Compilation and Communication Protocols for Voting Rules with a Dynamic Set of Candidates

Yann Chevaleyre
LIPN
Université Paris-Nord
Villetaneuse, France
chevaleyre@lipn.univ-paris13.fr

Jérôme Lang
LAMSADE
Université Paris-Dauphine
75775 Paris Cedex 16, France
lang@lamsade.dauphine.fr

Nicolas Maudet
LAMSADE
Université Paris-Dauphine
75775 Paris Cedex 16, France
maudet@lamsade.dauphine.fr

Jérôme Monnot
LAMSADE
Université Paris-Dauphine
75775 Paris Cedex 16, France
monnot@lamsade.dauphine.fr

ABSTRACT

We address the problem of designing communication protocols for voting rules when the set of candidates can evolve via the addition of new candidates.We show that the necessary amount of communication that must be transmitted between the voters and the central authority depends on the amount of space devoted to the storage of the votes over the initial set of candidates. This calls for a bicriteria evaluation of protocols. We consider a few usual voting rules, and three types of storage functions: *full storage*, where the full votes on the initial set of voters are stored; *null storage*, where nothing is stored; and *anonymous storage*, which lies in-between. For some of these pairs (voting rule, type of storage) we design protocols and show that they are asymptotically optimal by determining the communication complexity of the rule under the storage function considered.

Categories and Subject Descriptors

F.2.3 [**Theory of Computation**]: Tradeoffs among Complexity Measures; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence: Multiagent systems

General Terms

Theory

Keywords

voting; protocols; communication complexity

1. INTRODUCTION

The computational and knowledge-theoretic aspects of voting have received more and more attention in these last years. One prominent research stream focuses on the computational complexity of determining the winner of an election or of finding a strategic behaviour (such as manipulation, control, bribery etc.). An issue that has been significantly less often addressed is the *communication* complexity of voting (important exceptions being [4, 7] and [6], which we review below). When a group of agents have to find a common decision via a voting process, they must communicate their votes; this can be extremely costly in terms of communication, especially when the set of candidates has a very large size (for instance when it has a combinatorial structure). A high communication burden is a drawback that may severely hamper the practical applicability of a voting rule, perhaps even more than computational complexity, since the burden bears on individuals rather than on the computer.

Another important question has to do with the amount of space required for storing votes. In many contexts, the set of voters and/or the set of candidates may evolve in the course of the process. The question of preprocessing the information already gathered has been addressed recently in the context of a dynamic set of voters [2, 8]: the *compilation complexity* of a voting rule is the minimal amount of space needed to store the votes of the initial set of voters, so that the outcome can be determined once the last votes are known; this amount of space may vary in function of the number of remaining voters [8].

Now, not only the set of voters, but also the set of *candidates* can be dynamic. This class of situations has been recently studied in two papers [3, 9], which focus on the computational complexity of determining which of the initial candidates can still possibly win the election, but do not address compilation nor communication. A natural question that arises is how to compile a set of votes *about an initial set of candidates*, given that some new candidates may come later. It appears that defining a notion analogous to compilation complexity when the set of candidates evolves (instead of the set of voters) is not simple: when the set of *voters* is dynamic, it is natural to expect that the compilation of the first votes will be such that these voters will never have to communicate anything else. With an evolving set of candidates, it would not make sense to say that in the final stage, the voters will report their preferences *only about the new candidates*: except for very specific voting rules, we need the voters to compare these new candidates *between them* but also *with the initial candidates*. Therefore, it makes no sense to define the minimum amount of space needed to compute the final out-

ACM COPYRIGHT NOTICE. Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Publications Dept., ACM, Inc., fax +1 (212) 869-0481, or permissions@acm.org. TARK 2011, July 12-14, 2011, Groningen, The Netherlands. Copyright ©2011 ACM. ISBN 978-1-4503-0707-9, $10.00.

come once these new preferences are known, without saying anything about the amount of communication allowed in the second stage. A trivial response is to say that this minimum amount of space is zero: it suffices to elicit the votes entirely in the second stage. Now, in general we can save a significant amount of communication by storing part of the initial votes provided in the first stage. Therefore, what we have is *a trade-off between storage space and communication* —typically, the more storage space we use, the less communication we need in the second stage. Thus, instead of simply designing storage functions like in compilation complexity, or communication protocols like in communication complexity, here we design *compilation-communication* protocols, which specify how to compile the information provided in the first stage and how to elicit the information to be communicated in the second stage. Such protocols are evaluated with respect to both their storage space and their amount of communication.

The remainder of this paper is as follows. We start by briefly recalling background notions on voting in dynamic situations (Section 2). In Section 3 we make the model explicit, and relate it to standard communication and communication complexity of voting rules. In Section 4 we focus on the case where minimizing communication has the priority over minimizing the storage space, and we design asymptotically optimal protocols for a few usual voting rules. The optimality of these protocols is shown using the technique of *fooling sets*, borrowed from communication complexity [5]. In Section 5 we focus on another class of storage function, in particular where anonymity is required: the initial votes have to be compiled in a way that does not store the information “who votes what”. Section 6 concludes.

2. BACKGROUND

2.1 Voting rules

Let $\mathcal{C}$ be a finite set of candidates, with $|\mathcal{C}| = m$. Let $L(\mathcal{C})$ be the set of all linear orders over $\mathcal{C}$. A vote over $\mathcal{C}$ is an element of $L(\mathcal{C})$. We denote votes in the following way: $a \succ b \succ c$ is denoted by abc, etc. An *n-voter profile* P is a collection of n votes, that is, an element of $L(\mathcal{C})^n$. A voting rule is a function r from profiles to candidates. As the usual definition of most voting rules does not exclude the possibility of ties, we assume these ties are broken by a fixed priority order on candidates. In the paper we will mostly consider *scoring rules*. We start by defining scoring rule for a fixed number m of candidates. Given a *scoring vector* $\vec{s} = (s_1, \ldots, s_m)$ with $s_i \in N$ and $s_1 \geq \cdots \geq s_m$, for any vote $V \in L(\mathcal{C})$ and any $c \in \mathcal{C}$, let $s(V, c) = s_j$, where j is the rank of c in V. For any profile $P = (V_1, \ldots, V_n)$, let $s(P, c) = \sum_{i=1}^{n} s(V_i, c)$. Then the rule selects the candidate maximizing $s(P, c)$. In particular, *Borda* is based on the vector $(m-1, m-2, \ldots, 0)$, *K-approval* on the vector defined by $s_1 = \ldots = s_K = 1$ and $s_{K+1} = \ldots = s_m = 0$, and *plurality* on the vector $(1, 0, \ldots, 0)$. Because the number of candidates is not the same before and after the new candidates come in, we consider families of voting rules (for a varying number of candidates) rather than voting rules for a fixed number of candidates. Thus we have a collection $s^* = (\vec{s}^{\,m})_{m \geq 2}$ of scoring vectors, where for each m, $\vec{s}^{\,m} = (s_1^m, \ldots, s_m^m)$. For instance, Borda corresponds to $\vec{s}^{\,m} = (m-1, m-2, \ldots, 1, 0)$ for each m. For many usual voting rules such as Borda and K-approval, and *a fortiori* plurality and veto, there is an obvious way of defining them for a varying number of candidates, but this is not always the case. Lastly, with a slight abuse of language, we will identify the scoring rule (defined for an arbitrary number of candidates) induced by s^* with s^* itself: when we say “let s^* be a scoring rule” we mean “let r_{s^*} be scoring rule induced by a collection of scoring vectors s^*”.

2.2 Voting with a dynamic set of candidates

We recall this definition from [3]:

DEFINITION 1. *A* voting situation with varying candidates *is a quadruple $\Sigma = \langle N, X, P_X, k\rangle$ where N is a set of voters (with $|N| = n$), X a set of initial candidates (with $|X| = p$), $P_X = (V_1, \ldots, V_n)$ an n-voter X-profile, and k a positive integer.*

X denotes the set of initial candidates, P_X the initial profile, and k the number of new candidates (thus, $m = p + k$). If P is a C-profile and $C' \subseteq C$, then the projection of P on C', denoted by $P^{\downarrow C'}$, is obtained by deleting all candidates in $C \setminus C'$ in each of the votes of P, and leaving unchanged the ranking on the candidates of C'. For instance, if $P = \langle abcd, dcab\rangle$ then $P^{\downarrow\{a,b\}} = \langle ab, ab\rangle$ and $P^{\downarrow\{a,b,c\}} = \langle abc, cab\rangle$. The set of initial candidates is denoted by $X = \{x_1, \ldots, x_p\}$, and the set of the k new candidates by $Y = \{y_1, \ldots, y_k\}$. If P_X is an X-profile and P an $X \cup Y$-profile, then we say that P extends P_X if the projection of P on the candidates in X is exactly P_X. For instance, let $X = \{x_1, x_2, x_3\}$ and $Y = \{y_1, y_2\}$; the profile $P = \langle x_1y_1x_2y_2x_3, y_1y_2x_1x_2x_3, x_3x_2y_2y_1x_1\rangle$ extends the X-profile $P_X = \langle x_1x_2x_3, x_1x_2x_3, x_3x_2x_1\rangle$.

3. COMPILATION-COMMUNICATION PROTOCOLS

Given that all voters have already expressed their votes on X and that k new candidates $Y = \{y_1, \ldots, y_k\}$ come in, two questions arise: (i) how can we preprocess P_X so as to avoid restarting eliciting the preferences on $X \cup Y$ entirely? (ii) given the resulting compilation, how should we elicit the voters’ remaining preferences as cheaply as possible? It turns out that both questions are interdependent: the more information we store from P_X, the cheaper the elicitation about the new candidates. Therefore we have a *compilation-communication tradeoff*. To deal with this situation, we introduce the notion of *communication protocol modulo a compilation function*, or for short a *compilation-communication protocol*.

We briefly recall the definition of a (deterministic) voting protocol (see [5] for a general presentation of protocols and communication complexity, and [4] for voting protocols). Given a voting rule r, the voters have to compute together the winner, given that initially, every voter i knows only her vote V_i. In a protocol for computing $r(V_1, \ldots, V_n)$, in each stage, one of the voters announces a bit of information based on her private vote and the bits announced so far. Eventually, the communication terminates and all players know $r(V_1, \ldots, V_n)$. The goal is to minimize the worst-case (over all input vectors) number of bits sent. The deterministic communication complexity of a voting rule is the worst-case number of bits sent in the best deterministic protocol that computes it. The communication complexity of voting rules has been addressed by Conitzer and Sandholm [4], who give asymptotic bounds for a number of prominent voting rules. Segal [7] also analyzed the communication complexity of social choice rules in a broader sense, thanks to an original proof technique (but which requires a property *not* satisfied by most voting rules). Procaccia [6] gives an exact bound for the problem of determining the Condorcet winner (if any), using a different communication model.

The difference between these two frameworks and ours is that we start with some initial knowledge, consisting of the compilation of the initial votes. There are two (almost equivalent) ways of modelling the problem.

(a) a fully *distributed* model where the parties are the voters, and the private input of a voter consists of her complete vote plus the compiled initial profile;

(b) a *mediated* model where the parties are the voters plus a center ca, the private input of a voter consists of her complete vote, and the private knowledge of ca is the compilation of the initial votes.

The distributed model is typically relevant in multiagent system applications, while the mediated model fits more naturally online voting applications. While Conitzer and Sandholm [4] chose not to consider ca as an explicit party, we also consider the opposite choice.The main technical difference between both models is that the communication bits sent by ca are counted in (b) but not in (a). Both assumptions make sense. Counting the bits sent by ca (and received by the voters) corresponds to viewing ca as an agent like the others. Not counting them amounts to counting only the bits that the voters *send*, and not those they *receive*; this makes sense, because the communication burden on the voters is mainly caused by the need to send messages; receiving messages can therefore be considered to have no cost. We refer to these two models as *sending-receiving communication complexity* for (b), and *sending-only communication complexity* for (a). In all our protocols (with one exception that we discuss further), the amount of communication sent by ca is at most of the same order of magnitude as the amount of communication sent by the voters, hence the communication complexities of the protocol under the two models (a) and (b) are asymptotically equivalent. The same remark applies to the results in [4]: while their results are established under assumption (a), it is easy to show that they hold under assumption (b) as well.

DEFINITION 2. *Let N be a set of voters, X a set of initial candidates X, Y a set of new candidates, and r a voting rule. A* compilation-communication protocol *(CCP) for $\langle N, X, Y, r\rangle$ is a pair $\mu = \langle \sigma, \pi\rangle$ where*

- *σ is a compilation function, mapping any X-profile P_X into a string $\sigma(P_X)$.*
- *π is a communication protocol for computing $r(V_1, \ldots, V_n)$, given that the knowledge is distributed between $n+1$ parties: each voter i knows her own vote V_i over $X \cup Y$, and the center knows only $\sigma(P_X)$, where $P_X = \langle V_1^{\downarrow X}, \ldots, V_n^{\downarrow X}\rangle$.*

The *compilation cost of protocol μ for profile P_X*, denoted by $Comp(\mu, P_X)$, is the size of $\sigma(P_X)$. The *compilation cost of μ*, denoted by $Comp(\mu)$, is the worst-case compilation cost when considering all profiles, that is, $Comp(\mu) = \max_{P_X \in L(X)^n} Comp(\mu, P_X)$. Likewise, the *communication cost of μ for P_X*, denoted by $Comm(\mu, P_X)$, is the cost of π when applied to P_X, and the *communication cost of μ*, denoted by $Comm(\mu)$, is its worst-case communication cost when considering all profiles, or, equivalently, the communication complexity of π. Importantly, *we do not care about how the information $\sigma(P_X)$ was communicated to the center*, and the communication during this phase is not counted. The rationale behind this is that the elicitation of P_X is assumed to performed off-line, in a context where communication is not an issue. The *cost* of π is the vector

$$C(\mu) = \langle Comp(\mu), Comm(\mu)\rangle$$

At the two extremities of the spectrum we have two compilation functions:

μ_N**:** ***no storage.*** We do not store anything, *i.e.*, $\sigma_N(P_X)$ is the empty string.

μ_F**:** ***full storage.*** We store each ballot fully, voter by voter, *i.e.*, $\sigma_F(P_X) = P_X$.

Another natural compilation function consists in forgetting the identity of the voters. This makes sense especially if full storage has to be avoided for privacy reasons.

μ_A**:** ***anonymous storage.*** For each possible ranking $\succ$ on the initial candidates, we store the *number* of voters whose vote was $\succ$. Formally, the compilation $\sigma_A(P_X)$ of a profile P_X maps every ranking $\succ$ into a number $\sigma_A(P)(\succ)$, such that $\sum_{\succ \in L(X)} \sigma_A(P)(\succ) = n$.

EXAMPLE 1. *Let r be plurality, $n = 4$, $X = \{x_1, x_2, x_3\}$, $P_X = \langle x_1x_2x_3, x_2x_3x_1, x_3x_2x_1, x_1x_2x_3\rangle$, and $Y = \{y_1, y_2\}$. Consider the protocols:*

- *$\mu = \langle \sigma, \pi\rangle$ where $\sigma(P)$ stores the top candidate in X of every voter, therefore $\sigma(P_X) = \langle x_1, x_2, x_3, x_1\rangle$. π asks every voter to specify who is her preferred candidate among y_1, y_2 and her preferred candidate in X. The cost of μ is $\langle n \log p, n \log(k+1)\rangle$ (here with $p = 3$ and $k = 2$).*
- *$\mu_N = \langle \sigma_N, \pi_N\rangle$ where π_N asks every voter to specify her preferred candidate in $X \cup Y$. The cost of μ_N is $\langle 0, n \log(p+k)\rangle$.*
- *$\mu_F = \langle \sigma_F, \pi_F\rangle$, and $\pi_F = \pi$ as above. The cost of μ_F is $\langle n \log p!, n \log(k+1)\rangle$: storing P_X entirely instead of the top candidates does not help saving any bit of communication.*

Since $C(\pi)$ is not a single number but a vector of two numbers, defining the compilation-communication complexity of a rule with respect the addition of k new candidates as the cost of the best CC-protocol (as do [4] for the case of communication complexity and [2] for the case of compilation complexity) does not work. Let μ and μ' be two CC-protocols for r; we way that μ *(Pareto-)dominates* μ', denoted by $\mu \ll \mu'$, if $Comp(\mu) \leq Comp(\mu')$ and $Comm(\mu) \leq Comm(\mu')$, with one of these two inequalities being strict. A CC-protocol μ is Pareto-optimal if there is no μ' such that $C(\mu') \ll C(\mu)$. The set of best protocols for r is simply the set of all Pareto-optimal protocols, denoted by $BestProtocols(r)$, and the CC-complexity of r is defined as

$$CC(r) = \{C(\mu) \mid \mu \in BestProtocols(r)\}$$

In Example 1, μ_F is Pareto-dominated by μ, hence it is not among the best protocols.

Now, we are mainly interested in deriving bounds for the CC-complexity of a rule. Given a vector $\langle \alpha, \beta\rangle$ of integers, we say that $CC(r) \leq \langle \alpha, \beta\rangle$ if there exists $\langle \gamma, \delta\rangle$ in $CC(r)$ such that $\gamma \leq \alpha$ and $\delta \leq \beta$. Also, it makes sense to fix the compilation function σ and to look for an optimal protocol π such that $\langle \sigma, \pi\rangle$ is a compilation-communication protocol for r. This leads us to define the *communication complexity of r given to the compilation function σ* as the communication complexity of an optimal protocol π for r such that $\langle \sigma, \pi\rangle$ is a CC-protocol for r.

DEFINITION 3. *Let σ_1 and σ_2 be two compilation functions. We say that σ_2 is at least as strong as σ_1 (denoted by $\sigma_1 \rhd \sigma_2$), if there exists a compilation function σ' such that $\sigma'(\sigma_1(P_X)) = \sigma_2(P_X)$.*

Equivalently, σ_2 is at least as strong as σ_1 if σ_1 is at least as informative as σ_2, *i.e.*, if σ_1 retains at least as much information from P_X as σ_2. Note that two compilation functions may be incomparable in that respect. Clearly, the weaker the compilation function (*i.e.*, the more the central authority knows about

P_X), the cheaper the communication protocol. More precisely, for two CCP $\mu_1 = \langle\sigma_1, \pi_1\rangle$ and $\mu_2 = \langle\sigma_2, \pi_2\rangle$, if $\sigma_1 \rhd \sigma_2$ then $Comm(\mu_1) \leq Comm(\mu_2)$. Hence a lower bound on the communication complexity remains a lower bound for any stronger compilation function. We clearly have $\sigma_F \rhd \sigma_A \rhd \sigma_N$. Note that being less informative does not imply being less succinct, *i.e.*, $\sigma_1 \rhd \sigma_2$ does not imply $|\sigma_1| \geq |\sigma_2|$.

In the null storage case, the communication will be maximum. Note that since nothing from P_X has been stored, the distinction between the candidates in X and those in Y disappears, and we are therefore in the same setting as Conitzer and Sandholm [4]. Therefore, being an optimal protocol given the compilation function σ_N is equivalent to being an optimal protocol for a voting rule, and the cost of this protocol will have the form $\langle 0, \alpha\rangle$, where α is the communication cost of an optimal protocol for r given σ_N, and therefore, *the communication complexity of r given σ_N is the (deterministic) communication complexity of r as in [4]. In this case the results for specific rules can be directly retrieved from [4]*, and from now on we will not focus any longer on the null storage case.

To derive a lower bound on the communication complexity, one of the most popular techniques (although not always the best one) consists in exhibiting a *fooling set* of cardinality $2^{f(n,p,k)}$, that is, a set of n-voter $X \cup Y$-profiles subject to the constraint that any profile and the "fooling" mixture must be possible completions of the initial profile (given the compilation). We first give the definition of fooling sets. To make things simpler we formulate the definition directly in the context of voting rules, rather than in the general framework – see Section 3, and especially the first paragraph of page 4, of [4]. The definition is different due to the fact that the fooling sets must in our context be compatible with the initial profile.

DEFINITION 4. *Let r be a voting rule, P_X an n-voter initial profile on X, and $\sigma(P_X)$ a compilation of P_X. A fooling set for r extending P_X w.r.t. σ is a set of n-voter $X \cup Y$-profiles $P^1 = \langle V_1^1, \ldots, V_n^1\rangle$, $P^2 = \langle V_1^2, \ldots, V_n^2\rangle$, $P^q = \langle V_1^q, \ldots, V_n^q\rangle$ such that the following three conditions hold:*

- *there exists a candidate $c \in X \cup Y$ such that for any $i \leq q$, $r(P^i) = c$;*
- *for any $i \leq q$, $\sigma((P^i)^{\downarrow X}) = \sigma(P_X)$, where $(P^i)^{\downarrow X}$ is the projection of P^i to X;*
- *for any $1 \leq i \neq j \leq q$, there exists some vector $(s_1, \ldots, s_n) \in \{i, j\}^n$ such that $r(V_1^{s_1}, \ldots, V_n^{s_n}) \neq c$ and $\sigma((V_1^{s_1}, \ldots, V_n^{s_n})^{\downarrow X}) = \sigma(P_X)$.*

Then we use the classical result that the communication complexity of r is at least the logarithm of the size of any fooling set.

4. FULL STORAGE

In this Section we assume that the compilation function is σ_F. In this case, the communication will be minimum. Full storage is an obvious choice if the cost of storage space is negligible with respect to the cost of on-line communication, which is the case in most contexts. Let $Comm_F(r)$ be the communication complexity of r given full storage, and recall that $Comp_F(r) = n\log(p!)$. To derive an upper bound $f(n, p, k)$ (where n is the number of voters and p the number of initial candidates) of a voting rule r (depending on the numbers of voters n, initial candidates p and new candidates k), we must exhibit a protocol π which, whatever the initial profile P_X, determines the outcome using at most $f(n, p, k)$ communication bits. The straightforward protocol π consisting in asking each voter the rank of each new candidate y_i in her vote leads to a protocol $\langle\sigma_F, \pi\rangle$ in $CC(r)$.

PROPOSITION 1. *For any voting rule, the communication complexity of r with respect to the addition of k new candidates and full storage is $O(n(\log(p+k)! - \log(p!))$.*

PROOF. The central authority knows how a voter ranks candidates in X. The number of rankings of $X \cup Y$ that extend this fixed ranking on X is $\frac{(p+k)!}{p!}$. Therefore, a voter can communicate his vote V to the central authority, who knows already V_X using $\log\frac{(p+k)!}{p!} = \log(p+k)! - \log p!$ bits. ■

Now we study more specifically the communication complexity of several voting rules. We start by K-approval (whose communication complexity has never been addressed yet), then we consider Borda and Copeland.

4.1 K–approval

Clearly, it suffices for each voter to send the identities of the new candidates she approves. Depending on K and the number k of new candidates, this information can be communicated by two possible protocols:

π_1**:** send one bit (yes or no) for each of the k new candidates, to say whether it is approved or not;

π_2**:** give the names of the approved new candidates.

The communication cost of π_1 is in $O(nk)$, whereas that of π_2 is in $O(nK\log k)$. Therefore, π_1 is better if $K > \frac{k}{\log k}$, and π_2 is better if $K < \frac{k}{\log k}$. From this we get immediately:

PROPOSITION 2. *The communication complexity of K-approval with respect to the addition on k new candidates and full storage is in $O(\min(nK\log k, nk))$.*

Note that if K is a constant, then π_2 gets better at some point, and we have a communication complexity in $O(n\log k)$. However it is possible to design rules where K depends on the (total) number $m = p + k$ of candidates, such as $\sqrt{m}$-approval.

Importantly, the bound in Proposition 2 does not depend on the number p of initial candidates. Therefore we will try to obtain bounds for $p = 0$, which will also give the communication complexity of K-approval in the general case (or, equivalently, with null storage); this will complete the results established in [4], since they left K-approval out of their study.

PROPOSITION 3. *If $k \gg K$, the communication complexity of K-approval with respect to the addition on k new candidates and full storage is in $\Theta(nK.\log k)$.*

PROOF. The upper bound is a consequence of Proposition 2, after noticing that the bound is independent from p and thus holds for $p = 0$. For the lower bound, we exhibit a fooling set of size $k^{nK} = 2^{nK\log k}$. We have $n = \begin{pmatrix} k \\ K \end{pmatrix} = \frac{k!}{K!(k-K)!}$ voters: in a profile, we have one vote for every subset of K elements among the k (new) candidates. These n votes can be disposed in any order in the profile, therefore the number of profiles in the set F is $\left(\frac{k!}{K!(k-K)!}\right)!$. We have $\log\left(\frac{k!}{K!(k-K)!}\right)! \sim \frac{k!}{K!(k-K)!}\log\left(\frac{k!}{K!(k-K)!}\right) = n\log\left(\frac{k!}{K!(k-K)!}\right) \sim nK\log k$. Now, we have to show that F is a fooling set. First, in each of the profiles of F, each of the candidates gets exactly $\frac{Kn}{k}$ votes, therefore all candidates $y_1 \dots y_k$ are tied. Assume that after tie-breaking, the winner is y_1. Let P and Q be two profiles in F. Since $P \neq Q$, there must be a vote i such that $P_i \neq Q_i$, where P_i (resp. Q_i) is the set of K candidates approved by i in P (resp. in Q). Since $|P_i| = |Q_i|$, we have $Q_i \setminus P_i \neq \emptyset$. Consider the profile $P \otimes_i Q = \langle P_1, \dots, P_{i-1}, Q_i, P_{i+1}, \dots, P_n \rangle$. In $P \otimes_i Q$, every candidate in $Q_i \setminus P_i$ gets $\frac{Kn}{k} + 1$ votes, every candidate in $P_i \setminus Q_i$ gets $\frac{Kn}{k} - 1$ votes, and all other candidates still get $\frac{Kn}{k}$ votes. Therefore, the cowinners are the candidates in $Q_i \setminus P_i$. Symmetrically, the cowinners of the profile $Q \otimes_i P$ are the candidates in $P_i \setminus Q_i$ which are disjoint with $Q_i \setminus P_i$. Thus, after tie-breaking, either y_1 will not be the winner in $Q_i \setminus P_i$ or y_1 will not be the winner in $P_i \setminus Q_i$. This being true for any pair of profiles in F, we get that F is a fooling set[1]. ■

These results apply to plurality, by letting $K = 1$. Let $\mu_1 = \langle \sigma_F, \pi_1 \rangle$ and $\mu_2 = \langle \sigma_F, \pi_2 \rangle$. Since $Comp(\mu_1) = Comp(\mu_2) = n \log p!$, we have $CC(K\text{-approval}) \leq \langle n \log p!, \min(nk, nK \log k) \rangle$.

4.2 Borda and Copeland

Let $C(x, P)$ denote the Copeland score of candidate x in profile P, defined by $C(x, P) = \#\{y | x \succ^P_{maj} y\}$, where $x \succ^P_{maj} y$ stands for the fact that x is preferred over y by a majority of voters in P. The winner is the candidate maximizing $C(x, P)$ (or the candidate among these with highest priority). Next, we show that for the Borda and Copeland rules, the naive protocol described in Proposition 1 is asymptotically optimal.

PROPOSITION 4. *The communication complexity of Borda with respect to the addition of k new candidates and full storage is in* $\Theta(n(\log(p+k)! - \log(p!))$.

PROOF. The upper bound comes from Proposition 1. For the lower bound, we give a general fooling set which will work for several voting rules satisfying the *cancellation property*. We have $n = 2n'$ voters, p initial candidates $\{x_1, \dots, x_p\}$,k new candidates $\{y_1, \dots, y_k\}$, and an initial profile $P_X = \langle V_1, \dots, V_n \rangle$ satisfying the following property: for every $1 \leq i \leq n'$, if $V_{2i-1} = x_{\pi(1)} \succ x_{\pi(2)} \succ \dots x_{\pi(p)}$ then V_{2i} is the reversal of V_{2i-1}, *i.e.*, $V_{2i} = x_{\pi(p)} \succ \dots \succ x_{\pi(2)} \succ x_{\pi(1)}$. Now, the fooling set F contains the votes defined as follows. First, given a vote V_i over X and a vector $\vec{u} = (u^1, \dots, u^k)$ of k distinct elements of $\{1, \dots, p+k\}$, the vote $V_i[\vec{u}]$ is obtained by inserting in V_i candidate y_1 at rank u^1, y_2 at rank u^2, and so on, until y_k at rank u^k, and $Rev(V_i[\vec{u}])$ is the reversal of $V_i[\vec{u}]$. Now, for each collection $T = \langle \vec{t_1}, \dots, \vec{t_{n'}} \rangle$ of vectors $\vec{t_i} = (t^1_i, \dots, t^k_i)$ of k distinct elements of $\{1, \dots, p+k\}$, $P_X[T]$ is the profile containing, for every $i = 1, \dots, n'$, the votes $V_{2i-1}[\vec{t_1}]$ and $Rev(V_{2i-1}[\vec{t_1}])$. Note that $Rev(V_{2i-1}[\vec{t_1}])^{\downarrow X} = Rev(V_{2i-1}) = V_{2i}$, therefore $P_X[T]^{\downarrow X} = P_X$. For instance, let $P_X = \langle abc, cba, bca, acb \rangle$, $k = 2$, $\vec{t_1} = (3, 4)$ and $\vec{t_2} = (4, 1)$; we have $P_X[T] = \langle aby_1y_2c, cy_2y_1ba, y_2bcy_1a, ay_1cby_2 \rangle$. Now, the fooling set $F(P_X)$ is defined the set of all $P_X[T]$ obtained by letting T vary. We first count the number of profiles in $F(P_X)$. For a given $i \leq n'$, we have $p+k$ possibilities for the rank of y_1, then $p+k-1$ possibilities for the rank of y_2, and so on until y_k, for which we have $p+1$ possibilities. Therefore, for a given n' we have $(p+k)(p+k-1)\cdots(p+1) = \frac{(p+k)!}{p!}$ possibilities, and finally, in $F(P_X)$ we have $\left(\frac{(p+k)!}{p!}\right)^{n'}$ elements. Note that $\log\left(\frac{(p+k)!}{p!}\right)^{n'} = \frac{n}{2}(\log(p+k)! - \log(p!))$.

We now check that $F(P_X)$ is indeed a fooling set. In every profile $P_X[T]$ of $F(P_X)$, all candidates have the same Borda score, namely $n'(p+k-1)$, therefore the winner is the candidate with highest tie-breaking priority. Let $w \in X \cup Y$ be this candidate.

Let $P_X[T]$, $P_X[T']$ be two profiles in $F(P_X)$. Since $T \neq T'$, there must be a i such that $\vec{t_i} \neq \vec{t'_i}$. Without loss of generality, assume $i = 1$. The Borda score of w in the 4-voter profile $\langle V_1[\vec{t_1}], Rev(V_1[\vec{t_1}]), [V_1\vec{t'_1}], Rev(V_1[\vec{t'_1}]) \rangle$ is $2(p+k-1)$. Therefore, either $S_B(w, \langle V_1[\vec{t_1}], Rev(V_1[\vec{t'_1}]) \rangle \leq p+k-1$ or $S_B(w, \langle V_1[\vec{t'_1}], Rev(V_1[\vec{t_1}]) \rangle \leq p+k-1$. Again without loss of generality, suppose $S_B(w, \langle V_1[\vec{t_1}], Rev(V_1[\vec{t'_1}]) \rangle \leq p+k-1$. Consider the mixture $P_X[T] \otimes_2 P_X[T']$ identical to $P_X[T]$ except that it contains $rev(V_1[\vec{t'_1}])$ instead of $rev(V_1[\vec{t_1}])$. We have $S_B(x_1, P_X[T] \otimes_2 P_X[T']) \leq n'(p+k-1)$. Now, there must be a candidate $z \in X \cup Y$, $z \neq w$, such that $S_B(z, P_X[T] \otimes_2 P_X[T']) > n'(p+k-1)$: if this were not the case, then because the sum of the Borda scores of all candidates is $n'(p+k-1)$, and $S_B(z, P_X[T] \otimes_2 P_X[T']) \leq n'(p+k-1)$, we would have $S_B(z, P_X[T] \otimes_2 P_X[T']) = n'(p+k-1)$ for every z, which in turn would imply $S_B(z, \langle V_1[\vec{t_1}], Rev(V_1[\vec{t'_1}]) \rangle = p+k-1$ for every z, therefore we would have $Rev(V_1[\vec{t_1}]) = Rev(V_1[\vec{t'_1}])$, contradicting $\vec{t_1} \neq \vec{t'_1}$. Hence, there exists at least one candidate such $S_B(z, P_X[T] \otimes_2 P_X[T']) > S_B(w, P_X[T] \otimes_2 P_X[T'])$, which implies that the winner is not w. ■

PROPOSITION 5. *The communication complexity of Copeland with respect to the addition of k new candidates and full storage is in* $\Theta(n(\log(p+k)! - \log(p!))$.

PROOF. The upper bound comes from Proposition 1. For the lower bound, we reuse the same set of profiles $F(P_X)$ as in the proof of Proposition 4. We only have check that $F(P_X)$ is a fooling set. In every profile $P_X[T]$ of $F(P_X)$, all candidates are tied in the majority graph, therefore the winner is the candidate with highest tie-breaking priority. Let $w \in X \cup Y$ be this candidate.

Let $P_X[T]$, $P_X[T']$ be two profiles in $F(P_X)$. Since $T \neq T'$, there must be a i such that $\vec{t_i} \neq \vec{t'_i}$. Without loss of generality, assume $i = 1$. We consider two cases.

First case: the ranks of w in $P_X[T]$ and $P_X[T']$ are different. Without loss of generality, assume w is ranked in a better position in $P_X[T]$ than in $P_X[T']$. Consider the mixture $Q = P_X[T] \otimes_1 P_X[T']$ identical to $P_X[T]$ except that it contains $V_1[\vec{t'_1}]$ instead of $V_1[\vec{t_1}]$. For any $z \in X \cup Y$, we have $w \succ^Q_{maj} x$ if and only if $w \succ^{P_X[T]}_{maj} z$ and $z \succ^{P_X[T']}_{maj} w$. Because w is ranked in a better position in $P_X[T]$ than in $P_X[T']$, we have $\#\{z, z \succ^{P_X[T']}_{maj} w\} > \#\{z, z \succ^{P_X[T]}_{maj} w\}$, therefore we have $\#\{z, z \succ^Q_{maj} w\} > \#\{z, w \succ^Q_{maj} x\}$, which implies that $C(w, Q) < 0$; therefore, w is not the Copeland winner in Q.

Second case: the ranks of w in $P_X[T]$ and $P_X[T']$ are identical. We start by remarking that $\succ^Q_{maj}$ coincides with the majority graph constructed from the first two votes $\{V_1[\vec{t'_1}], V_1[rev(\vec{t_1})]\}$, and that a majority graph constructed from two votes is transitive, therefore $\succ^Q_{maj}$ is transitive. Since $P_X[T] \neq P_X[T']$, there must be $u \in X \cup Y$ and $v \in Y$ such that $u \succ^T_1 v$ and $v \succ^{T'}_1 u$. Therefore, in $Q = P_X[T] \otimes_1 P_X[T']$, v is two points ahead of

[1]In this proof, and more generally in all our proofs based on the fooling set technique, we could have exhibited a fooling set for which the winner in every element of the fooling set, and in any "successful" mixture of profiles, does not depend on the tie-breaking priority.

u, which implies that $v \succ_{maj}^{Q} u$, and therefore that $\succ_{maj}$ is not the empty graph. Now, since $\succ_{maj}$ is transitive and nonempty, it must contain a maximal element, whose Copeland score is strictly positive. Lastly, since the ranks of w in $P_X[T]$ and $P_X[T']$ are identical, we have $\#\{z, z \succ_{maj}^{Q} w\} = \#\{z, w \succ_{maj}^{Q} x\}$, and $C(w, Q) = 0$. Therefore, w is not the Copeland winner in Q. ■

5. STRONGER COMPILATIONS FOR SCORING RULES

In the specific case of scoring rules, we now introduce compilation functions that are stronger than full storage. Note first that it is sometimes possible to find a better compilation function than σ_F that does not need more communication. Therefore, an optimal protocol given full storage is not necessarily an optimal protocol, and there may be no protocol in $BestProtocols(r)$ of the form $\langle \sigma_F, \pi_F \rangle$. Now, there will be a protocol $\mu = \langle \sigma, \pi \rangle$ in $BestProtocols(r)$ such that the communication complexity of π is equal to the communication complexity of r given full storage (we may think, intuitively, that this is the case for π_F in Example 1, but further we show that this is not the case). The compilation complexity of such a protocol is, intuitively, the size we need to store P_X without requiring more communication than if P_X was stored entirely, and can be viewed as the counterpart of compilation complexity for a varying set of *voters*, as studied in [2, 8].

The first question that we ask (Section 5.1) is whether it is possible to reduce the compilation cost while still using the optimal communication protocol for full storage. We illustrate in particular that for K-approval, a partly anonymous compilation of scores is appropriate. Next, in Section 5.2 we consider the problem the other way round: starting from the stronger compilation consisting in storing only the score of each candidate, can we design protocols minimizing communication complexity? For the sending-only model, we will see that for a certain class of scoring rules, it is possible to design a protocol requiring the same amount of communication as with full storage.

5.1 Partly anonymous compilation of scores

Consider the case of K-approval. Technically, the question is whether π_1 and π_2 can be applied to a smaller compilation function, that is, if there exist protocols (σ_1, π_1) and (σ_2, π_2) for K-approval with $\sigma_F \rhd \sigma_1, \sigma_2$. In other words, what is the minimal information to be stored so that π_1 and π_2 are applicable? We note first that *storing, for each voter, the set of her first K candidates in X is not sufficient*: we must know which ones will be kicked out if some of the new candidates come into her first K candidates. Keeping the *ordered list* of the first K candidates of each voter, on the other hand, is enough, but is not optimal when $k \leq K$, as in this case, it is enough to store the following two pieces of information:

(a) for each voter, the ordered list of the candidates of X who may be kicked out of the first K candidates, that is, the candidates ranked in positions $K - k + 1$ to K;

(b) for each candidate in X, the number of voters who rank it in the first $K - k$ positions.

Let $P_X = \langle V_1, \ldots, V_n \rangle$. We define the compilation function $\sigma_{K,k}(P_X)$ as follows:

- if $k \leq K$: let A_i be the set of the first $K-k$ candidates in V_i, B_i be the ordered list consisting of the candidates ranked in positions $K-k+1$ to K in V_i, and for any $x \in X$, $C(x) = \#\{i | x \in A_i\}$. Then $\sigma_{K,k}(P_X) = \langle C(x), B_1, \ldots, B_n \rangle$.
- if $k > K$: $\sigma_{K,k}(P_X)$ contains, for every voter, the ordered list consisting of the candidates ranked in positions 1 to K in V_i.

Consider an example: $n = 3$, $p = 5$, $k = 2$, $K = 4$, $P_X = \langle abcde, dabec, bedac \rangle$. $\sigma_{4,2}(P) = \langle C(x), B_1, \ldots, B_n \rangle$ with $C(a) = 2$, $C(b) = 2$, $C(c) = 0$, $C(d) = C(e) = 1$, $B_1 = \langle c, d \rangle$, $B_2 = \langle b, e \rangle$, $B_3 = \langle d, a \rangle$.

The size of $\sigma_{K,k}$ is in $O(p \log n + n \log k!)$ if $k \leq K$, and $O(n \log K!)$ if $k > K$. Note that these bounds can be much smaller than the size of $\sigma_F(P_X)$, which is in $\Theta(n \log p!)$. Therefore, neither μ_1 nor μ_2 is Pareto-optimal, and

$$\begin{aligned} & CC(\text{K-approval}) \\ \leq \; & \langle \min(p \log n + n \log k!, n \log K!), \min(nk, nK \log k) \rangle \end{aligned}$$

If K is a constant, CC(K-approval) is in $O(\langle \min(n \log k!), nk \rangle)$.

5.2 Fully anonymous compilation of scores

Let P_X be an initial profile. We assume without loss of generality that the candidates of X are numbered by decreasing scores in profile P_X and that this order is lexicographically compatible with tie-breaking on X, *i.e.*, $s(P_X, x_1) \geq \cdots \geq s(P_X, x_m)$, and if $s(P_X, x_i) = \cdots = s(P_X, x_j)$, then $x_i > x_{i+1} > \ldots > x_j$. Importantly, *this ordering on candidates in X induced by the partial scores and the tie-breaking priority is common knowledge among the voters*. In the sending-only communication model, this assumption is harmless: the central authority will have sent this information to the voters, and voters receive this information for free. In the sending-receiving model, however, this amounts to sending all voters the ranking over X induced from P_X, which takes exactly $np \log p$ bits.

Moreover, in order to simplify the notations, if $Y = \{y_1, \ldots, y_k\}$ are the new candidates and P is the extension of P_X, then $s(y_1, P) \geq \cdots \geq s(y_k, P)$ and y_1 has the most favorable tie-breaking among candidates in Y. Lastly, for the sake of simplicity we assume that the tie-breaking priority favours any x_i against any y_j: it is therefore equal to the priority order $x_1 > \ldots > x_p > y_1 \ldots > y_k$. (This is not entirely without loss of generality, but the proof can be modified easily if we make whatever different assumption about the tie-breaking priority). Finally, we define $S_Y = \sum_{j=1}^{k} s(P, y_j)$.

DEFINITION 5. *Let s^* be a scoring rule.*

- *s^* is said to be non-increasing (resp. non-decreasing) if for any initial candidate $x \in X$ and for any extension P of P_X, we have $s(x, P_X) \geq s(x, P)$ (resp., $s(x, P_X) \leq s(x, P)$).*
- *s^* is said to be coherent if for any initial subset $X' \subseteq X$ of initial candidates and for any extension P of P_X on candidates $X \cup Y$, we have $\sum_{x \in X'} s(x, P_X) \leq \sum_{x \in X' \cup Y} s(x, P)$.*

The requirement to be non-increasing is the most important: for instance, the Borda rule is non-decreasing but not non-increasing, and thus the protocol is not applicable to it. Intuitively, the condition of coherence is not demanding. It merely rules out very strange rules, which would for instance be defined as 2-approval up to a number of candidates, then as plurality. Note that any *pure rules* used in the sense of [1] is coherent.

DEFINITION 6. *Let s^* be a non-increasing scoring rule.*

- *An initial candidate $x \in X$ is a potential winner if $s(x, P_X) \geq s(y_1, P)$*[2]*. Let X_{pot} the set of potential winners.*
- *An initial candidate $x \in X$ is a quasi- winner if $s(x, P) \geq s(y_1, P)$. Let X_{qw} the set of quasi-winners.*

Obviously, we get $X_{qw} \subseteq X_{pot}$ and if $X_{pot} \neq \emptyset$, then $|X_{qw}| \leq r = \max\{i : s(x_i, P_X) \geq s(y_1, P)\}$.

Lemma 1 gives some properties on quasi-winners. In the following, let s be a collection of scoring vectors corresponding to a coherent and non-increasing scoring rule.

LEMMA 1. *The following properties hold:*

(i) *If s^* is non-increasing scoring rule, then y_1 is the winner if $X_{pot} = \emptyset$ or equivalently if*

$$s(y_1, P) > s(x_1, P_X) \tag{1}$$

(ii) *If s^* is coherent, $X_{pot} \neq \emptyset$, and $r' \leq r = \max\{i : s(x_i, P_X) \geq s(y_1, P)\}$, then $x_{r'}$ is a quasi-winner if*

$$\sum_{i=1}^{r'} s(x_i, P_X) \geq S_Y + r' s(y_1, P) \tag{2}$$

PROOF. For (i). trivial by hypothesis of a non-increasing scoring rule.

For (ii). If a candidate in X is a quasi- winner, then inequality (1) does not hold. So, $r = \max\{i : s(x_i, P_X) \geq s(y_1, P)\}$ is well-defined. Let $r' \leq r$ and assume that inequality (2) holds. We have $\sum_{i=1}^{r'} s(x_i, P) \geq \sum_{i=1}^{r'} s(x_i, P_X) - S_Y$ because s is a coherent scoring rule (take $X' = \{x_1, \ldots, x_{r'}\}$ in definition 5). Since $r' \max\{s(x_i, P) : i \leq r'\} \geq \sum_{i=1}^{r'} s(x_i, P)$ and using inequality (2), we obtain the expected result. ■

LEMMA 2. *For decreasing and coherent scoring rules, a candidate x_i is the winner for P only if $i \leq S_Y + 1$ and $i \leq k\frac{\sum_{x \in X \cup Y} s(x,P)}{S_Y}$.*

PROOF. Assume that x_i is the winner. So, in particular we get $s(x_i, P) \geq s(y_1, P)$[3] and $s(x_i, P) \geq s(x_j, P) + 1$ by construction of the priority rule. Now, since s is decreasing and $j < i$, we get $s(x_j, P_X) \geq s(x_i, P_X) \geq s(x_i, P)$. Hence, $s(x_j, P_X) \geq s(x_j, P) + 1$. On the other hand, since s^* is coherent $S_Y \geq \sum_{j=1}^{i-1} (s(x_j, P_X) - s(x_j, P)) \geq i - 1$.

Since s^* is coherent, $\sum_{j=1}^{r} s(x_j, P_X) \leq \sum_{j=1}^{r} s(x_j, P) + S_Y \leq \sum_{x \in X \cup Y} s(x, P)$. On the other hand, $r \times \frac{S_Y}{k} \leq r \times s(y_1, P) \leq \sum_{j=1}^{r} s(x_j, P_X)$. Thus, $r \leq k\frac{\sum_{x \in X \cup Y} s(x,P)}{S_Y}$. Now, since s^* is non-increasing we get $i \leq r$. ■

We now give a protocol which works for *non-increasing* and *coherent* scoring rules.

Consider a non-increasing and coherent scoring rule s where $\sum_{x \in X} s(x, P) = f(n, m)$ for every profile P on candidates X and such that α is the number of distinct values in the scoring vector. For instance, for K-approval we have $f(n, m) = Kn$ and $\alpha = 2$. We now define the following protocol:

[2]We recall that we assume that the tie-breaking gives the priority to x; if y_1 had the priority over x then the inequality would have to be strict.

[3]The inequality is strict if y_1 has a higher priority rule.

(1) Ask for each voter the position (among the α distinct intervals) of each candidate from Y.

(2) If $s(y_1, P) > s(x_1, P_X)$ return y_1 and stop.

(3) Let $i_0 = \min\{S_Y + 1, k\frac{\sum_{x \in X \cup Y} s(x,P)}{S_Y}, p\}$. Ask each voter the position (among the α distinct intervals) of each candidate of X which had an index at most i_0 in P_X.

PROPOSITION 6. *The communication complexity of any non-increasing and coherent scoring rule with respect to the addition of k candidates is in $O(n\sqrt{kf(n,m)} \log \alpha)$ in the sending-only model where $m = p + k$.*

PROOF. The communication complexity given in Step (1) is upper bounded by $nk \log \alpha$. The communication complexity given in Step (3) is upper bounded by $n \times i_0 \log \alpha$. Now, by Lemma 2, we have $i_0 \leq \min(S_Y + 1, k\frac{\sum_{x \in X \cup Y} s(x,P)}{S_Y}) \leq \min(S_Y, k\frac{\sum_{x \in X \cup Y} s(x,P)}{S_Y}) + 1$. Let $\xi = k \sum_{x \in X \cup Y} s(x, P)$. If $S_Y \leq \frac{\xi}{S_Y}$ then $(S_Y)^2 \leq \xi$, $S_Y \leq \sqrt{\xi}$ and finally, $\min(S_Y, \frac{\xi}{S_Y}) \leq \sqrt{\xi}$. If $S_Y > \frac{\xi}{S_Y}$ then $(S_Y)^2 > \xi$, $S_Y > \sqrt{\xi}$, $S_Y\sqrt{\xi} > \xi$, $\frac{\xi}{S_Y} < \sqrt{\xi}$, and finally $\min(S_Y, \frac{\xi}{S_Y}) \leq \sqrt{\xi}$. Therefore, $i_0 \leq (k \times \sum_{x \in X \cup Y} s(x, P))^{1/2} + 1 = \sqrt{kf(n,m)} + 1$, from which we conclude that the communication complexity of (3) is in $O(n\sqrt{kf(n,m)} \log \alpha)$. ■

For K-approval, we can improve this bound to $O(nK \log(k + 1))$ by considering the following protocol. For any candidate $z \in X \cup Y$ and profile Q, $s_K(z, Q)$ denotes the K-approval score of z for Q.

(1) Ask each voter whether she approves at least one new candidate in Y; if it is the case, use protocol π_1 or π_2 (which one is best) of Section 4.1 to identify the candidates from Y that she approves. For $i = 1, \ldots, n$, let r_i be the number of candidates of Y approved by voter i.

(2) If $s_K(y_1, P) > s_K(x_1, P_X)$ then return y_1 and stop.

(3) Let $i_0 = \min\{S_Y + 1, \lceil K\frac{kn}{S_Y}\rceil, p\}$.
For each voter i such that $r_i > 0$: if, in her initial vote (in P_X), i ranked in the positions $K - r_i + 1, ..., K$ at least one candidate x_i such that $i \leq i_0$, then use protocol π_1 or π_2 (which one is best) of Section 4.1 to identify the candidates x_i that she ranked in positions $K - r_i + 1, ..., K$ in her initial vote and such that $i \leq i_0$.

EXAMPLE 2. *Let $K = 2$, $p = 4$, $n = 5$ and $k = 2$. The anonymous compilation of the initial profile P_X consists of the following scores:*

$$x_1 : 2;\ x_2 : 2;\ x_3 : 2;\ x_4 : 2;\ x_5 : 2$$

Step 1 of the protocol reveals that voter 1 approves y_1, y_2; that voter 2 approves y_1; and that voters 3, 4 and 5 do not approve any of the two new candidates. The scores of y_1, y_2 in the new profile P are

$$y_1 : 2;\ y_2 : 1$$

while the number of new candidates approved by the voters are

$$r_1 = 2;\ r_2 = 1;\ r_3 = r_4 = r_5 = 0$$

At step 2, since the score of y_1 in P is not strictly larger than the score of all the x_1 in P, we continue to Step 3.
At step 3, we first compute $S_Y = 3$ and

$$i_0 = \min\{S_Y + 1, \lceil K\frac{kn}{S_Y}\rceil, p\} = \min(4, 7, 5) = 4$$

which expresses that x_5 has no chance to be the winner (because in order x_5 to win, x_1, x_2, x_3, x_4 should all lose at least one point, which is not possible since y_1 and y_2 gather only 3 points).
Now, we ask voter 1 who among x_1, x_2, x_3, x_4 was ranked in position 1 or 2 in her initial vote; she answers $\{x_1, x_2\}$.
Next, we ask voter 2 who among x_1, x_2, x_3, x_4 was ranked in position 1 or 2 in her initial vote; she answers $\{x_4\}$.
Since $r_3 = r_4 = r_5 = 0$, we don't have anything to ask to voters 3, 4 and 5. The scores of x_1, x_2, x_3, x_4 in P are

$$x_1 : 1;\ x_2 : 1;\ x_3 : 2;\ x_4 : 1$$

Due to the tie-breaking priority, the winner is x_3.

PROPOSITION 5.1. *The communication complexity of K-approval with respect to the addition of k candidates is in $O(Kn\log(k + 1))$ in the sending-only model.*

PROOF. The communication complexity given in Step (3) is upper bounded by $S_Y \log(i_0 + 1)$. Actually, if a voter does not approve a candidate x_i with $i \leq i_0$ he returns 0 and else he returns its index. Moreover, since on the one hand a candidate approved in P_X and not approved in P for voter i is ranked at position between $K + 1$ and $K + r_i$ and on the other hand, we also get $\sum_{i=1}^{n} r_i = S_Y$, then we ask exactly S_Y questions. Using Lemma 2, we get $k\frac{\sum_{x\in X\cup Y} s_K(x,P)}{S_Y} = k\frac{Kn}{S_Y}$ for K-approval. Thus, we obtain

$$\begin{aligned} &S_Y \log(i_0 + 1) \\ &\leq S_Y \log(\tfrac{kKn}{S_Y} + 1) \\ &\leq S_Y \log(\tfrac{(k+1)Kn}{S_Y}) \end{aligned}$$

because $S_Y \leq Kn$. The mapping $f(x) = x\log(\frac{(k+1)Kn}{x})$ with $x \in [1, Kn]$ reaches its maximum value for $x = \min(\frac{(k+1)Kn}{e}, Kn) = O(Kn)$. Hence $S_Y \log(i_0 + 1) = O(Kn\log(k + 1))$. ■

The previous protocol can be improved further, by noticing that in the cases where $S_Y > \frac{nK}{2}$ (*i.e.*, the new candidates gather more than half of the points), it may be less consuming to ask each voter *whom among the initial candidates she still approves* instead of *whom among the initial candidates she no longer approves*.

6. CONCLUSION

This paper advances the (sparse) state of the art in communication and voting, investigating for the first time (to the best of our knowledge) the definition of elicitation protocols for a dynamic set of candidates and a compilation function used to store the initial profile. We emphasized the trade-off between storage and elicitation which has to be made in these situations, as storing more information generally allows to come up with protocols which alleviate the elicitation burden. This model is relevant in many contexts which may differ significantly in terms of the constraints that need to be considered. In a multiagent system involving distributed decision-making about some joint plan, the number of candidates may be huge and there may be concerns about the storage capacity of each agent. On the other hand, in an electronic voting platform, even for a moderate number of candidates, the stress should bear on minimizing the communication burden on voters, so as to be an incentive for participation. Furthermore, there may be exogenous constraints: for instance, privacy issues may require that the initial profile must be stored anonymously.

The first contribution of this paper was to set up this model. Our initial results gave insights on these different aspects, illustrating how the trade-off may be dealt with. For the Borda and Copeland rules, we note that our $\Theta(n(\log(p + k)! - \log(p!))$ can be seen as a generalization of the bounds proven in [4] (by setting $p = 0$). We also noted that the basic constructions used in the fooling sets are reminiscent of those of [4]. When k is small, the gain in communication due to storage can be seen clearly: for instance, if $k = 1$, then we get a communication complexity in $\Theta(n \log p)$ instead of $\Theta(np \log(p)$. We then identified the communication complexity of K-approval (which was not studied in [4]), and finally, we discussed stronger compilation functions. Surprisingly, we showed that for a family of scoring rules including K-approval, even under the much stronger compilation technique consisting of storing anonymously the score of each candidate, we can design a protocol which will not be more demanding *for the agents* than what can be achieved with full storage of the profile.

7. ACKNOWLEDGEMENTS

This work was partially supported by the ANR project ComSoc (ANR-09-BLAN-0305).

8. REFERENCES

[1] Nadja Betzler and Britta Dorn. Towards a dichotomy for the possible winner problem in elections based on scoring rules. *Journal of Comp. and System Sciences*, 76(8):812–836, 2010.
[2] Yann Chevaleyre, Jérôme Lang, Nicolas Maudet, and Guillaume Ravilly-Abadie. Compiling the votes of a subelectorate. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 97–102, 2009.
[3] Yann Chevaleyre, Jérôme Lang, Nicolas Maudet, and Jérôme Monnot. Possible winners when new candidates are added: the case of scoring rules. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI-2010)*, pages 762–767, 2010.
[4] Vincent Conitzer and Tuomas Sandholm. Communication complexity of common voting rules. In *Proceedings of the 6th ACM Conference on Electronic Commerce (EC-05)*, pages 78–87, 2005.
[5] Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.
[6] Ariel Procaccia. A note on the query complexity of the condorcet winner problem. *Information Processing Letters*, 108(6):390–393, 2008.
[7] Ilya Segal. The communication requirements of social choice rules and supporting budget sets. *Journal of Economic Theory*, 136(1):341-378, 2007
[8] Lirong Xia and Vincent Conitzer. Compilation complexity of common voting rules. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, pages 915–920, 2010.
[9] Lirong Xia, Jérôme Lang, and Jérôme Monnot. Possible winners when new alternatives join: New results coming up! In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2011)*, pages 829–836, 2011.