Weighted Electoral Control

Piotr Faliszewski
AGH University of Science and Technology
Krakow, Poland
faliszew@agh.edu.pl

Edith Hemaspaandra
Rochester Institute of Technology
Rochester, NY, USA
eh@cs.rit.edu

Lane A. Hemaspaandra
University of Rochester
Rochester, NY, USA
www.cs.rochester.edu/∼lane

ABSTRACT

Although manipulation and bribery have been extensively studied under weighted voting, there has been almost no work done on election control under weighted voting. This is unfortunate, since weighted voting appears in many important natural settings. In this paper, we study the complexity of controlling the outcome of weighted elections through adding and deleting voters. We obtain polynomial-time algorithms, NP-completeness results, and for many NP-complete cases, approximation algorithms. Our work shows that for quite a few important cases, either polynomial-time exact algorithms or polynomial-time approximation algorithms exist.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Security, Theory

Keywords

algorithms, control, weighted elections

1. INTRODUCTION

In many real-world election systems the voters come with weights. Examples range from stockholder elections weighted by shares, to the US Electoral College, to the often-used example of the Nassau County Board of Supervisors, to (in effect) any parliamentary system in which the parties typically vote as blocks, to Sweden's system of wealth-weighted voting instituted in 1866 (and no longer used) where "the wealthiest members of the rural communities received as many as 5,000 votes" and "in 10 percent of the districts the weighted votes of just three voters could be decisive" [5].

So it is not surprising that in the study of manipulative attacks on elections, weighted voting has been given great attention. For bribery and manipulation, two of the three most studied types of manipulative attacks on elections, study of the case of weighted voters has been extensively conducted. Yet for the remaining one of the three most studied types of attacks on elections, so-called control attacks, almost no attention has been given to the case of weighted voting; to the best of our knowledge, the only time this issue has been previously raised is in two M.S./Ph.D. theses [22, 18]. This lack of attention is troubling, since the key types of control attacks, such as adding and deleting voters, certainly do occur in many weighted elections.

We study the complexity in weighted elections of arguably the most important types of control, adding and deleting voters, for various election systems, with a particular focus on t-approval and t-veto. Control by deleting (adding) voters asks whether in a given election a given candidate can be made to win by deleting (adding) at most a certain number of the voters (at most a certain number of the members of the pool of potential additional voters). These control types model issues that are found in many electoral settings, ranging from human to electronic. They are (abstractions of) issues often faced by people seeking to steer an election, such as experts doing campaign management, and deciding for example which k people to offer rides to the polls.

Control was introduced (without weights) in 1992 in the seminal paper by Bartholdi, Tovey, and Trick [1], and control has been the subject of much attention since. That attention, and the present paper, are part of the line of work, started by Bartholdi, Tovey, and Trick, that seeks to determine for which types of manipulative attacks on elections the attacker's task requires just polynomial-time computation (see the Related Work section).

Some highlights of our results: Although weighted manipulation of scoring protocols is known to almost always be hard, Section 4.1 shows that weighted voter control for m-candidate t-approval elections is in P. Section 4.2 shows that weighted control by adding and deleting voters is NP-complete for every Condorcet-consistent election system. Earlier work of Lin [18] left open the question of at which t the complexity of t-veto and t-approval goes from P to NP-complete for control by adding/deleting voters in weighted elections; Section 4.3 resolves all six open cases. Section 4.4 gives polynomial-time approximation algorithms for control by adding and deleting voters under t-approval and t-veto in weighted elections.

2. PRELIMINARIES

Elections. We take an election to be a pair $E = (C, V)$, where C is a set of candidates and V is a collection of voters. Each voter has a preference order over the set C. A *prefer-*

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

ence order is a total, linear order that ranks the candidates from the most preferred one to the least preferred one. For example, if $C = \{a, b, c\}$ and some voter likes a best, then b, and then c, then his or her preference order is $a > b > c$.

In weighted elections, each voter v also has a positive integer weight $\omega(v)$. A voter of weight $\omega(v)$ is treated by the election system as $\omega(v)$ unweighted voters.

Election Rules. A voting rule is a function R that given an election $E = (C, V)$ returns a subset $R(E) \subseteq C$ of those candidates that are said to win the election (typically we expect to have a single winner, but sometimes ties can happen; we assume the nonunique-winner model where all tied-for-winning candidates are called winners).

An m-candidate scoring rule is defined through a nonincreasing vector $\alpha = (\alpha_1, \ldots, \alpha_m)$ of nonnegative integers. For each voter v, each candidate c receives $\alpha_{pos(v,c)}$ points, where $pos(v, c)$ is the position of c in v's preference order. The candidates with the maximum score are the winners. Many election rules are defined through families of scoring rules, with one scoring vector for each possible number of candidates. For example, plurality uses vectors of the form $(1, 0, \ldots, 0)$, t-approval uses vectors of the form $(\overbrace{1, \ldots, 1}^{t}, 0, \ldots, 0)$, and Borda's rule uses vectors of the form $(m-1, m-2, \ldots, 0)$, where m is the number of candidates. By t-veto, we mean the system which for m candidates uses the $(m-t)$-approval scoring vector. For t-approval (t-veto) we will often treat each vote as an approval vector that approves of the vote's top t (top $\|C\| - t$) candidates.

Given an election E and a voting rule R that assigns scores to the candidates, we write $score_E(c)$ to denote c's total score in E under R. The voting rule used will always be clear from context.

Given an election $E = (C, V)$, a candidate c is a (weak) Condorcet winner if for every other candidate $d \in C - \{c\}$, it holds that more than half (at least half) of the voters prefer c to d. Note that it is possible that there is no (weak) Condorcet winner in a given election.

Electoral Control. In this paper we focus on constructive control by adding/deleting voters in weighted elections and thus, due to space constraints, we will define only these two control problems. We use + to denote combining collections of voters.

DEFINITION 1. *Let R be a voting rule. In both weighted constructive control by adding voters under rule R (R-WCCAV) and weighted constructive control by deleting voters under rule R (R-WCCDV), our input contains a set of candidates C, a collection of weighted voters V (with preferences over C), a preferred candidate $p \in C$, and a nonnegative integer k. In R-WCCAV we also have an additional collection W of weighted voters (with preferences over C). In these problems we ask the following questions:*

***R*-WCCAV.** *Is there a subcollection W' of W, of at most k voters, such that $p \in R(C, V+W')$?*

***R*-WCCDV.** *Is there a subcollection V' of V, of at most k voters, such that $p \in R(C, V-V')$?*

We will consider approximation algorithms for WCCAV and WCCDV under t-approval/t-veto. In those cases we will assume that input instances do not contain the integer k and the goal is simply to find (when success is possible at all) as small as possible a collection of voters to add/delete such that p is a winner of the resulting election. For a positive integer h, an h-approximation algorithm for WCCAV/WCCDV is an algorithm that always finds a solution that adds/deletes at most h times more voters than is necessary. For t-approval/t-veto WCCDV it is always possible to ensure that p is a winner by deleting voters (it suffices to delete all the voters that do not approve of p, possibly being left with no voters at all). For t-approval/t-veto WCCAV, it is possible to ensure p's victory through adding voters if and only if p is a winner after we add all the voters that approve of p. These observations make it particularly convenient to consider approximation algorithms for t-approval and for t-veto. For other scoring rules (and other voting rules in general), such analysis would be much more complicated (indeed, the reader might want to compare our work with an attempt at creating a general election-problem approximation framework of Brelsford et al. [4]).

For certain voting rules R, we will compare the complexity of WCCAV and of the weighted coalitional manipulation problem (WCM). In WCM we are given a weighted election (C, V), a preferred candidate $p \in C$, and a sequence $k_1, \ldots, k_n$ of positive integers. We ask if it is possible to construct a collection $W = (w_1, \ldots, w_n)$ of n voters such that for each i, $1 \leq i \leq n$, $\omega(w_i) = k_i$, and p is a winner of R election $(C, V+W)$. (The voters in W are called *manipulators*.) WCM is similar to WCCAV in that we also add voters, but it differs in that (a) we have to add exactly n voters, and (b) we can pick the preference orders of the added voters.

Due to space restrictions we omit most of the discussion of what real-life scenarios the problems presented here model, and instead we point the reader to the survey of Faliszewski, Hemaspaandra, and Hemaspaandra [10].

Computational Complexity We assume familiarity with standard notions of complexity theory. In our NP-hardness proofs we use reductions from the following two standard NP-complete problems.

DEFINITION 2. *An instance of Partition consists of a sequence $(k_1, \ldots, k_n)$ of positive integers. We ask whether there is a set $I \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in I} k_i = \frac{1}{2} \sum_{i=1}^{n} k_i$.*

DEFINITION 3. *An instance of Exact-Cover-By-3-Sets (X3C) contains a set $B = \{b_1, \ldots, b_{3t}\}$ and a family $\mathcal{S} = (S_1, \ldots, S_n)$ of 3-element subsets of B. We ask whether there is a t-element set $I \subseteq \{1, \ldots, n\}$ such that $\bigcup_{i \in I} S_i = B$.*

3. RELATED WORK

The study of the complexity of (unweighted) electoral control was initiated by Bartholdi, Tovey, and Trick [1], who studied constructive control by adding/deleting/partitioning candidates/voters under the plurality and Condorcet rules. The various types of control model at least some of the flavor of actions that occur in the real world, such as voter suppression and targeted get-out-the-vote drives. A major motivation for the study of control was to obtain "complexity barrier" results, that is, results that show that detecting opportunities for various control attacks is computationally difficult (specifically, Bartholdi, Tovey, and Trick focused on and obtained many NP-hardness results).

This research direction was continued by Hemaspaandra, Hemaspaandra, and Rothe [15], who studied destructive control attacks on elections, where one's goal is to prevent a given candidate from being a winner through various control actions. Since then, many authors have studied electoral control in many varied settings and under many different rules; we refer the reader to the survey [10]. Some recent research, not covered in that survey, includes complexity-of-control results for the t-approval family of rules [18], for Bucklin's rule (and for fallback, its extension for truncated votes) [8, 7], for maximin [9], for range voting [19], and for Schultze's rule and the ranked pairs rule [20].

The complexity-barrier research line turned out to be very successful. For most voting rules that were considered a significant number of control attacks are NP-hard. Indeed, it is even possible to construct an artificial election system resistant to all types of control attacks [16]. However, there are also a number of results that suggest that in practice the complexity barrier might not be as strong as one might at first think. For example, Faliszewski et al. [12] and Brandt et al. [3] have shown that if the votes are restricted to being single-peaked, then many control problems that are known to be NP-complete become polynomial-time solvable. Indeed, this often holds even if elections are just nearly single-peaked [11], as many real-world elections seem to be. Similarly, some initial experimental results of Rothe and Schend [21]—published very recently—suggest that, at least under certain distributions and settings, some NP-hard control problems can be solved in practice on many instances. As part of a different line of research, Xia [24] has studied the asymptotic behavior of the number of voters that have to be added to/deleted from a randomly constructed election in a successful control action.

We mention that there are many problems related to control that researchers study, but whose discussion we have to omit due to limited space. (These problems include, e.g., cloning, adding candidates not included in voters' preference orders, and certain problems pertaining to truncated votes.)

The only papers that directly raise the issue of weighted control are, to the best of our knowledge, the theses of Russell [22] and Lin [18]. However, we also mention the paper of Baumeister et al. [2], where the authors consider the problem of affecting the result of an election through picking the weights of the voters. Their problem is similar to, though different than, simultaneous (multimode) addition and deletion of voters.

4. RESULTS

This section presents our results.

4.1 Scoring Protocols

Weighted manipulation of scoring protocols is always hard, unless the scoring protocol is in effect plurality or triviality [14]. In contrast, weighted voter control is easy for m-candidate t-approval.

THEOREM 4. *For all m and t, WCCAV and WCCDV for m-candidate t-approval are in* P.

PROOF. Let (C, V, W, p, k) be an instance of WCCAV for m-candidate t-approval. We can assume that we add only voters who approve of p. We can also assume that we add the heaviest voters with a particular set of approvals, i.e., if we have added ℓ voters approving $p, c_1, \ldots, c_{t-1}$, we can assume that we added the ℓ heaviest voters approving $p, c_1, \ldots, c_{t-1}$. Since there are only $\binom{m-1}{t-1}$—which is a constant—different sets of approvals to consider, it suffices to try all sequences of nonnegative integers $k_1, \ldots, k_{\binom{m-1}{t-1}}$ whose sum is at most k and for each such sequence to check whether adding the heaviest k_i voters of the ith approval collection makes p a winner.

The same argument works for WCCDV. Here, we delete only voters that do not approve of p, and again we delete the heaviest voters for each approval collection. □

One might think that the argument above works for any scoring protocol, but this is not the case. For example, consider the 3-candidate Borda instance where V consists of one weight-1 voter $b > p > a$ and W consists of a weight-2 and a weight-1 voter with preference order $a > p > b$. Then adding the weight-1 voter makes p a winner, but adding the weight-2 voter does not. And in fact, we have the following result.

THEOREM 5. *WCCAV and WCCDV for Borda are* NP*-complete. This result holds even when restricted to a fixed number $m \geq 3$ of candidates.*

PROOF. We reduce from Partition. Given a sequence $k_1, \ldots, k_t$ of positive integers that sum to $2K$, construct an election with one registered voter of weight K voting $b > p > a > \cdots$, and t unregistered voters with weights $k_1, \ldots, k_t$ voting $a > p > b > \cdots$. Set the addition limit to t. It is easy to see that for p to become a winner, b's score (relative to p) needs to go down by at least K, while a's score (relative to p) should not go up by more than K. It follows that $k_1, \ldots, k_t$ has a partition if and only if p can be made a winner.

We use the same construction for the deleting voters case. Now, all voters are registered and the deletion limit is t. Since we can't delete all voters, and since our goal is to make p a winner, we can't delete the one voter voting $b > p > a > \cdots$. The rest of the argument is identical to the adding voters case. □

The proof above also establishes the result for the case of control by unlimited addition of voters (i.e., no bound on the number of additions). The analogue of the above theorem in the different model in which we are bounding the total weight of votes that can be added/deleted has been obtained by Russell [22].

We conjecture that the cases of Theorem 4 are in effect the only P cases here.

CONJECTURE 6. *For all scoring protocols $(\alpha_1, \ldots, \alpha_m)$, if there exists an i, $1 < i < m$, such that $\alpha_1 > \alpha_i > \alpha_m$, then WCCAV and WCCDV for $(\alpha_1, \ldots, \alpha_m)$ are* NP*-complete, otherwise, they are in* P.

The candidate control cases for scoring protocols are not that interesting, since they are all in P by brute force (as the number of candidates is constant). For unbounded numbers of candidates, the unweighted candidate control versions (destructive as well as constructive) for plurality (and t-approval and t-veto) are already hard [1, 15, 6, 18].

WDCAV and WDCDV for scoring protocols (that is, the destructive variants of WCCAV and WCCDV where the goal is to ensure that a given candidate is not a winner) are easy: Simply loop through all candidates c, $c \neq p$, and greedily add or delete voters to boost the score of c relative to p as much as possible.

4.2 Manipulation Versus Control

Theorem 4 gives cases where WCCAV and WCCDV are easy, while WCM is hard. The opposite is also possible. An election system is Condorcet consistent if whenever there exists a Condorcet winner, he or she is the unique winner. An election system is weakCondorcet consistent if whenever there exist weak Condorcet winners, these are exactly the winners.

THEOREM 7. *For every weakCondorcet-consistent election system and for every Condorcet-consistent election system, WCCAV and WCCDV are* NP*-complete. This result holds even when restricted to a fixed number $m \geq 3$ of candidates.*

PROOF. To show that WCCAV is NP-complete, we reduce from Partition. Given a sequence $k_1, \ldots, k_t$ of positive integers that sum to $2K$, construct an election with two registered voters, one voter with weight 1 voting $p > a > b > \cdots$ and one voter with weight $2K$ voting $b > p > a > \cdots$, and t unregistered voters with weights $2k_1, \ldots, 2k_t$ voting $a > p > b > \cdots$. Set the addition limit to t. Suppose we add unregistered voters to the election with a total vote weight equal to $2L$.

- If $L < K$, then b is the Condorcet winner, and thus the unique winner of the election.
- If $L > K$, then a is the Condorcet winner, and thus the unique winner of the election.
- If $L = K$, then p is the Condorcet winner, and thus the unique winner of the election.

The WCCDV case uses the same construction. Now, all voters are registered and the deletion limit is t. Since we can delete at most t of our $t + 2$ voters, and since our goal is to make p a winner, we can't delete the sole voter voting $b > p > a$, since then a would be the Condorcet winner. The rest of the argument is similar to the adding voters case. □

Let Condorcet be the election system whose winner set is exactly the set of Condorcet winners. Let weakCondorcet be the election system whose winner set is exactly the set of weak Condorcet winners.

COROLLARY 8. *For Condorcet and weakCondorcet, WCM is in* P *and WCCAV and WCCDV are* NP*-complete. This result holds even when restricted to a fixed number $m \geq 3$ of candidates.*

Condorcet and weakCondorcet do not always have winners. For those who prefer their voting systems to always have at least one winner, we note that 3-candidate Llull (which is the election system where the score of each candidate c is the number of candidates d, $d \neq c$, such that at least half of the voters prefer c to d; candidates with the highest score are the winners) always has at least one winner, is easily seen to be weakCondorcet consistent, and WCM for this system is in P [13].

COROLLARY 9. *For 3-candidate Llull, WCM is in* P *and WCCAV and WCCDV are* NP*-complete.*

4.3 t-Approval with an Unbounded Number of Candidates

Lin [18] showed that for $t \geq 4$, WCCAV for t-approval and WCCDV for t-veto are NP-complete, and that for $t \geq 3$, WCCDV for t-approval and WCCAV for t-veto are NP-complete. These results hold even for the unweighted case. It is also known that the remaining unweighted cases are in P [1, 18] and that WCCAV and WCCDV for plurality and veto are in P [18].

In this section, we look at and solve the remaining open cases, WCCAV for 2-approval, 3-approval, and 2-veto, and WCCDV for 2-approval, 2-veto, and 3-veto. We start by showing that 2-veto-WCCAV is in P.

We will later show that a natural greedy-by-weight heuristic approach can never obtain better than a 2-approximation for 2-approval-WCCAV (see the comment about $t = 2$ at the start of the proof of Theorem 17). Nonetheless, we now show that 2-approval-WCCAV can be exactly solved in P, by an algorithm that iteratively removes all unnecessary voters.

THEOREM 10. *WCCAV for 2-approval is in* P.

PROOF. Run the following algorithm. (In this algorithm and the proof of correctness, whenever we speak of the r heaviest voters in voter set X, we mean the $\min(r, \|X\|)$ heaviest voters in X.)

On input (C, V, W, p, k)
for all $c \in C - \{p\}$ **do**
  let $s_c = score_{(C,V)}(c) - score_{(C,V)}(p)$.
end for
Delete all voters from W that do not approve of p.
repeat
  for all $c \in C - \{p\}$ **do**
    if the sum of the weights of the k heaviest voters in W that do not approve of c is less than s_c **then**
      reject
      {It is impossible to get $score(c) \leq score(p)$.}
    end if
  end for
  for all $c \in C - \{p\}$ and $\ell \in \{1, \ldots, k-1\}$ **do**
    if the sum of the weights of the $k - \ell$ heaviest voters in W that do not approve of c is less than s_c **then**
      delete all voters approving c except for the $\ell - 1$ heaviest such voters from W.
      {We need to add at least $k - \ell + 1$ voters that do not approve of c, and so we can add at most $\ell - 1$ voters approving c.}
    end if
  end for
until no more changes.
if $\|W\| \geq k$ **then**
  accept
  {We can make p a winner by adding the k heaviest voters from W.}
end if
if $\|W\| < k$ **then**
  accept if and only if adding all of W will make p a winner.
end if

Why does this work? It is easy to see that we never reject incorrectly in the repeat-until. Also, it is easy to see that if we add r voters approving $\{p, c\}$, we may assume that we add the r heaviest voters approving $\{p, c\}$ (this is also

crucial in the proof of Theorem 4), and so we never delete voters incorrectly in the second for loop in the repeat-until.

If we get through the repeat-until without rejecting, and we have fewer than k voters left in W, then adding all of W is the best we can do (since all voters in W approve p).

Finally, if we get through the repeat-until, and we have at least k voters left in W, then adding the k heaviest voters from W will make p a winner. Why? Let c be a candidate in $C - \{p\}$. Let r be the number of voters from W that are added and that approve of c. Since we made it through the repeat-until, we know that [the sum of the weights of the k heaviest voters in W that do not approve of c] is at least s_c. And so if $r = 0$, $score(c) - score(p) = s_c$ - [the sum of the weights of the k heaviest voters in W] ≤ 0. If $r > 0$, and there are at least r voters approving of c left in W, then [the sum of the weights of the $k - r$ heaviest voters in W that do not approve of c] is at least s_c. And so $score(c) - score(p) = s_c$ - [the sum of the weights of the $k - r$ heaviest voters in W that do not approve of c] ≤ 0 □

THEOREM 11. *WCCDV for 2-veto is in* P.

Instead of proving this theorem directly, we show a more general relation between the complexity of t-approval/t-veto WCCAV and WCCDV.

THEOREM 12. *For each fixed t, it holds that t-veto-WCCDV (t-approval-WCCDV) many-one reduces to t-approval-WCCAV (t-veto-WCCAV).*

PROOF. We give a reduction from t-veto-WCCDV to t-approval-WCCAV. The idea is that deleting a t-veto vote v from t-veto election (C, V) is equivalent, in terms of net effect on the scores, to adding a t-approval vote v' to this election, where v' approves exactly of the t candidates that v disapproves of. The problem with this approach is that we are to reduce t-veto-WCCDV to t-*approval*-WCCAV and thus we have to show how to implement t-veto scores with t-approval votes.

Let (C, V, p, k) be an instance of t-veto-WCCDV, where $V = (v_1, \ldots, v_n)$. Let $m = \|C\|$. Let $\omega_{\max}$ be the highest weight of a vote in V. We set D to be a set of up to $t - 1$ new candidates, such that $\|C\| + \|D\|$ is a multiple of t. We set V_0 to be a collection of $\frac{\|C\|+\|D\|}{t}$ t-approval votes, where each vote has weight $\omega_{\max}$ and each candidate in $C \cup D$ is approved of by exactly one of the votes in V_0. For each vote v_i in V we create a set $C_i = \{c_i^1, \ldots, c_i^{(t-1)(m-t)}\}$ and we create a collection of voters $V_i = (v_i^1, \ldots, v_i^{m-t})$. Each voter v_i^j, $1 \leq j \leq m - t$, has weight $\omega(v_i)$ and approves of the jth candidate approved by v and of the $t - 1$ candidates $c_i^{(j-1)(t-1)+1}, \cdots, c_i^{j(t-1)}$.

We form an election $E' = (C', V')$, where $C' = C \cup D \cup \bigcup_{i=1}^{n} C_i$ and $V' = V_0 + V_1 + \cdots + V_n$. It is easy to see that each candidate $c \in C$ has t-approval score $\omega_{\max} + score^{t\text{-veto}}_{(C,V)}(c)$, where $score^{t\text{-veto}}_{(C,V)}(c)$ is the t-veto score of c in (C, V). Further, each candidate $c \in C' - C$ has t-approval score at most $\omega_{\max}$ in E'.

We form an instance (C', V', W, p, k) of t-approval-WCCAV, where $W = (w_1, \ldots, w_n)$, and for each i, $1 \leq i \leq n$, $\omega(w_i) = \omega(v_i)$, and w_i approves exactly of those candidates that v_i disapproves of. It is easy to see that adding voter w_i to t-approval election (C', V') has the same net effect on the scores of the candidates in C as does deleting v_i from t-veto election (C, V).

A similar proof, though with a slightly more involved construction of the votes in V', gives a reduction from t-approval-WCCDV to t-veto-WCCAV. □

All other remaining cases are NP-complete. Interestingly, we need only a very limited set of weights in our reductions.

THEOREM 13. *WCCAV for 2-veto and 3-approval and WCCDV for 2-approval and 3-veto are* NP*-complete.*

PROOF. Due to of space limitations, we will give the proof only for WCCAV for 2-veto. This case is the simplest, but the proofs of the other cases are quite similar in flavor. We will reduce from X3C (exact cover by 3-sets). Let $(B, \mathcal{S})$ be our input X3C instance, where $B = \{b_1, ..., b_{3t}\}$ and $\mathcal{S} = \{S_1, ..., S_n\}$ is a family of 3-element subsets of B. W.l.o.g., we assume that $n \geq t$.

Given this instance of X3C, we construct the following instance (C, V, W, p, k) of WCCAV for 2-veto. We set $C = \{p\} \cup B \cup \{s_i \mid S_i \in \mathcal{S}\} \cup \{d\}$ (d is a dummy candidate that is used for padding) and V consists of the following voters:

weight	preference order	
3	$\cdots > p > d$	
2	$\cdots > b_j > d$	for all $1 \leq j \leq 3t$.

W consists of the following voters:

weight	preference order	
3	$\cdots > s_i > d$	for all $1 \leq i \leq n$
1	$\cdots > s_i > b_{i_1}$	and $S_i = \{b_{i_1}, b_{i_2}, b_{i_3}\}$
1	$\cdots > s_i > b_{i_2}$	
1	$\cdots > s_i > b_{i_3}$.	

We claim that $\mathcal{S}$ contains an exact cover if and only if p can become a winner after adding at most $n + 2t$ voters of W to V.

($\Rightarrow$): Add the $(n - t)$ weight-3 voters corresponding to the sets not in the cover and add the $3t$ weight-1 voters corresponding to the sets in the cover. Now compute the veto weight for each candidate (that is easier than computing the scores): $vetoes(p) = 3$, $vetoes(d) \geq 5$, $vetoes(b_j) = 3$, and $vetoes(s_i) = 3$. So, p is a winner.

($\Leftarrow$): Let W be a set of at most $n + 2t$ added voters that make p a winner. Note that $vetoes_W(b_j) \geq 1$ and $vetoes_W(s_i) \geq 3$. It is easy to see that W consists of $(n - t)$ weight-3 voters and $3t$ weight-1 voters. The $3t$ weight-1 voters together veto each b_j once and t of the s_i's 3 times. But then the weight-1 voters correspond to an exact cover. □

The construction above uses only weights 1, 2, and 3. It is even possible to use only weights 1 and 2, or 1 and 3.

4.4 Approximation and Greedy Algorithms

When problems are computationally difficult, such as being NP-complete, it is natural to wonder whether good polynomial-time approximation algorithms exist. So, motivated by the NP-completeness results discussed earlier in this paper for most cases of WCCAV/WCCDV for t-approval and t-veto, this section studies greedy and other approximation algorithms for those problems. (Recall that WCCAV is NP-complete for t-approval, $t \geq 3$, and for t-veto, $t \geq 2$, and WCCDV is NP-complete for t-approval, $t \geq 2$, and for t-veto, $t \geq 3$.) Although we are primarily interested in constructing good approximation algorithms, we are also interested in cases where particular greedy strategies can be shown to fail to provide good approximation algorithms, as doing so helps one eliminate such approaches

from consideration and sheds light on the approach's limits of applicability. First, we will establish a connection to the weighted multicover problem, and will use it to obtain approximation results. Then we will obtain an approximation algorithm that will work by direct action on our problem. Table 3 summarizes our results on approximation algorithms for t-approval/t-veto WCCAV/WCCDV.

Weighted Multicover Approach. Let us first consider the extent to which known algorithms for the Set-Cover family of problems apply to our setting. Specifically, we will use the following generalization of Set-Cover.

DEFINITION 14. *An instance of Weighted Multicover (WMC) consists of a set $B = \{b_1, \ldots, b_m\}$, a sequence $r = (r_1, \ldots, r_m)$ of nonnegative integers (covering requirements), a collection $\mathcal{S} = (S_1, \ldots, S_n)$ of subsets of B, and a sequence $\omega = (\omega_1, \ldots, \omega_n)$ of positive integers (weights of the sets in $\mathcal{S}$). The goal is to find a minimum-cardinality set $I \subseteq \{1, \ldots, n\}$ such that for each $b_j \in B$, it holds that $\sum_{i \in I \wedge b_j \in S_i} \omega_i \geq r_j$.*

That is, given a WMC instance we seek a smallest collection of subsets from $\mathcal{S}$ that satisfies the covering requirements of the elements of B (keeping in mind that a set of weight ω covers each of its element ω times). WMC is an extension of Set-Cover and is a special case of Multiset-Multicover [23]. (Even more generally, it is a special case of the Covering Integer Programs with Cardinality Constraints problem, see, e.g., [17]). We first observe that for t-approval both WCCAV and WCCDV naturally translate to equivalent WMC instances.

We consider WCCAV first. Let (C, V, W, p, k) be an instance of t-approval-WCCAV, where $W = (w_1, \ldots, w_n)$ is the collection of voters that we may add. We assume w.l.o.g. that each voter in W ranks p among its top t candidates (i.e., approves of p). This assumption is w.l.o.g. because if there is a solution to a t-approval-WCCAV instance then there is one that does not add voters that do not approve of p.

We form an instance $(B, r, \mathcal{S}, \omega)$ of WMC as follows. We set

$$B = \{b_c \mid c \in C \wedge score_{(C,V)}(c) > score_{(C,V)}(p)\},$$

and for each $b_c \in B$, we set its covering requirement to be $r_c = score_{(C,V)}(c) - score_{(C,V)}(p)$. For each vote $w \in W$, let S_w be the set of candidates that w does not approve of (by our assumptions, sets S_w never include p). We set $\mathcal{S} = (S_{w_1}, \ldots, S_{w_n})$ and we set $\omega = (\omega(w_1), \ldots, \omega(w_n))$. It is easy to see that if a set $I \subseteq \{1, \ldots, n\}$ is a solution to this instance of WMC (that is, if I satisfies all covering requirements), then adding the voters $\{w_i \mid i \in I\}$ to the election (C, V) ensures that p is a winner. The reason for this is the following: If we add voter w_i to the election then for each candidate $c \in S_{w_i}$, the difference between the score of c and the score of p decreases by $\omega(w_i)$, and for each candidate $c \in (C - \{p\}) - S_{w_i}$ this difference does not change. The covering requirements are set to guarantee that p's score will match or exceed the scores of all candidates in the election.

We stress that in the above construction we did not assume t to be a constant. Indeed, the construction applies to t-veto just as well as to t-approval. Further, based on this construction (and an analogous one for WCCDV) and an approximation algorithm of Kolliopoulos and Young [17] for minimum-cost covering integer programs with multiplicity constraints (a generalization of WMC), we obtain the following results.

THEOREM 15. *For each fixed $t \geq 1$, there is a $\mathcal{O}(\log m)$-approximation algorithm for t-approval-WCCAV. For each fixed $t \geq 2$, there is $\mathcal{O}(\log t)$-approximation algorithm for t-veto-WCCAV.*

THEOREM 16. *For each fixed $t \geq 2$, there is a $\mathcal{O}(\log t)$-approximation algorithm for t-approval-WCCDV. For each fixed $t \geq 1$, there is $\mathcal{O}(\log m)$-approximation algorithm for t-veto-WCCDV.*

Direct Approach. Using algorithms for WMC, we were able to obtain relatively strong algorithms for WCCAV/WCCDV under t-approval and t-veto. However, with this approach we did not find approximation algorithms for t-approval-WCCAV and t-veto-WCCDV whose approximation ratios do not depend on the size of the election. In the following we will seek direct algorithms for these problems. We also will observe that some greedy approaches fail to give constant-approximation algorithms for our control problems.

We now show that a very simple greedy approach yields a polynomial-time t-approximation algorithm for t-approval-WCCAV and t-veto-WCCDV. (Recall that this means that in cases when making p win is possible, the number of voters our algorithm adds/deletes to reach victory is never more than three times that of the optimal set of additions/deletions.)

Let GBW (greedy by weight) define the following very simple algorithm for WCCAV. (The votes are the weighted t-approval vectors induced by the preferences of the voters.) (Pre)discard all unregistered votes that do not approve of the preferred candidate p. Order the (remaining) unregistered votes from heaviest to lightest, breaking ties in voter weights in some simple, transparent way (for concreteness, let us say by lexicographic order on the votes' representations). GBW goes through the unregistered votes in that order, and as it reaches each vote it adds the vote exactly if the vote disapproves of at least one candidate whose score (i.e., total weight of approvals) is currently strictly greater than that of p. It stops successfully when p has become a winner and unsuccessfully if before that happens the algorithm runs out of votes to consider. The following result says that GBW is a t-approximation algorithm for t-approval-WCCAV (and also for t-approval-WCCDV, using the obvious analogue of GBW for t-veto, which we will also call GBW).

THEOREM 17. *Let $t \geq 3$. The polynomial-time greedy algorithm GBW is a t-approximation algorithm for t-approval-WCCAV and t-veto-WCCDV; and there are instances in which GBW's approximation factor on each of these problems is no better than t.*

PROOF. Examples showing that GBW can use t times too many additional votes are not hard to construct (and the lower bound even holds for $t = 2$, though in Section 4.3 we obtained an exact solution by a different approach); due to space we omit them. (One does have to be careful to set the "gap" pattern created by the unregistered voters to be a realizable one; we do this by a general trick that lets us set up certain patterns of gaps.)

We now turn to the t-approximation claims. We will prove the result for $t = 3$ and WCCAV, but it will be immedi-

ately clear that our proof straightforwardly generalizes to all greater t; and the WCCDV case follows using Theorem 12.

Clearly GBW is a polynomial-time algorithm. Consider a given input instance of t-approval-WCCAV, with preferred candidate p. W.l.o.g. assume all unregistered voters approve of p. We will say a candidate "has a gap" (under the current set of registered voters and whatever unregistered voters have already been added) if that candidate has strictly more weight of approvals than p does. For each candidate d who has a gap, $d \neq p$, define i_d to be the minimum number of unregistered voters one has to add to remove d's gap; that is, if one went from heaviest to lightest among the unregistered voters, adding in turn each that disapproved of d, i_d is the number of voters one would add before d no longer had a gap. If for any candidate d it holds that no integer realizes i_d, then control is impossible using the unregistered voter set. Clearly, any successful addition of voters must add at least $\max_d i_d$ voters (the max throughout this proof is over all candidates initially having a gap).

Let us henceforth assume that control is possible in the input case. We will show that after having added at most $3 \cdot \max_d i_d$ voters GBW will have made p a winner, and so GBW is a 3-approximation algorithm. By way of contradiction, suppose that after $3 \cdot \max_d i_d$ additions some candidate, z, still has a gap.

Case 1 [In at least $\max_d i_d$ *of the first* $3 \cdot \max_d i_d$ *votes added by GBW, z is not approved].* Since for the last one of these to be added z must still have had a gap before the addition, each earlier vote considered that disapproved z had a gap for z when it was considered and so would have been added when reached. So, keeping in mind that $i_z \leq \max_d i_d$, we in fact must have added the i_z heaviest voters disapproving of z, and so contrary to the assumption, z no longer has a gap after these additions.

Case 2 [Case 1 does not hold]. So z is approved in at least $1 + 2 \cdot \max_d i_d$ of the added voters. What made the final one of the added votes, call it v', eligible for addition? It must be that some candidate, say y, still had a gap just before v' was added.

Case 2a [y is disapproved in at least $\max_d i_d$ *of the* $2 \cdot \max_d i_d$ *votes added before v' that approved z].* Then, since until y's gap was removed no unregistered voters disapproving of y would be excluded by GBW, y's i_y heaviest voters will have been added. So contrary to Case 2's assumption, y is not a gap when we get to adding vote v'.

Case 2b [Case 2 holds but Case 2a does not]. Then y is approved in at least $1 + \max_d i_d$ of the $2 \cdot \max_d i_d$ votes before v' that GBW added that approve z. So we have $1 + \max_d i_d$ votes added approving of exactly z and y. But then who made the last of *those* $1 + \max_d i_d$ votes, call it v'', eligible to be added? It must be that some candidate w had a gap up through v''. But at the moment before adding v'' we would have added $\max_d i_d \geq i_w$ votes approving exactly z and y and so disapproving w, and since w allegedly still had a gap, we while doing so under GBW would have in fact added the i_w heaviest voters disapproving of w, and so w's gap would have been removed before v'', so contrary to our assumption w was not the gap that made v'' eligible. □

Given that for each t, the greedy by weight (GBW) heuristic gives a constant-approximation algorithm for t-approval-WCCAV, it might be natural to hope that GBW might give us a constant-approximation algorithm for t-veto-WCCAV. However, we have constructed a family of examples that establishes that even for the 2-veto WCCAV problem, GBW does not yield any constant approximation.

THEOREM 18. *GBW is not an $\mathcal{O}(1)$-approximation algorithm for 2-veto-WCCAV.*

One might wonder whether other greedy approaches will provide constant-approximation algorithms for 2-veto-WCCAV. We have studied seven other natural greedy algorithms, and for each of those, we have proven that the algorithm does not provide a constant-approximation algorithm for 2-veto-WCCAV. Among those seven other algorithms are such approaches as "greedy by maximum gap before" (the next vote to add is a heaviest vote among the remaining unregistered votes that both approve of p and veto someone whose current advantage over p is maximum), "greedy maximum gap after (the addition)," "greedy by maximum gap before, tie-breaking the selection based on minimizing the maximum gap after (the addition)," GBW modified in ways to "shake it up" by trying when possible to avoid having adjacent additions being identical in who-is-vetoed, or even by trying to have them avoid intersections in adjacent who-is-vetoed sets (this type of approach may seem a strange thing to do, but an approval analogue of it yielded our first $\mathcal{O}(1)$-approximation algorithm in one of the t-approval cases), and various others. We don't define these approaches in detail here since we have concrete examples showing all of them fail to yield $\mathcal{O}(1)$-approximation algorithms (and even somewhat stronger claims hold as to their failure to approximate well). Our planned full version will give more clearly these greedy approaches, and go over the related counterexample constructions. However, we mention one greedy heuristic for which we could not prove the lack of $\mathcal{O}(1)$-approximation performance. That heuristic is the attractive approach of adding next an unregistered vote that induces the largest decrease in the sum of the gaps, by which we mean the sum over all candidates with strictly more weight of approvals than p of how much more weight of approvals than p they have. Not only do we not have a proof that that is not an $\mathcal{O}(1)$-approximation for 2-veto-WCCAV, but also it is a strong candidate for giving an even better approximation algorithm for 3-approval-WCCAV than the 3-approximation algorithm given by GBW. For 3-approval-WCCAV, we can show that the sums-of-gaps approach cannot yield a $(3/2 - \epsilon)$-approximation algorithm, but this is just a weak first step toward understanding that heuristic, which we commend as an interesting open direction.

5. CONCLUSIONS

We have studied voter control under a number of voting rules, including t-approval, t-veto, Borda, and the family of (weak) Condorcet-consistent rules. We have completed the classification of the complexity of weighted voter control for t-approval and t-veto, and we have shown that the complexity of voter control can be quite different than the complexity of weighted coalitional manipulation: there are natural voting rules for which weighted coalitional manipulation is easy but weighted voter control is hard, and there are natural rules where the opposite is the case. Further, we have shown that for weighted voter control under t-approval and t-veto, there are good, natural approximation algorithms. Our results for voter control in weighted elections are summarized in Tables 1, 2, and 3.

	WCCAV	WCCDV
t-approval	P (Thm. 4)	P (Thm. 4)
Borda	NPC (Thm. 5)	NPC (Thm. 5)
(weak)Condorcet-consistent rules	NPC (Thm. 7)	NPC (Thm. 7)

Table 1: Our results for the complexity of control by adding/deleting voters in weighted elections for any fixed number of candidates, $m \geq 3$.

	WCCAV	WCCDV
t-approval		
$t = 2$	P (Thm. 10)	NPC (Thm. 13)
$t = 3$	NPC (Thm. 13)	NPC [18]
$t \geq 4$	NPC [18]	NPC [18]
t-veto		
$t = 2$	NPC (Thm. 13)	P (Thm. 11)
$t = 3$	NPC [18]	NPC (Thm. 13)
$t \geq 4$	NPC [18]	NPC [18]

Table 2: The complexity of control by adding and deleting voters for t-approval and t-veto with an unbounded number of candidates.

	t-approval	t-veto
WCCAV	$\mathcal{O}(\log m)$ [Thm. 15] t [Theorem 17]	$\mathcal{O}(\log t)$ [Thm. 15]
WCCDV	$\mathcal{O}(\log t)$ [Thm. 16]	$\mathcal{O}(\log m)$ [Thm. 16] t [Theorem 17]

Table 3: Approximation ratios of our algorithms for WCCAV and WCCDV under t-approval and t-veto.

Acknowledgments. Supported in part by grants AGH-11.11.120.865, NCN's DEC-2011/03/B/ST6/01393, and NSF-CCF-{0915792,1101452,1101479}, and two Bessel Awards.

6. REFERENCES

[1] J. Bartholdi, III, C. Tovey, and M. Trick. How hard is it to control an election? *Mathematical and Computer Modeling*, 16(8/9):27–40, 1992.

[2] D. Baumeister, M. Roos, J. Rothe, L. Schend, and L. Xia. The possible winner problem with uncertain weights. In *Proc. of ECAI-12*, pages 133–138, Aug. 2012.

[3] F. Brandt, M. Brill, E. Hemaspaandra, and L. Hemaspaandra. Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. In *Proc. of AAAI-10*, pages 715–722, July 2010.

[4] E. Brelsford, P. Faliszewski, E. Hemaspaandra, H. Schnoor, and I. Schnoor. Approximability of manipulating elections. In *Proc. of AAAI-08*, pages 44–49. AAAI Press, July 2008.

[5] R. Congleton. The Swedish transition to democracy (Chapter 14). In *Perfecting Parliament*. Cambridge University Press, 2011.

[6] E. Elkind, P. Faliszewski, and A. Slinko. Cloning in elections: Finding the possible winners. *JAIR*, 42:529–573, 2011.

[7] G. Erdélyi, L. Piras, and J. Rothe. The complexity of voter partition in Bucklin and fallback voting: Solving three open problems. In *Proc. of AAMAS-11*, pages 837–844, May 2011.

[8] G. Erdélyi and J. Rothe. Control complexity in fallback voting. In *Proc. of 16th Australasian Theory Symposium*, pages 39–48, Jan. 2010.

[9] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. Multimode attacks on elections. In *Proc. of IJCAI-09*, pages 128–133, July 2009.

[10] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. Using complexity to protect elections. *Commun. ACM*, 53(11):74–82, 2010.

[11] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. The complexity of manipulative attacks in nearly single-peaked electorates. In *Proc. of TARK-11*, pages 228–237, July 2011.

[12] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. The shield that never was: Societies with single-peaked preferences are more open to manipulation and control. *Information and Computation*, 209:89–107, 2011.

[13] P. Faliszewski, E. Hemaspaandra, and H. Schnoor. Copeland voting: Ties matter. In *Proc. of AAMAS-08*, pages 983–990, May 2008.

[14] E. Hemaspaandra and L. Hemaspaandra. Dichotomy for voting systems. *Journal of Computer and System Sciences*, 73(1):73–83, 2007.

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

[16] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Hybrid elections broaden complexity-theoretic resistance to control. *Mathematical Logic Quarterly*, 55(4):397–424, 2009.

[17] S. Kolliopoulos and N. Young. Approximation algorithms for covering/packing integer programs. *Journal of Computer and System Sciences*, 71(4):495–505, 2005.

[18] A. Lin. *Solving Hard Problems in Election Systems*. PhD thesis, Rochester Institute of Technology, Rochester, NY, 2012.

[19] C. Menton. Normalized range voting broadly resists control. Technical Report arXiv:1005.5698 [cs.GT], arXiv.org, May 2010. Revised June 2012.

[20] D. Parkes and L. Xia. A complexity-of-strategic-behavior comparison between Schulze's rule and ranked pairs. In *Proc. of AAAI-12*, pages 1429–1435, July 2012.

[21] J. Rothe and L. Schend. Control complexity in Bucklin, Fallback, and Plurality voting: An experimental approach. In *Proc. of SEA-12*, pages 356–368, June 2012.

[22] N. Russell. Complexity of control of Borda count elections. Master's thesis, Rochester Institute of Technology, 2007.

[23] V. Vazirani. *Approximation Algorithms*. Springer, 2003.

[24] L. Xia. How many vote operations are needed to manipulate a voting system? In *Proc. of COMSOC-12*, pages 443–454, Sept. 2012.