ECAI 2012
Luc De Raedt et al. (Eds.)

doi:10.3233/978-1-61499-098-7-612

Combining Voting Rules Together

Nina Narodytska[1] and **Toby Walsh**[2] and **Lirong Xia**[3]

Abstract. We propose a simple method for combining together voting rules that performs a run-off between the different winners of each voting rule. We prove that this combinator has several good properties. For instance, even if just one of the base voting rules has a desirable property like Condorcet consistency, the combination inherits this property. On the other hand, some important properties can be lost by the introduction of a run-off, including monotonicity and consistency. In addition, we prove that combining voting rules together in this way can make finding a manipulation more computationally difficult.

1 INTRODUCTION

An attractive idea in the Zeitgeist of contemporary culture is "The Wisdom of Crowds" [9]. This is the idea that, by bringing together diversity and independence of opinions, groups can be better at making decisions than the individuals that make up the group. For example, in 1907 Galton observed the wisdom of the crowd at guessing the weight of an ox in the West of England Fat Stock and Poultry Exhibition. The median of the 787 estimates was 1207 lb, within 1% of the correct weight of 1198 lb. We can view different voting rules as having different opinions on the "best" outcome to an election. We argue here that it may pay to combine these different opinions together. We provide theoretical evidence for this thesis. We argue that a combination of voting rules can inherit a desirable property like Condorcet consistency when only one of the base voting rules is itself Condorcet consistent. We also prove that combining voting rules together can make strategic voting more computationally difficult.

1.1 RELATED WORK

Different ways of combining voting rules to make manipulation computationally hard have been investigated recently. Conitzer and Sandholm [4] studied the impact on the computational complexity of manipulation of adding an initial round of the Cup rule to a voting rule. This initial round eliminates half the candidates and makes manipulation NP-hard to compute for several voting rule including plurality and Borda. Elkind and Lipmaa [7] extended this idea to a general technique for combining two voting rules. The first voting rule is run for some number of rounds to eliminate some of the candidates, before the second voting rule is applied to the candidates that remain. They proved that many such combinations of voting rules are NP-hard to manipulate. Note that theirs is a sequential combinator, in which the two rules are run in sequence, whilst ours (as we will see soon) is a parallel combinator, in which the two rules are run in parallel. More recently, Walsh and Xia [10] showed that using a lottery to eliminate some of the voters (instead of some of the candidates) is another mechanism to make manipulation intractable to compute.

[1] NICTA and UNSW, Sydney, Australia, email: ninan@cse.unsw.edu.au
[2] NICTA and UNSW, Sydney, Australia, email: toby.walsh@nicta.com.au
[3] Harvard University, Cambridge, MA, USA, email: lxia@seas.harvard.edu

2 VOTING RULES

Voting is a general mechanism to combine together the preferences of agents. Many different voting rules have been proposed over the years, providing different opinions as to the "best" outcome of an election. We formalise voting as follows. Let $C = \{c_1, \ldots, c_m\}$ be the set of *candidates* (or *alternatives*). A *profile* P is a sequence of n total orders over m candidates C. That is, $P = (V_1, \ldots, V_n)$, where for every $j \leq n$, V_j is a total order on C: $V_j = (c_{j_1} \succ \ldots \succ c_{j_m})$. A *voting rule* is a function mapping a profile P onto one candidate, the *winner*. Let $N_P(i, j)$ be the number of voters preferring i to j in P. Where P is obvious from the context, we write $N(i, j)$. Let $beats(i, j)$ be 1 iff $N(i, j) > \frac{n}{2}$ and 0 otherwise. We consider some of the most common voting rules.

• **Positional scoring rules:** Given a *scoring vector* $(w_1, \ldots, w_m)$ of weights, where $w_1 \geq w_2 \geq \cdots \geq w_m$ and $w_1 > w_m$, the ith candidate in a vote scores w_i, and the winner is the candidate with highest total score over all the votes. The **plurality** rule has the weight vector $(1, 0, \ldots, 0)$, the **veto** rule has the vector $(1, 1, \ldots, 1, 0)$, the **k-approval** rule has the vector $(1, \ldots, 1, 0 \ldots, 0)$ containing k 1s, and the **Borda** rule has the vector $(m - 1, m - 2, \ldots, 0)$.

• **Cup:** The winner is the result of a series of pairwise majority elections between candidates. Given the *agenda*, a binary tree in which the roots are labelled with candidates, we label the parent of two nodes by the winner of the pairwise majority election between the two children. The winner is the label of the root.

• **Copeland:** The candidate with the highest Copeland score wins. The Copeland score of candidate i is $\sum_{j \neq i} beats(i, j)$. The Copeland winner is the candidate that wins the most pairwise elections.

• **Maximin:** The maximin score of candidate i is $\min_{j \neq i} N(i, j)$. The candidate with the highest maximin score wins.

• **Single Transferable Vote (STV):** This rule requires up to $m - 1$ rounds. In each round, the candidate with the least number of voters ranking his/her first is eliminated until one of the remaining candidates has a majority.

• **Bucklin (simplified):** The Bucklin score of a candidate is the smallest k such that the k-approval score of the candidate is strictly larger than $n/2$. The candidate with the smallest Bucklin score wins.

Note that in some cases, there can be multiple winning candidates (e.g. multiple candidates with the highest Borda score). We therefore may also need a tie-breaking mechanism. All above voting rules can be extended to choose a winner for profiles with weights. In this paper, we study the *manipulation* problem (with weighted votes), defined as follows.

Definition 1 *In a* manipulation *problem, we are given an instance*

$(r, P^{NM}, \vec{w}^{NM}, c, k, \vec{w}^{M})$, where r is a voting rule, P^{NM} is the non-manipulators' profile, $\vec{w}^{NM}$ represents the weights of P^{NM}, c is the alternative preferred by the manipulators, k is the number of manipulators, and $\vec{w}^{M} = (w_1, \ldots, w_k)$ represents the weights of the manipulators. We are asked whether there exists a profile P^{M} of indivisible votes for the manipulators such that $c \in r((P^{NM}, P^{M}), (\vec{w}^{NM}, \vec{w}^{M}))$.

When all weights equal to 1, the problem is called manipulation with unweighted votes. In this paper, we assume that the manipulators controls the tie-breaking mechanism, that is, all ties are broken in favor of c.

A large number of normative properties that voting rules might possess have been put forward including the following.

• **Unanimity:** If a candidate is ranked in the top place by all voters, then this candidate wins.

• **Monotonicity:** If we move the winner up a voter's preference order, while keeping preferences unchanged, then the winner should not change.

• **Consistency:** If two sets of votes select the same winner then the union of these two sets should also select the same winner.

• **Majority criterion:** If the majority of voters rank a same candidate at the top, then this candidate wins.

• **Condorcet consistency:** If a *Condorcet winner* exists (a candidate who beats all others in pairwise elections) then this candidate wins.

• **Condorcet loser criterion:** If a *Condorcet loser* exists (a candidate who is beaten by all others in pairwise elections) then this candidate does not win.

Such properties can be used to compare voting rules. For example, whilst STV satisfies the majority criterion, Borda does not. On the other hand, Borda is monotonic but STV is not.

3 VOTING RULE COMBINATOR

We consider a simple combinator, written $+$, for combining together two or more voting rules. This combinator collects together the set of winners from the different rules. If all rules agree, this is the overall winner. Otherwise we recursively call the combination of voting rules on the original profile that is restricted to this set of winning candidates. If the recursion does not eliminate any candidates, we call some tie-breaking mechanism on the remaining candidates. For example, $plurality + veto$ collects together the plurality and veto winners of an election. If they are the same candidate, then this is the winner. Otherwise, there is a runoff in which we call $plurality + veto$ on the plurality and veto winners. As both plurality and veto on two candidates compute the majority winner, the overall winner of $plurality + veto$ is the winner of a majority election between the plurality and veto winners.

This combinator has some simple algebraic properties. For example, it is idempotent and commutative. That is, $X + X = X$ and $X + Y = Y + X$. It has other more complex algebraic properties. For example, $(X + Y) + X = X + Y$.

In addition, many normative properties are inherited from the base voting rules. Interestingly, it is sometimes enough for just one of the base voting rules to have a normative property for the composition to have the same property.

Proposition 1 *For unanimity, the majority criterion, Condorcet consistency, and the Condorcet loser criterion, if one of X_1 to X_k and the tie-breaking mechanism satisfy the property, then $X_1 + \ldots + X_k$ also satisfy the same property.*

On the other hand, there are some properties which can be lost by the introduction of a run-off.

Proposition 2 (Monotonicity) *$Plurality$ and $Borda$ are both monotonic but $plurality + Borda$ is not.*

Proof: Suppose we have 6 votes for $b \succ c \succ a$, 4 votes for $c \succ a \succ b$, and 3 votes for $a \succ b \succ c$ and 3 votes for $a \succ c \succ b$. Tie-breaking for both Borda and plurality is $c \succ a \succ b$. Now c is the Borda winner and a is the plurality winner. By tie-breaking, c wins the run-off. However, if we modify one vote for $a \succ c \succ b$ to $c \succ a \succ b$, then b becomes the plurality winner and wins the run-off. Hence, $plurality + Borda$ is not monotonic. □

We give a stronger result for consistency. Scoring rules are consistent, but the combination of any two different scoring rules is not. By "different rules" we mean that there exists a profile for which these two rules select different winners, each of which has the strictly highest score. If two scoring rules are different, then their scoring vectors must be different. We note that the reverse is not true.

Proposition 3 (Consistency) *Let X and Y be any two different scoring rules, then $X + Y$ is not consistent.*

Proof: Let $s(P, a)$ and $r(P, a)$ be the score given to candidate a by X and Y in profile P respectively. Since X and Y are different, there exists P_1 over a_1 to a_m such that X on P_1 selects a_1 and Y on P_1 selects a_2. Then $s(P_1, a_1) > s(P_1, a_2)$ but $r(P_1, a_1) < r(P_1, a_2)$. WLOG suppose a_1 beats a_2 in pairwise elections in P_1 and tie breaking elects a_1 in favour of a_2 when they have the same top score. Let P_2 consist of m votes V_1 to V_m where for $i < m$, V_i ranks a_2 in ith place and a_1 in $i + 1$th place, and V_m ranks a_1 in 1st place and a_2 in last place. Then $s(P_2, a_1) = s(P_2, a_2)$ and $r(P_2, a_1) = r(P_2, a_2)$. Let k be such that $k(r(P_1, a_2) - r(P_1, a_1)) > r(V_m, a_1) - r(V_m, a_2)$, and P_3 be the following profile of cyclic permutations: $a_1 \succ a_2 \succ a_3 \succ \ldots \succ a_m$, $a_1 \succ a_2 \succ a_4 \succ \ldots \succ a_3$, …, $a_1 \succ a_2 \succ a_m \succ \ldots \succ a_{m-1}$, $a_2 \succ a_1 \succ a_3 \succ \ldots \succ a_m$, $a_2 \succ a_1 \succ a_4 \succ \ldots \succ a_3$, …, $a_2 \succ a_1 \succ a_m \succ \ldots \succ a_{m-1}$. Let P_4 be k copies of P_2, and P_5 be km copies of P_1, V_m and $km|P_1|$ copies of P_3. Now $X + Y$ on P_4 or P_5 selects a_1 as winner. But $X + Y$ on $P_4 \cup P_5$ selects a_2. □

The next proposition characterizes pairs of different positional scoring rules.

Proposition 4 *Let s and r be two positional scoring rules whose scoring vectors are $\vec{s} = (s_1, \ldots, s_m)$ and $\vec{r} = (r_1, \ldots, r_m)$ respectively. s and r are different if and only if there does not exist $\alpha, \beta \in \mathbb{R}$ where $\alpha > 0$ such that $\vec{s} = \alpha\vec{r} + \beta$.*

Proof: The "only if" part is straightforward: if $\vec{s} = \alpha\vec{r} + \beta$ where $\alpha > 0$, then for any profile P and any candidate c, $s(P, c) = \alpha r(P, c) + \beta$. Therefore, if c has strictly the highest score for s, then c also has strictly the highest score for r, which means that s and r are not different.

We now prove the "if" part. First, because $s_1 > s_m$ and $r_1 > r_m$, there does not exist $\alpha < 0$ and β such that $\vec{s} = \alpha\vec{r} + \beta$. Therefore, if for any α, β we have $\vec{s} = \alpha\vec{r} + \beta$, then there exist two pairs of positions (i_1, i_2) and (i'_1, i'_2) where $i_1 < i_2$ and $i'_1 < i'_2$, such that $\frac{s_{i_1} - s_{i_2}}{s_{i'_1} - s_{i'_2}} \neq \frac{r_{i_1} - r_{i_2}}{r_{i'_1} - r_{i'_2}}$. We note that in this case $m \geq 3$. We construct a profile $P = P_1 \cup P_2$ as follows.

P_1 is a profile where $s(P_1, c_1) > s(P_1, c_2)$ and $r(P_1, c_1) < r(P_1, c_2)$. WLOG $\frac{s_{i_1} - s_{i_2}}{s_{i'_1} - s_{i'_2}} > \frac{r_{i_1} - r_{i_2}}{r_{i'_1} - r_{i'_2}}$. Let T, T' be two natu-

ral numbers such that $\frac{s_{i_1} - s_{i_2}}{s_{i'_1} - s_{i'_2}} > \frac{T}{T'} > \frac{r_{i_1} - r_{i_2}}{r_{i'_1} - r_{i'_2}}$. Let V_1 (respectively, V'_1) denote the vote where c_1 (respectively, c_2) is ranked in the i_1th (respectively, i'_1th) position and c_2 (respectively, c_1) is ranked in the i_2th (respectively, i'_2th) position. The other candidates in V_1 and V'_1 are ranked arbitrarily. Let $P_1 = \{T' \cdot V_1\} \cup \{T \cdot V'_1\}$. We have $s(P_1, c_1) - s(P_1, c_2) = T'(s_{i_1} - s_{i_2}) - T(s_{i'_1} - s_{i'_2}) > 0$ and $r(P_1, c_1) - r(P_1, c_2) = T'(r_{i_1} - r_{i_2}) - T(r_{i'_1} - r_{i'_2}) < 0$.

We now define P_2. Let M denote the cyclic permutation among $\{c_3, \ldots, c_m\}$, that is, $M : c_3 \to c_4 \to \cdots \to c_m \to c_3$, Let $V = [c_1 \succ c_2 \succ \cdots \succ c_m]$, $V' = [c_2 \succ c_1 \succ \cdots \succ c_m]$ and $P^* = \{V, V', M(V), M(V'), \ldots, M^{m-3}(V), M^{m-3}(V')\}$. Since $s_1 > s_m$ and $r_1 > r_m$, we have that for any $i \geq 3$, $s(P^*, c_1) = s(P^*, c_2) > s(P^*, c_i)$. Let $P_2 = (T + T' + 1) \cdot P^*$.

Let $P = P_1 \cup P_2$, we have that c_1 (respectively, c_2) has the strictly highest score in P for $\vec{s}$ (respectively, $\vec{r}$), which means that s is different from r. □

It follows immediately from Proposition 3 and Proposition 4 that $plurality + Borda$ is not consistent.

The combinator also does not satisfy a number of algebraic properties. The first property is that the combinator is not associative.

Proposition 5 *There exists voting rules X, Y and Z for which $(X + Y) + Z \neq X + (Y + Z)$.*

Proof: Consider the three rules X, Y and Z that elect the plurality winner, and when there is tied plurality winner, X elects x before any other candidate, Y elects y and Z elects z. Suppose we have a Condorcet cycle with one vote for $x > y > z$, one for $y > z > x$ and one for $z > x > y$. Then $X + Y$ elects x, and thus $(X + Y) + Z$ elects z. Similarly, $Y + Z$ elects y, and thus $X + (Y + Z)$ elects x. □

The combinator does not distribute over itself.

Proposition 6 *There exists voting rules X, Y and Z for which $(X + Y) + Z \neq (X + Z) + (Y + Z)$.*

Proof: Consider the same three rules and set of votes. Then $(X + Y) + Z$ elects z as before. Now, $X + Z$ elects x, $Y + Z$ elects y, and thus $(X + Z) + (Y + Z)$ elects x. □

There is no majority consistent voting rule that acts as an identity.

Proposition 7 *There does not exist a majority consistent voting rule I such that for any voting rule X, we have $X + I = X$.*

Proof: Consider the same three rules and set of votes. The proof uses contradiction. Suppose $X + I = X$, $Y + I = Y$ and $Z + I = Z$. As $X + I = X$, it must be that I elects x or some candidate that x beats (which is only y). As $Y + I = Y$, it must be that I elects y or some candidate that y beats (which is only z). The only candidate common to the two cases is y. Therefore, I must elect y given these votes. As $Z + I = Z$, it must be that I elects z or some candidate that z beats (which is only x). This is a contradiction. □

4 STRATEGIC VOTING

Combining voting rules together can hinder strategic voting. One appealing escape from the Gibbard-Satterthwaite theorem was proposed by Bartholdi, Tovey and Trick [2]. Perhaps it is computationally so difficult to find a successful manipulation that voters have little option but to report their true preferences? As is common in the literature, we consider two different settings: unweighted votes where the number of candidates is large and we have just one or two manipulators, and weighted votes where the number of candidates is small but we have a coalition of manipulators. Whilst unweighted votes are perhaps more common in practice, the weighted case informs us about the unweighted case when we have probabilistic information about the votes [5]. Since there are many possible combinations of common voting rules, we give a few illustrative results covering some of the more interesting cases. With unweighted votes, we prove that computational resistance to manipulation is typically inherited from the base rules. With weighted votes, our results are stronger. We prove that there are many combinations of voting rules where the base rules are polynomial to manipulate but their combination is NP-hard. Combining voting rules thus offers another mechanism to make manipulation more computationally difficult.

A FIRST OBSERVATION

It seems natural that the combination of voting rules inherits the computational complexity of manipulating the base rules. However, there is not a simple connection between the computational complexity of the bases rules and their combination. In this section, we show two examples of artificial voting rules to illustrate this discrepancy. In the first example, manipulation for the base rules are NP-hard, but manipulation for their combination can be computed in polynomial-time; in the second example, manipulation for the base rules are in P, but manipulation for their combination is NP-hard to compute.

Proposition 8 *There exist voting rules X and Y for which computing a manipulation is NP-hard but computing a manipulation of $X + Y$ is polynomial.*

Proof: We give a reduction from (1 in 3)-SAT on positive clauses. Boolean variables 1 to n are represented by the candidates 1 to n. We also have two additional candidates 0 and -1. Any vote with 0 in first place represents a clause. The first three candidates besides 0 and -1 are the literals in the clause. Any vote with -1 in first place represents a truth assignment. The positive literals in the truth assignment are those Boolean variables whose candidates appear between -1 and 0 in the vote. With 2 candidates, X and Y both elect the majority winner. With 3 or more candidates, X elects candidate -1 if there is a truth assignment in the votes that satisfies exactly one out of the three literals in each clause represented by the votes and otherwise elects 0. Computing a manipulation of X is NP-hard. Similarly, with 3 or more candidates, Y elects candidate 0 if there is a truth assignment in the votes that satisfies exactly one out of the 3 literals in every clause represented by the votes and otherwise elects -1. Computing a manipulation of Y is NP-hard. However, $X + Y$ is polynomial to manipulate since 0 and -1 always go through to the runoff where the majority candidate wins. □

Proposition 9 *There exist voting rules X and Y for which computing a manipulation is polynomial but computing a manipulation of $X + Y$ is NP-hard.*

Proof: The proof uses a similar reduction from (1 in 3)-SAT on positive clauses. With 2 candidates, X and Y both elect the majority winner. With 3 or more candidates, X elects candidate -1 if there is a truth assignment in the votes that satisfies at least one out of the three literals in each clause represented by the votes and otherwise elects 0, whilst Y elects candidate -1 if there is a truth assignment in the votes that satisfies at most one out of the three literals in each clause

represented by the votes and otherwise elects 0. Computing a manipulation of X or Y is polynomial since we can simply construct either the vote that sets each Boolean variable to true or to false. However, computing a manipulation of $X + Y$ as it may require solving a (1 in 3)-SAT problem on positive clauses. □

UNWEIGHTED VOTES, TRACTABLE CASES

If computing a manipulation of the base rules is polynomial, it often remains polynomial to compute a manipulation of the combined rules. However, manipulations may now be more complex to compute. We need to find a manipulation of one base rule that is compatible with the other base rules, and that also wins the runoff. We illustrate this for various combinations of scoring rules.

Proposition 10 *Computing a manipulation of $plurality + veto$ is polynomial.*

Proof: We present a polynomial-time algorithm that checks whether k manipulators can make c win in the following two steps: we first check for every candidate a, whether the manipulators can make c to be the plurality winner for $P \cup M$ while a is the veto winner, and c beats a in the runoff (or $c = a$). Then, we check for every candidate a whether the manipulators can make c to be the veto winner while a is the plurality winner, and c beats a in the runoff.

For the first step, let S be a subset of candidates that beat a in P under $veto$. We denote Δ_s as the difference in the veto score of s, $s \in S$, and a in P. If the veto scores are equal and $s \succ a$ in the tie-breaking rule then we set $\Delta_s = 1$. If $\sum_{s \in S} \Delta_s > k$ then a can not win under $veto$. Otherwise, we place s in last positions in exactly Δ_s manipulator votes. This placing is necessary for a to win under $veto$. We place c in the first position and a in the second position in all votes in M. We fill the remaining positions arbitrarily. This manipulation is optimal under an assumption that a wins under $veto$ as c is always placed in the first position. For each possible candidate for a, we check if such a manipulation is possible and check if c is the winner of the run-off round. If we find a manipulation we stop. The special case when $a = c$ is analogous.

For the second step, let b be the candidate with maximum $plurality$ score in P. We denote Δ_a to be the difference in the plurality scores of b and a. We place a in the Δ_a first positions in the manipulator votes. The condition is necessary for a to win under $plurality$. We put a in the second position in the remaining votes. We put c in the second position after a in Δ_a manipulator votes and put c in the first position in the remaining $k - \Delta_a$ votes. To ensure that c wins under $veto$ we perform the same procedure as above. The only simplification is that we do not need to worry about tie-breaking rule as c wins tie-breaking by assumption. We fill the remaining positions arbitrarily. This manipulation is optimal under an assumption that a wins under $plurality$, as c is placed in the first position unless a has to occupy it. For each possible candidate a we check if such a manipulation is possible and check if c is the winner of the run-off round. If we find a manipulation we stop. Otherwise, there is no manipulation. □

It is also in P to decide if a single agent can manipulate an election for any combination of scoring rules. Interestingly, we can use a perfect matching algorithm to compute this manipulation.

Proposition 11 *Computing a manipulation of $X + Y$ is polynomial for a single manipulator and any pair of scoring rules, X and Y.*

Proof: Suppose there is a manipulating vote v such that c wins $P \cup \{v\}$ under $X + Y$. Let X and Y have the scoring vectors $(x_1, \ldots, x_m)$ and $(y_1, \ldots, y_m)$. As is common in the literature, we assume tie-breaking is in favour of c. Suppose c wins under X in a successful manipulation. The case that c wins under Y is dual. Suppose another candidate a wins under Y, c is placed at position i and a is placed at position j in v. We show how to construct this vote if it exists by finding a perfect matching in a bipartite graph. For each candidate besides c and a, we introduce a vertex in the first partition. For each position in $[1, m] \setminus \{i, j\}$ we introduce a vertex in the second partition. For each candidate c_k besides c and a we connect the corresponding vertex with a vertex t in the second partition iff (1) the score of c_k in P under X less the score of c in P under X is less than or equal to $x_i - x_k$, and (2) the score of c_k in P under Y less the score of a in P under Y is less than or equal to $y_j - y_t$, or if two differences are equal then a is before c_k in the tie-breaking rule. In other words, we look for a placement of the remaining candidates in v such that c wins in $P \cup \{v\}$ under X, a wins in $P \cup \{v\}$ under Y, c is at position i and a is at position j in v. There exists a perfect matching in this graph iff there is a manipulating vote that satisfies our assumption. If $a = c$, the reasoning is similar but we only need to fix the position of c. Using this procedure, we check for each candidate a and for each pair of positions (i, j) if there exists a vote v such that c wins in $P \cup \{v\}$ under X, a wins in $P \cup \{v\}$ under Y, c is at position i and a is at position j in v. If such a vote exists, we also check if c beats a in the run-off round. If c loses to a in the run-off for all combinations of a and (i, j) then no manipulation exists. □

UNWEIGHTED VOTES, INTRACTABLE CASES

We begin with combinations involving STV. This was the first commonly used voting rule shown to be NP-hard to manipulate by a single manipulator [1]. Not surprisingly, even when combined with voting rules which are polynomial to manipulate like plurality, veto, or k-approval, manipulation remains NP-hard to compute.

Proposition 12 *Computing a manipulation of $X + STV$ is NP-hard for $X \in \{plurality, k\text{-approval}, veto, Borda\}$ for one manipulator.*

Proof: (Sketch) Consider the NP-hardness proof for manipulation of STV [1]. We denote the profile constructed in the proof P. The main idea is to modify P so that the preferred candidate c can win under $X + STV$ iff c can win the modified election under STV. For reasons of space, we illustrate this for $X = Borda$. Other proofs are similar. Candidate w (who is the other possible winner of P) has the top Borda score. Hence, c must win by winning the STV election (which is possible iff there is a 3-cover). We still have the problem that w beats c in the run-off. Hence, we introduce a dummy candidate g' after c in each vote. This makes sure that the score of g' is greater than or equal to $(n-6)|P|$. We also introduce $G = \lfloor \frac{|P|}{n} \rfloor$ blocks of n votes. Let $P' = \bigcup_{k=1}^{G} \bigcup_{i=1}^{n} (g' \succ c_i \succ \ldots \succ c_{i-1})$. The Borda score of g' in $P \cup P'$ is greater than that of any other candidate. In an STV election on $P \cup P'$, g' reaches the last round. Therefore, the elimination order remains determined by the votes in P. Hence, if there is a 3-cover, the candidate c can reach the last round. In the worst case, when $|P|$ is divisible by n, the plurality scores of c and g' are the same and c wins by tie-breaking. □

We turn next to combinations of Borda voting, where it is NP-hard to manipulate with two manipulators [6, 3].

Proposition 13 *Computing a manipulation of $X + Borda$ by two manipulators is NP-hard for $X \in \{plurality, k\text{-approval}, veto\}$*

Proof: (Sketch) Consider the NP-hardness proof for manipulation of Borda which uses a reduction from the permutation sums problem [6]. Due to the spaces constraint, we consider only $veto + Borda$. Other proofs are similar but much longer and more complex. The reduction uses the following construction to inflate scores to a desired target. To increase the score of candidate c_i by 1 more than candidates $c, c_1 \ldots, c_{i-1}, c_{i+1}, \ldots, c_{n-1}$ and by 2 more than candidate c_n we consider the following pair of votes:

$$c_i \succ c_n \succ c_1 \succ \ldots \succ c_{n-1}$$
$$c_{n-1} \succ c_{n-2} \succ \ldots \succ c_1 \succ c_i \succ c_n$$

We change the construction by putting c in the last place in the first vote in each pair of votes and first place in the second vote, and leaving all other candidates unchanged when we increase the score of $c_i \neq c$ by one. This modification does not change the desired properties of these votes. Note that c and c_n cannot be winners under $veto$. Hence, c must win under $Borda$ and then win the run-off. This is possible iff there exists a solution for permutation sums problem. □

WEIGHTED VOTES, TRACTABLE CASES

We focus on elections with weighted votes and 3 candidates. This is the fewest number of candidates which can give intractability. All scoring rules besides plurality (e.g. Borda, veto, 2-approval) are NP-hard to manipulate in this case [8]. We therefore focus on combinations of the voting rules: plurality, cup, Copeland, maximin and Bucklin. Computing a manipulation of each of these rules is polynomial in this case. We were unable to find a proof in the literature that Bucklin is polynomial to manipulate with weighted votes, so we provide one below.

Proposition 14 *Computing a manipulation of $Bucklin$ is polynomial with weighted votes and 3 candidates.*

Proof: It is always optimal to place the preferred candidate c in the first position as this only decreases the scores of the other 2 candidates, a and b. We argue that the winner is chosen in one of the first two rounds. In the first round, if c still loses to a or b then there is no manipulation that makes c win. In the second round, we must have at least one candidate with a majority. Suppose we did not. Then the sum of scores of the 3 candidates is at most $3n/2$. But the sum of the approval votes is $2n$ which is a contradiction. Hence, if c does not get a majority in this round, one of the other candidates wins regardless of the manipulating votes. □

We recall that in this paper all ties are broken in favor of c, which is crucial in the proof of the above proposition. In fact, we can show that if some other tie-breaking mechanisms are used, then Bucklin is hard to manipulate with weighted votes, even for 3 candidates.

We next identify several cases where computing a manipulation for combinations of these voting rules is tractable.

Proposition 15 *Computing a manipulation of $Copeland + cup$, or of $Copeland + Bucklin$ is polynomial with weighted votes and 3 candidates.*

Proof: First we consider the outcome of c vs a and c vs b assuming that c is ranked at the first position by all manipulators.

Case 1. Suppose c is a Condorcet loser. In this case, c can only win if c wins under both $Copeland$ and Y. However, c must lose under $Copeland$ because $Copeland$ never elects the Condorcet loser.

Case 2. Suppose c is a Condorcet winner. Then c is a winner of both rules as they are both Condorcet consistent.

Case 3. Suppose there exists a candidate a such that $N_{P\cup M}(a,c) > N_{P\cup M}(c,a)$ and $N_{P\cup M}(b,c) \leq N_{P\cup M}(c,b)$ even if c is ranked first by all manipulators. We argue that if there is a manipulation, then all manipulators can vote $c \succ b \succ a$. We consider the case that c wins under $Copeland$ and b wins under cup. The other cases (b wins under $Copeland$, c wins under $Bucklin$, etc.) are similar. For c to win under $Copeland$, all candidates must to have the $Copeland$ score of 0 as, by assumption, c loses to a. Hence, the maximum $Copeland$ score of c is 0. Therefore, for c to win the following holds $N_{E\cup M}(b,a) > N_{E\cup M}(a,b)$ and $N_{E\cup M}(c,b) > N_{E\cup M}(b,c)$. The only possible agenda is a vs c, and the winner playing b. In all other agendas, b loses to c in one of the rounds. For b to win cup, $N_{P\cup M}(b,a) \geq N_{P\cup M}(a,b)$ and tie-breaking has $c \succ b \succ a$. The manipulation vote $c \succ b \succ a$ will only help achieve the inequalities in both cases. □

Proposition 16 *Computing a manipulation of $Bucklin + cup$ is polynomial with weighted votes and 3 candidates.*

Proof: We consider three possible outcomes of pairwise comparison between c vs a and c vs b assuming that c is ranked at the first position by all manipulators.

Case 1. Suppose c is a Condorcet loser after the manipulation. c can only win overall if c wins under both $Bucklin$ and cup. However, c must lose under cup.

Case 2. Suppose c is a Condorcet winner. Then c must be a winner of cup as this is Condorcet consistent. Hence, regardless of the rest of the manipulating votes, c reaches the run-off round and beats any other candidate.

Case 3. Suppose there exists candidate a such that $N_{P\cup M}(a,c) > N_{P\cup M}(c,a)$ and $N_{P\cup M}(b,c) \leq N_{P\cup M}(c,b)$. Note that M must guarantee that a does not reach the run-off round as c loses to a in the pairwise elections. There are two sub-cases: c wins under cup and b wins under $Bucklin$ in $P \cup M$, or b wins under cup and c wins under $Bucklin$. As shown in the proof of the last Proposition, if there is a manipulation, $c \succ b \succ a$ will work in both cases. □

WEIGHTED VOTES, INTRACTABLE CASES

We continue to focus on combinations of the voting rules: plurality, cup, Copeland, maximin and Bucklin. We give several results which show that there exists combinations of these voting rules where manipulation is intractable to compute despite the fact that all the base rules being combined are polynomial to manipulate. These results provide support for our argument that combining voting rules is a mechanism to increase the complexity of manipulation.

Proposition 17 *Computing a manipulation of $plurality + Y$ where $Y \in \{cup, Copeland, maximin, Bucklin\}$, is NP-complete with weighted votes and 3 candidates.*

Proof: (Sketch) We consider the case $plurality + cup$. Other proofs are similar but longer. We reduce from a PARTITION problem in which we want to decide if integers k_i with sum $2K$ divide into two equal sums of size K. Consider the following profile:

$4K$	$a \succ b \succ c$	$4K$	$a \succ c \succ b$
$1K$	$b \succ a \succ c$	$9K$	$b \succ c \succ a$

For each integer k_i, we have a member of the manipulating coalition with weight $2k_i$. The tie-breaking rule is $c \succ a \succ b$. The cup has a

play b, and the winner meets c. Note that b cannot reach the run-off as they beat c in pairwise elections whatever the manipulators do. Note that c cannot win the $plurality$ rule. Hence a must be the $plurality$ winner. The run-off is a, the $plurality$ winner against c, the cup winner (which is the same as the final round of the cup). For this to occur, the manipulators have to partition their votes so that exactly $2K$ manipulators put c above a and $2K$ put a in the first position (and above c).[4] Therefore there exists a manipulation iff there exists a partition. □.

Proposition 18 *Computing a manipulation of $Copeland + Y$ where $Y \in \{plurality, maximin\}$, is NP-complete with weighted votes and 3 candidates.*

Proof: (Sketch) We consider the case $Copeland + plurality$. Other proofs are similar but longer. We again reduce from a PARTITION problem. Consider the following profile:

$$\begin{array}{llll} 7K & b \succ c \succ a & K & b \succ a \succ c \\ 4K & a \succ c \succ b & 2K & a \succ b \succ c \\ 1 & c \succ a \succ b & & \end{array}$$

For each integer k_i, we have a member of the manipulating coalition with weight $2k_i$. Now, b must not reach the run-off round and a must win $plurality$ by similar arguments to the last proof. Hence c must be the $Copeland$ winner. For this to occur, the manipulators have to partition their votes so that exactly $2K$ manipulators put c above a, $2K$ manipulators put a in the first position (and above c) and put b in the last position in all votes. Therefore there exists a manipulation iff there exists a partition. □

Proposition 19 *Computing a manipulation of $maximin + Y$ where $Y \in \{plurality, cup, Copeland, Bucklin\}$, is NP-complete with weighted votes and 3 candidates.*

Proof: (Sketch) We consider the case $maximin + plurality$. Other proofs are similar but longer. We reduce from a PARTITION problem in which we want to decide if integers k_i with sum $2K$ divide into two equal sums of size K. Consider the following profile:

$$\begin{array}{llll} 4K & b \succ c \succ a & 2K & b \succ c \succ a \\ 2K & a \succ b \succ c & 2K & a \succ c \succ b \end{array}$$

For each integer k_i, we have a member of the manipulating coalition with weight $2k_i$. Now, b must not reach the run-off round and a must win $plurality$ by similar arguments to the last proof. Hence c must be the $maximin$ winner. For a to win $plurality$, manipulators with total weight at least $2K$ must rank a first. Before the manipulators vote, the $maximin$ score of a is $4K$, of b is $6K$ and of c is $2K$. We note that c must be ranked above b in all manipulators votes and above a in $2K$ manipulators votes, otherwise c loses to b under $maximin$. As $2K$ manipulators must vote $a \succ c \succ b$, we have $N_{P \cup M}(a, b) \geq 6K$, $N_{P \cup M}(c, b) \geq 4K$ and $N_{P \cup M}(a, c) \geq 6K$. This increases the $maximin$ score of a to $6K$ and of c to $4K$. Now c must be ranked above a in at least $2K$ manipulators votes to increase its $maximin$ score to $6K$. Hence, the only possible option is if $2K$ manipulators vote $a \succ c \succ b$ and $2K$ vote and $c \succ a \succ b$ with weight $2K$. In this case the $maximin$ score of all candidates are the same and equal to $6K$. By the tie-breaking rule, c wins. Therefore, there exists a manipulation iff there exists a partition. □

We summarize our results about weighted manipulation in the following table.

[4] Here we abuse the notation by saying "$2K$ manipulators", which we meant "manipulators whose weights sum up to $2K$".

$X + Y$	plurality	maximin	Copeland	cup	Bucklin
plurality	P	NPC	NPC	NPC	NPC
maximin	–	P	NPC	NPC	NPC
Copeland	–	–	P	P	P
cup	–	–	–	P	P
Bucklin	–	–	–	–	P

Table 1. Computational complexity of coalition manipulation with weighted votes and 3 candidates

5 CONCLUSION

We have put forwards a simple method for combining together voting rules that performs a run-off between the different winners of each voting rule. We have provided theoretical evidence for the value of this combinator. We proved that a combination of voting rules can inherit a desirable property like Condorcet consistency or the majority criterion from just one base voting rule. On the other hand, two important properties can be lost by the introduction of a run-off: monotonicity, consistency. We showed that the combinator satisfies several algebraic properties, e.g. it is idempotent and commutative, and does not satisfy other properties, including associativity and distributivity. Combining voting rules also tends to increase the computational difficulty of finding a manipulation. For instance, with weighted votes, we proved that computing a manipulation for a simple combination like $plurality$ and cup is NP-hard, even though $plurality$ and cup on their own are polynomial to manipulate.

REFERENCES

[1] J.J. Bartholdi and J.B. Orlin, 'Single transferable vote resists strategic voting', *Social Choice and Welfare*, **8**(4), 341–354, (1991).

[2] J.J. Bartholdi, C.A. Tovey, and M.A. Trick, 'The computational difficulty of manipulating an election', *Social Choice and Welfare*, **6**(3), 227–241, (1989).

[3] N. Betzler, R. Niedermeier, and G.J. Woeginger, 'Unweighted coalitional manipulation under the Borda rule is NP-hard', in *Proc. of IJCAI*, pp. 55–60, (2011).

[4] V. Conitzer and T. Sandholm, 'Universal voting protocol tweaks to make manipulation hard', in *Proc. of IJCAI*, pp. 781–788, (2003).

[5] V. Conitzer, T. Sandholm, and J. Lang, 'When are elections with few candidates hard to manipulate', *JACM*, **54** (3), 1–33, (2007).

[6] J. Davies, G. Katsirelos, N. Narodytska, and T. Walsh, 'Complexity of and algorithms for Borda manipulation', in *Proc. of AAAI*, pp. 657–662, (2011).

[7] E. Elkind and H. Lipmaa, 'Hybrid voting protocols and hardness of manipulation', in *Proc. of ISAAC'05*, pp. 24–26,(2005).

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

[9] James Surowiecki, *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*, Little Brown & Co, 2004.

[10] T. Walsh and L. Xia, 'Lot-based voting rules', in *Proc. of AAMAS*, (2012).

[11] M. Zuckerman, A.D. Procaccia, and J.S. Rosenschein, 'Algorithms for the coalitional manipulation problem', in *Proc. of SODA*, pp. 277–286, (2008).