Computational Social Choice: Autumn 2010

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

Plan for Today

Today 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 procedures
- Compilation of Intermediate Election Results
- Communication Complexity of Voting Procedures

Note: We will mostly concentrate on positional scoring rules, particularly Borda and plurality, to exemplify the general ideas, but several other voting procedures 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 procedure).

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 $\underline{b^\ell} \in \mathcal{L}(\mathcal{X})^\mathcal{N}$ refines a profile of partial ballots $\underline{b^p} \in \mathcal{P}(\mathcal{X})^\mathcal{N}$ if b_i^ℓ refines b_i^p for each voter $i \in \mathcal{N}$.

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

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 $\underline{b} \in \mathcal{P}(\mathcal{X})^{\mathcal{N}}$, an alternative $x^\star \in \mathcal{X}$ is called a *necessary winner* under voting procedure F if $x^\star \in F(\underline{b^\star})$ for *all* profiles of linear ballots $\underline{b^\star} \in \mathcal{L}(\mathcal{X})^{\mathcal{N}}$ that refine $\underline{b}$.

Remark: The set of necessary winners is always 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.

Connection with General Ballot Languages

In the setting we explore today,

- we are given a profile of partial ballots and
- we wonder who can/will win the election if we refine this to a profile of standard linear ballots.

That is, we are working within standard voting theory, where ballots are linear orders (albeit asking a nonstandard question).

Contrast this with some other settings we have explored previously:

- We may want to develop a voting theory for non-linear ballot languages (cf. lecture on circumventing manipulation).
- Compact representation languages may give rise to non-linear ballots (cf. lecture on combinatorial domains).

U. Endriss, M.S. Pini, F. Rossi, and K.B. Venable. Preference Aggregation over Restricted Ballot Languages: Sincerity and Strategy-Proofness. Proc. IJCAI-2009.

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 procedures* (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

Remark: For the combination of unweighted voters and a bounded number of alternatives everything is polynomial.

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 $\underline{b} \in \mathcal{P}(\mathcal{X})^{\mathcal{N}}$; alternative $x^\star \in \mathcal{X}$
Question: Is $x^\star$ a possible winner under voting procedure 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 procedures, computing possible winners is not trivial. But for plurality it is at least polynomial:

Theorem 1 (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 *veto 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 1

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

Proof: By reduction from EXACT3COVER (details omitted).

In fact, the result of Xia and Conitzer covers a large class of positional scoring rules. Betzler and Dorn (2010) showed that POSSIBLEWINNER is NP-complete for almost all PSRs (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 3 (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. ✓

Remark: Chevaleyre et al. state their theorem for a bounded number k of new alternatives, but it holds for both the bounded and the unbounded case.

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 PSRs

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

- General case: polynomial for plurality and veto
- General case: NP-complete for Borda (and most PSRs)
- Missing alternatives only: polynomial for Borda

  There are both tractability and intractability results for other PSRs. Most (all?) known intractability results apply even when the number of new alternatives is bounded.
- Missing voters only: complexity *unknown* for Borda

  This is equivalent to constructive coalitional manipulation. Recall that we have seen that for *weighted* voters this is NP-complete, even for 3 alternatives (cf. lecture on circumventing manipulation).

Compilation of Intermediate Election Results

Given a partial ballot profile and a voting procedure, 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 additional voters* may be known or unknown. We shall discuss the case where it is *unknown*.

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

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 procedure* is at most $n\lceil \log(m!) \rceil$ (just store all ballots)
- The compilation complexity of any *anonymous* voting procedure 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 procedure is $\lceil \log m \rceil$ (just store the choice of the dictator)
- The compilation complexity of any *constant* voting procedure (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 procedure.

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

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 procedure 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 Plurality

The compilation complexity of *plurality* will be at most $m\lceil\log(n+1)\rceil$ (store the plurality score $\in \{0,\ldots,n\}$ for each alternative).

But we can do better:

Theorem 4 (Chevaleyre et al., 2010) *The compilation complexity of the plurality rule (for an unknown number of additional voters) is exactly* $\lceil\log\binom{m+n-1}{n}\rceil$, *which is in* $\Theta(m\log(1+\frac{n}{m})+n\log(1+\frac{m}{n}))$.

Recall that $f(n)\in\Theta(g(n))$ means that f is asymptotically bounded both from above and from below by g.

The proof for the exact bound is on the next slide. The rest follows from *Stirling's approximation* ($\log(n!)$ is in $\Theta(n\log n)$).

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

Proof

Ballots $\underline{b}$ and $\underline{b'}$ are plurality-equivalent *iff* for each alternative x the number of times x occurs in the the top position in $\underline{b}$ is equal to the number of times x occurs in the the top position in $\underline{b'}$.

There are n (= number of voters) top positions available.

How many ways are there to award top positions to m alternatives, so that the sum of numbers of top positions adds up to n?

That is, how many solutions are there for this equation?

$$X_1 + X_2 + \cdots + X_m \quad = \quad n \quad \text{with } X_i \in \{0, 1, 2, \ldots\}$$

This is equal to the number of ways of picking n balls from an urn with m balls *with replacement*, which in turn is equal to picking n balls from an urn with $m + n - 1$ balls *without replacement*: $\binom{m+n-1}{n}$.

The logarithm of the above is the compilation complexity. ✓

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 5 (Chevaleyre et al., 2010) *The compilation complexity of the Borda rule (for an unknown number of additional voters) is in* $\Theta(m \log(nm))$.

Proof: We have to prove an upper and a lower bound.

- Upper bound: next slide
- Lower bound: proof omitted. The idea is to exhibit a sufficiently large number of equivalence classes.

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 $O(m\log(nm))$. ✓

Communication Complexity

How much information do voters need to transmit to allow us to compute the winner of an election?

Let n be the number of voters and m the number of alternatives.
We need $\lceil \log m \rceil$ bits to communicate the identity of an alternative.
Upper bounds are easy to derive:

- Plurality: $O(n \log m)$ — each voter sends one alternative name
- Approval: $O(nm)$ — let each voter communicate a bit-string of length m to encode their approved subset of alternatives
- Borda (any rule with linear orders as ballots): $O(n\, m \log m)$ — each voter sends m alternative names in turn

Conitzer and Sandholm (2005) show that many of these bounds are tight, using tools from *communication complexity*.

V. Conitzer and T. Sandholm. Communication Complexity of Common Voting Rules. Proc. ACM Conference on Electronic Commerce 2005.

Upper Bound for STV

Recall STV: eliminate plurality losers until an alternative gets $> 50\%$

In the worst case, the full ranking of some voters may be required to compute the winner. Clearly, $O(n\,m\log m)$ is an upper bound.

Conitzer and Sandholm (2005) show that this is *not* a tight upper bound and give a better one: $O(n\log^2 m)$

- Protocol: in the first round, each voter names their favourite alternative; in subsequent rounds any voter whose favourite alternative has just been eliminated names their new favourite
- Insight: the number of voters who get their favourite alternative eliminated is small, namely $\leqslant \frac{n}{m-i+1}$ in round i
- $\sum_{i=1}^{m-1} \frac{n}{m-i+1} = n \cdot \sum_{i=2}^{m} \frac{1}{i} \leqslant n \ln m$ (harmonic number)
  So: at most $O(n\log m)$ transmissions of size $O(\log m)$ ✓

Compilation vs. Communication Complexity

What are the connections between the compilation complexity and the communication complexity of a voting procedure?

Not straightforward:

- Compilation complexity is related to communication complexity: it is the complexity of communicating the ballots of the first n voters to the centre—in one go, as here we don't have the freedom of an interactive protocol.
- But an important difference is that for compilation purposes, we need to be able to identify an equivalence class of profiles, rather than just a set of winners.

Indeed, the known results in the literature suggest that there are no simple links between compilation and communication complexity.

Summary

We have discussed several situations where only *partial information* regarding the ballots is available. Examples:

- Some voters arrive late.
- Some alternatives enter the election late.
- Elicitation proceeds in stages (ballots are partial orders).

This gives rise to a number of questions:

- *Possible winners:* which alternatives do have a chance of winning?
- *Necessary winners:* will certain alternatives surely win?
- *Compilation:* how can we efficiently represent the relevant information elicited so far?

Finally, we have discussed the *communication complexity* of voting procedures and how it relates to the complexity of compilation.