Design an algorithm for the problem.
- Prove the algorithm correct.
- Give a bound on the algorithm's running time.

Design Techniques

We'll explore algorithm design by enumerating a set of design techniques. And learning to recognize problems that likely belong to one class or another.

Questions

?