Very Hard Electoral Control Problems

Zack Fitzsimmons, Edith Hemaspaandra, Alexander Hoover, and David E. Narváez

Abstract

It is important to understand how the outcome of an election can be modified by an agent with control over the structure of the election. Electoral control has been studied for many election systems, but for all of those systems, the winner problem is in P, and so control is in NP. There are election systems, such as Kemeny, that have many desirable properties, but whose winner problems are not in NP. Thus for such systems control is not in NP, and in fact we show that it is typically complete for Σ_2^p (i.e., $\mathrm{NP}^{\mathrm{NP}}$, a class at the second level of the polynomial hierarchy). This is a very high level of complexity. What does this mean in practice? Approaches that perform quite well for solving NP problems do not necessarily work for Σ_2^p-complete problems. However, answer set programming is particularly suited to express problems in Σ_2^p, and we discuss how our control problems can be encoded.

1 Introduction

The study of elections often deals with trade-offs for different properties that an election system satisfies. Elections have a wide range of applications from voting in political elections to applications to artificial intelligence (see, e.g., [9]). And given the role of elections in multiagent system settings, it is important that we study the computational properties of election systems.

Attacks on the structure of an election, referred to as control, were introduced by Bartholdi, Tovey, and Trick [3] and model natural scenarios where an agent with control over the structure of an election, modifies the structure (e.g., by adding candidates) to ensure that their preferred candidate wins. It is important to study how these types of attacks on the structure of an election can affect the outcome and how computationally difficult it is to determine if such an attack exists.

The complexity of electoral control has been studied for many different natural elections systems (see, e.g., [19]). However, for all of those systems the winner problems are in P, and so the standard control problems are in NP.

There are election systems that have many desirable social-choice properties, but whose winner problems are not in NP (assuming NP $\neq$ coNP). One important example is the Kemeny rule [28], whose winner problem is Θ_2^p-complete (i.e., complete for parallel access to NP) [26] and so the complexity of the standard control problems for Kemeny are all Θ_2^p-hard and thus also not in NP. For election systems with Θ_2^p-complete winner problems, control is in Σ_2^p (i.e., $\mathrm{NP}^{\mathrm{NP}}$, a class at the second level of the polynomial hierarchy), and we show that control is typically Σ_2^p-complete for such systems.

This is a very high level of complexity. And so a natural question to ask is if Kemeny control can be solved in practice. We mention here that there has been a long line of work that considers ways to solve hard election problems in practice. This includes work on computing Kemeny winners in practice (see, e.g., [12, 5, 1, 4]) and on solving election-attack problems in practice (see, e.g., [33]). The work on solving hard election-attack problems has been restricted to problems in NP, and such approaches do not work for Σ_2^p-complete problems. Answer set programming (ASP) is an approach that has been recently applied for winner determination in voting, including for systems with hard winner problems [11]. ASP is particularly suited to express Σ_2^p problems. However, this requires the use of more

advanced techniques than encoding computationally easier problems.

We make the following main contributions.

- We obtain the first natural Σ_2^p-completeness results for elections.
- We define several new natural and simple Σ_2^p-complete graph problems to prove our results.
- We show for the most commonly-studied election systems with Θ_2^p-complete winner problems, including the Kemeny rule, that control is typically Σ_2^p-complete.
- We build upon recent work on combining ASP with voting [11], and show how ASP can be used to solve our control problems using a technique that is accessible to ASP nonexperts: We encode our control problems by combining meta-interpretation [18] and normalization [7] into a guess and check framework for Σ_2^p.

2 Preliminaries

An election consists of a set of candidates C, and a set of voters V, where each voter has a vote that strictly ranks each candidate from most to least preferred. An election system, $\mathcal{E}$, maps an election (C, V) to a set of winner(s), where the winner set can be any subset of the candidate set. The winner problem for an election system, $\mathcal{E}$-Winner, is defined in the following way. Given an election (C, V) and a candidate $p \in C$, is p a winner of the election using election system $\mathcal{E}$?

We consider the most-important election systems with Θ_2^p-complete winner problems: Kemeny, Young, and Dodgson.

A candidate is a Kemeny winner if it is the most-preferred candidate in a Kemeny consensus [28], which is a total order ">" that minimizes the sum of Kendall's Tau distances (i.e., number of pairwise disagreements) with the voters in an election, i.e., minimizes $\sum_{a,b \in C, a>b} \|\{v \in V \mid b >_v a\}\|$, where $>_v$ denotes the preference of voter v.

A candidate is a Young winner if it can become a weak Condorcet winner (a candidate that beats-or-ties every other candidate pairwise) by deleting the fewest voters.

A candidate is a Dodgson winner if it can become a Condorcet winner (a candidate that beats every other candidate pairwise) with the fewest swaps between adjacent candidates in the voters' rankings [14].

Electoral control models the actions of an agent, referred to as the election chair, who modifies the structure of the election (e.g., the voters) to ensure that their preferred candidate wins (in the constructive case) [3] or that their despised candidate does not win (in the destructive case) [25].[1] We formally define constructive control by adding candidates (CCAC) below, which models the real-world scenario of an election chair adding spoiler candidates to an election to ensure that their preferred candidate wins.

Name: $\mathcal{E}$-CCAC

Given: A set of registered candidates C, a set of unregistered candidates D, a set of voters V having preferences over $C \cup D$, an addition limit k, and a preferred candidate $p \in C$.

Question: Does there exist a set $D' \subseteq D$ such that $\|D'\| \leq k$ and p is a winner of $(C \cup D', V)$ using election system $\mathcal{E}$?

[1] We mention here that early work that considered electoral control generally used the unique winner model where the goal of the chair is to ensure that their preferred (despised) candidate is the only (not the only) winner.

	Adding	Deleting
Voters	Young (Thm 10) Kemeny′ (Thm 7)	Young (Thm 9)
Candidates	Kemeny (Thm 6) Dodgson (Thm 11)	Kemeny (∗) (Thm 5) Dodgson (Thm 11)

Table 1: Summary of our Σ_2^p-completeness results for control. Kemeny′ refers to a natural variant of Kemeny defined in [15] and (∗) refers to the variant of control where the chair can delete only certain candidates.

Computational Complexity Our results concern the complexity classes Θ_2^p and Σ_2^p. Θ_2^p is the class of problems that can be solved by a polynomial-time machine with parallel access to an NP oracle, and $\Sigma_2^p = \mathrm{NP}^{\mathrm{NP}}$ is the class of problems solvable by a nondeterministic polynomial-time machine with access to an NP oracle, and is a class at the second level of the polynomial hierarchy (see, e.g., [32]).

Note that $\mathrm{NP} \cup \mathrm{coNP} \subseteq \Theta_2^p \subseteq \mathrm{P}^{\mathrm{NP}} \subseteq \Sigma_2^p$.

3 Complexity Results

In this section we show that control problems for Kemeny, Young, and Dodgson elections are typically Σ_2^p-complete.

Observation 1 *For an election system $\mathcal{E}$, the complexity of each standard control action is in* $\mathrm{NP}^{\mathcal{E}\text{-Winner}}$.

It is easy to see that the above observation holds. For a given election, guess the control action of the chair (e.g., the set of candidates to be added) and then use the oracle to check that the preferred candidate is a winner (or that the despised candidate is not a winner). In the case of partition control, which will not be discussed further in this paper, the oracle will also be used to determine which candidates participate in the runoff.

The winner problems for Kemeny, Young, and Dodgson are each in Θ_2^p, and so the complexity of each standard control action is in $\mathrm{NP}^{\Theta_2^p}$, which is equivalent to Σ_2^p.

Corollary 2 *For Kemeny, Young, and Dodgson elections, the complexity of each standard control action is in Σ_2^p.*

As mentioned in [8], these control problems inherit Θ_2^p-hardness from their winner problems.

We will now show that these control problems are typically Σ_2^p-complete. Our Σ_2^p-completeness results are summarized in Table 1 and we conjecture that for Kemeny, Young, and Dodgson elections, the complexity of each standard control action is Σ_2^p-complete.[2]

We mention here that there are far fewer completeness results for Σ_2^p than there are for NP (see Schaefer and Umans [36, 37] for a list of completeness results in the polynomial hierarchy). An important reason why proving Σ_2^p-hardness is difficult is the scarcity of "simple" Σ_2^p-complete problems to reduce from. For example, scoring versions of Kemeny, Young, and Dodgson are proven NP-hard by reductions from simple NP-complete problems such as Vertex-Cover, but prior to this paper there were no Σ_2^p-complete simple versions

[2]de Haan [13] shows Σ_2^p-hardness for control by adding/deleting issues for an analogue of Kemeny for judgment aggregation. Since our setting is much more restrictive, this lower bound does not at all imply our lower bound.

of Vertex-Cover that we could use to show that related control-by-adding problems are Σ_2^p-hard.

Below we define simple and natural Σ_2^p-complete versions of Vertex-Cover (and the analogous Independent-Set versions are also Σ_2^p-complete). We will see that these problems are particularly useful to show that our control problems are Σ_2^p-hard. Of course, we need to show that our new simple problems are Σ_2^p-hard, which is difficult. But we can then reuse our simple problems to obtain simpler Σ_2^p-hard proofs for multiple other problems.

The following problem (and its closely related Independent-Set analogue) is particularly useful to reduce to control-by-adding problems. For example, we will see that this problem quite easily reduces to Kemeny-CCAC and that the Independent-Set analogue of this problem reduces quite easily to Young-CCAV (control by adding voters).

Name: Vertex-Cover-Member-Add

Given: Graph $G = (V \cup V', E)$, set of addable vertices V', addition limit k, and vertex $\hat{v} \in V$.

Question: Does there exist a set $W \subseteq V'$ of at most k addable vertices such that $\hat{v}$ is a member of a minimum vertex cover[3] of $(V \cup W, E)$?

Theorem 3 *Vertex-Cover-Member-Add is Σ_2^p-complete.*

To show the essence of the argument of the proof of Theorem 3 and avoid some of the more finicky details of the proof, we prove that Vertex-Cover-Member-Select is Σ_2^p-complete, and then briefly discuss how this proof can be adapted for Vertex-Cover-Member-Add.[4]

Name: Vertex-Cover-Member-Select

Given: Graph $G = (V, E)$, a set $V' \subseteq V$ of deletable vertices, delete limit k, and vertex $\hat{v} \in V$.

Question: Does there exist a set $W \subseteq V'$ of at most k deletable vertices such that $\hat{v}$ is a member of a minimum vertex cover of $G - W$?

Lemma 4 *Vertex-Cover-Member-Select is Σ_2^p-complete.*

Proof. Membership in Σ_2^p is easy to see: Guess at most k deletable vertices to delete, then guess a vertex cover containing $\hat{v}$ and use the NP oracle to check that the guessed vertex cover is a minimum vertex cover.

To show hardness, we reduce from the following Σ_2^p-complete problem, QSAT_2 [38, 39]: all true formulas of the form $\exists x_1 \cdots \exists x_n \neg(\exists y_1 \cdots \exists y_n \phi(x_1, \ldots, x_n, y_1, \ldots, y_n))$, where ϕ is a formula in 3cnf.[5]

We first recall the standard reduction from 3SAT to Vertex-Cover [27]. Let G be the graph constructed by this reduction on $\phi(x_1, \ldots, x_n, y_1, \ldots, y_n)$, where ϕ is in 3cnf. Let

[3] A vertex cover of a graph is a set of vertices such that every edge is incident with at least one vertex in the set.

[4] We can easily prove Vertex-Cover-Member-Select Σ_2^p-hard by reducing the Σ_2^p-complete problem Generalized-Node-Deletion to it, in which we are given a graph, and two integers k and ℓ, and we ask if we can delete at most k vertices such that the remaining graph does not contain a clique of size $\ell + 1$ [35]. For the reduction, simply output $(H, V(H), k, \hat{v})$, where H is the graph $(\overline{G} \cup (\{\hat{v}\}, \emptyset)) + \overline{K_{\ell+1}}$.

However, this proof does not generalize to Vertex-Cover-Member-Add. In particular, the "adding" analogue of Generalized-Node-Deletion in which we ask if we can *add* up to k vertices such that the resulting graph does not have a clique of size $\ell + 1$, is in P (since it is best to add nothing). And the version where we ask if there is a clique of size $\ell + 1$ after adding is in NP.

[5] Note that we have the same number of variables in each quantified block (can simply pad to get this). Also, we pull the negation out of the formula so that the formula is in 3cnf and not in 3dnf.

$\phi = \psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_m$ and for each $i, 1 \leq i \leq m$, $\psi_i = c_{i,1} \vee c_{i,2} \vee c_{i,3}$. Graph G consists of $4n + 3m$ vertices: a vertex for each x_i, $\overline{x_i}$, y_i, and $\overline{y_i}$ and for each clause $i, 1 \leq i \leq m$, three vertices $c_{i,1}$, $c_{i,2}$, and $c_{i,3}$, and the following edges:

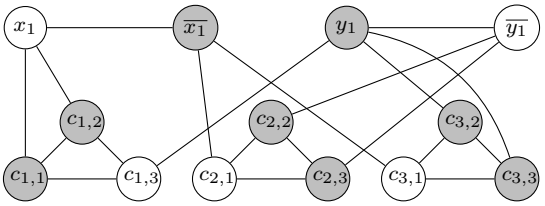
- for each $i, 1 \leq i \leq n$, the edges $\{x_i, \overline{x_i}\}$ and $\{y_i, \overline{y_i}\}$,
- for each $i, 1 \leq i \leq m$, the edges $\{c_{i,1}, c_{i,2}\}, \{c_{i,1}, c_{i,3}\}$, and $\{c_{i,2}, c_{i,3}\}$, (i.e., the complete graph on three vertices),
- and for each vertex $c_{i,j}$ we have an edge to its corresponding vertex candidate (e.g., if $c_{i,j} = x_t$ in ϕ then we have the edge $\{c_{i,j}, x_t\}$).

Below we give the graph corresponding to the formula $\phi = (x_1 \vee x_1 \vee y_1) \wedge (\overline{x_1} \vee \overline{y_1} \vee \overline{y_1}) \wedge (\overline{x_1} \vee y_1 \vee y_1)$.

Note that every vertex cover of G contains at least one of each $\{x_i, \overline{x_i}\}$, at least one of each $\{y_i, \overline{y_i}\}$, and at least two of each $\{c_{i,1}, c_{i,2}, c_{i,3}\}$, so G does not have a vertex cover of size less than $2n + 2m$. The properties below follow immediately from the proof from [27].

1. If X is a vertex cover of size $2n + 2m$, then $X \cap \{x_i, \overline{x_i}, y_i, \overline{y_i} \mid 1 \leq i \leq n\}$ corresponds to a satisfying assignment for ϕ.
2. If α is a satisfying assignment for ϕ, then there is a vertex cover X of size $2n + 2m$ such that $X \cap \{x_i, \overline{x_i}, y_i, \overline{y_i} \mid 1 \leq i \leq n\}$ corresponds to this assignment.

Below we include an example of this construction given the formula $\phi = (x_1 \vee x_1 \vee y_1) \wedge (\overline{x_1} \vee \overline{y_1} \vee \overline{y_1}) \wedge (\overline{x_1} \vee y_1 \vee y_1)$.



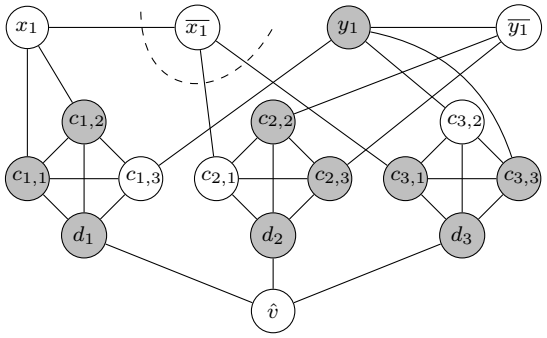
In the figure above vertices that are in a minimum vertex cover are shaded in gray, and this corresponds to the satisfying assignment $x_1 = 0$, $y_1 = 1$ for ϕ.

For the reduction from QSAT_2 to Vertex-Cover-Member-Select, we construct the graph H, which is a modified version of the graph G. For each clause $i, 1 \leq i \leq m$, instead of the complete graph on three vertices, $\{c_{i,1}, c_{i,2}, c_{i,3}\}$, we add an extra vertex d_i and have the complete graph on four vertices, $\{c_{i,1}, c_{i,2}, c_{i,3}, d_i\}$, and we connect the fourth vertex d_i of each clause gadget to a special new vertex $\hat{v}$. So our graph H consists of $4n + 4m + 1$ vertices and the edges as just described. Below we give the graph corresponding to the same formula as the previous example.

Note that every vertex cover of H contains at least one of each $\{x_i, \overline{x_i}\}$, at least one of each $\{y_i, \overline{y_i}\}$, and at least three of each $\{c_{i,1}, c_{i,2}, c_{i,3}, d_i\}$. So H does not have a vertex cover of size less than $2n + 3m$, and there is a vertex cover of size $2n + 3m + 1$ that includes $\hat{v}$. Note that H has the following properties.

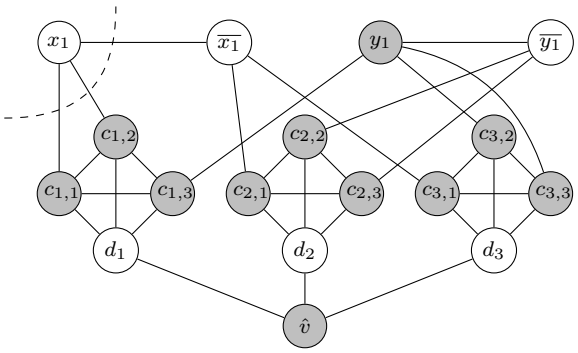
1. If X is a vertex cover of size $2n + 3m$, then $\hat{v} \notin X$ and $X \cap \{x_i, \overline{x_i}, y_i, \overline{y_i} \mid 1 \leq i \leq n\}$ corresponds to a satisfying assignment for ϕ.
2. If α is a satisfying assignment for ϕ, then there is a vertex cover of size $2n + 3m$ such that $X \cap \{x_i, \overline{x_i}, y_i, \overline{y_i} \mid 1 \leq i \leq n\}$ corresponds to this assignment.

Below we include an example of this construction given the formula $\exists x_1 \neg(\exists y_1 \phi(x_1, y_1))$, where $\phi = (x_1 \vee x_1 \vee y_1) \wedge (\overline{x_1} \vee \overline{y_1} \vee \overline{y_1}) \wedge (\overline{x_1} \vee y_1 \vee y_1)$.



Note that in the figure when $\overline{x_1}$ is removed (i.e., setting $x_1 = 0$) that a minimum-size vertex cover (shaded in gray) is of size $n + 3m = 10$ and that $\phi(0, y_1)$ is satisfied with $y_1 = 1$.

We have repeated the same graph below, except now the vertex x_1 is removed (i.e., setting $x_1 = 1$).



The vertices shaded in gray above correspond to a minimum-size vertex cover of size $n + 3m + 1 = 11$. Note that $\phi(1, y_1)$ is not satisfiable and that this vertex cover includes $\hat{v}$.

We will show that $\exists x_1 \cdots \exists x_n \neg(\exists y_1 \cdots \exists y_n \phi(x_1, \ldots, x_n, y_1, \ldots, y_n))$ if and only if we can delete at most n vertices in $\{x_i, \overline{x_i} \mid 1 \leq i \leq n\}$ such that $\hat{v}$ is a member of a minimum vertex cover of H-after-deletion.

From the listed properties of H and the above example, it is not too hard to see that the statement above holds as long as the vertices deleted from H correspond to an assignment to the x-variables. However, it is possible for the set of deleted vertices to contain neither or both of $\{x_i, \overline{x_i}\}$. We handle these cases below.

Suppose that W is a set of at most n vertices from $\{x_i, \overline{x_i} \mid 1 \leq i \leq n\}$ such that $\hat{v}$ is a member of a minimum vertex cover of $H - W$. Let $\widehat{n} \leq n$ be the number of $\{x_i, \overline{x_i}\}$ pairs that are undeleted, i.e., for which $\{x_i, \overline{x_i}\} \cap W = \emptyset$. Note that the size of a vertex cover of $H - W$ is at least $\widehat{n} + n + 3m$, and that vertex covers of that size do not include $\hat{v}$. Since $\hat{v}$ is a member of a minimum size vertex cover of $H - W$, it follows that $H - W$ does not have a vertex cover of size $\widehat{n} + n + 3m$.

Let $\alpha \in \{0, 1\}^n$ be an assignment to $x_1 \cdots x_n$ that is consistent with W, in the sense that for all $i, 1 \leq i \leq n$, if $W \cap \{x_i, \overline{x_i}\} = \{x_i\}$, then $\alpha_i = 1$ and if $W \cap \{x_i, \overline{x_i}\} = \{\overline{x_i}\}$, then $\alpha_i = 0$. To complete the proof, suppose for a contradiction that $\phi(\alpha_1, \ldots, \alpha_n, y_1, \ldots, y_n)$ is satisfiable. Then H has a vertex cover X of size $2n + 3m$ such that $X \cap \{x_i, \overline{x_i} \mid 1 \leq i \leq n\}$ corresponds to α. Note that $X - W$ is a vertex cover of $H - W$. But $\|X \cap W\| = n - \widehat{n}$, and so $\|X - W\| = 2n + 3m - (n - \widehat{n}) = \widehat{n} + n + 3m$, which is a contradiction. ❑

Vertex-Cover-Member-Select reduces to the corresponding Kemeny control problem Kemeny-CCDC*.

Name: $\mathcal{E}$-CCDC*

Given: An election (C, V), a set of deletable candidates $D \subseteq C$, a delete limit k, and a preferred candidate $p \in C$.

Question: Does there exist a set $D' \subseteq D$ of at most k deletable candidates such that p is a winner of $(C - D', V)$ using election system $\mathcal{E}$?

Note that $\mathcal{E}$-CCDC* is more structured than $\mathcal{E}$-CCDC (where the set of deletable candidates is C), since the chair can only delete from a subset of the candidate set.

Theorem 5 *Kemeny-CCDC* is Σ_2^p-complete.*[6]

Proof Sketch. Kemeny-Score was shown to be NP-hard by a reduction from Feedback-Arc-Set[7] by Bartholdi, Tovey, and Trick [2], which was shown to be NP-hard by Karp [27] by a reduction from Vertex-Cover. Both these reductions are straightforward. To show the Θ_2^p-hardness of Kemeny-Winner, Hemaspaandra, Spakowski, and Vogel [26] define Θ_2^p-complete versions of Vertex-Cover and Feedback-Arc-Set, namely Vertex-Cover-Member and Feedback-Arc-Set-Member. They show that Vertex-Cover-Member is Θ_2^p-complete. They then show that Vertex-Cover-Member reduces to Feedback-Arc-Set-Member, which then reduces to Kemeny-Winner. These two reductions are similar to the NP reductions and also straightforward. The same happens in our Σ_2^p case: Vertex-Cover-Member-Select easily and straightforwardly reduces to Feedback-Arc-Set-Member-Select, which easily and straightforwardly reduces to Kemeny-CCDC*. For details, see Appendix A.1. ❑

Vertex-Cover-Member-Add easily and similarly reduces to Feedback-Arc-Set-Member-Add (defined in Appendix A.2) to Kemeny-CCAC. To show that Vertex-Cover-Member-Add is Σ_2^p-hard, we use a similar reduction as for Vertex-Cover-Member-Select. The main difference is that we need an edge and two vertices for each x_i and for each $\overline{x_i}$. See Appendix A.2 for the proof.

Theorem 6 *Kemeny-CCAC is Σ_2^p-complete.*

What about Kemeny-CCDC? We define the following.

Name: Vertex-Cover-Member-Delete

Given: Graph $G = (V, E)$, delete limit k, and vertex $\hat{v} \in V$.

Question: Does there exist a set $W \subseteq V$ of at most k vertices such that $\hat{v}$ is a member of a minimum vertex cover of $G - W$?

This problem is also Σ_2^p-complete. (The lower bound follows using the same reduction as in Footnote 4.) Unfortunately, the reduction from Vertex-Cover to Feedback-Arc-Set does not easily turn into a reduction from Vertex-Cover-Member-Delete to Feedback-Arc-Set-Member-Delete. The problem is that the reduction from Vertex-Cover to Feedback-Arc-Set turns every vertex v into a pair of vertices $v \to v'$. So, one special vertex $\hat{v}$ corresponds to two special vertices $\hat{v} \to \hat{v}'$. And the problem is that we can not ensure that both special

[6] We mention in passing that this result holds even for four voters, using the construction from Dwork et al. [15]. For details, see Appendix A.1.

[7] A feedback arc set of a directed graph is a set of arcs such that deleting this set makes the graph acyclic.

vertices are not deleted. (This is in fact the only problem; the version of Kemeny-CCDC where there are two special candidates that we can not delete is Σ_2^p-complete.) A similar issue occurs when we try to show that Kemeny-CCAV is Σ_2^p-complete. It is easy to show that Feedback-Arc-Set-Member-Add-Arcs, where we add arcs instead of vertices, is Σ_2^p-complete. The problem is that in the reduction from Feedback-Arc-Set to Kemeny-Score, each arc corresponds to two voters. However, we were able to show this result for what we here call Kemeny$'$, the natural variant of Kemeny from [15] where the voters do not necessarily list all of the candidates in their votes and unlisted candidates in a vote do not contribute to the distance to the Kemeny consensus and so do not increase the Kemeny score. In this case, one arc will correspond to one voter.

Theorem 7 *Kemeny$'$-CCAV is Σ_2^p-complete.*

Finally, we explain how Independent-Set-Member-Delete, the Independent-Set analogue of Vertex-Cover-Member-Delete, which is also Σ_2^p-complete, is useful to show that Young-CCDV is Σ_2^p-complete.

Name: Independent-Set-Member-Delete

Given: Graph $G = (V, E)$, delete limit k, and vertex $\hat{v} \in V$.

Question: Does there exist a set $W \subseteq V$ such that $\|W\| \leq k$ and $\hat{v}$ is a member of a maximum independent set of $G - W$?

Theorem 8 *Independent-Set-Member-Delete is Σ_2^p-complete.*

Proof. Independent-Set-Member-Delete is clearly in Σ_2^p. We will show that Independent-Set-Member-Delete is Σ_2^p-complete by reducing the Σ_2^p-complete problem Generalized-Node-Deletion to it, in which we are given a graph, and two integers k and ℓ, and we ask if we can delete at most k vertices such that the remaining graph does not contain a clique of size $\ell + 1$ [35]. For the reduction, simply output $(H, k, \hat{v})$, where H is the graph $\overline{G} + \overline{K_\ell}$, and $\hat{v}$ is a vertex in $\overline{K_\ell}$.

If we can delete at most k vertices in G such that the remaining graph does not contain a clique of size $\ell + 1$, then $\overline{G}$ after deletion does not contain an independent set of size $\ell + 1$. It follows that after deletion, $\overline{K_\ell}$ is a largest independent set in H and $\hat{v}$ is an element of this independent set.

For the converse, suppose we delete at most k vertices from H such that $\hat{v}$ is a member of a largest independent set. This independent set has size at most ℓ, and so after deletion, $\overline{G}$ does not have an independent set of size $\ell + 1$. ❑

Theorem 9 *Young-CCDV is Σ_2^p-complete.*

Proof. For G a graph, $\alpha(G)$ is the independence number of G, i.e., the size of a maximum independent set of G. For v a vertex, $\alpha_v(G)$ is the size of a maximum independent set of G that contains v.

Given a graph $G = (V, E)$, a delete limit $k \leq \|V\|$, and a vertex $\hat{v} \in V$, below we construct an election with two special candidates p and q such that our instance is a positive instance of Independent-Set-Member-Delete if and only if we can make the Young score[8] of p at least 2 and at least as high as the Young score of q by deleting at most k voters.

[8]The Young score of a candidate is the size of a largest subset of the voters that make it a weak Condorcet winner. A Young winner is a candidate with highest Young score.

This is not quite the same as making p a Young winner, since other candidates could have higher scores and since it is also possible for a Young winner to have a score below 2. However, this can easily be handled as follows: We can use the trick from [34] to change the election such that the Young scores of p and q remain the same, the Young scores of all other candidates are at most 2, and such that there is a candidate with a Young score of at least 2.

We use candidate q to witness the size of a maximum independent set. The main idea on how to do this is implicit in the construction from Rothe, Spakowski, and Vogel [34]. The new twist is to use candidate p to witness the size of a maximum independent set containing $\hat{v}$. In order to do so, we make sure that all voters corresponding to vertices connected to $\hat{v}$ are not in a set of voters that realizes the Young score of p, by making these voters maximally unattractive to p, by ranking p last.

Given a graph $G = (V, E)$, a delete limit $k \leq \|V\|$, and a vertex $\hat{v} \in V$, without loss of generality assume that G has no isolated vertices and that for every set W of at most k vertices, $\alpha(G - W) \geq 3$ (the latter property can for example be ensured by adding two $(k+1)$-cliques to G). Let the candidate set be $E \cup \{a, p, q\}$ and let the voter set consist of:

Type IA For each $v \in V$ such that $\{v, \hat{v}\} \notin E$, one voter corresponding to v voting $(\{e \in E \mid v \in e\} > a > q > p > \cdots)$.

Type IB For each $v \in V$ such that $\{v, \hat{v}\} \in E$, one voter corresponding to v voting $(\{e \in E \mid v \in e\} > a > q > \cdots > p)$.

Type II One voter voting $(p > q > \cdots > a)$.

Type III $2\|V\|$ voters voting $(\cdots > p > q > a)$.

Suppose X is a set of at most k voters that we delete. If X contains the voter of Type II, the Young scores of p and q are 0 (since there are no isolated vertices). If not, let W be the set of of vertices corresponding to the Type-I voters in X. Careful and tedious inspection shows that the Young score of q is $2\alpha(G-W)$ (this Young score is realized by a set of voters whose Type-I voters correspond to a maximum independent set) and that the Young score of p is $2\alpha_{\hat{v}}(G - W)$ (this Young score is realized by a set of voters whose Type-I voters are all of Type-IA and correspond to a maximum independent set that contains $\hat{v}$). For details, see the appendix. ❑

The same construction gives a reduction from Independent-Set-Member-Add to Young-CCAV.

Theorem 10 *Young-CCAV is Σ_2^p-complete.*

Previous complexity results for Dodgson elections do not reduce from problems related to Vertex-Cover or Independent-Set. However, in Dodgson there is more flexibility in how to construct the voters in a reduction, and so we were able to directly reduce QSAT_2 to Dodgson-CCDC and Dodgson-CCAC, though the constructions are quite involved. The proofs can be found in the appendix.

Theorem 11 *Dodgson-CCDC and CCAC are Σ_2^p-complete.*

4 Solving Control Problems with ASP

Answer set programming (ASP) is a paradigm for encoding computationally difficult problems in a declarative way (see [10]). Using modern ASP input languages like the one in

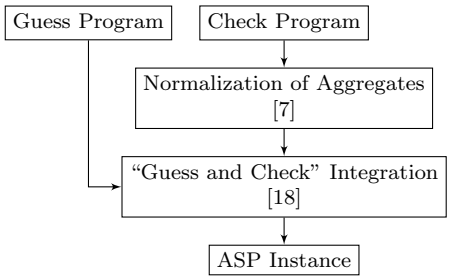


Figure 1: Illustration of our "guess and check" approach.

the `gringo` grounder [22], which extends conventional ASP with aggregate functions like `#sum` and `#count`, one can use variable names and predicates. A descriptive naming scheme usually leads to natural encodings of problems when compared to other approaches such as encoding into Boolean satisfiability problems.

The idea of using ASP to solve computational problems in voting was first proposed by Konczak [29]. Recent work by Charwat and Pfandler [11] provides winner-problem encodings for many election systems, including systems with hard winner problems, and mentions encoding control problems as future work. The predicates used in their encodings are arguably self-explanatory and the use of aggregates provides succinct representations of the different voting rules they consider.

In this section we address the issue of encoding the control problems we described in previous sections, using ASP. Encodings for NP problems are fairly straightforward and common in the literature. However, though it is known that ASP can encode problems in Σ_2^p [16], doing so often requires advanced techniques that may not be suitable for nonexperts (see, e.g., the discussion by Eiter and Polleres [18]). The work of Eiter and Polleres [18] addresses the issue of having to craft cumbersome ASP encodings of Σ_2^p problems in the following way: They provide a template "meta-interpreter" and a transformation of ASP programs that can, together, be used to integrate an ASP program that guesses a solution to the Σ_2^p problem with an ASP program that checks whether the guessed solution is incorrect. The combination of these two then amounts to a "guess and check" encoding of a Σ_2^p problem.[9] While this is a significant step towards simplifying the encoding of Σ_2^p problems as ASP programs, for our specific application it has the drawback that it does not support ASP programs extended with aggregates (e.g., `#count`). This is problematic for our ASP encodings since not using aggregates would lead to more complex, less intuitive programs.

To work around this issue, we need a way to transform ASP programs with aggregates into equivalent ASP programs that do not employ aggregates. The `lp2normal` tool [7] provides such a transformation. We can combine `lp2normal` with meta-interpretation to express control problems with elections while harnessing the full expressive power of modern ASP input languages. Figure 1 illustrates the interaction between the several parts of this approach. We note that the transformation to eliminate aggregates is only needed for the check program, since the integration between the guess and the check programs only transforms the latter.

4.1 Preliminaries on Answer Set Programming

We briefly state our definitions for ASP. (See Gebser et al. [21] for more detailed definitions.) A *normal logic program* is comprised of a finite set of rules of the form

[9]Note that this is different and much more involved than the guess and check encoding of an NP problem (see [17]).

$a \leftarrow b_1, \ldots, b_m$, not $b_{m+1}, \ldots,$ not b_n where each of $a, b_1, \ldots, b_n$ are atoms and each atom is a predicate of the form $p(t_1, \ldots, t_k)$ such that $k \geq 1$ and each t_i is a constant or a variable. We indicate that p is a k-ary predicate by writing p/k. In the rule, "," refers to conjunction, and "not" refers to default negation. A rule is satisfied (i.e., the head of a rule is true), if the body is true. Uppercase characters are used to denote variables. A *ground* program is a program that contains no variables. A *stable model* (answer set) is a subset of the ground atoms that satisfies each rule.

Our encodings also utilize the following extensions to the above syntax, but each can be expressed as a normal rule (see [21]). A *fact* is a rule with no body and an *integrity constraint* is a rule with no head. So, a fact occurs in every answer set, and an integrity constraint eliminates answer sets where its body is satisfied. We additionally use choice rules with cardinality constraints, which can be used to generate subsets of ground atoms within a given bound, and aggregates such as `#count` and `#sum` that count/sum ground atoms in a statement.

4.2 Encoding Control in ASP

We assume that the input to our encodings is given as a list of facts. So for $\mathcal{E}$-CCAC, our input consists of a fact for each registered candidate, each unregistered candidate, the addition limit, the preferred candidate, and facts that describe the voters. For the voters, we follow the approach used in Democratix [11], where each distinct vote $(c_1 >_i > \cdots >_i c_m)$ is represented by m atoms of the form $\text{ip}(i, j, c)$ meaning candidate c is the jth-preferred candidate by vote i. The corresponding count is represented by $\text{voteCount}(i, k)$, meaning k voters have vote i.

We illustrate the encoding of the control problems we discussed in this paper by showing guess and check programs for Kemeny-CCAC. Due to space limitations, we focus on the crucial parts of each program. Appendix D contains full listings of these programs together with more detailed explanations. Figure 2 shows the relevant parts of the guess program which assumes an input as described in Democratix [11] but extended with predicates that define the parameters of the control problem. We start by guessing a subset of at most K of the unregistered candidates (marked by the ucandidate/1 input predicate) to add to the election, where K is the addition limit given as part of the input. Next, a "guess" preference gpref/2 over the candidates (with $\text{gpref}(i, c)$ meaning candidate c is the ith-preferred candidate) is guessed as per the rules in Democratix. The Kendall's Tau distance of this guessed preference with respect to the votes is calculated in Line 2 and finally, Line 3 guarantees that the preferred candidate is a winner in the guessed preference. Figure 3 shows the relevant parts of the checking part. It is overall very similar to the guess part, except that it guesses a "check" preference cpref/2 that has a Kendall's Tau distance strictly less than that of gpref/2 (Line 4) (it needs to be strictly less because we are working under the nonunique-winner model). Line 5 guarantees that the preferred candidate does not win in this preference. Combining the guess and check programs using the approach outlined in the previous section we obtain an ASP program that is satisfiable if and only if there is a subset of unregistered candidates that can be added such that the preferred candidate wins the election.

4.3 Similar Encoding Approaches

The framework of *stable-unstable* semantics by Bogaerts et al. [6] provides a similar "guess and check" strategy to solving problems in Σ_2^p. They mention that an advantage of the stable-unstable semantics is that they can be easily extended to represent problems at any level of the polynomial hierarchy. In their implementation both the guess and the check

$$\{\text{candidate}(C) : \text{ucandidate}(C)\}\ K \leftarrow \text{limit}(K). \quad (1)$$

% A preference gpref/2 and a "worse" rank gwrankC/3 are guessed as in Democratix [11]

$$\text{gkt}(K) \leftarrow K = \#sum\{N, C_1, C_2 : \text{gwrankC}(C_1, C_2, N)\}. \quad (2)$$

$$\leftarrow \text{preferredCand}(C), not\ \text{gpref}(1, C). \quad (3)$$

Figure 2: Relevant parts of the encoding of the guess part of Kemeny-CCAC.

% A new preference cpref/2 and corresponding rank cwrankC/3 are guessed as in Figure 2

$$\leftarrow \text{gkt}(K), K\ \#sum\{N, C_1, C_2 : \text{cwrankC}(C_2, C_1, N)\} \quad (4)$$

$$\leftarrow \text{preferredCand}(X), \text{cpref}(1, X). \quad (5)$$

Figure 3: Relevant parts of the encoding of the check part of Kemeny-CCAC.

program are normalized, essentially discarding aggregates altogether. It is an interesting direction for future work to determine if not exploiting aggregates as implemented in modern ASP solvers like `clasp` [23] could hurt the performance of solvers employed in the task of solving very hard control problems in election systems.

5 Future Work

In addition to the future work on ASP described at the end of the previous section, it will be interesting to see if our newly-defined simple Σ_2^p-complete problems will be useful in proving other problems Σ_2^p-hard, in particular the remaining control cases and other election-attack problem such as manipulation for systems with hard winner problems.

Acknowledgments: We thank the referees for their helpful comments and suggestions. This work was supported in part by a National Science Foundation Graduate Research Fellowship under NSF grant no. DGE-1102937.

References

[1] A. Ali and M. Meilă. Experiments with Kemeny ranking: What works when? *Mathematical Social Sciences*, 64(1):28–40, 2012.

[2] J. Bartholdi, III, C. Tovey, and M. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6(2):157–165, 1989.

[3] 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.

[4] N. Betzler, R. Bredereck, and R. Niedermeier. Theoretical and empirical evaluation of data reduction for exact Kemeny Rank Aggregation. *Autonomous Agents and Multi-Agent Systems*, 28(5):721–748, 2014.

[5] N. Betzler, M. Fellows, J. Guo, R. Niedermeier, and F. Rosamond. Fixed-parameter algorithms for Kemeny scores. In *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management*, pages 60–71, June 2008.

[6] B. Bogaerts, T. Janhunen, and S. Tasharrofi. Stable-unstable semantics: Beyond NP with normal logic programs. *Theory and Practice of Logic Programming*, 16(5-6):570–586, 2016.

[7] J. Bomanson, M. Gebser, and T. Janhunen. Improving the normalization of weight rules in answer set programs. In *Proceedings of the 12th European Conference on Logics in Artificial Intelligence*, pages 166–180, Sept. 2014.

[8] F. Brandt, M. Brill, E. Hemaspaandra, and L. Hemaspaandra. Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. *Journal of Artificial Intelligence Research*, 53:439–496, 2015.

[9] F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. Procaccia. *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

[10] G. Brewka, T. Eiter, and M. Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[11] G. Charwat and A. Pfandler. Democratix: A declarative approach to winner determination. In *Proceedings of the 4th International Conference on Algorithmic Decision Theory*, pages 253–269, Sept. 2015.

[12] V. Conitzer, A. Davenport, and J. Kalagnanam. Improved bounds for computing Kemeny rankings. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 620–626, July 2006.

[13] R. de Haan. Complexity results for manipulation, bribery and control of the kemeny judgment aggregation procedure. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems*, pages 1151–1159, May 2017.

[14] C. Dodgson. A method of taking votes on more than two issues. Pamphlet printed by the Clarendon Press, Oxford, and headed "not yet published", 1876.

[15] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th International World Wide Web Conference*, pages 613–622, Mar. 2001.

[16] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive datalog. *ACM Transactions on Database Systems*, 22(3):364–418, 1997.

[17] T. Eiter, G. Ianni, and T. Krennwallner. Answer set programming: A primer. In *Reasoning Web. Semantic Technologies for Information Systems*, pages 40–110, Aug/Sep 2009.

[18] T. Eiter and A. Polleres. Towards automated integration of guess and check programs in answer set programming: A meta-interpreter and applications. *Theory and Practice of Logic Programming*, 6(1-2):23–60, Jan. 2006.

[19] P. Faliszewski and J. Rothe. Control and bribery in voting. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. Procaccia, editors, *Handbook of Computational Social Choice*, pages 146–168. Cambridge University Press, 2016.

[20] Z. Fitzsimmons and E. Hemaspaandra. High-multiplicity election problems. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*, July 2018. To appear.

[21] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[22] M. Gebser, R. Kaminski, M. Ostrowski, T. Schaub, and S. Thiele. On the input language of ASP grounder gringo. In *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 502–508, Sept. 2009.

[23] M. Gebser, B. Kaufmann, and T. Schaub. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence*, 187-188:52–89, 2012.

[24] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *Journal of the ACM*, 44(6):806–825, 1997.

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

[26] E. Hemaspaandra, H. Spakowski, and J. Vogel. The complexity of Kemeny elections. *Theoretical Computer Science*, 349(3):382–391, 2005.

[27] R. Karp. Reducibilities among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103, 1972.

[28] J. Kemeny. Mathematics without numbers. *Daedalus*, 88:577–591, 1959.

[29] K. Konczak. Voting theory in answer set programming. In *Proceedings of the 20th Workshop on Logic Programming*, pages 45–53, 2006.

[30] D. McGarvey. A theorem on the construction of voting paradoxes. *Econometrica*, 21(4):608–610, 1953.

[31] A. McLoughlin. The complexity of computing the covering radius of a code. *IEEE Transactions on Information Theory*, 30:800–804, 1984.

[32] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[33] J. Rothe and L. Schend. Challenges to complexity shields that are supposed to protect elections against manipulation and control: A survey. *Annals of Mathematics and Artificial Intelligence*, 68(1–3):161–193, 2013.

[34] J. Rothe, H. Spakowski, and J. Vogel. Exact complexity of the winner problem for Young elections. *Theory of Computing Systems*, 36(4):375–386, 2003.

[35] V. Rutenburg. Propositional truth maintenance systems: Classification and complexity analysis. *Annals of Mathematics and Artificial Intelligence*, 10(3):207–231, 1994.

[36] M. Schaefer and C. Umans. Completeness in the polynomial-time hierarchy: Part I: A compendium. *SIGACT News*, 33(3):32–49, 2002.

[37] M. Schaefer and C. Umans. Completeness in the polynomial-time hierarchy: Part II. *SIGACT News*, 33(4), 2002.

[38] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[39] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):23–33, 1976.

Zack Fitzsimmons
Dept. of Mathematics and Computer Science
College of the Holy Cross
Worcester, MA 01610
Email: zfitzsim@holycross.edu

Edith Hemaspaandra
Dept. of Computer Science
Rochester Institute of Technology
Rochester, NY 14623
Email: eh@cs.rit.edu

Alexander Hoover
Dept. of Computer Science
Rochester Institute of Technology
Rochester, NY 14623
Email: ash4519@rit.edu

David E. Narváez
College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14623
Email: den9562@rit.edu

A Kemeny

A.1 Proof of Theorem 5: Kemeny-CCDC* is Σ_2^p-complete

Kemeny-Score was shown to be NP-hard by a reduction from Feedback-Arc-Set by Bartholdi, Tovey, and Trick [2], which in turn was shown to be NP-hard by Karp [27]. The complete proof of the NP-hardness of Kemeny-Score can be viewed as consisting of the following three steps.

1. Vertex-Cover is NP-hard [27].
2. Vertex-Cover $\leq_m^p$ Feedback-Arc-Set [27].
3. Feedback-Arc-Set $\leq_m^p$ Kemeny-Score [2].

Kemeny-Winner was shown to be Θ_2^p-hard by a chain of three reductions, basically a "lifted" version of the NP-hardness proof for Kemeny-Score. For the lifted-to-Θ_2^p version, Hemaspaandra, Spakowski, and Vogel [26] define suitable Θ_2^p-complete equivalent problems, and show the following.[10]

1. Vertex-Cover-Member is Θ_2^p-hard.
2. Vertex-Cover-Member $\leq_m^p$ Feedback-Arc-Set-Member.
3. Feedback-Arc-Set-Member $\leq_m^p$ Kemeny-Winner.

[10] Fitzsimmons and Hemaspaandra [20] recently showed a similar "lifting" to Δ_2^p, to show that Kemeny-Winner for weighted elections (elections where each voter has a corresponding integral weight) and for high-multiplicity elections (where the voters are represented as a list of distinct votes and their corresponding counts) is hard for Δ_2^p.

In order to show that Kemeny-CCDC* is Σ_2^p-hard, we lift the reductions to Σ_2^p. We already defined an appropriate Σ_2^p-complete analogue of Vertex-Cover and below we define an appropriate Σ_2^p-complete analogue of Feedback-Arc-Set, and we show the following.

1. Vertex-Cover-Member-Select is Σ_2^p-hard (Lemma 4).
2. Vertex-Cover-Member-Select $\leq_m^p$ Feedback-Arc-Set-Member-Select (Lemma 12).
3. Feedback-Arc-Set-Member-Select $\leq_m^p$ Kemeny-CCDC* (Lemma 13).

The Σ_2^p-complete analogue of Feedback-Arc-Set is defined as follows.

Name: Feedback-Arc-Set-Member-Select

Given: Irreflexive and antisymmetric directed graph $G = (V, A)$, a set $V' \subseteq V - \{\hat{v}\}$ of deletable vertices, delete limit k, and vertex $\hat{v} \in V$.

Question: Does there exist a set $W \subseteq V'$ of at most k deletable vertices such that some minimum size feedback arc set of $G - W$ contains all arcs entering $\hat{v}$?

The second and third parts of the lifting use similar constructions as in the NP, Θ_2^p, and Δ_2^p cases. As we saw previously in the proof Lemma 4, the proof that Vertex-Cover-Member-Select is Σ_2^p-hard requires significantly more work.

Lemma 12 *Vertex-Cover-Member-Select $\leq_m^p$ Feedback-Arc-Set-Member-Select.*

Proof. Given a graph G, define digraph $f(G) = (\widehat{V}, A)$ as in the standard reduction from Vertex-Cover to Feedback-Arc-Set from [27].

1. $\widehat{V} = \{v, v' \mid v \in V\}$.
2. $A = \{(v, v') \mid v \in V\} \cup \{(v', w), (w', v) \mid \{v, w\} \in E\}$.

We know from [26, Lemma 4.8] that for every graph G' and vertex $\hat{v}$ in G', G' has a minimum size vertex cover containing $\hat{v}$ if and only if $f(G')$ has a minimum size feedback arc set containing $(\hat{v}, \hat{v}')$, which is the only arc entering $\hat{v}'$.

Our reduction from Vertex-Cover-Member-Select to Feedback-Arc-Set-Member-Select maps $(G, V', k, \hat{v})$ to $(H, V' - \{\hat{v}\}, k, \hat{v}')$, where $H = f(G)$. Let $W \subseteq V' - \{\hat{v}\}$. Then from [26, Lemma 4.8], $G - W$ has a minimum size vertex cover containing $\hat{v}$ if and only if $f(G - W)$ has a minimum size feedback arc set containing $(\hat{v}, \hat{v}')$, which is the only arc entering $\hat{v}'$.

Now consider H. Note that if we delete a vertex v, we do not have any cycles going through v' and so A' is a minimum feedback arc set of $H - W$ if and only if A' is a minimum feedback arc set of $H - W - \{v' \mid v \in W\}$. Since $H - W - \{v' \mid v \in W\} = f(G - W)$, it follows that $G - W$ has a minimum size vertex cover containing $\hat{v}$ if and only if $H - W$ has a minimum size feedback arc set containing $(\hat{v}, \hat{v}')$, which is the only arc entering $\hat{v}'$. ❑

Lemma 13 *Feedback-Arc-Set-Member-Select $\leq_m^p$ Kemeny-CCDC*.*

Proof. We use the construction from [2]. Given an irreflexive and antisymmetric digraph $G = (C, A)$, let C be the set of candidates and use McGarvey's construction [30] to construct in polynomial time a set of voters such that for every arc (v, w) in A, there are exactly two more voters who prefer v to w than who prefer w to v and if there are no arcs between v and w then the same number of voters prefer v to w as w to v. From the proof of

Lemma 4.2 of [26], it holds that for every $C' \subseteq C$ and for every $c \in C'$ that some minimum feedback arc set of $(C', A \cap (C' \times C'))$ contains all arcs entering c if and only if c is a Kemeny winner of (C', V). This then shows that Feedback-Arc-Set-Member-Select reduces to Kemeny-CCDC*. ❑

This completes the proof that Kemeny-CCDC* is Σ_2^p-complete.

Do we always get this jump from a Θ_2^p-complete winner problem to a Σ_2^p-complete control problem? Dwork et al. [15] show that Kemeny-Winner is already NP-hard if we have four voters. And Fitzsimmons and Hemaspaandra [20] show that Kemeny-Winner for four voters is still Θ_2^p-complete. Certainly the voter control cases for Kemeny with four voters are still in Θ_2^p, and the control by adding voters and control by deleting voters cases are clearly Θ_2^p-complete. What about the candidate control cases?

Theorem 14 *Kemeny-CCDC* is Σ_2^p-complete, even for four voters.*

Proof. To show hardness, we argue as in the proof that Kemeny-Winner for four voters is Θ_2^p-hard from [20].

Given an irreflexive and antisymmetric digraph $G = (V, A)$ and vertex $\hat{v} \in V$, we first compute an irreflexive and antisymmetric digraph $\widehat{G}$ as done in [15]. $\widehat{G} = (\widehat{V}, \widehat{A})$ such that $\widehat{V} = V \cup A$ and $\widehat{A} = \{(v, (v, w)), ((v, w), w) \mid (v, w) \in A\}$. Let $W \subseteq V - \{\hat{v}\}$. Note that $G - W$ has a feedback arc set of size ℓ that contains all arcs entering $\hat{v}$ if and only if $\widehat{G} - W$ has a feedback arc set of size ℓ that contains all arcs entering $\hat{v}$. It follows that $(G, V', k, \hat{v})$ is in Feedback-Arc-Set-Member-Select if and only if $(\widehat{G}, V', k, \hat{v})$ is in Feedback-Arc-Set-Member-Select. Now apply the reduction from Feedback-Arc-Set-Member-Select to Kemeny-CCDC* from Lemma 13 to $(\widehat{G}, V', k, \hat{v})$. The construction from Dwork et al. [15] shows that we need only four voters in the election. ❑

A.2 Proof of Theorem 6: Kemeny-CCAC is Σ_2^p-complete

Membership follows from Corollary 2. The proof of hardness is similar to the proof of hardness for CCDC*: We define suitable Σ_2^p-complete versions of Vertex-Cover and Feedback-Arc-Set and show the following.

1. Vertex-Cover-Member-Add is Σ_2^p-hard.
2. Vertex-Cover-Member-Add $\leq_m^p$ Feedback-Arc-Set-Member-Add.
3. Feedback-Arc-Set-Member-Add $\leq_m^p$ Kemeny-CCAC.

Name: Vertex-Cover-Member-Add

Given: Graph $G = (V \cup V', E)$, set of addable vertices V', addition limit k, and vertex $\hat{v} \in V$.

Question: Does there exist a set $W \subseteq V'$ of at most k addable vertices such that $\hat{v}$ is a member of a minimum vertex cover of $(V \cup W, E)$?[11]

Name: Feedback-Arc-Set-Member-Add

Given: Irreflexive and antisymmetric directed graph $G = (V \cup V', A)$, a set V' of addable vertices, addition limit k, and vertex $\hat{v} \in V$.

[11] We slightly abuse notation by writing $(V \cup W, E)$ instead of $(V \cup W, E')$, where E' is the restriction of E to $V \cup W$.

Question: Does there exist a set $W \subseteq V'$ of at most k addable vertices such that some minimum size feedback arc set of $(V \cup W, A)$ contains all arcs entering $\hat{v}$?

The proof that Vertex-Cover-Member-Add reduces to Feedback-Arc-Set-Member-Add is similar to the proof of Lemma 12 and the proof that Feedback-Arc-Set-Member-Add reduces to Kemeny-CCAC is similar to the proof of Lemma 13. This leaves the following lemma to prove.

Lemma 15 *Vertex-Cover-Member-Add is Σ_2^p-complete.*

Proof. The upper bound is immediate. For the lower bound, we will again reduce from QSAT_2. We will modify the construction of the proof of Lemma 4, which showed Σ_2^p-hardness for Vertex-Cover-Member-Select. Let H be the graph from the proof of Lemma 4. We add an addable vertex x_i' connected only to x_i and an addable vertex $\overline{x_i}'$ connected only to $\overline{x_i}$. So, $V = V(H)$, $V' = \{x_i', \overline{x_i}' \mid 1 \leq i \leq n\}$, and $E = E(H) \cup \{\{x_i, x_i'\}, \{\overline{x_i}, \overline{x_i}'\} \mid 1 \leq i \leq n\}$.[12]

Let $W \subseteq V'$. Let $H' = (V \cup W, E)$. Some relevant properties of H' are as follows.

1. If X is a vertex cover, then X is still a vertex cover if we replace x_i' by x_i and $\overline{x_i}'$ by $\overline{x_i}$. In particular, this implies that there is a minimum vertex cover that does not contain any vertex in W.

2. Every vertex cover of H' contains at least one of $\{x_i, \overline{x_i}\}$, at least one of $\{y_i, \overline{y_i}\}$, at least three of $\{a_j, b_j, c_j, d_j\}$, at least one of $\{x_i, x_i'\}$ for $x_i' \in W$, and at least one of $\{\overline{x_i}, \overline{x_i}\}$ for $\overline{x_i}' \in W$.

3. If X is a vertex cover of size $2n + 3m$, then $\hat{v} \notin X$ and $V' \cap X = \emptyset$ and $X \cap \{x_i, \overline{x_i}, y_i, \overline{y_i} \mid 1 \leq i \leq n\}$ corresponds to a satisfying assignment for ϕ.

4. If α is a satisfying assignment for ϕ, then there is a vertex cover X of size $2n + 3m$ of (V, E) such that $X \cap \{x_i, \overline{x_i}, y_i, \overline{y_i} \mid 1 \leq i \leq n\}$ corresponds to this assignment.

5. If for all $i, 1 \leq i \leq n$, $\{x_i', \overline{x_i}'\} \not\subseteq W$, then there is a vertex cover of size $2n + 3m + 1$ that includes $\hat{v}$ and that does not contain any vertex in W.

We will show that $\exists x_1 \cdots \exists x_n \neg(\exists y_1 \cdots \exists y_n \phi(x_1, \ldots, x_n, y_1, \ldots, y_n))$ if and only if there exists a set $W \subseteq V'$ of at most n addable vertices such that $\hat{v}$ is a member of a minimum vertex cover of $(V \cup W, E)$.

First assume that $\exists x_1 \cdots \exists x_n \neg(\exists y_1 \cdots \exists y_n \phi(x_1, \ldots, x_n, y_1, \ldots, y_n))$. Let $\alpha \in \{0,1\}^n$ be an assignment to $x_1 \cdots x_n$ such that $\phi(\alpha_1, \ldots, \alpha_n, y_1, \ldots, y_n)$ is not satisfiable. Let W be the set of vertices in V' corresponding to α, i.e., $W = \{x_i' \mid \alpha_i = 1\} \cup \{\overline{x_i}' \mid \alpha_i = 0\}$. Then $(V \cup W, E)$ does not have a vertex cover of size $2n + 3m$ (by 3), and $(V \cup W, E)$ does have a vertex cover of size $2n + 3m + 1$ that includes $\hat{v}$ (by 5).

For the converse, suppose that W is a set of at most n vertices from $\{x_i', \overline{x_i}' \mid 1 \leq i \leq n\}$ such that $\hat{v}$ is a member of a minimum vertex cover of $(V \cup W, E)$. If W corresponds to an assignment α, it is easy to see that $\phi(\alpha_1, \ldots, \alpha_n, y_1, \ldots, y_n)$ is not satisfiable. But W does not necessarily correspond to an assignment, since it is possible for W to contain neither or both of $\{x_i', \overline{x_i}'\}$. Let $\widehat{n}$ be the number of $\{x_i', \overline{x_i}'\}$ pairs that are unadded, i.e., for which $\{x_i', \overline{x_i}'\} \cap W = \emptyset$. Note (by 2) that the size of a vertex cover of $(V \cup W, E)$ is at least $\|W\| + \widehat{n} + n + 3m$, and that vertex covers of that size do not include $\hat{v}$. Since $\hat{v}$ is a member of a minimum size vertex cover of $(V \cup W, E)$, it follows that $(V \cup W, E)$ does not have a vertex cover of size $\|W\| + \widehat{n} + n + 3m$.

[12] We can not simply use H with the x-vertices as addable vertices, since in that case d would be a member of a minimum vertex cover after adding any nonempty set of vertices that corresponds to an assignment.

Let $\alpha \in \{0,1\}^n$ be an assignment to $x_1 \cdots x_n$ that is consistent with W, in the sense that for all $i, 1 \leq i \leq n$, if $W \cap \{x_i', \overline{x_i}'\} = \{x_i'\}$, then $\alpha_i = 1$ and if $W \cap \{x_i', \overline{x_i}'\} = \{\overline{x_i}'\}$, then $\alpha_i = 0$. To complete the proof, suppose for a contradiction that $\phi(\alpha_1, \ldots, \alpha_n, y_1, \ldots, y_n)$ is satisfiable. Then (by 4) (V, E) has a vertex cover X of size $2n+3m$ such that $X \cap \{x_i, \overline{x_i} \mid 1 \leq i \leq n\}$ corresponds to α. Clearly, $X \cup \{x_i \mid x_i' \in W\} \cup \{\overline{x_i} \mid \overline{x_i}' \in W\}$ is a vertex cover of $(V \cup W, E)$. Since $\|X \cap (\{x_i \mid x_i' \in W\} \cup \{\overline{x_i} \mid \overline{x_i}' \in W\})\| = n - \widehat{n}$, the size of this vertex cover is $\|W\| + \|X\| - (n + \widehat{n}) = \|W\| + \widehat{n} + n + 3m$, which implies that $(V \cup W, E)$ has a vertex cover of size $\|W\| + \widehat{n} + n + 3m$, which is a contradiction. ❑

A.3 Proof of Theorem 7: Kemeny′-CCAV is Σ_2^p-complete

Note that in the reduction from feedback arc set problems to Kemeny control problems, vertices correspond to candidates and arcs roughly correspond to voters. Also note that arc (v, v') in Feedback-Arc-Set-Member-Add basically correspond to vertex v. And so we can easily define a version of Feedback-Arc-Set-Member-Add where we add arcs instead of vertices.

Name: Feedback-Arc-Set-Member-Add-Arcs

Given: Irreflexive and antisymmetric directed graph $G = (V, A \cup B)$, a set B of addable arcs, addition limit k, and vertex $\hat{v} \in V$.

Question: Does there exist a set $B' \subseteq B$ of at most k addable arcs such that some minimum size feedback arc set of $(V, A \cup B')$ contains all arcs entering $\hat{v}$?

It is easy to see that Vertex-Cover-Member-Add $\leq_m^p$ Feedback-Arc-Set-Member-Add-Arcs: Given a graph $G = (V \cup W, E)$, define digraph $f(G) = (\widehat{V}, A \cup B)$ as in the standard reduction from Vertex-Cover to Feedback-Arc-Set from [27], with the set of addable arcs B being the arcs corresponding to the addable vertices W.

1. $\widehat{V} = \{v, v' \mid v \in V \cup W\}$.
2. $A = \{(v, v') \mid v \in V\} \cup \{(v', w), (w', v) \mid \{v, w\} \in E\}$.
3. $B = \{(v, v') \mid v \in W\}$.

Our reduction from Vertex-Cover-Member-Add to Feedback-Arc-Set-Member-Add-Arcs maps $(G, W, k, \hat{v})$ to $((\widehat{V}, A \cup B), B, k, \hat{v}')$.

As mentioned previously, in the reduction from feedback arc set problems to Kemeny control problems, vertices correspond to candidates and arcs roughly correspond to voters. If the voters vote with total orders, each arc corresponds to two voters: arc $a \to b$ corresponds to a voter voting $a > b > C - \{a, b\}$ and a voter voting $(C - \{a, b\})^r > a > b$ [30].

There exist variations of Kemeny where the voters do not necessarily list all of the candidates in their votes. Dwork et al. [15] consider a natural variant of Kemeny, which we here call Kemeny′, where unlisted candidates in a vote do not contribute to the distance to the Kemeny consensus and so do not increase the Kemeny score. (The Kemeny consensus is still a complete total order.) So, in this model, a voter voting $a > b$ corresponds exactly to an arc $a \to b$. This gives a straightforward reduction from Feedback-Arc-Set-Member-Add-Arcs to Kemeny′-CCAV.

B Young

B.1 Details of the proof of Theorem 9

Given graph $G = (V, E)$, a delete limit $k \leq \|V\|$, and a vertex $\hat{v} \in V$, such that G has no isolated vertices and for every set W of at most k vertices, $\alpha(G - W) \geq 3$, we will show that $((V, E), k, \hat{v})$ is a positive instance of Independent-Set-Member-Delete if and only if we can make the Young score of p at least 2 and at least as high as the Young score of q by deleting at most k voters.

Let $W \subseteq V$ be a set of at most k vertices such that $\hat{v}$ is in a maximum independent set of $G - W$. Delete the voters corresponding to W. We will show that the Young score of p is at least 2 and at least as high as the Young score of q. Let $\widehat{W}$ be a maximum independent set of $G - W$ such that $\hat{v} \in \widehat{W}$. Then all the voters corresponding to $\widehat{W}$ are of Type IA. It is easy to see that p and q are weak Condorcet winners in the set of voters that consists of the voters corresponding to $\widehat{W}$, the voter of Type II, and $\|\widehat{W}\| - 1$ voters of Type III. It follows that the Young scores of p and q are at least $2\alpha(G - W)$. It also follows from the argument from [34, Lemma 2.4] that the Young score of q is at most $2\alpha(G - W)$.

For the converse, suppose X is a set of at most k voters such that after deletion, p's Young score is at least 2 and at least as high as q's Young score. Then X does not contain the voter of Type II. Let W be the set of vertices corresponding to the Type-I voters in X. Then the Young score of q is $2\alpha(G - W)$. Let $\widehat{X}$ be a set of voters of size $2\alpha(G - W)$ such that p is a weak Condorcet winner of $\widehat{X}$. Without loss of generality, assume that $\widehat{X}$ does not contain voters of Type IB (if it does, we can replace such a voter by a voter of Type III). In order for p to be a weak Condorcet winner, the vertices corresponding to the voters of Type I in $\widehat{X}$ form an independent set of $G - W$. Since all voters in $\widehat{X}$ of Type I are of Type IA, this independent set contains no vertices connected to $\hat{v}$ and so we can safely add $\hat{v}$ to this independent set, giving us an independent set containing $\hat{v}$ of size at least $\alpha(G - W)$.

C Dodgson Elections

In this section we consider the complexity of control for Dodgson elections [14]. Recall that the Dodgson score of a candidate c using the Dodgson election system is the minimum number of swaps between adjacent candidates in votes such that c is a Condorcet winner and that the candidate(s) with the minimum score are the winner(s) of the Dodgson election. We will use the notation DodgsonScore(c) to denote the Dodgson score of a candidate c in a given election. As we did with Kemeny, we first consider the complexity of CCDC*.

Theorem 16 *Dodgson-CCDC* is Σ_2^p-complete.*

Proof. In our hardness proofs for control for Kemeny and Young elections, our approach was to define a new simple Σ_2^p-complete analogue of Vertex-Cover or Independent-Set (the problems used to show NP-hardness for the score problems) to then reduce to a control problem. Dodgson score was originally shown NP-hard by a reduction from Exact-Cover-by-3-Sets [2]. [24] provides a reduction from Three-Dimensional-Matching (3DM) as part of the proof of Θ_2^p-completeness of the winner problem. There does exist a simple Σ_2^p-complete analogue of 3DM [31]. However, this analogue does not straightforwardly reduce to Dodgson control problems.

Fortunately, in Dodgson there is a lot of flexibility in how to construct the voters in a reduction, and so we will directly reduce QSAT_2 to Dodgson-CCDC*, though the construction is quite involved. To better understand our reduction, it helps to first consider a reduction from 3SAT to the Dodgson score problem, in which we ask if the Dodgson score of a distinguished candidate is at most k.

Let $\phi(z_1, \ldots, z_n)$ be a Boolean formula in 3cnf, where $\phi = \psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_m$ and for each $i, 1 \leq i \leq m$, $\psi_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$. Without loss of generality, let $n \geq 1$ and $m \geq 1$. We now construct an instance of the Dodgson score problem. An example of this construction is given in Example 1 below. We begin by showing the core parts of the construction (Blocks I and II).

Let $C = \{\widehat{z}_1, \ldots, \widehat{z}_n\} \cup \{c_1, \ldots, c_m\} \cup \{c_{i,1}, c_{i,2}, c_{i,3}, \ldots, c_{m,1}, c_{m,2}, c_{m,3}\} \cup \{q, b\}$. Let there be the following voters. Note that "$\cdots$" in a vote denotes that the remaining candidates are strictly ranked in a arbitrary, fixed, and easy to compute order. Similarly, a set in a vote denotes that the candidates in that set are strictly ranked with respect to that order.

Block I For each $i, 1 \leq i \leq m$ and $j, 1 \leq j \leq 3$,

- One voter voting: $(c_i > c_{i,j} > q > \cdots)$.

Block II For each $t, 1 \leq t \leq n$,

- One voter voting: $(\widehat{z}_t > \{c_{i,j} \mid \ell_{i,j} = z_t\} > q > \cdots)$.
- One voter voting: $(\widehat{z}_t > \{c_{i,j} \mid \ell_{i,j} = \overline{z_t}\} > q > \cdots)$.

The above election can be easily padded so that the following properties hold.

- q needs one vote over each c_i, each $c_{i,j}$, and each $\widehat{z}_i$, and no votes over b.
- One swap in votes other than the Block I and Block II votes does not give q a vote over any of the c_i, $c_{i,j}$, or $\widehat{z}_i$ candidates, i.e., one swap in votes other than Block I and Block II votes is useless.

Note that for each $c_{i,j}$ candidate, $2n + 3m - 2$ (of the $2n + 3m$) voters in Blocks I and II prefer q to $c_{i,j}$, that for each $\widehat{z}_i$ candidate, $2n + 3m - 2$ voters in Blocks I and II prefer q to $\widehat{z}_i$, and that for each c_i candidate, $2n + 3m - 3$ of the voters in Blocks I and II prefer q to c_i. We can ensure that q needs one vote over each of these candidates (by making q tie pairwise with each of these candidates) by adding the following $2n + 3m - 4$ voters. We use the buffer candidate "b" to ensure that one swap outside of Blocks I and II does not give q a vote over any of the c_i, $c_{i,j}$, or $\widehat{z}_i$ candidates.

Block III

- One voter voting: $(\cdots > b > q > \{c_1, \ldots, c_m\})$.
- $2n + 3m - 5$ voters voting: $(\cdots > b > q)$.

We will show in Lemma 17 that ϕ is satisfiable if and only if DodgsonScore$(q) \leq 4m + n$.

Before proving the correctness of this construction, let's first consider the following example, which will show how assignments to variables correspond to swaps between adjacent candidates.

Example 1 *Given the formula* $\phi(z_1, z_2) = (z_1 \vee \overline{z_2} \vee z_1) \wedge (\overline{z_2} \vee \overline{z_1} \vee z_2)$*, our construction gives the following election.*

$C = \{c_1, c_2, c_{1,1}, c_{1,2}, c_{1,3}, c_{2,1}, c_{2,2}, c_{2,3}, \widehat{z}_1, \widehat{z}_2, q, b\}$*, and we have the following voters. (We do not name the voters in Block III since we do not need to swap* q *in these votes.)*

Block I

- v_1 *voting:* $(c_1 > c_{1,1} > q > \cdots)$.
- v_2 *voting:* $(c_1 > c_{1,2} > q > \cdots)$.

- v_3 *voting:* $(c_1 > c_{1,3} > q > \cdots)$.
- v_4 *voting:* $(c_1 > c_{2,1} > q > \cdots)$.
- v_5 *voting:* $(c_1 > c_{2,2} > q > \cdots)$.
- v_6 *voting:* $(c_1 > c_{2,3} > q > \cdots)$.

Block II

- v_7 *voting:* $(\widehat{z}_1 > c_{1,1} > c_{1,3} > q > \cdots)$.
- v_8 *voting:* $(\widehat{z}_1 > c_{2,2} > q > \cdots)$.
- v_9 *voting:* $(\widehat{z}_2 > c_{2,3} > q > \cdots)$.
- v_{10} *voting:* $(\widehat{z}_2 > c_{1,2} > c_{2,1} > q > \cdots)$.

Block III

- *One voter voting:* $(\cdots > b > q > c_1 > c_2)$.
- *Five voters voting:* $(\cdots > b > q)$.

It is easy to see that the assignment $z_1 = 1, z_2 = 0$ is a satisfying assignment for ϕ. We will now show that this implies that DodgsonScore$(q) \leq 10$.

Swapping q over a $c_{i,j}$ candidate in its Block I vote will correspond to that clause literal being true and swapping q over a $c_{i,j}$ candidate in its Block II vote will correspond to that clause literal being false. Since z_1 is true, swap q with $c_{1,1}$ in v_1 and $c_{1,3}$ in v_3 (these correspond to positive z_1-literals in ϕ) and then in Block II swap q with $c_{2,2}$ and $\widehat{z}_1$ in v_8 (this vote corresponds to the negated z_1-literals). Similarly, since z_2 is false, swap q with $c_{1,2}$ in v_2, swap q with $c_{2,1}$ in v_4, and then in Block II swap q with $c_{2,3}$ and $\widehat{z}_2$ in v_9. This takes eight swaps for q to gain one vote over each of the $\widehat{z}_i$ and $c_{i,j}$ candidates.

Note that since ϕ is satisfiable, for each clause there exists at least one true literal, and we just swapped q over the true clause literals in Block I. So, to gain one vote over c_1, swap q with c_1 in v_1 (we already swapped q with $c_{1,1}$) and to gain one vote over c_2, swap q with c_2 in v_2 (we already swapped q with $c_{1,2}$). This takes exactly two swaps. Thus the Dodgson score of q is ≤ 10.

We will now show the following lemma.

Lemma 17 *ϕ is satisfiable if and only if DodgsonScore$(q) \leq 4m + n$*

Proof. First suppose that $\phi(z_1, \ldots, z_n)$ is satisfiable. Fix an assignment $\alpha \in \{0,1\}^n$ for ϕ. We argue as in Example 1.

Recall that q needs one vote over each $\widehat{z}_i$, each c_i, and each $c_{i,j}$ candidate. For each $t, 1 \leq t \leq n$ if z_t is false in the assignment (i.e., $\alpha_t = 0$) then for each $c_{i,j}$ such that $\ell_{i,j} = z_t$, swap q with $c_{i,j}$ in the corresponding votes in Block I and swap q up to the top in the corresponding z_t vote in Block II, and if z_t is true in the assignment (i.e., $\alpha_t = 1$) then for each $c_{i,j}$ such that $\ell_{i,j} = \overline{z_t}$, swap q with $c_{i,j}$ in the corresponding votes in Block I and swap q up to the top in the corresponding $\overline{z_t}$ vote in Block II. For example, if z_t is false in the satisfying assignment do the following.

- Swap q with $c_{i,j}$ in each Block I vote $(c_i > c_{i,j} > q > \cdots)$ for which $\ell_{i,j} = \overline{z_t}$ to get $(c_i > q > c_{i,j} > \cdots)$.
- Swap q up to the top in the Block II vote $(\widehat{z}_t > \{c_{i,j} \mid \ell_{i,j} = z_t\} > q > \cdots)$ to get $(q > \widehat{z}_t > \{c_{i,j} \mid \ell_{i,j} = z_t\} > \cdots)$.

This takes $3m + n$ swaps.

Since α is a satisfying assignment for ϕ we know that for each clause there exists a clause literal that is true. In our election, this means that for each $i, 1 \leq i \leq m$, there exists a $j, 1 \leq j \leq 3$, such that q has been swapped in the vote $(c_i > c_{i,j} > q > \cdots)$ to get $(c_i > q > c_{i,j} > \cdots)$. So with a total of m swaps q can gain one vote over each clause candidate c_i. This all takes a total of $4m + n$ swaps.

For the converse, suppose that DodgsonScore$(q) \leq 4m + n$. The Dodgson score of q will always be at least $4m + n$, since q must gain one vote over $4m + n$ candidates (each c_i, each $c_{i,j}$, and each $\widehat{z}_i$ candidate). To avoid "wasting" swaps, q must swap only with these candidates and only in Block I and Block II, since these are the only votes were q is directly adjacent to these candidates. Since each swap must gain a necessary vote, for each $t, 1 \leq t \leq n$, we either swap q up in the Block II vote $(\widehat{z}_t > \{c_{i,j} \mid \ell_{i,j} = z_t\} > q > \cdots)$ to get $(q > \widehat{z}_t > \{c_{i,j} \mid \ell_{i,j} = z_t\} > \cdots)$ or swap q up in the Block II vote $(\widehat{z}_t > \{c_{i,j} \mid \ell_{i,j} = \overline{z_t}\} > q > \cdots)$ to get $(q > \widehat{z}_t > \{c_{i,j} \mid \ell_{i,j} = \overline{z_t}\} > \cdots)$. Note that this is the only way for q to gain one vote over $\widehat{z}_t$ without wasting swaps. In the first case set z_t to false and in the second case set z_t to true. This gives us a satisfying assignment for ϕ since for each $i, 1 \leq i \leq m$, q must also swap with the clause candidate c_i, which implies that for some $j, 1 \leq j \leq 3$, q swaps with $c_{i,j}$ in Block I, and since q only swaps over each of these candidates exactly once, q did not swap with this $c_{i,j}$ in Block II. So $c_{i,j}$ corresponds to a true literal in the constructed satisfying assignment. ❑

We will now use the idea from the reduction above to show that Dodgson-CCDC* is Σ_2^p-complete.

Membership in Σ_2^p follows from Corollary 2. To show hardness, we reduce from QSAT$_2$ and recall that it is defined as all true formulas of the form $\exists x_1 \cdots \exists x_n \neg(\exists y_1 \cdots \exists y_n \phi(x_1, \ldots, x_n, y_1, \ldots, y_n))$, where ϕ is a formula in 3cnf, where $\phi = \psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_m$ and for each $i, 1 \leq i \leq m$, $\psi_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$. Let $X = \{x_1, \ldots, x_n\}$ and $\overline{X} = \{\overline{x_1}, \ldots, \overline{x_n}\}$ denote the positive and negative x-literals and let $Y = \{y_1, \ldots, y_n\}$ and $\overline{Y} = \{\overline{y_1}, \ldots, \overline{y_n}\}$ denote the positive and negative y-literals. Without loss of generality, let $n > m$ and $n > 6$. Let $\widehat{m}$ be the number of occurrences of y-literals, i.e., $\widehat{m} = \|\{(i,j) \mid 1 \leq i \leq m, 1 \leq j \leq 3, \text{ and } \ell_{i,j} \in Y \cup \overline{Y}\}\|$. Note that $\widehat{m} \leq 3m$.

We now construct an instance of Dodgson-CCDC*. Let the candidate set C consist of the x-variable candidates $\{\widehat{x}_1, \ldots, \widehat{x}_n\}$, the y-variable candidates $\{\widehat{y}_1, \ldots, \widehat{y}_n\}$, the clause candidates $\{c_1, \ldots, c_m\}$, a clause-literal candidate for each occurrence of a y-literal $\{c_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq 3, \text{ and } \ell_{i,j} \in Y \cup \overline{Y}\}$ (note that there are $\widehat{m}$ clause-literal candidates), the buffer candidates $\{b_1, b_2, b_3, b_4\}$, the positive and negated x-literal candidates $X \cup \overline{X}$ and the candidates p, q, and d. Let p be the preferred candidate of the chair, let $X \cup \overline{X}$ be the set of deletable candidates, and let the delete limit be $2n$ (i.e., we can delete any subset of $X \cup \overline{X}$; this will be useful when we look at CCAC). We have the following $16n + 8m + 2\widehat{m} - 8$ voters.

Block IA For each $i, 1 \leq i \leq m$, and $j, 1 \leq j \leq 3$, such that $\ell_{i,j} = y_t$ or $\overline{y_t}$,

- One voter voting: $(c_i > c_{i,j} > q > b_1 > p > d > \cdots)$.

Block IB For each $i, 1 \leq i \leq m$, and $j, 1 \leq j \leq 3$, such that $\ell_{i,j} = x_t$ or $\overline{x_t}$,

- If $\ell_{i,j} = x_t$, one voter voting: $(c_i > x_t > q > b_1 > p > d > \cdots)$.
- If $\ell_{i,j} = \overline{x_t}$, one voter voting: $(c_i > \overline{x_t} > q > b_1 > p > d > \cdots)$.

Block II For each $t, 1 \leq t \leq n$,

- One voter voting: $(\widehat{y}_t > \{c_{i,j} \mid \ell_{i,j} = y_t\} > q > b_1 > p > d > \cdots)$.

- One voter voting: $(\widehat{y}_t > \{c_{i,j} \mid \ell_{i,j} = \overline{y_t}\} > q > b_1 > p > d > \cdots)$.

Block III For each $t, 1 \leq t \leq n$,

- One voter voting: $(\widehat{x}_t > x_t > p > q > d > \cdots)$.
- One voter voting: $(\widehat{x}_t > \overline{x_t} > p > q > d > \cdots)$.

Block IV

- One voter voting: $(d > \cdots > b_2 > \{c_1, \ldots, c_m\} > \{\text{all } c_{i,j}\} > \{\widehat{y}_1, \ldots, \widehat{y}_n\} > p > b_1 > q)$.
- One voter voting: $(d > \cdots > b_2 > \{\widehat{x}_1, \ldots, \widehat{x}_n\} > q > b_1 > b_3 > p)$.

Block V

- One voter voting: $(d > \cdots > b_2 > p > q > \{c_1, \ldots, c_m\})$.
- $2n + 3m - 6$ voters voting: $(d > \cdots > b_2 > p > b_1 > q)$.

Block VI

- $4n + m + \widehat{m} - 2$ voters voting: $(p > q > d > \cdots)$.
- $2n + m + \widehat{m} - 4$ voters voting: $(\cdots > b_3 > q > b_4 > d > p > b_1 > b_2 > X \cup \overline{X})$.
- $4n - 1$ voters voting: $(d > \cdots > b_2 > q > b_1 > p > b_3 > b_4 > X \cup \overline{X})$.
- Two voters voting: $(\cdots > b_3 > d > p > b_4 > q > b_1 > b_2 > X \cup \overline{X})$.

Note the following about the election above.

1. q needs one vote over each $\widehat{x}_i$, each $\widehat{y}_i$, each c_i, each $c_{i,j}$, and p, and no votes over other candidates. Note that DodgsonScore$(q) \geq 2n + m + \widehat{m} + 1$.

2. The votes in Block IA and the Block II votes function in the same way to determine (part of) the score of q as in the Dodgson score reduction above. And q can only gain a vote over any of the $\widehat{y}_i$, c_i, or $c_{i,j}$ candidates without wasting a swap in Block I and Block II.

3. p needs one vote over each $\widehat{x}_i$, each $\widehat{y}_i$, each c_i, each $c_{i,j}$, and q, and no votes over other candidates. p needs at least two swaps to gain one vote over q and to accomplish this p needs to swap over a buffer candidate. Note that DodgsonScore$(p) \geq 2n + m + \widehat{m} + 2$.

4. d needs $2n+m+\widehat{m}-1$ votes over q, 3 votes over p, and no votes over other candidates. Note that regardless of which candidates in $X \cup \overline{X}$ are deleted, DodgsonScore$(d) = 2n + m + \widehat{m} + 2$.

5. Each candidate in $C-\{p, q, d\}$ needs at least $2n+m+\widehat{m}+2$ votes over d. So regardless of which candidates in $X \cup \overline{X}$ are deleted, for each $c \in C-\{p, q, d\}$, DodgsonScore$(c) \geq 2n + m + \widehat{m} + 2$.

Let's consider the Dodgson score of p. It is easy to see for every $X' \subseteq X \cup \overline{X}$, in the election $(C - X', V)$, p can gain each needed vote over the $\widehat{y}_i$, c_i, and $c_{i,j}$ candidates in the Block IV vote and then use two more swaps to gain one vote over q. What remains is to show how p can gain one vote over each of the $\widehat{x}_i$ candidates. When at least one of the $\{x_i, \overline{x_i}\}$ candidates is deleted, p can gain one vote over the corresponding $\widehat{x}_i$ candidate in Block III, using one swap. So, we can state the following observation.

Observation 18 *For every $X' \subseteq X \cup \overline{X}$,*

1. *The Dodgson score of p with X' removed is $\geq 2n + m + \widehat{m} + 2$.*

2. *The Dodgson score of p with X' removed is $2n + m + \widehat{m} + 2$ if and only if for each $i, 1 \leq i \leq n$, at least one of each x_i or $\overline{x_i}$ in is X'.*

We now consider the Dodgson score of q.

Lemma 19 *For every $X' \subseteq X \cup \overline{X}$,*

1. *The Dodgson score of q with X' removed is $\geq 2n + m + \widehat{m} + 1$.*

2. *The Dodgson score of q with X' removed is $2n + m + \widehat{m} + 1$ if and only if ϕ with x-literals in X' set to true and x-literals not in X' set to false is satisfiable.*

Proof.

1. Clearly the Dodgson score of q with X' removed is $\geq 2n + m + \widehat{m} + 1$, since q needs one vote over each $\widehat{x}_i$, each $\widehat{y}_i$, each c_i, each $c_{i,j}$, and p, and does not need any votes over other candidates.

2. Suppose that q can be made a Condorcet winner in the election $(C - X', V)$ with exactly $2n + m + \widehat{m} + 1$ swaps. This means that every swap must gain a necessary vote, since q needs one vote each over $2n + m + \widehat{m} + 1$ candidates. With the exception of candidate p and the $\widehat{x}_i$ candidates, q is only ever not separated from candidates it needs votes over by a buffer candidate in Block I and Block II. So $n + m + \widehat{m}$ of these swaps must occur there. Note that q can easily swap over p (in Block III), and the $\widehat{x}_i$ candidates (in Block IV) with exactly $n + 1$ swaps.

   Consider the candidates deleted in X'. If $\ell_{i,j} = x_t$ ($\overline{x_t}$) then q can gain one vote over c_i in the corresponding Block I vote with only one swap. This will correspond to setting x_t to true (false) in the corresponding assignment and setting x_t ($\overline{x_t}$) to true means that the ith clause is true.

   Now let's consider the y-literals. Since the Dodgson score of q is $2n + m + \widehat{m} + 1$, q can swap over the $\widehat{y}_i$, c_i, and $c_{i,j}$ candidates without wasting swaps. It follows from a similar argument as in the proof of Lemma 17 that since q can swap over the remaining $n + m + \widehat{m}$ candidates without wasting a swap, ϕ with the x-literals in X' set to true and x-literals not in X' set to false is satisfiable.

   For the other direction, consider a fixed satisfying assignment for y-variables in ϕ, where the assignment to the x-literals is set by the deleted candidates X'. For each t, $1 \leq t \leq n$, if y_t is true (false) in the satisfying assignment, swap q up over the $c_{i,j}$ candidates and $\widehat{y}_t$ in the $\overline{y_t}$ (y_t) vote in Block II. Then for each t, $1 \leq t \leq n$, if y_t is true (false) swap q with $c_{i,j}$ in the corresponding $\ell_{i,j} = y_t$ ($\overline{y_t}$) Block I votes. This takes $n + m$ swaps and handles the $c_{i,j}$ and $\widehat{y}_i$ candidates. Since we are looking at a satisfying assignment, for all i, there exists a j, $1 \leq j \leq 3$ such that $c_i > c_{i,j} > q$ has been swapped to $c_i > q > c_{i,j}$ or x_t ($\overline{x_t}$) has been deleted so that $c_i > q$. With k extra swaps, p beats c_i pairwise for all i. q must additionally gain one vote over p and one vote over each $\widehat{x}_i$ candidate and it is clear to see how this can be accomplished with $n + 1$ swaps. This all takes $2n + m + \widehat{m} + 1$ swaps.

❑

We will now show that $\exists x_1 \cdots \exists x_n \neg(\exists y_1 \cdots \exists y_n \phi(x_1, \ldots, x_n, y_1, \ldots, y_n)) \in \text{QSAT}_2$ if and only if there exists $X' \subseteq X \cup \overline{X}$ such that p is a Dodgson winner of $(C - X', V)$.

Suppose that $\exists x_1 \cdots \exists x_n \neg(\exists y_1 \cdots \exists y_n \phi(x_1, \ldots, x_n, y_1, \ldots, y_n) \in \text{QSAT}_2$. Fix an assignment $\alpha \in \{0,1\}^n$ for the x-literals such that ϕ is not satisfiable. Let the candidates deleted be $X' = \{x_i \mid \alpha_i = 1\} \cup \{\overline{x_i} \mid \alpha_i = 0\}$. We know from Observation 18 that the Dodgson score of p with X' deleted is $2n + m + \widehat{m} + 2$ and by Lemma 19 we know that $\text{DodgsonScore}(q) > 2n + m + \widehat{m} + 1$. And as stated in the list of properties given after the construction, the Dodgson score of each candidate in $C - \{p, q, d\}$ is at least $2n + m + \widehat{m} + 2$. Additionally, the Dodgson score of d is $2n + m + \widehat{m} + 2$ regardless of which candidates in $X \cup \overline{X}$ are deleted. It follows that p is a Dodgson winner of $(C - X', V)$.

Suppose that there exists an $X' \subseteq X \cup \overline{X}$ such that p is a winner. Regardless of which candidates in $X \cup \overline{X}$ are deleted, $\text{DodgsonScore}(d) = 2n + m + \widehat{m} + 2$ and $\text{DodgsonScore}(p) \geq 2n + m + \widehat{m} + 2$. Since p is a winner, $\text{DodgsonScore}(p) = 2n + m + \widehat{m} + 2$ and it follows from Observation 18 that X' contains at least one of each $\{x_i, \overline{x_i}\}$. Let $\alpha \in \{0,1\}^n$ be an assignment to the x-literals that is consistent with the candidates in X', that is, for all $i, 1 \leq i \leq n$, if $\alpha_i = 1$, then $x_i \in X'$ and if $\alpha_i = 0$, then $\overline{x_i} \in X'$. Suppose that $\phi(\alpha_1, \ldots, \alpha_n, y_1, \ldots, y_n)$ is satisfiable. Then by Lemma 19, $\text{DodgsonScore}(q) = 2n + m + \widehat{m} + 1$. However, since $\text{DodgsonScore}(p) = 2n + m + \widehat{m} + 2$, p is not a winner, which is a contradiction. It follows that $\phi(\alpha_1, \ldots, \alpha_n, y_1, \ldots, y_n)$ is not satisfiable. ❑

The analogous result for CCAC follows almost immediately.

Theorem 20 *Dodgson-CCAC is Σ_2^p-complete.*

Proof. Membership in Σ_2^p follows from Observation 1. To show hardness, we reduce from QSAT_2 and use the same construction as for the case of the proof of Theorem 16, with the obvious modification that the set of deletable candidates $(X \cup \overline{X})$ is now the set of unregistered candidates and the delete limit is now the addition limit of $2n$. Note that p can be a winner by deleting a set of candidates $\widehat{X} \subseteq X \cup X'$ from (C, V) if and only if p can be made a winner by adding $(X \cup X') - \widehat{X}$ to $(C - (X \cup X'), V)$, which immediately gives the result. ❑

The construction used for the proof of Dodgson-CCDC* can be adapted to show the following.

Theorem 21 *Dodgson-CCDC is Σ_2^p-complete.*

Proof Sketch. Membership in Σ_2^p follows from Corollary 2. We will adapt the construction used in the proof of Theorem 16. Set the delete limit to n. Our main challenge is to ensure that the arguments from the previous proof still go through when we delete up to n arbitrary candidates. We start with Blocks I, II, III, and IV and do the following.

1. To ensure that the buffer candidates function in the same way, we replace each buffer candidate by $n + 1$ copies of that candidate.

2. We also replace the candidate "d," which in the Dodgson-CCDC* proof, regardless of which candidates in $X \cup \overline{X}$ are deleted, always has Dodgson score equal to the minimum-possible Dodgson score of p, by $n + 1$ copies: $\{d_1, \ldots, d_{n+1}\}$. And we replace each occurrence of d in the construction by $d_1 > d_2 > \cdots > d_{n+1}$. This will ensure that regardless of which candidates in $C - \{q\}$ are deleted, the Dodgson score of the first d_i that is not deleted is the minimum-possible Dodgson score of p. (We will mention why we do not delete q later.)

3. We now will make sure that the candidates deleted correspond to an assignment. In the Dodgson-CCDC* proof, this was accomplished by the Block III votes and the $\widehat{x}_i$ candidates.

Old Block III For each $t, 1 \leq t \leq n$,

- One voter voting: $(\widehat{x}_t > x_t > p > \cdots)$.
- One voter voting: $(\widehat{x}_t > \overline{x_t} > p > \cdots)$.

In our new construction we create $2n$ of each of the $\widehat{x}_i$ candidates: $\{\widehat{x}_1^1, \ldots, \widehat{x}_1^{2n}, \ldots, \widehat{x}_n^1, \ldots, \widehat{x}_n^{2n}\}$ and we make sure that p needs one vote over each of these $2n^2$ variables. We then change Block III to be the following $4n^2$ votes.

New Block III For each $t, 1 \leq t \leq n$, and $s, 1 \leq s \leq 2n$,

- One voter voting: $(\widehat{x}_t^s > x_t > p > \cdots)$.
- One voter voting: $(\widehat{x}_t^s > \overline{x_t} > p > \cdots)$.

Now for each x_i or $\overline{x_i}$ deleted, p can gain the necessary vote over each of the at least n nondeleted $\widehat{x}_i^s$ candidates without wasting swaps, and so one such deletion decreases the Dodgson score of p by at least n. Note that if for an i, $1 \leq i \leq n$, both of x_i and $\overline{x_i}$ were deleted, or if a candidate in $C - (X \cup \overline{X})$ was deleted, then the Dodgson score of p would decrease by at most 1.

In order for p to tie with "d" by deleting at most n candidates, we need to delete exactly one of each $\{x_i, \overline{x_i}\}$, which corresponds to an assignment. Note that this also implies that we do not delete q.

4. In the Dodgson-CCDC* proof, to get its needed vote over q, p needs to waste a swap over q and does this by swapping over a buffer candidate. Since there are now $n+1$ copies of each buffer candidate, we need to modify the votes so that p still wastes exactly one swap to gain its vote over q. Recall the Block II votes from the Dodgson-CCDC* proof.

   Old Block II For each $t, 1 \leq t \leq n$,

   - One voter voting: $(\widehat{y}_t > \{c_{i,j} \mid \ell_{i,j} = y_t\} > q > b_1 > p > d > \cdots)$.
   - One voter voting: $(\widehat{y}_t > \{c_{i,j} \mid \ell_{i,j} = \overline{y_t}\} > q > b_1 > p > d > \cdots)$.

   In our new construction, we replace the buffer candidate with candidates from $X \cup \overline{X}$ in the following way.

   New Block II For each $t, 1 \leq t \leq n$,

   - One voter voting: $(\widehat{y}_t > \{c_{i,j} \mid \ell_{i,j} = y_t\} > q > \{x_t, \overline{x_t}\} > p > d > \cdots)$.
   - One voter voting: $(\widehat{y}_t > \{c_{i,j} \mid \ell_{i,j} = \overline{y_t}\} > q > \{x_t, \overline{x_t}\} > p > d > \cdots)$.

   Since from above we know that an assignment must be deleted, we know that one of each $\{x_i, \overline{x_i}\}$ remains after deletion, and so to gain one vote over q, p swaps over a remaining x-literal candidate in a New Block II vote to then swap over q. Note that in votes outside of Block II where p is ranked below q, p is separated from q by at least $n+1$ buffer candidates.

5. We must then pad the above construction (Blocks I-IV) so that the following hold.

   - q needs one vote over each $\widehat{x}_i^j$, each $\widehat{y}_i$, each c_i, each $c_{i,j}$, and p, and no votes over other candidates.
   - q can only gain a vote over any of the $\widehat{y}_i$, c_i, or $c_{i,j}$ candidates without wasting a swap in Block I and Block II.
   - p needs one vote over each $\widehat{x}_i^j$, each $\widehat{y}_i$, each c_i, each $c_{i,j}$, and q, and no votes over other candidates. p wastes a swap to gain a vote over q.

- Each $d_i \in D$ needs $2n^2 + n + m + \widehat{m} + 2$ votes in total over p and q, and no votes over candidates in $C - (\{p, q\} \cup D)$. d_1 can gain these $2n^2 + n + m + \widehat{m} + 2$ votes without wasting swaps.
- Each candidate in $C - (\{p, q\} \cup D)$ needs at least $2n^2 + n + m + \widehat{m} + 2$ votes over the candidates in D.

❑

D ASP Encodings

We present the complete encoding of the guess and check ASP programs for some of the control actions discussed in this paper (see Section 4 for details about the "guess and check" approach for Σ_2^p problems). Our encodings are written in a language mostly based on the input language to the `gringo` grounder [22]. We start by showing the Kemeny-CCAC guess program.

$$\text{candidate}(1..M) \leftarrow \text{rcandnum}(M). \tag{6}$$
$$\text{ucandidate}((\text{M} + 1)..(M + N)) \leftarrow \text{rcandnum}(M), \text{ucandnum}(N). \tag{7}$$
% Guess a subset of unregistered candidates of size at most K to add.
$$\{\text{candidate}(C) : \text{ucandidate}(C)\}\ K \leftarrow \text{limit}(K). \tag{8}$$
$$\text{candnum}(N) \leftarrow N = \{\text{candidate}(C)\}. \tag{9}$$
$$\text{domain}(1..N) \leftarrow \text{candnum}(N). \tag{10}$$
% Number of times candidate C1 is worse-ranked than candidate C2 in the preference profile
$$\text{wrank}(P, C1, C2) \leftarrow \text{ip}(P, Pos1, C1), \text{ip}(P, Pos2, C2), Pos1 > Pos2. \tag{11}$$
$$\text{wrankC}(C1, C2, N) \leftarrow \text{candidate}(C1), \text{candidate}(C2), N = \#sum\{VC, P : \text{prefVC}(P, VC), \text{wrank}(P, C1, C2)\}. \tag{12}$$
% Guess preference relation
$$1\{\text{gpref}(D, C) : \text{candidate}(C)\} \leftarrow \text{domain}(D). \tag{13}$$
$$\leftarrow \text{gpref}(Pos1, C), \text{gpref}(Pos2, C), Pos1! = Pos2. \tag{14}$$
% In the guessed preference relation C1 is better-ranked than C2
$$\text{grank}(C1, C2) \leftarrow \text{gpref}(Pos1, C1), \text{gpref}(Pos2, C2), Pos1 < Pos2. \tag{15}$$
% Number of votes that disagree on C1 being better-ranked than C2
$$\text{gwrankC}(C1, C2, N) \leftarrow \text{grank}(C1, C2), \text{wrankC}(C2, C1, N). \tag{16}$$
$$\text{gkt}(K) \leftarrow K = \#sum\{N, C1, C2 : \text{gwrankC}(C2, C1, N)\}. \tag{17}$$
% Preferred candidate must win
$$\leftarrow \text{preferredCand}(C), not\ \text{gpref}(1, C). \tag{18}$$

The manipulation action happens in Line 8, where a subset of the unregistered candidates is guessed. A preference is guessed in Line 13 and its Kendall's Tau distance to the vote is calculated in Line 17. The guessed preference must be such that the preferred candidate is the winner (Line 18) gpref/1 becomes the preference to beat in the check program, which we show next.

$$1\ \{\text{cpref}(Pos, C) : \text{candidate}(C)\}\ 1 \leftarrow \text{domain}(Pos). \tag{19}$$
$$\leftarrow \text{cpref}(Pos1, C), \text{cpref}(Pos2, C), Pos1! = Pos2. \tag{20}$$
% In the guessed preference relation C1 is better-ranked than C2
$$\text{crank}(C1, C2) \leftarrow \text{cpref}(Pos1, C1), \text{cpref}(Pos2, C2), Pos1 < Pos2. \tag{21}$$
% Number of votes that disagree on C1 being better-ranked than C2
$$\text{cwrankC}(C1, C2, N) \leftarrow \text{crank}(C1, C2), \text{wrankC}(C2, C1, N). \tag{22}$$
% Change $<$ to $\leq$ for stronger control (no multiple winners)
$$\leftarrow \text{gkt}(K), K\ \#sum\{N, C1, C2 : \text{cwrankC}(C2, C1, N)\}. \tag{23}$$
$$\leftarrow \text{preferredCand}(X), \text{cpref}(1, X). \tag{24}$$

In this check program, a different preference cpref/1 where the preferred candidate is *not* the winner is guessed. The purpose of this new preference is beating the gpref/1 preference guessed before. The integrity constraint in Line 23 achieves that by discarding any answer set where the Kendall's Tau distance of cpref/1 is at least that of gpref/1. Thus, unless gpref/1 minimizes the Kendall's Tau distance over all possible preferences, a preference cpref/1 can be guessed that has a smaller distance. It follows that the interplay between the guess and check programs effectively minimizes the Kendall's Tau distance of gpref/1.

Taking a closer look at the integrity constraint in Line 23, we can see where the choice of the winner model (in our case, the multiple winners model) plays a role. In our model, it is enough for the gpref/1 preference to have the same Kendall's Tau distance as the cpref/1 preference for the preferred candidate to win. In the unique winner model, a preference cpref/1 whose Kendall's Tau distance equals that of the gpref/1 would be enough to assume the preferred candidate is not a winner. To work under the unique winner model, it is enough to modify the integrity constraint in Line 23 to discard any preference cpref/1 whose distance is at least $K+1$, where K is the distance of gpref/1.

We now present the encoding for the guess part of Kemeny-CCDC. The same check program used for Kemeny-CCAC applies to the check part of Kemeny-CCDC.

$$\text{icandidate}(1..M) \leftarrow \text{candnum}(M). \tag{25}$$

% Guess a subset of candidates of size at least K fewer than the initial total number of candidates.

$$\text{candidate}(C) \leftarrow \text{preferredCand}(C). \tag{26}$$

$$X - K\ \{\text{candidate}(C) : \text{icandidate}(C)\}\ X \leftarrow \text{candnum}(X), \text{limit}(K). \tag{27}$$

$$\text{domain}(1..N) \leftarrow N = \{\text{candidate}(C)\}. \tag{28}$$

% Number of times candidate C1 is worse-ranked than candidate C2 in the preference profile

$$\text{wrank}(P, C1, C2) \leftarrow \text{p}(P, Pos1, C1), \text{p}(P, Pos2, C2), Pos1 > Pos2. \tag{29}$$

$$\text{wrankC}(C1, C2, N) \leftarrow \text{candidate}(C1), \text{candidate}(C2), N = \#sum\{VC, P : \text{prefVC}(P, VC), \text{wrank}(P, C1, C2)\}. \tag{30}$$

% Guess preference relation

$$1\ \{\text{gpref}(D, C) : \text{candidate}(C)\} \leftarrow \text{domain}(D). \tag{31}$$

$$\leftarrow \text{gpref}(Pos1, C), \text{gpref}(Pos2, C), Pos1 != Pos2. \tag{32}$$

% In the guessed preference relation C1 is better-ranked than C2

$$\text{grank}(C1, C2) \leftarrow \text{gpref}(Pos1, C1), \text{gpref}(Pos2, C2), Pos1 < Pos2. \tag{33}$$

% Number of votes that disagree on C1 being better-ranked than C2

$$\text{gwrankC}(C1, C2, N) \leftarrow \text{grank}(C1, C2), \text{wrankC}(C2, C1, N). \tag{34}$$

$$\text{gkt}(K) \leftarrow K = \#sum\{N, C1, C2 : \text{gwrankC}(C2, C1, N)\}. \tag{35}$$

% Preferred candidate must win

$$\leftarrow \text{preferredCand}(C), not\ \text{gpref}(1, C). \tag{36}$$

This encoding is similar to that of Kemeny-CCAC except that instead of guessing a subset of unregistered candidates to add to the election, a subset of the initial candidates is dropped from the election. To do this, initial candidates are marked by the icandidate/1 predicate as part of the input, while the candidates that will ultimately participate in the election are marked by the candidate/1 predicate in Line 27. Note that the previous line guarantees that the preferred candidate is part of the election.

The next pair of programs show respectively the guess and check parts of Young-CCAC.

$$\text{candidate}(1..M) \leftarrow \text{rcandnum}(M). \tag{37}$$
$$\text{ucandidate}((\text{M}+1)..(M+N)) \leftarrow \text{rcandnum}(M), \text{ucandnum}(N). \tag{38}$$
% Guess a subset of unregistered candidates of size at most K to add.
$$\{\text{candidate}(C) : \text{ucandidate}(C)\}\ K \leftarrow \text{limit}(K). \tag{39}$$
% Guess the number of kept votes per preference relation
$$\text{keepMax}(0..V) \leftarrow \text{voternum}(V). \tag{40}$$
$$1\ \{\text{gkeep}(P,X) : X <= VC, \text{keepMax}(X)\}\ 1 \leftarrow \text{prefVC}(P,VC). \tag{41}$$
% Count the number of kept voters
$$\text{gcountKeep}(N) \leftarrow N = \#sum\{X, P : \text{gkeep}(P,X)\}. \tag{42}$$
% Compute Condorcet winner (considering only kept votes)
$$\text{prefer}(P,C1,C2) \leftarrow \text{ip}(P,Pos1,C1), \text{ip}(P,Pos2,C2), Pos1 < Pos2. \tag{43}$$
$$\text{gpreferCount}(C1,C2,N) \leftarrow \text{candidate}(C1), \text{candidate}(C2), C1! = C2, N = \#sum\{X, P : \text{gkeep}(P,X), \text{prefer}(P,C1,C2)\}. \tag{44}$$
$$\text{noGWinner}(C) \leftarrow \text{gpreferCount}(C,\_,N), \text{gcountKeep}(X), N*2 < X. \tag{45}$$
$$\text{gwinner}(C) \leftarrow \text{candidate}(C), not\ \text{noGWinner}(C). \tag{46}$$
% Keep answer-set only in case there is a Condorcet winner (with removing votes)
$$anyGWinner \leftarrow \text{gwinner}(\_). \tag{47}$$
$$\leftarrow not\ anyGWinner. \tag{48}$$
% Ensure the preferred candidate is the gwinner
$$\leftarrow \text{preferredCand}(X), not\ \text{gwinner}(X). \tag{49}$$

Guessing the subset of unregistered candidates to add is achieved in the same way as in the Kemeny-CCAC encoding. The next part of the encoding deals with the deletion of votes. Because votes are specified as pairs of a profile P and the number of voters that voted P (the predicate prefVC/2), we guess in Line 41 a number of votes to keep per profile P, which cannot exceed the original vote count for that profile. From Lines 43 to 46 the (weak) Condorcet winners are calculated but Line 48 ensures there is at least one winner for this guess. Finally, Line 49 guarantees that for this guess of added candidates and deleted votes, the preferred candidate is a winner.

$$1\ \{\text{ckeep}(P,X) : X <= VC, \text{keepMax}(X)\}\ 1 \leftarrow \text{prefVC}(P,VC). \tag{50}$$
% Count the number of kept voters
$$\text{ccountKeep}(N) \leftarrow N = \#sum\{X, P : \text{ckeep}(P,X)\}. \tag{51}$$
% Compute Condorcet winner (considering only kept votes)
$$\text{cpreferCount}(C1,C2,N) \leftarrow \text{candidate}(C1), \text{candidate}(C2), C1! = C2, N = \#sum\{X, P : \text{ckeep}(P,X), \text{prefer}(P,C1,C2)\}. \tag{52}$$
$$\text{noCWinner}(C) \leftarrow \text{cpreferCount}(C,\_,N), \text{ccountKeep}(X), N*2 < X. \tag{53}$$
$$\text{cwinner}(C) \leftarrow \text{candidate}(C), not\ \text{noCWinner}(C). \tag{54}$$
% Keep answer-set only in case there is a Condorcet winner (with removing votes)
$$anyCWinner \leftarrow \text{cwinner}(\_). \tag{55}$$
$$\leftarrow not\ anyCWinner. \tag{56}$$
% Ensure the preferred candidate is not a cwinner
$$\leftarrow \text{preferredCand}(X), \text{cwinner}(X). \tag{57}$$
% Ensure ckeep keeps strictly more than gkeep
$$\leftarrow \text{gcountKeep}(Kg), \text{ccountKeep}(Kc), Kc <= Kg. \tag{58}$$

The check program for Young-CCAC guesses a different deletion of votes for which the (weak) Condorcet winners are calculated. In this program, the preferred candidate must not be one of these winners, which is guaranteed by the constraint in Line 57. Line 58 ensures the guess in the check program deletes strictly less votes than that of the guess program. Two simple modifications can be made in order to adapt these programs to other Young

elections: for the single winner model, the constraint in Line 58 can be relaxed to require the number of deleted votes in the check program to be less or equal to that of the guess program; and to consider strong Condorcet winners the only change needed is to use "less than or equal to" in the criteria at Lines 45 and 53, since this guarantees that candidates will only be marked as winners if they actually beat every other candidate.