Compiling the votes of a subelectorate

Yann Chevaleyre (LAMSADE – CNRS & Université Paris-Dauphine)
Jérôme Lang (LAMSADE – CNRS & Université Paris-Dauphine)
Nicolas Maudet (LAMSADE – CNRS & Université Paris-Dauphine)
Guillaume Ravilly-Abadie (LAMSADE – CNRS & U. Paris-Dauphine)

In some voting contexts, the votes do not come all together at the same time:

- general elections in Italy: the votes of the citizens living abroad is known only a few days after the rest of the votes;
- choosing a date for a meeting: some participants express their preferences later than the others.

We might want to “preprocess” the information given by the subelectorate so as to “prepare the ground” for the time when the latecomers will have expressed their votes.

“Preparing the ground”?

- *minimize on-line time*: compile the information, using as much off-line time and space as needed, in such a way that once the newcomers have expressed their vote, the outcome can be computed *as fast as possible*.
  $\Rightarrow$ knowledge compilation (very relevant to voting, especially for hard voting rules – deserves another paper)
- *minimize storage space*: synthesize the information contained in the votes of the subelectorate, using *as less space as possible*, while keeping enough information so as to be able to compute the outcome once the newcomers have expressed their votes.
  $\Rightarrow$ the topic of this talk

A specific context where it is particularly useful to compile the vote of a subelectorate: *verification of the outcome of a vote by the population*.
A frequent situation:

- the electorate is split into different districts; each district counts its ballots separately and communicates the outcome to the central authority (e.g. the Ministry of Innner Affairs), which, after gathering the outcomes from all districts, determines the final outcome;
- in each district, the voters can check that the local results are sound;
- local results are made public and voters can check the final outcome from these local outcomes.

space needed to synthesize the votes of a district
= amount of information the district has to send to the central authority

If this amount of information is too large, it is impractical to publish the results locally, and therefore, difficult to check the final outcome and voters may be reluctant to accept the voting rule.

Related issues:

- *complexity of vote elicitation* (Conitzer & Sandholm 02): given a voting rule r, a set of known votes S, and a set of t new voters, is the outcome of the vote already determined from S?
- *computation of possible and necessary winners* (Konczak & Lang 05, Pini et al. 07, Walsh 08, Xia & Conitzer08): given a voting rule r, a set of incomplete votes, who are the candidates who can still possibly win the election, and is there a candidate who surely wins it?

$\Rightarrow$ when the outcome of the vote is already determined from S (= existence of a necessary winner), the space needed to synthesize the known votes is just the binary encoding of the winner.

- *communication complexity of voting rules* (Conitzer & Sandholm 05):

$\Rightarrow$ see later.

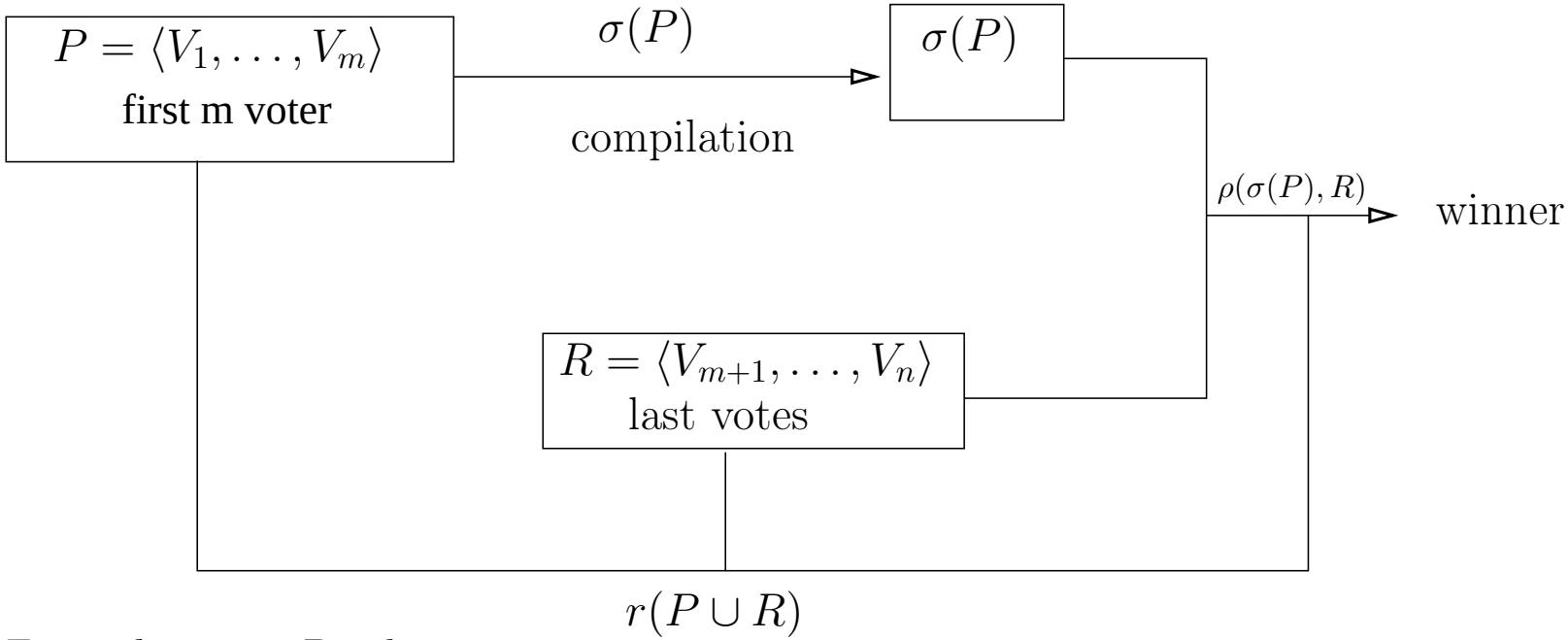
Voting rules

- X set of candidates; $p = |X|$.
- n voters.
- *vote* = linear order over X.
- $\mathcal{P}_X$ = set of all linear orders over X.
- *profile*: $P = \langle V_1, \ldots, V_n \rangle$ where each V_i is a vote.
- voting rule r: function from $\mathcal{P}_X^n$ to X.

The compilation problem:

Input only $m(\leq n)$ voters have expressed their vote. $P = \langle V_1, \ldots, V_m \rangle$ = **partial** profile obtained from these m voters.

Question what is the minimal size needed to compile P, while still being able to compute r when the last votes come in?



Example: r_B = Borda.

$\sigma(P)$: vector of partial Borda scores $\langle s_B(x \mid P) \rangle_{x \in X}$

$P = \langle abc, abc, cba, bca \rangle \mapsto \sigma(P) = \langle a : 4; b : 5; c : 3 \rangle$.

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

$R = \langle cab, abc \rangle \mapsto \langle a : 4 + 3; b : 5 + 1; c : 3 + 2 \rangle \mapsto \rho(\sigma(P), R) = a.$

Compilation function for (r, k):

$$\sigma : \mathcal{P}_X^m \to \{0,1\}^*$$

such that there exists a function

$$\rho : \{0,1\}^* \times \mathcal{P}_X^k \to X$$

with $\rho(\sigma(P), R) = r(P \cup R)$ for every $P \in \mathcal{P}_X^m$ and every $R \in \mathcal{P}_X^k$.

Similar definition for k not fixed.

Size of a compilation function
Let σ be a compilation function for (r, k).

$$Size(\sigma) = \max\{|\sigma(P)| \mid P \in \mathcal{P}_X^m\}$$

Compilation complexity of (r, k):

$$C(r, k) = \min\{Size(\sigma) \mid \sigma \text{ is a compilation function for } (r, k)\}$$

$C(r, k)$ is the minimum space needed to compile the m-voter partial profile P without knowing the remaining k-voter profile R
(does not take into account the off-line time needed to compute σ, nor the off-line time needed to compute ρ)
Similarly: $C(r) = \min\{Size(\sigma) \mid \sigma \text{ is a compilation function for } r\}$

Compilation complexity as one-round communication complexity

One-round communication complexity :

- two agents A and B have to compute a function f.
- each of them knows only a part of the input.
- *one-round protocol*: A sends only one message to B, and then B sends the output to A.
- *one-round communication complexity* of f: worst-case number of bits of the best one-round protocol for f.

One-round communication complexity $\approx$ compilation complexity

- A = set of voters having already expressed their votes
- B = set of remaining voters;
- compilation of the votes of A = information that A must send to B.
- minor difference: B does not send back the output to A.

Compilation complexity vs. standard communication complexity

Communication complexity of voting rules (Conitzer & Sandholm 05): given a voting rule r and a set of voters, what is the worst-case cost of the best protocol allowing to compute the outcome of the election? Major difference with communication complexity: the agents may use any protocol (not necessarily one-round).

Therefore: communication complexity is never smaller than standard communication complexity.

(However, comparing our lower bounds to those in (Conitzer & Sandholm 05) is not so simple – see the paper.)

Equivalent profiles for a voting rule

r voting rule;
k number of remaining voters.

- two partial profiles P and Q are (r,k)-*equivalent* if no matter the remaining unknown k votes, they will lead to the same outcome:

$$\text{for every } R \in \mathcal{P}_X^k \text{ we have } r(P \cup R) = r(Q \cup R)$$

- P and Q are r-*equivalent* if they are (r,k)-equivalent for every $k \geq 0$.

Example: r_P = plurality with tie-breaking priority order $b > a > c$.

- $\langle abc, abc, abc, abc\rangle$ and $\langle abc, abc, acb, acb\rangle$ are r_P-equivalent;
- $P_1 = \langle abc, abc, abc, abc\rangle$ and $P_2 = \langle abc, abc, abc, bca\rangle$ are (r_P,k)-equivalent for $k = 1$ but not for $k \geq 2$.
  For $k = 2$, take $R = \langle bca, bca\rangle$: $r_P(P_1 \cup R) = a \neq r_P(P_2 \cup R) = b$.

A useful result (similar to a result in (Kushilevitz & Nisan, 97)):

- r a voting rule.
- m number of initial voters
- p number of candidates.

If the number of equivalence classes for the r-equivalence relation is $g(m,p)$ then the compilation complexity of r is exactly $\lceil \log g(m,p) \rceil$. (A similar result holds for the compilation complexity of (r,k).)

Corollary:

- the communication complexity of a voting rule is $\leq m \log(p!)$;
- the communication complexity of an anonymous voting rule is $\leq \min(m \log(p!), p! \log m)$.
- the compilation complexity of a dictatorship is $\log p$;
- the compilation complexity of r is 0 if and only if r is constant.

Methodology

- seek a characterization of the equivalence classes
- count the number of equivalence classes

Case study 1: plurality

r_P = plurality

Lemma: for any partial m-voter profile P and $x \in X$, let $ntop(P, x)$ be the number of votes in P ranking x first.

P and P' are r_P-equivalent *iff* for every x, $ntop(P, x) = ntop(P', x)$.

Corollary: compilation complexity of $r_P = \lceil \log L(m, p) \rceil$ where

$L(m, p)$=

= number of vectors of positive integers $\langle \alpha_1, \ldots, \alpha_p \rangle$ s.t. $\sum_{i=1}^{p} \alpha_p = m$

= $\binom{p+m-1}{m}$

Corollary

The compilation complexity of r_P is $\Theta\left(p \log \frac{m}{p} + m \log \frac{p}{m}\right)$

Case study 2: Borda

r_B = Borda

Lemma:

Let $score_B(x, P)$ = Borda score of x obtained from the partial profile P

$$P \text{ and } P' \text{ are equivalent for } r_B \textit{ iff } \text{for every } x,$$
$$score_B(x, P) = score_B(x, P').$$

Theorem

The compilation complexity of r_B is at most $(p-1)\log m(p-1)$.

Lower bound: more difficult (idea = focus on a subset of vectors of Borda scores). See the paper.

Theorem

The compilation complexity of the Borda rule is $\Theta(p \log mp)$.

Case study 3: Rules based on the weighted majority graph

Let P be a (partial) profile and r a voting rule.

- $N_P(x, y)$ = number voters in P preferring x to y.
- *majority graph* M_P = directed graph whose set of vertices is X and containing an edge from x to y if and only if $N_P(x, y) > N_P(y, x)$.
- *weighted majority graph* $\mathcal{M}_P$: same as M_P, where each edge from x to y is weighted by $N(x, y)$
- r is *based on the majority graph* if $r(P)$ can be computed from M_P
- r *based on the weighted majority graph* if $r(P)$ can be computed from $\mathcal{M}_P$.

Case study 3: Rules based on the weighted majority graph

Lemma Let r based on the weighted majority graph. If $\mathcal{M}_P = \mathcal{M}_{P'}$ then P and P' are r-equivalent.
(Note: if r based on the (non-weighted) majority graph, we still need $\mathcal{M}_P$)

$T(m, p)$: number of all weighted tournaments on X that can be obtained as the weighted majority graph of some m-voter profile.

Theorem The compilation complexity of any rule based on the weighted majority graph is at most $\log T(m, p)$.

Case study 3: Rules based on the weighted majority graph

(The compilation complexity of any rule based on the weighted majority graph is at most $\log T(m,p)$)
Getting a lower bound is not possible without a further assumption on r.
Lemma For any Condorcet-consistent rule r, P r-equivalent to P' implies $\mathcal{M}_P = \mathcal{M}_{P'}$.
Therefore:

- the compilation complexity of a Condorcet-consistent rule is at least $\log T(m,p)$.
- if r is both Condorcet-consistent and based on the majority graph, then its compilation complexity is $\log T(m,p)$.
- the compilation complexity of the following rules is $\log T(m,p)$: *Copeland*, *Simpson/maximin*, *Slater*, *Banks*, *uncovered set*, *Schwartz*.

And finally: $\log T(m,p) = \Theta(p^2 \log m)$

Case study 4: Plurality with runoff

r_2 = plurality with runoff.

Lemma P and Q are r_2-equivalent *iff* this two conditions holds

- for every x, $ntop(P, x) = ntop(Q, x)$
- for every x, y, $N_P(x, y) = N_Q(x, y)$.

Theorem: the compilation complexity of plurality with runoff is $\log L(m, p) + \log T(m, p)$.

Further issues

- more results when the number of remaining voters is not fixed;
- for a given voting rule, determine the probability that
  - a voting process can be stopped after only m votes are known;
  - the central authority would make a mistake were it forced to commit on a winner in situations where no candidate is yet guaranteed to prevail;
- design new ways of computing NP-hard voting rules using an off-line compilation step so that their on-line computation time becomes polynomial in the size of the initial profile.