Z. Abel, B. Chapman, E. Demaine
revised Wednesday 24th January, 2024

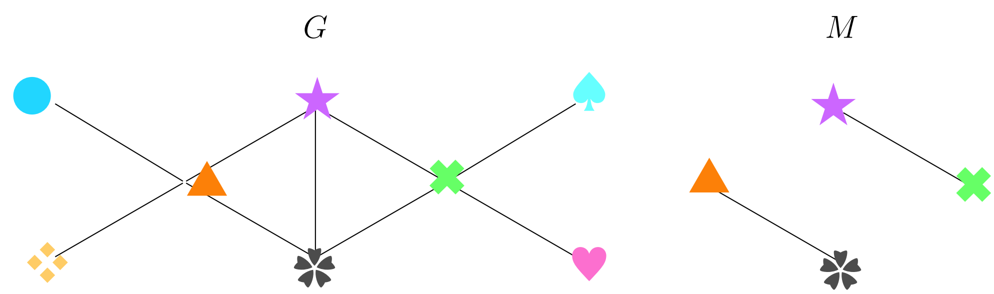
Lecture 12: Matching

1 Matching

Definition 1. *A* matching *in a graph G is a subgraph M of G in which every vertex has degree* 1*.*

I.e. a matching is a disjoint set of edges with their endpoints. We often equate a matching M with its edge set.

Example: M is a matching of size 2 in G.



Applications

- Online dating services
- Assign tasks to servers
- Assign crews to flights, planes to gates

Maximum Matching Problem

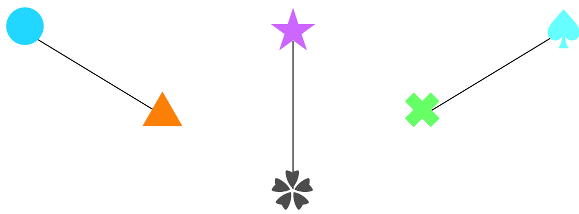
- Edges represent compatibility
- Goal is to find the maximum number of compatible pairs
- In above example, we have a matching of size 2.

- Can we do better?

Definition 2. *A* maximal *matching in G is a matching M in G such that for every other matching M' in G, $M \not\subset M'$.*

Definition 3. *A* maximum *matching in G is a matching M in G such that for every other matching M' in G, $|M'| \leq |M|$.*

Example: M above is maximal, but not maximum. M' is a larger one:



Claim 1. *M' is a maximum matching.*

Proof. G has eight vertices, so a larger matching includes every vertex. However,  and are both degree-1 and adjacent to  .  can only be paired with one of them, so the other must be unpaired. Hence no matching is larger than M'. $\square$

Definition 4. *A matching M in $G = (V, E)$ is* perfect *if it has $\dfrac{|V|}{2}$ edges.*

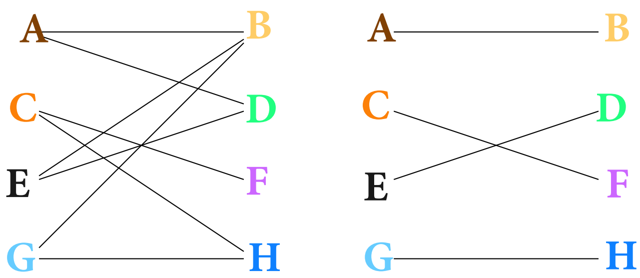
(Non)-example: G above has no perfect matching.

1.1 Matchings in Bipartite Graphs

- Airline: match planes with gates (but not planes with each other or gates with each other)
- Online dating: match heterosexual men with heterosexual women, but not heterosexual men with each other or heterosexual women with each other
- Bipartite graphs don't need the two sides to have the same size
- Perfect matching in a bipartite graph means two sides do have same size

Example: M is a perfect matching in G:

G M



- Sometimes, some matches are more desirable than others
- Represented with a *weight*
- Lower weight means more desirable

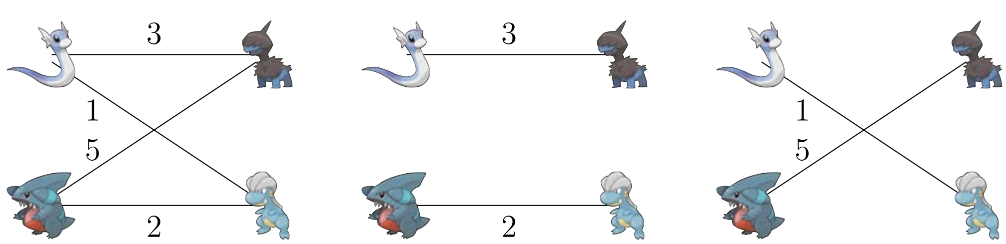
Minimum Weight Matching Problem

Definition 5. *A* weighted graph *is a graph* $G = (V, E)$ *along with a function* $w : E \to \mathbb{R}$.

Definition 6. *Given a weighted graph* (G, w), *the* weight $w(M)$ *of a matching* M *is given by* $w(M) := \sum_{e \in M} w(e)$.

Note: non-edges are often considered to have ∞ weight.

Minimum Weight Matching Problem Given a weighted graph G, find a perfect matching in G with minimum weight (if one exists).



G M M'

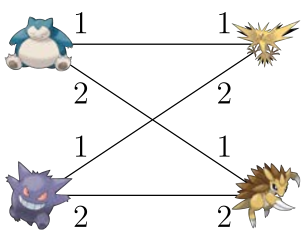
Example: M and M' are perfect matchings in G. M is a min weight perfect matching.

Both Maximum Matching Problem and Min-Weight Matching Problem have fast algorithms! (Beyond scope of this class)

2 Stable Matching Problem

- Similar to Min-Weight Matching Problem
- Bipartite graph with N *Applicants* and N *Evaluators*
- Each Applicant has a ranked preference list of Evaluators
- Each Evaluator has a ranked preference list of Applicants
- Lists are complete and have no draws, i.e. each Applicant ranks every Evaluator and vice versa
- Preferences NEED NOT BE SYMMETRIC!
- Historically: Stable Marriage Problem, Stable Mating Problem

Example: 4 Pokémon want to train.  and  cannot train together, nor can  and .  and  both prefer to train with .  and  both prefer to train with .



- What if we match  with  and  with ?
-  and  are happy!
- Clearly we can't make everyone happy, so is this the best we can do?
- Problem:  and  can just ignore their partners and train together!

Definition 7. *A* rogue couple *in a matching M is a pair (x, y) that are not matched in M but prefer each other over their matches.*

Definition 8. *A matching M is* stable *iff there are no rogue couples.*

Goal: Find a perfect stable matching

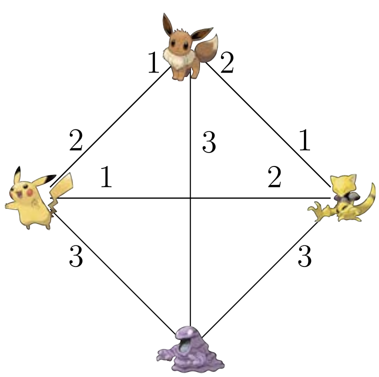
Does such a matching exist?

- In example, match  with  and  with 
-  and  are happy, so are not in a rogue couple
-  and  aren't a rogue couple either because they are matched
- No rogue couples, so matching is stable
- Stable matching exists in this example, but always?

2.1 Non-Bipartite

If the graph is allowed to be non-bipartite (anyone can be matched with anyone else), then stable matchings may not exist!

Example:  likes ,  likes , and  likes . Nobody likes , whose preferences won't even matter...



Theorem 2. *There is no stable matching.*

Proof. Assume for sake of contradiction that there is a stable matching M. WLOG  is matched with . Now  is matched with  but prefers .  prefers everyone over .  and  are a rogue couple, so M is not stable. □

- Perhaps unsurprising
- Not much reason a priori why stable matchings should exist
- However, in bipartite case, there is always a stable matching!

3 Gale-Shapley Algorithm

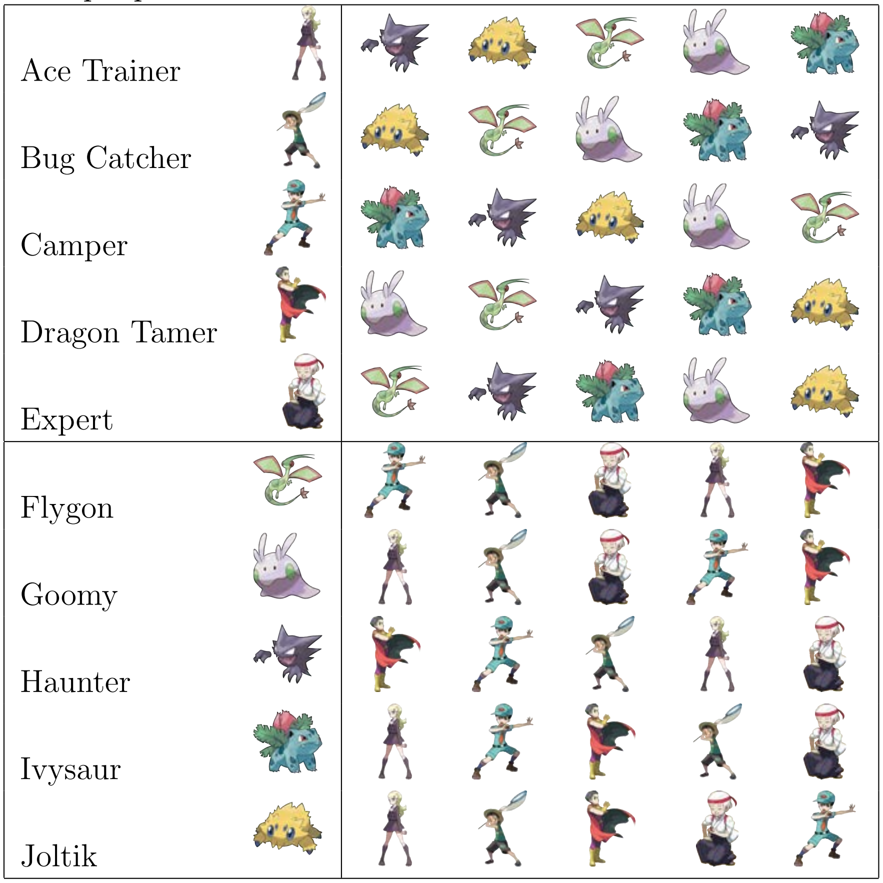
- Developed by David Gale and Lloyd Shapley in 1962
- Extended and applied by Alvin Roth in 1980s
- Nobel Prize in Economics in 2012
- Applications
  - Online dating
  - Residents and hospitals
  - Students and NYC schools
  - Akamai: user requests and servers
- Other names:
  - Stable Matching Algorithm
  - Mating Algorithm
  - Deferred Acceptance Algorithm
  - Propose and Reject Algorithm

3.1 Algorithm Description

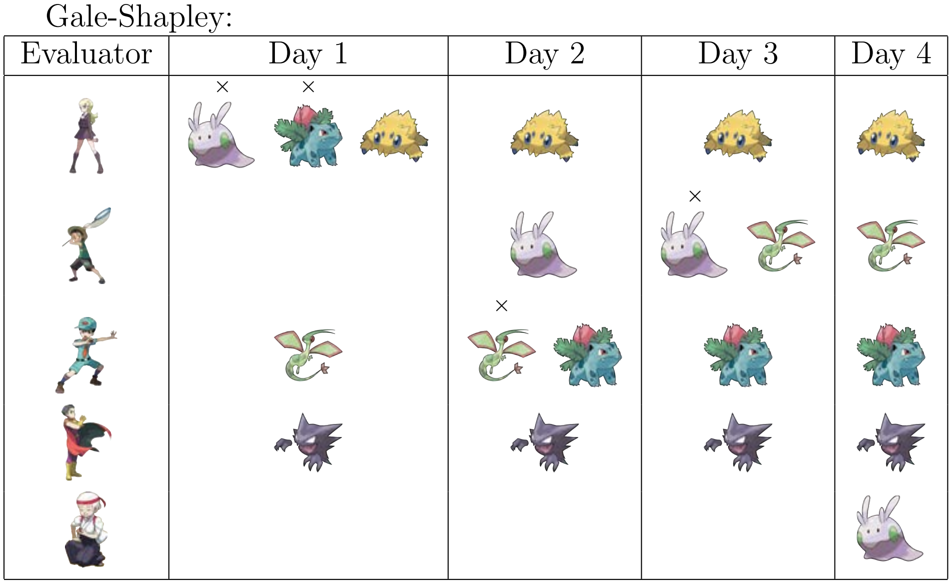
Algorithm is executed in a series of discrete days. Each Applicant keeps track of their preference list, with modifications. Every day:

- Each Applicant a applies to the evaluator e who is a's favorite among those who remain on a's preference list. (If no such Evaluator exists, a stays home and is sad.)
- Each Evaluator e with more than one Applicant rejects all applicants except for e's favorite among e's current Applicants.
- If a was rejected by e, then e is crossed off of a's preference list.
- If every Evaluator has at most one applicant (i.e. if no preference lists were modified), match every Evaluator with their current Applicant (if any).

Example preference lists:



Gale-Shapley:



- Everyone is paired after four days
- Stable?
  - Is F in a rogue couple?

  - F got second choice, might be rogue with first choice.
  - F's first choice C got his first choice, so he's happy.
  - Is G in rogue couple?
  - G got third choice E, might be rogue with first or second choice.
  - G's first choice A got J, whom she likes better than G.
  - G's second choice B got F, whom he likes better than G.
  - Is H in rogue couple?
  - No, got first choice
  - Is I in rogue couple?
  - I got second choice C, might be rogue with first choice A.
  - A doesn't like I.
  - Is J in rogue couple?
  - No, got first choice

In fact, Gale-Shapley *always* produces a stable matching!

To prove that G-S works, we want to prove several things:

Theorem 3. *G-S terminates.*

In fact:

Theorem 4. *G-S terminates* quickly*.*

Theorem 5. *G-S matches everyone.*

Theorem 6. *G-S produces no rogue couples.*

We will prove these by modeling the algorithm as a state machine.

- State: set of rejections that have occurred
- Start state: No rejections
- Transitions: Passage of a single day

Start with termination:

Theorem 7. *G-S terminates by day $N^2 + 1$.*

Actually, G-S always terminates by day $(N-1)^2+1$ and usually terminates much faster, but this is good enough for now.

Proof. Let $f(s)$ be the total number of Evaluators remaining on the Applicants' preference lists. This is a natural-number valued derived variable, and it is strictly decreasing by construction. It starts with value N^2, so there can be at most N^2 transitions. □

Lemma 8. *For every Evaluator e, e's favorite Applicant never gets worse over time.*

Proof. Suppose a is e's favorite Applicant today. Then either a is matched with e (if the algorithm terminates today) or a is among e's Applicants tomorrow. In the latter case, e's favorite Applicant tomorrow is at least as good as a. □

Lemma 9. *For every Applicant a, a's Evaluator never gets better over time.*

Proof. Suppose e is a's Evaluator today. Then e is a's favorite among the Evaluators who have not been crossed off of a's list. If e rejects a, then tomorrow a goes to a different (and hence less preferred) Evaluator who is still on the list. If e does not reject a, then a returns to e (or is matched with e if the algorithm terminates today). □

We can now prove Theorem 5.

Proof of Theorem 5. Assume for sake of contradiction that not everyone is matched. Then there exists at least one Applicant a who is not matched upon termination. Then a must have crossed every Evaluator off of their preference list. Every Evaluator rejected a, so by Lemma 8 must have a match upon termination. There are N Evaluators, so all N Applicants must also be matched. This is a contradiction, so G-S must match everyone □

And now for the main result:

Proof of Theorem 6. Assume for sake of contradiction that there is a rogue couple (a, e). Both are matched, but not to each other. We now condition on whether or not e rejected a.

- If so, e prefers their match to a by Lemma 8, so a and e are not rogue.
- If not, since a's Evaluator is always the favorite out of those who haven't rejected a, a prefers their match to e, so a and e are not rogue.

In either case, we have a contradiction, so G-S must produce a stable matching. □

3.2 Fairness

- G-S terminates (quickly)
- G-S always matches everyone
- G-S always produces a stable matching
- One more consideration, esp. in practice...

Is it better to be an Applicant or an Evaluator?

- Things never get worse for Evaluators or better for Applicants...
- OtOH, things start well for Applicants...
- Fundamental question in sociology: who has power in courtship?
- Seems difficult to answer?
- Very formally: Applicants hold all of the power

Definition 9. *For any party x, a party y is a* feasible match *for x iff there exists a stable matching in which x is matched with y.*

- Everyone has at least one feasible match - the one given by Gale-Shapley
- Are feasible matches unique?
- Not necessarily (and in fact not usually)

Definition 10. *For any party x, x's* optimal match *is x's favorite among all of x's feasible matches.*

Definition 11. *For any party x, x's* pessimal match *is x's least favorite among all of x's feasible matches.*

We finish with a remarkable pair of theorems:

Theorem 10. *Gale-Shapley matches every Applicant with their optimal match!*

Theorem 11. *Gale-Shapley matches every Evaluator with their pessimal match!*

See proofs in recitation tomorrow!