CSE 421
Algorithms

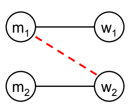
Richard Anderson (for Anna Karlin)
Winter 2006
Lecture 2

Announcements

- It's on the web.
- Homework 1, Due Jan 12
  – It's on the web
- Subscribe to the mailing list
- Anna will have an office hour Monday, Jan 9, 11am-noon. CSE 594

Formal notions

- Perfect matching
- Ranked preference lists
- Stability



Algorithm

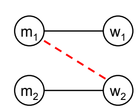
Initially all m in M and w in W are free
While there is a free m
    w highest on m's list that m has not proposed to
    if w is free, then match (m, w)
    else
        suppose (m_2, w) is matched
        if w prefers m to m_2
            unmatch (m_2, w)
            match (m, w)

Does this work?

- Does it terminate?
- Is the result a stable matching?
- Begin by identifying invariants and measures of progress
  – m's proposals get worse
  – Once w is matched, w stays matched
  – w's partners get better

The resulting matching is stable

- Suppose
  – m_1 prefers w_2 to w_1
  – w_2 prefers m_1 to m_2



- How could this happen?

Result

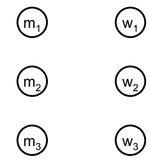
- Simple, $O(n^2)$ algorithm to compute a stable matching
- Corollary
  - A stable matching always exists

A closer look

- Stable matchings are not necessarily fair

m_1: w_1 w_2 w_3
m_2: w_2 w_3 w_1
m_3: w_3 w_1 w_2

w_1: m_2 m_3 m_1
w_2: m_3 m_1 m_2
w_3: m_1 m_2 m_3



Algorithm under specified

- Many different ways of picking m's to propose
- Surprising result
  - All orderings of picking free m's give the same result
- Proving this type of result
  - Reordering argument
  - Prove algorithm is computing something mores specific
    - Show property of the solution – so it computes a specific stable matching

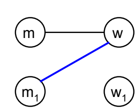
Proposal Algorithm finds the best possible solution for M

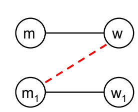
- And the worst possible for W
- (m, w) is valid if (m, w) is in some stable matching
- best(m): the highest ranked w for m such that (m, w) is valid
- $S^* = \{(m, best(m)\}$
- Every execution of the proposal algorithm computes S^*

Proof

- Argument by contradiction
- Suppose the algorithm computes a matching S different from S^*
- There must be some m rejected by a valid partner.
- Let m be the first man rejected by a valid partner w. w rejects m for m_1.
- w = best(m)

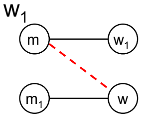
- S^+ stable matching including (m, w)
- Suppose m_1 is paired with w_1 in S^+
- m_1 prefers w to w_1
- w prefers m_1 to m
- Hence, (m_1, w) is an instability in S^+





The proposal algorithm is worst case for W

- In S*, each w is paired with its worst valid partner
- Suppose (m, w) in S* but not m is not the worst valid partner of w
- S^- a stable matching containing the worst valid partner of w
- Let (m_1, w) be in S^-, w prefers m to m_1
- Let (m, w_1) be in S^-, m prefers w to w_1
- (m, w) is an instability in S^-



Could you do better?

- Is there a fair matching
- Design a configuration for problem of size n:
  - M proposal algorithm:
    - All m's get first choice, all w's get last choice
  - W proposal algorithm:
    - All w's get first choice, all m's get last choice
  - There is a stable matching where everyone gets their second choice

Key ideas for Stable Matching

- Formalizing real world problem
  - Model: graph and preference lists
  - Mechanism: stability condition
- Specification of algorithm with a natural operation
  - Proposal
- Establishing termination of process through invariants and progress measure
- Underspecification of algorithm
- Establishing uniqueness of solution

Question

- Goodness of a stable matching:
  - Add up the ranks of all the matched pairs
  - M-rank, W-rank
- Suppose that the preferences are completely random
  - If there are n M's, and n W's, what is the expected value of the M-rank and the W-rank

Expected Ranks

- Expected M rank
- Expected W rank

Expected M rank

- Expected M rank is the number of steps until all M's are matched
  - (Also is the expected run time of the algorithm)
- Each steps "selects a w at random"
  - O(n log n) total steps
  - Average M rank: O(log n)

Expected W rank

- If a w receives k random proposals, the expected rank for w is n/(k+1).
- On the average, a w receives O(log n) proposals
  - The average w rank is O(n/log n)

What is the run time of the Stable Matching Algorithm?

```
Initially all m in M and w in W are free
While there is a free m                Executed at most n^2 times
     w highest on m's list that m has not proposed to
     if w is free, then match (m, w)
     else
          suppose (m2, w) is matched
          if w prefers m to m2
               unmatch (m2, w)
               match (m, w)
```

O(1) time per iteration

- Find free m
- Find next available w
- If w is matched, determine m_2
- Test if w prefer m to m_2
- Update matching

What does it mean for an algorithm to be efficient?