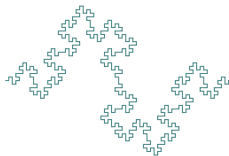


# Voting in a Dynamic World

Nicolas Maudet

*LAMSADE, Univ. Paris-Dauphine*

Joint work with: Y. Chevaleyre, J. Lang, J. Monnot, G. Ravilly-Abadie



May 3, 2011

# INTRODUCTION

INTRODUCTION

BACKGROUND: VOTING

INCOMPLETE PROFILES

MISSING VOTERS

MISSING CANDIDATES

RELATED WORKS

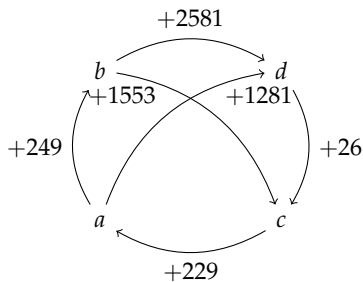
# VOTING RULES

Consider a set  $\mathcal{N} = \{1, \dots, n\}$  voters, and a set  $\mathcal{C}$  of  $p$  candidates

- ▶ a **vote** is a (strict) ranking of the different candidates
- ▶ a **profile**  $P$  is the collection of the votes of each agent
- ▶ a **voting rule**  $r$  maps a profile to single winning candidate, denoted  $r(P)$
- ▶ among common voting rules, we can distinguish:
  - ▶ scoring rules, where a vector of scores indicates the number of point(s) each position provides
  - ▶ rules based on pairwise comparisons of candidates
  - ▶ multiple round rules

## AN EXAMPLE

Take the following profile  $P$  involving 4703 voters and the corresponding (weighted) majority graph  $\mathcal{M}_P$



1340:	$a \succ b \succ c \succ d$
827:	$b \succ a \succ d \succ c$
75:	$d \succ a \succ b \succ c$
961:	$c \succ a \succ b \succ d$
1400:	$d \succ b \succ c \succ a$
100:	$d \succ c \succ a \succ b$

# VOTING WITH INCOMPLETE PROFILES

There are many cases where profiles can be **incomplete**

- (i) cannot compare candidates (**intrinsic** incompleteness)
- (ii) there are **too many** candidates to be ranked
- (iii) messages may be **lost, delayed, or faulty**

When profiles are incomplete, one may either rely on:

- ▶ further *communication* to elicitate the (relevant) missing information
- ▶ computation of *possible winner(s)*, i.e. candidates who win in at least one completion of the profile

# MISSING VOTERS AND MISSING CANDIDATES

In his general version, the problem of voting under incomplete preferences makes no assumption on incompleteness:

→ We have considered two specific sub-cases of the problem.

1	2	3	4	5
<i>a</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>a</i>
<i>b</i>	<i>c</i>	<i>a</i>	<i>a</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>c</i>	<i>b</i>	<i>c</i>

Missing voters:

1	2	3	4	5
<i>a</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>a</i>
<i>b</i>	<i>c</i>	<i>a</i>	<i>a</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>c</i>	<i>b</i>	<i>c</i>

Missing candidates:

1	2	3	4	5
<i>a</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>a</i>
<i>b</i>	<i>c</i>	<i>a</i>	<i>a</i>	<i>b</i>
<i>c</i>	<i>a</i>	<i>c</i>	<i>b</i>	<i>c</i>

# EXAMPLE: ON-LINE POLLS



Figure: Did everyone vote?

INTRODUCTION

BACKGROUND: VOTING

INCOMPLETE PROFILES

MISSING VOTERS

MISSING CANDIDATES

RELATED WORKS

# COMPILING PROFILES

Unknown number of missing voters: how to store the profile?

1	2	3	...
<i>a</i>	<i>b</i>	<i>b</i>	
<i>b</i>	<i>c</i>	<i>a</i>	
<i>c</i>	<i>a</i>	<i>c</i>	



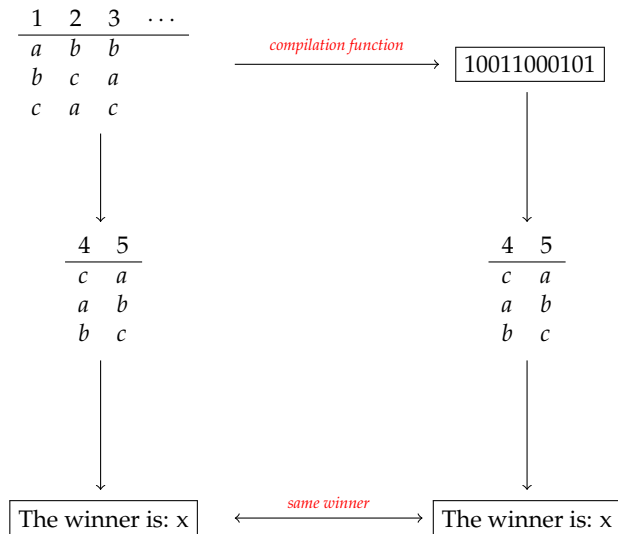
4	5
<i>c</i>	<i>a</i>
<i>a</i>	<i>b</i>
<i>b</i>	<i>c</i>



The winner is:  $x$

# COMPILING PROFILES

Unknown number of missing voters: how to store the profile?



# COMPILATION FUNCTIONS

We are after the best compilation functions for each voting rule. To start with, for any anonymous voting rule, compiling the profile into the corresponding **voting situation** is possible:

Profile:	$\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ \hline a & b & b & c & a \\ b & c & a & a & b \\ c & a & c & b & c \end{array}$	Voting situation:	$\begin{array}{cccc} 2 & 1 & 1 & 1 \\ \hline a & b & b & c \\ b & c & a & a \\ c & a & c & b \end{array}$
----------	--	-------------------	---

Hence the compilation requires at most  $\min(n \log p!, p! \log n)$ .

# COMPILATION FUNCTIONS

We are after the best compilation functions for each voting rule. To start with, for any anonymous voting rule, compiling the profile into the corresponding **voting situation** is possible:

Profile:	$\begin{matrix} 1 & 2 & 3 & 4 & 5 \\ a & b & b & c & a \\ b & c & a & a & b \\ c & a & c & b & c \end{matrix}$	Voting situation:	$\begin{matrix} 2 & 1 & 1 & 1 \\ a & b & b & c \\ b & c & a & a \\ c & a & c & b \end{matrix}$
----------	--	-------------------	--

Hence the compilation requires at most  $\min(n \log p!, p! \log n)$ .

- ▶ On our example, we get  $\min(4703 \log 24, 24 \log 4703)$ : the anonymous compilation is more efficient (312 bits *vs.* 23515 bits).

# COMPILATION FUNCTIONS FOR SPECIFIC RULES

- ▶ Intuitively, for specific voting rules one can get much better compilations, eg. for plurality just compile the score yields  $p \log n$ .
- ▶ But these are upper bounds: how do we know that no better compilation is possible by a very smart guy?
- ▶ Lower bounds: borrow notions from **communication complexity**.

# COMPILING PROFILES: METHODOLOGY

- ▶ two profiles are **equivalent** for a voting rule if they return the same winner for any possible completion.
- ▶ the key is to **characterize** equivalence classes for each rules, and **enumerate** them (not always easy...).
- ▶ the **compilation complexity** is given by taking the log of this number.

<i>Voting rule</i>	<i>Characterization of equiv.</i>	<i>Compilation complexity</i>
Any voting rule	same profiles	$O(np \log p)$
Anonymous	same voting situations	$O(p! \log n)$
STV	for all $Z \subseteq C$ and $x \notin Z$ , $score_{pl}(x, P^{-Z}) = score_{pl}(x, Q^{-Z})$	$\Omega(2^p \log n)$ $O(p2^p \log n)$
Plurality/runoff	$\mathcal{M}_P = \mathcal{M}_Q$ and $score_{pl}(x, P) = score_{pl}(x, Q)$	$\Theta(p^2 \log n)$
Cond. WMG	$\mathcal{M}_P = \mathcal{M}_Q$	$O(p^2 \log n)$
Borda	$score_B(x, P) = score_B(x, Q)$	$\Theta(p \log np)$
Plurality	$score_{pl}(x, P) = score_{pl}(x, Q)$	$\Theta\left(p \log\left(1 + \frac{n}{p}\right) + n \log\left(1 + \frac{p}{n}\right)\right)$

INTRODUCTION

BACKGROUND: VOTING

INCOMPLETE PROFILES

MISSING VOTERS

MISSING CANDIDATES

RELATED WORKS

# POSSIBLE WINNERS

- ▶ Given a voting situation  $\pi$  and a voting rules  $r$ ,  $x \in X$  is a possible winner (wrt  $\pi$  and  $r$ ) if there is a profile  $P$  extending  $P_X$  st.  $r(P) = x$

Note that the necessary winner problem is not very relevant here, any new candidate being (under mild conditions) a possible winner.

## Question

Study the possible winner problem with new candidates focusing on scoring rules  $\langle s_1, \dots, s_p \rangle$  ( $s_i \geq s_{i+1}$  and  $s_1 > s_p$ ).

## EXAMPLE

### Example (Plurality, 1 new candidate)

---

1:  $a \succ d \succ c \succ b$ 2:  $a \succ b \succ c \succ d$ 3:  $a \succ d \succ c \succ b$ 4:  $d \succ a \succ c \succ b$ 5:  $b \succ a \succ c \succ d$ 6:  $b \succ d \succ a \succ c$ 7:  $c \succ d \succ a \succ b$ 8:  $c \succ b \succ d \succ a$ Tie-breaking:  $a > b > c > d > y$ 

Plurality scores:

 $s(a) = 3, s(b) = 2, s(c) = 2, s(d) = 1$ 

Who are the possible winners?

certainly  $a$  is...

## EXAMPLE

## Example (Plurality, 1 new candidate)

1:  $a \succ d \succ c \succ b \succ y$ 2:  $y \succ a \succ b \succ c \succ d$ 3:  $y \succ a \succ d \succ c \succ b$ 4:  $d \succ a \succ c \succ b \succ y$ 5:  $b \succ a \succ c \succ d \succ y$ 6:  $b \succ d \succ a \succ c \succ y$ 7:  $c \succ d \succ a \succ b \succ y$ 8:  $c \succ b \succ d \succ a \succ y$ Tie-breaking:  $a > b > c > d > y$ 

Plurality scores:

 $s(a) = 3, s(b) = 2, s(c) = 2, s(d) = 1$ 

Who are the possible winners?

 $b$  is as well...

## EXAMPLE

## Example (Plurality, 1 new candidate)

1:  $a \succ d \succ c \succ b \succ y$ 2:  $y \succ a \succ b \succ c \succ d$ 3:  $y \succ a \succ d \succ c \succ b$ 4:  $d \succ a \succ c \succ b \succ y$ 5:  $y \succ b \succ a \succ c \succ d$ 6:  $b \succ d \succ a \succ c \succ y$ 7:  $c \succ d \succ a \succ b \succ y$ 8:  $c \succ b \succ d \succ a \succ y$ Tie-breaking:  $a > b > c > d > y$ 

Plurality scores:

 $s(a) = 3, s(b) = 2, s(c) = 2, s(d) = 1$ 

Who are the possible winners?

 $c$  is not.

## EXAMPLE

## Example (Plurality, 2 new candidates)

- 
- 1:  $a \succ d \succ c \succ b \succ y_1 \succ y_2$
- 2:  $y_1 \succ a \succ b \succ c \succ d \succ y_2$  Tie-breaking:  $a > b > c > d > y$
- 3:  $y_2 \succ a \succ d \succ c \succ b \succ y_1$
- 4:  $d \succ a \succ c \succ b \succ y_1 \succ y_2$  Plurality scores:
- 5:  $y_1 \succ b \succ a \succ c \succ d \succ y_2$   $s(a) = 3, s(b) = 2, s(c) = 2, s(d) = 1$
- 6:  $b \succ d \succ a \succ c \succ y_1 \succ y_2$  Who are the possible winners?
- 7:  $c \succ d \succ a \succ b \succ y_1 \succ y_2$  now  $c$  is.
- 8:  $c \succ b \succ d \succ a \succ y_1 \succ y_2$

## PLURALITY: AN EASY CASE

The general condition is easy to state. Intuitively:

- ▶ each new candidate can be placed at the top to decrease the score of a candidate;
- ▶ for each candidate with a higher score than  $x$  we must put the new candidate on top a number of times equal to the difference of scores (+1 if that candidate has priority in the tie-breaking rule);
- ▶ the score of the new candidate must not be higher (or indeed equal if the new candidate has priority) than the current score of  $x$ .

Generalizes to  $k$  new candidates.

$$\text{top}(P_X, x) \geq \frac{1}{k} \sum_{z \in X} \max(0, \text{top}(P_X, z) - \text{top}(P_X, x))$$

# BORDA

Consider the scoring vector  $\langle p-1, p-2, \dots, 0 \rangle$ .

For a given candidate  $x$ , the best situation is that the new candidates  $y_i$  are placed right after  $x$  in the profile.

$$\langle 4, 3, 2, 1, 0 \rangle$$

$$a \succ x \succ y \succ b \succ c$$

# BORDA

Consider the scoring vector  $\langle p - 1, p - 2, \dots, 0 \rangle$ .

For a given candidate  $x$ , the best situation is that the new candidates  $y_i$  are placed right after  $x$  in the profile.

$$\langle 4, 3, 2, 1, 0 \rangle$$

$$a \succ x \succ y \succ b \succ c$$

Holds in general for rules where  $\forall i : (s_i - s_{i+1}) \geq (s_{i+1} - s_{i+2})$

# BORDA

Consider the scoring vector  $\langle p-1, p-2, \dots, 0 \rangle$ .

For a given candidate  $x$ , the best situation is that the new candidates  $y_i$  are placed right after  $x$  in the profile.

$$\langle 4, 3, 2, 1, 0 \rangle$$

$$a \succ x \succ y \succ b \succ c$$

Holds in general for rules where  $\forall i : (s_i - s_{i+1}) \geq (s_{i+1} - s_{i+2})$   
 But the property doesn't hold if the score vector is "convex":

$$\langle 10, 3, 2, 1, 0 \rangle$$

$$a \succ x \succ y \succ b \succ c$$

# BORDA

Consider the scoring vector  $\langle p-1, p-2, \dots, 0 \rangle$ .

For a given candidate  $x$ , the best situation is that the new candidates  $y_i$  are placed right after  $x$  in the profile.

$$\langle 4, 3, 2, 1, 0 \rangle$$

$$a \succ x \succ y \succ b \succ c$$

Holds in general for rules where  $\forall i: (s_i - s_{i+1}) \geq (s_{i+1} - s_{i+2})$

But the property doesn't hold if the score vector is "convex":

$$\langle 10, 3, 2, 1, 0 \rangle$$

$$y \succ a \succ x \succ b \succ c$$

Maybe good to put  $y$  above  $x$  here...

# BORDA

This means that the condition is also easy to state.

A candidate can only gain points against another candidate when it is above. Let  $N(P_X, x, z)$  the number of times  $x$  this happens.

$$k \geq \max_{z \in X \setminus \{x\}} \frac{\max(0, s(P_X, z) - s(P_X, x))}{N(P_X, x, z)}$$

## EXAMPLE

## Example (Borda)

1:  $a \succ d \succ c \succ b$ 2:  $a \succ b \succ c \succ d$ 3:  $a \succ d \succ c \succ b$ 4:  $d \succ a \succ c \succ b$ 5:  $b \succ a \succ c \succ d$ 6:  $b \succ d \succ a \succ c$ 7:  $c \succ d \succ a \succ b$ 8:  $c \succ b \succ d \succ a$ 

Borda scores:

$$s(a) = 15, s(b) = 10, s(c) = 11, s(d) = 12$$

$$\delta(b, a) = (15 - 10)/3 = 5/3$$

$$\delta(b, c) = (11 - 10)/3 = 1/3$$

$$\delta(b, d) = (12 - 10)/4 = 1/2$$

Hence 2 new candidates are required for  $b$ .**And the possible winners are...**with 1 new candidate  $d$ with 2 new candidates  $b$  and  $c$

# $k$ -APPROVAL

$$\langle 1, \dots, 1, 0, \dots, 0 \rangle$$

For one candidate, the condition can be easily stated.

Intuitively:

- ▶ Only candidates lying in the last position of the approval set can be “pushed away”;
- ▶ Candidates with a higher score than  $x$  must appear in the last approved position sufficiently many times;
- ▶ Overall, the score of the new candidate must not exceed the current score of  $x$ .

# $k$ -APPROVAL

$$\langle 1, \dots, 1, 0, \dots, 0 \rangle$$

For one candidate, the condition can be easily stated.

Intuitively:

- ▶ Only candidates lying in the last position of the approval set can be “pushed away”;
- ▶ Candidates with a higher score than  $x$  must appear in the last approved position sufficiently many times;
- ▶ Overall, the score of the new candidate must not exceed the current score of  $x$ .

Generalizes to more than one new candidate ?

## 4-APPROVAL IS HARD

### Theorem

*Deciding if  $x$  is a possible winner for 4-approval w.r.t. the addition of 3 candidates is NP-complete*

**Proof (sketch):** Reduction to the perfect 3D-matching problem.  
Given: a collection of triples  $S_1 \cup \dots \cup S_n$  where  $S_i = (a_i, b_i, c_i)$ , find a perfect matching if there exists one.

## 4-APPROVAL IS HARD

$$\begin{array}{c}
 S_1 \succ a_1 \succ b_1 \succ c_1 \\
 S_2 \succ a_2 \succ b_2 \succ c_2 \\
 \vdots \\
 + n \text{ times } x \succ \dots \\
 + \text{lots of fancy votes}
 \end{array}$$

The voting profile is build such that:

- ▶  $score(S_i) = 1, score(x) = n$
- ▶  $score(a_i) = score(b_i) = score(c_i) = n + 1$

To make  $x$  win, add 3 candidates  $w_1, w_2, w_3$  such that:

- ▶ These candidates must appear at most  $n$  times (otherwise they win)
- ▶ They must lower the score of each  $a_i, b_i, c_i$ . e.g.  
 $S_1 \succ a_1 \succ b_1 \succ c_1$  becomes  $S_1 \succ \mathbf{w}_1 \succ \mathbf{w}_2 \succ \mathbf{w}_3$
- ▶ The only way to do this is to *remove from top candidates each  $a_i, b_i, c_i$  exactly once*, which is 3DM.

# COMPUTATIONAL SOCIAL CHOICE

This work belong to the rapidly expanding field of “computational social choice”. Many other topics studied:

- ▶ complexity of computing the result of the election  
P. Faliszewski, E. Hemaspaandra, L. A. Hemaspaandra, J. Rothe. A Richer Understanding of the Complexity of Election Systems. CoRR, 2006.
- ▶ complexity of manipulating the election (by the candidates, by the chair): hardness may actually be a good thing here! But worst-case results are challenged...  
AI’s War on Manipulation: Are We Winning?. P. Faliszewski and A. D. Procaccia. AI Magazine 31(4), 2010.
- ▶ communication complexity of voting rules  
V. Conitzer and T. Sandholm. Communication Complexity of Common Voting Rules. In Proceedings of EC-05
- ▶ and much more...  
Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. A Short Introduction to Computational Social Choice. In Proc. SOFSEM-2007