Tutorial: Computational Voting Theory

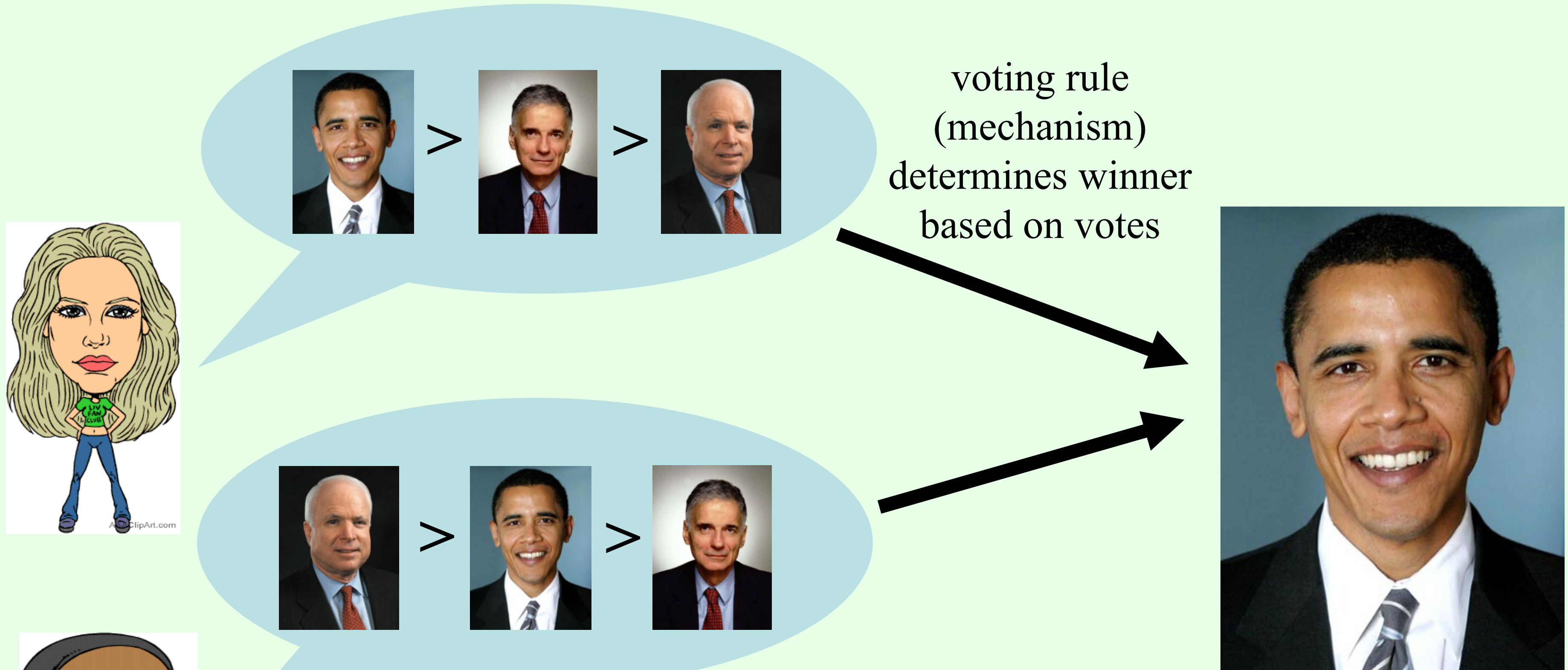
Vincent Conitzer & Ariel D. Procaccia

Outline

- 1. Introduction to voting theory
- 2. Hard-to-compute rules
- 3. Using computational hardness to prevent manipulation and other undesirable behavior in elections
- 4. Selected topics (time permitting)

Introduction to voting theory

Voting over alternatives



- Can vote over other things too
  - Where to go for dinner tonight, other joint plans, …

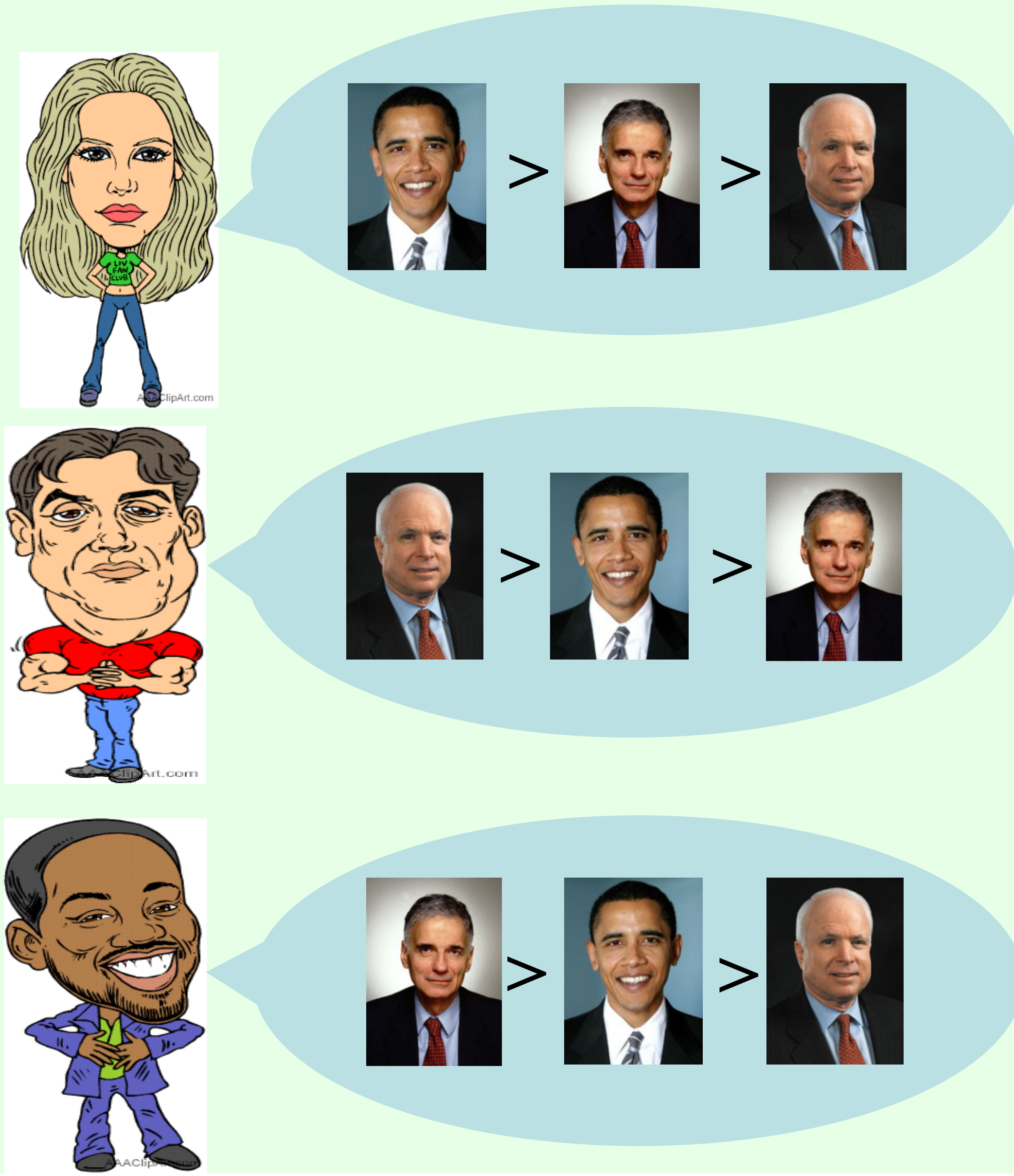
Voting (rank aggregation)

- Set of m candidates (aka. alternatives, outcomes)
- n voters; each voter ranks all the candidates
  - E.g., for set of candidates {a, b, c, d}, one possible vote is b > a > d > c
  - Submitted ranking is called a vote
- A voting rule takes as input a vector of votes (submitted by the voters), and as output produces either:
  - the winning candidate, or
  - an aggregate ranking of all candidates
- Can vote over just about anything
  - political representatives, award nominees, where to go for dinner tonight, joint plans, allocations of tasks/resources, …
  - Also can consider other applications: e.g., aggregating search engines' rankings into a single ranking

Example voting rules

- Scoring rules are defined by a vector (a_1, a_2, …, a_m); being ranked ith in a vote gives the candidate a_i points
  - Plurality is defined by (1, 0, 0, …, 0) (winner is candidate that is ranked first most often)
  - Veto (or anti-plurality) is defined by (1, 1, …, 1, 0) (winner is candidate that is ranked last the least often)
  - Borda is defined by (m-1, m-2, …, 0)
- Plurality with (2-candidate) runoff: top two candidates in terms of plurality score proceed to runoff; whichever is ranked higher than the other by more voters, wins
- Single Transferable Vote (STV, aka. Instant Runoff): candidate with lowest plurality score drops out; if you voted for that candidate, your vote transfers to the next (live) candidate on your list; repeat until one candidate remains
- Similar runoffs can be defined for rules other than plurality

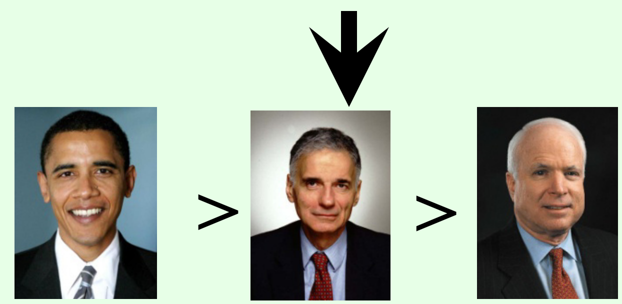
Pairwise elections



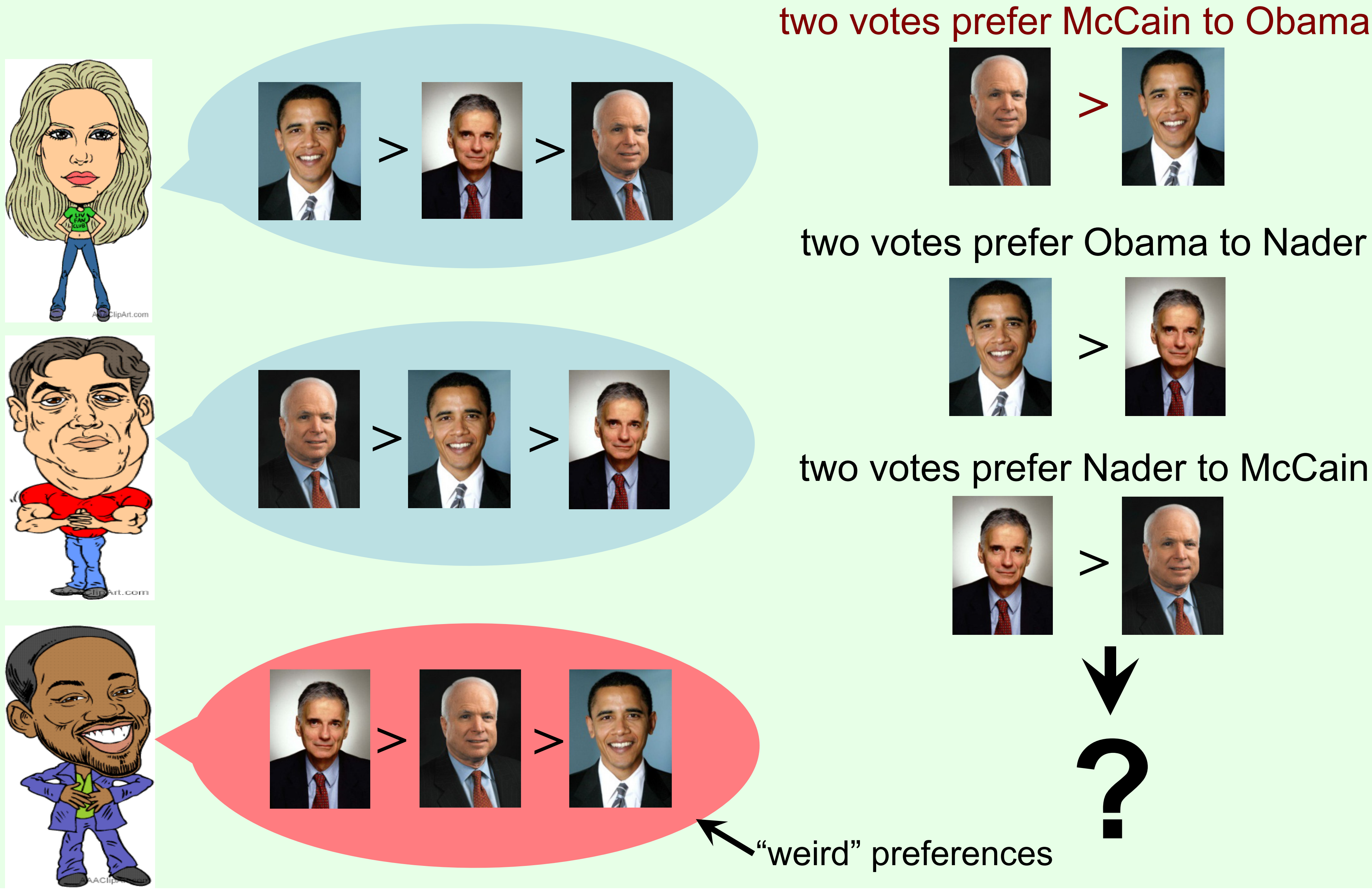
two votes prefer Obama to McCain

two votes prefer Obama to Nader

two votes prefer Nader to McCain



Condorcet cycles



Voting rules based on pairwise elections

- Copeland: candidate gets two points for each pairwise election it wins, one point for each pairwise election it ties
- Maximin (aka. Simpson): candidate whose worst pairwise result is the best wins
- Slater: create an overall ranking of the candidates that is inconsistent with as few pairwise elections as possible
  - NP-hard!
- Cup/pairwise elimination: pair candidates, losers of pairwise elections drop out, repeat

Even more voting rules…

- Kemeny: create an overall ranking of the candidates that has as few *disagreements* as possible (where a disagreement is with a vote on a pair of candidates)
  - NP-hard!
- Bucklin: start with k=1 and increase k gradually until some candidate is among the top k candidates in more than half the votes; that candidate wins
- Approval (not a ranking-based rule): every voter labels each candidate as approved or disapproved, candidate with the most approvals wins

Choosing a rule

- How do we choose a rule from all of these rules?
- How do we know that there does not exist another, “perfect” rule?
- Let us look at some criteria that we would like our voting rule to satisfy

Condorcet criterion

- A candidate is the Condorcet winner if it wins all of its pairwise elections
- Does not always exist…
- … but the Condorcet criterion says that if it does exist, it should win

- Many rules do not satisfy this
- E.g. for plurality:
  - b > a > c > d
  - c > a > b > d
  - d > a > b > c
- a is the Condorcet winner, but it does not win under plurality

Majority criterion

- If a candidate is ranked first by a majority (> ½) of the votes, that candidate should win
  - Relationship to Condorcet criterion?

- Some rules do not even satisfy this
- E.g. Borda:
  - a > b > c > d > e
  - a > b > c > d > e
  - c > b > d > e > a
- a is the majority winner, but it does not win under Borda

Monotonicity criteria

- Informally, monotonicity means that “ranking a candidate higher should help that candidate,” but there are multiple nonequivalent definitions
- A weak monotonicity requirement: if
  - candidate w wins for the current votes,
  - we then improve the position of w in some of the votes and leave everything else the same,

  then w should still win.
- E.g., STV does not satisfy this:
  - 7 votes b > c > a
  - 7 votes a > b > c
  - 6 votes c > a > b
- c drops out first, its votes transfer to a, a wins
- But if 2 votes b > c > a change to a > b > c, b drops out first, its 5 votes transfer to c, and c wins

Monotonicity criteria…

- A strong monotonicity requirement: if
  - candidate w wins for the current votes,
  - we then change the votes in such a way that for each vote, if a candidate c was ranked below w originally, c is still ranked below w in the new vote

  then w should still win.
- Note the other candidates can jump around in the vote, as long as they don't jump ahead of w
- None of our rules satisfy this

Independence of irrelevant alternatives

- Independence of irrelevant alternatives criterion: if
  - the rule ranks a above b for the current votes,
  - we then change the votes but do not change which is ahead between a and b in each vote

  then a should still be ranked ahead of b.
- None of our rules satisfy this

Arrow's impossibility theorem [1951]

- Suppose there are at least 3 candidates
- Then there exists no rule that is simultaneously:
  - Pareto efficient (if all votes rank a above b, then the rule ranks a above b),
  - nondictatorial (there does not exist a voter such that the rule simply always copies that voter's ranking), and
  - independent of irrelevant alternatives

Muller-Satterthwaite impossibility theorem [1977]

- Suppose there are at least 3 candidates
- Then there exists no rule that simultaneously:
  - satisfies unanimity (if all votes rank a first, then a should win),
  - is nondictatorial (there does not exist a voter such that the rule simply always selects that voter's first candidate as the winner), and
  - is monotone (in the strong sense).

Manipulability

- Sometimes, a voter is better off revealing her preferences insincerely, aka. manipulating
- E.g. plurality
  - Suppose a voter prefers a > b > c
  - Also suppose she knows that the other votes are
    - 2 times b > c > a
    - 2 times c > a > b
  - Voting truthfully will lead to a tie between b and c
  - She would be better off voting e.g. b > a > c, guaranteeing b wins
- All our rules are (sometimes) manipulable

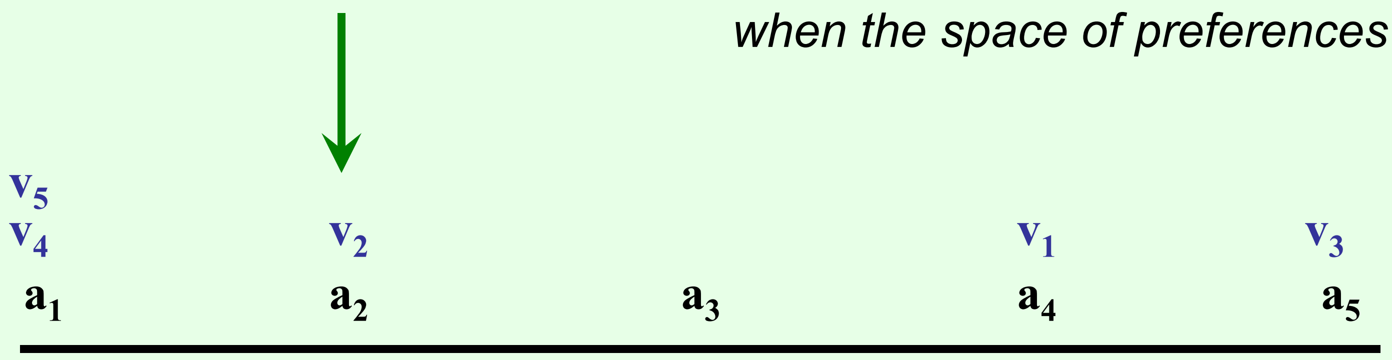
Gibbard-Satterthwaite impossibility theorem

- Suppose there are at least 3 candidates
- There exists no rule that is simultaneously:
  - onto (for every candidate, there are some votes that would make that candidate win),
  - nondictatorial (there does not exist a voter such that the rule simply always selects that voter's first candidate as the winner), and
  - nonmanipulable

Single-peaked preferences

- Suppose candidates are ordered on a line
- Every voter prefers candidates that are closer to her most preferred candidate
- Let every voter report only her most preferred candidate ("peak")
- Choose the median voter's peak as the winner
  - This will also be the Condorcet winner
- Nonmanipulable!

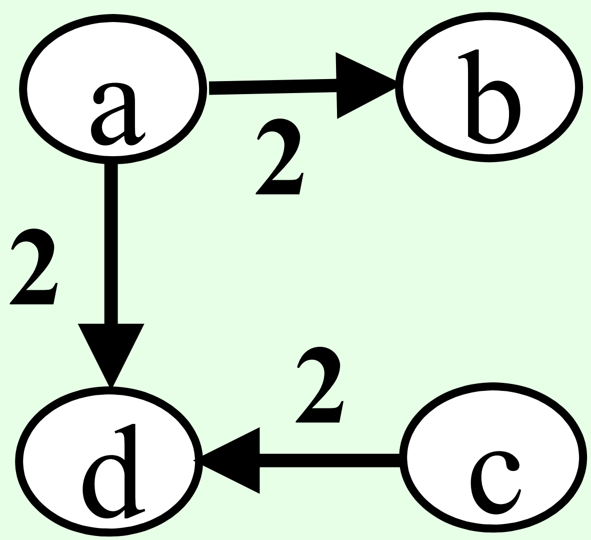
Impossibility results do not necessarily hold when the space of preferences is restricted



Hard-to-compute rules

Pairwise election graphs

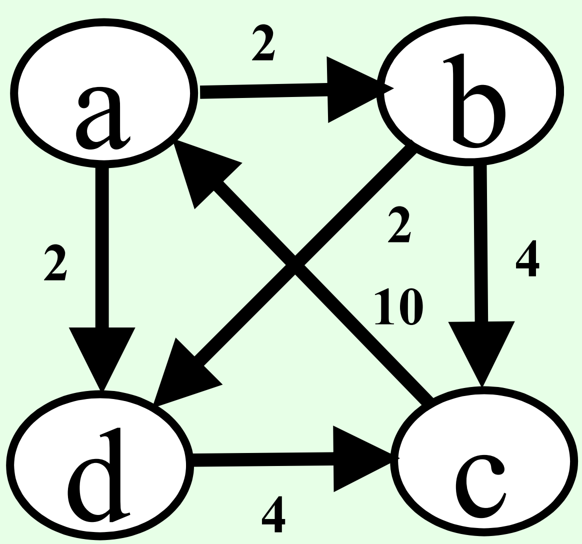
- Pairwise election between a and b: compare how often a is ranked above b vs. how often b is ranked above a
- Graph representation: edge from winner to loser (no edge if tie), weight = margin of victory
- E.g., for votes a > b > c > d, c > a > d > b this gives



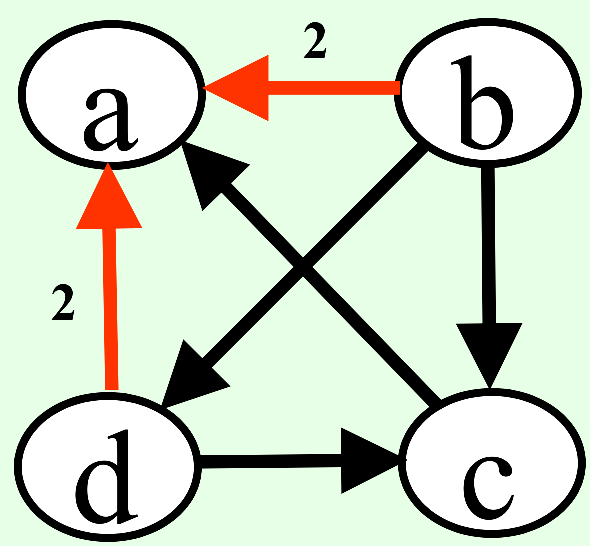
Kemeny on pairwise election graphs

- Final ranking = acyclic tournament graph
  - Edge (a, b) means a ranked above b
  - Acyclic = no cycles, tournament = edge between every pair
- Kemeny ranking seeks to minimize the total weight of the inverted edges

pairwise election graph



Kemeny ranking

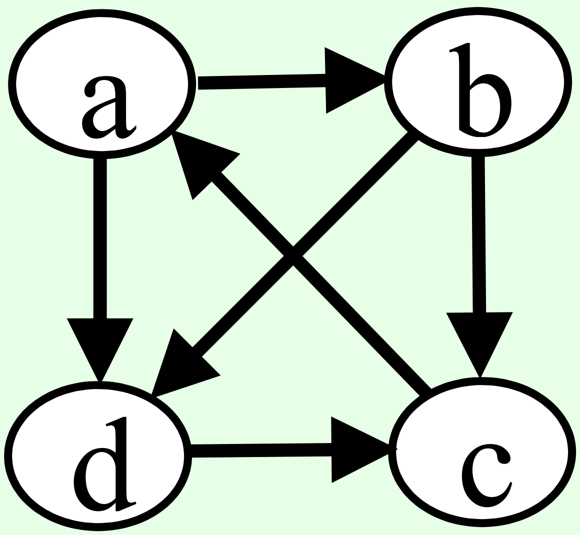


(b > d > c > a)

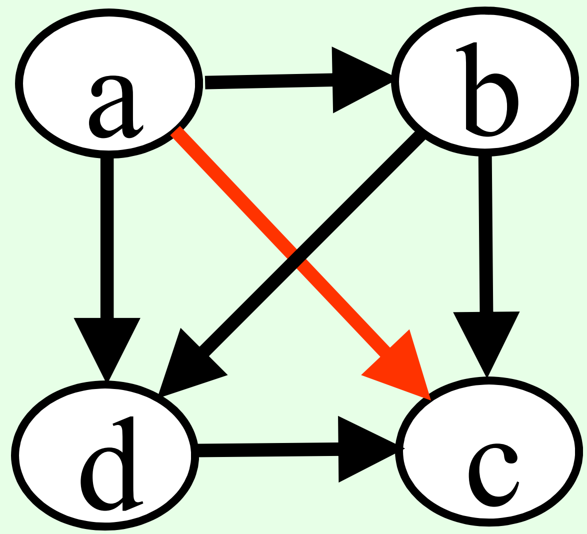
Slater on pairwise election graphs

- Final ranking = acyclic tournament graph
- Slater ranking seeks to minimize the number of inverted edges

pairwise election graph



Slater ranking



(a > b > d > c)

An integer program for computing Kemeny/Slater rankings

$y_{(a, b)}$ is 1 if a is ranked below b, 0 otherwise

$w_{(a, b)}$ is the weight on edge (a, b) (if it exists)

in the case of Slater, weights are always 1

minimize: $\Sigma_{e \in E} w_e y_e$

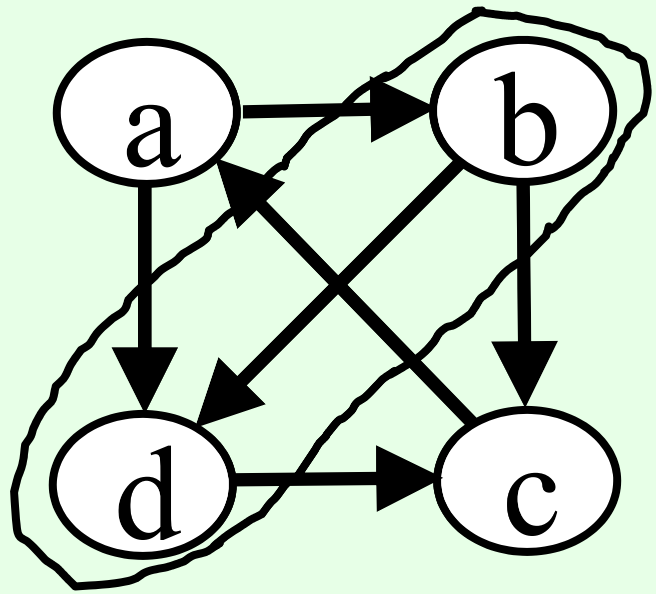
subject to:

for all $a, b \in V$, $y_{(a, b)} + y_{(b, a)} = 1$

for all $a, b, c \in V$, $y_{(a, b)} + y_{(b, c)} + y_{(c, a)} \geq 1$

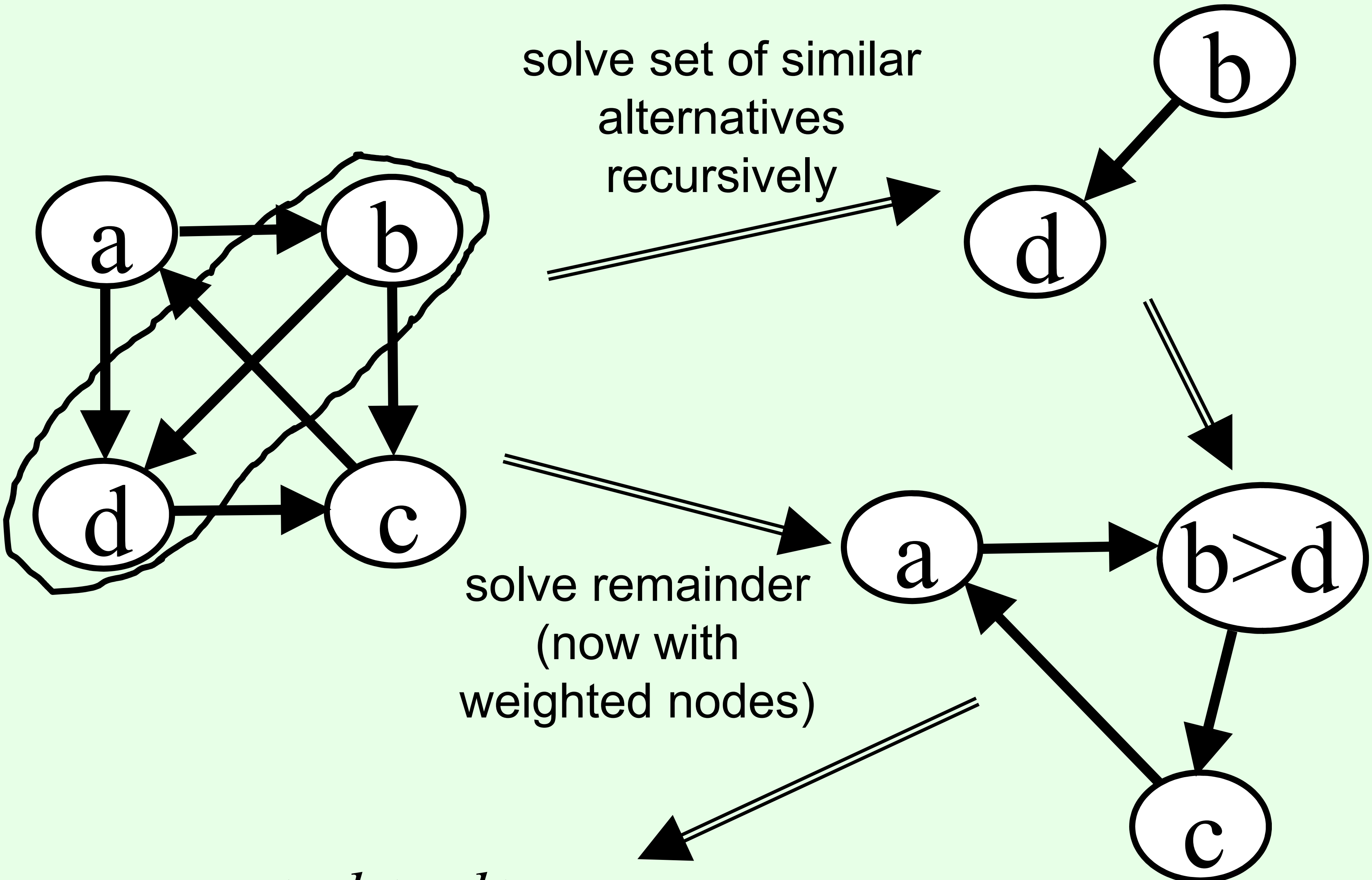
Preprocessing trick for Slater

- Set S of similar alternatives: against any alternative x outside of the set, all alternatives in S have the same result against x



- There exists a Slater ranking where all alternatives in S are adjacent
- A nontrivial set of similar alternatives can be found in polynomial time (if one exists)

Preprocessing trick for Slater…



$a > b > d > c$

A few references for computing Kemeny / Slater rankings

- Betzler et al. How similarity helps to efficiently compute Kemeny rankings. AAMAS’09
- Conitzer. Computing Slater rankings using similarities among candidates. AAAI’06
- Conitzer et al. Improved bounds for computing Kemeny rankings. AAAI’06
- Davenport and Kalagnanam. A computational study of the Kemeny rule for preference aggregation. AAAI’04
- Meila et al. Consensus ranking under the exponential model. UAI’07

Computational hardness as a barrier to manipulation

Inevitability of manipulability

- Ideally, our mechanisms are strategy-proof, but may be too much to ask for
- Recall Gibbard-Satterthwaite theorem:

  Suppose there are at least 3 alternatives

  There exists no rule that is simultaneously:
  - onto (for every alternative, there are some votes that would make that alternative win),
  - nondictatorial, and
  - strategy-proof
- Typically don't want a rule that is dictatorial or not onto
- With restricted preferences (e.g., single-peaked preferences), we may still be able to get strategy-proofness
- Also if payments are possible and preferences are quasilinear

Computational hardness as a barrier to manipulation

- A (successful) manipulation is a way of misreporting one's preferences that leads to a better result for oneself
- Gibbard-Satterthwaite only tells us that for some instances, successful manipulations exist
- It does not say that these manipulations are always easy to find
- Do voting rules exist for which manipulations are computationally hard to find?

A formal computational problem

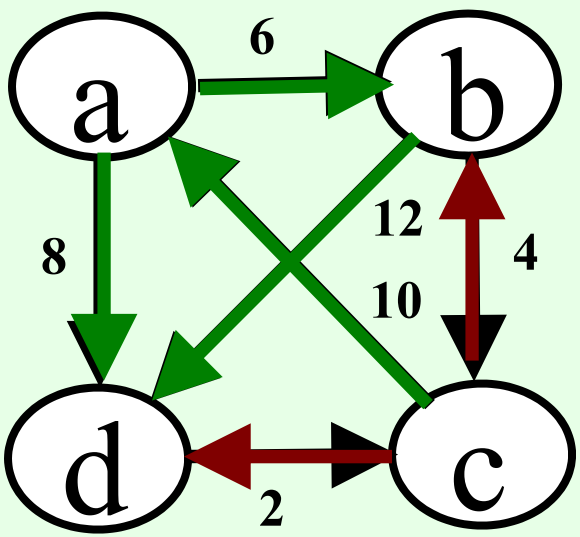
- The simplest version of the manipulation problem:
- CONSTRUCTIVE-MANIPULATION:
  - We are given a voting rule *r*, the (unweighted) votes of the other voters, and an alternative *p*.
  - We are asked if we can cast our (single) vote to make *p* win.
- E.g., for the Borda rule:
  - Voter 1 votes A > B > C
  - Voter 2 votes B > A > C
  - Voter 3 votes C > A > B
- Borda scores are now: A: 4, B: 3, C: 2
- Can we make B win?
- Answer: YES. Vote B > C > A (Borda scores: A: 4, B: 5, C: 3)

Early research

- **Theorem.** CONSTRUCTIVE-MANIPULATION is NP-complete for the second-order Copeland rule. [Bartholdi, Tovey, Trick 1989]
  - Second order Copeland = alternative's score is sum of Copeland scores of alternatives it defeats
- **Theorem.** CONSTRUCTIVE-MANIPULATION is NP-complete for the STV rule. [Bartholdi, Orlin 1991]
- Most other rules are easy to manipulate (in P)

Ranked pairs rule [Tideman 1987]

- Order pairwise elections by decreasing strength of victory
- Successively "lock in" results of pairwise elections unless it causes a cycle



Final ranking:
c>a>b>d

- **Theorem.** CONSTRUCTIVE-MANIPULATION is NP-complete for the ranked pairs rule [Xia et al. IJCAI 2009]

"Tweaking" voting rules

- It would be nice to be able to tweak rules:
  - Change the rule slightly so that
    - Hardness of manipulation is increased (significantly)
    - Many of the original rule's properties still hold
- It would also be nice to have a single, universal tweak for all (or many) rules
- One such tweak: add a preround [Conitzer & Sandholm IJCAI 03]

Adding a preround

[Conitzer & Sandholm IJCAI-03]

- A preround proceeds as follows:
  - *Pair* the alternatives
  - Each alternative faces its opponent in a *pairwise election*
  - The winners proceed to the original rule
- Makes many rules hard to manipulate

Preround example (with Borda)

STEP 1:
A. Collect votes and
B. Match alternatives
(no order required)

Voter 1: A>B>C>D>E>F
Voter 2: D>E>F>A>B>C
Voter 3: F>D>B>E>C>A

Match A with B
Match C with F
Match D with E

STEP 2:
Determine winners of preround

A vs B: A ranked higher by 1,2
C vs F: F ranked higher by 2,3
D vs E: D ranked higher by all

STEP 3:
Infer votes on remaining alternatives

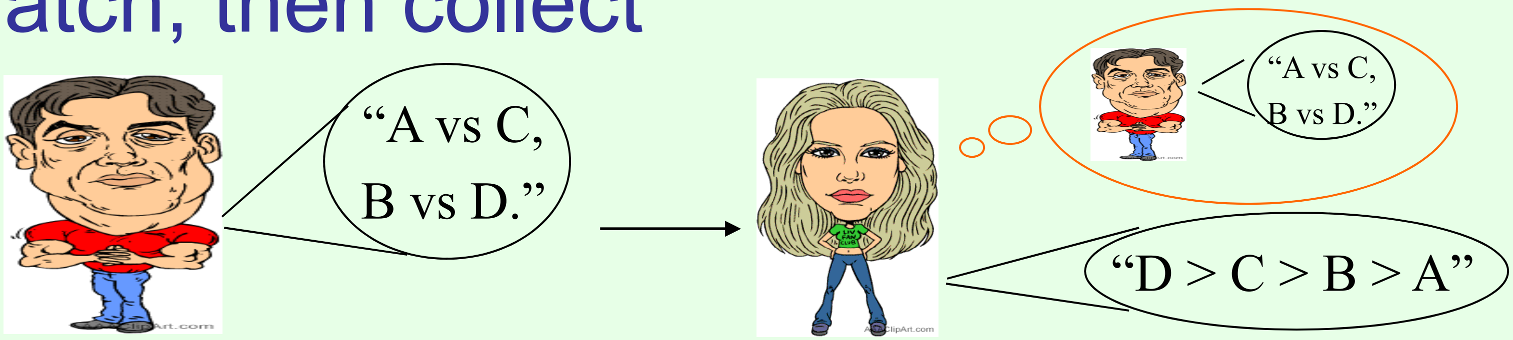
Voter 1: A>D>F
Voter 2: D>F>A
Voter 3: F>D>A

STEP 4:
Execute original rule
(Borda)

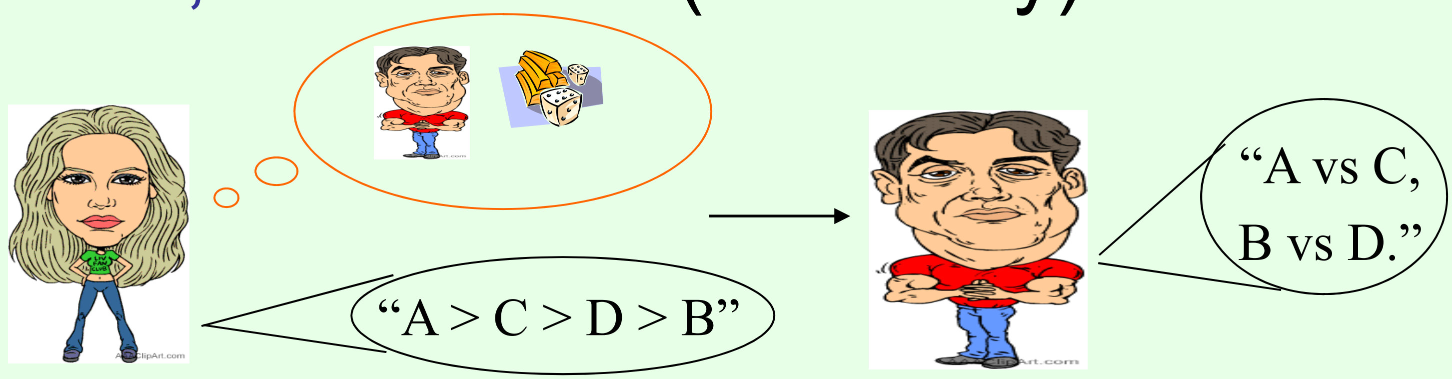
A gets 2 points
F gets 3 points
D gets 4 points and wins!

Matching first, or vote collection first?

- Match, then collect

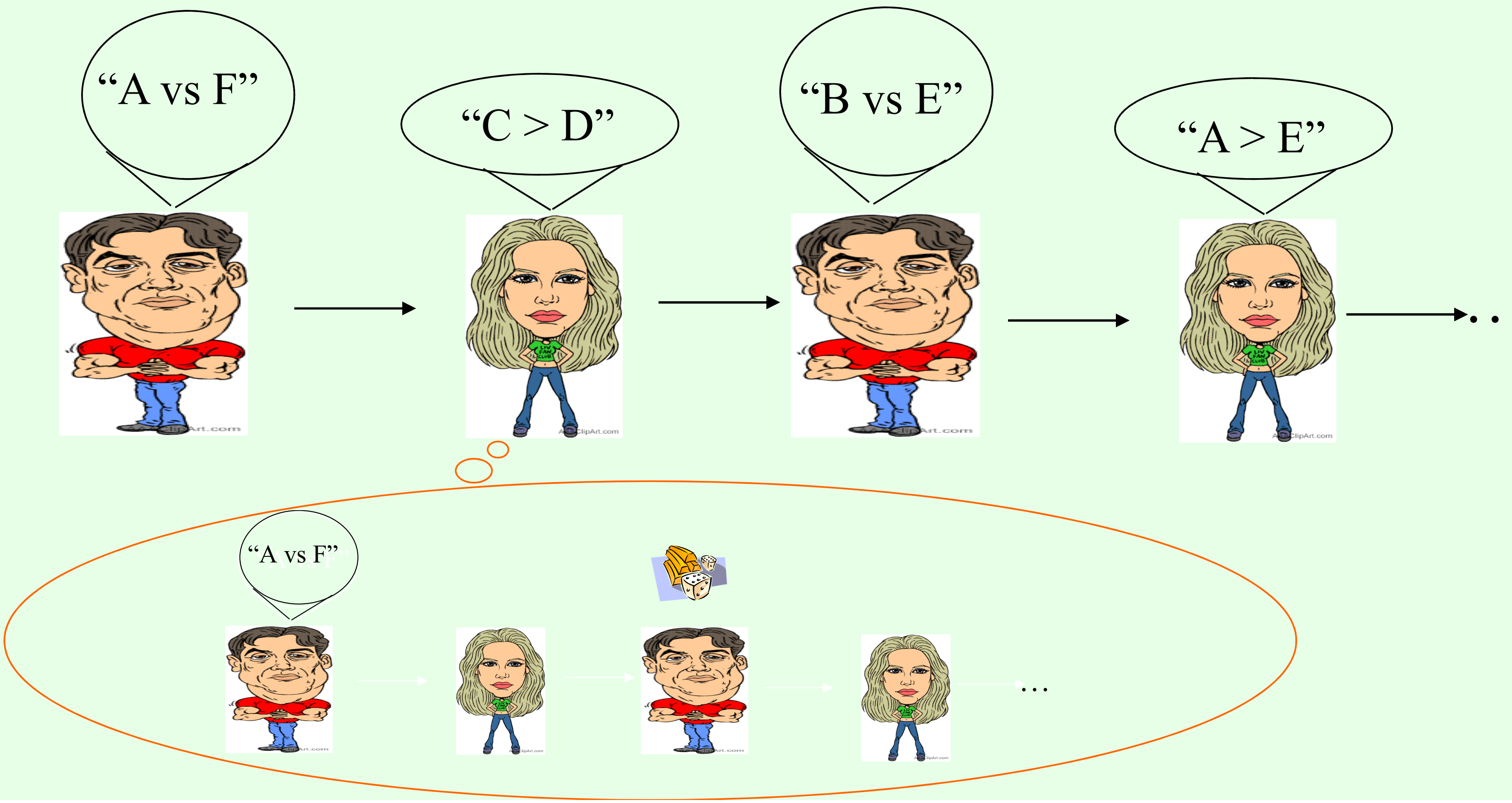


- Collect, then match (randomly)



Could also interleave…

- Elicitor alternates between:
  - (Randomly) announcing part of the matching
  - Eliciting part of each voter's vote



How hard is manipulation when a preround is added?

- Manipulation hardness differs depending on the order/interleaving of preround matching and vote collection:
- **Theorem.** NP-hard if preround matching is done first
- **Theorem.** #P-hard if vote collection is done first
- **Theorem.** PSPACE-hard if the two are interleaved (for a complicated interleaving protocol)
- In each case, the tweak introduces the hardness for *any* rule satisfying certain sufficient conditions
  - All of Plurality, Borda, Maximin, STV satisfy the conditions in all cases, so they are hard to manipulate with the preround

What if there are few alternatives? [Conitzer et al. JACM 2007]

- The previous results rely on the number of alternatives (m) being unbounded
- There is a recursive algorithm for manipulating STV with $O(1.62^m)$ calls (and usually much fewer)
- E.g., 20 alternatives: $1.62^{20} = 15500$

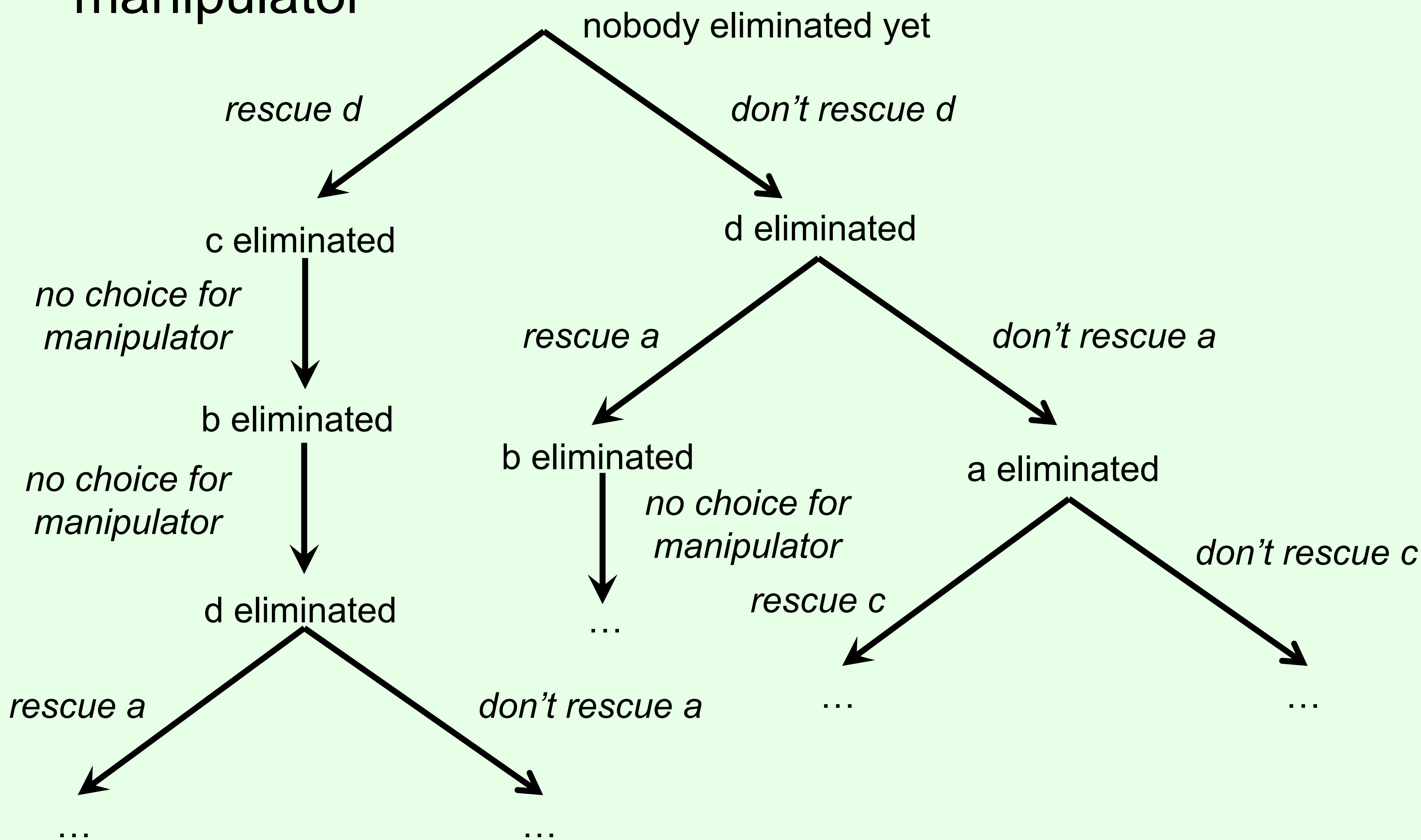
- Sometimes the alternative space is much larger
  - Voting over allocations of goods/tasks
  - California governor elections

- But what if it is not?
  - A typical election for a representative will only have a few

STV manipulation algorithm

[Conitzer et al. JACM 2007]

- Idea: simulate election under various actions for the manipulator



Analysis of algorithm

- Let $T(m)$ be the maximum number of recursive calls to the algorithm (nodes in the tree) for m alternatives
- Let $T'(m)$ be the maximum number of recursive calls to the algorithm (nodes in the tree) for m alternatives given that the manipulator's vote is currently committed
- $T(m) \leq 1 + T(m-1) + T'(m-1)$
- $T'(m) \leq 1 + T(m-1)$
- Combining the two: $T(m) \leq 2 + T(m-1) + T(m-2)$
- The solution is $O(((1+\sqrt{5})/2)^m)$
- Note this is only worst-case; in practice manipulator probably won't make a difference in most rounds
  - Walsh [CARE 2009] shows this algorithm is highly effective in experiments (simulation)

Manipulation complexity *with few alternatives*

- Ideally, would like hardness results for *constant* number of alternatives
- But then manipulator can simply evaluate each possible vote
  - assuming the others' votes are known & executing rule is in P
- Even for coalitions of manipulators, there are only polynomially many *effectively different* vote profiles (if rule is anonymous)
- However, if we place *weights* on votes, complexity may return…

Unbounded #alternatives

	Unweighted voters	Weighted voters
Individual manipulation	Can be hard	Can be hard
Coalitional manipulation	Can be hard	Can be hard

Constant #alternatives

	Unweighted voters	Weighted voters
Individual manipulation	easy	easy
Coalitional manipulation	easy	Potentially hard

Constructive manipulation now becomes:

- We are given the weighted votes of the others (with the weights)
- And we are given the weights of members of our coalition
- Can we make our preferred alternative *p* win?
- E.g., another Borda example:
- Voter 1 (weight 4): A>B>C, voter 2 (weight 7): B>A>C
- Manipulators: one with weight 4, one with weight 9
- Can we make C win?
- Yes! Solution: weight 4 voter votes C>B>A, weight 9 voter votes C>A>B
  - Borda scores: A: 24, B: 22, C: 26

A simple example of hardness

- We want: given the other voters' votes…
- … it is NP-hard to find votes for the manipulators to achieve their objective
- Simple example: veto rule, constructive manipulation, 3 alternatives
- Suppose, from the given votes, *p* has received 2K-1 more vetoes than *a*, and 2K-1 more than *b*
- The manipulators' combined weight is 4K
  - every manipulator has a weight that is a multiple of 2
- The only way for *p* to win is if the manipulators veto *a* with 2K weight, and *b* with 2K weight
- But this is doing PARTITION => NP-hard!

What does it mean for a rule to be *easy* to manipulate?

- Given the other voters' votes…
- …there is a polynomial-time algorithm to find votes for the manipulators to achieve their objective
- If the rule is computationally easy to run, then it is easy to check whether a given vector of votes for the manipulators is successful
- **Lemma:** Suppose the rule satisfies (for some number of alternatives):
  - If there is a successful manipulation…
  - … then there is a successful manipulation where all manipulators vote identically.
- Then the rule is easy to manipulate (for that number of alternatives)
  - Simply check all possible orderings of the alternatives (constant)

Example: Maximin with 3 alternatives is easy to manipulate constructively

- Recall: alternative's Maximin score = worst score in any pairwise election
- 3 alternatives: *p, a, b*. Manipulators want *p* to win
- Suppose there exists a vote vector for the manipulators that makes *p* win
- WLOG can assume that all manipulators rank p first
  - So, they either vote *p* > *a* > *b* or *p* > *b* > *a*
- Case I: *a*'s worst pairwise is against *b*, *b*'s worst against *a*
  - One of them would have a maximin score of at least half the vote weight, and win (or be tied for first) => cannot happen
- Case II: one of *a* and *b*'s worst pairwise is against *p*
  - Say it is *a*; then can have all the manipulators vote *p* > *a* > *b*
    - Will not affect *p* or *a*'s score, can only decrease *b*'s score

Results for *constructive* manipulation

Number of candidates	2	3	4,5,6	≥ 7
Borda	P	NP-c	NP-c	NP-c
veto	P	NP-c*	NP-c*	NP-c*
STV	P	NP-c	NP-c	NP-c
plurality with runoff	P	NP-c*	NP-c*	NP-c*
Copeland	P	P*	NP-c	NP-c
maximin	P	P*	NP-c	NP-c
randomized cup	P	P*	P*	NP-c
regular cup	P	P	P	P
plurality	P	P	P	P

Complexity of CONSTRUCTIVE CW-MANIPULATION

Destructive manipulation

- Exactly the same, except:
- Instead of a preferred alternative
- We now have a hated alternative
- Our goal is to make sure that the hated alternative does not win (whoever else wins)

Results for *destructive* manipulation

Number of candidates	2	≥ 3
STV	P	NP-c*
plurality with runoff	P	NP-c*
randomized cup	P	?
Borda	P	P
veto	P	P*
Copeland	P	P
maximin	P	P
regular cup	P	P
plurality	P	P

Complexity of DESTRUCTIVE CW-MANIPULATION

Hardness is only worst-case…

- Results such as NP-hardness suggest that the runtime of any successful manipulation algorithm is going to grow dramatically on some instances
- But there may be algorithms that solve most instances fast
- Can we make most manipulable instances hard to solve?

Bad news…

- Increasingly many results suggest that **many instances are in fact easy to manipulate**
- Heuristic algorithms and/or experimental (simulation) evaluation [Conitzer & Sandholm AAAI-06, Procaccia & Rosenschein JAIR-07, Conitzer et al. JACM-07, Walsh IJCAI-09 / CARE-09]
- Algorithms that only have a small "window of error" of instances on which they fail [Zuckerman et al. AIJ-09, Xia et al. EC-10]
- Results showing that whether the manipulators can make a difference depends primarily on their number
  - If n nonmanipulator votes drawn i.i.d., with high probability, $o(\sqrt{n})$ manipulators cannot make a difference, $\omega(\sqrt{n})$ can make any alternative win that the nonmanipulators are not systematically biased against [Procaccia & Rosenschein AAMAS-07, Xia & Conitzer EC-08a]
  - Border case of $\Theta(\sqrt{n})$ has been investigated [Walsh IJCAI-09]
- Quantitative versions of Gibbard-Satterthwaite showing that under certain conditions, for some voter, even a random manipulation on a random instance has significant probability of succeeding [Friedgut, Kalai, Nisan FOCS-08; Xia & Conitzer EC-08b; Dobzinski & Procaccia WINE-08, Isaksson et al. 09]

Weak monotonicity

voting rule · alternative set · nonmanipulator votes · nonmanipulator weights · manipulator weights

- An instance (R, C, v, k_v, k_w)

  is weakly monotone if for every pair of alternatives c_1, c_2 in C, one of the following two conditions holds:

- either: c_2 does not win for any manipulator votes w,
- or: if all manipulators rank c_2 first and c_1 last, then c_1 does not win.

A simple manipulation algorithm

[Conitzer & Sandholm AAAI 06]

Find-Two-Winners(R, C, v, k_v, k_w)

- choose arbitrary manipulator votes w_1
- $c_1 \leftarrow R(C, v, k_v, w_1, k_w)$
- for every c_2 in C, $c_2 \neq c_1$
  - choose w_2 in which every manipulator ranks c_2 first and c_1 last
  - $c \leftarrow R(C, v, k_v, w_2, k_w)$
  - if $c \neq c_1$ return $\{(w_1, c_1), (w_2, c)\}$
- return $\{(w_1, c_1)\}$

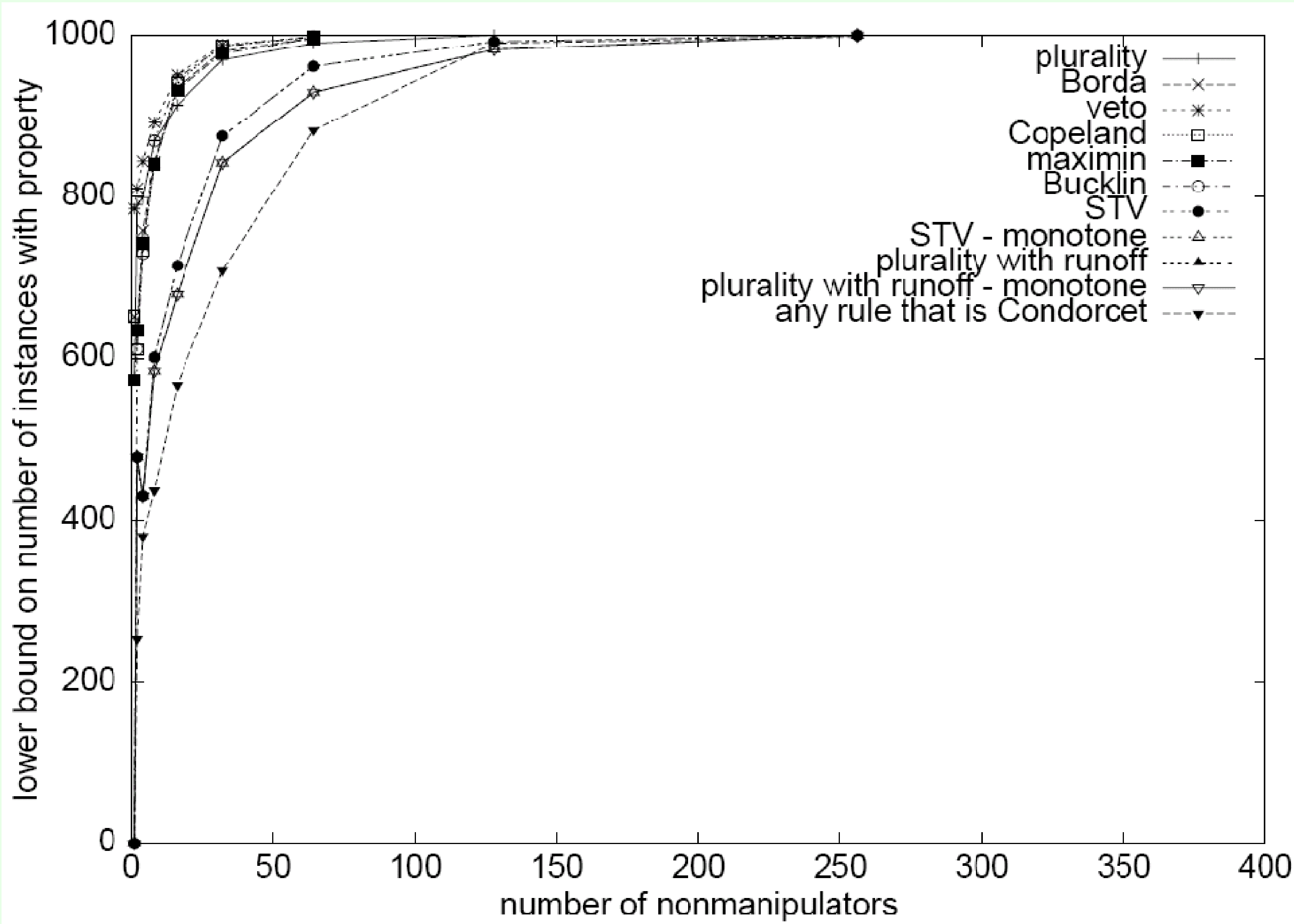
Correctness of the algorithm

- **Theorem.** Find-Two-Winners succeeds on every instance that
  - (a) is weakly monotone, and
  - (b) allows the manipulators to make either of exactly two alternatives win.
- **Proof.**
  - The algorithm is sound (never returns a wrong (w, c) pair).
  - By (b), all that remains to show is that it will return a second pair, that is, that it will terminate early.
  - Suppose it reaches the round where c_2 is the other alternative that can win.
  - If $c = c_1$ then by weak monotonicity (a), c_2 can never win (contradiction).
  - So the algorithm must terminate.

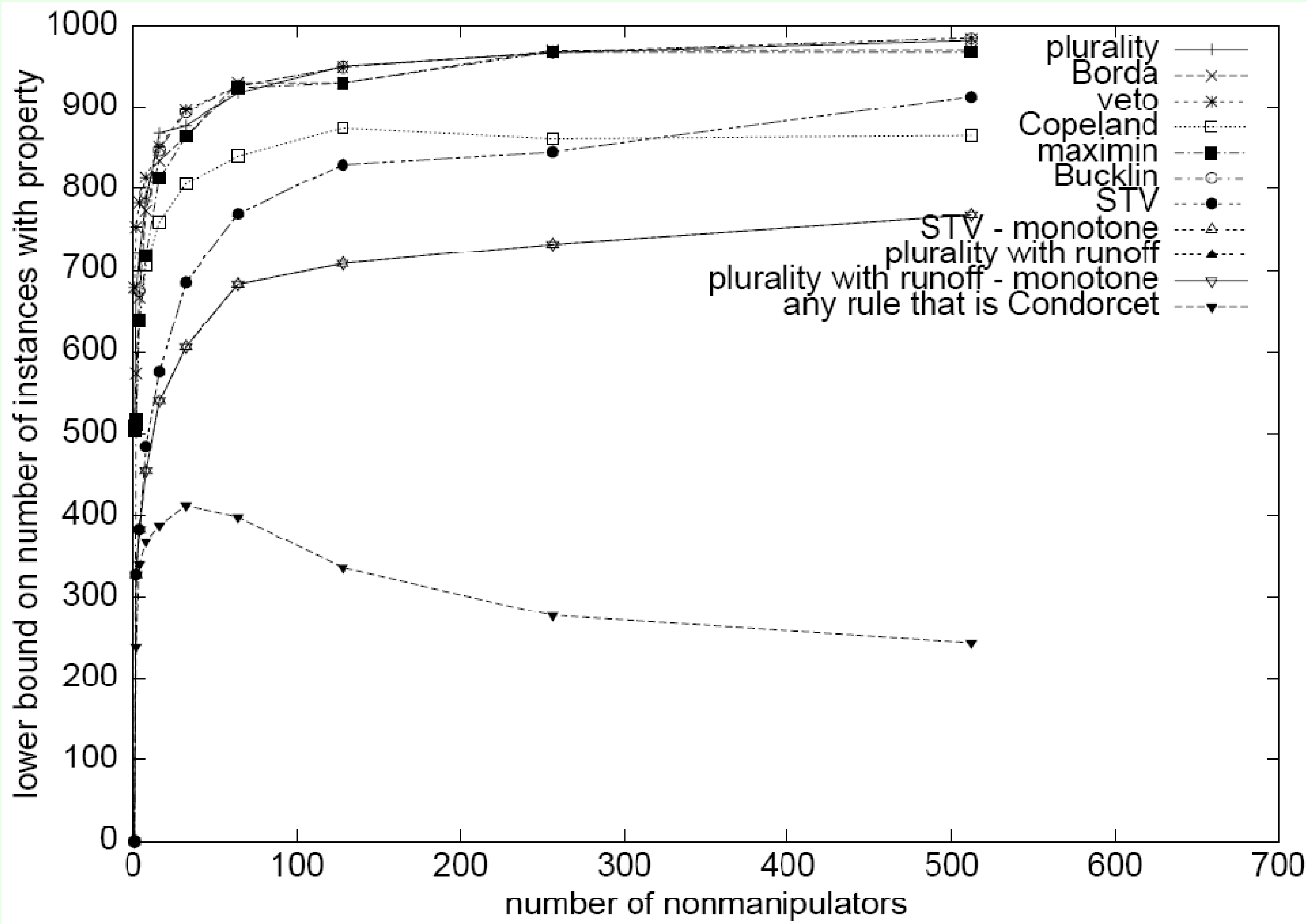
Experimental evaluation

- For what % of manipulable instances do properties (a) and (b) hold?
  - Depends on distribution over instances…
- Use Condorcet's distribution for nonmanipulator votes
  - There exists a correct ranking t of the alternatives
  - Roughly: a voter ranks a pair of alternatives correctly with probability p, incorrectly with probability $1-p$
    - Independently? This can cause cycles…
  - More precisely: a voter has a given ranking r with probability proportional to $p^{a(r, t)}(1-p)^{d(r, t)}$ where $a(r, t)$ = # pairs of alternatives on which r and t agree, and $d(r, t)$ = # pairs on which they disagree
- Manipulators all have weight 1
- Nonmanipulable instances are thrown away

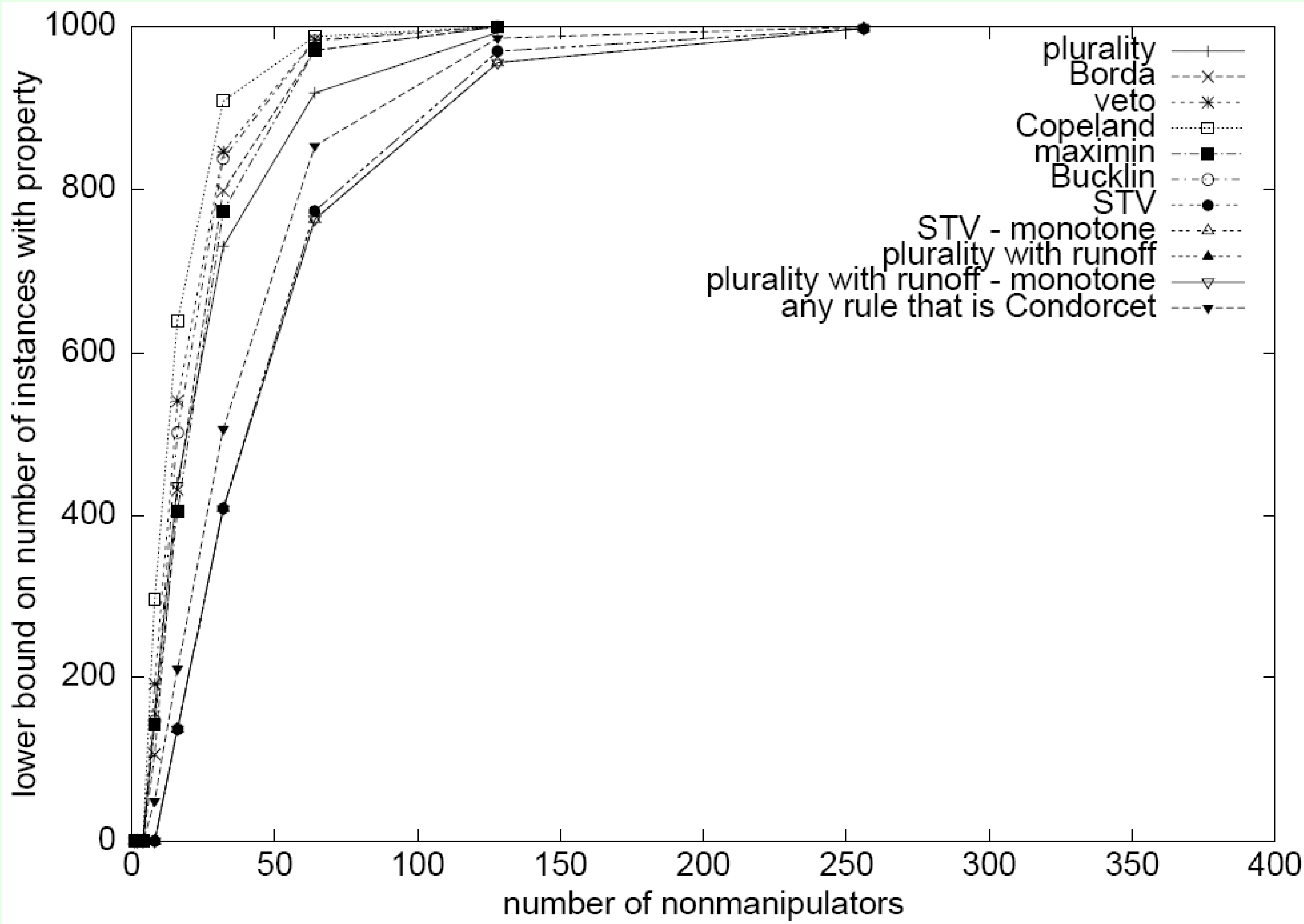
p=.6, one manipulator, 3 alternatives



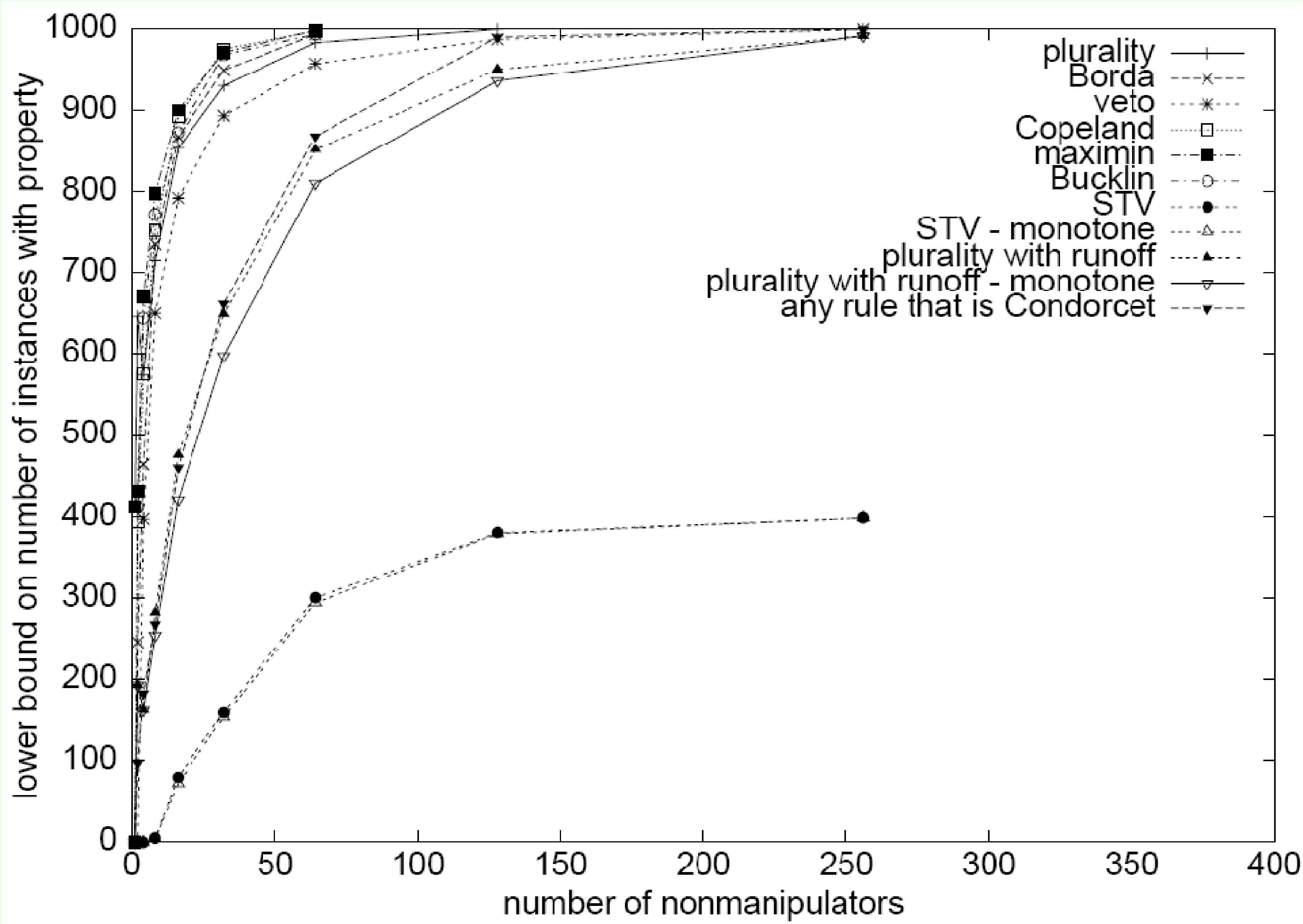
p=.5, one manipulator, 3 alternatives



p=.6, 5 manipulators, 3 alternatives



p=.6, one manipulator, 5 alternatives



Control problems [Bartholdi et al. 1992]

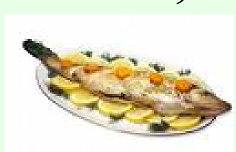
- Imagine that the chairperson of the election controls whether some alternatives participate
- Suppose there are 5 alternatives, a, b, c, d, e
- Chair controls whether c, d, e run (can choose any subset); chair wants b to win
- Rule is plurality; voters' preferences are:
- a > b > c > d > e (11 votes)
- b > a > c > d > e (10 votes)
- c > e > b > a > e (2 votes)
- d > b > a > c > e (2 votes)
- c > a > b > d > e (2 votes)
- e > a > b > c > e (2 votes)
- Can the chair make b win?
- NP-hard

many other types of control, e.g., introducing additional voters

see also various work by Faliszewksi, Hemaspaandra, Hemaspaandra, Rothe

Combinatorial alternative spaces

Multi-issue domains

- Suppose the set of alternatives can be uniquely characterized by multiple issues
- Let $I=\{x_1,...,x_p\}$ be the set of p issues
- Let D_i be the set of values that the i-th issue can take, then $A=D_1\times ... \times D_p$
- Example:
  - $I=\{$Main dish, Wine$\}$
  - $A=\{$   $\} \times \{$   $\}$

Example: joint plan

[Brams, Kilgour & Zwicker SCW 98]

- The citizens of LA county vote to directly determine a government plan
- Plan composed of multiple sub-plans for several issues
  - E.g.,

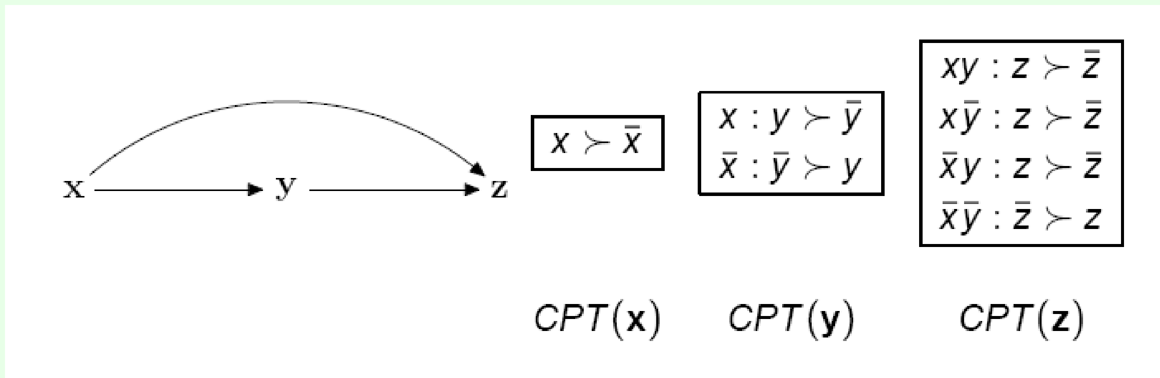
CP-net [Boutilier et al. UAI-99/JAIR-04]

- A compact representation for partial orders (preferences) on multi-issue domains
- An CP-net consists of
  - A set of variables $x_1,\ldots,x_p$, taking values on $D_1,\ldots,D_p$
  - A directed graph G over $x_1,\ldots,x_p$
  - Conditional preference tables (CPTs) indicating the conditional preferences over x_i, given the values of its parents in G

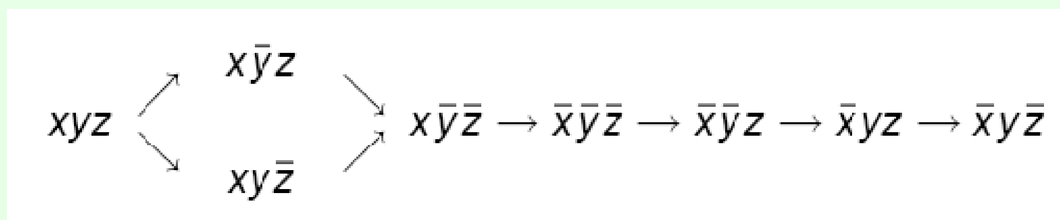
CP-net: an example

Variables: x, y, z. $\quad D_x = \{x, \overline{x}\}, D_y = \{y, \overline{y}\}, D_z = \{z, \overline{z}\}$.

DAG, CPTs:



This CP-net encodes the following partial order:



Sequential voting rules
[Lang IJCAI-07/Lang and Xia MSS-09]

- Inputs:
  - A set of issues $x_1,\ldots,x_p$, taking values on $A=D_1\times\ldots\times D_p$
  - A linear order O over the issues. W.l.o.g. $O=x_1>\ldots>x_p$
  - p local voting rules $r_1,\ldots,r_p$
  - A profile $P=(V_1,\ldots,V_n)$ of O-legal linear orders
    - *O*-legal means that preferences for each issue depend only on values of issues earlier in *O*
- **Basic idea**: use r_1 to decide x_1's value, then r_2 to decide x_2's value (conditioning on x_1's value), *etc.*
- Let $\mathrm{Seq}_O(r_1,\ldots,r_p)$ denote the sequential voting rule

Sequential rule: an example

- Issues: main dish, wine
- Order: main dish > wine
- Local rules are majority rules
- V_1:  >  ,  :  >  ,  :  > 
- V_2:  >  ,  :  >  ,  :  > 
- V_3:  >  ,  :  >  ,  :  > 
- **Step 1:** 
- **Step 2: given**  **,**  **is the winner for wine**
- **Winner: (**  **,**  **)**

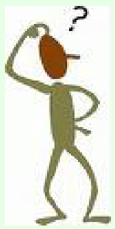
- Xia et al. [AAAI'08, AAMAS'10] study rules that do not require CP-nets to be acyclic

Strategic sequential voting

- Binary issues (two possible values each)
- Voters vote simultaneously on issues, one issue after another
- For each issue, the majority rule is used to determine the value of that issue
- Game-theoretic analysis?

Strategic voting in multi-issue domains

S **T**



$$V_1 : st > \bar{s}t > s\bar{t} > \bar{s}\bar{t}$$
$$V_2 : s\bar{t} > st > \bar{s}t > \bar{s}\bar{t}$$
$$V_3 : \bar{s}t > \bar{s}\bar{t} > s\bar{t} > st$$

- In the first stage, the voters vote simultaneously to determine **S**; then, in the second stage, the voters vote simultaneously to determine **T**
- If **S** is built, then in the second step $t > \bar{t}$, $\bar{t} > t$, $\bar{t} > t$ so the winner is $s\bar{t}$
- If **S** is **not** built, then in the 2nd step $t > \bar{t}$, $t > \bar{t}$, $t > \bar{t}$ so the winner is $\bar{s}t$
- In the first step, the voters are effectively comparing $s\bar{t}$ and $\bar{s}t$, so the votes are $\bar{s} > s$, $s > \bar{s}$, $\bar{s} > s$, and the final winner is $\bar{s}t$

[Xia et al. 2010; see also Farquharson 69, McKelvey & Niemi JET 78, Moulin Econometrica 79, Gretlein IJGT 83, Dutta & Sen SCW 93]

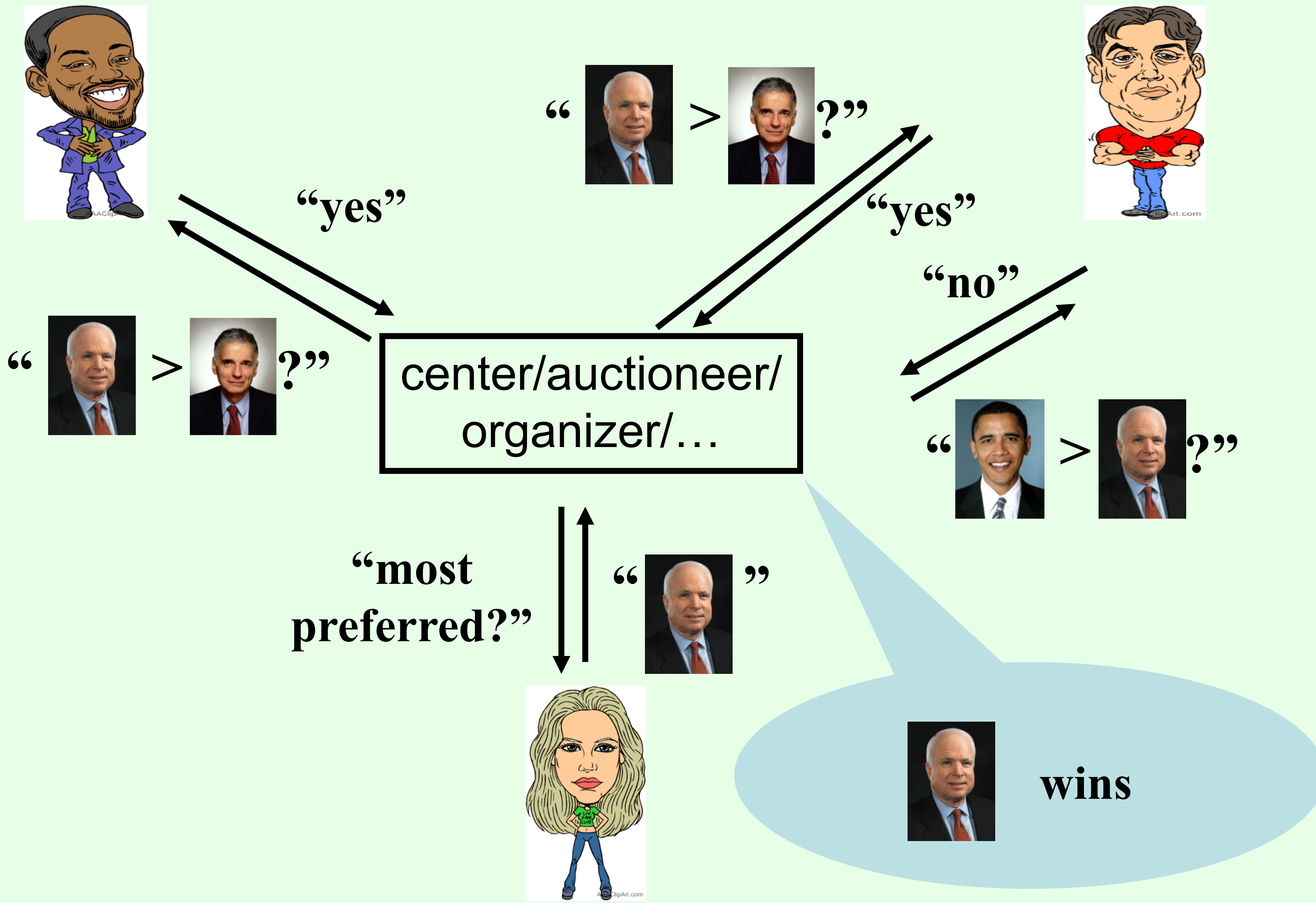
Multiple-election paradoxes for strategic voting [Xia et al. 2010]

- **Theorem** (informally). For any $p \geq 2$ and any $n \geq 2p^2 + 1$, there exists a profile such that the strategic winner is
  - ranked almost at the bottom (exponentially low positions) in every vote
  - Pareto dominated by almost every other alternative
  - an almost Condorcet loser

  - multiple-election paradoxes [Brams, Kilgour & Zwicker SCW 98], [Scarsini SCW 98], [Lacy & Niou JTP 00], [Saari & Sieberg 01 APSR], [Lang & Xia MSS 09]

Preference elicitation / communication complexity

Preference elicitation (elections)



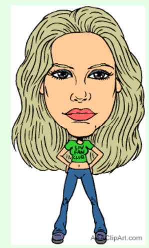
Elicitation algorithms

- Suppose agents always answer truthfully
- Design elicitation algorithm to minimize queries for given rule
- What is a good elicitation algorithm for STV?
- What about Bucklin?

An elicitation algorithm for the Bucklin voting rule based on binary search

[Conitzer & Sandholm EC'05]

- Alternatives: A B C D E F G H


- Top 4? {A B C D} {A B F G} {A C E H}
- Top 2? {A D} {B F} {C H}
- Top 3? {A C D} {B F G} {C E H}

Total communication is nm + nm/2 + nm/4 + … ≤ 2nm bits
(n number of voters, m number of candidates)

Getting involved in this community

- Community mailing list

https://lists.duke.edu/sympa/subscribe/comsoc

- Computational Social Choice (COMSOC) workshop (deadline May 15…)

http://ccc.cs.uni-duesseldorf.de/COMSOC-2010/

A few useful overviews

- Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. A Short Introduction to Computational Social Choice. In *Proc. 33rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM-2007), LNCS 4362, Springer-Verlag, 2007.*
- V. Conitzer. Making decisions based on the preferences of multiple agents. *Communications of the ACM, 53(3):84–94, 2010.*
- V. Conitzer. Comparing Multiagent Systems Research in Combinatorial Auctions and Voting. To appear in the *Annals of Mathematics and Artificial Intelligence*.
- P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. A richer understanding of the complexity of election systems. In S. Ravi and S. Shukla, editors, *Fundamental Problems in Computing: Essays in Honor of Professor Daniel J. Rosenkrantz,* chapter 14, pages 375–406. Springer, 2009.
- P. Faliszewski and A. Procaccia. AI's War on Manipulation: Are We Winning? To appear in *AI Magazine*.
- L. Xia. Computational Social Choice: Strategic and Combinatorial Aspects. *AAAI'10 Doctoral Consortium.*