COMPUTATIONAL SOCIAL CHOICE

Maria Silvia Pini
Francesca Rossi
K. Brent Venable

PhD course in Computer Science
University of Bologna & University of Padova
June 2012

Computational Social Choice

- It is an **interdisciplinary field** at the interface of
  - social choice theory
  - computer science and AI

- Main goals
  1. Application of techniques of computer science, such as complexity analysis or algorithm design, to the study of social choice mechanisms, such as voting procedures
  2. Importing concepts from social choice theory into computing. For instance, the study of preference aggregation mechanisms is relevant to multiagent systems

Chevaleyre, Endriss, Lang, Maudet, 2007
A short introduction to Computational Social Choice

(Computational) Social Choice

- Voting procedures
- Impossibility results
- Manipulation

Social choice problems

- Circumventing manipulation
- Uncertainty
- Voting in combinatorial domains

Computational techniques

Outline

- Impossibility results
- Attempts to modify the winner
  - Manipulation
  - Control
  - Bribery
- Complexity barrier against manipulation
- Uncertainty in preference aggregation
  - Preference aggregation with incompleteness and incomparability
  - Voting tree
- Related work

Impossibility results

Which rule?

- Since there are so many rules, which one should we choose?

- Let us look at some criteria that we would like our voting rule to satisfy

Monotonicity criteria (1)

- 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

Monotonicity criteria (2)

- 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 weak monotonicity
  - 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 (3)

- 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

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.

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

Nobel prize
in Economics 1972

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)

Manipulation

Manipulability

- Sometimes, a voter is better off revealing her preferences insincerely, aka. manipulating

- Example for 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

Allan Gibbard

Mark Satterthwaite

Gibbard-Satterthwaite impossibility theorem

- Suppose there are at least 3 candidates
- If f is onto and nonmanipulable

  Then is dictatorial

Proof

- Step 1: If f is onto and nonmanipulable

  Then f is monotone
- Step 2: If f is onto, nonmanipulable, and monotone

  Then f is unanimous
- Step 3: If f is monotone and unanimous

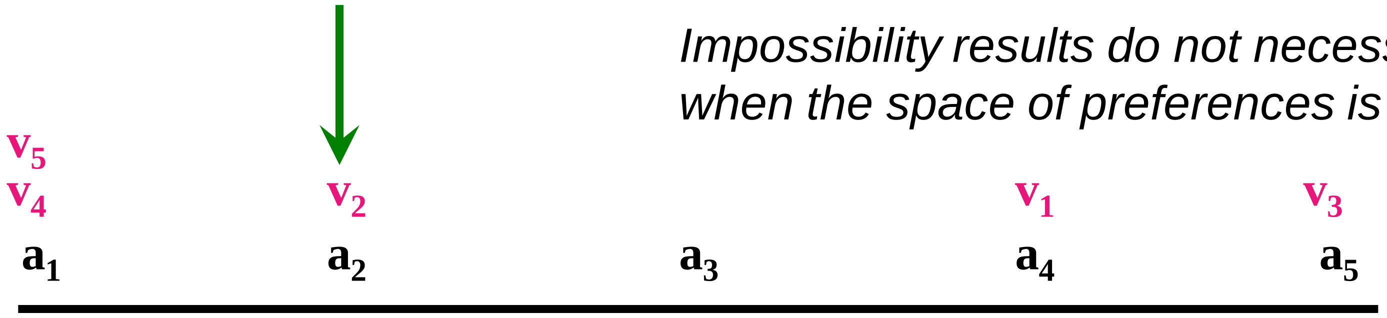
  Then f is dictatorial (Muller-Satterthwaite theorem)
- Step 4: If f is onto and nonmanipulable

  Then (by steps 1, 2, 3) f is dictatorial

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



Constructive/destructive manipulation

- Two kinds of manipulation
  - Constructive maniplation
    - Goal: to make a certain candidate win
  - Destructive manipularion
    - Goal: to make a certain candidate a loser

Conitzer, Sandholm, and Lang. When are elections with few candidates hard to manipulate, J. ACM, 2007

Constructive manipulation

- The simplest version of the manipulation problem:

- CONSTRUCTIVE-MANIPULATION:
  - We are given a voting rule *r*
  - the (unweighted) votes of the other voters
  - an alternative *p*
  - We are asked if we can cast **our (single) vote** to make *p* win.

Constructive manipulation

- **Example 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
    - A: 4
    - B: 3
    - C: 2
  - **A is the winner**
  - **Can we make B win by adding my vote?**
  - Answer: YES.
    - My vote: B > C > A (Borda scores: A: 4, B: 5, C: 3)

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)

Destructive manipulation

- DESTRUCTIVE-MANIPULATION:
  - We are given a voting rule *r*
  - the (unweighted) votes of the other voters
  - an alternative *p*
  - We are asked if we can cast **our (single) vote** to make *p* a loser.

Coalitions

- It will rarely be the case that a single voter can make a difference. So we should look into **manipulation by a coalition** of voters.

- New problems
  - Coalitional constructive manipulation
  - Coalitional destructive manipulation

Constructive coalitional manipulation

- CONSTRUCTIVE-COALITIONAL-MANIPULATION:
  - We are given a voting rule *r*
  - a set S of votes (the nonmanipulators votes)
  - a set T of votes that area still open (the manipolator votes)
  - an alternative *p*
  - We are asked if we can cast **votes in T so that p wins**

Destructive coalitional manipulation

- DESTRUCTIVE-COALITIONAL-MANIPULATION:
  - We are given a voting rule *r*
  - a set S of votes (the nonmanipulators votes)
  - a set T of votes that area still open (the manipolator votes)
  - an alternative *p*
  - We are asked if we can cast **votes in T so that p does not win**

Weighted voters

- Variants of the problem
  - Voters may be **weighted**
  - Examples:
    - countries in the EU;
    - shareholders of a company
- New problems
  - Weighted constructive (coalitional) manipulation
  - Weighted destructive (coalitional) manipulation

Control

The control problem

- The control problem refers to situations where a chair seeks to change the outcome of an election
  - by adding/deleting voters
  - by partitioning voters
  - by adding/deleting candidates

- Assumptions:
  - the chair knows all the voters' preferences and
  - all votes are cast simultaneously

Bartholdi, Tovey, and Trick. How hard is it to control an election? *Math. And Computer Modeling, 1992.*

Constructive/destructive control

- Constructive control
  - It refers to situations where a chair seeks to make a certain outcome the winner of an election

- Destructive control
  - It refers to situations where a chair seeks to make a certain outcome a loser of an election

Control by deleting voters

- Let E be a rule
- Constructive control by deleting voters
  - we are given
    - an election (C,V)
    - a distinguished candidate $c \in C$
    - a nonnegative integer $k \leq ||V||$
  - we ask whether we can delete at most k voters from V such that c is an E winner of the resulting election

Control by adding voters

- Let E be a rule
- Constructive control by adding voters
  - we are given
    - a candidate set C
    - a list V of registered voters with preferences over C
    - a list V′ of as yet unregistered voters with preferences over C
    - a distinguished candidate $c \in C$
    - a nonnegative integer $k \leq ||V'||$
  - the question is whether we can add to V at most k voters from V′ such that c is an E winner of the resulting election

Control by partitioning voters

- Let E be a rule
- Constructive control by adding voters
  - we are given
    - An election (C,V)
    - a distinguished candidate $c \in C$
  - we ask whether V can be partitioned into two sublists, V1 and V2, such that c is the unique winner of the two-stage election in which the winners of the two first-stage subelections (C,V1) and (C,V2) runs against each other in the final stage

Control by adding candidates

- Let E be a rule
- Constructive control by adding candidates
  - we are given
    - a candidate set $C \cup D$ with $C \cap D = \emptyset$
      - C is the set of originally qualified candidates
      - D is the set of spoiler candidates that may be added
    - a list V of registered voters with preferences over C
    - a distinguished candidate $c \in C$
    - a nonnegative integer k
- The question is whether we can add to C at most k candidates from D such that c is an E winner of the resulting election

Control by deleting candidates

- Let E be a rule
- Constructive control by deleting candidates
  - we are given
    - a candidate set C
    - a list V of registered voters with preferences over C
    - a distinguished candidate $c \in C$
    - a nonnegative integer k
- The question is whether we can remove from C at most k candidates such that c is an E winner of the resulting election

Example of control

- 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)

Outline

- Impossibility results
- Attempts to modify the winner
  - Manipulation
  - Control
  - Bribery
- Complexity barrier against manipulation
- Uncertainty in preference aggregation
  - Preference aggregation with incompleteness and incomparability
  - Voting tree
- Related work

Bribery

The bribery problem

- In bribery
  - there is an external agent who wishes to change the outcome of the election
  - To do this, he offers payments (within a budget) to voters for changing the preference orders to his liking

Bribery

- Let *R be a voting rule*
- *R-BRIBERY* problem
  - we are given
    - an election *E = (C, V)*
    - *a designated* candidate *p in C*
    - *a natural number B*
  - we ask if it is possible to ensure that *p is an R-winner* of *E through* ***changing the votes of at most B*** **voters.**

$-Bribery

- In *R-BRIBERY, effectively,* ***each voter has the same*** **unit cost:** We only care about bribing as few voters as possible
- **However, in many settings, the voters might have different prices**, depending, for example,
  - on how much a particular voter cares about the result of the election or
  - on the nature of the bribery

- *R-$BRIBERY where* ***each voter v has a price*** $\boldsymbol{p_v}$ **for changing his vote** (after we pay *v the* p_v *units,* we obtain full control over *v's vote)*

Swap-Bribery

- R-SWAP-BRIBERY
  - each voter v has a cost function p_v such that for each two candidates c, c’**, p_v (c, c’) is the cost of swapping c and c’ on v’s preference list** (provided c and c’ are ranked next to each other).
  - For example
    - a voter might be willing to swap his two least favorite candidates at a small cost
    - but he would never— irrespective of the payment — change the top-ranked candidate
  - The goal of the briber is to find a sequence of adjacent swaps
    - that leads to his or her **preferred candidate’s victory**, and
    - that **has lowest cost**

Manipulation and \$Bribery

- **Manipulation is a special case of** \$BRIBERY
  - the manipulation problem is a bribery problem where
    - the prices of manipulators are very low
    - the prices of nonmanipulators are very high
    - our budget allows us to buy the votes of all the manipulators but none of the nonmanipulators

Uncertainty about votes

Possible and necessary winners

- Setting: some (parts of) votes are missing
- Possible winner
  - There is a way for remaining votes to be cast so that he win
- Necessary winner
  - However remaining preferences are cast, he must win

Konzak and Lang. Voting Procedures with Incomplete Preferences. IJCAI workshop 2005

Pini, Rossi, Venable, Walsh. Incompleteness and Incomparability in Preference Aggregation. IJCAI 2007

Preference elicitation and the possible/necessary winner problem

- Preference elicitation
  - Some preferences may be missing
  - Time consuming, costly, difficult, …
  - **Want to terminate elicitation as soon as winner fixed**

- Closely connected to preference elicitation
  - **Elicitation can only be terminated iff possible winner set = necessary winner set**

Manipulation and the possible winner problem

- **Manipulation is a special case** of the possible winner problem, where
  - the nonmanipulators have **fully specified** preference orders
  - the manipulators have **completely unspecifed** preference orders

Complexity barrier against manipulation

The complexity shield (1)

- The Gibbard-Satterthwaite Theorem shows that strategic manipulation can never be rule out

- Idea: So it is always possible to manipulate; but may it may also difficult?
- Tools from complexity theory can make this idea precise

- Let F be a voting rule, if manipulation is computationally intractable for F, then F might be considered resistant to manipulation

The complexity shield (2)

- **Standard procedures** turn out to be **easy to manipulate**
- It might still be possible to design new ones that are resistant
- This approach is most interesting for voting procedures for which winner determination is tractable

Manipulability as a decision problem

- F: voting rule
- **Manipulability(F)**
  - **Instance:** Set of votes for all **except one voter**; alternative x
  - **Question:** Is there a vote for the final voter such that x wins?

If this can be answered in polynomial time, then F is easy to manipulate

Manipulability complexity

- If Manipulability(F) is computationally intractable, then manipulability may be considered less of a worry for procedure F

- Remark: We assume that the manipulator knows all the other votes
  - This **unrealistic assumption** is reasonable for intractability results
  - If manipulation is intractable even under such favorable conditions, then all the better
  - For tractability results, one can assume to have polls

Plurality is easy to manipulate

- **TH: Manipulability(Plurality)ϵ** P
- Proof
- Simply **vote for x**, the alternative to be made winner by means of manipulation. If manipulation is possible at all, this will work. Otherwise not.

- **General: Manipulability(F) ϵ P for any rule F with**
  - **polynomial winner determination problem and**
  - **polynomial number of votes**

Bartholdi,Tovey,Trick. The Computational Difficulty of Manipulating an Election. *Social Choice and Welfare 6(3): 227–241, 1989.*

Borda is easy to manipulate

- **MANIPULABILITY(Borda) ϵ P**
- **Proof**
  - **Place x** (the alternative to be made winner through manipulation) **at the top of your vote**
  - Then inductively proceed as follows: Check if any of the remaining **alternatives** can be put next on the ballot **without preventing x from winning**. If yes, do so. (If no, manipulation is impossible.)

Bartholdi,Tovey,Trick. The Computational Difficulty of Manipulating an Election. *Social Choice and Welfare 6(3): 227–241.*]

Algorithm Greedy-Manipulation

- **Input**
  - preferences of all other voters
  - a distinguished candidate c
- **Output**
  - a preference order that, together with those of all the other voters, will ensure that c is a winner, or
  - a claim that no such preference order exists,
- **Initialization:** Place c at the top of the preference order.
- **Iterative Step**. Determine whether any candidate can be placed in the next lower position (independent of other choices) without preventing c from winning
  - If so, place such a candidate in the next position
  - otherwise terminate claiming that c cannot win

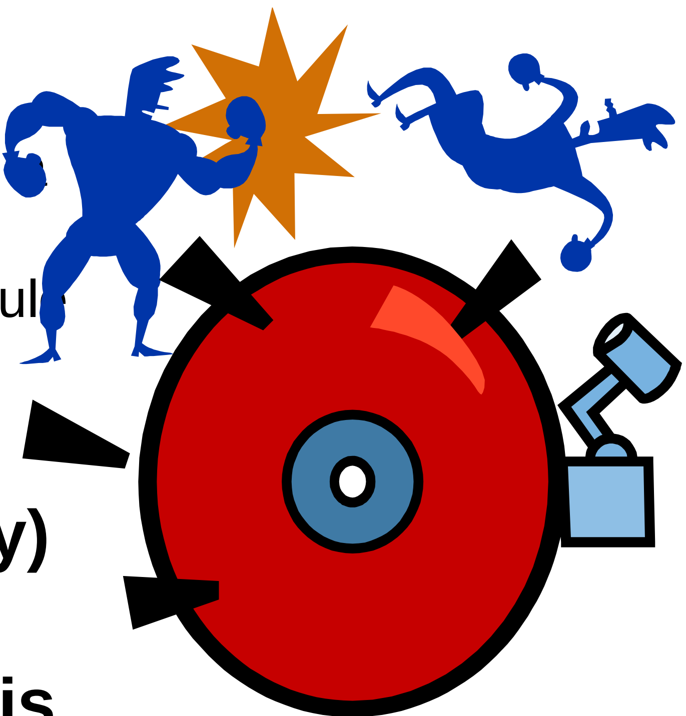
STV is difficult to manipulate

- **MANIPULABILITY(STV) ϵ NP-complete**
- Proof
- **NP-membership** is clear: **checking** whether **a given vote** makes x win can be **done in polynomial time** (just try it, STV is polynomial to compute)
- **NP-hardness: by reduction from 3-Cover (X3C)**
  - In an X3C instance
    - we are given
      - a set
      - a collection of subsets of size 3 of this set
    - we are asked if we can cover all of the elements in the set with nonoverlapping subsets

Bartholdi, J., and Orlin, J. Single Transferable Vote Resists Strategic Voting. *Social Choice and Welfare 1991*

Adding a preround

- A preround proceeds as follows:
  - *Pair* the candidates
  - Each candidate faces its opponent in a *pairwise knockout election*
  - The winners proceed to the original rule
- **P-R**: voting rule obtained running first a preround and the rule R
- **TH: Manipulability(P-Plurality) is NP-complete.**
- **TH: Manipulability(P-Borda) is NP-complete.**
- Also holds for other rules



Conitzer, Sandholm. Universal Voting Protocol Tweaks to Make Manipulation Hard. In Proc. IJCAI 2003

Preround example (with Borda)

STEP 1:

A. Collect votes and

B. Match alternatives

(no order required)

STEP 2:

Determine winners of preround

STEP 3:

Infer votes on remaining alternatives

STEP 4:

Execute original rule

(Borda)

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

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

Voter 1: A>D>F

Voter 2: D>F>A

Voter 3: F>D>A

A gets 2 points

F gets 3 points

D gets 4 points and thus it wins

Coalitions and weights

- Manipulation can be done by
  - a single voter
  - a coalition of voters
- It will rarely be the case that a single voter can make a difference. So we should look into manipulation by a coalition of voters

- Manipulation can be done by
  - Weighted voters
  - Unweighted voters

- Manipulation may be
  - constructive: making alternative x a unique or tied winner
  - destructive: ensuring x does not win

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 theorem
  - It 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?

Inevitability of manipulability

- 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
  - nonmanipulable
- Typically don't want a rule that is dictatorial or not onto

(Coalitional) Manipulation with weighted/unweighted votes

Do voting rules exist for which manipulations are computationally hard to find?

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

- 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

Constructive weighted manipulation

- 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?

Constructive weighted manipulation

Borda example

- Voters
  - 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

Veto is NP-hard to manipulate with 3 or more candidates

- **TH: WEIGHTED-COALITIONAL-CONSTRUCTIVE-MANIPULABILITY(Veto) is NP-complete with 3 or more candidates.**
- **Proof**
- In NP since we can just give the manipulation
- To show NP-hardness, we give a simple reduction of **PARTITION**
  - Given m integers k_i with sum 2K, is there a partition with sum K?
  - Reduce to manipulate election so *p* wins against *a* or *b*
- Assume *one agent with weight 2K-1 has vetoed p*
- Each of the votes of the m manipulators has weight $2k_i$
  - their combined weight is 4K
- **The only way for *p* to win** is if the manipulators can veto a with 2K weight, and b with 2K weight
- But this solves the PARTITION problem

Weighted-coalitional 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

Conitzer, Sandholm, and Lang. When are elections with few candidates hard to manipulate, J. ACM, 2007

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)

Weighted-coalitional 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

Conitzer, Sandholm, and Lang. When are elections with few candidates hard to manipulate, J. ACM, 2007

Weighted-coalitional 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

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

Destructive manipulation with weighted votes

- If constructive manipulation is easy then destructive manipulation is easy
- Destructive manipulation can be easy even though constructive manipulation is hard
  - E.g. Borda is
    - Polynomial to manipulate destructively
    - NP-hard to manipulate constructively for 3 or more candidates for a weighted coalition

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**

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]

Uncertainty in Preference aggregation

Outline

- Uncertainty in preference aggregation
  - Preference aggregation with incompleteness and incomparability
    - Incompleteness: missing preferences
    - Incomparability: incomparable pairs
  - Preference aggregation in voting trees
    - Simple voting trees
    - Voting trees

Preference aggregation with incompleteness and incomparability

Motivation - (1)

- How to combine preferences of multiple agents in presence of incompleteness and incomparability in their preference orderings over a set of outcomes?
- Incompleteness: absence of knowledge on relationship between pairs of outcomes
  - ongoing preference elicitation
  - agents' privacy
- Incomparability: some elements cannot be compared
  - novel incomparable to a biography
  - fast expensive car incomparable to slow cheap car

Motivation - (2)

- Goal: aggregate the agents' preferences into a single pref. ordering
- Since there are incomplete preferences, we focus on computing:
  - **Possible winners** (PW):

    outcomes that can be the most preferred ones for the agents
  - **Necessary winners** (NW):

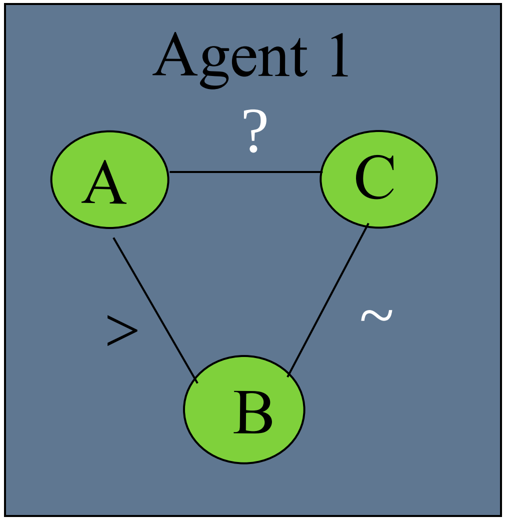
    outcomes that are always the most preferred ones for the agents
- Useful for preference elicitation

Outline

- Basic notions on preferences
- Possible and necessary winners
- Computing PW and NW: *NP-hard*
- Approximating PW and NW: *NP-hard*
- Sufficient conditions on preference aggregation such that computing PW and NW is *polynomial*
- How PW and NW are useful in preference elicitation

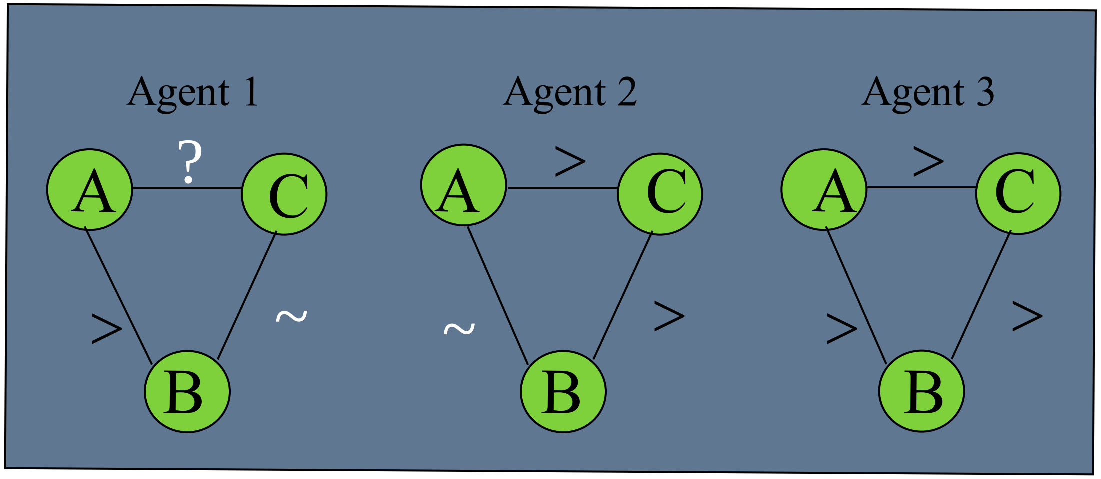
Basic notions - (1)

- Multi-agent scenario: each agent expresses his preferences via an (incomplete) partial ordering over the possible outcomes
  - preferences over outcomes A and B
    - A>B or A<B (ordered)
    - A=B (in a tie)
    - A~B (incomparable)
    - A?B (not specified)
  - Example: A,B,C outcomes



Basic notions - (2)

- Incomplete profile: *sequence* of partial orders over outcomes, one for every agent, where at least one partial order is incomplete

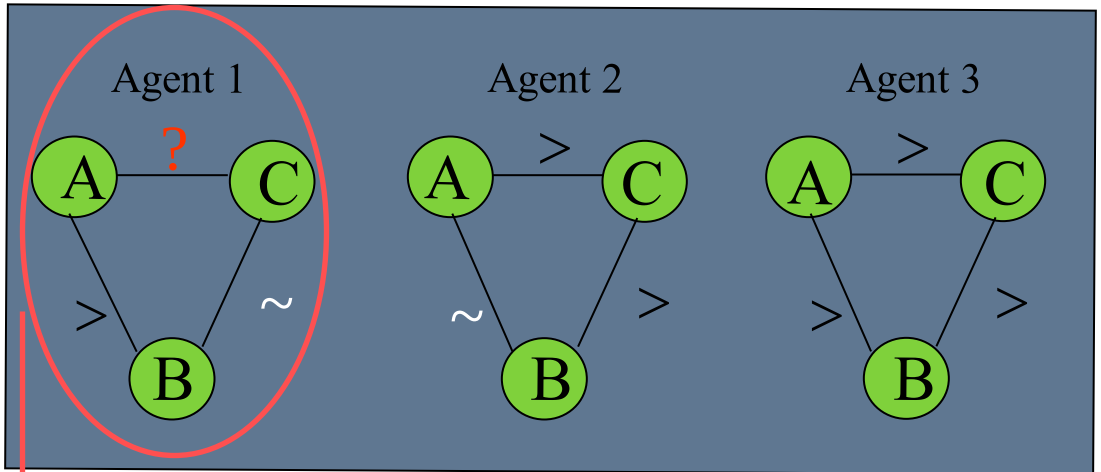


- Preference aggregation function: incomplete profiles $\rightarrow$ sets of P0s

We will consider only functions that take polynomial time to apply

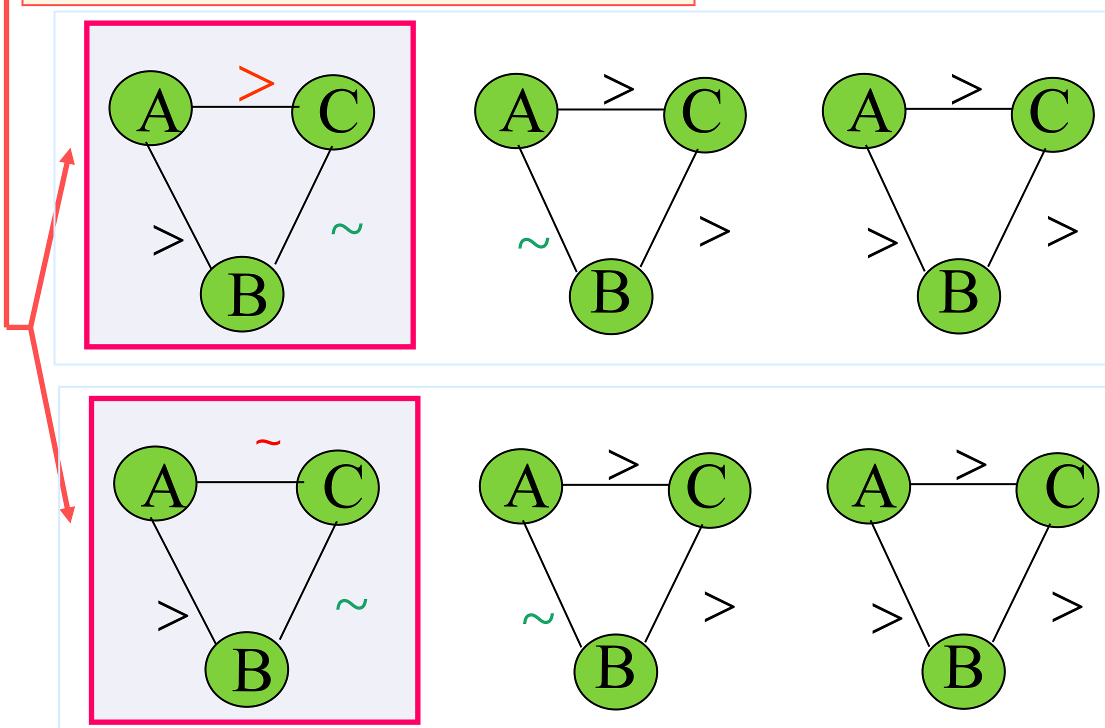
Pref. aggr. function:

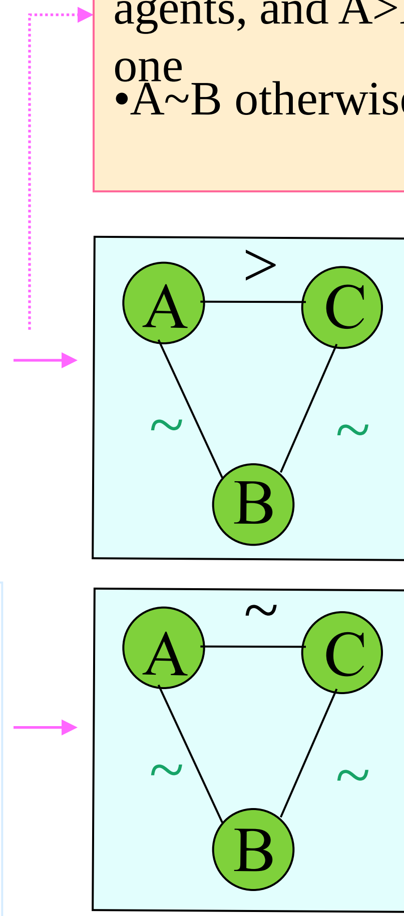
incomplete profiles → sets of P0s

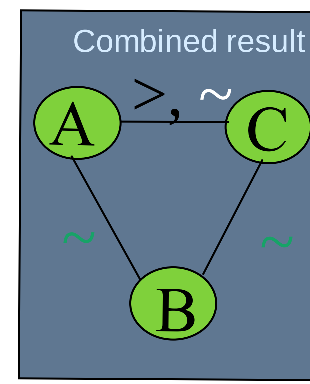


Pareto: POs → PO
- A>B iff A>B or A=B for all agents, and A>B for at least one
- A~B otherwise

only completions that are POs!

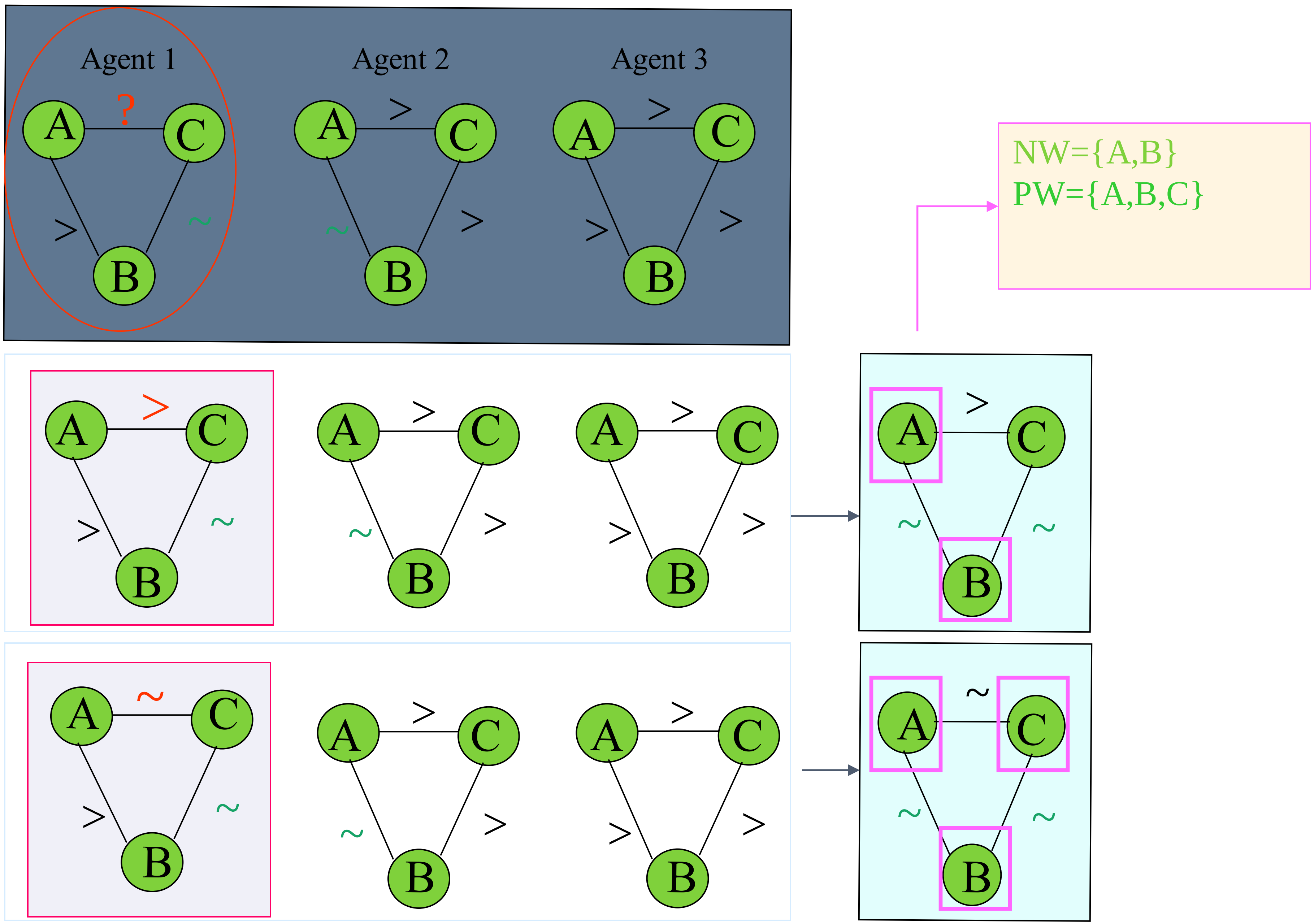






Possible and necessary winners

- We extend notions of PW and NW to POs
- Necessary winners
  - outcomes which are maximal in every completion
    - winners no matter how incompleteness is resolved
- Possible winners
  - outcomes which are maximal in at least one completion
    - winners in at least one way in which incompleteness is resolved



Agent 1
Agent 2
Agent 3
NW={A,B}
PW={A,B,C}

PW and NW: complexity results

- Computing PW and NW is NP-hard
  (even restricting to incomplete TOs)
  - deciding if an outcome is
    - a possible winner: NP-complete
    - a necessary winner: coNP-complete
- Computing *good approximations* of PW and NW is NP-hard
  - good approximation: for all k positive integer
    - a superset PW* s.t. |PW*| < k |PW|
    - a subset NW* s.t. . |NW*| >1/k |NW|, whenever |NW|>0

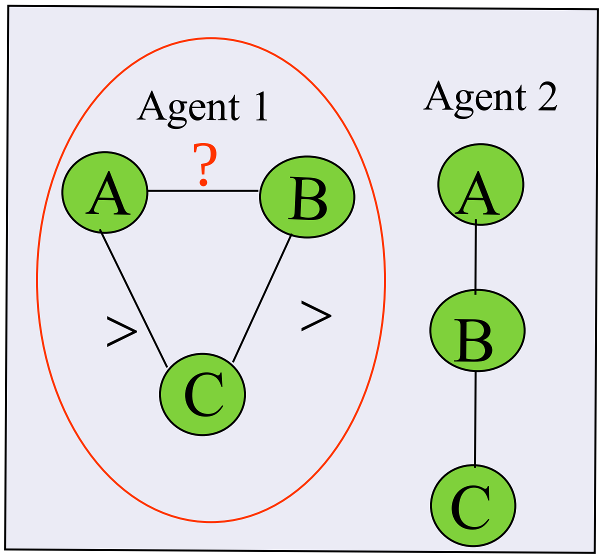
PW and NW: tractable case

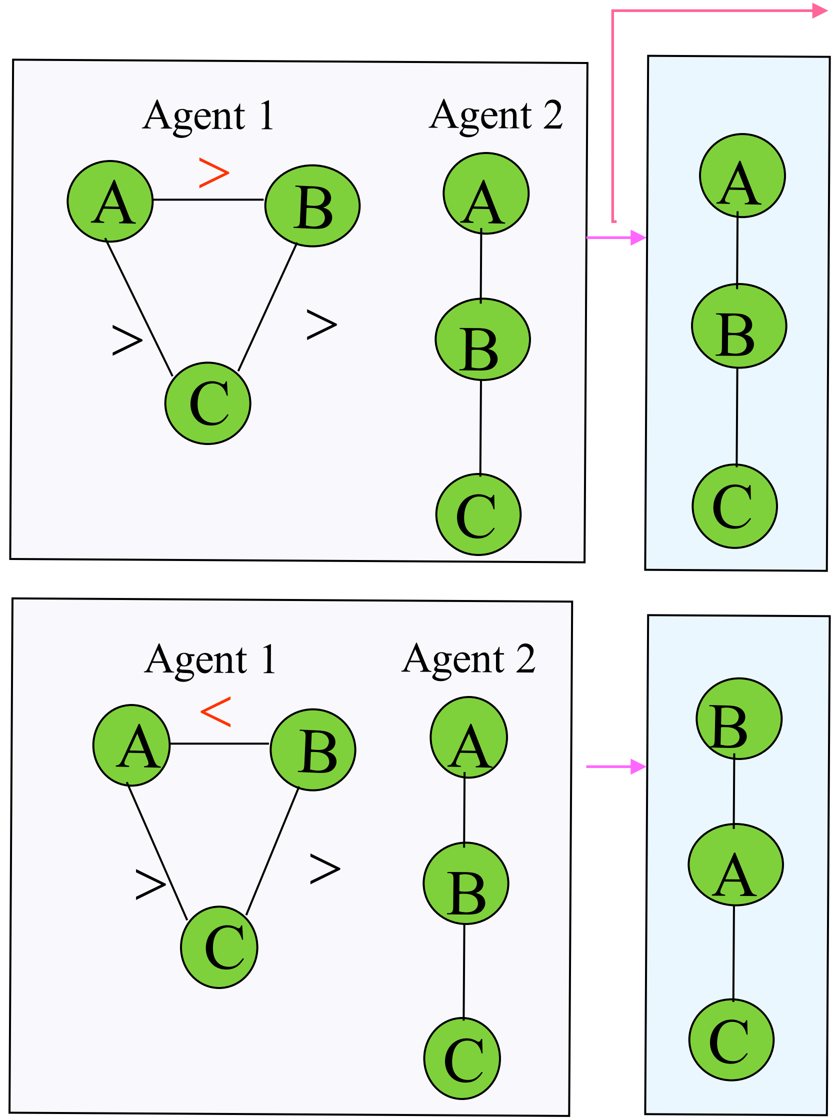
- Given the combined result, PW and NW are easy to find
  - A in NW if no arc (A-B) with B>A
  - A in PW if all arcs (A-B) with B>A contain also other labels
- Computing the combined result: in general NP-hard
- **If f is IIA and monotonic**
  - we can compute an **upper approximation (cr*) in polynomial time**
  - Also, **given cr*, polynomial to compute PW and NW**
    - algorithm not affected by approximation
  - **IIA**: when rel(A,B) in the result depends only by rel(A,B) given by the agents
  - **monotonic**: when we improve an outcome in a profile (for ex. we pass from A>B to A=B), then it improves also in the result

Cr*: upper approximation of the combined result

- Obtained by:
  - Considering two profile completions:
    - (A?B) replaced with (A>B) for every agent
    - (A?B) replaced with (A<B) for every agent
    - Then two results ($A\ r_1\ B$) and ($A\ r_2\ B$)
  - In cr*, put (A r B) where r is {r_1,r_2,everything between them}
  - Order of relations: <, = and ~, >
  - f is IIA and monotonic ➔ cr* upper approx.of cr
  - Approximation only on arcs with all four labels
    - involves only = and ~

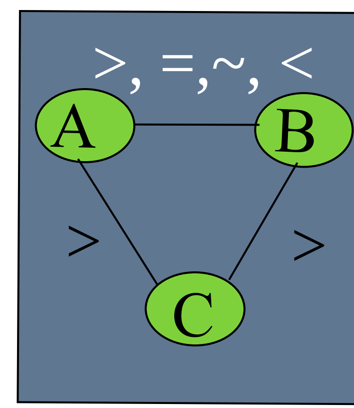
cr* upper approx.of cr: example with Lex





Lex:
agents are ordered, ArB given by the first agent in the order that doesn't declare A=B

cr*



PW = {A,B}
NW = ∅

Computing PW and NW

- Algorithm computing NW and PW in polynomial time, given cr*
  - Input
    - f: **IIA**, **monotonic** pref. aggregation function
    - ip: incomplete profile over outcomes in Ω
    - cr*(f,ip): approximation of combined result
  - Output
    - P, N: sets of outcomes

Computing PW and NW easily

Input: **f**: IIA, monotonic pref. aggr. function,

ip: incomplete profile,

cr*(f,ip): approximation of combined result

Output: **P**, **N**: sets of outcomes

$P \leftarrow \Omega$, $N \leftarrow \Omega$

foreach $A \in \Omega$ **do**

- **if** $\exists\, C \in \Omega$ s.t. $\{<\} \subseteq$ **rel*(A,C)** **then**
  - $N \leftarrow N - \{A\}$
- **if** $\exists\, C \in \Omega$ s.t. $\{<\} =$ **rel*(A,C)** **then**
  - $P \leftarrow P - \{A\}$

return P,N

It terminates in $O(|\Omega|^2)$ time with N=NW and P=PW

IIA+monotone pref. aggr. functions

- Pareto: given any two outcomes A and B
  - A>B iff A>B or A=B for all agents and A>B for at least one
  - A~B otherwise
- Lex
  - agents are ordered and, given any two outcomes A and B, the relation between them in the result is the one given by the first agent in the order that doesn't declare A=B
- Approval voting
  - tractability result already proven in [Konczak and Lang, 2005] since it is a positional scoring rule

Preference elicitation - (1)

- Process of asking queries to agents in order to determine their preferences over outcomes

[Chen and Pu, 2004]

- At each stage in eliciting preference there is a set of possible and necessary winners
- PW = NW $\rightarrow$ preference elicitation is over, no matter how incompleteness is resolved
- Checking when PW = NW: **hard** in general

[Conitzer and Sandholm, 2002]

- We prove that pref.elicitation is **easy** if f is IIA

Preference elicitation - (2)

- $PW = NW \rightarrow$ preference elicitation is over
  - At the beginning: $NW = \emptyset$ $PW = \Omega$
  - As preferences are declared: $NW \uparrow$ $PW \downarrow$
  - If $PW \supset NW$, and $\mathbf{A} \in PW - NW$, A can become a loser or necessary winner
    - Enough to perform **ask(A,B),** $\forall$**B**$\in$**PW**
      - $C \notin PW$ is a loser $\rightarrow$ dominated
      - f is **IIA** $\rightarrow$ ask(A.B) involves only A-B preferences
  - $O(|PW|^2)$ steps to remove incompleteness

Preference elicitation - (3)

- f is IIA $\rightarrow$ determining set of winners via pref. elicitation is polynomial in |agents| and |outcomes|

Winner determination

Input: **f**: IIA, pol. computable pref. aggr. function,
P, N: set of outcomes

Output: **W**: set of outcomes

We can use P and N returned by previous algorithm

wins: bool, P←PW, N←NW
while P≠N **do**
- **choose** A∈P–N
  - wins ← true, Pa ← P – {A}
  - **repeat**
    - **choose** B∈Pa
    - **if** ∃agent s.t. A?B **then**
      - ask(A,B)
      - compute f(A,B)
    - **if** f(A,B)=(A>B) **then**
      - P ← P – {B}
    - **if** f(A,B)=(A<B) **then**
      - P ← P – {A}; wins ← false
    - Pa ← Pa – {B}
  - **until** f(A,B) = (A<B) or Pa = ∅
  - **if** wins=true **then**
    - N ← N ∪ {A}

W ←N, **return** W

Main results

- Computing PW and NW : NP-hard
- Computing good approximations of PW and NW: NP-hard
- Computing the combined result: NP-hard
- If f IIA+monotonic (and pol. computable) then
  - computing an approximation of cr is polynomial
  - computing PW and NW is polynomial
- if f IIA then
  - preference elicitation (i.e., until PW=NW) is polynomial

Pini,Rossi,Venable,Walsh, Incompleteness and Incomparability in Preference Aggregation: Complexity Results. Artificial Intelligence 2011

Future work

- Adding constraints to agents' preferences
  - possible and necessary winner must be also feasible
- Expressing preferences via compact knowledge representation formalisms (Ex.: CP-nets and soft constraints)
  - determining PW and NW directly from these compact formalisms
- Adding possibility distribution over the completions of an incomplete preference relation between outcomes

Winner determination in voting trees

Outline

- Background
  - Incomplete preferences
  - Incomplete profiles
- Complete majority graph
  - Condorcet winner
  - Schwartz winner
  - Fair Schwartz winner
- Incomplete majority graph
  - Possible/necessary Condorcet winners
  - Possible/necessary Schwartz winner
- Winner determination for (simple) voting tree
  - From the majority graph
  - From the weighted/unweighted profile
- Complexity results
- Balanced agendas

Preferences

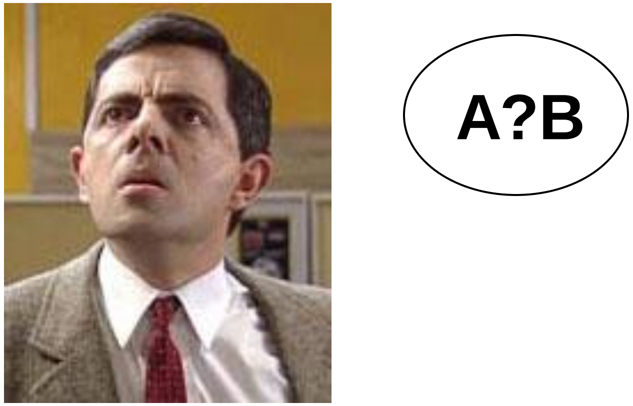
- Agents express their preferences over candidates by a (possibly incomplete) total order

Why?

An agent may state a preference over a pair of candidates

Other agents may not know their preference

…or may not want to disclose it



Profiles

- When many (*n*) agents are involved:
- **Profile:** sequence of n total orders

A>B>C>D

A>C>B>D

B>D>A>C

C>B>D>A

D>A>C>B

- **Incomplete profile:** one or more total orders are incomplete

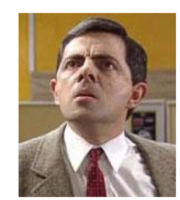
A>B>C>D

A>C>B, **A?D**, **B?D**, **C?D**

B>D>A>C

C>B>A,D>A **B?D**,**C?D**

D>A>B,C>B **A?C**, **D?C**



Complete Weighted profiles

- **Complete weighted profile:**
  - Each agent has a given weight
  - all preferences are known





weights	20	2	2	2	2
preferences	A>B>C	C>B>A	B>A>C	C>B>A	A>C>B

Incomplete Weighted profiles

- **Incomplete weighted profile:**
  - Each agent has a given weight
  - Some preferences are not known




20	2	5	10	2
A>B>C>D	A>C>B, **A?D**, **B?D**, **C?D**	B>D>A>C	C>B>A,D>A **B?D**,**C?D**	D>A>B,C>B **A?C**, **D?C**

Majority Graph of a Profile

- Given profile **P**, its majority graph **M(P)** is s.t.:
  - Nodes correspond to candidates
  - Directed edge A→B iff majority says A>B

n is odd!

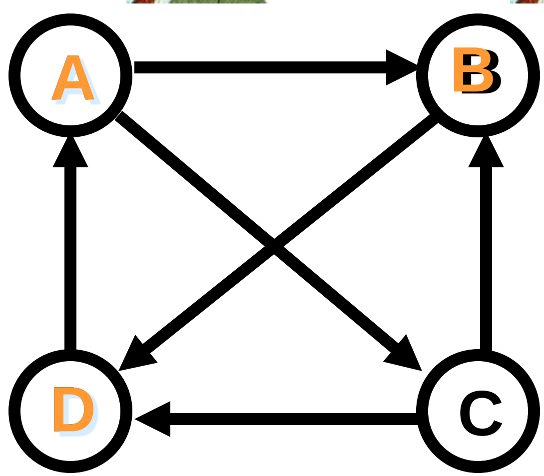
A>B>C>D | A>C>B>D | B>D>A>C | C>B>D>A | D>A>C>B





Profile P

Majority Graph M(P)



Relation of the majority graph not transitive!

Majority Graph of an Incomplete Profile

- Given an incomplete profile **P**, its majority graph **M(P)** is s.t.:
  - Nodes correspond to candidates
  - Directed edge A →B iff more than half says A>B
  - **No edge if no majority**

A>B>C>D

A>C>B, **A?D**, **B?D**, **C?D**

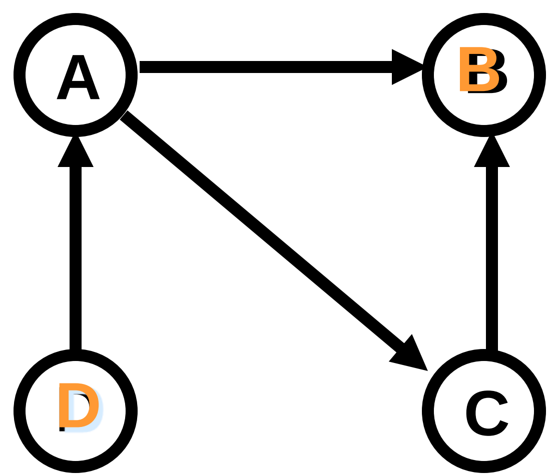
B>D>A>C

C>B>A,D>A **B?D**,**C?D**

D>A>B,C>B **A?C**, **D?C**

Incomplete Profile P

Incomplete Majority Graph M(P)



Majority Graph of an Incomplete Weighted Profile

- Given an incomplete weighted profile **P**, its majority graph **M(P)** is s.t.:
  - Nodes correspond to candidates
  - Directed edge A→B iff the weighted majority says A>B
  - **No edge if no weighted majority**

A>B>C>D

15

A>C>B, **A?D**, **B?D**, **C?D**

3

B>D>A>C

5

C>B>A,D>A **B?D**,**C?D**

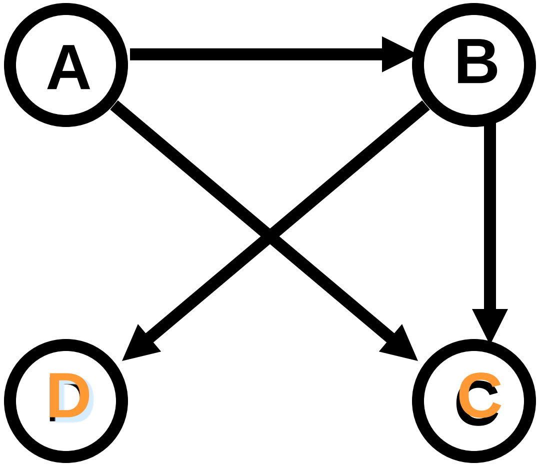
10

D>A>B,C>B **A?C**, **D?C**

2

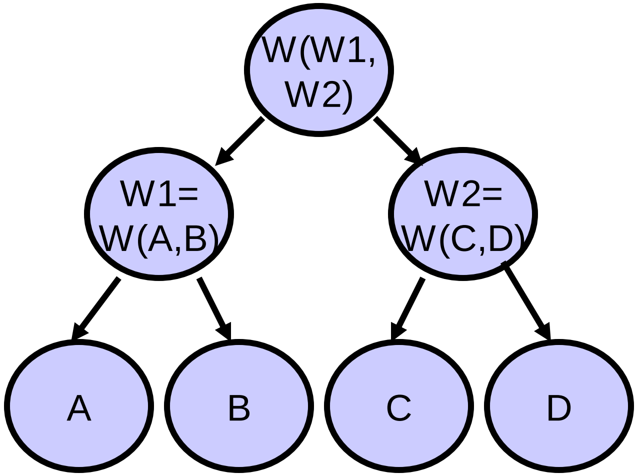
Incomplete Profile P

Incomplete Majority Graph M(P)

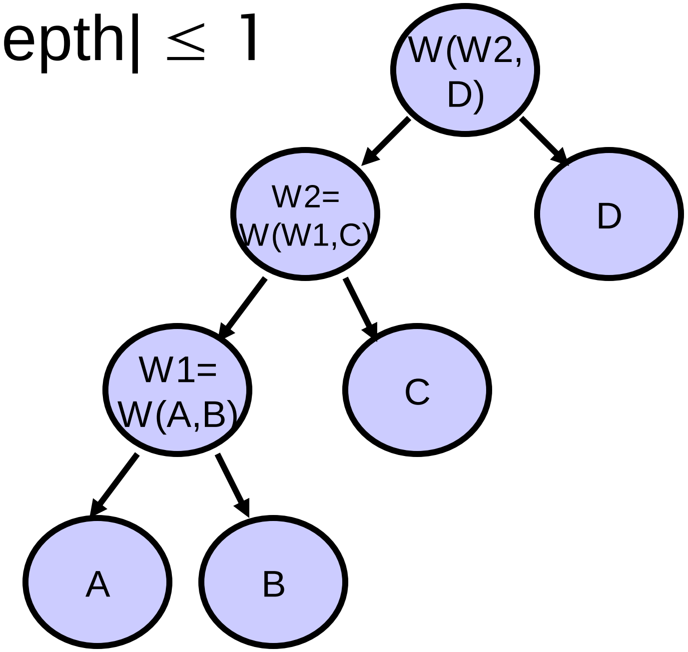


Binary voting tree

- Given a set of candidates, a binary voting tree T is such that
  - Terminal node = candidate
  - Non-terminal node = winner of its two children
  - Balanced iff |maxdepth - mindepth| $\leq$ 1



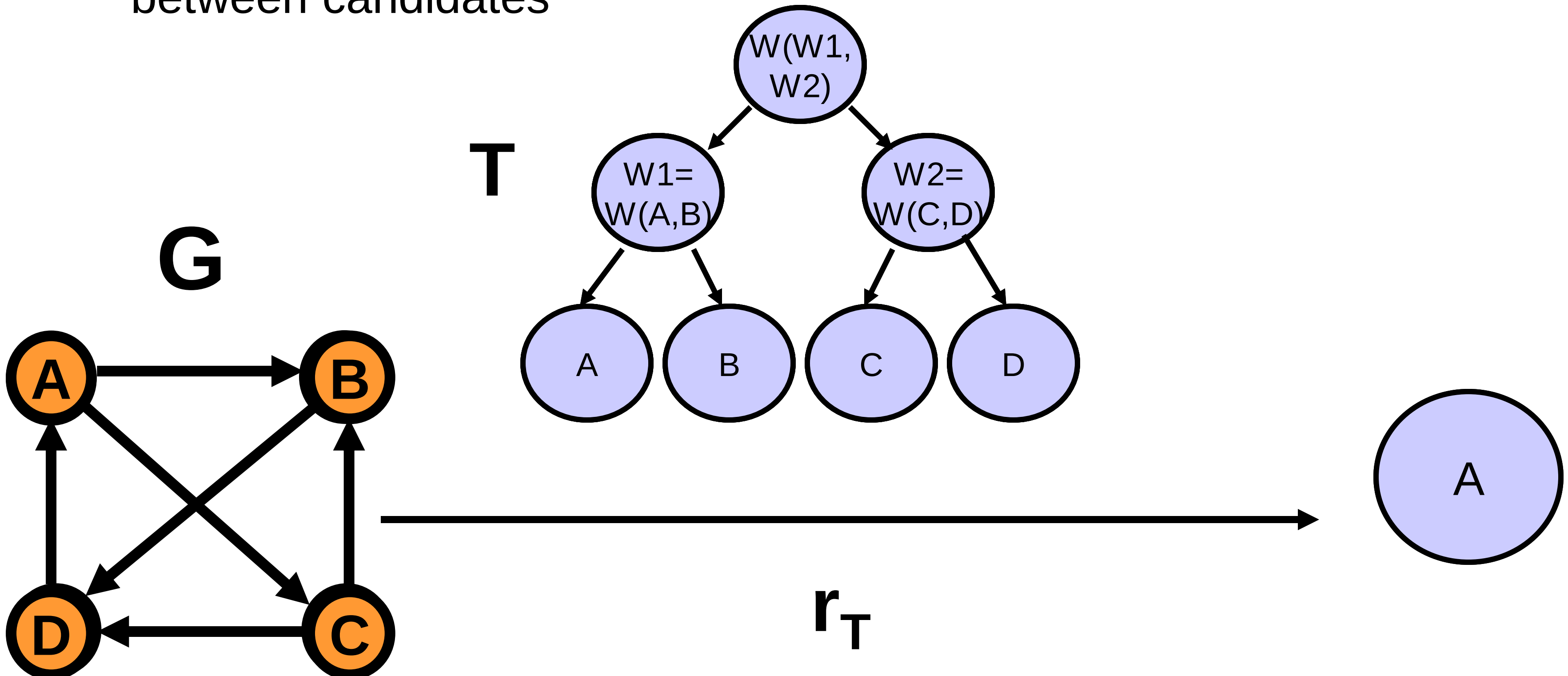
Balanced



Unbalanced

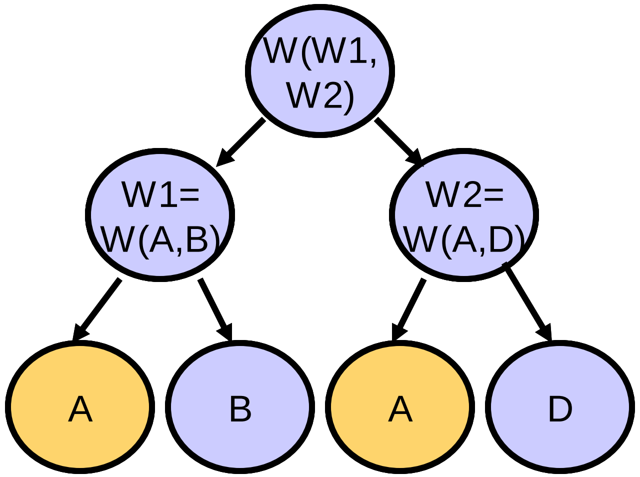
Simple voting tree

- **Binary voting tree T → voting rule r_T**
- **r_T: majority graph G → candidate (winner)**
- Every candidate can appear once in the leaves
- Sequence of pairwise comparisons (also called **agenda**) between candidates

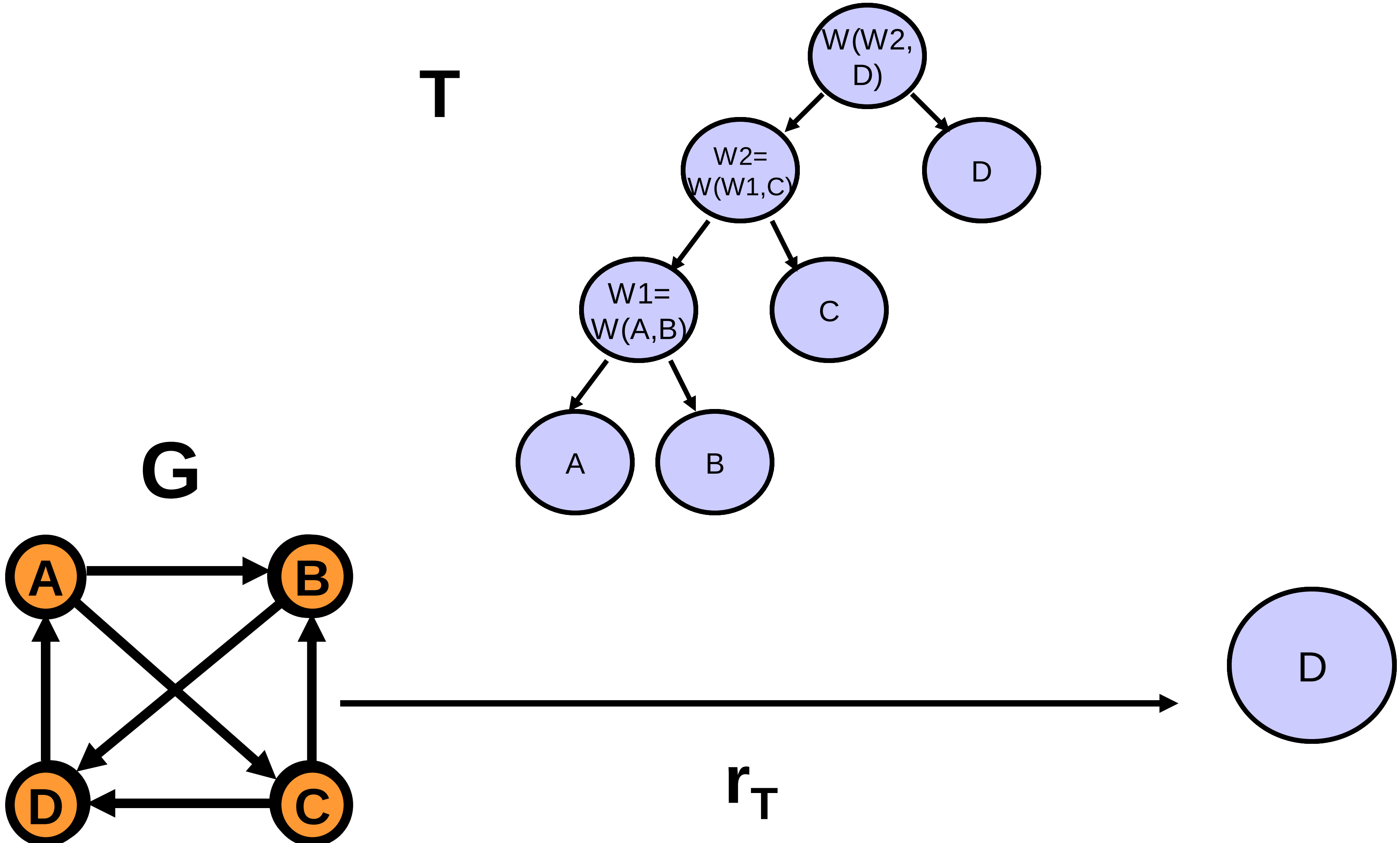


Voting tree

- Voting tree: an extension of simple voting tree where
  - every candidate can appear several times as *leaf*



Different tree, different winner

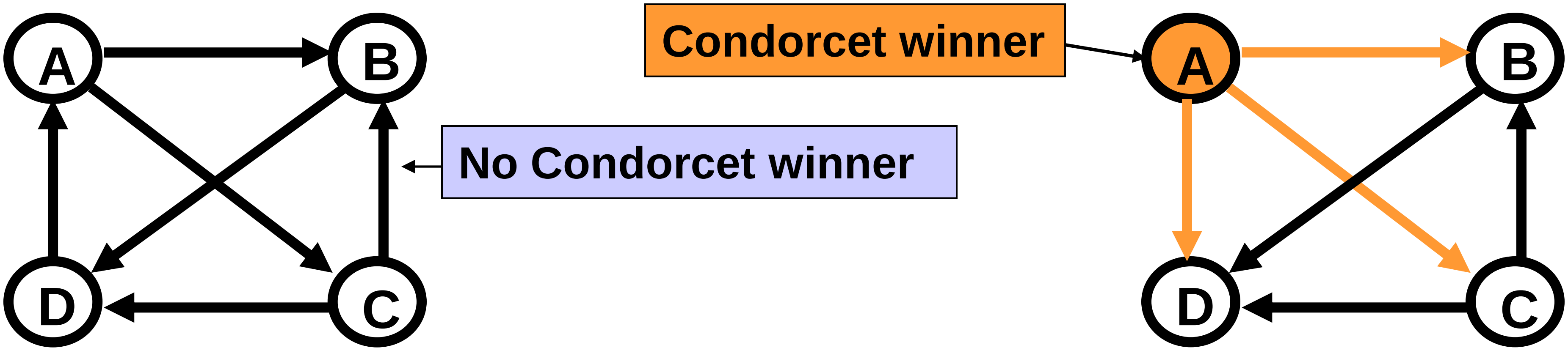


Simple voting trees

Condorcet winner

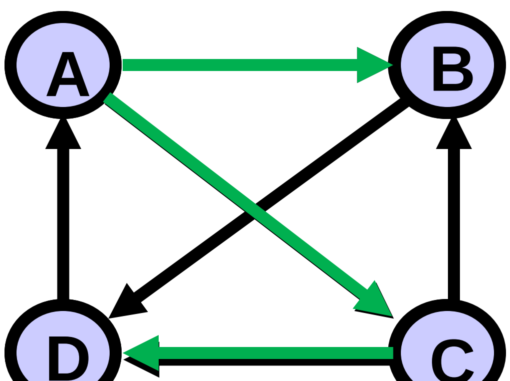
- Given a profile P, candidate A is a Condorcet winner iff $\forall T$, binary tree, $r_T(M(P))=A$.
- Given M(P), A is a Condorcet winner iff its node in M(P) has **only outgoing edges**
  - Polynomial time

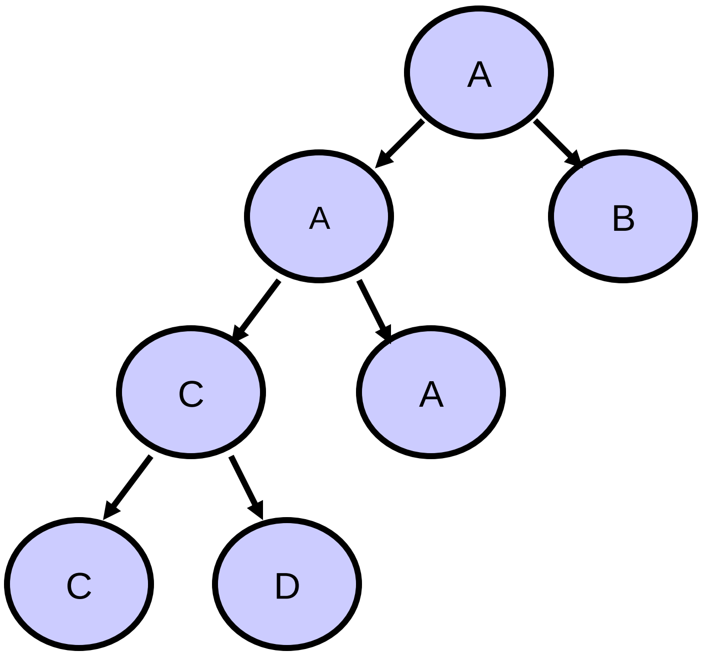
If ∃, then unique



Schwartz winners

- Given a profile P, candidate A is a Schwartz winner iff ∃T, binary tree, such that $r_T(M(P))=A$.
- Given M(P), candidate A is a possible winner iff **there is path from node A to every other node**
  - Polynomial time





Incomplete preferences

- Who will win? **Different types of uncertainty:**
  - Unknown voting tree
  - Incomplete preferences
    - incomplete profile
    - incomplete majority graph

Possible Schwarz (PS) winner A: $\exists$ *completion* of **maj. graph /profile**, $\exists$ voting tree s.t. A wins

***Necessary Schwartz* (NS) winner A**: $\forall$ *completion* of **maj. graph/profile,** $\exists$ voting tree s.t. A wins

Possible Condorcet (PC) winner A: $\exists$ *completion* of **maj. graph/profile,** s.t. $\forall$ voting tree A wins

Necessary Condorcet (NC) winner A: $\forall$ *completion* of **maj. graph/profile,** s.t. $\forall$ voting tree A wins

Incomplete preferences

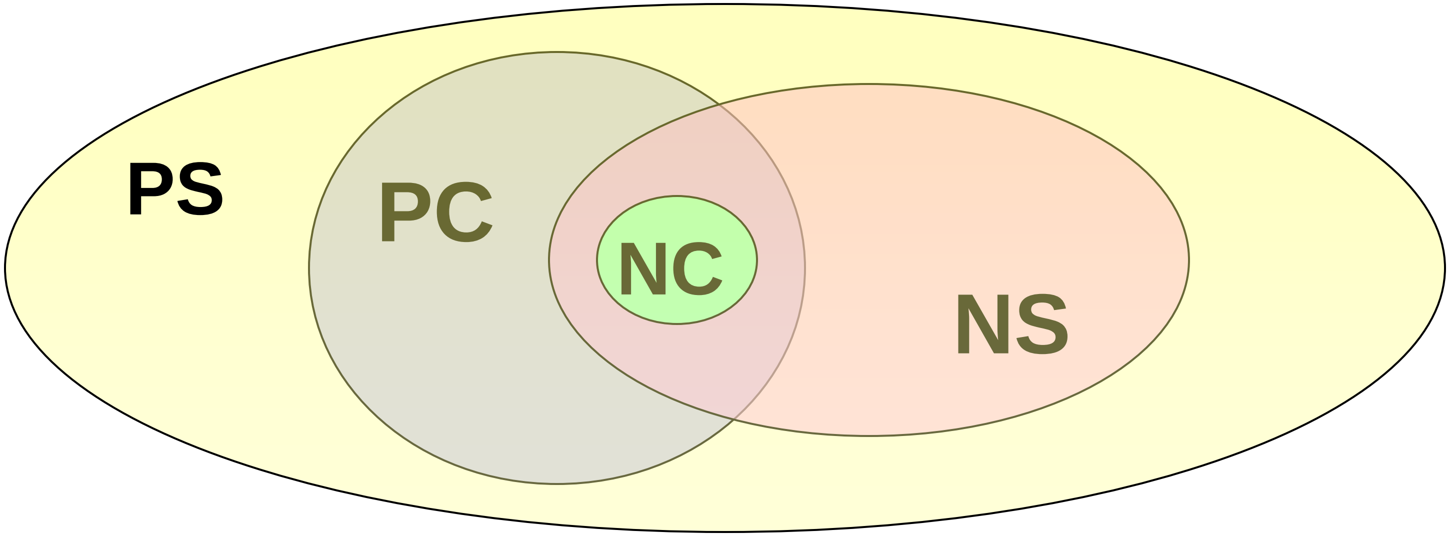
Possible Schwarz (PS) winner A: $\exists$ *completion* of **maj. graph /profile**, $\exists$ voting tree s.t. A wins

***Necessary Schwartz* (NS) winner A**: $\forall$ *completion* of **maj. graph/profile**, $\exists$ voting tree s.t. A wins

Possible Condorcet (PC) winner A: $\exists$ *completion* of **maj. graph/profile,** s.t. $\forall$ voting tree A wins

Necessary Condorcet (NC) winner A: $\forall$ *completion* of **maj. graph/profile,** s.t. $\forall$ voting tree A wins

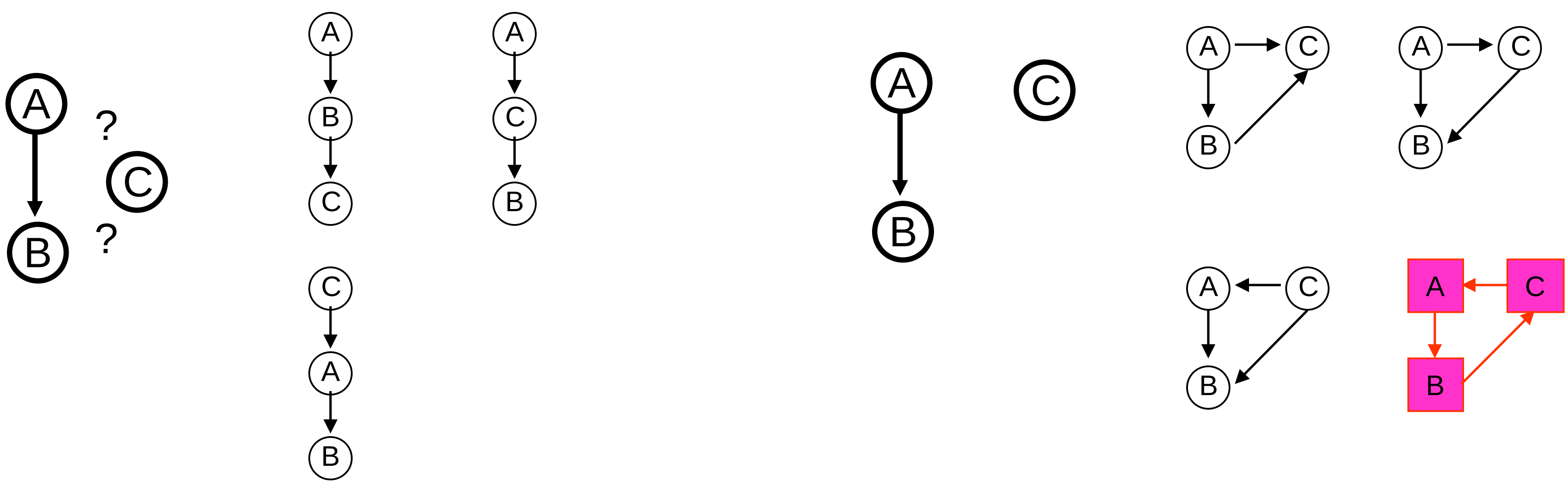
$NC \subseteq PC \cap NS \qquad PC \cup NS \subseteq PS$



Completions of the Majority graph and Profile

- **∀completion of the profile P → ∃ completion of the maj. graph M(P)**
- **Not vice versa** (transitivity!)
- Completions(M(P)) ⊇ {M(P')| P' completion of P}
- Example: 1 agent

Profile **profile completions** **Majority graph** **Maj. G. completions**



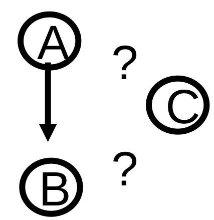
Possible Schwartz winners

$PS(P) \subseteq PS(M(P))$

- **P** : unweighted profile
- **M(P)** : majority graph
- **PS(P):** A∈PS(P) iff ∃ completion of profile P, ∃ voting tree s.t. A wins
- **PS(M(P)):** A∈WP(M(P)) iff ∃ completion of maj. graph M(P) , ∃ voting tree s.t. A wins

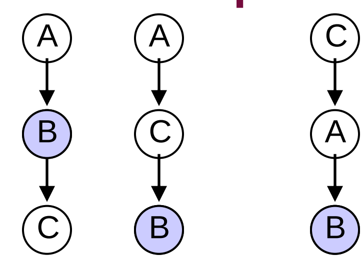
∃ completion of P → ∃ completion of M(P)

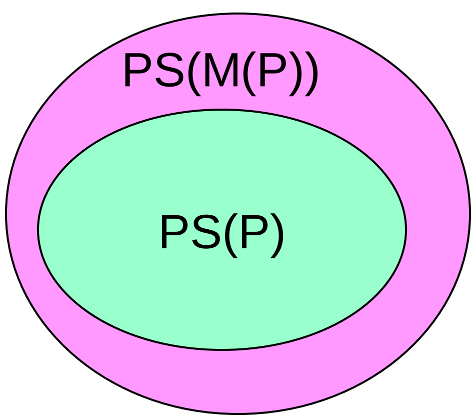
Profile



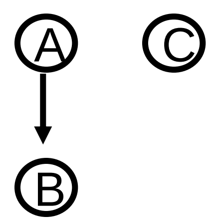
B ∉ PS(P)

profile completions



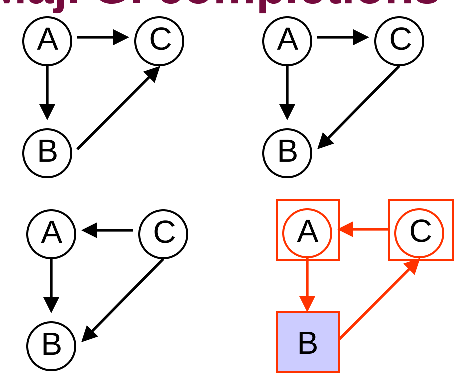


Majority graph



B ∈ PS(M(P))

Maj. G. completions



Possible Condorcet winners

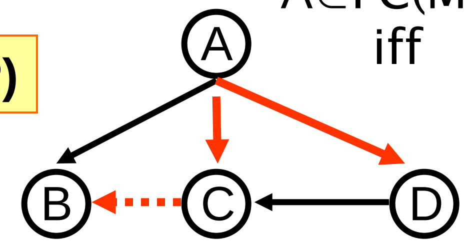
PC(P)=PC(M(P))

- **P** : unweighted profile
- **M(P)** : majority graph
- **PC(P):** A∈PC(P) iff ∃ completion of profile P, ∀ voting tree s.t. A wins
- **PC(M(P):** A∈PC(M(P)) iff ∃ completion of maj. graph M(P) , ∀ voting tree s.t. A wins

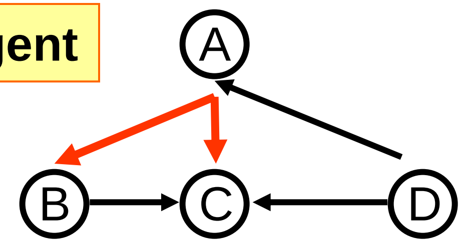
∃ completion of P → ∃ completion of M(P)

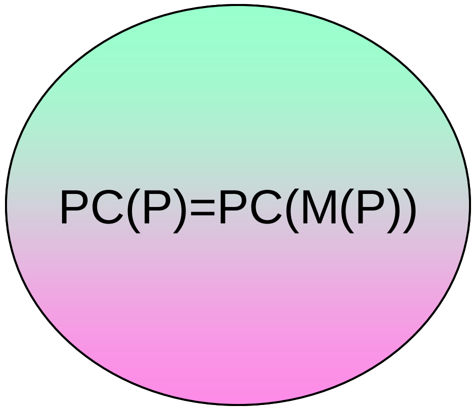
A∈PC(M(P)) iff

M(P)



Agent





Putting a candidate above all others never causes transitivity problems

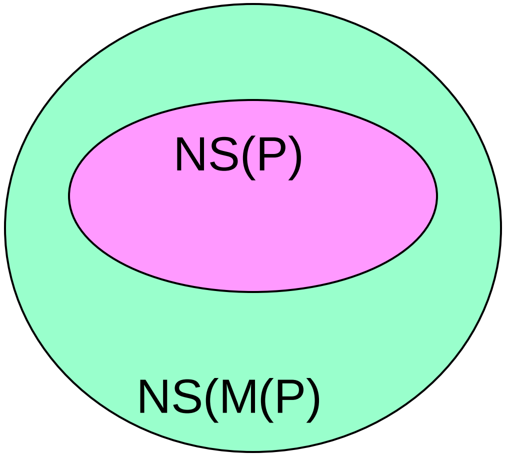
If A∈PC(M(P)) then A∈PC(P)

Necessary Schwartz winners

$NS(M(P)) \subseteq NS(P)$

- **P** : unweighted profile
- **M(P)** : majority graph
- **NS(P):** $A \in NS(P)$ iff $\forall$ completion of profile P, $\exists$ voting tree s.t. A wins
- **NS(M(P)):** $A \in NS(M(P))$ iff $\forall$ completion of maj. graph M(P) , $\exists$ voting tree s.t. A wins

$\text{Completions}(M(P)) \supseteq \{M(P') \mid P' \text{ completion of } P\}$

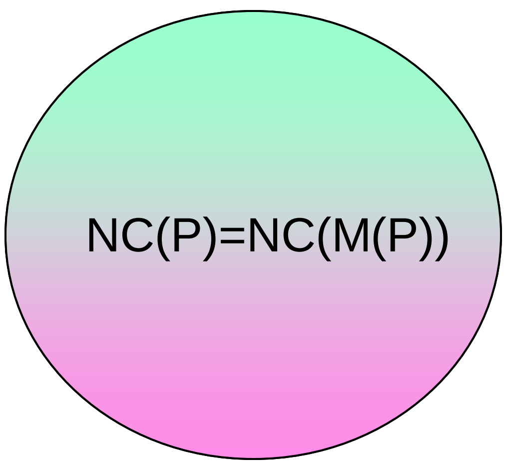


Necessary Condorcet winners

NC(M(P))=NC(P)

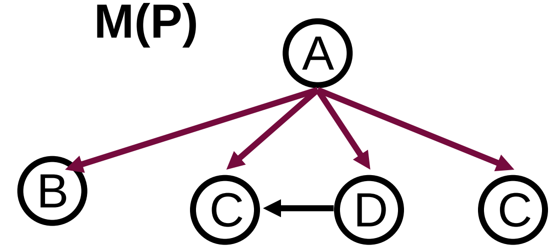
- **P** : unweighted profile
- **M(P)** : majority graph
- **NC(P):** $A \in NC(P)$ iff $\forall$ completion of profile P, $\forall$ voting tree s.t. A wins
- **NC(M(P)):** $A \in NC(M(P))$ iff $\forall$ completion of maj. graph M(P) , $\forall$ voting tree s.t. A wins

Completions(M(P)) $\supseteq$ {M(P')| P' completion of P}



If $A \in NC(P)$ then

M(P)



No arrows involving A
can be missing or against A

$A \in NC(M(P))$

Computing majority graph winners

- Polynomial for simple voting trees for all types of winners
  - A is a Possible Schwartz winner iff it is possible to complete the majority graph such that every outcome is reachable from A
  - A is a necessary Schwartz winner iff, ∀B, there is a path from A to B in G
  - A is possible Condorcet winner iff A has no ingoing edges
  - A is a necessary Condorcet winner iff A has outgoing edges to all other candidates

[Lang, Pini,Rossi,Venable,Walsh, IJCAI 07]
[Pini,Rossi,Venable,Walsh, KR08]

Outline

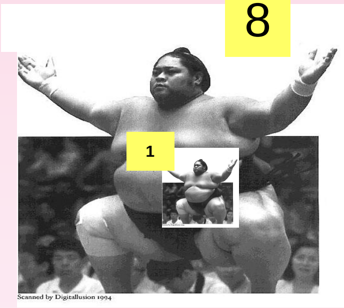
- Background
  - Incomplete preferences
  - Incomplete profiles
- Complete majority graph
  - Condorcet winner
  - Schwartz winner
  - Fair Schwartz winner
- Incomplete majority graph
  - Possible/necessary Condorcet winners
  - Possible/necessary Schwartz winner
- Winner determination for (simple) voting tree
  - From the majority graph
  - From the weighted/unweighted profile
- Complexity results
- Balanced agendas

From weighted to unweighted

Weighted profile P

A>B>C>D

8



A>C>B, **A?D**, **B?D**, **C?D**

3

B>D>A>C

1

C>B>A,D>A **B?D**,**C?D**

10

D>A>B,C>B **A?C**, **D?C**

2

M(P)=M(P’)

NC(P)=NC(P’)

Completions(P’)⊇Completions(P)

PC(P)=PC(P’)

Unweighted profile P’

A>B>C>D

A>C>B, **A?D**, **B?D**, **C?D**

B>D>A>C

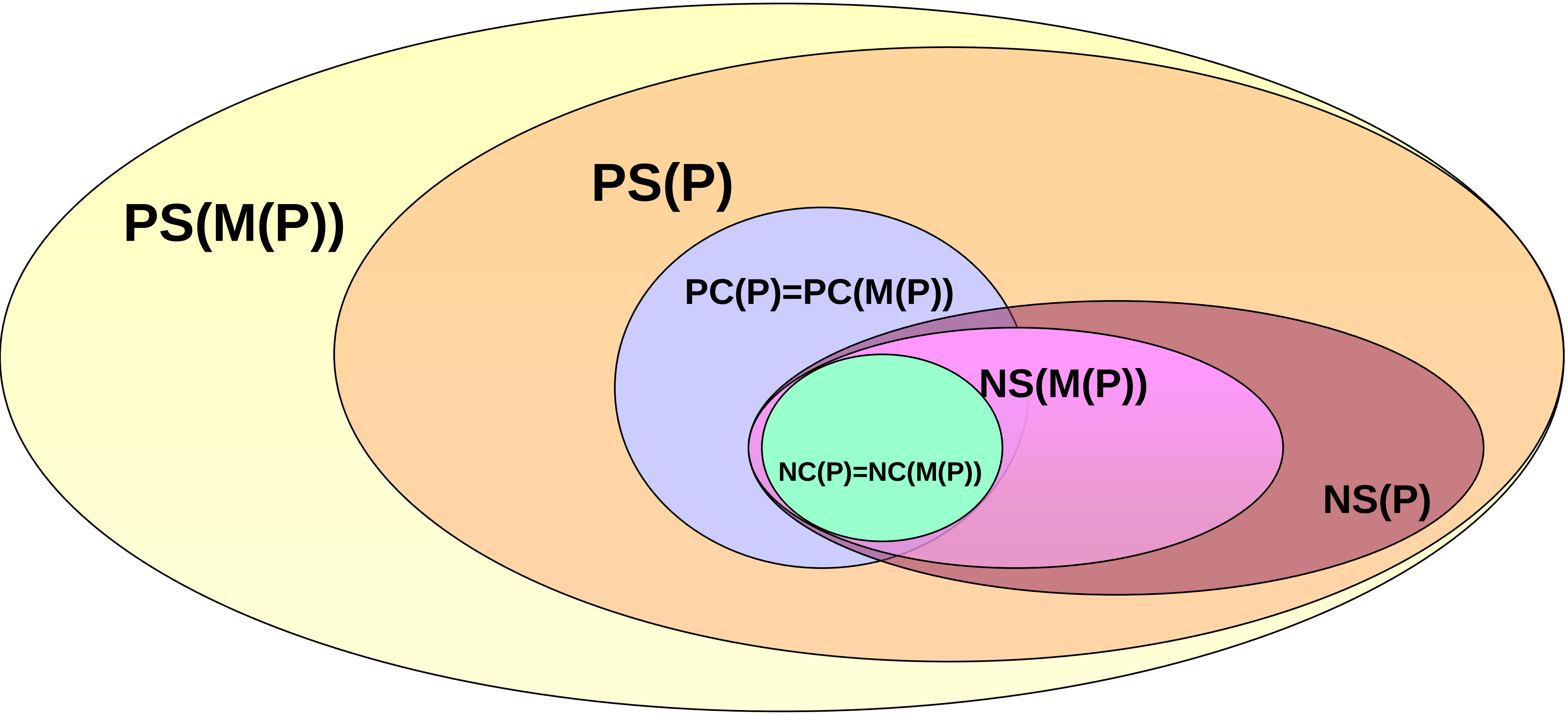
1

C>B>A,D>A **B?D**,**C?D**

D>A>B,C>B **A?C**, **D?C**

Winners sets
(weighted or unweighted profile)

- **PC**(P) **= PC**(M(P)) and **NC**(P) **= NC**(M(P))
- **PS**(P) ⊃ **PS**(M(P)) and **NS**(P) ⊇ **NS**(M(P))



Complexity results: Possible Condorcet Winners

Theorem:

- P incomplete weighted profile
- **“A ∈PC(P)?” is polynomial**

Proof

1. P → unweighted P’
2. PC(P)=PC(P’)
3. PC(P’)=PC(M(P’))
4. A∈PC(M(P’)) iff all arrows involving A in M(P’) do not point against A (polynomial test)

Complexity results: Necessary Condorcet Winners

Theorem:

- P incomplete weighted profile
- **"A $\in$NC(P)?" is polynomial**

Proof

1. P → unweighted P'
2. NC(P)=NC(P')
3. NC(P')=NC(M(P'))
4. A$\in$NC(M(P')) iff all arrows involving A in M(P') are not missing and do not point against A (polynomial test)

Complexity results: Possible Schwartz Winners

Theorem:

- P incomplete weighted profile, 3 or more candidates
- **“A ∈PS(P)?” is NP-complete**

Proof

Reduction from the number partitioning problem

Complexity results

	X= incomplete **maj.graph** Y=tree *[Lang et al. iIJCAI'07]*	X=incomplete **weighted profile** Y=tree
PossibleCondorcet $\exists X \ \forall Y$	EASY No ingoing edges	EASY Same set as
Necessary Condorcet $\forall X \ \forall Y$	EASY Only outgoing edges	EASY Same set as
Possible Schwartz $\exists X \ \exists Y$	EASY Completion with path to every candidate	NP-complete Reduction from the number partitioning problem
Necessary Schwartz $\forall X \ \exists Y$	EASY Path to every candidate	?

Fair Possible Schwartz Winners

- Some possible winners may win only on *very unbalanced trees*, competing only few times**.** **UNFAIR!**

- ***Fair* possible Schwartz (FPS) winner A** :
  ∃ completion of maj. graph/profile, ∃ **balanced** simple voting tree s.t. A wins

- Fairness comes from the fact that both finalists will have faced the same number of competitions, or the same number plus or minus one.

Complexity results: Fair Possible Schwartz Winners

Theorem:

- P incomplete weighted profile, 3 or more candidates
- **“A ∈FPS(P)?” is NP-complete**

Proof

1. When there are 3 candidates, then every simple voting tree is balanced
2. Conclude as for PS(P)

Fixed trees: possible and necessary winners

- **T: simple voting tree**
- **A:** a candidate

- ***Necessary winner*** **(NW)**: $\forall$ completion of maj. graph/profile, A wins **in the fixed tree T**

- ***Possible winner*** **(PW) :** $\exists$ completion of maj. graph/profile, A wins **in the fixed tree T**

Determining possible/necessary winners for simple voting trees

Algorithm 1: *Win*

```
Input: T: simple voting tree, G: incomplete maj. graph;
Output: W: set of candidates;
if  root(T)≠nil and left(T)=right(T)=nil  then
      W ← root(T);
else
      W1 ← Win(left(T), G);
      W2 ← Win(right(T), G);
      W  ← W1 ∪ W2;
      foreach  s∈W1 do
            if  s <m r,  ∀r∈W2  then
                  W ← W – {s};
      foreach  r∈W2 do
            if  r <m s,  ∀s∈W1  then
                  W ← W – {r};
return W;
```

W contains possible winners

If |W|=1 → necessary winner

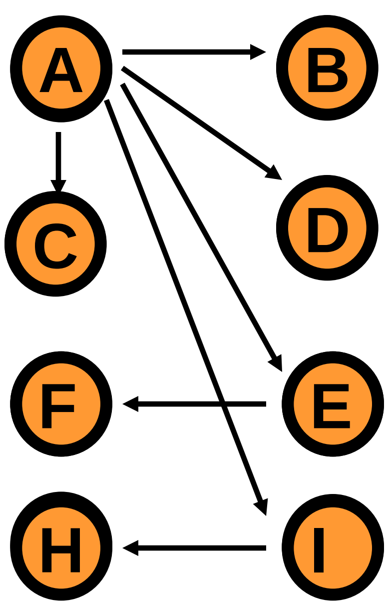
Th.: "A ∈PW(G)?", "A ∈NW(G)?", are **polynomial**

Possible and necessary winners: an example

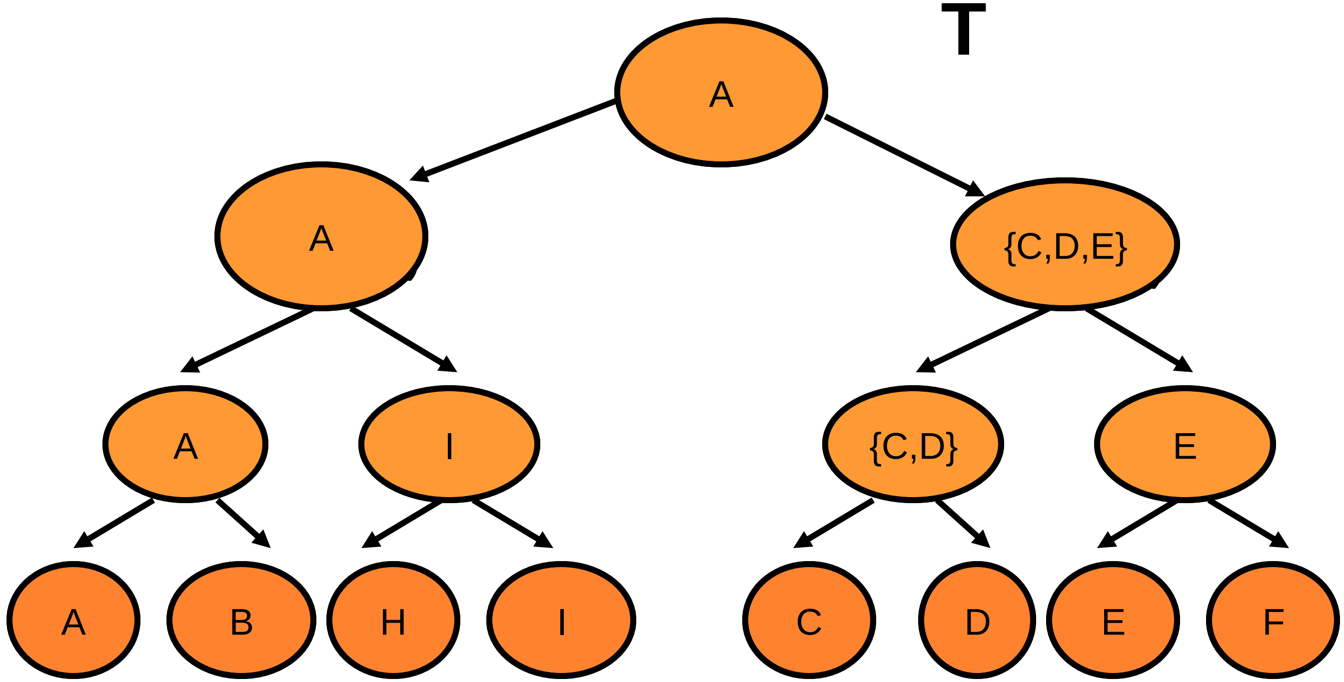
- **Ω={A, B, C, D, E, F, H, I}**: set of candidates
- **T**: simple voting tree
- **G:** incomplete majority graph

Win returns a single candidate A→ A is a **NW**

G



T



Complexity result: Possible winners

Theorem:

- P incomplete weighted profile,
- 3 or more candidates
- T simple voting tree
- **“A $\in$PW(P,T)?” is NP-complete**

Proof

- Reduction from the number partitioning problem

This theorem holds also when T is balanced

when there are 3 candidates, every simple voting tree is balanced

Complexity result: Necessary winners

Theorem:

- P incomplete weighted profile,
- 4 or more candidates
- T simple voting tree
- **"A $\in$NW(P,T)?" is coNP-complete**

Proof

- Reduction from the number partitioning problem

Summary: Winners with missing preferences

A is a :

Possible Schwartz winner (PS) if $\exists$ completion of maj. graph /profile, $\exists$ (simple) voting tree

Necessary Schwartz winner (NS) if $\forall$ completion of maj. graph/profile, $\exists$ (simple) voting tree

Possible Condorcet winner (PC) if $\exists$ completion of maj. graph/profile, $\forall$(simple) voting tree

Necessary Condorcet winner (NC) if $\forall$ completion of maj. graph/profile, $\forall$ (simple) voting tree

A wins

When tree T is fixed:

Possible winner A (PW): $\exists$ completion of maj. graph/profile s.t. A wins given T

Necessary winner A (NW): $\forall$ completion of maj. graph/profile s.t. A wins given T

M(P) / P	Weights n bounded	No Weights, n bounded	Weights, n unbounded	No Weights, n unbounded
PS	P / NP-c	P / P	P / NP-c	P / ?
NS	P / ?	P / P	P / ?	P / ?
PC	P / P	P / P	P / P	P / P
NC	P / P	P / P	P / P	P / P
FPS	P / NP-c	P / P	? / NP-c	? / ?
PW	P / NP-c	P / P	P / NP-c	P / NP
NW	P / coNP-c	P / P	P / coNP-c	P / coNP

Lang,Pini,Rossi, Salvagnin,Venable,Walsh,
Journal of Autonomous Agents and Multiagent Systems 2012

Majority graph vs profile

- What was known about winners

	Simple voting trees
Possible Schwartz	$\neq$
Necessary Schwartz	?
Possible Condorcet	=
Necessary Condorcet	=
Possible winners	?
Necessary winners	?

Lang,Pini,Rossi,Venable,Walsh, IJCAI 07
Pini,Rossi,Venable,Walsh, KR08

Necessary Schwartz winners

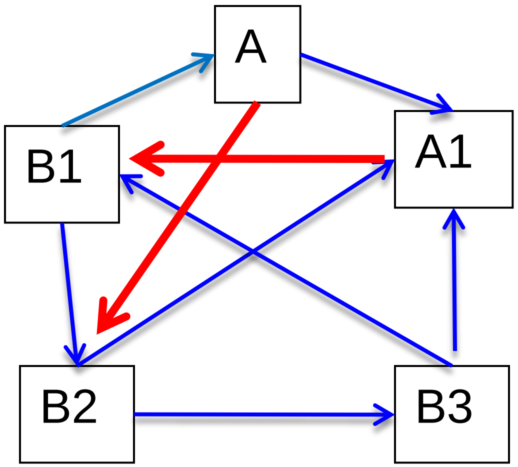
	Simple voting trees
Possible Schwartz	≠
Necessary Schwartz	≠
Possible Condorcet	=
Necessary Condorcet	=
Possible winners	?
Necessary winners	?

∀ completion of maj. graph/profile, ∃ (simple) voting tree

But = with 3 candidates

- Consider this **incomplete profile** with 5 agents and 5 candidates
- agent 1: (A1>B2>B3, A>B1)
- agent 2: (B2>B3>A1>B1>A)
- agent 3: (A>A1>B3>B1>B2)
- agent 4: (B1>A>B2>B3>A1)
- agent 5: (B3>B1>B2>A>A1)

Incomplete Majority Graph G



- A is a **not Necessary Schwartz winner from the majority graph** (no path from A to B1)
- A is a **Necessary Schwartz Winner from the profile**: 2 possible completions for P:
  - 1st completion: **A1>A** → A1>B1 for transitivity → A1>B1 in G
    Tree: B2,B3 →B2,B1 → B1,A1 → A1,A →A wins
  - 2nd completion : **A>A1** → A> B2 for transitivity → A>B2 in G
    Tree: B1,B3 →B3,B2 → B2,A→ A,A1 →A wins

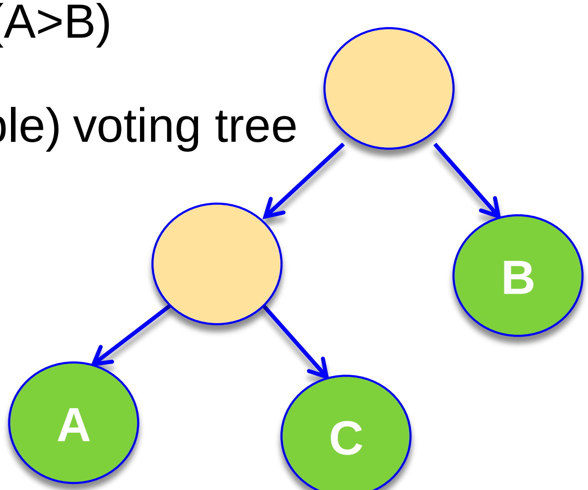
Possible winners

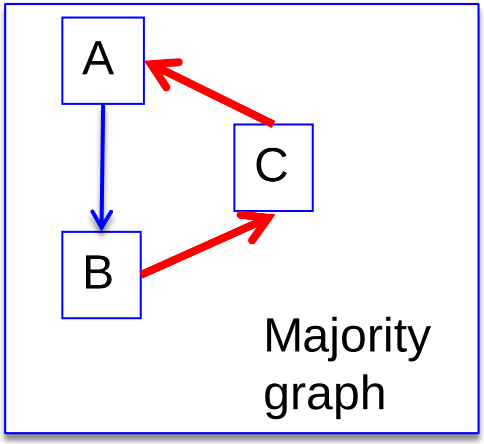
	Simple voting trees
Possible Schwartz	≠
Necessary Schwartz	≠
Possible Condorcet	=
Necessary Condorcet	=
Possible winners	≠
Necessary winners	?

∃ completion of maj. graph/profile, Fixed (simple) voting tree

Consider this **incomplete profile** with 1 agent and 3 candidates
agent 1: (A>B)

(Simple) voting tree





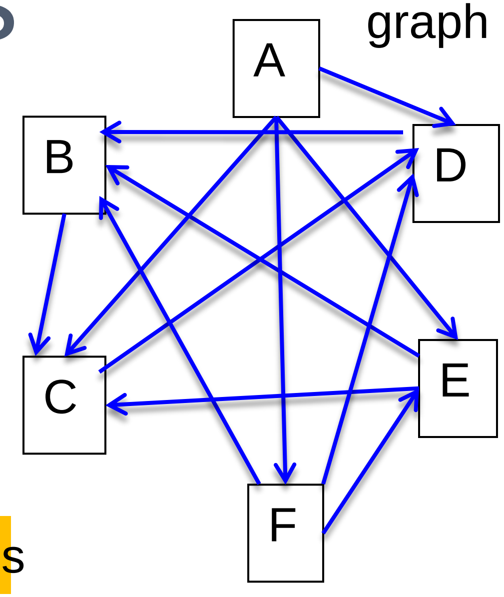
B is a Possible Winner from the majority graph
B is not a Possible Winner from the profile

Necessary winners

	Simple voting trees
Possible Schwartz	≠
Necessary Schwartz	≠
Possible Condorcet	=
Necessary Condorcet	=
Possible winners	≠
Necessary winners	≠

But = with 3 candidates

Majority graph



Consider this **incomplete profile** with 5 agents and 5 candidates

agent 1: (E>B>C, F>D>A)   E>F   F>E

agent 2: (A>E>F>D>B>C)

agent 3: (A>C>D>F>E>B)

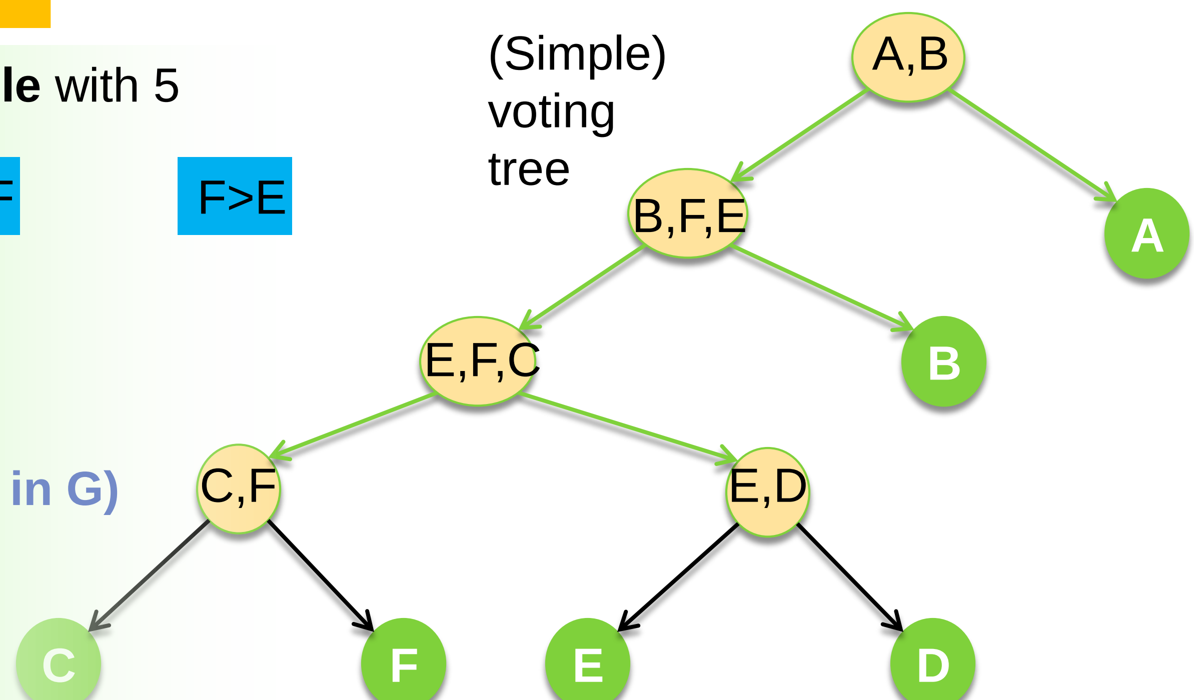
agent 4: (C>D>F>E>B>A)

agent 5: (B>A>F>E>C>D)

(E>F ➔ E>D in G)

No Necessary Winners from the majority graph

(Simple) voting tree

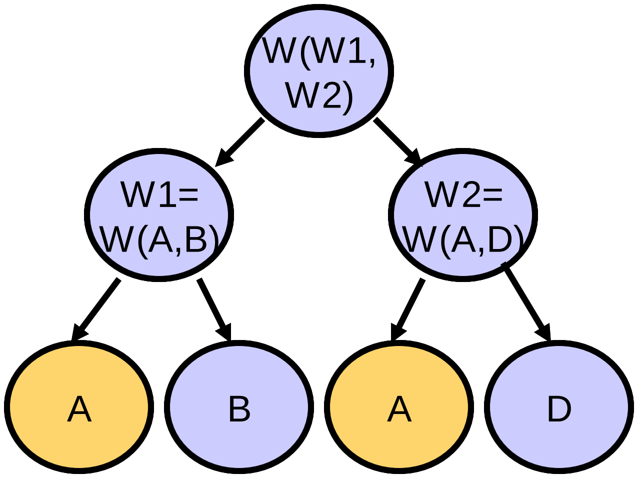


A is a Necessary winner from the profile

Voting trees

Voting tree

- Voting tree: an extension of simple voting tree where
  - every candidate can appear several times as *lea*



Results for voting trees

	Simple voting trees	Voting trees
Possible Schwartz	≠	≠
Necessary Schwartz	≠	≠
Possible Condorcet	=	=
Necessary Condorcet	=	=
Possible winners	≠	≠
Necessary winners	≠	≠

All **inequality results** transfer automatically from simple voting trees that are a special case of voting trees.

All **equality results** can be derived from the proofs since it is never required for a candidate to appear in at most one leaf

Computing majority graph winners

- Polynomial for simple voting trees for all types of winners
  - A is a Possible Schwartz winner iff it is possible to complete the majority graph G such that every outcome is reachable from A
  - A is a necessary Schwartz winner iff, ∀B, there is a path from A to B in G
  - A is possible Condorcet winner iff A has no ingoing edges in G
  - A is a necessary Condorcet winner iff A has outgoing edges to all other candidates in G

[Lang, Pini,Rossi,Venable,Walsh, IJCAI 07]

[Pini,Rossi,Venable,Walsh, KR08]

- All results transfer to voting trees

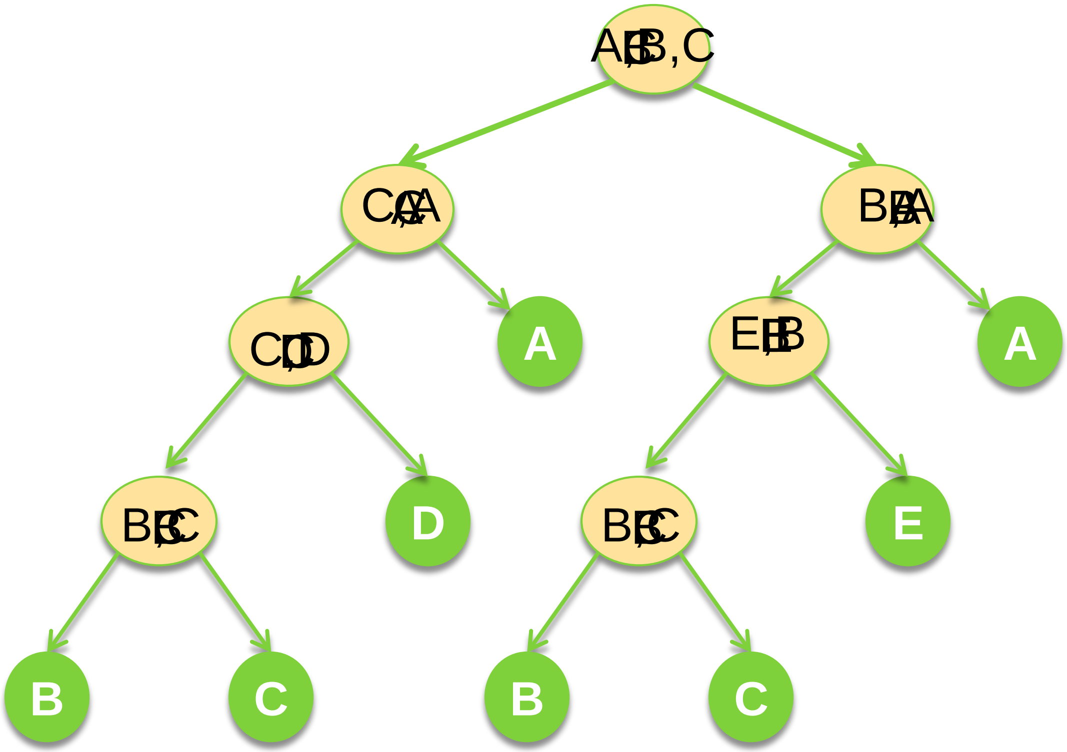
Computing winners from majority graphs

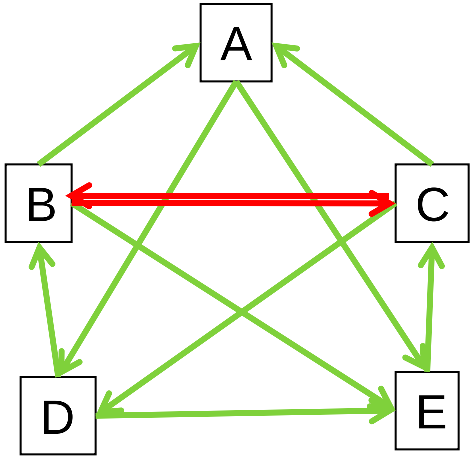
- For simple voting trees it is polynomial:
  1. If root(T)≠∅ and right(T)=∅ and left(T)=∅ then winner=label(root(T))
  2. otherwise the winners are the possible winners of each branch that beat at least one of the possible winners of the other branch
  3. if only one winner is returned then it is a necessary winner

[Pini,Rossi,Venable,Walsh, CLIMA 07]

- However this procedure does not work for voting trees
- An upper approximation of possible winners is computed
- Lower approximation of Necessary winners

Upper approximation of possible winners





Win returns {A,B,C} as possible winners

But A can never win

B>C

B<C

Profile vs majority graph: summary and future work

	Simple voting trees	Voting trees
Possible Schwartz	≠	≠
Necessary Schwartz	≠ = for 3 candidates	≠
Possible Condorcet	=	=
Necessary Condorcet	=	=
Possible winners	≠	≠
Necessary winners	≠ = for 3 candidates	≠

Complexity and algorithms for

- Necessary Schwartz winner from profile for (simple) voting trees
- Possible and necessary winners from profile and from majority graph with voting trees

Winners over balanced agenda

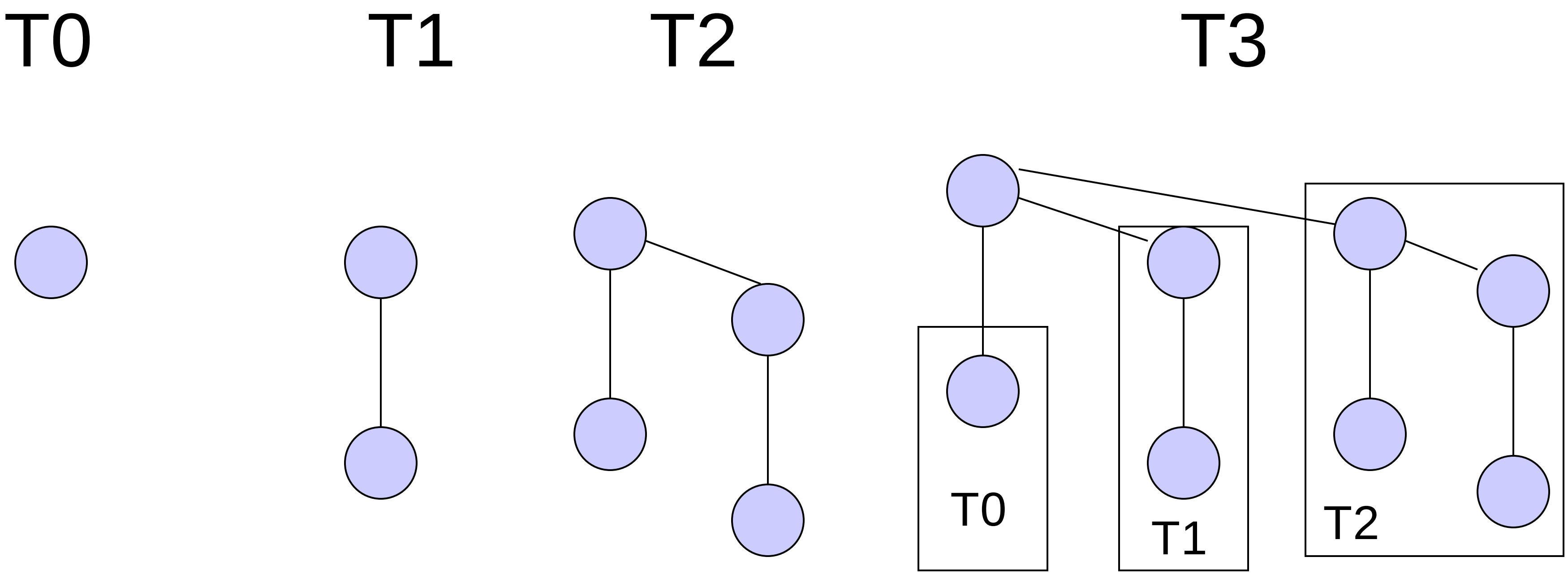
Computing winners for balanced agendas

- Given a **complete majority graph** G, A is a fair Schwartz winner if there is **a balanced tree** where A wins

- Given a majority graph G with **2^k nodes**, candidate A is a fair Schwartz winner iff it exists a binomial tree T_k:
  - Covering G (arrows from father to child)
  - Rooted at A

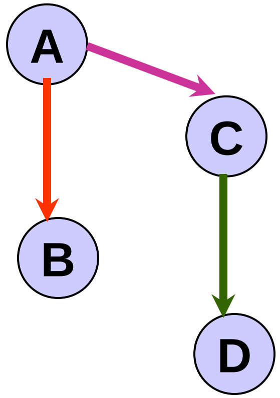
Binomial trees

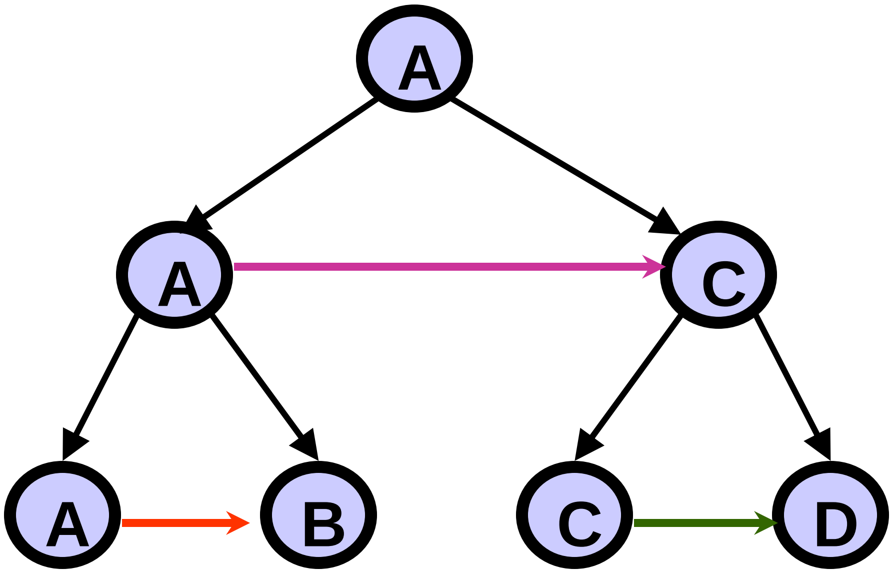
- Binomial tree
  - $T_0 \rightarrow$ 1 node
  - $T_K \rightarrow$ the root has k children and the i-th child is the root of a T_{k-i}
- T_k has 2^k nodes



From binomial tree to a balanced voting tree

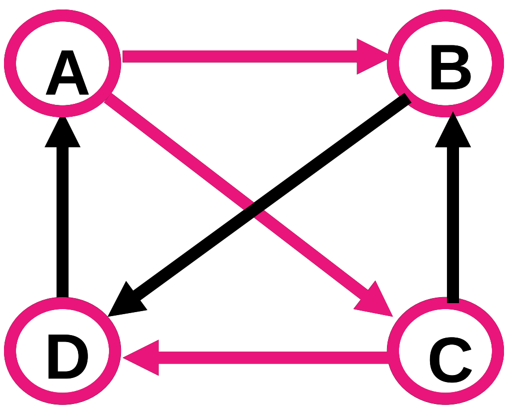
- Node of binomial tree ←→ leafs of voting tree
- Edge A→B: knock-out competition between A and B where A wins
- Incoming edge of leafs → initial knock-out competition

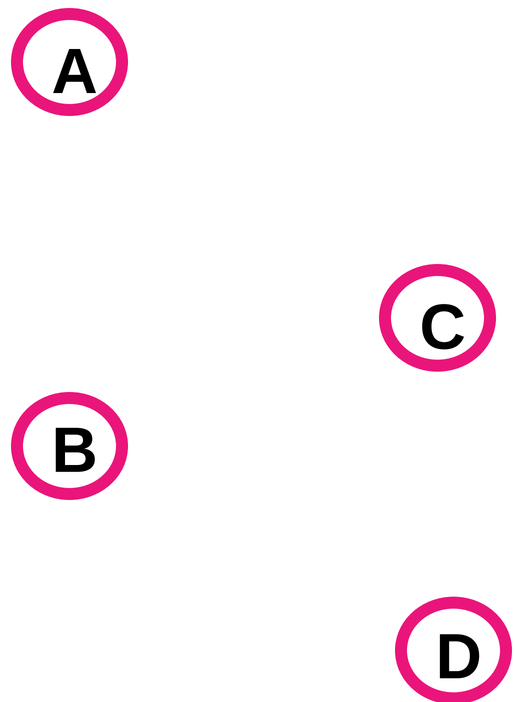


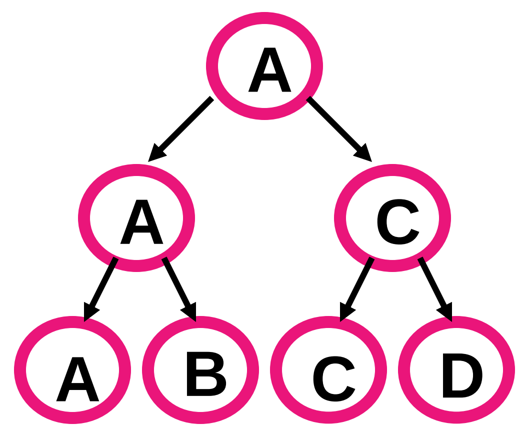


Determining fair Schwartz winners

- Given a majority graph G with 2^k nodes, candidate A is a fair possible winner iff it exists a binomial tree T_k:
  - Covering G (arrows from father to child)
  - Rooted at A







Complexity of determining fair Schwartz winners

- Th: “is A a fair Schwartz winner of minimum weight?” is NP-complete.
  - Proof: Polynomial reduction from the Exact Cover problem.

- Weighted majority graphs are used in social choice theory
  - weights may represent, for example, the amount of disagreement

Variants of classical possible & necessary winner problems

Unique winner and co-winner

- C: a candidate
  - Unique winner: C is the unique winner
  - Co-winner: C is in the set of winners

- Possible co-winner
- Possible unique winner

- Necessary co-winner
- Necessary unique winner

Unbounded n. of candidates, **unweighted votes**

	Possible winner	Necessary winner
STV	NP-complete (Bartholdi, Orlin 1991)	coNP-complete (Bartholdi, Orlin 1991)
Plurality	P	P
Veto	P	P
Pos. Scoring	NP-complete	P
Copeland	NP-complete	coNP-complete
Maximim	NP-complete	P
Bucklin	NP-complete	P
Ranked Pairs	NP-complete	coNP-complete
Voting trees	NP-complete	coNP-complete
Plurality with runoff	NP-complete (unique winner) P (co-winner)	P (unique winner) coNP-complete (co-winner)

Conitzer, Xia. Determining Possible and Necessary Winners Given Partial Orders. Journal of Artificial Intelligence Research 2011

New candidates

In some voting situations, some new candidates may show up in the course of the process

We may want to determine **which of the initial candidates are possible winners**, given that a fixed number k of new candidates will be added

Example: *suppose that*

- *the voters' preferences about a set of initial alternatives have already been elicited*
- *we know that a given number k of new alternatives will join the election*
- *we ask who among the initial alternatives can possibly win the election in the end*

New candidates: complexity results for scoring rules

- Question: what is the complexity of deciding if x is a possible winner with respect to the addition of three new candidates?

Voting rule	Possible winner
Borda	P
Plurality	P
Veto	P
3-approval	NP-complete

Chevaleyre et al. Possible Winners when New Candidates Are Added: The Case of Scoring Rules. AAAI 2010 and submitted to MSS 2010

New candidates: complexity results for other voting rules

Voting rule	Possible winner
Approval	P (def1) NP-complete (def2)
Bucklin	NP-complete
Copeland_0	NP-complete
Simpson (aka maximin)	NP-complete
Plurality with runoff	P

All NP-hardness results are proved by reductions from the Exact Cover problem (denoted by X3C)

Xia, Lang, Monnot. Possible Winners when New Alternatives join: New results coming up. AAMAS 2011

Approval definitions

- Definition 1 assumes that the threshold approved/unacceptable cannot move
  - any alternative approved in C is still approved in C′ (the extension of C)

- Definition 2 assumes that the threshold can stay the same or move upward (because the set of alternatives grows)
  - Some alternatives approved initially may be disapproved

Xia, Lang, Monnot. Possible Winners when New Alternatives join: New results coming up. AAMAS 2011

Possible and necessary winners of partial tournament (aka incomplete majority graph)

Voting rule	Possible winner	Necessary winner
Copeland	P	P
Uncovered set	P	P
Borda*	P	P
Maximin*	P	P
Ranked pairs*	NP-complete	NP-complete

* = for weighted tournament

H. Aziz, M. Brill, F. Fischer, P. Harrenstein, J. Lang, and H. G. Seedig. **Possible and necessary winners of partial tournaments**. AAMAS 2012

Other related papers on possible/necessary winners

1. Elkind et al. **Cloning in Elections: Finding the Possible Winners.** J. Artif. Intell. Res. (JAIR) 42: 529-573 (2011)
   - It considers the problem of manipulating elections by cloning candidates

2. Baumeister et al. **The Possible Winner Problem with Uncertain Weight.** ECAI'12
   - It considers elections where not some of the voters' preferences, but some of their weights, are uncertain.

3. Edith and Erdeli: **Manipulation Under Voting Rule Uncertainty.** AAMAS'12
   - the manipulator(s) know that the election will be conducted using a voting rule from a given list, and need to select their votes so as to succeed no matter which voting rule will eventually be chosen

Related papers on control

- Erdéli et al. The complexity of voter partition in Bucklin and fallback voting: solving three open problems. AAMAS 2011: 837-844

- Hemaspandra et al.: Online control ECAI 2012

- Faliszewski et al. The shield that never was: Societies with single-peaked preferences are more open to manipulation and control. Inf. Comput. 209(2): 89-107 (2011)

Related papers on bribery

- P. Faliszewski. Nonuniform bribery. AAMAS 2008, pp.1569–1572, 2008.

- Faliszewski et al. :How Hard Is Bribery in Elections? J. Artif. Intell. Res. (JAIR) 35: 485-532 (2009)

COMPUTATIONAL SOCIAL CHOICE

Thank you!

PhD course in Computer Science
University of Bologna & University of Padova
June 2012

Maria Silvia Pini (pini@dei.unipd.it)