Computational Social Choice: Lecture 1

Ulle Endriss
Institute for Logic, Language and Computation
University of Amsterdam

Outline

Social choice theory is the study of collective decision making (as in voting, fair division, matching, ...). Over the past decade or so, this has become a major topic also in *computer science*. Important:

$$\textit{computational} \neq \textit{using a computer}$$

We'll focus on some of the *methodologies* used in COMSOC research:

- *Computational complexity* (specifically of strategic behaviour)
- Analysis of *communication and information* requirements
- Working with alternatives with a *combinatorial structure* [Friday]

But first, some examples for social choice problems ...

F. Brandt, V. Conitzer, and U. Endriss. Computational Social Choice. In G. Weiss (ed.), *Multiagent Systems*, MIT Press, 2013.

Three Voting Rules

How should n *voters* choose from a set of m *alternatives*?

Here are three *voting rules* (there are many more):

- *Plurality:* elect the alternative ranked first most often (i.e., each voter assigns 1 point to an alternative of her choice, and the alternative receiving the most points wins)
- *Plurality with runoff:* run a plurality election and retain the two front-runners; then run a majority contest between them
- *Borda:* each voter gives $m-1$ points to the alternative she ranks first, $m-2$ to the alternative she ranks second, etc.; and the alternative with the most points wins

Example: Choosing a Beverage for Lunch

Consider this election with nine *voters* having to choose from three *alternatives* (namely what beverage to order for a common lunch):

4 Dutchmen:	Milk	$\succ$	Beer	$\succ$ Wine
3 Frenchmen:	Wine	$\succ$	Beer	$\succ$ Milk
2 Germans:	Beer	$\succ$	Wine	$\succ$ Milk

Which beverage *wins* the election for

- the plurality rule?
- plurality with runoff?
- the Borda rule?

Example: Voting in Multi-issue Elections

Suppose 13 voters are asked to each vote *yes* or *no* on three issues; and we use the plurality rule for each issue independently:

- 3 voters each vote for YNN, NYN, NNY.
- 1 voter each votes for YYY, YYN, YNY, NYY.
- No voter votes for NNN.

But then NNN wins: 7 out of 13 vote *no* on each issue (*paradox!*).

What to do instead? The number of (combinatorial) alternatives is *exponential* in the number of issues (e.g., $2^3 = 8$), so even just representing the voters' preferences is a challenge ...

S.J. Brams, D.M. Kilgour, and W.S. Zwicker. The Paradox of Multiple Elections. *Social Choice and Welfare*, 15(2):211–236, 1998.

Judgment Aggregation

Preferences are not the only structures we may wish to aggregate.

In JA we aggregate people's judgments regarding complex propositions.

	p	$p \rightarrow q$	q
Judge 1:	Yes	Yes	Yes
Judge 2:	Yes	No	No
Judge 3:	No	Yes	No
	?		

Fair Division

Fair division is the problem of dividing one or several goods amongst two or more agents in a way that satisfies a suitable fairness criterion. One instance of this problem is *cake cutting*.

For *two agents*, we can use the *cut-and-choose* procedure:

- *One agent cuts the cake in two pieces (she considers to be of equal value), and the other chooses one of them (the piece she prefers).*

The cut-and-choose procedure is *proportional:*

- Each agent is guaranteed at least one half (general: $1/n$) according to her own valuation.

What if there are more than two agents? Is proportionality the best way of measuring fairness? What about other types of goods?

Computational Social Choice

Research can be broadly classified along two dimensions —

The kind of *social choice problem* studied, e.g.:

- electing a winner given individual preferences over candidates
- aggregating individual judgements into a collective verdict
- fairly dividing a cake given individual tastes
- finding a stable matching of students to schools

The kind of *computational technique* employed, e.g.:

- algorithm design to implement complex mechanisms
- complexity theory to understand limitations
- logical modelling to fully formalise intuitions
- knowledge representation techniques to compactly model problems
- deployment in a multiagent system

Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. A Short Introduction to Computational Social Choice. Proc. SOFSEM-2007.

Voting Theory

In these lectures we will focus on the problem of voting to exemplify some of the ideas in COMSOC, but similar questions may also be asked for other frameworks of collective decision making ...

Formal Framework

Finite set of *voters* $\mathcal{N} = \{1, \ldots, n\}$ and finite set of *alternatives* $\mathcal{X}$.

Each voter expresses a *preference* over the alternatives by providing a linear order on $\mathcal{X}$ (her *ballot*). $\mathcal{L}(\mathcal{X})$ is the set of all such linear orders. A *profile* $\boldsymbol{R} = (R_1, \ldots, R_n)$ fixes one preference/ballot for each voter.

A *voting rule* F is a function mapping every possible profile to a (nonempty) set of winning alternatives:

$$F : \mathcal{L}(\mathcal{X})^n \to 2^{\mathcal{X}} \setminus \{\emptyset\}$$

If $|F(\boldsymbol{R})| = 1$ for every profile $\boldsymbol{R}$, then F is called *resolute*.

Example: Electing a President

Remember Florida 2000 (simplified):

49%:	Bush $\succ$ Gore $\succ$ Nader
20%:	Gore $\succ$ Nader $\succ$ Bush
20%:	Gore $\succ$ Bush $\succ$ Nader
11%:	Nader $\succ$ Gore $\succ$ Bush

Questions:

- Who wins?
- Is that a fair outcome?
- What would your advice to the Nader-supporters have been?

Strategic Manipulation

Our example demonstrates that the plurality rule is not *strategy-proof*: sometimes voters have an incentive to report false preferences. Worse:

Theorem 1 (Gibbard-Satterthwaite) *Any resolute voting rule for $\geqslant 3$ alternatives that is surjective and strategy-proof is a dictatorship.*

What to do?

- domain restrictions (e.g., single-peakedness)
- change of model (e.g., irresoluteness + certain strong assumptions)
- look for rules with low frequency of manipulability
- look for rules that are *"difficult" to manipulate*

A. Gibbard. Manipulation of Voting Schemes: A General Result. *Econometrica*, 41(4):587–601, 1973.

M.A. Satterthwaite. Strategy-proofness and Arrow's Conditions. *Journal of Economic Theory*, 10:187–217, 1975.

Complexity as a Barrier against Manipulation

The Gibbard-Satterthwaite Theorem shows that strategic manipulation cannot be ruled out.

Idea: So it's always *possible* to manipulate; but maybe it's also *difficult*? Tools from *complexity theory* can make this idea precise.

- If manipulation is computationally intractable for F, then F might be considered *resistant* (albeit still not *immune*) to manipulation.
- Even if standard voting rules turn out to be easy to manipulate, it might still be possible to *design new ones* that are resistant.

Computational Complexity Theory

Computational complexity theory studies the hardness of classes of (decision) problems in terms of the time (and other resources) they require to be solved, relative to their size. Examples:

- for a given *graph* with n nodes, deciding whether node A is *reachable* from node B is possible in $O(n^2)$ steps
- for a given propositional *formula* of length n, deciding whether it is *satisfiable* is possible in $O(2^n)$ steps

Two important complexity classes:

- *P:* the class of problems that can be solved in polynomial time
- *NP*: the class of problems for which a supposed solution can be verified in polynomial time (unknown whether P $\neq$ NP)

NP-complete means: amongst the hardest problems within NP.

C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

Classical Results

The seminal paper by Bartholdi, Tovey and Trick (1989) starts by showing that manipulation is in fact *easy* for a range of commonly used voting rules, and then presents one system (a variant of the Copeland rule) for which manipulation is NP-complete. Next:

- We first present a couple of these easiness results, namely for *plurality* and for the *Borda rule*.
- We then mention a result from a follow-up paper by Bartholdi and Orlin (1991): the manipulation of *STV* (Hare) is *NP-complete*.

J.J. Bartholdi III, C.A. Tovey, and M.A. Trick. The Computational Difficulty of Manipulating an Election. *Soc. Choice and Welfare*, 6(3):227–241, 1989.

J.J. Bartholdi III and J.B. Orlin. Single Transferable Vote Resists Strategic Voting. *Social Choice and Welfare*, 8(4):341–354, 1991.

Manipulability as a Decision Problem

We can cast the problem of manipulability, for a particular voting rule F, as a decision problem:

$\textsc{Manipulability}(F)$
Instance: Set of ballots for all but one voter; alternative x.
Question: Is there a ballot for the final voter such that x wins?

A manipulator has to solve $\textsc{Manipulability}(F)$ for all alternatives, in order of her preference. (Note that in practice the manipulator does not just want a yes/no answer, but the manipulating ballot.)

If $\textsc{Manipulability}(F)$ is computationally intractable, then manipulability may be considered less of a worry for F.

Remark: We assume that the manipulator knows all the other ballots. This unrealistic assumption is intentional: if manipulation is intractable even under such favourable conditions, then all the better.

Manipulating the Plurality Rule

Recall plurality: the alternative(s) ranked first most often win(s)

The plurality rule is easy to manipulate (trivial):

- Simply vote for x, the alternative to be made winner by means of manipulation. If manipulation is possible at all, this will work. Otherwise manipulation is not possible.

That is, we have $\textsc{Manipulability}(\mathit{plurality}) \in \mathrm{P}$.

General: $\textsc{Manipulability}(F) \in \mathrm{P}$ for any rule F with polynomial winner determination problem and polynomial number of ballots.

Manipulating the Borda Rule

Recall Borda: submit a ranking (super-polynomially many choices!) and give $m-1$ points to 1st ranked, $m-2$ points to 2nd ranked, etc.

The Borda rule is also easy to manipulate. Use a *greedy algorithm:*

- Place x (the alternative to be made winner through manipulation) at the top of your ballot.
- 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.)

After convincing ourselves that this algorithm is indeed correct, we also get $\textsc{Manipulability}(Borda) \in \mathrm{P}$.

J.J. Bartholdi III, C.A. Tovey, and M.A. Trick. The Computational Difficulty of Manipulating an Election. *Soc. Choice and Welfare*, 6(3):227–241, 1989.

Intractability of Manipulating STV

Single Transferable Vote (STV): eliminate plurality losers until an alternative is ranked first by $> 50\%$ of the voters

Theorem 2 (Bartholdi and Orlin, 1991) $\textsc{Manipulability}(STV)$ *is NP-complete.*

Proof: Omitted.

J.J. Bartholdi III and J.B. Orlin. Single Transferable Vote Resists Strategic Voting. *Social Choice and Welfare*, 8(4):341–354, 1991.

Coalitional Manipulation

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.

Variants of the problem:

- Ballots may be *weighted* or *unweighted*.

  Examples: countries in the EU; shareholders of a company

- Manipulation may be *constructive* (making alternative x a *unique* or *tied* winner) or *destructive* (ensuring x does not win).

Decision Problems

On the following slides, we will consider two decision problems, for a given voting rule F:

ConstructiveManipulability(F)
Instance: Set of weighted ballots; set of weighted manipulators; $x \in \mathcal{X}$.
Question: Are there ballots for the manipulators such that x wins?

DestructiveManipulability(F)
Instance: Set of weighted ballots; set of weighted manipulators; $x \in \mathcal{X}$.
Question: Are there ballots for the manipulators such that x loses?

Constructive Manipulation under Borda

In the context of coalitional manipulation with weighted voters, we can get hardness results for elections with small numbers of alternatives:

Theorem 3 (Conitzer et al., 2007) *Under the Borda rule, the constructive coalitional manipulation problem with weighted voters is NP-complete for $\geqslant 3$ alternatives.*

Proof: We have to prove NP-membership and NP-hardness:

- NP-membership: easy (if you guess ballots for the manipulators, we can check that it works in polynomial time)
- NP-hardness: for three alternatives by reduction from PARTITION (next slide); hardness for more alternatives follows

V. Conitzer, T. Sandholm, and J. Lang. When are Elections with Few Candidates Hard to Manipulate? *Journal of the ACM*, 54(3), Article 14, 2007.

Proof of NP-hardness

We will use a reduction from the NP-complete PARTITION problem:

PARTITION
Instance: $(w_1, \ldots, w_n) \in \mathbb{N}^n$
Question: Is there a set $I \subseteq \{1, \ldots, n\}$ s.t. $\sum_{i \in I} w_i = \frac{1}{2} \sum_{i=1}^{n} w_i$?

Let $K := \sum_{i=1}^{n} w_i$. Given an instance of PARTITION, we construct an election with $n + 2$ weighted voters and three alternatives:

- two voters with weight $\frac{1}{2}K - \frac{1}{4}$, voting $(x \succ y \succ z)$ and $(y \succ x \succ z)$
- a coalition of n voters with weights $w_1, \ldots, w_n$ who want z to win

Clearly, each manipulator should vote either $(z \succ x \succ y)$ or $(z \succ y \succ x)$.
Suppose there does exist a partition. Then they can vote like this:

- manipulators corresponding to elements in I vote $(z \succ x \succ y)$
- manipulators corresponding to elements outside I vote $(z \succ y \succ x)$

Scores: $2K$ for z; $\frac{1}{2}K + (\frac{1}{2}K - \frac{1}{4}) \cdot (2 + 1) = 2K - \frac{3}{4}$ for both x and y
If there is no partition, then either x or y will get at least 1 point more.
Hence, manipulation is feasible *iff* there exists a partition. ✓

Destructive Manipulation under Borda

Theorem 4 (Conitzer et al., 2007) *Under the Borda rule, the destructive coalitional manip. problem with weighted voters is in P.*

Proof: Let x be the alternative the manipulators want to lose. The following algorithm will find a manipulation, if one exists:

> For each alternative $y \neq x$, try letting all manipulators rank y first, x last, and the other alternatives in any fixed order.
>
> If x loses in one of these $m-1$ elections, then manipulation is possible; otherwise it is not.

Correctness of the algorithm follows from the fact that (a) the best we can do about x is not to give x any points and, (b) if any other alternative y has a chance of beating x, she will do so if we give y a maximal number of points. ✓

V. Conitzer, T. Sandholm, and J. Lang. When are Elections with Few Candidates Hard to Manipulate? *Journal of the ACM*, 54(3), Article 14, 2007.

Worst-Case vs. Average-Case Complexity

NP-hardness is only a *worst-case* notion. Do NP-hardness barriers provide sufficient protection against manipulation?

What about the *average complexity* of strategic manipulation?

Some recent work suggests that it might be impossible to find a voting rule that is *usually* hard to manipulation—for a suitable definition of "usual". See Faliszewski and Procaccia (2010) for a discussion.

P. Faliszewski and A.D. Procaccia. AI's War on Manipulation: Are We Winning? *AI Magazine*, 31(4):53–64, 2010.

Controlling Elections

Strategic manipulation is not the only undesirable form of behaviour in voting we may want to contain by means of complexity barriers ...

People have studied the computational complexity of a range of different types of *control* in elections:

- Adding or removing *candidates*.
- Adding or removing *voters*.
- Redefining *districts* (if your party is likely to win district A with an 80% majority and lose district B by a small margin, you might win both districts if you carefully redraw the district borders ...).

See Faliszewski et al. (2009) for an introduction to this area.

P. Faliszewski, E. Hemaspaandra, L.A. Hemaspaandra, and J. Rothe. *A Richer Understanding of the Complexity of Election Systems*. In *Fundamental Problems in Computing*, Springer-Verlag, 2009.

Bribery in Elections

Bribery is the problem of finding $\leqslant K$ voters such that a suitable change of their ballots will make a given candidate x win.

- Connection to *manipulation:* in the (coalitional) manipulation problem the names of the voters changing ballot are part of the input, while for the bribery problem we need to choose them.
- Several *variants* of the bribery problem have been studied: when each voter has a possibly different "price"; when bribes depend on the extent of the change in the bribed voter's ballot; etc.

People have studied the complexity of several variants of the bribery problem for various voting rules (e.g., Faliszewski et al., 2009).

P. Faliszewski, E. Hemaspaandra, and L.A. Hemaspaandra. How Hard is Bribery in Elections? *Journal of Artificial Intelligence Research*, 35:485–532, 2009.

Information and Communication

Next we will discuss a range of questions concerning the role of *information* and *communication* in voting:

- The Possible Winner Problem
  - its many interpretations and applications
  - its complexity, for various settings and voting rules
- Compilation of Intermediate Election Results

Note: We will mostly concentrate on positional scoring rules, particularly Borda and plurality, to exemplify the general ideas, but several other voting rules have been analysed as well.

Possible Winners

Idea: If we only have partial information about the ballots, we may ask which alternatives are *possible winners* (for a given voting rule).

Let $\mathcal{P}(\mathcal{X})$ be the class of *partial orders* on the set of alternatives $\mathcal{X}$.

Terminology: The linear order $(\succ_\ell) \in \mathcal{L}(\mathcal{X})$ *refines* the partial order $(\succ_p) \in \mathcal{P}(\mathcal{X})$ if $(\succ_\ell) \supseteq (\succ_p)$, i.e., if $x \succ_\ell y$ whenever $x \succ_p y$.
Similarly, a profile of linear ballots $\boldsymbol{R}^\ell \in \mathcal{L}(\mathcal{X})^n$ refines a profile of partial ballots $\boldsymbol{R}^p \in \mathcal{P}(\mathcal{X})^n$ if R_i^ℓ refines R_i^p for each voter $i \in \mathcal{N}$.

Definition: Given a partial profile $\boldsymbol{R} \in \mathcal{P}(\mathcal{X})^n$, an alternative $x^\star \in \mathcal{X}$ is called a *possible winner* under voting rule F if $x^\star \in F(\boldsymbol{R}^\star)$ for *some* profile of linear ballots $\boldsymbol{R}^\star \in \mathcal{L}(\mathcal{X})^n$ that refines $\boldsymbol{R}$.

The concept was originally introduced by Konczak and Lang (2005).

K. Konczak and J. Lang. *Voting Procedures with Incomplete Preferences*. Proc. Multidisciplinary Workshop on Advances in Preference Handling 2005.

Necessary Winners

Analogously, we can also define the set of necessary winners of an election with partial information about ballots:

Given a profile of partial ballots $\boldsymbol{R} \in \mathcal{P}(\mathcal{X})^n$, an alternative $x^\star \in \mathcal{X}$ is called a *necessary winner* under voting rule F if $x^\star \in F(\boldsymbol{R}^\star)$ for *all* profiles of linear ballots $\boldsymbol{R}^\star \in \mathcal{L}(\mathcal{X})^n$ that refine $\boldsymbol{R}$.

Remark: The set of necessary winners is a (not necessarily proper) *subset* of the set of possible winners.

Connection with Preference Elicitation

Eliciting preference/ballot information from voters is costly, so it is interesting to develop protocols for goal-directed elicitation:

- *Coarse elicitation:* Ask each voter for her ballot in turn.
- *Fine elicitation:* Ask voters to rank pairs of alternatives one-by-one, in an appropriate order.

Observe the following connection:

- We can *stop eliciting* ballot information as soon as the sets of *possible and necessary winners coincide*.

More on elicitation: Conitzer and Sandholm (2002), Walsh (2008)

V. Conitzer and T. Sandholm. Vote Elicitation: Complexity and Strategy-Proofness. Proc. AAAI-2002.

T. Walsh. Complexity of Terminating Preference Elicitation. Proc. AAMAS-2008.

Special Case: Missing Voters

A first natural special case of having only partial ballot information is when some *voters are missing:*

- Some voters have ballots that are *linear orders* (note: a linear order is a special case of a partial order).
- All other voters have *empty relations* as ballots: $x \not\succ y$ for all $x, y \in \mathcal{X}$ (also a special case of a partial order)

Possible scenarios:

- We might be in the midst of a coarse elicitation procedure.
- Postal ballots may only arrive a few days after election day.

Missing Voters and Coalitional Manipulation

There are close links to the coalitional manipulation problem:

- The *possible winner* problem with missing voters is equivalent to the *constructive coalitional manipulation* problem:

  A coalition of voters can collude to make $x^\star$ a (joint) winner *iff* $x^\star$ is a possible winner when only those voters are missing.

- The *necessary winner* problem is the complement of the *destructive coalitional manipulation* problem:

  A coalition of of voters can collude to bar $x^\star$ from winning *iff* $x^\star$ is *not* a necessary winner when only those voters are missing.

Special Case: Missing Alternatives

A second natural special case of having only partial ballot information is when some *alternatives are missing:*

- Distinguish "old" alternatives $\mathcal{X}_1$ and "new" alternatives $\mathcal{X}_2$.
- All ballots are complete on $\mathcal{X}_1$: $x \succ y$ or $y \succ x$ for all $x, y \in \mathcal{X}_1$.
- All ballots are empty on pairs involving at least one new alternative: $x \not\succ y$ if $x \in \mathcal{X}_2$ or $y \in \mathcal{X}_2$

Possible scenario:

- Some alternatives (e.g., a new plan) become available only after voting has started.

Remark: We had briefly discussed control by adding alternatives and bribery before. This is related, but different (now we don't know or control how the new alternatives will be ranked by the voters).

Computational Complexity

There are a large number of complexity results for the possible and necessary winner problems in the literature:

- for a range of *voting rules* (for which the standard winner determination problem is tractable, otherwise it's hopeless)
- for *weighted* and *unweighted* voters
- for *bounded* and *unbounded* numbers of alternatives
- for the *general* problem and for *special* cases

Next, we will see some of these results, for positional scoring rules with unweighted voters and unbounded numbers of alternatives.

The Possible Winner Problem

This is the variant of the problem we will consider:

$\textsc{PossibleWinner}(F)$
Instance: profile of partial ballots $\boldsymbol{R} \in \mathcal{P}(\mathcal{X})^n$; alternative $x^\star \in \mathcal{X}$
Question: Is $x^\star$ a possible winner under voting rule F?

Note that ballots are *unweighted* and that the number of alternatives is *unbounded* (the crucial parameter for the complexity will be the number of alternatives, not the number of voters).

Possible Winners under Plurality

Even for the very simplest of voting rules, computing possible winners is not trivial. But for plurality it is at least polynomial:

Theorem 5 (Betzler and Dorn, 2010) *Under the plurality rule, the possible winner problem can be decided in polynomial time.*

The original proof of Betzler and Dorn is based on flow networks. Here we will instead use a reduction to bipartite matching.

Remark: Computing possible winners under the *antiplurality rule* is also polynomial, and the proof is very similar.

N. Betzler and B. Dorn. Towards a Dichotomy for the Possible Winner Problem in Elections Based on Scoring Rules. *Journal of Computer and System Sciences*, 76(8):812–836, 2010.

Bipartite Matching

A *perfect matching* of a graph $G = (V, E)$ is a set of edges $E' \subseteq E$ such that each vertex in V is adjacent to *exactly* one edge in E'.

A graph $G = (V, E)$ is called *bipartite* if the set of vertices V can be partitioned into sets V_1 and V_2 such that each edge in E is adjacent to both a vertex in V_1 and a vertex in V_2.

BIPARTITE MATCHING
Instance: Bipartite graph G.
Question: Does G have a perfect matching?

BIPARTITE MATCHING can be decided in polynomial time, using for instance the Hopcroft-Karp Algorithm (Hopcroft and Karp, 1973).

J.E. Hopcroft and R.M. Karp. An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

Proof of Theorem 5

Recall: Given a profile of partial ballots, we want to show that we can decide in polynomial time whether $x^\star$ is a possible plurality winner.

An alternative can get a point from a partial ballot *iff* it is undominated.

Suppose $x^\star$ is undominated in $K \leqslant n$ ballots. So $x^\star$ can get K points.

Can we choose one alternative from each of the remaining sets of undominated alternatives without one of them getting more than K points?

If $K \cdot (m-1) < n-K$, then one rival must get more points than $x^\star$. ✓

Otherwise, construct a bipartite graph G:

- Lefthand vertices: one for each remaining set of undominated alternatives; and $K \cdot (m-1) - (n-K)$ "dummy" vertices
- Righthand vertices: K copies of each alternative other than $x^\star$
- Edges: link each set-vertex to each copy of each of its members; and link each dummy vertex to all vertices on the right

Now we have a one-to-one correspondence between perfect matchings of G and ballot refinements that give each rival of $x^\star$ at most K points. ✓

Possible Winners under Borda

Theorem 6 (Xia and Conitzer, 2008) *Under the Borda rule, deciding whether an alternative is a possible winner is NP-complete.*

Proof: Omitted.

In fact, the result of Xia and Conitzer covers many positional scoring rules. Betzler and Dorn (2010) showed that PossibleWinner is NP-complete for almost all PSR's (except for plurality and veto).

L. Xia and V. Conitzer. Determining Possible and Necessary Winners under Common Voting Rules Given Partial Orders. Proc. AAAI-2008.

N. Betzler and B. Dorn. Towards a Dichotomy for the Possible Winner Problem in Elections Based on Scoring Rules. *Journal of Computer and System Sciences*, 76(8):812–836, 2010.

Possible Winners for Missing Alternatives

Recall that for the possible winner problem with missing alternatives only, we are given a complete ballot profile for the "old" alternatives and have to decide whether the "new" alternatives can be inserted so that $x^\star$ wins.

Theorem 7 (Chevaleyre et al., 2010) *Under the Borda rule, the possible winner problem restricted to the case of missing alternatives can be decided in polynomial time.*

Proof: Trivial if $x^\star$ is new. Else: The best we can do for $x^\star$ is to insert the new alternatives (in any order) just below $x^\star$ in each ballot. This maximises the point difference between $x^\star$ and its old rivals (and $x^\star$ beats all new rivals). So we only need to check whether $x^\star$ wins for this one profile. ✓

Y. Chevaleyre, J. Lang, N. Maudet, and J. Monnot. Possible Winners when New Candidates are Added: The Case of Scoring Rules. Proc. AAAI-2010.

Overview: Possible Winners under PSR's

We have seen several results regarding the computational complexity of the PossibleWinner problem for *positional scoring rules:*

- General case: polynomial for plurality and antiplurality
- General case: NP-complete for Borda (and most PSR's)
- Missing alternatives only: polynomial for Borda
- Missing voters only: NP-complete for for Borda (not discussed)

  This is equivalent to the constructive coalitional manipulation problem. For *weighted* voters, we have seen an NP-hardness proof before (there is a similar result for unweighted voters).

Compilation of Intermediate Election Results

Given a partial ballot profile and a voting rule, how much information do we need to store to be to be able to compute the election winners once the remaining ballot information comes in?

An instance of this general question:

- If some voters have voted already (using complete linear ballots), what is the most compact way of *compiling* the intermediate election result so as to be able to compute the winner(s) once the remaining voters have voted?

The number of bits required to store the relevant information for a given voting rule F is called the *compilation complexity* of F.

Remark: The *number of additional voters* may be known or unknown. We shall discuss the case where it is *unknown*.

Compilation Complexity

Let n be the number of (old) voters and m the number of alternatives.
We need $\lceil \log m \rceil$ bits to represent the name of one alternative.

Some basic observations:

- The compilation complexity of *any voting rule* is at most $n\lceil \log(m!) \rceil$ (just store all ballots)
- The compilation complexity of any *anonymous* voting rule is at most $\min\{n\lceil \log(m!) \rceil, m!\lceil \log(n+1) \rceil\}$ (as we can also store for each ballot the number of voters choosing it, if that's shorter)
- The compilation complexity of any *dictatorial* voting rule is $\lceil \log m \rceil$ (just store the choice of the dictator)
- The compilation complexity of any *constant* voting rule (always electing the same winner) is 0.

Y. Chevaleyre, J. Lang, N. Maudet, and G. Ravailly-Abadie. Compiling the Votes of a Subelectorate. Proc. IJCAI-2009.

Equivalence Classes

Let F be a voting rule.

Call two ballot profiles $\boldsymbol{R}$ and $\boldsymbol{R'}$ *F-equivalent* if for any additional ballot profile $\boldsymbol{R^\star}$, we have $F(\boldsymbol{R} \oplus \boldsymbol{R^\star}) = F(\boldsymbol{R'} \oplus \boldsymbol{R^\star})$.

The most space-efficient (but not time-efficient!, which is ok) way to compile a ballot profile is to store the index of its equivalence class.

Hence, if $g(F, n, m)$ is the number of equivalence classes for voting rule F, n (old) voters, and m alternatives, then the compilation complexity of F is (exactly) $\lceil \log g(F, n, m) \rceil$.

This suggests a proof technique for proving compilation complexity results: count the equivalence classes!

Compilation Complexity of Borda

Two ballots are *Borda-equivalent* if they assign the same Borda scores. But this doesn't tell us how many equivalence classes there are (some combinations of Borda scores will be impossible). So we cannot use this to get the *exact* compilation complexity of Borda.

Theorem 8 (Chevaleyre et al., 2009) *The compilation complexity of the Borda rule is in* $O(m \log(nm))$.

Proof: Next slide.

Remark: Chevaleyre et al. also prove a matching lower bound.

Y. Chevaleyre, J. Lang, N. Maudet, and G. Ravailly-Abadie. Compiling the Votes of a Subelectorate. Proc. IJCAI-2009.

Proof of the Upper Bound

We want to establish an upper bound on the compilation complexity of the *Borda* rule for n voters and m alternatives:

- The maximal Borda score of each alternative is $n(m-1)$, because at best they all rank that alternative first.
- So we can store the scores of the first $m-1$ alternatives using $\lceil (m-1)\log(n(m-1))\rceil$ bits; the mth score can be inferred.
- Hence, the compilation complexity of Borda is at most $\lceil (m-1)\log(n(m-1))\rceil$, and thus in $O(m\log(nm))$. ✓

Summary

We have reviewed several examples for typical research questions investigated in computational social choice.

- *Strategic manipulation:* can high complexity protect elections?
- *Possible winners:* which alternatives do have a chance of winning?
- *Compilation:* how can we efficiently represent the relevant information elicited for the voters so far?

In the second lecture, we will look into the problem of dealing with large numbers of alternatives for *voting in combinatorial domains*.

Meanwhile, more information is available in the book chapter cited below and on the website of my Amsterdam course:

`http://www.illc.uva.nl/~ulle/teaching/comsoc/`

F. Brandt, V. Conitzer, and U. Endriss. Computational Social Choice. In G. Weiss (ed.), *Multiagent Systems*, MIT Press, 2013.