Last class: linear programming and computation

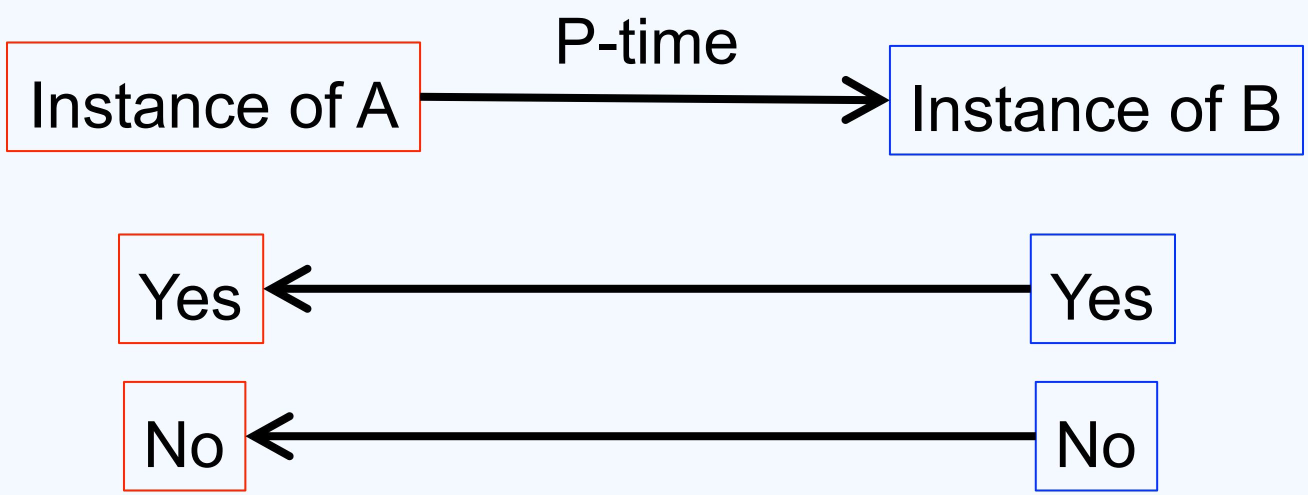
- Linear programming
  - variables are positive real numbers
  - all constraints are linear, the objective is linear
  - in P
- (Mixed) Integer programming
  - (Some) All variables are integer
  - NP-hard
- Basic computation
  - Big O
  - Polynomial-time reduction

Today's schedule

- A real proof of NP-hardness (completeness)
- Computational social choice: the easy-to-compute axiom
  - voting rules that can be computed in P
    - satisfies the axiom
  - Kemeny: a(nother) real proof of NP-hardness
  - IP formulation of Kemeny

How a reduction works?

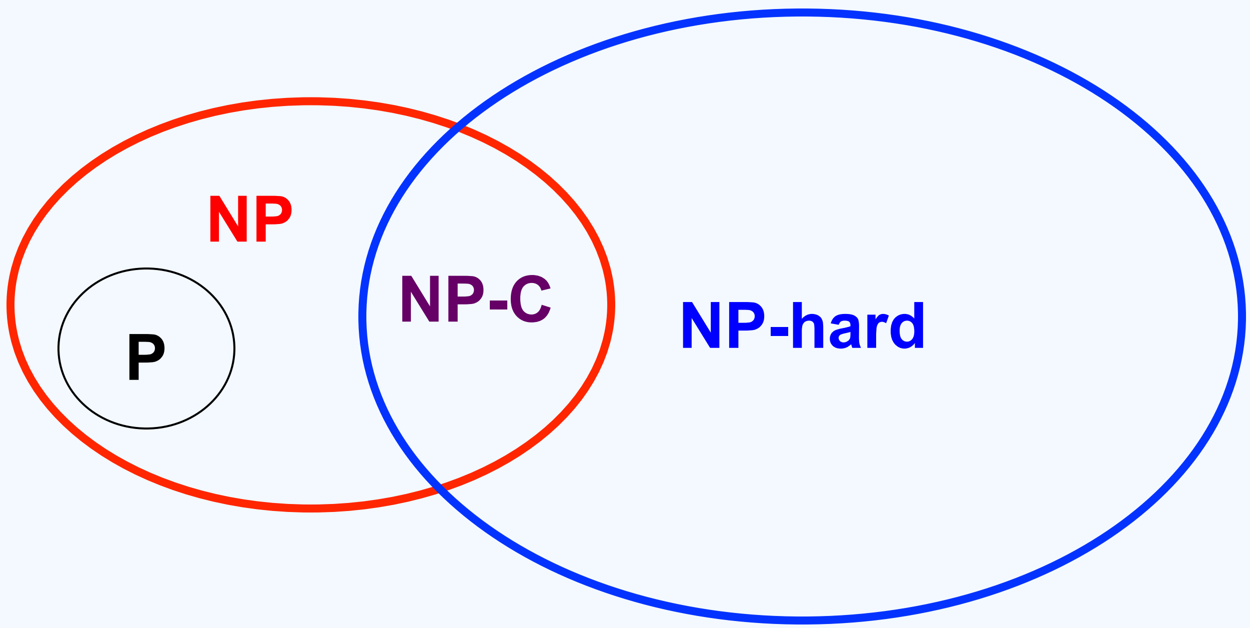
- Polynomial-time reduction: convert an instance of A to an instance of another decision problem B in polynomial-time
  - so that answer to A is “yes” if and only if the answer to B is “yes”



- If you can do this for all instances of A, then it proves that B is HARDER than A w.r.t. polynomial-time reduction
- But it does not mean that B is always harder than A

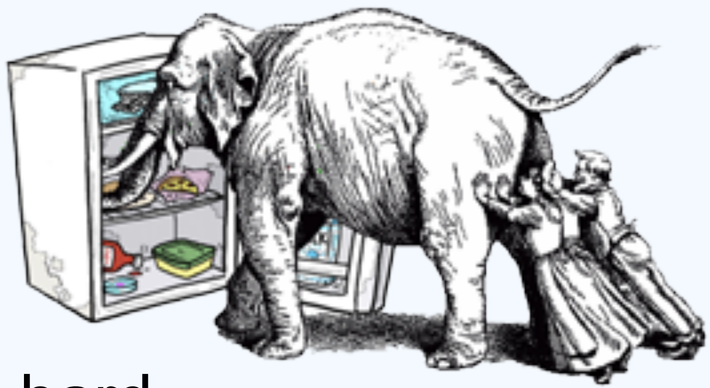
NP-hard and NP-complete problems

- NP-hard problems
  - the decision problems “harder” than any problem in NP
  - for any problem A in NP there exits a P-time reduction from A
- NP-complete problems
  - the decision problems in NP that are NP-hard
  - the “hardest” problems in NP



How to prove a problem is NP-hard

- How to put an elephant in a fridge
  - Step 1. open the door
  - Step 2. put the elephant in
  - Step 3. close the door



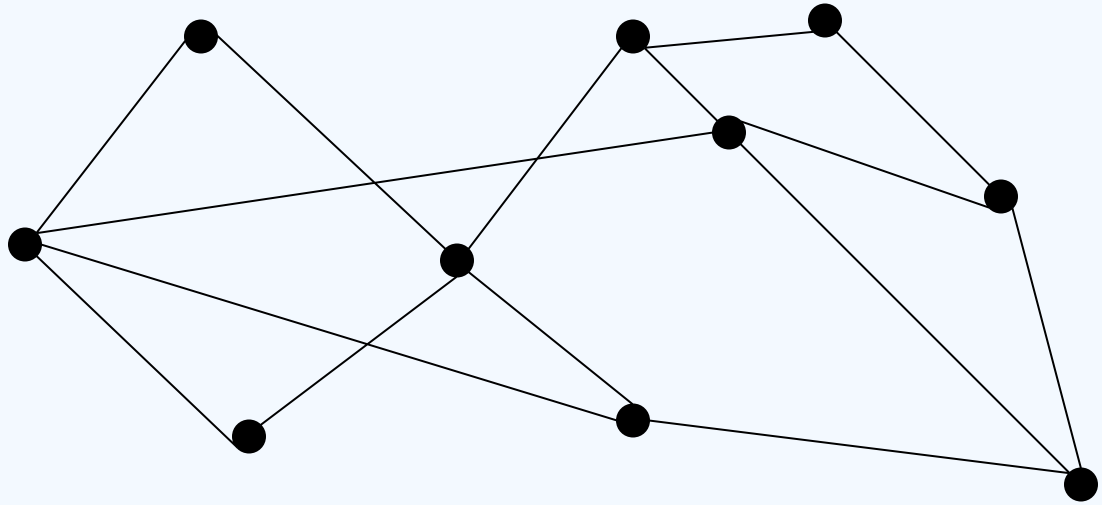
- To prove a decision problem B is NP-hard
  - Step 1. find a problem A
  - Step 2. prove that A is NP-hard
  - Step 3. find a p-time reduction from A to B
- To prove B is NP-complete
  - prove B is NP-hard
  - prove B is in NP (find a p-time verification for any correct answer)

The first known NP-complete problem

- 3SAT
  - Input: a logical formula F in conjunction normal form (CNF) where each clause has exactly 3 literals
    - F = $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$
  - Answer: Is F satisfiable?
- 3SAT is NP-complete (Cook-Levin theorem)

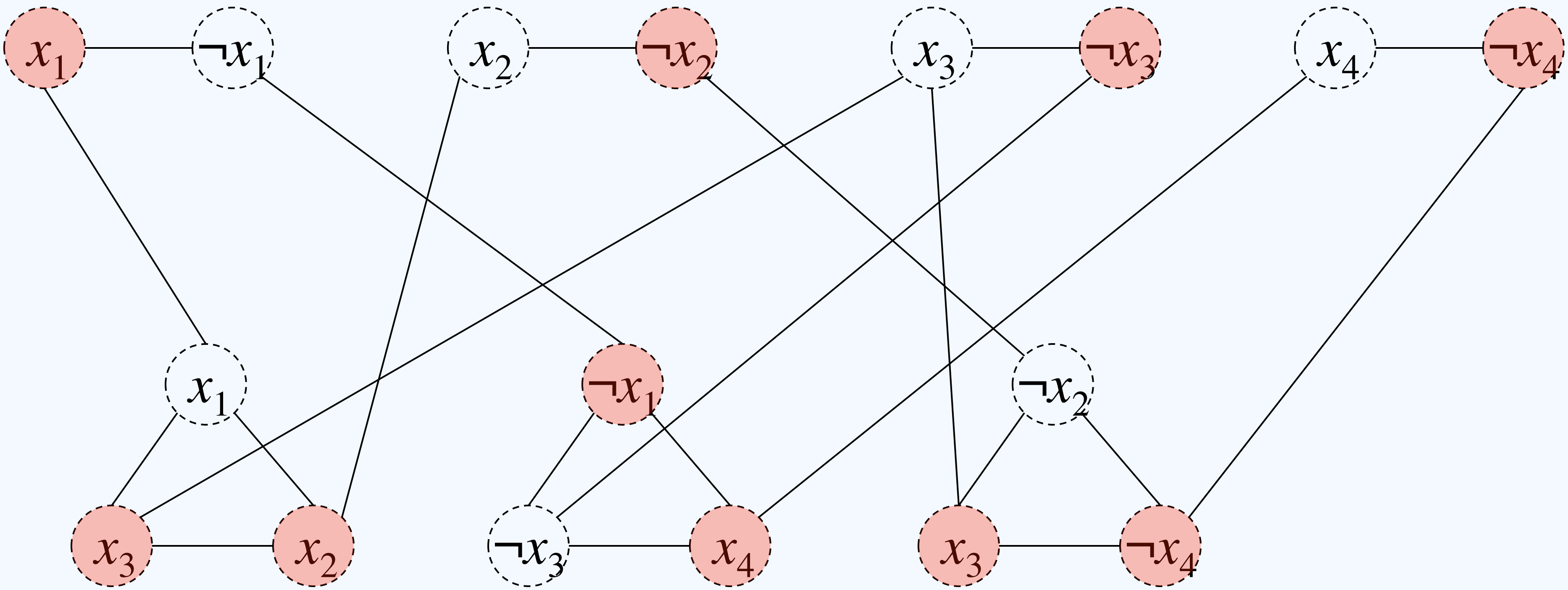
Vertex cover (VC)

- Vertex cover (VC):
  - Given a undirected graph and a natural number k.
  - Does there exists a set S of no more than k vertices so that every edge has an endpoint in S
- Example: Does there exists a vertex cover of 4?



VC is NP-complete

- Given F= $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$
- Does there exist a vertex cover of 4+2*3?



Notes

- More details: http://cgm.cs.mcgill.ca/~athens/cs507/Projects/2001/CW/npproof.html
- A yes to B must correspond to a yes to A
  - if yes↔no then this proves coNP-hardness
- The best source for NP-complete problems
  - *Computers and Intractability: A Guide to the Theory of NP-Completeness*
  - by M. R. Garey and D. S. Johnson
  - cited for >46k times [Google Scholar]
    - vs the “most cited book” *The Structure of Scientific Revolutions* 59K

The easy-to-compute axiom

- A voting rule satisfies the easy-to-compute axiom if computing the winner can be done in polynomial time
  - P: easy to compute
  - NP-hard: hard to compute
  - assuming P≠NP

The winner determination problem

- Given: a voting rule r
- Input: a preference profile D and an alternative c
  - input size: $nm\log m$
- Output: is c the winner of r under D?

Computing positional scoring rules

- If following the description of r the winner can be computed in p-time, then r satisfies the easy-to-compute axiom
- Positional scoring rule
  - For each alternative (m iter)
    - for each vote in D (n iter)
      - find the position of m, find the score of this position
  - Find the alternative with the largest score (m iter)
  - Total time $O(mn+m)=O(mn)$

Computing the weighted majority graph

- For each pair of alternatives c,d (m(m-1) iter)
  - let k = 0
  - for each vote R
    - if c>d add 1 to the counter k
    - if d>c subtract 1 from k
  - the weight on the edge $c \rightarrow d$ is k

Kemeny's rule

- Kendall tau distance
  - $K(R,W)$= # {different pairwise comparisons}

$$K(b > c > a, a > b > c) = 2$$

- Kemeny(D)=argmin$_W$ $K(D,W)$

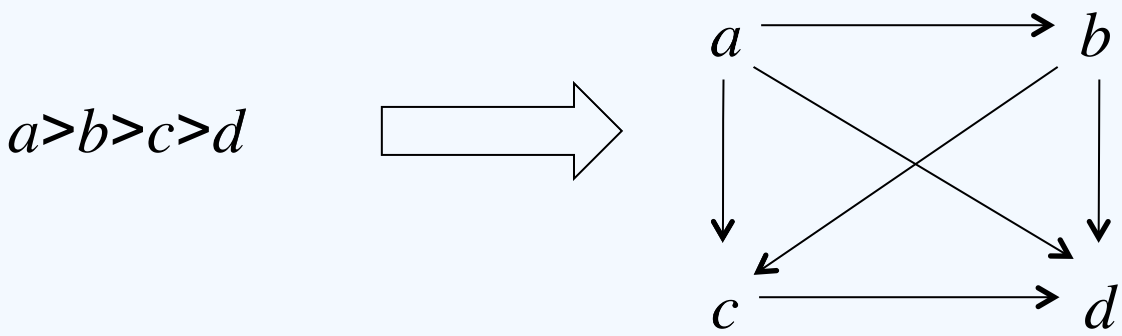
  =argmin$_W$ $\Sigma_{R\in D} K(R,W)$
- For single winner, choose the top-ranked alternative in Kemeny(D)

Computing the Kemeny winner

- For each linear order W (m! iter)
  - for each vote R in D (n iter)
    - compute $\mathrm{K}(R,W)$
- Find W^* with the smallest total distance
  - $W^* = \mathrm{argmin}_W \mathrm{K}(D,W) = \mathrm{argmin}_W \Sigma_{R\in D}\mathrm{K}(R,W)$
  - top-ranked alternative at W^* is the winner
- Takes exponential $O(m!n)$ time!

Kemeny

- Ranking $R \rightarrow$ direct acyclic complete graph $G(R)$



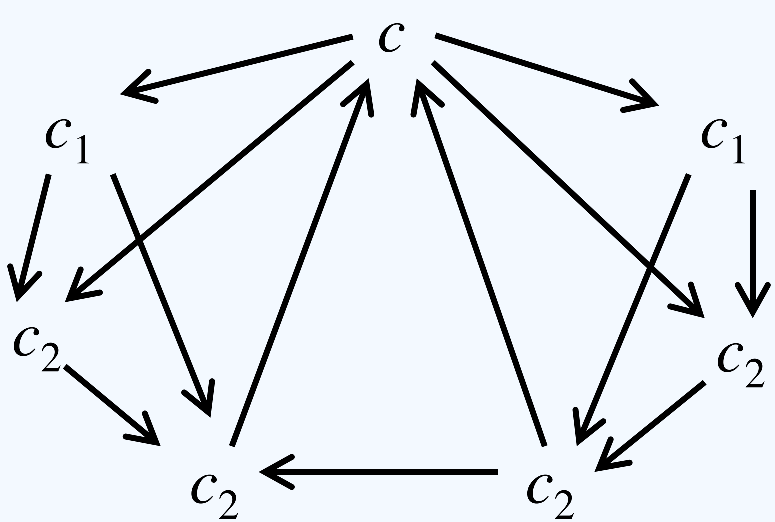
- Given the WMG $G(D)$ of the input profile D
- $\mathrm{K}(D,W) = \Sigma_{a\rightarrow b \in G(W)} (n\text{-}w(a\rightarrow b))/2$

$$= \text{constant} - \Sigma_{a\rightarrow b \in G(W)} w(a\rightarrow b)/2$$

- $\text{argmin}_W \, \mathrm{K}(D,W) = \text{argmax}_W \Sigma_{a\rightarrow b \in G(W)} w(a\rightarrow b)$

Kemeny is NP-hard to compute

- Reduction from feedback arc set
  - Given a directed graph and a number k
  - does there exist a way to eliminate no more than k edges to obtain an acyclic graph?



Satisfiability of easy-to-compute

<table>
<tr><th>Rule</th><th>Complexity</th></tr>
<tr><td>Positional scoring</td><td rowspan="6">P</td></tr>
<tr><td>Plurality w/ runoff</td></tr>
<tr><td>STV</td></tr>
<tr><td>Copeland</td></tr>
<tr><td>Maximin</td></tr>
<tr><td>Ranked pairs</td></tr>
<tr><td>Kemeny</td><td rowspan="3">NP-hard</td></tr>
<tr><td>Slater</td></tr>
<tr><td>Dodgson</td></tr>
</table>

Solving Kemeny in practice

- For each pair of alternatives a, b there is a binary variable x_{ab}
  - x_{ab} = 1 if $a>b$ in W
  - x_{ab} = 0 if $b>a$ in W
- max $\Sigma_{a,b} w(a\rightarrow b) x_{ab}$

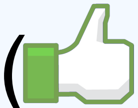
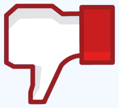
  s.t. for all a, b, $x_{ab}+x_{ba}=1$ No edges in both directions

  for all a, b, c, $x_{ab}+x_{bc}+x_{ca}\leq 2$ No cycle of 3 vertices
- Do we need to worry about cycles of >3 vertices? Homework

Advanced computational techniques

- Approximation
- Randomization
- Fixed-parameter analysis

Next class: combinatorial voting

- In California, voters voted on 11 binary issues ( / )
  - 2^{11}=2048 combinations in total
  - $5/11$ are about budget and taxes

- Prop.30 Increase sales and some income tax for education
- Prop.38 Increase income tax on almost everyone for education