CAPSTONE PROJECT FINAL PAPER

CROIX GYUREK (ADVISED BY DR. MOHAMMAD AL HASAN)

Abstract. I prove two computational hardness results for problems in the domain of multi-winner approval voting. First, I define the notion of core stability, a property of voting methods borrowed from game theory, and show that verifying core stability is co-NP-complete. Then I present a failed attempt to prove that finding a core-stable winner set is NP-hard, and discuss my experience with that problem. Finally, I prove that for the voting method known as Sequential Proportional Approval Voting, which is computable in polynomial time if a tiebreak scheme is given, it is NP-hard to determine if there exists some tiebreak order can elect a committee with a given total score, thus rendering the method vulnerable to delays if multiple rounds are close to being tied.

1. Introduction

For about seven or eight years I have been fascinated with voting and elections, particularly the very mathematical area of alternate voting methods like approval voting, instant-runoff voting, and the entire family of Condorcet methods. As I have been thinking about the broken nature of U.S. politics, the concept of proportional representation as a way to fix gerrymandering stands out as a good reform, although I still believe in allowing voters to vote for candidates, rather than parties, to allow greater accountability. Among the possible ways to do this, approval ballots (where voters can either "approve" or "disapprove" each candidate independently) seem like the simplest possible ballot design.

My work has been influenced largely by the soon-to-be-printed book by Lackner and Skowron, [3]. They define the multi-winner approval voting environment in the following way:

Definition 1.1. *A* k-winner approval voting election*, hereafter simply "*approval election*", is a quadruple* $E = (V, C, A, k)$ *where* V *is a set of* voters*,* C *a set of* candidates*,* A *a function* $A : V \to \mathcal{P}(C)$*, and* k *is a positive integer less than or equal to* $|C|$*. For any voter* $v \in V$*, we call* $A(v)$ *the* ballot *for* v*. A subset* $W \subseteq C$ *with* $|W| = k$ *is a* winner set *for* E*.*

It remains to specify exactly how the winners are determined. The most obvious way to determine the winners is to elect the candidates with the most approvals:

Definition 1.2. Naïve Approval Voting *gives each candidate a score based on the number of voters who approve that candidate:* $S(c) = |\{v \in V : c \in A(v)\}|$*. The winners are the* k *candidates with the highest values of* $S(c)$*.*

In certain situations this rule is sufficient, but in political environments aiming for representation of diverse groups, this is strongly biased towards the largest faction. For example, if $k = 5$ and the Purple and Orange parties each run 5 candidates, 60% of voters approve all Purple candidates, and 40% approve all Orange candidates, then all five winners will be Purple. This lack of representation is the motivation behind voting methods that aim to elect a "proportional" winner set. In a purely partisan election, such as the example above, a selection of 3 Purple and 2 Orange winners is clearly "proportional". However, in a general approval election, where voters may vote as they please, this is harder to define.

A more "proportional" voting method using approval ballots can be created by, intuitively speaking, giving more weight to voters who approved fewer winners, since they are less represented. This can be formalized as follows:

Definition 1.3. *Given an approval election* (V, C, A, k)*, for a given winner set W, the* proportional satisfaction *of a voter v approving exactly m winners is* $p_W(v) = \sum_{i=1}^{m} \frac{1}{i}$.

Proportional Approval Voting *(PAV) elects the winner set W that maximizes the total proportional satisfaction of the electorate, that is,* $W = \operatorname{argmax}_S \sum_{v \in V} p_S(v)$*, where S ranges over k-sized subsets of C.*

This method has been shown to be NP-hard to compute the winner set, see [1]. However, it does satisfy several formalizations of "proportionality", such as Extended Justified Representation (see [3], p. 60). A similar, and computationally faster, rule can be implemented by using a greedy algorithm to pick winners one at a time:

Definition 1.4. Sequential Proportional Approval Voting *(SeqPAV) begins with* $W_0 = \emptyset$*. For each* $i = 1, \ldots, k$*, the next winner* w_i *is the un-elected candidate who, along with* $\{w_j : 1 \leq j < i\}$*, provides the greatest proportional satisfaction:*

$$w_i = \operatorname*{argmax}_{c \in C \setminus W_{i-1}} \sum_{v \in V} p_{W_{i-1} \cup \{c\}}(v)$$

The final winner set is $W = \{w_i\}_{i=1}^{k}$.

This method can be restated in the following terms: All voters start with a "weight" of 1. The first winner, w_1, is the candidate approved by the most voters. For the second roung, all voters who approved w_1 have their weight set to $\frac{1}{2}$, and the candidate with the most *weighted* approvals is w_2. This process continues, where a voter who approved r winners has a weight of $\frac{1}{1+r}$ (and weights are recalculated after each new candidate is elected), until all k winners have been selected. I will return to SeqPAV later.

2. Core Stability

One way to formulate the concept of proportionality is to require that any fraction α of the electorate be able to "control" at least $\lfloor \alpha k \rfloor$ winning candidates. The notion of "the core", or "core stability", defined in [3] defines "control" assuming that a voter *prefers* a set of candidates T_1 over T_2 if that voter approves (strictly) more candidates in T_1 than T_2. Core stability, then, requires that no group of voters that is α of the population can find a set T of $\lfloor \alpha k \rfloor$ candidates they unanimously prefer over the actual winners W. More precisely:

Definition 2.1 ([3])**.** *Given an election* $E = (V, C, A, k)$*, we say that a winner set* $W \subseteq C$*, where* $|W| = k$*, is* core-stable*, or* in the core*, if there does not exist* $T \subseteq C$ *and* $V^* \subseteq V$ *such that:*

(1) $\frac{|V^*|}{|V|} \geq \frac{|T|}{k}$*, and*
(2) *for every voter* $v \in V^*$*,* $|A(v) \cap T| > |A(v) \cap W|$ *(that is, all voters in* V^* *prefer T to W).*

If such a T and V^* *exist, then T and* V^* witness *that W is not core-stable.*

Note that the sets W and T are **not** required to be disjoint. This will be important in the proof of Theorem 3.3, where the sets W and T constructed will have a quite large intersection. According to [3], it is not known whether a core-stable W exists for every election E or not.

3. Core Stability is co-NP-Complete

Definition 3.1. *The* Verify-Core-Stable *problem: Given an Approval Election* (V, C, A, k) *and a winner set* $W \subseteq C$ *with* $|W| = k$*, is W core-stable?*

To prove this is co-NP-complete, we will reduce the NP-complete ([2]) Dominating Set problem to it, which asks for subsets of vertices of a graph that are adjacent to every vertex.

Definition 3.2. *The* Dominating-Set *problem: Given a graph $G = (V_G, E_G)$ and a positive integer t, does there exist a subset of vertices $Y \subseteq V_G$ such that $|Y| = t$ and for every vertex $x \in V_G \setminus Y$ there is a vertex $y \in Y$ with $\{x, y\} \in E_G$?*

Theorem 3.3. Verify-Core-Stable *is co-NP-complete.*

Proof. First, it is clear that Verify-Core-Stable is in co-NP, because if (V^*, T) witnesses that W is not core-stable, then this can be checked in polynomial time, by simply iterating over all voters $v \in V^*$ and checking whether or not $|A(v) \cap T| > |A(v) \cap W|$.

Now, to show that Verify-Core-Stable is co-NP-hard, we use a reduction from the Dominating Set Problem. Given an instance (V_G, E_G, t) of Dominating-Set, let n be the number of vertices in V_G, and create an election instance as follows:

- The **voters** are $\{v_x : x \in X\}$, one voter for each vertex.
- The **candidates** come in three types. There are n^2 universally approved candidates $U = \{u_j\}_{j=1}^{n^2}$, plus one candidate for each vertex, $\{c_x : x \in X\}$, plus t dummy candidates $\{d_j\}_{j=1}^{t}$ approved by no voters.
- The **ballot** for voter v_x approves all of the universally approved candidates, as well as the vertex candidates adjacent to or equal to x: $A(v_x) = \{c_y : y \in V_G, x = y \text{ or } \{x, y\} \in E_G\} \cup U$.
- Finally, we set the **winner set** $W = \{c_j\}_{j=1}^{n^2} \cup \{d_j\}_{j=1}^{t}$.

Here, $|V| = n$ and $k = n^2 + t$. Currently, each voter approves exactly n^2 winners. For W to not be core-stable, it is necessary to find $T \subseteq C$ and $V^* \subseteq V$ such that $|V^*|k \geq |T||V|$, that is, $|V^*|(n^2 + t) \geq |T|n$. Since $|A(v_x) \cap T| \leq |T|$, we need that $|T| \geq n^2 + 1$. Hence, $|V^*| \geq \frac{n^3+n}{n^2+t} = \frac{n+1/n}{1+t/n^2}$. The reader can verify, since $t < n$, that this fraction is greater than $n - 1$ and so $|V^*|$ must equal $|V| = n$.

At this point, we require that every voter approves at least $n^2 + 1$ candidates in T. We can get n^2 of these by using $U = \{u_j\}_{j=1}^{n^2}$. After this, we can select at most t additional candidates. The only way to ensure each voter approves at least one extra candidate from T is to select candidates from c. The condition that each voter v_x approves some $c_y \in T$ is equivalent to the condition that the set $Y = \{y : c_y \in T\}$ is a dominating set for X, since v_x approves c_y iff $x = y$ or $x \sim y$ in the graph. Also, since we must have $|T| \leq k$, it follows that $|Y| \leq k - |U| = (n^2 + t) - n^2 = t$. Hence, if T witnesses that W is not core-stable, then $U \subseteq T$ and $Y = \{y : c_y \in T\}$ is a dominating set for X of size at most t. The converse is also true: if Y is a t-sized dominating set for X, then $T = U \cup \{c_y : y \in Y\}$ witnesses that W is not core-stable. This finishes the proof. □

4. Finding Core-Stable Winner Sets

Conjecture 4.1. *Given an election (V, C, A, k) and a subset $C^* \subseteq C$, it is NP-hard to determine if there exists a core-stable $W \subseteq C^*$.*

I was unable to prove this. The closest that I got was attempting the following weaker result:

Definition 4.2. *Given an election $E = (V, C, A, k)$ and an integer $q \leq k$, a winner set $W \subseteq C$ is* q-core-stable *if there does not exist $T \subseteq C$ such that $|T| = q$ and T witnesses that W is core-unstable.*

Conjecture 4.3. *For any fixed integer $q \geq 3$, the following problem is NP-hard: Given an election E, a subset of candidates $C^* \subseteq C$, determine if there exists a q-core-stable $W \subseteq C^*$.*

Idea (not a proof). We would reduce this to the q-exact cover problem: given the set $X = \{1, 2, \ldots, qn\}$ and qn sets $S_j = \{x_{j,1}, \ldots, x_{j,q}\} \subseteq X, 1 \leq j \leq qn$, determine if there exists a subcollection of n sets, $\{S_{j_i} : 1 \leq i \leq q\}$, whose union is X. (Since all sets have cardinality q, this would have to be a disjoint union if it equals X.)

The election would be created as follows, where the question marks represent numbers that may need to be changed to make the argument work:

- The **voters** are copies of the numbers, $\{v_{1,i}, \ldots, v_{nq,i} : 1 \leq i \leq ?\}$.
- The $q(n+2)$ **candidates** are the sets, $\{S_j\}_{j=1}^{qn}$, as well as some number of extra candidates $\{d_j\}_{j=1}^{?}$.
- The **ballots** are $A(v_{x,i}) = \{S_j : x \in S_j\} \cup \{d_j\}_{j=1}^{q}$. In other words, each voter approves the sets it is contained in, and the first q of the d candidates.
- We let C^* be the collection $\{S_j\}_{j=1}^{nq} \cup \{d_j\}_{j=1}^{q-1}$.

Finally, we set the number of winners k to... something sufficiently large. The idea is that if $\mathcal{A}$ is a collection of n sets that covers X exactly, then all voters will approve the correct number of candidates. However, I seem to have failed because it is possible to miss only one number, rather than needing to miss q numbers.

I tried a similar argument with the voters being sets, but this was near impossible to convert into the actual problem which requires a search for candidate sets. Hence, this conjecture remains unresolved. □

This does not truly prove that Find-Core-Stable is NP-hard, however. For one, although "does there exist a core-stable $W \subseteq C^*$" is the conjunction of "does there exist a q-core-stable $W \subseteq C^*$" for all $1 \leq q \leq k$, there is no reason that the conjunction of NP-hard problems must be NP-hard. I have not fully investigated the process of converting find problems into decision problems, other than that each should be reducible to the other. If there were a polynomial-time algorithm for the problem in Conjecture 4.1, then it could be used to solve Find-Core-Stable by repeatedly querying increasing subsets of C until the answer becomes yes, permanently placing the last candidate added in C^*, and then starting over. This uses fewer than $k|C|$ runs of the assumed polynomial-time algorithm, so Find-Core-Stable would be in P. The contrapositive is this:

Proposition 4.4. *If* Find-Core-Stable *is NP-hard, then Conjecture 4.1 is true.*

Unfortunately, this is not the direction of implication I want. I was inspired by the link between the Subset Sum decision and finding problems.[1] Given a decision Subset Sum oracle, if it answers yes for a given set of numbers, we can find a subset in linear time by recursively calling the oracle on $S \setminus \{\max S\}$; if the answer is no, that maximum was essential, and if

The difficulty I ran into with Conjecture 4.1 is that solving the find-problem does not immediately solve the C^* problem, since the found committee may be outside of C^*. In fact, there seem to be very few NP-hard find problems that are guaranteed to have a solution. (It may well be that core-stable winner sets always exist.) The closest I can think of is the discrete logarithm problem (given positive integers (g, n, b) with n prime and g a primitive root of n, find a such that $g^a \equiv b \mod n$); it is quickly verifiable that n is prime and g is a primitive root, so such an a must exist, but determining the a is at least hard enough to base our entire digital livelihoods on (although it has not been proven NP-complete since it is in BQP).

5. Sequential Proportional Approval Voting

Story: *After a while of realizing what I had gotten myself into, trying to reason about something that was intuitively "doubly hard" (it involved an exponential number of exponential checks), I decided to find a lower hanging fruit based on a specific voting method, so that my results can be more concrete. For this, I turned back to [3]'s list of open problems, one of which involved Thiele methods.*

Math: As we saw earlier, the Proportional Approval Voting method has nice theoretical properties but is NP-complete to determine the winner of. A compromise method called Sequential PAV (SeqPAV) exists that is clearly computable in polynomial time, assuming that elections are not tied.

[1] Decision problem: Given a set S of integers and an integer m, determine whether or not there exists a subset $T \subseteq S$ whose sum is exactly m. The finding problem asks to explicitly find T.

In real elections, exact ties (in, for instance, U.S. federal and statewide elections) are incredibly rare because the number of voters is very large [citation needed]. However, if the election *were* tied (or close to tied) in an early round, the multi-round nature of SeqPAV would delay the count until the correct round winner could be determined.[2]

It would be convenient if there were an algorithm to determine all possible results coming out of ties, or more precisely, *near-ties* where candidates' scores differ by a small enough margin to require a recount. (The "tiebreak order" represents the relatively small quantity of extra votes that ultimately decides the election.) This way, as soon as the *exact* vote counts are confirmed, the winning committee can be determined immediately, without having to manually re-count the remaining rounds. Unfortunately, this can theoretically result in an exponential number of tiebreak possibilities, requiring an exponential number of recounts if done upfront. For a very simple example, consider the following SeqPAV election:

Example 5.1. *Let there be 9 voters $V = \{v_1, v_2, \ldots, v_9\}$, 4 candidates $C = \{a, b, c, d\}$, and 2 winners to be elected. Define the ballots as follows:*

$$A(v_1) = \{a, b\} \quad A(v_3) = \{a, b, c\} \quad A(v_5) = \{a, b\} \quad A(v_7) = \{c, d\} \quad A(v_9) = \{d\}$$
$$A(v_2) = \{a, b\} \quad A(v_4) = \{a, c, d\} \quad A(v_6) = \{b, c\} \quad A(v_8) = \{d\}$$

Under SeqPAV, in the first round, the scores for (a, b, c, d) respectively are $(5, 5, 4, 4)$. We have a tie between a and b. If the tie is broken in favor of a, then voters v_1 through v_5 have their weights reduced to $\frac{1}{2}$, and the score gains in the second round become $(0, \frac{4}{2}+1, \frac{1}{2}+3, \frac{2}{2}+2) = (0, 3, \frac{7}{2}, 3)$, so c wins the second seat (a's gain is zero since a was already elected). However, if the tie is broken in favor of b, then it is v_1, v_2, v_3, v_5, v_6 who lose influence, so the score gains are $(\frac{4}{2}+1, 0, \frac{1}{2}+3, 4)$, resulting in a tie between c and d.

In this example, the tiebreak only affected one candidate, but if there were more than two winners there could potentially be a tie or near-tie in each round, resulting in many different final outcomes. This "chaos" effect can be made precise with the following hardness result.

Theorem 5.2. *The following problem is NP-complete: Given an approval election E and score $s \in \mathbb{Q}$, determine whether or not there exists a winner set W with PAV score at least s, such that W can be obtained from applying the SeqPAV algorithm to E for some tie-breaking order τ.*

Proof. This problem is certainly in NP, since one must merely specify the order τ; the SeqPAV algorithm with tiebreak order τ can be computed in polynomial time and the final score checked.

To show hardness, we reduce from the NP-complete ([2]) Independent Set problem. Given a graph $G = (V_G, E_G)$ and integer t, we construct an approval election with $k = t$ winners as follows:

- Let d be the maximum degree of vertices in G.
- The **candidates** are the vertices of G: $C := V_G$.
- The **voters** come in two forms: the edges E_G, and additional voters for each vertex to being the total number of voters on each vertex to d: $\{v_{x,j} : x \in V_G, 1 \leq j \leq d - deg(x)\}$.
- The **ballots** are as follows: Edge voters approve the two candidates they connect, $A(\{x, y\}) = \{x, y\}$. Vertex voters $v_{x,j}$ approve only $\{x\}$. Hence, all candidates are approved by exactly d voters.
- Finally, the **target score** is $s = dk$.

The only way a winner set W can reach the score dk is if the candidates in W each receive support from a *different* set of d voters. This will happen if and only if no edge voter approves two distinct candidates of W, i.e. W is an independent set in G. Furthermore, if W_j is a partial winner

[2] A similar effect happened in the 2022 U.S. House of Representatives election, which used instant-runoff voting, another multi-round system. The first-round vote counts of Begich and Palin were close, and the choice of which one to eliminate first could have determined whether or not the other could beat Rep. Peltola.

set of size $j < k$, then for any candidate x, the increase in PAV score from electing x will be d if and only if no voter approving x also approves a candidate in W_j. Since the only voters who approve x are the vertex and edge voters, this is equivalent to x not being part of any edge including W_j, i.e. x is not adjacent to any vertex (candidate) in W_j. Hence, under any tiebreaking rule in which x takes priority over all candidates not in or adjacent to W_j,

Thus, W is an independent set if and only if W can be elected under SeqPAV with some tiebreaking rule. □

6. Conclusions

The result on co-NP-hardness was already known, but I was unable to access the proof due to a citation error and decided to independently prove it myself. The SeqPAV result is more interesting; has practical importance because it appears to destroy the purpose of SeqPAV as an "efficient" approximation to PAV. Although with perfect, instant tabulation and a fixed tiebreak order (or a random oracle, if tiebreaks are random) SeqPAV is in P, the messy, noisy, heuristic nature of real-world elections (at least in the US) always threatens to introduce multiple layers of delays. Personally, I almost wonder if it would be better to use PAV with a manageable number of winners (like 3 or 5) with algorithms[3] to prune the $O(|C|^k)$ search space, which can be engineered ahead of time, than implement SeqPAV and watch it delay an entire area's representation from being seated.

Of course, PAV can theoretically result in ties as well. However, assuming the entire data is collected at once, this still only requires one round of recounting. Another question concerns randomly selecting a fair winner, since determining the probability distribution is almost certainly NP-hard. This could be circumvented by creating a primary tiebreak, invariant under candidate permutations, with the property that any ties that persist must be of the form $\{W \subseteq T : |W| = k\}$ for some subset T of the candidates.

References

[1] Haris Aziz et al. *Computational Aspects of Multi-Winner Approval Voting*. 2014. doi: 10.48550/ARXIV.1407.3247. url: https://arxiv.org/abs/1407.3247.

[2] Michael R Garey and David S Johnson. "Computers and intractability". In: *A Guide to the* (1979).

[3] Martin Lackner and Piotr Skowron. *Multi-Winner Voting with Approval Preferences*. Springer International Publishing, 2023. doi: 10.1007/978-3-031-09016-5. url: https://doi.org/10.1007/978-3-031-09016-5.

[3] For instance, if a_k is the number of approvals that the k-th most approved candidate gets, it is easy to show that no candidate with fewer than $\frac{a_k}{k}$ approvals can possibly be elected. Hence, a large number of write-in or poorly known candidates could be eliminated immediately. With an algorithm like branch-and-bound this principle can be applied recursively to sub-cases.