Compilation Complexity of Common Voting Rules*

Lirong Xia
Department of Computer Science
Duke University
Durham, NC 27708, USA
lxia@cs.duke.edu

Vincent Conitzer
Department of Computer Science
Duke University
Durham, NC 27708, USA
conitzer@cs.duke.edu

Abstract

In computational social choice, one important problem is to take the votes of a subelectorate (subset of the voters), and summarize them using a small number of bits. This needs to be done in such a way that, if all that we know is the summary, as well as the votes of voters outside the subelectorate, we can conclude which of the m alternatives wins. This corresponds to the notion of *compilation complexity*, the minimum number of bits required to summarize the votes for a particular rule, which was introduced by Chevaleyre et al. [IJCAI-09]. We study three different types of compilation complexity. The first, studied by Chevaleyre et al., depends on the size of the subelectorate but not on the size of the complement (the voters outside the subelectorate). The second depends on the size of the complement but not on the size of the subelectorate. The third depends on both.

We first investigate the relations among the three types of compilation complexity. Then, we give upper and lower bounds on all three types of compilation complexity for the most prominent voting rules. We show that for l-approval (when $l \leq m/2$), Borda, and Bucklin, the bounds for all three types are asymptotically tight, up to a multiplicative constant; for l-approval (when $l > m/2$), plurality with runoff, all Condorcet consistent rules that are based on unweighted majority graphs (including Copeland and voting trees), and all Condorcet consistent rules that are based on the order of pairwise elections (including ranked pairs and maximin), the bounds for all three types are asymptotically tight up to a multiplicative constant when the sizes of the subelectorate and its complement are both larger than $m^{1+\epsilon}$ for some $\epsilon > 0$.

Introduction

In multiagent systems, often the agents want to make a joint decision, despite the fact that they have different preferences over the m alternatives. A natural way to do this is by *voting* over the alternatives. In a typical voting setting, the voters (agents) first report their (ordinal) preferences simultaneously. Then, a *voting rule* is applied to choose the winning alternative. However, the assumption that the voters report their votes simultaneously is often violated in real-life: usually, there is some period for voting, and each agent can cast her vote anywhere in this period (but only once). In this case, it can be practical to concisely summarize (compile) the votes that have arrived so far. This has a potential privacy benefit: in principle, the detailed information of exactly which voter cast which vote can be thrown away, so that there is less risk that an outside party finds out how a particular voter voted.[1] As we will see, this approach in some cases also allows the winner to be announced much faster after the last vote arrives, compared to the case where we do not do any computation until the very end.

There are additional benefits to compiling the votes of a subelectorate. For example, suppose that votes are collected in multiple locations. Rather than bringing all the votes (whether in physical or digital form) to the same place, the votes can simply be compiled locally, and then forwarded to a central office. Compiling the votes of subelectorates is also algorithmically useful for computing the backward induction winner in Stackelberg voting games (Xia and Conitzer 2010).

Chevaleyre et al. (2009) introduced the notion of *compilation complexity* as the minimum number of bits needed to store the summary of the votes of the subelectorate for a voting rule. More precisely, the setting is as follows. Suppose the votes are divided into the subelectorate, consisting of k known votes, and the complement, consisting of u unknown votes. In the Chevaleyre et al. framework, u is considered to be unknown. Hence, the compilation complexity is defined as a function of the voting rule r and k, as follows. It is the least number of bits needed to be able to summarize any profile of k votes, in such a way that for any number u, if we are given any profile of the u votes in the complement, then from this profile and the summary of the subelectorate, we can infer the winner under r for the total electorate. Formally, when summarizing a subelectorate, a profile P for the subelectorate is mapped to a string of bits $f(P) \in \{0,1\}^*$, representing the votes in the subelectorate. We require that for any two profiles P_1 and P_2, if $f(P_1) = f(P_2)$, then for any profile Q for the complement

*We thank Yann Chevaleyre, Jérôme Lang, Nicolas Maudet, and anonymous AAAI reviewers for helpful comments. Lirong Xia is supported by a James B. Duke Fellowship and Vincent Conitzer is supported by an Alfred P. Sloan Research Fellowship. We also thank NSF for support under award numbers IIS-0812113 and CAREER 0953756.

[1] This may make it more difficult to verify the winner of the election later so it requires a large degree of trust in the system.

(which can have any size in the Chevaleyre et al. setting), we must have $r(P_1 \cup Q) = r(P_2 \cup Q)$, where r is the voting rule. (If not, then f is leaving out relevant information.) f is called a *compilation function*. For given r, m, k, some compilation function f uses the minimum number of bits; that number of bits is the compilation complexity, denoted by $C_{m,k,?}(r)$. The question mark indicates that the number u is unknown.

It has been pointed out (Chevaleyre et al. 2009) that $C_{m,k,?}(r)$ is related to the following core computational social choice problems. It is related to the *complexity of terminating elicitation* (Conitzer and Sandholm 2002; Walsh 2008): given a subset of the votes, is the winner already determined? This problem, in turn, is related to the *complexity of possible and necessary winner determination* (Konczak and Lang 2005; Pini et al. 2007; Xia and Conitzer 2008a; Betzler, Hemmann, and Niedermeier 2009): given a set of incomplete votes (modeled as partial orders) and an alternative c, we are asked whether there exists an extension (resp., for any extension), c is the winner. Another related problem is the *communication complexity of voting rules* (Conitzer and Sandholm 2005): what is the smallest number of bits that must be transferred (among the voters and the center) to compute the winner?

We can extend the Chevaleyre et al. framework by considering cases where the number of unknown votes, u, is known (even though the votes themselves are not). This is a natural assumption in many cases where we know the size of the electorate. When both k and u are known, the compilation complexity is $C_{m,k,u}(r)$. We note that $C_{m,k,u}(r) \leq C_{m,k,?}(r)$, because if the summary contains enough information to infer the winner for any number of additional votes, it also contains enough to infer it for a specific number u of unknown votes. Moreover, in some cases this inequality is strict. As an illustrative example, in the extreme case where $u = 0$, the only summary that we need is simply which alternative won, so that $C_{m,k,0}(r) = \lceil \log m \rceil$ for any m and r. In fact, that argument does not even depend on k—so we can also say that $C_{m,?,0}(r) = \lceil \log m \rceil$. Here, $C_{m,?,u}(r)$ corresponds to using a compilation function that works for any profile consisting of any number of known votes, but a fixed number of unknown votes—in some sense, the opposite of the Chevaleyre et al. framework. It always holds that $C_{m,k,u}(r) \leq C_{m,?,u}(r)$.

Chevaleyre et al. focused strictly on $C_{m,k,?}(r)$, and conducted case studies for some common voting rules. While these results provide upper bounds on $C_{m,k,u}(r)$, these upper bounds are good only when u is much larger than k; and, as we will see, they will become quite loose when k is much larger than u. The authors briefly discussed the idea of using $C_{m,?,u}(r)$ as another upper bound. However, no tight bound was proved for the most intriguing, general case $C_{m,k,u}(r)$ (both k and u are known) for any common voting rule.

Our contributions. In this paper, we first study the relations among $C_{m,k,?}(r)$, $C_{m,?,u}(r)$, and $C_{m,k,u}(r)$. (For all common voting rules r, $C_{m,?,?}(r) = \infty$.) We recall that $C_{m,k,?}(r)$ and $C_{m,?,u}(r)$ are upper bounds on $C_{m,k,u}(r)$ (Chevaleyre et al. 2009). Our first result shows that, conversely, $C_{m,k,?}(r)$ and $C_{m,?,u}(r)$ can also be bounded above by a function that takes limits of $C_{m,k,u}(r)$ in a particular way, for anonymous voting rules r satisfying a condition called *h-canceling-out* (which includes all the rules in this paper).

Then, we study upper and lower bounds on compilation complexity for most common voting rules in use, including l-approval, Borda, Bucklin, Copeland, maximin, plurality with runoff, ranked pairs, and voting trees. We obtain asymptotically tight bounds on the three types of compilation complexity for some rules. Notably, when k and u are both at least slightly larger than m (more precisely, both are larger than $m^{1+\epsilon}$ for some $\epsilon > 0$), all bounds derived in this paper are asymptotically tight. Our results are summarized in Table 1.

Discussion. Among the three types of compilation complexity, $C_{m,k,?}$ and $C_{m,k,u}$ seem the most natural. The former is useful when we do not know how many unknown votes there are; the latter is useful when we do. The former also serves as a useful upper bound on the latter.

It must be admitted that $C_{m,?,u}$ is clearly less natural. Nevertheless, it does fit certain situations. For example, suppose that the members of a coalition want to manipulate the election and vote at the last moment, based on everybody else's votes. They need to summarize the votes submitted by everyone else, and then submit their votes in response immediately. A natural strategy for them is the following: in advance of the election, build a lookup table that maps each possible summary of the others' votes to a bundle of votes that the coalition will submit. Hence, the coalition requires a compilation function, preferably one that results in a small table (that is, one with low compilation complexity). However, when the coalition builds the table, it may not be clear yet how large the number k of voters outside the coalition will be. If so, their summarization scheme needs to work for subelectorate profiles of all possible sizes, corresponding to $C_{m,?,u}$. (If k is known in advance, this problem corresponds to $C_{m,k,u}$.) Indeed, as we will see, if u (the coalition size) is small, then the lookup table will be small.

That being said, we believe that the main use of $C_{m,?,u}$ is as an upper bound on $C_{m,k,u}$, as suggested by Chevaleyre et al. In fact, perhaps the simplest approach to compiling votes when both k and u are known is the following. If k is smaller than u, the chair compiles the votes as if she does not know u (corresponding to $C_{m,k,?}$); if u is smaller than k, the chair compiles the votes as if she does not know k (corresponding to $C_{m,?,u}$). Our lower bounds on $C_{m,k,u}$ show that, surprisingly, this very intuitive and simple approach is almost the best we can do for most common voting rules, in an asymptotic sense.

Preliminaries

Let $\mathcal{X}$ be the set of *alternatives*, $|\mathcal{X}| = m$. A vote is a linear order over $\mathcal{X}$. The set of all linear orders over $\mathcal{X}$ is denoted by $L(\mathcal{X})$. An *n-profile* P is a collection of n votes for some $n \in \mathbb{N}$, that is, $P \in L(\mathcal{X})^n$. A *voting rule* r is a mapping that assigns to each profile a unique winning alternative. That is, $r : \{\emptyset\} \cup L(\mathcal{X}) \cup L(\mathcal{X})^2 \cup \ldots \rightarrow \mathcal{X}$. Some common voting rules are listed below. Ties are assumed to be broken according to a fixed order over the alternatives, $c_1 > \ldots > c_m$.

• **(Positional) scoring rules**: Given a *scoring vector* $\vec{v} = (v(1), \ldots, v(m))$, for any vote $V \in L(\mathcal{X})$ and any $c \in \mathcal{X}$, let $s(V, c) = v(j)$, where j is the rank of c in V. For any profile $P = (V_1, \ldots, V_n)$, let $s(P, c) = \sum_{i=1}^{n} s(V_i, c)$. The rule will select $c \in \mathcal{X}$ so that $s(P, c)$ is maximized. Some examples of positional scoring rules are *Borda*, for which the scoring vector is $(m-1, m-2, \ldots, 0)$; *l-approval* (App_l, with $l \leq m$), for which the scoring vector is $v(1) = \ldots = v(l) = 1$ and $v_{l+1} = \ldots = v_m = 0$; *plurality*, for which the scoring vector is $(1, 0, \ldots, 0)$; and *veto*, for which the scoring vector is $(1, \ldots, 1, 0)$.
• **Copeland**: For any two alternatives c_i and c_j, we can simulate a *pairwise election* between them, by seeing how many votes prefer c_i to c_j, and how many prefer c_j to c_i; the winner of the pairwise election is the one preferred more often. Then, an alternative receives one point for each win in a pairwise election, and zero points for each loss. (The results in this paper hold regardless of the number of points for a tie.) The winner is the alternative that has the highest score.
• **Bucklin**: An alternative c's Bucklin score is the smallest number t such that more than half of the votes rank c among the top t alternatives. The winner is the alternative that has the lowest Bucklin score. (We do not consider any further tie-breaking for Bucklin.)
• **Maximin**: Let $N_P(c_i, c_j)$ denote the number of votes that rank c_i ahead of c_j in P. The winner is the alternative c that maximizes $\min\{N_P(c, c') : c' \in \mathcal{X}, c' \neq c\}$.
• **Plurality with runoff**: The election has two rounds. In the first round, all alternatives are eliminated except the two with the highest plurality scores. In the second round (runoff), the winner is the alternative that wins the pairwise election between them.
• **Ranked pairs**: This rule first creates an entire ranking of all the alternatives. $N_P(c_i, c_j)$ is defined as for the maximin rule. In each step, we will consider a pair of alternatives c_i, c_j that we have not previously considered; specifically, we choose the remaining pair with the highest $N_P(c_i, c_j)$. We then fix the ordering $c_i \succ c_j$, unless this contradicts orderings that we fixed previously (that is, it violates transitivity). We continue until we have considered all pairs of alternatives (hence we end up with a full ranking). The alternative at the top of the ranking wins.
• **Voting trees**: A voting tree is a binary tree with m leaves, where each leaf is associated with an alternative. In each round, there is a pairwise election between an alternative c_i and its sibling c_j: if a majority of voters prefers c_i to c_j, then c_j is eliminated, and c_i is associated with the parent of these two nodes; and vice versa. The alternative that is associated with the root of the tree (wins all its rounds) is the winner. The voting rule that corresponds to a balanced voting tree is also known as the *cup* rule.

For any profile P, we let M_P denote the *weighted majority graph* of P, defined as follows. M_P is a directed graph whose vertices are the alternatives. For $i \neq j$, if $N_P(c_i, c_j) > 0$, then there is an edge (c_i, c_j) with weight $w_{ij}(M_P) = N_P(c_i, c_j)$. Also, for $i < j$, if $N_P(c_i, c_j) = 0$, then there is an edge (c_i, c_j) with weight $w_{ij}(M_P) = 0$. We say a voting rule r is based on the:
• **Weighted majority graph (WMG)**, if for any pair of profiles P_1, P_2 such that $M_{P_1} = M_{P_2}$, we have $r(P_1) = r(P_2)$.
• **Order of pairwise elections (OPE)**, if for any pair of profiles P_1, P_2 with the property that for any $i_1, i_2, j_1, j_2 \leq m$, $w_{i_1 j_1}(M_{P_1}) \geq w_{i_2 j_2}(M_{P_1}) \iff w_{i_1 j_1}(M_{P_2}) \geq w_{i_2 j_2}(M_{P_2})$, we must have $r(P_1) = r(P_2)$. That is, the winner can be determined by comparing the magnitudes of pairwise results.
• **Unweighted majority graph (UMG)**, if for any pair of profiles P_1, P_2 such that for any i, j, $w_{ij}(M_{P_1}) \geq 0 \iff w_{ij}(M_{P_2}) \geq 0$, we have $r(P_1) = r(P_2)$. That is, only the signs (but not the magnitudes) of the pairwise results are used to determine the winner.

Note that any UMG-based rule is an OPE-based rule, and any OPE-based rule is a WMG-based rule. Copeland and voting trees are based on UMG; maximin and ranked pairs are based on OPE; and all of them are based on WMG.

A voting rule r is *Condorcet consistent* if it always selects the Condorcet winner (that is, the alternative that wins each of its pairwise elections) whenever one exists.

Let k denote the number of known votes in the subelectorate and let u denote the number of unknown votes in the complement. We first consider the case where k is known and u is not. For a voting rule r, a function $f^r_{m,k,?} : L(\mathcal{X})^k \to \{0,1\}^*$ is a *compilation function* if for any $P_1, P_2 \in L(\mathcal{X})^k$ with $f^r_{m,k,?}(P_1) = f^r_{m,k,?}(P_2)$, and any profile Q, we must have that $r(P_1 \cup Q) = r(P_2 \cup Q)$. Similarly, we can define compilation functions for the other two cases (u is known but k is unknown, and both u and k are known). In these cases, $f^r_{m,?,u} : \{\emptyset\} \cup L(\mathcal{X})^1 \cup L(\mathcal{X})^2 \cup \ldots \to \{0,1\}^*$ (resp., $f^r_{m,k,u} : L(\mathcal{X})^k \to \{0,1\}^*$) is a compilation function if for any pair of profiles (resp., k-profiles) P_1 and P_2, if $f^r_{m,?,u}(P_1) = f^r_{m,?,u}(P_2)$ (resp., $f^r_{m,k,u}(P_1) = f^r_{m,k,u}(P_2)$), then for any u-profile Q, we have that $r(P_1 \cup Q) = r(P_2 \cup Q)$.

Now, we are ready to define the three types of compilation complexity. Let $\text{len} : \{0,1\}^* \to \mathbb{N}$ return the length of a string. Then, for the case where k is known but u is not, $C_{m,k,?}(r) = \min_{f^r_{m,k,?}} \max_{P \in L(\mathcal{X})^k} \text{len}(f^r_{m,k,?}(P))$ gives the length of strings that we need to compile profiles, i.e., the compilation complexity. Similarly, $C_{m,k,u}(r) = \min_{f^r_{m,k,u}} \max_{P \in L(\mathcal{X})^k} \text{len}(f^r_{m,k,u}(P))$ and $C_{m,?,u}(r) = \min_{f^r_{m,?,u}} \max_{P \in \{\emptyset\} \cup L(\mathcal{X})^1 \cup L(\mathcal{X})^2 \cup \ldots} \text{len}(f^r_{m,?,u}(P))$. In each case, the min must be taken over valid compilation functions for that case.

Relations among the three types of compilation complexity

In this section, we invesigate the relations among the three types of compilation complexity for the same voting rule r. First, as Chevaleyre et al. (2009) pointed out, $C_{m,k,?}(r)$ and $C_{m,?,u}(r)$ are both at least as large as $C_{m,k,u}(r)$. They also provided loose upper bounds for any rule r ($O(k \log(m!))$ on $C_{m,k,?}$, and $O((m!)^u \log m)$ on $C_{m,?,u}$).

Lemma 1 (Chevaleyre et al. 2009) *$\forall m, k, u \in \mathbb{N}$, and for any rule r, $C_{m,k,u}(r) \leq \min\left(C_{m,k,?}(r), C_{m,?,u}(r)\right)$.*

This lemma will be frequently used in the paper in the following way. Once we obtain an upper bound on $C_{m,k,?}(r)$

or $C_{m,?,u}(r)$, we immediately obtain an upper bound on $C_{m,k,u}(r)$; conversely, once we obtain a lower bound on $C_{m,k,u}(r)$, we immediately have lower bounds on $C_{m,k,?}(r)$ and $C_{m,?,u}(r)$, respectively. We may also ask: If we obtain an *upper* bound on $C_{m,k,u}(r)$, can we use this to derive upper bounds on $C_{m,k,?}(r)$ and $C_{m,?,u}(r)$ in some way?

One naïve approach is to take the limit of $C_{m,k,u}(r)$ as $u \to \infty$ (to obtain a bound on $C_{m,k,?}(r)$). However, this approach does not work well. For one, it is not guaranteed that the limit of $C_{m,k,u}(r)$ when u goes to infinity exists. Another issue is that, even if $\lim_{u\to\infty} C_{m,k,u}(r)$ does exist, it might be smaller than $C_{m,k,?}(r)$, as shown in the following example. (A similar example exists for $C_{m,?,u}(r)$.)

Example 1 *Let $\{A, B, C\}$ be an arbitrary partition of the possible votes, $L(\mathcal{X})$. Let r be the voting rule defined as follows. For any profile P, we let:*

$$r(P) = \begin{cases} c_1 & \text{if } P \cap A \neq \emptyset \\ c_2 & \text{otherwise} \end{cases} \quad \text{if } |P| \text{ is odd;}$$

$$r(P) = \begin{cases} c_1 & \text{if } P \cap B \neq \emptyset \\ c_2 & \text{otherwise} \end{cases} \quad \text{if } |P| \text{ is even.}$$

If we know in advance that the total number of votes will be odd (even), then we only need to keep track of one bit of information, namely, whether there has been a vote in A (B). Hence, for any u, we have that $C_{m,k,u}(r) = 1$, so $\lim_{u\to\infty} C_{m,k,u}(r) = 1$. However, if we do not know whether there will be an odd or even number of votes, then when we want to compile P, we need to keep track of which of four possible states we are in: (1) $P \cap (A \cup B) = \emptyset$, (2) $P \cap A \neq \emptyset$ but $P \cap B = \emptyset$, (3) $P \cap B \neq \emptyset$ but $P \cap A = \emptyset$, (4) $P \cap B \neq \emptyset$ and $P \cap A \neq \emptyset$. Hence, we need two bits to keep track of all possibilities, so $C_{m,k,?}(r) = 2$.

Of course, the rule in Example 1 is very artificial. It turns out that common voting rules have structure that does allow us to obtain upper bounds on $C_{m,k,?}(r)$ and $C_{m,?,u}(r)$ from an upper bound on $C_{m,k,u}(r)$. This result works for all anonymous voting rules for which there exists an h-profile P that always cancels out.

Definition 1 *An anonymous voting rule r satisfies h-*canceling-out*, for $h \in \mathbb{N}$, if there exists an h-profile P such that for any profile P', $r(P \cup P') = r(P')$.*

Most common voting rules satisfy h-canceling-out for some h, e.g., all common rules studied in this paper (except Bucklin).

Proposition 1 *Plurality with runoff satisfies $2m$-canceling-out; all positional scoring rules satisfy m-canceling-out; Borda and all WMG-based voting rules satisfy 2-canceling-out.*

The next proposition states that for any r that satisfies h-canceling-out, the limit of $C_{m,k,u}(r)$ as u (or k) goes to infinity does exist, if we increase u (or k) by h each time.

Proposition 2 *Let r be an anonymous voting rule that satisfies h-canceling-out. For any k (resp., u) and any $0 \leq h' \leq h-1$, $\lim_{i\to\infty} C_{m,k,h'+ih}(r)$ (resp., $\lim_{i\to\infty} C_{m,h'+ih,u}(r)$) exists and is finite.*

Finally, we obtain the following bounds on $C_{m,k,?}(r)$ and $C_{m,?,u}(r)$: $C_{m,k,?}(r)$ (resp., $C_{m,?,u}(r)$) is bounded below by the largest $\lim_{i\to\infty} C_{m,k,h'+ih}(r)$ (resp., $\lim_{i\to\infty} C_{m,h'+ih,u}(r)$) among all $h' < h$, and is bounded above by the sum of such limits over all $h' < h$. This implies the lower and upper bounds are within a factor h.

Proposition 3 *Let r be an anonymous voting rule that satisfies h-canceling-out. For any $0 \leq h' \leq h-1$, let $p_{k,h'} = \lim_{i\to\infty} C_{m,k,h'+ih}(r)$ and $\hat{p}_{u,h'} = \lim_{i\to\infty} C_{m,h'+ih,u}(r)$. For any $k, u, m \in \mathbb{N}$, we have $\max_{h'} p_{k,h'} \leq C_{m,k,?}(r) \leq \sum_{h'=0}^{h-1} p_{k,h'}$ and $\max_{h'} \hat{p}_{u,h'} \leq C_{m,?,u}(r) \leq \sum_{h'=0}^{h-1} \hat{p}_{u,h'}$.*

Compilation complexity of common rules

In this section, we study upper and lower bounds on all three types of compilation complexity for common voting rules, including l-approval, Borda, Bucklin, plurality with runoff, all Condorcet-consistent rules (including Copeland, maximin, ranked pairs, and voting trees), all WMG-based rules (including all OPE-based rules and all UMG-based rules), all OPE-based rules (including maximin, ranked pairs, and all UMG-based rules), and all UMG-based rules (including Copeland and voting trees). For each voting rule r, we obtain bounds in the following way. We first derive upper bounds on $C_{m,k,?}(r)$ and $C_{m,?,u}(r)$, and apply Lemma 1 to obtain an upper bound on $C_{m,k,u}(r)$. Then, we derive a lower bound on $C_{m,k,u}(r)$, and apply Lemma 1 to obtain lower bounds on $C_{m,k,?}(r)$ and $C_{m,?,u}(r)$.

For each rule that we study, the upper bounds on $C_{m,k,?}(r)$ and $C_{m,?,u}(r)$ are proved by explicitly giving compilation functions $f^r_{m,k,?}$ and $f^r_{m,?,u}$. All our compilation functions share a common characteristic: $f^r_{m,k,?}$ maps each profile P to a vector that represents some type of scores (different types for different rules) that the alternatives obtain in the profile P. For example, in a plurality election, we only need to keep track of all alternatives' current plurality scores. For any k-profile P, $f^r_{m,?,u}(P)$ is obtained by deriving a (rule-specific) lower bound, such that if the value of one of the scores in $f^r_{m,k,?}(P)$ is below the lower bound, it will not make any difference to increase it to the lower bound—as long as there are only u remaining votes. For example, if in a plurality election, a receives 10 votes from the subelectorate and b only 3, and only $u = 4$ votes remain in the complement, we can increase b's score to 5 and b is still guaranteed to lose. Better yet, we can then subtract 5 from everyone's score (so that a is at 5 and b is at 0). If we do this, then we are guaranteed that the largest score that we ever need in $f^{\text{Plu}}_{m,?,4}(P)$ is 5.

The compilation functions $f^r_{m,k,?}$ and $f^r_{m,?,u}$ naturally induce a compilation function $f^r_{m,k,u}$ for $C_{m,k,u}(r)$, as follows. If the upper bound for $C_{m,k,?}(r)$ is smaller than the upper bound for $C_{m,?,u}(r)$, then we let $f^r_{m,k,u} = f^r_{m,k,?}$; otherwise, we let $f^r_{m,k,u} = f^r_{m,?,u}$. If we can prove a matching lower bound for $C_{m,k,u}(r)$ (in the asymptotic sense), then the compilation function $f^r_{m,k,u}$ defined above is optimal up to a multiplicative constant. To prove a lower bound on $f^r_{m,k,u}(r)$, we construct a set of k-profiles P that must be mapped to different strings by any compilation function. The lower bound is obtained by taking the logarithm of the set's size. The proofs for the lower bounds are technically much more involved. We only show how to construct compilation functions (in proof sketches). The remaining parts

of the proofs are omitted due to the space constraint. The full version with all proofs can be found online.

Theorem 1 (l-approval) *Let $l' = m - l$.*

• *When $l \leq m/2$, $C_{m,k,?}(App_l)$ is $\Theta\big(m\log(1 + kl/m) + kl\log(1 + m/kl)\big)$; $C_{m,?,u}(App_l)$ is $\Theta(m\log u)$; and $C_{m,k,u}(App_l)$ is $\Theta\big(\min(m\log(1+kl/m)+kl\log(1+m/kl), m\log u)\big)$.*

• *When $l > m/2$, $C_{m,k,?}(App_l)$ is $\Theta\big(m\log(1+kl'/m)+kl'\log(1 + m/kl')\big)$; $C_{m,?,u}(App_l)$ is $\Omega\big(m\log(ul'/m)\big)$ and is $O(m\log u)$; and $C_{m,k,u}(App_l)$ is $\Omega\big(\min(m\log(1 + kl'/m) + kl'\log(1 + m/kl'), m\log(ul'/m))\big)$ and is $O\big(\min(m\log(1+kl'/m)+kl'\log(1+m/kl'), m\log u)\big)$.*

Proof sketch. [$C_{m,k,?}(\text{App}_l)$] When $l \leq m/2$, we let $f^{\text{App}_l}_{m,k,?}(P) = (s(P,c_1),\ldots,s(P,c_m))$, that is, the total scores of the alternatives in P; when $l > m/2$, we let $f^{\text{App}_l}_{m,k,?}(P) = (k - s(P,c_1),\ldots,k - s(P,c_1))$, that is, the "complements" of the total scores of the alternatives in P.

[$C_{m,?,u}(\text{App}_l)$] Let P be a k-profile. Let c be an alternative that obtains the highest score under P. The idea in $f^{\text{App}_l}_{m,?,u}$ is that if the score of an alternative is smaller than $s(P,c) - u$, then that alternative cannot be the winner. Therefore, P can be compiled into a vector consisting of the differences between $s(P,c)$ and the scores of the alternatives in P, where the differences are capped above by $u + 1$. Formally, let $e_{max} = \max\big(s(P,c_1),\ldots,s(P,c_m)\big)$. We let $f^{\text{App}_l}_{m,?,u}(P) = (\max(s(P,c_1)+u+1-e_{max},0),\ldots,\max(s(P,c_m)+u+1-e_{max},0))$. □

Theorem 2 (Borda) *$C_{m,k,?}(Borda)$ is $\Theta(m\log(km))$,[2] $C_{m,?,u}(Borda)$ is $\Theta(m\log(um))$, and $C_{m,k,u}(Borda)$ is $\Theta(m\log(\min(km,um)))$.*

Proof sketch. [$C_{m,k,?}$(Borda)] Let $f^{\text{Borda}}_{m,k,?}$ be the compilation function that maps each k-profile to a vector of m natural numbers representing the total score of each alternative. That is, for any k-profile P, $f^{\text{Borda}}_{m,k,?}(P) = (s(P,c_1),\ldots,s(P,c_m))$ (Chevaleyre et al. 2009).

[$C_{m,?,u}$(Borda)] We let $f^{Borda}_{m,?,u}$ be the compilation function defined as follows. For any k-profile, suppose $f^{\text{Borda}}_{m,k,?}(P) = (e_1,\ldots,e_m)$. Let $e_{max} = \max(e_1,\ldots,e_m)$. We let $f^{\text{Borda}}_{m,?,u}(P) = (\max(e_1 + um + 1 - e_{max},0),\ldots,\max(e_m + um + 1 - e_{max},0))$. □

Theorem 3 (Bucklin) *For any k, u such that $\min(k,u) \geq m \geq 12$, $C_{m,k,?}(Bu)$ is $\Theta(m^2\log k)$, $C_{m,?,u}(Bu)$ is $\Theta(m^2\log u)$, and $C_{m,k,u}(Bu)$ is $\Theta(m^2\log(\min(k,u)))$.*

Proof sketch. [$C_{m,k,?}$(Bu)] Let $f^{\text{Bu}}_{m,k,?}$ be the compilation function that maps each profile to a vector of m^2 natural numbers, representing the number of times that each alternative is ranked in each position in the profile. Formally, let $f_{\text{Bu}}(P,c,i)$ be the number of times that c is ranked in the ith position. For any k-profile P, we let $f^{\text{Bu}}_{m,k,?}(P) = (f_{\text{Bu}}(P,c_1,1),\ldots,f_{\text{Bu}}(P,c_1,m),\ldots,f_{\text{Bu}}(P,c_m,1),\ldots,f_{\text{Bu}}(P,c_m,m))$.

[$C_{m,?,u}$(Bu)] Let $f^{*}_{\text{Bu}}(P,c,i)$ be the number of times that c is ranked anywhere in the top i positions. We note that if $f^{*}_{\text{Bu}}(P,c,i)$ is no more than $\lfloor(k+u)/2\rfloor - u$, then effectively it is the same as if $f^{*}_{\text{Bu}}(P,c,i)$ is $\lfloor(k+u)/2\rfloor - u$, because there are only u remaining voters. Similarly, if $f^{*}_{\text{Bu}}(P,c,i)$ is at least $\lfloor(k+u)/2\rfloor + 1$, then it is the same as if $f^{*}_{\text{Bu}}(P,c,i)$ is $\lfloor(k+u)/2\rfloor + 1$. Therefore, we let $f^{\text{Bu}}_{m,?,u}(P)$ consist of all the $f^{*}_{\text{Bu}}(P,c,i)$, bounded by the interval $\lceil\lfloor(k+u)/2\rfloor - u, \lfloor(k+u)/2\rfloor + 1\rceil$ (we can subtract $\lfloor(k+u)/2\rfloor - u\rceil$ to make the interval start at 0). □

Proposition 4 *For any UMG-based rule r_{UMG}, $C_{m,?,u}(r_{UMG})$ is $O(m^2\log u)$; for any OPE-based rule r_{OPE}, $C_{m,?,u}(r_{OPE})$ is $O(m^2\log(um))$.*

Proof sketch. For any UMG-based rule, the compilation function $f^{\text{UMG}}_{m,?,u}(P)$ produces a modified weighted majority graph M'_P (from the original majority graph M_P with weights $w_{ij}(M_P)$) in the following way. For each edge $c_i \to c_j$ such that $w_{ij}(M_P) > u+1$, let $w_{ij}(M'_P) = u+1$; for any other edge $c_i \to c_j$, let $w_{ij}(M'_P) = w_{ij}(M_P)$.

For any OPE-based rule, the compilation function $f^{\text{OPE}}_{m,?,u}(P)$ produces a different modified weighted majority graph M''_P: First, we order the non-negative weights of the edges in M_P from large to small. Let $e_1 \geq \ldots \geq e_{m(m+1)/2} \geq 0$ denote the weights. Then, we construct another set of weights for these edges, denoted by $e''_1 \geq \ldots \geq e''_{m(m+1)/2} \geq 0$, such that for any $j \leq m(m+1)/2$, if $e_j - e_{j+1} > u+1$, then $e''_j - e''_{j+1} = u+1$; otherwise, $e''_j - e''_{j+1} = e_j - e_{j+1}$ (and $e''_{m(m+1)/2+1} = 0$). Let M''_P denote the graph with the weights $e''_1,\ldots,e''_{m(m+1)/2}$. □

Proposition 5 *There exists a constant $q > 0$ such that for any Condorcet consistent voting rule r, $C_{m,k,u}(r)$ is $\Omega(m^2\log(\min(\lfloor k\log m/(qm)\rfloor - 1, u)))$, $C_{m,k,?}(r)$ is $\Omega(m^2\log(\lfloor k/(qm)\rfloor - 2))$,[3] and $C_{m,?,u}(r)$ is $\Omega(m^2\log u)$.*

The following two theorems follow directly from Propositions 4 and 5.

Theorem 4 (Condorcet-consistent OPE-based rules) *Let r be a Condorcet-consistent OPE-based rule (*e.g.*, maximin or ranked pairs). There exists a constant $q > 0$ such that the following holds.*

• *$C_{m,k,?}(r)$ is $\Omega(m^2\log(\lfloor k/qm\rfloor - 2))$ and is $O(m^2\log k)$;*
• *$C_{m,?,u}(r)$ is $\Omega(m^2\log u)$ and is $O(m^2\log(um))$;*
• *$C_{m,k,u}(r)$ is $\Omega(m^2\log(\min(\lfloor k\log m/(qm)\rfloor - 1, u)))$ and is $O(m^2\log(\min(k,um)))$.*

Theorem 5 (Condorcet consistent UMG-based rules) *Let r be a Condorcet-consistent UMG-based rule (*e.g.*, Copeland or a voting tree). There exists a constant $q > 0$ such that the following holds.*

• *$C_{m,k,?}(r)$ is $\Omega(m^2\log(\lfloor k/qm\rfloor - 2))$ and is $O(m^2\log k)$;*
• *$C_{m,?,u}(r)$ is $\Theta(m^2\log u)$;*
• *$C_{m,k,u}(r)$ is $\Omega(m^2\log(\min(\lfloor k\log m/(qm)\rfloor - 1, u)))$ and is $O(m^2\log(\min(k,u)))$.*

[2] This was proved in (Chevaleyre et al. 2009). There was a small mistake in the proof for the lower bound, which can be fixed.

[3] It was claimed that $C_{m,k,?}(r) = \Omega(m^2\log k)$ in (Chevaleyre et al. 2009). However, there was a small bug in their proof. Our proof uses a theorem by Erdös and Moser (1964).

Voting rule		$C_{m,k,?}$	$C_{m,?,u}$	$C_{m,k,u}$
l-approval ($l \leq m/2$)	(Theorem 1)	$\Theta\big(m \log(1+kl/m) + kl \log(1+m/kl)\big)$	$\Theta(m \log u)$	$\Theta\big(\min(m \log(1+kl/m) + kl \log(1+m/kl), m \log u)\big)$
l-approval ($l > m/2$)	(Theorem 1)	$\Theta\big(m \log(1+kl'/m) + kl' \log(1+m/kl')\big)$	$\Omega\big(m \log(ul'/m)\big)$ $O(m \log u)$	$\Omega\big(m \log(ul'/m)\big)$, and $O\big(\min(m \log(1+kl'/m) + kl' \log(1+m/kl'), m \log u)\big)$
Borda	(Theorem 2)	$\Theta\big(m \log(km)\big)$ [4]	$\Theta\big(m \log(um)\big)$	$\Theta\big(m \log \min(km, um)\big)$
Bucklin ($\min(u,k) \geq m \geq 12$)	(Theorem 3)	$\Theta(m^2 \log k)$	$\Theta(m^2 \log u)$	$\Theta\big(m^2 \log(\min(k,u))\big)$
Plurality with runoff ($k = \Omega(m^{1+\epsilon})$)	(Theorem 6)	$\Theta(m^2 \log k)$	$\Theta(m^2 \log u)$	$\Theta\big(m^2 \log(\min(k,u))\big)$
WMG-based rules (incl. UMG/OPE-based rules)	(Proposition 4)	$O(m^2 \log k)$ [4]	$O((m!)^u \log m)$ [4]	$O\big(\min(m^2 \log k, mu \log u)\big)$
OPE-based rules (incl. maximin and ranked pairs)	(Proposition 4)	$O(m^2 \log k)$ [4]	$O\big(m^2 \log(um)\big)$	$O\big(m^2 \log(\min(k,um))\big)$
UMG-based rules (incl. Copeland and voting trees)	(Proposition 4)	$O(m^2 \log k)$ [4]	$O(m^2 \log u)$	$O\big(m^2 \log(\min(k,u))\big)$
Condorcet consistent rules (incl. Copeland, maximin, ranked pairs, voting trees)	(Proposition 5)	$\Omega\big(m^2 \log(\lfloor k/(qm) \rfloor - 2)\big)$	$\Omega(m^2 \log u)$	$\Omega\big(m^2 \log(\min(u, \lfloor k \log m/(qm) \rfloor - 1))\big)$

Table 1: Compilation complexity of common voting rules.

Theorem 6 (Plurality with runoff) *For any $\epsilon > 0$, any $k = \Omega(m^{1+\epsilon})$, and any $u \in \mathbb{N}$, $C_{m,k,?}$(Pluo) is $\Theta(m^2 \log k)$, $C_{m,?,u}$(Pluo) is $\Theta(m^2 \log u)$, and $C_{m,k,u}$(Pluo) is $\Theta(m^2 \log(\min(k,u)))$.*

Proof sketch. For any k-profile P, let $f^{\text{Pluo}}_{m,k,?}(P) = (f^{\text{Plu}}_{m,k,?}(P), f^{\text{UMG}}_{m,k,?}(P))$ (Chevaleyre et al. 2009); for any profile P, we let $f^{\text{Pluo}}_{m,?,u}(P) = (f^{\text{Plu}}_{m,?,u}(P), f^{\text{UMG}}_{m,?,u}(P))$. □

Remark. For any compilation function f in this section and any profile P in the domain of f, computing $f(P)$ only takes polynomial time.

Conclusion and future work

In this paper, we studied three types of compilation complexity, denoted by $C_{m,k,?}$, $C_{m,?,u}$, and $C_{m,k,u}$. We first discussed the relations among them and showed that for any anonymous voting rule r that satisfies h-canceling-out, the limit of $C_{m,k,u}(r)$ when u (resp., k) goes to infinity can be used to compute an upper bound for $C_{m,k,?}(r)$ (resp., $C_{m,?,u}(r)$). We then conducted case studies of the three types of compilation complexity for common voting rules. The results are summarized in Table 1. We note that the bounds for l-approval ($l \leq m/2$), Borda, and Bucklin are asymptotically tight. For the others, when k and u are both sufficiently large (both are $m^{1+\epsilon}$ for some $\epsilon > 0$), the bounds are asymptotically tight. The upper bounds correspond to easy-to-compute compilation functions that can be used to compile the results of subelectorates, and the lower bounds show that we cannot do (much) better. Generally, the lower bounds are more difficult to prove.

Future research can take a number of directions. A natural direction is to investigate the compilation complexity of other rules, such as STV. Another direction is to consider situations where the votes come in one at a time and we need to maintain the compilation of the votes that have been cast so far, throughout the election. It is not difficult to see that the compilation functions in this paper can easily be updated as more votes come in. A larger direction is to more deeply understand the connections between compilation complexity and other concepts in computational social choice. For example, compilation complexity has similarities to communication complexity, as discussed in (Chevaleyre et al. 2009). Also, there is a relationship to the interpretation of voting rules as generalized scoring rules (GSRs) (Xia and Conitzer 2008b): we can always compile a voting rule by keeping track of the current GSR scores, resulting in an upper bound that depends on the number of scores in the GSR.

[4] Proved in (Chevaleyre et al. 2009).

References

Betzler, N.; Hemmann, S.; and Niedermeier, R. 2009. A multivariate complexity analysis of determining possible winners given incomplete votes. In *IJCAI*, 53–58.

Chevaleyre, Y.; Lang, J.; Maudet, N.; and Ravilly-Abadie, G. 2009. Compiling the votes of a subelectorate. In *IJCAI*, 97–102.

Conitzer, V., and Sandholm, T. 2002. Vote elicitation: Complexity and strategy-proofness. In *AAAI*, 392–397.

Conitzer, V., and Sandholm, T. 2005. Communication complexity of common voting rules. In *EC*, 78–87.

Erdös, P., and Moser, L. 1964. On the representation of directed graphs as unions of orderings. *Math. Inst. Hung. Acad. Sci.* 9:125–132.

Konczak, K., and Lang, J. 2005. Voting procedures with incomplete preferences. In *Multidisciplinary Workshop on Advances in Preference Handling*.

Pini, M. S.; Rossi, F.; Venable, K. B.; and Walsh, T. 2007. Incompleteness and incomparability in preference aggregation. In *IJCAI*, 1464–1469.

Walsh, T. 2008. Complexity of terminating preference elicitation. In *AAMAS*, 967–974.

Xia, L., and Conitzer, V. 2008a. Determining possible and necessary winners under common voting rules given partial orders. In *AAAI*, 196–201.

Xia, L., and Conitzer, V. 2008b. Generalized scoring rules and the frequency of coalitional manipulability. In *EC*, 109–118.

Xia, L., and Conitzer, V. 2010. Stackelberg Voting Games: Computational Aspects and Paradoxes. In *AAAI*.