CS 163 & CS 265: Graph Algorithms

Week 8: Matching

Lecture 23: Stable matching

David Eppstein
University of California, Irvine

Fall Quarter, 2022

Overview

Stable matching intuition

We have two sets of n vertices in a complete bipartite graph (any vertex on one side can be matched to any vertex on the other side)

We don't have numeric weights on the edges; instead each vertex has a preference ordering among the vertices on the other side

(like in the voting problem from week 4)

We want a matching that is stable: no pair of vertices would both prefer to be matched to each other (throwing the rest of the matching into disarray) over their assigned matches

Stable matching always exists, can be found in linear time

Real-world applications, I

National Resident Matching Program

- US medical school students / residency training programs
- Has used stable matching algorithms since 1952
- Its algorithms were only recognized to produce stable matchings by Gale and Shapley in 1962, who won the 2012 Nobel Prize in Economics for their work
- Updated in 1997 to be fairer to students and take into account more real-world complications [Roth and Peranson 1999]

Real-world applications, II

French graduating high school students and positions at universities

- Coordinated on a national basis rather than university-by-university
- Handled interactively over the summer before each student starts university studies
- Students receive admission offers throughout the summer and are allowed to choose between new offers and what they previously accepted
- This process simulates the Gale–Shapley algorithm
- Does not require preferences to be listed

[Mathieu 2018]

Real-world applications, III

Content-delivery networks

- Pair web browsers with servers that can deliver requested web content
- Browser preferences: a server that will provide content quickly
- Server preferences: browsers that will be cheap to serve

[Maggs and Sitaraman 2015]

Trigger warning for readings

Not actually a real-world application:
pairing men and women into married couples

- People's preferences do not form a bipartite graph
- People are unwilling to accept centralized algorithmic decisions over their love lives
- The problem requires all participants to be paired, but in the real world some people prefer remaining unpaired
- Opportunities to pair many couples at once are unusual

Nevertheless, much of reading material on this problem talks about men, women, and "stable marriage" instead of stable marriage

Instead, in these lecture notes, I will use "stable matching" and talk about pairing applicants with positions

Problem statement

Input:

- n applicants and n positions
- Each applicant lists all positions in their preferred order
- Each position lists all applicants in their preferred order

Input size: # edges in complete graph $m = n^2$ or total length of all preference lists $O(m)$

Output:

- Perfect matching between applicants and positions
- Must be stable: there is no pair (x, y) where x is an applicant that prefers y over their assigned match, and y is a position that prefers x over their assigned match

Variations allow unequal numbers of applicants and positions, or incomplete preference lists, but to keep things simple we won't.

The Gale–Shapley algorithm

The Gale–Shapley algorithm

Start with an empty matching (all positions unfilled)

While there are still unfilled positions:

- Choose any set U of unfilled positions
  (could be an arbitrary one, could be all of them)
- Each position in U sends an offer to the next applicant on its list (the first one it has not already sent an offer to)
- Each applicant who receives an offer better than their current position (or who does not already have a position) accepts the best offer they receive

Gale–Shapley implementation and analysis

Simplest method: process a single offer at a time

- Decorate positions with iterator into preference list, so we can choose who to send the next offer to in constant time
- Maintain a collection of unfilled positions, so we can find one in constant time
- Decorate each applicant with a dictionary mapping positions to numbers (rank in preference list), so we can compare preferences for two positions in constant time
- Decorate each applicant with their position (if they have one)
- Repeat: Find an unfilled position, make an offer, determine whether the applicant will accept the offer (leaving whatever position they already have), and if so update the matching and the unfilled positions

$O(1)$ time/offer, and $\leq m$ offers, so total time $O(m)$

Why does it produce a matching?

(The other possibility: some position runs through their entire list of preferences and the algorithm crashes because there is no next candidate for them to send an offer to. Why can't this happen?)

Once an applicant receives their first offer, they become matched, and remain matched for the rest of the algorithm

Before the algorithm crashes, some position would have to make an offer to every applicant, resulting in a situation in which every applicant is matched

Because there are equal numbers of applicants and positions, if every applicant is matched, every position is also matched, and the algorithm terminates

Why is the matching stable?

It would be unstable if there exists an applicant x and position y that prefer each other to their assigned match

Suppose for a contradiction that we have an unstable pair (x, y)

Once x receives an offer, they only accept better offers, so they prefer their assigned match to all the positions they rejected
$\Rightarrow y$ could not have made an offer to x, because x would have taken it and kept it over the match they were actually assigned

y makes offers to applicants in order by y's preferences $\Rightarrow y$ must have made offers to all candidates they prefer over their assigned match $\Rightarrow y$ must have made an offer to x

But these two cases are contradictory, so (x, y) cannot exist

Example

Applicant preferences:

- a: RSQP
- b: RQSP
- c: SPQR
- d: SQRP

Position preferences:

- P: abcd
- Q: abdc
- R: acbd
- S: bacd

P makes an offer to a, accepted: a—P
Q makes an offer to a, accepted: a—Q
P makes an offer to b, accepted: a—Q b—P
R makes an offer to a, accepted: a—R b—P
Q makes an offer to b, accepted: a—R b—Q
P makes an offer to c, accepted: a—R b—Q c—P
S makes an offer to b, rejected
S makes an offer to a, rejected
S makes an offer to c, accepted: a—R b—Q c—S
P makes an offer to d, accepted: a—R b—Q c—S d—P

The bigger picture

Which matching do you get?

There can be exponentially many different stable matchings but not factorially many [Karlin et al. 2018]

The one chosen by Gale–Shapley gives each position the best possible applicant (among the ones they can be stably matched to)

It gives each applicant the worst possible applicant (among the ones they can be stably matched to)

It is possible (but more complicated) to find a maximum-weight stable matching using techniques related to flow (next week) [Irving et al. 1987]

Honesty

It is impossible for any position (or any coalition of positions) to get a better match by providing inaccurate preferences (a preference ordering that does not represent their true preferences)

That is, the positions cannot game the system

It is sometimes possible for the applicants to get a better match with false preferences

So if you're worried about one group gaming the system, give that group the role of the positions

[Roth 1982]

Complexity

Many ways of modifying the stable matching problem to include additional real-world constraints lead to NP-hard optimization problems, and/or problems where no stable match exists

- Non-bipartite stable matchings might not exist but can be found in polynomial time if they do exist

  [Irving 1985]

- For medical students and residencies, you might want married couples of students to be assigned to the same place; finding a stable matching under these constraints is NP-complete [Ronn 1990]

Morals of the story

It is possible to use preference orderings to define optimization problems like stable matchings without using numerical weights

Stable matchings have important real-world applications

They can be found in polynomial time using the Gale—Shapley algorithm

Considerations such as which side of the matching should get their highest stable preference or which side should be incentivized to be honest can be handled by swapping roles of applicants and positions

References I

D. Gale and L. S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69(1):9–15, 1962. doi:10.2307/2312726.

Robert W. Irving. An efficient algorithm for the “stable roommates” problem. *Journal of Algorithms*, 6(4):577–595, 1985. doi:10.1016/0196-6774(85)90033-1.

Robert W. Irving, Paul Leather, and Dan Gusfield. An efficient algorithm for the “optimal” stable marriage. *Journal of the ACM*, 34(3): 532–543, 1987. doi:10.1145/28869.28871.

Anna R. Karlin, Shayan Oveis Gharan, and Robbie Weber. A simply exponential upper bound on the maximum number of stable matchings. In *Proc. 50th ACM Symp. Theory of Computing (STOC 2018)*, pages 920–925, 2018. doi:10.1145/3188745.3188848.

Bruce M. Maggs and Ramesh K. Sitaraman. Algorithmic nuggets in content delivery. *ACM SIGCOMM Computer Communication Review*, 45(3):52–66, July 2015. doi:10.1145/2805789.2805800.

References II

Claire Mathieu. College admission algorithms in the real world. Invited presentation at European Symposium of Algorithms 2018, Aalto University, Helsinki, Finland, August 2018. URL `https://archive.org/details/podcast_cast-it-video_claire-mathieu-college-admiss_1000419029804`.

Eytan Ronn. NP-complete stable matching problems. *Journal of Algorithms*, 11(2):285–304, 1990. doi:`10.1016/0196-6774(90)90007-2`.

Alvin E. Roth. The economics of matching: Stability and incentives. *Mathematics of Operations Research*, 7(4):617–628, 1982. doi:`10.1287/moor.7.4.617`.

Alvin E. Roth and Elliott Peranson. The redesign of the matching market for American physicians: Some engineering aspects of economic design. *American Economic Review*, 89(4):748–780, September 1999. doi:`10.1257/aer.89.4.748`.