Robust Winners and Winner Determination Policies under Candidate Uncertainty

Craig Boutilier
Department of Computer Science
University of Toronto
cebly@cs.toronto.edu

Jérôme Lang
LAMSADE
Université Paris-Dauphine
lang@lamsade.dauphine.fr

Joel Oren
Department of Computer Science
University of Toronto
oren@cs.toronto.edu

Héctor Palacios
Departament de Tecnologies
Universitat Pompeu Fabra
hector.palacios@upf.edu

Abstract

We consider voting situations in which some candidates may turn out to be unavailable. When determining availability is costly (e.g., in terms of money, time, or computation), voting prior to determining candidate availability and testing the winner's availability *after* the vote may be beneficial. However, since few voting rules are robust to candidate deletion, winner determination requires a number of such availability tests. We outline a model for analyzing such problems, defining *robust winners* relative to potential candidate unavailability. We assess the complexity of computing robust winners for several voting rules. Assuming a distribution over availability, and costs for availability tests/queries, we describe algorithms for computing *optimal query policies*, which minimize the expected cost of determining true winners.

Introduction

There are many social choice situations in which members of a group must specify their preferences over a set of alternatives or *candidates* without knowing whether any specific candidate is in fact viable or *available* for selection. Selecting a winner requires knowing which candidates are available, but determining availability may in fact be costly. In such a setting, voting over the set of *potential* candidates *prior to determining availability* often makes sense. For example, a group of dinner companions may attempt to (partially) determine their aggregate preferences prior to calling restaurants to check reservation availability. A committee deciding among various public projects may vote prior to knowing the feasibility or precise cost of any project, since assessment (e.g., engineering estimates, environmental studies) is itself costly. In AI planning, a group may vote on a *goal* to pursue prior to knowing its feasibility, since planning for a goal can be computationally expensive. In each case, as estimate of the potential winners can narrow the set of required *availability tests*, and reduce the financial, time or computational cost of determining the true winner.

Unfortunately, few voting rules are robust to candidate deletion, so declaring a winner generally requires testing availability. We describe a model for addressing such problems, identify a number of key concepts and computational questions, and make some first steps toward solving them.

We assume a set of *potential candidates* X, voter preferences over X, and a voting rule r. Since candidate availability is uncertain, we assume a distribution over potential *available sets* $A \subseteq X$, and assume that each candidate x can be *queried*, for a cost, to determine its availability. Our aim is to determine the winner of the election over the *actual* available set A in a way that minimizes expected query cost (e.g., number of phone calls, planning problems to be solved, etc.). We focus on *query policies* that propose a (conditional) sequence, or tree, of queries that extracts enough information about A to declare a winner. Critically, we *need not know* A *precisely* to determine the winner of an election. Responses to certain queries may render others irrelevant; e.g., if x is a *majority* winner in a plurality election, the status of other candidates is irrelevant once we know x is available. Given an *information set* about the availability of some candidates, we say x is a *robust winner* if this information suffices to determine that x wins regardless of the availability of the remaining candidates.

The problem of determining a robust winner has tight connections to that of control by candidate addition, thus can be computationally difficult for some rules, and easy for others. However, our primary concern is minimizing query costs. Query policies need only pose enough tests to determine a robust winner. We formulate this problem as one of constructing a minimal cost *decision tree* whose features denote availability of specific candidates, and whose goal is to classify available sets A according to their winners. We describe a (relatively) inexpensive dynamic programming algorithm for optimal query policies, and also investigate more tractable decision tree induction methods based on *information gain*. Together with several observations about query complexity, our empirical results demonstrate the value and feasibility of well-designed query policies.

Voting with Uncertain Availability

We first outline our model for robust winners with uncertain candidate availability and briefly discuss related work.

Background: We assume a standard voting model, with a set of n voters N and a set of m *potential candidates* X, with each voter $i \in N$ having a complete, strict preference ordering or *vote* v_i over X, with *vote profile* $\mathbf{v}$ denoting the vector of all votes. A *voting rule* r maps every profile to a

(unique) winning candidate (with ties broken in some fashion). We consider specific rules below, but our framework is completely general. We assume familiarity with the plurality, Borda and Copeland rules. Given a profile $\mathbf{v}$, let $M(\mathbf{v})$ be its majority graph, and $M(\mathbf{v})^*$ the transitive closure of $M(\mathbf{v})$. For ease of exposition, we assume n is odd so that $M(\mathbf{v})$ is a tournament. $N(x, y, \mathbf{v})$ denotes the number of votes in $\mathbf{v}$ that rank x above y. The *top cycle* $TC(\mathbf{v})$ of $\mathbf{v}$ is the set of all candidates x s.t. for all $y \neq x$, $(x, y) \in M(\mathbf{v})^*$. The *uncovered set* $UC(\mathbf{v})$ of $\mathbf{v}$ is the set of all candidates x from which there is a path of length at most 2 in $M(\mathbf{v})$ to every $y \neq x$. TC and UC (plus a tie-breaking mechanism) can also be used as voting rules. We assume r can be applied to profiles over arbitrary subsets of X. Let $\mathbf{v}(A)$ denote the restriction of $\mathbf{v}$ to candidates A, obtained by deleting elements of $X \setminus A$ from each vote, and $r(\mathbf{v}(A))$ the winner w.r.t. this restricted profile.

The Model: Suppose certain candidates in X may be unavailable. There are two ways to address this. First, we might check the availability of all candidates in X, and elicit votes over the set of available candidates $A \subseteq X$. This has the advantage of minimizing vote elicitation costs: voters need not rank or compare unavailable candidates. However, if candidate availability testing is costly, as discussed above, this may require far more availability tests than are actually needed given voter preferences.

Instead we could first elicit voter rankings over the entire set X, then use this information to focus on "relevant" availability tests. This can reduce the cost of availability tests, and is appropriate when they are costly relative to preference elicitation, as in our examples above. Determining suitable tests is, nonetheless, far from straightforward.

One obvious approach is to use the voting rule r to rank candidates, test availability in the order of this aggregate ranking, and select the first available candidate as the winner. However, this works only if the choice function implemented by r is *rationalizable*, which is rarely the case. Consider a profile $\mathbf{v}$ with 4 votes abc, 3 votes bca and 2 votes cab. Ranking candidates by plurality score gives ranking abc. The policy above, once learning a is available, selects a as the winner without further tests; but if we learned b is unavailable and c is available, the true plurality winner for $\mathbf{v}(\{a, c\})$ is c. This paradox arises since, under mild conditions, no non-dictatorial voting rule is robust to the deletion of non-winning alternatives (Dutta, et al. 2001). Thus, choosing a winner usually requires confirming the availability of specific sets of candidates.[1] Furthermore, minimizing the costs of such tests is non-trivial.

To model candidate availability we partition X into a (possibly empty) *known set* $Y \subseteq X$ of candidates that are sure to be available, and an *unknown set* $U = X \setminus Y$ for which availability is uncertain. (Candidates unavailable *a priori* are removed from X.) Let $\mathcal{A}$ denote the family of subsets $Y \subseteq A \subseteq X$, where $A \in \mathcal{A}$ is a possible *available set*. The general *unavailable candidate model* (Lu and Boutilier 2010) requires a distribution P over 2^U, where $P(S)$ denotes the probability that $S \subseteq U$ is the true available set of candidates in U. We assume for simplicity that the availability of each candidate $x \in U$ is given by probability p_x, and is independent of that of other candidates. This induces the obvious distribution over $\mathcal{A}$.

For any $x \in U$, one can *query* x using an *availability test* (e.g., call for a restaurant reservation, compute a plan for a goal x), which incurs a cost c_x. Informally, a *query policy* consists of a tree whose interior nodes are labeled by queries, edges by availability, and leaves by winners. We desire policies that, given a profile $\mathbf{v}$, determine the winner w.r.t. the *true* available set A with minimum expected query cost. After any sequence of queries and responses, we have refined information about A: an *information set* is an ordered pair $Q = \langle Q^+, Q^- \rangle$, where $Q^+, Q^- \subseteq U$, and $Q^+ \cap Q^- = \emptyset$; intuitively, Q^+ (resp. Q^-) is the set of candidates for which positive (resp. negative) availability has been verified.

Clearly, winners can often be determined without full knowledge of candidate availability. In our example above, knowing that a and b are available suffices to declare a the winner: availability of c is irrelevant (knowing a is available and c is unavailable is also sufficient to select a).

Definition 1 *Let r be a voting rule, $\mathbf{v}$ a profile over candidate set X, and $Y \subseteq X$ a set of candidates known to be available. $x \in Y$ is a* robust winner *w.r.t. $\langle X, Y, \mathbf{v}, r \rangle$ if, for any A s.t. $Y \subseteq A \subseteq X$, $r(\mathbf{v}(A)) = x$.*

Intuitively, a winner is robust if it not only wins in the original profile $\mathbf{v}$, but continues to win no matter which candidates in $X \setminus Y$ are deleted. The existence of a robust winner relative to an information set is necessary and sufficient to stop any querying process. Specifically, we say that information set Q is *r-sufficient* if there is a robust winner x w.r.t. $\langle X \setminus Q^-, Y \cup Q^+, \mathbf{v}, r \rangle$. For any r-sufficient information set Q, let $w(Q)$ denote this (unique) robust winner.

Related Work: Lu and Boutilier (2010) study a setting where the set of available candidates is unknown at the time of voting, and assume a distribution over available sets A as we do. Unlike our model, they assume a's availability cannot be tested without "offering it the win," hence focus on optimal *ranking policies*, as discussed above (see also (Baldiga and Green. 2013)). Wojtas and Faliszewski (2012) also consider candidates with uncertain availability in a counting version of *control by adding candidates* (see below), a problem closely related to ours. They assume a known available set Y, a distribution over subsets of $X \setminus Y$, votes over X, and consider the complexity of computing the probability that some $x \in X$ wins.

Chevaleyre *et al.* (2012; 2011), study the possible winner problem where the candidate set is not known at vote time, but take the opposite perspective to ours. An initial *lower bound* on the candidate set is known, and new candidates may join after initial preferences have been elicited; *preference elicitation* (as opposed to availability testing) protocols are developed to identify the winner. Rastegari *et al.* (2013) also develop optimal knowledge-gathering policies in a social choice context, though in a different stable matching setting. Our notion of robust winners differs from that in (Procaccia, et al. 2007; Xia 2012; Shiryaev, et al. 2013), where

[1] At a minimum, one might require that the winner itself be available, but we consider exceptions to this below.

a winner is robust if it remains a winner after some changes in the votes. Finally, our model is also strongly related to *control via adding candidates* (as we elaborate below).

Computing Robust Winners

We first consider the problem of identifying or verifying robust winners given some available set. If x is a robust winner w.r.t. $\langle X, Y, \mathbf{v}, r\rangle$, it must meet these obvious necessary conditions: $x \in Y$, $x = r(Y)$ (setting $A = Y$) and $x = r(\mathbf{v})$ (setting $A = X$). The following key result makes a connection to destructive control via adding candidates (Bartholdi, et al. 1992), in which an initial candidate set can be augmented by "spoiler" candidates, and a chair, knowing the votes, attempts to find certain spoilers whose addition makes her preferred candidate win.

Proposition 1 *Candidate $x \in Y$ is a robust winner w.r.t. $\langle X, Y, \mathbf{v}, r\rangle$ if there is no destructive control against x by adding candidates, where the spoiler set is $U = X \setminus Y$.*

The proof is immediate: the chair has destructive control against x via adding candidates iff there is a spoiler set $S \subseteq U$ such that $r(\mathbf{v}(Y \cup S)) \neq x$ (i.e., x is not a robust winner). Since the robust winner problem is equivalent to the complement of the problem of DESTRUCTIVE CONTROL BY ADDING CANDIDATES, results in election control directly determine the complexity of checking the existence of robust winners for several voting rules: it is coNP-complete for plurality (Bartholdi, et al. 1992; Hemaspaandra, et al. 2007), Bucklin (Erdélyi et al. 2011) and ranked pairs with immediate tie-breaking (Parkes and Xia 2012); and it is polynomial for Copeland (Faliszewski, et al. 2008) and maximin (Faliszewski, et al. 2011). The latter two results come with efficient algorithms for the robust winner problem. These results suggest that the problem tends to be simpler for voting rules based on the majority graph, since the majority preference between two candidates x, y does not depend on the availability of others. We provide a simple characterization of robust winners for top cycle and the uncovered set (recall, we assume n odd; and for simplicity we assume favorable tie-breaking):

Proposition 2 *Let r be the top cycle rule. x is a robust winner w.r.t. $\langle X, Y, \mathbf{v}, r\rangle$ iff, for all $y \in X$, there is a path from x to y in $M(\mathbf{v})$ that goes only through candidates in Y.*

Proposition 3 *Let r be the uncovered set rule. x is a robust winner w.r.t. $\langle X, Y, \mathbf{v}, r\rangle$ iff, for all $y \in X$, either $x \to y$ is in $M(\mathbf{v})$ or there is a $z \in Y$ such that $x \to z$ and $z \to y$ are both in $M(\mathbf{v})$.*

Surprisingly, the prominent Borda rule lacks similar results w.r.t. control. However, we can show that:

Proposition 4 *$x \in Y$ is a robust Borda winner for $\mathbf{v}$ (assuming unfavorable tie-breaking) iff for all $z \in X \setminus \{x\}$:*

$$B(x, \mathbf{v}(A(x,z))) > B(z, \mathbf{v}(A(x,z)))$$

where $B(x, \cdot)$ is the Borda score of x, and

$$A(x,z) = Y \cup \{z\} \cup \{t \in X \setminus (Y \cup \{z\}) \mid N(z,t,\mathbf{v}) > N(x,t,\mathbf{v})\}.$$

Here $B(x, \mathbf{v}(A(x,z)))$ is the Borda score of x when assuming that all "unsure" candidates whose pairwise score vs. z is larger (resp., no larger) than that vs. x are available (resp., unavailable). The key point in the proof is that, for all $z \neq x$, the maximum value of $B(z, \mathbf{v}(A)) - B(x, \mathbf{v}(A))$ over available sets A containing z is obtained by $A(x,z)$.

Corollary 1 *Checking if x is a robust winner for top cycle, uncovered set and Borda can be done in polynomial time.*

Another interesting notion is that of an *irrelevant candidate*, which can be exploited in computing both robustness and optimal query policies.

Definition 2 *Let $\mathbf{v}$ be a profile over X, $x \in X$, $Y \subseteq X \setminus \{x\}$ be the known available candidates, and r a voting rule. Candidate x is* irrelevant *w.r.t. $\langle \mathbf{v}, Y, r\rangle$ if for any $A \subseteq X \setminus (Y \cup \{x\})$, we have $r(\mathbf{v}(Y \cup A \cup \{x\})) = r(\mathbf{v}(Y \cup A))$.*

If x is irrelevant for Y, we need not consider the availability of x when testing the robustness of any candidate in $X \setminus (Y \cup \{x\})$ w.r.t. Y (or any superset of Y). Similarly, in any policy for determining winners, an availability test for an irrelevant x is useless once Y (or any superset) is known to be available, a fact we exploit below. The following simple characterization of irrelevant candidates for a rich class of voting rules states that once we know that at least one candidate in the top cycle is available, removing anyone outside the top cycle cannot impact the choice of winner.

Proposition 5 *Let r be s.t. for any profile $\mathbf{v}$, if $x \notin TC(\mathbf{v})$ then $r(\mathbf{v}(X \setminus \{x\})) = r(\mathbf{v})$. For any $\mathbf{v}$, any $Y \subseteq X$ s.t. $Y \cap TC(\mathbf{v}) \neq \emptyset$, and any $x \in X \setminus TC(\mathbf{v})$, x is irrelevant w.r.t. $\langle \mathbf{v}, Y, r\rangle$.*

Prop. 5 applies, in particular, to the top cycle, Copeland, Slater and Banks rules.[2]

Minimizing Expected Query Cost

We now address computing optimal query policies that allow one to declare a (robust) winner, formulating the problem as one of *cost-sensitive decision tree construction*.

Optimal Query Policies

A *query policy* is a binary decision tree T in which each non-leaf node n is labeled by a query $q(n) \in U$, the two outgoing edges from non-leaf node n are labeled by responses yes (or "available") and no (or "unavailable"), and each leaf l is labeled by an element $w(l)$ of X (the proposed *winner* given the query-response path to l). Let $yes(n)$ and $no(n)$ denote the yes/no successors of node n in T. Any path from the root of T to a leaf l induces the obvious information set $Q(l) = \langle Q^+(l), Q^-(l)\rangle$. A policy/tree T is *r-sufficient* w.r.t. $\mathbf{v}$ if: (a) the information set $Q(l)$ for each leaf l is r-sufficient; and

[2] For Slater and Banks, this is easy to check. For Copeland, we give a brief proof sketch. Denote by $C(x, \mathbf{v})$ the Copeland score of x in $\mathbf{v}$. Let $q = |X \setminus TC(v)|$ and $x \notin TC(v)$. For every $y \in TC(\mathbf{v})$: $C(y, \mathbf{v}) \geq q$ and $C(y, \mathbf{v}(X \setminus \{x\})) \geq q - 1$. For $z \notin TC(\mathbf{v})$: $C(z, \mathbf{v}) \leq q - 1$ and $C(z, \mathbf{v}(X \setminus \{x\})) \leq q - 2$: z is not a Copeland winner in $v(X \setminus \{x\})$. For $y, y' \in TC(\mathbf{v})$: $C(y, \mathbf{v}(X \setminus \{x\})) \geq C(y', \mathbf{v}(X \setminus \{x\}))$ iff $C(y, \mathbf{v}) \geq C(y', \mathbf{v})$: Copeland winners in $\mathbf{v}$ and $\mathbf{v}(X \setminus \{x\})$ coincide.

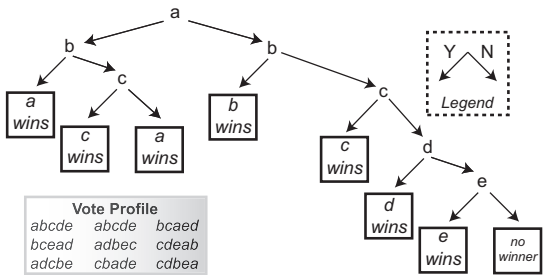

Figure 1: A vote profile and plurality-sufficient query policy.

(b) each leaf l is labeled with the robust winner $w(Q(l))$. For any $A \in \mathcal{A}$ and tree T, let $l(A)$ denote the (unique) leaf that will be reached when responses are dictated by A, and $\pi(A)$ the corresponding path. Fig. 1 shows a vote profile and an r-sufficient tree for plurality voting.

The *query cost* of a (complete) path π in T is $c(\pi) = \sum_{x \in \pi} c(x)$, i.e., the sum of the costs of the queries on π. The *expected query cost* $c(T)$ of policy T is simply $E_{A \sim P} c(\pi(A))$. This can be computed in bottom up fashion as follows, with $c(T)$ being the cost of the root of T:

$$\begin{aligned} c(l) =& 0 \text{ for any leaf node } l; \\ c(n) =& c_{q(n)} + p_{q(n)} c(yes(n)) + (1 - p_{q(n)}) c(no(n)) \\ & \text{for non-leaf node } n. \end{aligned}$$

If all candidates are available with probability $p = 0.9$, the tree in Fig. 1 has an expected query cost of 2.10. Our aim is to find an optimal policy T (i.e., an r-sufficient tree with minimal cost):

$$\arg\min\{c(T) : T \text{ is } r\text{-sufficient for } \mathbf{v}\}.$$

Computing a minimal cost policy can be cast as a standard decision tree construction problem (Quinlan 1993), taking as input an initial set of training examples

$$\{\langle A, w(A)\rangle : A \in \mathcal{A}\},$$

where A is any possible available set—we can view it as a $|U|$-vector of binary features indicating availability of unknown candidates—and $w(A)$ its winner. This set has exponential size in $|U|$, requiring winner computation for exponentially many "elections" (of various sizes). Even if winner determination for a fixed candidate set is easy (i.e., polynomial time) for the voting rule in question, constructing this training set will be difficult (indeed, NP-hard in general[3]).

Cost-optimal decision trees can be computed using dynamic programming (DP) (Garey 1972), but for general binary classification, is NP-complete, even with uniform probabilities and costs (Hyafil and Rivest 1976). However, given the importance of minimizing query costs, even intense computation will often be worthwhile.

Standard DP for decision trees uses *sets of training examples* $E \subseteq \mathcal{E} = 2^{\mathcal{A}}$. A set E is *pure* if all examples in E are labeled with the same winner. A *split* of E on candidate/query q partitions E into those examples $E_q^+ \subseteq E$ where q is available, and $E_q^- \subseteq E$ where q is not. The *optimal cost function* c^* for any E is the cost of the optimal policy given that the available set A lies within E:

$$c^*(E) = \begin{cases} 0 & \text{if } E \text{ is pure} \\ \min_{q \in U} \; p(q^+|E) c^*(E_q^+) & \\ \qquad + p(q^-|E) c^*(E_q^-) + c_q & \text{if } E \text{ is not pure} \end{cases}$$

Since $c^*(E)$ depends only on the costs for subsets of E, DP can be used, computing costs for smaller sets first.

While naïve DP is doubly exponential in $|U|$ (since $|\mathcal{E}| = 2^{2^{|U|}}$), our problem structure restricts the possible subsets of training examples. Since the set $E_0 = \mathcal{A}$ at the root consists of all subsets A, every response path of length k gives a training set E equal to E_0 restricted to k responses. Thus the only realizable sets E have size $2^{|U|-k}$, for some $0 \leq k \leq |U|$, of this form. Hence, the number of "reachable" example sets is (only) exponential: $\sum_{k=0}^{|U|} 2^k \binom{|U|}{k} = 3^{|U|}$. Thus the optimal query policy can be computed in $O(3^{|U|})$ time for any profile and any voting rule.

The DP approach exploits no structure in the profile, nor any special properties of the voting rule. Refinements for specific voting rules could greatly prune the subsets A that need to be considered. For example, for rules such as Copeland, Prop. 5 allows us to "collapse" potentially large numbers of candidate subsets—those that vary only in the availability of "irrelevant candidates"—and treat them as a single training example with a unique winner. Similarly, rules that satisfy the "majority winner property" (i.e., if a candidate is first in a majority of votes, it must win) admit significant pruning: in any subset Y with a majority winner x, x's availability renders all others irrelevant. Such pruning reduces both the set $\mathcal{A}$ of training examples, and the number of subsets E of $\mathcal{A}$ that must be considered.

Myopic Query Tree Construction

Because of the complexity of optimal decision tree construction, *myopic* approaches are widely used. C4.5 is among the most popular, and is based on *information gain* (Quinlan 1993). Extensions to cost-sensitive classification also exist (Greiner, et al. 1992; Turney 1995), and such schemes can be adapted to our setting easily.

For any set of examples $E \subseteq \mathcal{E}$, define $w(E)$ to be the set of winners $w(A)$ that occur in some example $A \in E$. Let $p_E(x) = \sum\{P(A) : A \in E, w(A) = x\}$ be the probability that x wins in training set E. The *entropy* of E is:

$$H(E) = \sum_{x \in w(E)} -p_E(x) \log p_E(x).$$

The *conditional entropy* of E given q is:

$$H(E|q) = p(q^+|E) H(E_q^+) + p(q^-|E) H(E_q^+).$$

The *information gain* associated with query q is:

$$IG(E|q) = H(E) - H(E|q).$$

Myopic decision tree construction begins at with a single root node associated with initial training set $E_0 = \mathcal{A}$. At each iteration, any "unprocessed" node n is evaluated and becomes processed: if n is pure, it becomes a leaf in the (final) tree. Otherwise, the gain $IG(E(n)|q)$ of each (nonredundant) query q is evaluated, and the query with greatest

[3] We thank a reviewer of a working paper for this observation.

gain is applied to n, creating two new children of n, with the appropriate edge labels, and each associated with the positive (resp., negative) examples from $E(n)$. When no nodes remain unprocessed, all leaves in the tree are pure.

Processing a node n is linear in the number of training examples at n (at most $|\mathcal{A}|$) and the number of splits being evaluated (at most $|U|$). Hence the complexity of myopic induction is linear in (a possibly pruned) $\mathcal{A}$ and the size of the resulting tree. Since $\mathcal{A}$ has size $2^{|U|}$ in the worst case (if unpruned), complexity is $O(2^{|U|})$ for trees of bounded size, i.e., significantly more efficient than DP. The myopic method is not guaranteed to produce a policy with minimal expected query cost. However, it works well in practice (see below); and it is guaranteed to provide a *correct* policy that determines the true winner.

Various forms of policy approximation can be used if we are willing to admit the possibility of error in declaring a winner. One approximation allows terminating the querying process at *impure leaves*, requiring only that we be "sure enough" about its identity to allow winner prediction despite residual uncertainty. This is analogous to cost-sensitive classification (Greiner, et al. 1992; Turney 1995), where both tests and *prediction errors* have costs. In our model, we have two types of misclassification errors: (a) if we choose a winner who turns out to be unavailable; (b) if we choose a winner that is available, but is not the true winner given the actual (unknown) available set. In general, we expect the former to be much worse than the latter. This can be implemented in both DP and the myopic algorithm. In the latter, we simply stop splitting leaves when one winner has sufficiently high probability.

Another approximation uses of *sampled availability sets*, with examples A drawn from the distribution P over $\mathcal{A}$, thereby reducing the number of training examples to make myopic tree construction fully tractable. Sample complexity results then become vital (Greiner, et al. 1992). We leave these approximations to future research.

Query Complexity

Apart from optimizing query policies, the theoretical question of both worst-case and average-case query complexity is of interest. Here we sketch some partial results that suggest the types of questions one might ask in our model.

Worst-case results take the form: given a voting rule r and availability distribution P, what is the greatest (over vote profiles $\mathbf{v}$) expected (over availabilities) query cost of the optimal query policy? If availability is highly likely, we can construct profiles where determining the winner requires almost m queries in expectation, for both plurality and Borda. For plurality, consider candidates $X = \{c_1, \ldots, c_{2m}\} \cup \{x, y\}$, and known available set $Y = \{x, y\}$. Define a profile over $2m$ voters where x and y are each ranked second in exactly half of the rankings: $c_i \succ x \succ \ldots$ for voters $i = 1, \ldots, m$, and $c_i \succ y \succ \ldots$ for $i = m+1, \ldots, 2m$ (other candidates are ordered arbitrarily). Let $p = 0.5$. The plurality score of x is the number of candidates in $\{c_i\}_{i=1}^{m}$ that are unavailable (similarly for y). As m increases, one of x or y wins with high probability. However, one can show (due to concentration of the binomial distribution) that the difference in their plurality scores becomes sub-linear, with high probability, as m grows. Hence, $\Omega(m)$ queries are needed to determine the winner. Similar constructions work for Borda and Copeland. As a result, we have:

Proposition 6 *For plurality, Borda and Copeland, worst-case (over profiles) expected query complexity for determining a robust winner is* $\Omega(m)$.

One can also analyze expected optimal query cost for vote profiles drawn from particular distributions (e.g., impartial culture, Mallows models, mixtures, etc.). As availability probabilities approach 1 (i.e., unavailability is rare), constructing optimal policies becomes easier, as does analysis of query complexity. Assume $p_x = p = 1 - O(\varepsilon)$ for all x and all query costs are identical. The query policy $Extreme(\mathbf{v})$ (informally) proceeds as follows: initialize the *potential set* X with all candidates, the *known set* $Y = \emptyset$, and the *current winner* $w = r(\mathbf{v}(X)) = r(\mathbf{v})$. Then repeat:

1. find a minimum-size subset Z of $X \setminus Y$ s.t. w is a robust winner for $Y \cup Z$ ($w \in Z$ if w is not known to be available);
2. check availability of all candidates in Z; add to Y those that are available, and remove from X those that are not;
3. if all candidates in Z are available, stop and output w;
4. if not, recompute $w = r(\mathbf{v}(X))$, and go back to step 1.

It is not hard to show that $Extreme(\mathbf{v})$ terminates, and returns the true winner $r(\mathbf{v}(A))$ for the actual available set A if at least one candidate is available. If $Y \subseteq X$ is a smallest (cardinality) set of candidates such that $r(\mathbf{v})$ is a robust winner for Y, then its expected query cost is $|Y| + O(\varepsilon)$. We can also show that any r-sufficient policy has an expected cost of at least $|Y| - O(\varepsilon)$.[4] These facts prove:

Proposition 7 *The policy* $Extreme(\cdot)$ *is asymptotically optimal as* $\varepsilon \to 0$.

Empirical Evaluation

We now discuss experiments that test the effectiveness of our algorithms for computing query policies, and examine the expected costs of these policies for various voting rules, preference distributions and availability distributions. We generate vote profiles using *Mallows distributions* over rankings (Mallows 1957), given by a modal ranking σ over X and dispersion $\phi \in (0, 1]$: the probability of vote v is $\Pr(v|\sigma, \phi) \propto \phi^{d(r,\sigma)}$, where d is Kendall's τ-distance. Smaller ϕ concentrates mass around σ while $\phi = 1$ gives the uniform distribution (i.e., *impartial culture*). We use $m = 10$ candidates and $n = 100$ voters, generating profiles for $\phi \in \{0.3, 0.8, 1.0\}$, and consider three voting rules: plurality, Borda and Copeland. We vary the availability probabilities p with each candidate having the same p. Results for each problem instance (combination of voting rule, ϕ, p) are averaged over 25 random vote profiles.

Before exploring query policies, we measure the probability of selecting an incorrect winner using a policy that selects the *naïve winner*, $r(\mathbf{v})$, ignoring candidate unavailability. An obvious lower bound on this error is $1 - p$ (i.e., when

[4] This bound is discontinuous at $\varepsilon = 0$, but then all candidates are available, so the query cost is zero. Thanks to a reviewer for pointing this out.

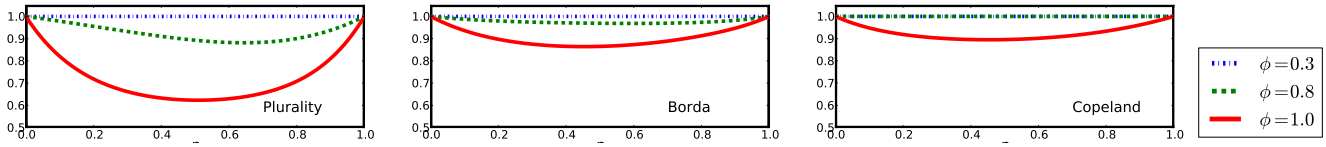

Figure 2: Probability an available naïve winner is the true winner.

	Query Cost. $\phi = 0.8$			Query Cost. $\phi = 1.0$			Tree Size. $\phi = 0.8$			Tree Size. $\phi = 1.0$		
p	0.3	0.5	0.9	0.3	0.5	0.9	0.3	0.5	0.9	0.3	0.5	0.9
Plur-DP	4.1 (3.2,5.2)	3.4 (2.0,5.4)	2.7 (1.1,5.4)	6.7 (5.9,7.6)	6.6 (4.9,7.4)	5.4 (2.4,7.9)	52.0 (9,128)	49.6 (9,124)	57.0 (9,148)	233 (133,311)	221 (121,302)	249 (136,318)
Borda-DP	3.7 (3.2,4.5)	2.7 (2.0,3.9)	1.7 (1.1,5.0)	5.4 (4.4,6.7)	4.8 (3.2,6.4)	3.3 (1.2,6.2)	24.4 (9,61)	24.0 (9,57)	26.2 (9,63)	114 (42,209)	110 (41,197)	125 (44,226)
Cope-DP	3.2 (3.2,3.6)	2.0 (2.0,2.5)	1.1 (1.1,1.3)	4.6 (3.4,5.9)	3.6 (2.1,5.6)	2.2 (1.1,4.5)	10.3 (9,19)	10.3 (9,19)	10.3 (9,19)	58.4 (17,161)	57.8 (17,160)	63.1 (17,185)
Plur-IG	4.1 (3.2,5.2)	3.5 (2.0,5.5)	2.8 (1.1,5.6)	7.0 (6.2,8.0)	6.9 (5.0,7.7)	5.6 (2.4,8.2)	59.5 (9,140)	55.4 (9,140)	62.2 (9,163)	290 (163,379)	258 (145,351)	296 (171,402)
Borda-IG	3.7 (3.2,4.6)	2.7 (2.0,3.9)	1.7 (1.1,5.0)	5.5 (4.5,7.0)	4.9 (3.2,6.7)	3.3 (1.2,6.2)	26.8 (9,63)	24.1 (9,59)	26.5 (9,68)	136 (49,264)	117 (42,229)	135 (46,253)
Cope-IG	3.2 (3.2,3.6)	2.0 (2.0,2.5)	1.1 (1.1,1.3)	4.7 (3.5,6.7)	3.7 (2.1,5.9)	2.2 (1.1,4.5)	10.3 (9,19)	10.3 (9,19)	10.4 (9,20)	67.1 (21,211)	60.8 (18,178)	70.2 (18,213)

Table 1: Avg. query cost and tree size (min, max) for optimal (DP) and myopic (IG) query policies

the winner is unavailable). Fig. 2 shows this error probability *conditional on the winner being available* for the three voting rules considered and different ϕ, as we vary p. For p near 1, the naïve winner is, of course, almost always correct. At the other extreme, the naïve winner is also usually correct, since it is highly likely to be the only available option. When preferences are very peaked ($\phi = 0.3$), candidate availability has little impact (most voters have similar rankings; but as they become more uniform the impact is dramatic. This suggests that testing availability is important even for reasonably high values of p. These results give only a crude sense of the "degree of robustness" of a winner *who is assumed to be available*, even for low p, and provide minimal insight into the value of intelligent availability testing.

We now consider the expected number of queries needed to determine the winner in the settings described above (using the same values of ϕ) under different availability probabilities: $p = 0.3, 0.5, 0.9$. Results for all three voting rules and six of nine parameter settings, with average expected cost (min, max) over 25 trials, are shown in the left half of Table 1.[5] In most settings, optimal query policies offer significant savings relative to the approach that first tests the availability of all ten candidates. The myopic heuristic tends to produce trees with costs very close to the optimum: even in problems with the largest gap (i.e., plurality with $\phi = 1$), myopic trees have an average expected cost of only 0.3 more queries than optimal; in most cases, myopic trees are almost identical to the optimum; so the more efficient myopic approach effectively minimizes query costs in practice. Not surprisingly, we see strong (negative) correlations between cost and availability probability in all three rules. Query cost is also correlated with dispersion ϕ: when ϕ is greater (more uniform), costs are higher since preferences are more diverse. When dispersion ϕ is low, given a fixed p, expected cost is the essentially the same for each rule, and the myopic approach is virtually optimal.

The right half of Table 1 shows the sizes of the decision trees that result when running both of our algorithms: tree size is only indirectly related to expected query cost, since the relative balance of the trees also impacts costs. Nonetheless we see an expected correlation, though plurality tends to result in larger trees, especially for $\phi = 1$. The myopic trees are not significantly larger than the optimal trees, though the differences in size are somewhat more pronounced than the differences in query cost.

We also ran experiments to test the effectiveness of querying for *approximate robustness*, that is, terminating the querying process when the information set ensures that a specific candidate is the true winner with probability at least $1-\delta$. Space precludes a full discussion, but using a modified version of the DP algorithm, we computed optimal query policies for values of $\delta \in \{0.001, 0.01, 0.1\}$ (i.e., exactly optimal policies given the goal of finding a $1-\delta$-robust winner). With plurality, dispersion $\phi = 1.0$ and $p = 0.9$, fully robust policies had an expected cost of 5.42 queries on average. Allowing $1 - \delta$-robust winners greatly reduced the expected cost: with $\delta = 0.001$ average expected cost was 4.97 queries; for $\delta = 0.01$, 4.36 queries; and for $\delta = 0.1$, 3.04 queries. For $p = 0.3$, setting $\delta = 0.1$ saw a reduction to 5.28 queries (compared to 6.73 for exact robustness). Other voting rules exhibited similar patterns.

Future Directions

We have offered a new perspective on voting in the unavailable candidate model, assuming that testing the viability or availability of candidates is costly. Using robust winners, irrelevant candidates, and query policies, our algorithms for computing query policies were shown to be effective, and empirical results demonstrated the value of optimal querying. A number of important research directions remain, including: efficient methods for pruning available sets w.r.t. specific voting rules; sample-based methods for reducing training set size; further development of policies that "predict" winners; deeper theoretical study of query and communication complexity; and analysis of manipulation.

Acknowledgments: Thanks to the reviewers for helpful suggestions. This work was supported in part by NSERC and by MICINN project TIN2011-27652-C03-02.

[5]Results for $\phi = 0.3$ are not shown, as they are identical for all three voting rules and both algorithms. For $p = 0.3$, avg. query cost is 3.2; $p = 0.5$, avg. cost is 2.0; and $p = 0.9$ avg. cost is 1.1. Tree sizes (right half of the table) for $\phi = 0.3$ are *constant* (size is always 9 nodes) across all rules, algorithms, and p values.

References

Baldiga, K., and Green., J. 2013. Assent-maximizing social choice. *Social Choice and Welfare* 40(2):439–460.

Bartholdi, J.; Tovey, C.; and Trick, M. 1992. How hard is it to control an election? *Social Choice and Welfare* 16(8-9):27–40.

Chevaleyre, Y.; Lang, J.; Maudet, N.; and Monnot, J. 2011. Compilation/communication protocols for voting rules with a dynamic set of candidates. In *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-11)*, 153–160.

Chevaleyre, Y.; Lang, J.; Maudet, N.; Monnot, J.; and Xia, L. 2012. New candidates welcome! possible winners with respect to the addition of new candidates. *Mathematical Social Sciences* 64(1):74–88.

Dutta, B.; Jackson, M. O.; and Breton, M. L. 2001. Strategic candidacy and voting procedures. *Econometrica* 69(4):1013–1037.

Erdélyi, G.; Fellows, M. R.; Piras, L.; and örg Rothe, J. 2011. Control complexity in Bucklin and fallback voting. arXiv 1103.2230.

Faliszewski, P.; Hemaspaandra, E.; and Hemaspaandra, L. 2011. Multimode control attacks on elections. *Journal of Artificial Intelligence Research* 40:305–351.

Faliszewski, P.; Hemaspaandra, E.; and Schnoor, H. 2008. Copeland voting: Ties matter. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-08)*, 983–990.

Garey, M. R. 1972. Optimal binary identification procedures. *SIAM Journal of Applied Mathematics* 23:173–186.

Greiner, R.; Grove, A. J.; and Roth, D. 1992. Learning cost-sensitive active classifiers. *Artificial Intelligence* 139(2):137–174.

Hemaspaandra, E.; Hemaspaandra, L.; and Rothe, J. 2007. Anyone but him: The complexity of precluding an alternative. *Artificial Intelligence* 171(5-6):255–285.

Hyafil, L., and Rivest, R. L. 1976. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters* 5:15–17.

Lu, T., and Boutilier, C. 2010. The unavailable candidate model: A decision-theoretic view of social choice. In *Proceedings of the Eleventh ACM Conference on Electronic Commerce (ACM EC-10)*, 263–274.

Mallows, C. L. 1957. Non-null ranking models. *Biometrika* 44:114–130.

Parkes, D. C., and Xia, L. 2012. A complexity-of-strategic-behavior comparison between schulze's rule and ranked pairs. In *Proceedings of the Twenty-sixth AAAI Conference on Artificial Intelligence (AAAI-12)*, 1429–1435.

Procaccia, A. D.; Rosenschein, J. S.; and Kaminka, G. A. 2007. On the robustness of preference aggregation in noisy environments. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-07)*, 416–422.

Quinlan, J. R. 1993. *C45: Programs for Machine Learning*. Morgan Kaufmann.

Rastegari, B.; Condon, A.; Immorlica, N.; and Leyton-Brown, K. 2013. Two-sided matching with partial information. In *Proceedings of the Fourteenth ACM Conference on Electronic Commerce (ACM EC-13)*, 733–750.

Shiryaev, D.; Yu, L.; and Elkind, E. 2013. On elections with robust winners. In *Proceedings of the Twelfth Conference on Autonomous Agents and Multiagent Systems (AAMAS-13)*, 415–422.

Turney, P. D. 1995. Cost-sensitive classification: Empirical vvaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research* 2:369–409.

Wojtas, K., and Faliszewski, P. 2012. Possible winners in noisy elections. In *Proceedings of the Twenty-sixth AAAI Conference on Artificial Intelligence (AAAI-12)*, 1499–1505.

Xia, L. 2012. Computing the margin of victory for various voting rules. In *Proceedings of the Thirteenth ACM Conference on Electronic Commerce (ACM EC-12)*, 982–999.