Towards automatic argumentation about voting rules

Michael Kirsten [1] *Olivier Cailloux* [2]

[1] Dept. of Informatics, Karlsruhe Institute of Technology (KIT)

[2] LAMSADE, Université Paris-Dauphine

3rd July, 2018

https://github.com/oliviercailloux/voting-rule-argumentation-pres

Introduction

Context

- Voting rule: a systematic way of aggregating different opinions and decide
- Multiple reasonable ways of doing this
- Different voting rules have different interesting properties
- None satisfy all desirable properties

Our goal

We want to easily communicate about strengths and weaknesses of voting rules

Outline

Outline

Voting rule

Alternatives $\mathscr{A} = \{a, b, c, d, \dots\}$; $|\mathscr{A}| = m$

Possible voters $\mathscr{N} = \{1, 2, \dots\}$

Voters $\emptyset \subset N \subseteq \mathscr{N}$

Profile Partial function $\mathbf{R}$ from $\mathscr{N}$ to linear orders on $\mathscr{A}$.

Voting rule Function f mapping each $\mathbf{R}$ to winners $\emptyset \subset A \subseteq \mathscr{A}$.

$$\underbrace{\begin{array}{c} \begin{array}{cc} R_1 & R_2 \end{array} \\ \boxed{\begin{array}{cc} a & b \\ b & a \\ c & c \end{array}} \end{array}}_{\mathbf{R}} \xrightarrow{f} A = \{a, b\}$$

Example profile

	nb voters					
	33	16	3	8	18	22
1	a	b	c	c	d	e
2	b	d	d	e	e	c
3	c	c	b	b	c	b
4	d	e	a	d	b	d
5	e	a	e	a	a	a

Who wins?

Example profile

	nb voters					
	33	16	3	8	18	22
1	a	b	c	c	d	e
2	b	d	d	e	e	c
3	c	c	b	b	c	b
4	d	e	a	d	b	d
5	e	a	e	a	a	a

Who wins?

- Most top-1: a
- c is in the top 3 for everybody
- Delete worst first, lowest nb of pref: c, b, e, $a \Rightarrow d$
- Delete worst first, from bottom: a, e, d, $b \Rightarrow c$
- Borda: b

Borda

Given a profile $\mathbf{R}$:

- Score of $a \in \mathscr{A}$: number of alternatives it beats
- The highest scores win

$$\mathbf{R} = \begin{array}{ccccc} a & a & a & b & b \\ b & b & b & c & c \\ c & c & c & a & a \end{array}$$

- Score a is . . . ?

Borda

Given a profile $\mathbf{R}$:

- Score of $a \in \mathscr{A}$: number of alternatives it beats
- The highest scores win

$$\mathbf{R} = \begin{array}{ccccc} a & a & a & b & b \\ b & b & b & c & c \\ c & c & c & a & a \end{array}$$

- Score a is …? $2 + 2 + 2 = 6$
- Score b is $1 + 1 + 1 + 2 + 2 = 7$
- Score c is $1 + 1 = 2$

Winner: b.

Copeland

Given a profile $\mathbf{R}$:

- Score of $a \in \mathscr{A}$: number of alternatives against which it obtains a strict majority ...
- ... minus: number of alternatives that obtains a strict majority against a
- The highest scores win

$$\mathbf{R} = \begin{matrix} a & a & a & b & b \\ b & b & b & c & c \\ c & c & c & a & a \end{matrix}$$

- Score a is ...?

Copeland

Given a profile $\mathbf{R}$:

- Score of $a \in \mathscr{A}$: number of alternatives against which it obtains a strict majority …
- … minus: number of alternatives that obtains a strict majority against a
- The highest scores win

$$\mathbf{R} = \begin{matrix} a & a & a & b & b \\ b & b & b & c & c \\ c & c & c & a & a \end{matrix}$$

- Score a is …? $|\{b, c\}| - |\emptyset| = 2$
- Score b is $|\{c\}| - |\{a\}| = 0$
- Score c is $|\emptyset| - |\{a, b\}| = -2$

Winner: a.

Axiomatic analysis

> *Rather than dream up a multitude of arbitration schemes and determine whether or not each withstands the best of plausibility in a host of special cases, let us invert the procedure. Let us examine our subjective intuition of fairness and formulate this as a set of precise desiderata that any acceptable arbitration scheme must fulfil. Once these desiderata are formalized as axioms, then the problem is reduced to a mathematical investigation of the existence of and characterization of arbitration schemes which satisfy the axioms.*

Luce and Raiffa [1957, p. 121]

What is an axiom?

- An axiom (for us) is a principle
- Expressed formally
- That dictates some behavior of a voting rule
- In some conditions
- Usually seen as something to be satisfied
- Ideally, some combination of axioms defines exactly one rule
- Some axioms can be shown to be incompatible

Unanimity

Unanimity

We may not select as winner someone who has some unanimously preferred alternative.

$$\mathbf{R} = \begin{matrix} a & a & b \\ b & b & c \\ c & c & a \end{matrix}$$

Constraint?

Unanimity

Unanimity

We may not select as winner someone who has some unanimously preferred alternative.

$$\mathbf{R} = \begin{matrix} a & a & b \\ b & b & c \\ c & c & a \end{matrix}$$

Constraint? Do not take c, as b is unanimously preferred to it.

Unanimity

Unanimity

We may not select as winner someone who has some unanimously preferred alternative.

$$\mathbf{R} = \begin{matrix} a & a & b \\ b & b & c \\ c & c & a \end{matrix}$$

Constraint? Do not take c, as b is unanimously preferred to it.

$$\mathbf{R} = \begin{matrix} a & a & b \\ b & c & c \\ c & b & a \end{matrix}$$

Constraint?

Unanimity

Unanimity

We may not select as winner someone who has some unanimously preferred alternative.

$$\mathbf{R} = \begin{array}{ccc} a & a & b \\ b & b & c \\ c & c & a \end{array}$$

Constraint? Do not take c, as b is unanimously preferred to it.

$$\mathbf{R} = \begin{array}{ccc} a & a & b \\ b & c & c \\ c & b & a \end{array}$$

Constraint? No constraint.

Condorcet's principle

Condorcet's principle

We ought to take the Condorcet winner as sole winner if it exists.

- a *beats* b iff more than half the voters prefer a to b.
- a is a *Condorcet winner* iff a beats every other alternative.

$$\mathbf{R} = \begin{array}{ccccc} a & a & a & b & b \\ b & b & b & c & c \\ c & c & c & a & a \end{array}$$

Who wins?

Condorcet's principle

Condorcet's principle

We ought to take the Condorcet winner as sole winner if it exists.

- a *beats* b iff more than half the voters prefer a to b.
- a is a *Condorcet winner* iff a beats every other alternative.

$$\mathbf{R} = \begin{array}{ccccc} a & a & a & b & b \\ b & b & b & c & c \\ c & c & c & a & a \end{array}$$

Who wins? a.

Borda does not satisfy Condorcet

$$\mathbf{R} = \begin{array}{ccccc} a & a & a & b & b \\ b & b & b & c & c \\ c & c & c & a & a \end{array}$$

- Borda winner?

Borda does not satisfy Condorcet

$$\mathbf{R} = \begin{array}{ccccc} a & a & a & b & b \\ b & b & b & c & c \\ c & c & c & a & a \end{array}$$

- Borda winner? b.
- Condorcet winner?

Borda does not satisfy Condorcet

$$\mathbf{R} = \begin{matrix} a & a & a & b & b \\ b & b & b & c & c \\ c & c & c & a & a \end{matrix}$$

- Borda winner? b.
- Condorcet winner? a.

Cancellation

Cancellation

When all pairs of alternatives (a, b) in a profile are such that a is preferred to b as many times as b to a, we ought to select all alternatives as winners.

$$f\begin{pmatrix} a & b & c & c \\ b & a & a & b \\ c & c & b & a \end{pmatrix} = \mathscr{A}$$

Reinforcement

Reinforcement

When joining two sets of voters, exactly those winners that each set accepts should be selected, if possible.

$$\mathbf{R}_1 = \begin{matrix} a & b \\ b & a \\ c & c \end{matrix}, A_1 = \{a, b\}, \mathbf{R}_2 = \begin{matrix} a & b & a \\ b & a & c \\ c & c & b \end{matrix}, A_2 = \{a\},$$

$$\mathbf{R} = \begin{matrix} a & b & a & b & a \\ b & a & b & a & c \\ c & c & c & c & b \end{matrix}. \text{ Winners?}$$

Reinforcement

Reinforcement

When joining two sets of voters, exactly those winners that each set accepts should be selected, if possible.

$$\mathbf{R}_1 = \begin{matrix} a & b \\ b & a \\ c & c \end{matrix}, A_1 = \{a, b\}, \mathbf{R}_2 = \begin{matrix} a & b & a \\ b & a & c \\ c & c & b \end{matrix}, A_2 = \{a\},$$

$$\mathbf{R} = \begin{matrix} a & b & a & b & a \\ b & a & b & a & c \\ c & c & c & c & b \end{matrix}. \text{ Winners? } \{a\}$$

Our objective

Automatically produce “arguments” of the kind: Voting rule f does not satisfy axiom a on profile R.

- To better understand their differences
- To help debate and choose a voting rule
- To empirically investigate attitudes towards given voting rules

Outline

Overview

- Given a voting rule f and an axiom a
- a indicates, given $\mathbf{R}$ and winners W, if $(\mathbf{R}, W)$ fails the axiom

$$\mathbf{R} \xrightarrow{f} W \qquad (\mathbf{R}, W) \xrightarrow{a} \text{pass / fail}$$

Objective

Find $\mathbf{R}$ such that $(\mathbf{R}, f(\mathbf{R}))$ fails a

Example

- $f =$ Borda
- $a =$ Condorcet
- $f(\mathbf{R}) = \{ b \}$ (with $\mathbf{R}$ as used before)
- $a(\mathbf{R}, \{ b \})$ fails

Overview

- Given implementations $\texttt{algo\_}f$ and $\texttt{algo\_}a$

$$\mathbf{R} \xrightarrow{\texttt{algo\_}f} W \qquad (\mathbf{R}, W) \xrightarrow{\texttt{algo\_}a} \text{pass / fail}$$

- We view it as a whole program `algo`
- We use SBMC, a software for checking program properties
- We let SBMC search for an input $\mathbf{R}$ that fails `algo`
- Similar to searching for existence of a bug

Overview

- Given implementations $\texttt{algo\_}f$ and $\texttt{algo\_}a$

$$\mathbf{R} \xrightarrow{\texttt{algo\_}f} W \qquad (\mathbf{R}, W) \xrightarrow{\texttt{algo\_}a} \text{pass / fail}$$

algo

- We view it as a whole program `algo`
- We use SBMC, a software for checking program properties
- We let SBMC search for an input $\mathbf{R}$ that fails `algo`
- Similar to searching for existence of a bug

Checking properties

```
assume (x > 0);
i = 0;
x0 = x;
while (x < y) {
    x += y;
    i += 1;
}
assert (x0 + y*i >= x);
```

- Given an algorithm with parameters (e.g., x, y)
- Check that some property holds
- For all possible parameters
- . . . that satisfy given assumptions

$\Rightarrow$ Search for (x, y) that satisfy assumptions and fail assertion

Software Bounded Model Checking (SBMC)

Specification

- Properties specified using `assume` and `assert` statements
- A program `Prog` is **correct** iff:

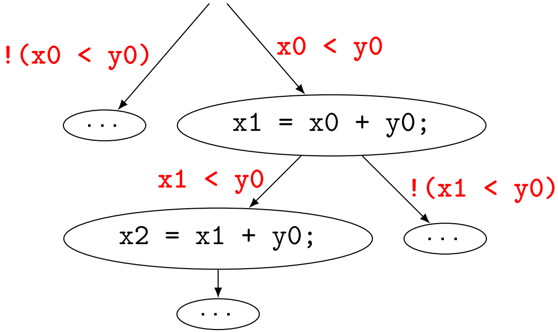
$$\texttt{Prog} \wedge \bigwedge \texttt{assume} \Rightarrow \bigwedge \texttt{assert}$$

- `Prog` is automatically generated logical encoding of the program

- SBMC tool converts program into SAT
- Exhaustive check by unwinding the control flow graph
- Bounded in number of loop unwindings and recursions
- Special “unwinding assertion” claims added to check whether longer program paths may be possible

Taking care of loops in SBMC

```
while(x < y) x = x + y;
```



Specifying and Verifying Properties in SBMC

Verification

- Checking properties for programs generally undecidable
- SBMC analyses only program runs up to **bounded** length
- Property checking becomes decidable by logical encoding
- Can be decided using SAT- or SMT-solver

Outline

Borda fails Condorcet

A minimal counter-example (found in less than one second):

$$\mathbf{R} = \begin{array}{ccc} c & c & b \\ b & b & a \\ a & a & c \end{array}$$

Borda rule elects $\{a, c\}$ instead of the Condorcet winner c.
The example can be easily inspected manually.

Borda fails Weak Majority

A minimal counter-example in nb alternatives (<1 sec):

$$\mathbf{R} = \begin{matrix} a & a & a & b & b \\ b & b & b & c & c \\ c & c & c & a & a \end{matrix}$$

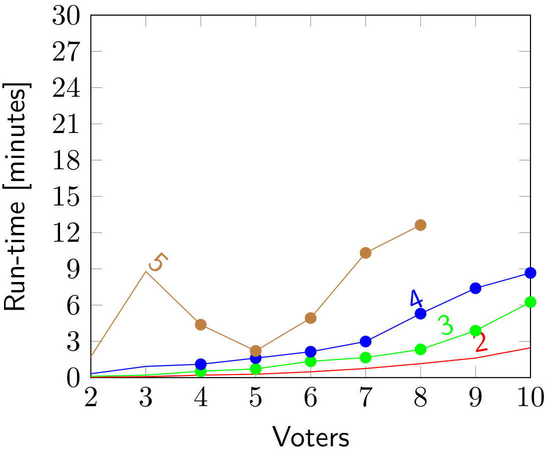
Borda elects b instead of the majority winner a.
A minimal counter-example in nb voters (<1 sec):

$$\mathbf{R} = \begin{matrix} d & d & c \\ c & c & a \\ a & b & b \\ b & a & d \end{matrix}$$

Borda elects c instead of the majority winner d.

Copeland fails Reinforcement

Run-times for 2, 3, 4 and 5 alternatives



Copeland fails Reinforcement

A minimal counter-example (found in 32 seconds):

$$\mathbf{R}_1 = \begin{matrix} b & a \\ a & c \\ c & b \end{matrix}, \quad \mathbf{R}_2 = \begin{matrix} a & b \\ b & a \\ c & c \end{matrix}$$

- Elected for $\mathbf{R}_1$ and $\mathbf{R}_2$: a and $\{a, b\}$ respectively.
- For the joined profile $\mathbf{R}_1 \cup \mathbf{R}_2$, Copeland elects $\{a, b\}$ instead of a.

Thank you for your attention!

References

R. Luce and H. Raiffa. *Games and Decisions*. J. Wiley, New York, 1957.

License