Régles de vote : calcul, communication, vérification

Jérôme Lang
LAMSADE
CNRS – Université Paris-Dauphine, PSL

Maths en Mouvement 2022

Voting

1. n voters
2. m candidates (C = set of candidates)
3. a *profile* = a collection of n preference relations (rankings)

$$P = (\succ_1, \ldots, \succ_n)$$

$\succ_i$ *vote* expressed by voter i.

Here is a 100-voter profile over $C = \{a, b, c, d, e\}$

33 votes:	$a \succ b \succ c \succ d \succ e$
16 votes:	$b \succ d \succ c \succ e \succ a$
3 votes:	$c \succ d \succ b \succ a \succ e$
8 votes:	$c \succ e \succ b \succ d \succ a$
18 votes:	$d \succ e \succ c \succ b \succ a$
22 votes:	$e \succ c \succ b \succ d \succ a$

Positional scoring rules

- n voters, m candidates
- fixed list of m integers $s_1 \geq \ldots \geq s_m$, with $s_1 > s_m$
- if voter i ranks candidate x in position j then $score_i(x) = s_j$
- winner(s): candidate(s) maximizing

$$s(x) = \sum_{i=1}^{n} score_i(x)$$

plurality $s_1 = 1,\ s_2 = \ldots = s_m = 0 \mapsto$ winner: a

Borda $s_1 = m-1,\ s_2 = m-2,\ \ldots s_m = 0 \mapsto$ winner: b

- How many steps does an algorithm need to compute the winner(s)?

Positional scoring rules

- n voters, m candidates
- fixed list of m integers $s_1 \geq \ldots \geq s_m$, with $s_1 > s_m$
- if voter i ranks candidate x in position j then $score_i(x) = s_j$
- winner(s): candidate(s) maximizing

$$s(x) = \sum_{i=1}^{n} score_i(x)$$

plurality $s_1 = 1,\ s_2 = \ldots = s_m = 0 \ \mapsto$ winner: a

Borda $s_1 = m-1,\ s_2 = m-2,\ \ldots s_m = 0 \mapsto$ winner: b

- How many steps does an algorithm need to compute the winner(s)?
- $O(nm)$ in the general case (sometimes less)
- Positional rules are polynomial-time computable

Majority graph

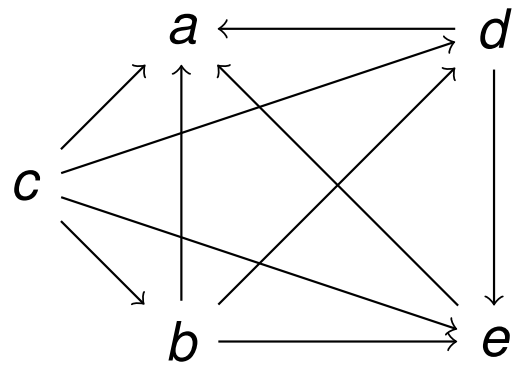
pairwise majority

given any two alternatives $x, y \in X$, use the majority rule to determine whether the group prefers x to y or vice versa.

Does this work? Sometimes yes:

33 votes:	$a \succ b \succ c \succ d \succ e$
16 votes:	$b \succ d \succ c \succ e \succ a$
3 votes:	$c \succ d \succ b \succ a \succ e$
8 votes:	$c \succ e \succ b \succ d \succ a$
18 votes:	$d \succ e \succ c \succ b \succ a$
22 votes:	$e \succ c \succ b \succ d \succ a$

associated majority graph



Collective preference relation: $c \succ b \succ d \succ e \succ a$

Winner: c

Majority graph

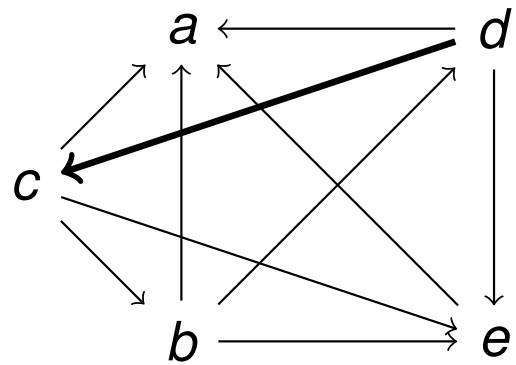
pairwise majority

given any two alternatives $x, y \in X$, use the majority rule to determine whether the group prefers x to y or vice versa.

Does this work? Sometimes no:

33 votes:	$a \succ b \succ \mathbf{d} \succ \mathbf{c} \succ e$
16 votes:	$b \succ d \succ c \succ e \succ a$
3 votes:	$c \succ d \succ b \succ a \succ e$
8 votes:	$c \succ e \succ b \succ d \succ a$
18 votes:	$d \succ e \succ c \succ b \succ a$
22 votes:	$e \succ c \succ b \succ d \succ a$

associated majority graph

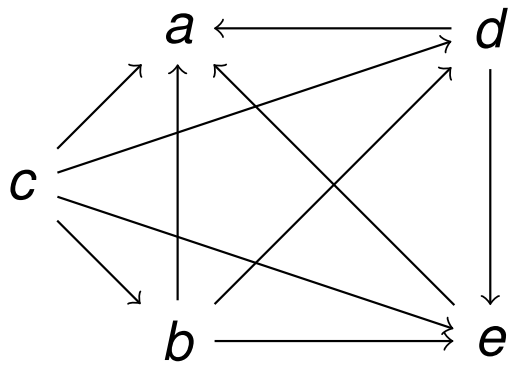


Collective preference relation: $\{b \succ c \succ d \succ b \succ \ldots\} \succ e \succ a$;

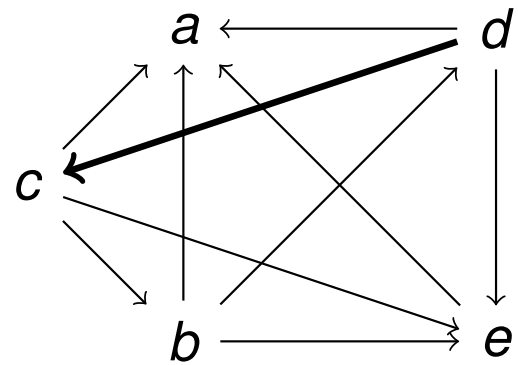
Winner: ?

Condorcet winner

- $N(x,y) = \#\{i, x \succ_i y\}$ number of voters who prefer x to y.
- x *Condorcet winner* if for all $y \neq x$, $N(x,y) > \frac{n}{2}$



c Condorcet winner



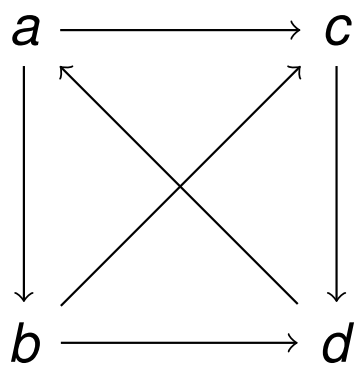
no Condorcet winner

- sometimes there is no Condorcet winner
- when there is a Condorcet winner, it is unique
- a rule is *Condorcet-consistent* if it outputs the Condorcet winner whenever there is one.

The Copeland rule

- P profile $\mapsto$ $M(P)$ directed graph associated with P
- A voting rule r is *based on the majority graph* if $r(P) = f(M(P))$ for some function f.
- For simplicity, assume an odd number of voters: the majority graph is a complete asymmetric graph (a *tournament*).

- $Cop(x)$ = number of candidates y such that $M(P)$ contains $x \longrightarrow y$.
- Copeland winner(s): $\operatorname{argmax}_{c \in C} Cop(x)$.



$$C(a) = 2$$
$$C(b) = 2$$
$$C(c) = 1$$
$$C(d) = 1$$

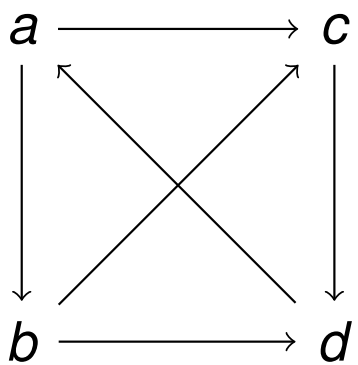
Copeland winners: a, b

The Copeland rule

- Computing the Copeland winner(s)
  1. construct the majority graph: for each pair (x, y), determine the direction of the edge. $O(nm^2)$
  2. for each candidate, compute its Copeland score and output the candidate(s) with largest score $O(m \log m)$
- complexity: $O(nm^2)$
- The Copeland rule is polynomial-time computable

The Banks rule

- look for the maximal subsets C' or C such that the restriction of $M(P)$ to C' is transitive.
- the restriction of $M(P)$ to these subsets are called *maximal transitive subtournaments* of $M(P)$
- x is a Banks winner if x is dominating in some maximal subtournament of $M(P)$.



Maximal subtournaments of $M(P)$:

- $\{a, b, c\}$ winner: a
- $\{b, c, d\}$ winner: b
- $\{a, d\}$ winner: d

Banks winners: a, b, d

The Banks rule

Computing *one* Banks winner:

- construct the majority graph $M(P)$
- take an arbitrary order of candidates $x_1 \dots x_m$
- $S \leftarrow \emptyset$

  for $i = 1 \dots m$:

  if the restriction of $M(P)$ to $S \cup \{x_i\}$ is transitive
  then $S \leftarrow S \cup \{x_i\}$
  end if

  end for
- return the dominating candidate in S

Complexity: $O(nm^2 + m^3)$

The Banks rule

Computing *all* Banks winners:

- construct the majority graph
- find all maximum transitive subtournaments
- for each of them, determine the winner
- return all winners

Complexity: $O(nm^2 + 2^m)$

The Banks rule

- *some* Banks winner can be found in polynomial time
- finding *all* seemingly needs exponential time
- even finding out whether a given candidate x is a Banks winner seemingly needs exponential time
- this problem is NP-hard
- if x is a Banks winner then there is a succinct certificate that proves it: a maximal subtournament in which x is dominating
- this problem is in NP
- finding out whether a given candidate x is a Banks winner is both in NP and NP-hard: it is NP-complete

The maximin rule

- $N_P(x,y) = \#\{i, x \succ_i y\}$ number of voters who prefer x to y.
- A voting rule r is *based on the weighted majority graph* if $r(P) = g(N_P)$ for some function g.

- maximin score: $S_m(x) = \min_{y \neq x} N_P(x,y)$
- winner(s) maximize $S_m(x)$

N_P	a	b	c	d	e	S_m
a	–	33	33	33	36	33
b	67	–	49	79	52	49
c	67	51	–	33	60	33
d	67	21	67	–	70	21
e	66	48	40	30	–	30

maximin winner: b

Computing the maximin rule: $O(nm^2)$

The Kemeny rule

- for each ranking of candidates R, the Kemeny score of R is

$$K(R) = \sum \{N_P(x,y) \mid (x,y) \text{ such that } x >_R y\}$$

- Kemeny consensus = ranking with maximal Kemeny score
- Kemeny winner: best candidate in some Kemeny consensus

N_P	a	b	c	d	e
a	–	33	33	33	36
b	67	–	49	79	52
c	67	51	–	33	60
d	67	21	67	–	70
e	66	48	40	30	–

$K(bcdea) = ?$

The Kemeny rule

- for each ranking of candidates R, the Kemeny score of R is

$$K(R) = \sum_{x,y:x>_R y} N(x,y)$$

- Kemeny consensus = ranking with maximal Kemeny score
- Kemeny winner: best candidate in some Kemeny consensus

N_P	a	b	c	d	e
a	–	33	33	33	36
b	67	–	49	79	52
c	67	51	–	33	60
d	67	21	67	–	70
e	66	48	40	30	–

$$K(bcdea) = 610$$

The Kemeny rule

- for each ranking of candidates R, the Kemeny score of R is

$$K(R) = \sum_{x,y|x>_R y} N(x,y)$$

- Kemeny consensus = ranking with maximal Kemeny score
- Kemeny winner: best candidate in some Kemeny consensus

N_P	a	b	c	d	e
a	–	33	33	33	36
b	67	–	49	79	52
c	67	51	–	33	60
d	67	21	67	–	70
e	66	48	40	30	–

$K(bcdea) = 610$
$K(bdcea) = 644$
Kemeny consensus: $bdcea$
Kemeny winner: b

The Kemeny rule

Computing the Kemeny rule:

- for each ranking R, compute $K(R)$: 2^m iterations
- determining whether x is a Kemeny winner is NP-hard
- can we find a Kemeny winner in polynomial time?
- if x is a Kemeny winner, is there a succinct certificate for x?
- *most probably not*: winner determination needs logarithmically many NP-oracles
- the problem is Θ_2^{P}-complete

The Kemeny rule

Computing the Kemeny rule:

- Polynomial approximation
  - a $4/3$-approximation algorithm based on linear programming
  - a $11/7$-approximation algorithm (more sophisticated)
  - existence of a polynomial-time approximation scheme (but not efficient in practice)
- Parameterized complexity

  Winner can be computed in time $O(2^m m^2 n)$
- Practical algorithms
  - translation into integer linear programming
  - branch and bound,
  - heuristic search based on Borda scores

Computing voting rules

Three classes of rules:

- winner determination in P: easy to compute
  - positional scoring rules, Copeland, maximin, ...
- winner determination is NP-complete: not easy to compute but easy to verify a solution using a succinct certificate
  - Banks, ...
- winner determination is beyond NP: not even easy to verify.
  - Kemeny, ...

Is there a life after NP-hardness?

- efficient computation: design algorithms that do as well as possible, possibly using heuristics, or translations into well-known frameworks (such as integer linear programming).
- fixed-parameter complexity: isolate the components of the problem and find the main cause(s) of hardness.
- approximation: design algorithms that produce a (generally suboptimal) result, with some performance guarantee.
  - The approximation of a voting rule is a new voting rule that may be interesting *per se*.

Multiple round rules

Plurality with runoff

- let x, y the two candidates with the highest plurality score (use tie-breaking rule if necessary)
- winner: majority winner between x and y

33	a $\succ$b $\succ$c $\succ$d $\succ$e
16	b $\succ$d $\succ$c $\succ$e $\succ$a
3	c $\succ$d $\succ$b $\succ$a $\succ$e
8	c $\succ$e $\succ$b $\succ$d $\succ$a
18	d $\succ$e $\succ$c $\succ$b $\succ$a
22	e $\succ$c $\succ$b $\succ$d $\succ$a

- first step: keep a and e
- winner: e

Multiple round rules

Single transferable vote (STV)

Repeat

$x :=$ candidate ranked first by the fewest voters;
eliminate x from all ballots
{votes for x transferred to the next best remaining candidate}

Until some candidate y is ranked first by more than half of the votes;
Winner: y

- When there are only 3 candidates, STV coincides with plurality with runoff.
- STV is used for political elections in several countries (at least Australia and Ireland)

Single transferable vote (STV)

33	a $\succ$ b $\succ$ c $\succ$ d $\succ$ e
16	b $\succ$ d $\succ$ c $\succ$ e $\succ$ a
3	c $\succ$ d $\succ$ b $\succ$ a $\succ$ e
8	c $\succ$ e $\succ$ b $\succ$ d $\succ$ a
18	d $\succ$ e $\succ$ c $\succ$ b $\succ$ a
22	e $\succ$ c $\succ$ b $\succ$ d $\succ$ a

33	a $\succ$ b $\succ$ d $\succ$ e
16	b $\succ$ d $\succ$ e $\succ$ a
3	d $\succ$ b $\succ$ a $\succ$ e
8	e $\succ$ b $\succ$ d $\succ$ a
18	d $\succ$ e $\succ$ b $\succ$ a
22	e $\succ$ b $\succ$ d $\succ$ a

33	a $\succ$ d $\succ$ e
16	d $\succ$ e $\succ$ a
3	d $\succ$ a $\succ$ e
8	e $\succ$ d $\succ$ a
18	d $\succ$ e $\succ$ a
22	e $\succ$ d $\succ$ a

33	a $\succ$ d
16	d $\succ$ a
3	d $\succ$ a
8	d $\succ$ a
18	d $\succ$ a
22	d $\succ$ a

winner: *d*

Single transferable vote (STV)

(*) How do we handle ties in STV?

STV^T ties are broken immediately using a tie-breaking priority T: polynomial

STV^{PU} exploring all possibilities and possible use tie-breaking at the very last moment: NP-complete

4	$a \succ d \succ b \succ c$
3	$b \succ c \succ d \succ a$
2	$c \succ d \succ a \succ b$
2	$d \succ b \succ c \succ a$

Tie-breaking :
$a > b > d > c$

- break ties immediately: c eliminated, then b, winner: d
- parallel universes:
  - branch 1 (above): winner: d
  - branch 2: d eliminated, then c, winner: a
  - winners $\{a, d\}$, winner: a.

Computing voting rules

Three classes of rules:

- winner determination in P: easy to compute
  - positional scoring rule, Copeland, maximin, plurality with runoff, STV^T, and others
- winner determination is NP-complete: not easy to compute but easy to verify a solution using a succinct certificate
  - Banks, STV^{PU}, and others
- winner determination is beyond NP: not even easy to verify.
  - Kemeny, Slater, and others

Communication complexity of voting rules

- *Voting rule*
  - profile $(V_1, \ldots, V_n) \mapsto$ winner(s) $r(V_1, \ldots, V_n)$
  - does not specify how the votes V_i are elicited from the voters by the central authority.
- *Protocol for a voting rule r*
  - informally: similar to an algorithm, except that instructions are replaced by communication actions, and such that communication actions are based on the *private information* of the agents.
  - V_i is the private information of agent (voter) i.
- *Communication complexity of a voting rule r*:
  - minimum cost of a protocol for r.

Communication complexity of voting rules

- An obvious protocol that works for *any* voting rule r:
  1. every voter i sends her vote V_i to the central authority
  2. the central authority sends back the name of the winner to all voters

  - step 1: $n \log(m!) = O(nm \log m)$ bits
  - step 2: ignored (or else: $n \log m$ bits)
    from now on, we shall ignore step the cost of information flow from the central authority to the voters.
- The communication complexity of an arbitrary voting rule r is in $O(nm \log m)$

Communication complexity: plurality with runoff

- An easy protocol for plurality with runoff:
  1. voters send the name of their most preferred candidate to the central authority
  2. the central authority sends the names of the two finalists to the voters
  3. voters send the name of their preferred finalist to the central authority

  - step 1: $n \log m$ bits
  - step 2: ignored (or else: $2n \log m$ bits)
  - step 3: n bits
  - total: $O(n \log m)$
  - lower bound matches (Conitzer & Sandholm, 05)
- the communication complexity of plurality with runoff is in $\Theta(n \log m)$

Communication complexity: STV

- A protocol for STV (Conitzer & Sandholm, 05)
  1. voters send their most preferred candidate to the central authority (C)
  2. let x be the candidate ranked first in the smallest number of votes. All voters who had x ranked first receive a message from C asking them to send the name of their next preferred candidate.
  3. repeat step 2 until there is a candidate ranked first in a majority of votes

  - after doing t times step 2: x ranked first in at most $\frac{n}{m-t}$ votes
  - cost of protocol

$$\leq n \log m(1 + 1/m + 1/m-1 + \ldots + 1/2) = O(n(\log m)^2)$$

  - lower bound $\Omega(n \log m)$
  - gap still open!

Multiwinner voting rules

- n voters
- m candidates (C = set of candidates)
- k number of candidates to be elected

Condorcet winning sets

- S is a Condorcet winning set for profile P if for each $x \notin S$, a majority of votes rank at least one element of S above x.
- $P = (abcd, cdab, dabc)$:
  - $\{a, c\}$ is a Condorcet winning set;
  - $\{b, c\}$ is not a Condorcet winning set because of a.
- Condorcet dimension of a profile P = cardinality of the smallest Condorcet winning set for P

Condorcet winning sets

- S is a Condorcet winning set for profile P if for each $x \notin S$, a majority of votes rank at least one element of S above x.
- $P = (abcd, cdab, dabc)$:
  - $\{a, c\}$ is a Condorcet winning set;
  - $\{b, c\}$ is not a Condorcet winning set because of a.
- Condorcet dimension of a profile P = cardinality of the smallest Condorcet winning set for P

- P has Condorcet dimension 1 if it has a Condorcet winner

Condorcet winning sets

- S is a Condorcet winning set for profile P if for each $x \notin S$, a majority of votes rank at least one element of S above x.
- $P = (abcd, cdab, dabc)$:
  - $\{a, c\}$ is a Condorcet winning set;
  - $\{b, c\}$ is not a Condorcet winning set because of a.
- Condorcet dimension of a profile P = cardinality of the smallest Condorcet winning set for P

- P has Condorcet dimension 1 if it has a Condorcet winner
- $P = (abcd, cdab, dabc)$ has Condorcet dimension 2: no Condorcet winner; $\{a, c\}$ Condorcet winning set.

Condorcet winning sets

- S is a Condorcet winning set for profile P if for each $x \notin S$, a majority of votes rank at least one element of S above x.
- $P = (abcd, cdab, dabc)$:
  - $\{a, c\}$ is a Condorcet winning set;
  - $\{b, c\}$ is not a Condorcet winning set because of a.
- Condorcet dimension of a profile P = cardinality of the smallest Condorcet winning set for P

- P has Condorcet dimension 1 if it has a Condorcet winner
- $P = (abcd, cdab, dabc)$ has Condorcet dimension 2: no Condorcet winner; $\{a, c\}$ Condorcet winning set.
- there exists a 6-candidate 6-voter profile of Condorcet dimension 3.

Condorcet winning sets

- S is a Condorcet winning set for profile P if for each $x \notin S$, a majority of votes rank at least one element of S above x.
- $P = (abcd, cdab, dabc)$:
  - $\{a, c\}$ is a Condorcet winning set;
  - $\{b, c\}$ is not a Condorcet winning set because of a.
- Condorcet dimension of a profile P = cardinality of the smallest Condorcet winning set for P

- P has Condorcet dimension 1 if it has a Condorcet winner
- $P = (abcd, cdab, dabc)$ has Condorcet dimension 2: no Condorcet winner; $\{a, c\}$ Condorcet winning set.
- there exists a 6-candidate 6-voter profile of Condorcet dimension 3.
- does there exist a profile of Condorcet dimension n, for all n?

Condorcet winning sets

- S is a Condorcet winning set for profile P if for each $x \notin S$, a majority of votes rank at least one element of S above x.
- $P = (abcd, cdab, dabc)$:
  - $\{a, c\}$ is a Condorcet winning set;
  - $\{b, c\}$ is not a Condorcet winning set because of a.
- Condorcet dimension of a profile P = cardinality of the smallest Condorcet winning set for P

- P has Condorcet dimension 1 if it has a Condorcet winner
- $P = (abcd, cdab, dabc)$ has Condorcet dimension 2: no Condorcet winner; $\{a, c\}$ Condorcet winning set.
- there exists a 6-candidate 6-voter profile of Condorcet dimension 3.
- does there exist a profile of Condorcet dimension n, for all n? *Nobody knows.*

Compiling votes: what, why, how

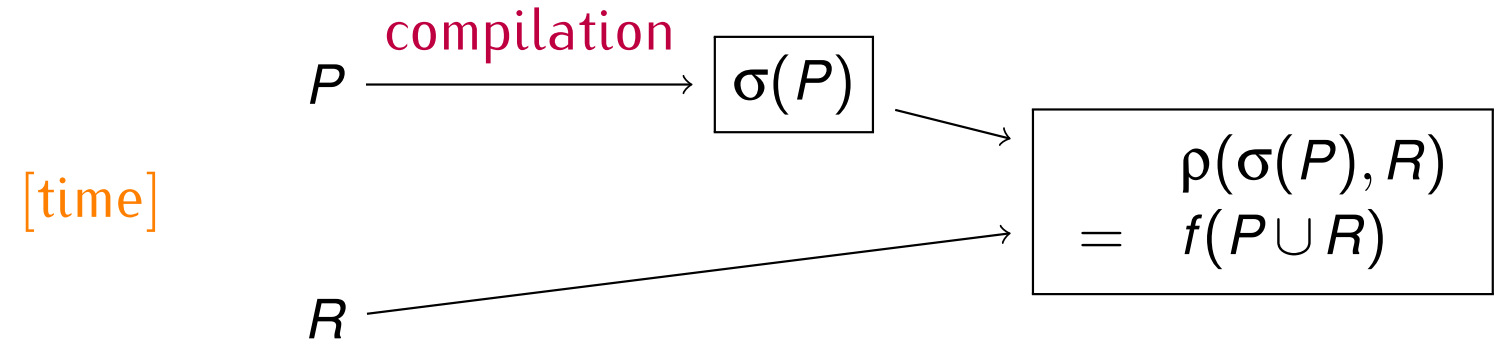
Two contexts, two motivations:

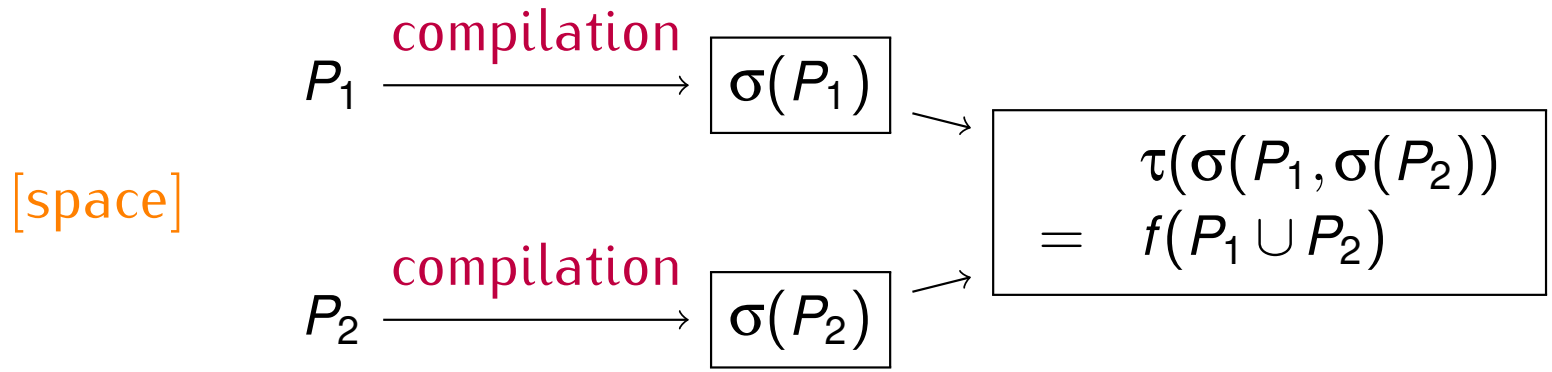
- time
  - votes may not come all together at the same time

    (who has ever organised a Doodle-like poll where everyone voted on time?)
  - preprocess the information given by the votes obtained so far, so as to "prepare the ground" for the time when the remaining votes are known, using *as little space as possible*
- space
  - the electorate is split into different districts (or voting polls)
  - each district counts its ballots separately and sends the outcome to the central authority, which determines the final outcome
  - which (minimal) information should the districts send?
  - useful for verification:
    - in each district, the voters can check the local results
    - local results are made public; they are sufficient for computing the final outcome
    - if storing the local results takes too much space, it is impractical to publish the results locally

Compilation functions for a voting rule

- f voting rule (resolute or irresolute)

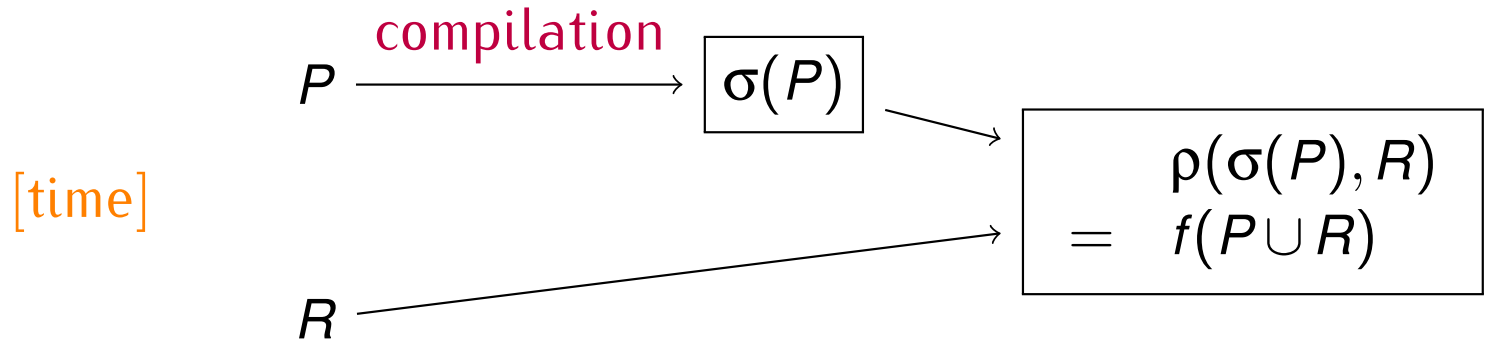




- σ is a compilation function for f if
  - time there is a function ρ such that $\rho(\sigma(P), R) = f(P \cup R)$ for all P, R.
  - space there is a function τ such that $\tau(\sigma(P_1), \ldots, \sigma(P_k)) = f(P_1 \cup \ldots \cup P_k)$ for all $P_1, \ldots, P_k$

Compilation functions for a voting rule

- f voting rule (resolute or irresolute)



- f = Borda
  - $\sigma(P)$: vector of (partial) Borda scores obtained from votes in P

$$P = \langle abc,\ abc,\ cba,\ bca \rangle \mapsto \sigma(P) = \langle a:3;\ b:5;\ c:3 \rangle$$

  - $\rho(\sigma(P), R) = \text{argmax}_{x \in X} (s_B(x \mid P) + s_B(x \mid R))$

$$R = \langle cab, abc \rangle \mapsto \langle a:3+3;\ b:5+1;\ c:3+2 \rangle \to \rho(\sigma(P), R) = b$$

Compilation complexity of a voting rule

- σ compilation function for f
- $Size(\sigma) = \max\{|\sigma(P)|,\ P \text{ partial profile}\}$
- compilation complexity of f:

$$C(f) = \min\{Size(\sigma) \mid \sigma \text{ compilation function for } r\}$$

- σ optimal compilation if $Size(\Sigma) = C(f)$
- $C(f)$ = size of an *optimal compilation function* for f = minimum space needed to compile P

Equivalent profiles for a voting rule

- f voting rule
- two partial profiles P and Q are *f-equivalent* (noted $P \sim_f Q$) if

$$\text{for every } R \text{ we have } f(P \cup R) = f(Q \cup R)$$

- $P = (abc, cba)$ and $Q = (bac, cab)$ are Borda-equivalent
- $P = (abc, bca)$ and $Q = (bac, cba)$ are not Borda-equivalent:
  - $R = (acb)$
  - $f(P \cup R) = \{a\}$
  - $f(Q \cup R) = \{a, b, c\}$

Compilation complexity

A useful result (similar result for one-round communication complexity, Kushilevitz & Nisan 97):

- f voting rule
- n number of initial voters, m number of candidates.
- **Theorem**: if the equivalence relation for f has $g(n,m)$ equivalence classes then

$$C(r) = \lceil \log g(n,m) \rceil$$

Consequences:

- for any f, $C(f) \leq n \log(m!)$
- if f is anonymous: $C(f) \leq \min(n \log(m!), m! \log n)$
- $C(\text{dictatorship}) = \log m$
- $C(f) = 0$ if and only if f is constant

Compilation complexity: positional scoring rules

- f_s induced by scoring vector s
- $S_P(x)$ score that x obtains from votes in P
- P and Q are equivalent *iff* for all x, $S_P(x) = S_Q(x)$
- optimal compilation: $(S_P(x))_{x \in X}$
- number of equivalence classes = number of ways of distributing scores to candidates

In particular:

$$C(\text{plurality}) = \Theta\left(m \log \frac{n}{m} + n \log \frac{m}{n}\right)$$

$$C(\text{Borda}) = \Theta(m \log nm)$$

Compilation complexity: tournament solutions

- M_P majority graph for P
- f based on the majority graph: $f(P) = h(M_P)$
- is $P \mapsto M_P$ a compilation function for f?

Compilation complexity: tournament solutions

- M_P majority graph for P
- f based on the majority graph: $f(P) = h(M_P)$
- is $P \mapsto M_P$ a compilation function for f? no because knowing M_P does not allow us to determine $M_{P \cup R}$
- m_P pairwise majority matrix (weighted majority graph)
- $m_P = m_Q$ implies $P \sim_f Q$.

Compilation complexity: tournament solutions

- M_P majority graph for P
- f based on the majority graph: $f(P) = h(M_P)$
- is $P \mapsto M_P$ a compilation function for f? no because knowing M_P does not allow us to determine $M_{P \cup R}$
- m_P pairwise majority matrix (weighted majority graph)
- $m_P = m_Q$ implies $P \sim_f Q$.
- the converse does not hold (e.g., constant rules)

Compilation complexity: tournament solutions

- M_P majority graph for P
- f based on the majority graph: $f(P) = h(M_P)$
- is $P \mapsto M_P$ a compilation function for f? no because knowing M_P does not allow us to determine $M_{P \cup R}$
- m_P pairwise majority matrix (weighted majority graph)
- $m_P = m_Q$ implies $P \sim_f Q$.
- the converse does not hold (e.g., constant rules)
- if f is Condorcet-consistent: if $P \sim_f Q$ then $m_P = m_Q$
- If f is a tournament solution (Condorcet-consistent + based on the majority graph):
  - $P \sim_f Q$ *iff* $m_P = m_Q$
  - numer of equivalence classes = number of pairwise majority matrices that are implemented by some n-voter profile
  - $C(f) = \Theta(m^2 \log n)$

Compilation complexity: tournament solutions

- m_P pairwise majority matrix (weighted majority graph)
- f is based on the pairwise majority matrix: $f(P) = h(m_P)$
- $m_P = m_Q$ *implies* $P \sim_f Q$.
- if f is a weighted tournament solution (based on pmm + Condorcet-consistent):
  - $P \sim_f Q$ if and only if $m_P = m_Q$
  - $C(f) = \Theta(m^2 \log n)$

Compilation complexity: plurality with runoff

- $P \sim_{\text{pl-ro}} Q$ if and only if these two conditions hold:
  - for every x, $ntop(P, x) = ntop(Q, x)$
  - $m_P = m_Q$
- optimal compilation: plurality scores + pairwise majority matrix
- $C(\text{pl-ro}) = \Theta(m^2 \log n)$

Compilation complexity: STV

- for $Z \subseteq X$ and profile P, let P^{-Z} be obtained from P by removing candidates in Z
- $ntop(P^{-Z}, x)$ number of votes in P^{-Z} who rank x on top
- $P \sim_{STV} Q$ iff for all $Z \subseteq X$ and $x \notin Z$, $ntop(P^{-Z}, x) = ntop(Q^{-Z}, x)$
- optimal compilation: $ntop(P^{-Z}, x)_{Z \subseteq X, x \notin Z}$
- $C(STV)$: lower and upper bounds do not coincide; see the paper.

Advertising

- An experimental voting platform: *Whale* (developed by Sylvain Bouveret, LIG): `http://whale3.noiraudes.net/`
- Testing various voting rules for the French presidential elections (2027, 2022) `https://vote.imag.fr`