Manipulating Two Stage Voting Rules

Nina Narodytska
NICTA and UNSW
Sydney, Australia
nina.narodytska@nicta.com.au

Toby Walsh
NICTA and UNSW
Sydney, Australia
toby.walsh@nicta.com.au

ABSTRACT

We study the computational complexity of computing a manipulation of a two stage voting rule. An example of a two stage voting rule is Black's procedure. The first stage of Black's procedure selects the Condorcet winner if it exists, otherwise the second stage selects the Borda winner. In general, we argue that there is no connection between the computational complexity of manipulating the two stages of such a voting rule and that of the whole. However, we also demonstrate that we can increase the complexity of even a very simple base rule by adding a simple first stage to the front of the base rule. In particular, whilst Plurality is polynomial to manipulate, we show that the two stage rule that selects the Condorcet winner if it exists and otherwise computes the Plurality winner is NP-hard to manipulate with three or more candidates, weighted votes and a coalition of manipulators. In fact, with any scoring rule, computing a coalition manipulation of the two stage rule that selects the Condorcet winner if they exist and otherwise applies the scoring rule is NP-hard with three or more candidates and weighted votes. It follows that computing a coalition manipulation of Black's procedure is NP-hard with weighted votes. With unweighted votes, we prove that the complexity of manipulating Black's procedure is inherited from the Borda rule that it includes. More specifically, a single manipulator can compute a manipulation of Black's procedure in polynomial time, but computing a manipulation is NP-hard for two manipulators. With two stage voting rules, we can also allow agents to re-vote between rounds. We study the impact of such re-voting on manipulation.

Categories and Subject Descriptors

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

General Terms

Economics

Keywords

social choice, voting, manipulation

1. INTRODUCTION

There exist several voting procedures that work in stages. For example, Black's procedure is a two stage voting rule whose first stage elects the Condorcet winner if one exists, and otherwise moves to a second stage which elects the Borda winner [12]. As a second example, the French presidential elections use a two stage runoff voting system. If there is a majority winner in the first stage, then this candidate is the overall winner, otherwise we go to the second stage where there is a runoff vote between the two candidates with the most votes in the first round. Such two stage voting rules can inherit a number of attractive axiomatic properties from their parts. For example, Black's procedure inherits Condorcet consistency from its first part, and properties like monotonicity, participation and the Condorcet loser property from its second part. Inheriting such properties from its parts might be considered an attractive feature of two stage voting rules. On the other hand, a less desirable property of one of the base rules can infect the overall two stage rule. For instance, it has been shown that, with single peaked votes, many types of control and manipulation problems are polynomial for Black's procedure [4]. This polynomial cost is essentially inherited from the first stage of the rule which selects the Condorcet winner (which must exist with single peaked votes). Such vulnerability to manipulation and control might be considered an undesirable property for a two stage voting rule. This raises several interesting questions from the perspective of computational social choice. For example, with unrestricted votes as opposed to single peaked votes, are two stage voting rules more or less computationally difficult to manipulate than single stage voting rules? How does the computational complexity of manipulating a two stage voting rule depend on the computational complexity of manipulating the two rules that it composes? In this paper, we address such questions.

Our work builds upon recent research that looks at methods to combine together voting rules. In [10], Davies *et al.* considered a recursive combinator that successively eliminates the least popular candidate(s). This captures voting rules proposed in the past like those of Nanson, Baldwin or Coombs (all described in more detail in the next section). By comparison, we consider here a sequential combinator where the first rule eliminates all but the most popular candidates and the second rule then decides between those that remain. This captures voting rules proposed in the past like Black's procedure. Perhaps closest to this work is the sequential combinator introduced by Elkind and Lipmaa in [11]. This is an intermediate position between the two extremes of eliminating the least popular and all but the most popular candidates. Elkind and Lipmaa's combinator eliminates candidates by applying some given number of rounds of the first rule before using the second rule to decide between the candidates that remain. Even more recently, Nardoytska *el al.* have considered a parallel combinator that combines together the opinions of two (or more) different voting rules [16]. This combinator applies both rules simultaneously and compares their results. As well as proving computational properties of

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

existing two stage voting rules like Black's procedure, this paper strengthens the evidence that adding multiple rounds to voting often increases the computational resistance to manipulation.

2. BACKGROUND

A *profile* is a sequence of n total orders over m candidates. A *voting rule* is a function mapping a profile onto a set of *winners* (strictly speaking this is a social choice correspondence). We consider some of the most common voting rules.

Scoring rules: Given a *scoring vector* $(w_1, \ldots, w_m)$ of weights, 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 $(1, 1, \ldots, 1, 0)$, and the **Borda** rule has the vector $(m-1, m-2, \ldots, 0)$. With **Majority**, a candidate wins if they have half or more of the votes. With two candidates, **Plurality** is the same as **Majority** voting. We say that one scoring rule is *isomorphic* to another iff the scoring vector of one is a linear transformation of the other. That is, there exist α and β such that the two scoring vectors are related by $(u_1, \ldots, u_m) = (\alpha v_1 + \beta, \ldots, \alpha v_m + \beta)$.

Cup: The winner is the result of a series of pairwise majority elections between candidates. Given the *agenda*, a binary tree in which the leaves 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.

Black's procedure: This rule has two stages. We first determine if there is a *Condorcet winner*, a candidate that beats all others in pairwise majority comparisons. If there is, this is the winner. Otherwise, we return the result of the *Borda* rule.

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

Nanson's and Baldwin's rules: These are iterated versions of the Borda rule. In Nanson's rule, we compute the Borda scores and eliminate all candidates with less than half the mean score. We repeat until a single candidate remains. In Baldwin's rule, we compute the Borda scores and eliminate the candidate with the lowest score. We again repeat until there is a single winner.

Coombs' rule: This is an iterated version of the Veto rule. We repeatedly eliminate the candidate with the most vetoes until we have one candidate with a strict majority.

We consider both unweighted and integer weighted votes. A weighted vote can simply be viewed as a block of identical unweighted votes. Given a fixed profile of votes cast by the non-manipulators, the **weighted coalition manipulation problem** is to decide if there exist strategic votes for the manipulators, each with a given weight, so that a given candidate wins. Given a fixed profile of votes cast by the non-manipulators, the **unweighted manipulation problem** for one manipulator (two manipulators) is to decide if there exist strategic votes for the one manipulator (two manipulators) so that a given candidate wins. We suppose that any ties in the operation of a voting rule are broken lexicographically, and that candidates are renamed so that the candidate preferred by the manipulators is always first in this order.

3. TWO STAGE VOTING RULES

We consider a general class of two stage voting rules. Given voting rules X and Y, the rule XTHENY applies the voting Y to the profile constructed by eliminating all but the winning candidates from the voting rule X. Both X and Y can themselves be two stage voting rules giving us the possibility to construct multi-stage voting rules. For example, Black's procedure is $Condorcet$THEN$Borda$ where $Condorcet$ is the multi-winner rule that elects the Condorcet winner if it exists, and otherwise elects all candidates. As a second example, Plurality with Runoff is $TopTwo$THEN$Majority$ where $TopTwo$ is the multi-winner voting rule that elects the candidates with the two most plurality votes. There are many possible rules that we might choose to combine this way. $Condorcet$ is an attractive choice for the first rule as it guarantees that the resulting combination is Condorcet consistent. However, there are other interesting choices including:

CondorcetLoser: This rule elects all candidates except, when it exists, the Condorcet loser.

CopelandSet: This rule elects all candidates in the Copeland set. The Copeland score of a candidate is the number of candidates that it beats less the number of candidates that beats it. The Copeland set contains those candidates with the maximal Copeland score. When there is a Condorcet winner, this is the only candidate in the Copeland set.

SmithSet: This rule elects all candidates in the Smith set. This is the smallest non-empty set of candidates such that every candidate in the set beats every candidate outside the set in pairwise elections. When there is a Condorcet winner, this is the only candidate in the Smith set. Voting rules like Nanson's and Kemeny are guaranteed to pick candidates from the Smith set.

SchwartzSet: This rule elects all candidates in the Schwartz set. The Schwartz set is a subset of the Smith set and is the union of all the undominated sets. A set is undominated if every candidate inside the set is pairwise unbeaten by every candidate outside, and no non-empty proper subset satisfies this property. When there is a Condorcet winner, this is the only candidate in the Schwartz set.

We also consider recursively defined multi-stage voting rules. We suppose any recursion terminates when either we have a single candidate left, or the set of candidates left does not reduce in size. For example, we can recursively define STV by:

$$STV = PluralityLoser\text{THEN}STV$$

$PluralityLoser$ is the rule that elects all candidates but the candidate with the fewest first place votes. As a second example, we can recursively define Baldwin's rule by:

$$Baldwin = BordaLoser\text{THEN}Baldwin$$

$BordaLoser$ is the multi-winner rule that elects all candidates but the candidate with the lowest Borda score. Nanson's rule can be defined recursively in a similar way. As a third example, we can define Coombs' rule by:

$$Coombs = Majority\text{THEN}(VetoLoser\text{THEN}Coombs)$$

$Majority$ elects the candidate with a majority of first place votes or, if there is no such candidate, elects all candidates, and $VetoLoser$ is the rule that elects all candidates but the candidate with the most last placed votes.

4. SOME AXIOMATIC AND ALGEBRAIC PROPERTIES

It is interesting to consider which axiomatic properties are inherited from the base rules being combined. For example, it is simple to see that we can inherit Condorcet consistency or the Condorcet loser properties.

PROPOSITION 1. *For any voting rule X, $Condorcet$THENX, $CopelandSet$THENX, $SchwartzSet$THENX, and $SmithSet$THENX are Condorcet consistent. Similarly, for any voting rule Y, the combination $CondorcerLoser$THENY satisfies the Condorcet loser property.*

With recursively defined rules, we can give a similar result. We say that a multi-winner rule is Condorcet consistent if it includes the Condorcet winner in the set of winners, and satisfies the Condorcet loser property if the set of winners never includes the Condorcet loser.

PROPOSITION 2. *Suppose Y is recursively defined by $Y = X$THENY and X is Condorcet consistent. Then Y is also Condorcet consistent. Similarly, if X satisfies the Condorcet loser property then Y does also.*

Note that the Borda loser is never the Condorcet winner. Hence, the multi-winner rule $BordaLoser$ is Condorcet consistent. Thus, it follows from Proposition 2 that Baldwin's rule (which is recursively defined using $BordaLoser$) is also Condorcet consistent. There are also axiomatic properties which can be lost by combining together voting rules. For example, the Borda loser rule which eliminates the lowest Borda scoring candidate is monotonic since increasing one's preference for a candidate can only prevent them from being the Borda loser. However, Baldwin's rule, which is the recursive version of the Borda loser rule, is not monotonic. It will therefore be interesting to identify conditions under which two stage voting rules are monotonic.

The THEN combinator has a number of interesting algebraic properties. For example, the $Identity$ rule that returns all candidates is a left and right identity of the THEN combinator. Note that the THEN combinator is neither commutative nor associative. If a voting rule is recursively defined then it is idempotent (that is, XTHEN$X = X$). More complex identities can be derived such as the following.

PROPOSITION 3. *If X is idempotent then XTHEN$(X$THEN$Y) = X$THENY and $(Y$THEN$X)$THEN$X = Y$THENX.*

PROPOSITION 4. $SmithSet$THEN$Nanson = Nanson$.

PROPOSITION 5. *If X is Condorcet consistent and only returns the Condorcet winner when they exist then $Condorcet$THEN$X = X$.*

5. COMPLEXITY OF MANIPULATION

One of the main contributions of this paper is to consider the impact of two stage voting rules on the computational complexity of computing a manipulation. As in previous studies (e.g. [2, 7]), we consider manipulation with unweighted votes and a small number of manipulators, and manipulation with weighted votes, a coalition of manipulators and a small number of candidates. As is common in the literature, we break ties in favour of the manipulators.

5.1 Weighted votes, general results

With weighted votes, we first argue that there is no connection in general between the computational complexity of computing a manipulation of a two stage voting rule and the computational complexity of manipulating its parts.

PROPOSITION 6. *There exist voting rules X and Y with the following properties for weighted votes:*

1. *computing coalition manipulations of X, Y and XTHENY are polynomial;*
2. *computing coalition manipulations of X and Y are polynomial but of XTHENY is NP-hard;*
3. *computing a coalition manipulation of X is polynomial and of Y is NP-hard, but of XTHENY is polynomial;*
4. *computing a coalition manipulation of X is polynomial, but of Y and XTHENY are NP-hard;*
5. *computing a coalition manipulation of X is NP-hard, but of Y and XTHENY are polynomial;*
6. *computing a coalition manipulation of X is NP-hard and of Y is polynomial, but of XTHENY is NP-hard;*
7. *computing coalition manipulations of X and Y are NP-hard but of XTHENY is polynomial;*
8. *computing coalition manipulations of X, Y and XTHENY are NP-hard.*

Proof: The NP-hardness results are derived from the NP-hardness of computing a weighted coalition manipulation of STV with three or more candidates [7].

1. Consider $X = FirstRoundCup$ and $Y = Cup$ where $FirstRoundCup$ is the multi-winner rule that runs one round of the Cup voting rule. Note that $FirstRoundCup$THENCup is the Cup rule itself, and both $FirstRoundCup$ and Cup are polynomial to manipulate by a coalition even with weighted votes [7].
2. Consider $X = TopTwo$ and $Y = Plurality$ where $TopTwo$ elects the two candidates with the two highest plurality scores. With three or fewer candidates, $TopTwo$THEN$Plurality$ is plurality with runoff. This is NP-hard to manipulate by a weighted coalition with three or more candidates [7].
3. Consider $X = Plurality'$ and $Y = STV$ where $Plurality'$ is the decisive form of plurality that tie-breaks in favour of the manipulators. Note that XTHENY is again $Plurality'$ which is polynomial to manipulate by a coalition even with weighted votes [7].
4. Consider $X = Identity$ and $Y = STV$ where $Identity$ is the identity rule that elects all the candidates in the election. Note that XTHENY is also STV.
5. Consider $X = STV_1$ which is the multi-winner voting rule that elects both the STV winner and the candidate with the lexicographically smallest label, and Y elects the candidate with the lexicographically smallest label. Note that XTHENY always elects the candidate with the lexicographically smallest label. Such a rule is polynomial to manipulate by a coalition even with weighted votes.
6. Consider $X = STV$ and $Y = Identity$. Note that XTHENY is again STV.
7. Consider $X = STV_2$ and $Y = STV_3$ where STV_2 is the multi-winner rule that elects the STV winner as well as those candidates with the lexicographically smallest and largest names, and STV_3 elects the plurality winner from the candidates with the lexicographically smallest and largest names if

there are thress or fewer candidates and otherwise elects the STV winner. Note that XTHENY elects the plurality winner from the candidates with the lexicographically smallest and largest names. Computing a coalition manipulation of such a rule is polynomial even with weighted votes.

8. Consider $X = Y = STV'$ where STV' is the decisive form of STV where we tie-break in favour of the manipulators. Note that XTHENY is also STV'.

■

5.2 Weighted votes, specific rules

With weighted votes, we already know that several multi-stage voting rules are NP-hard to manipulate including STV, Plurality with runoff, Baldwin's rule (all with 3 candidates), and Nanson's rule (with 4 candidates) [7, 15]. We first show that computing a manipulation of $Condorcet$THENX with weighted votes is NP-hard for any scoring rule X. This contrasts to scoring rules in general where computing a coalition manipulation is NP-hard for any rule that is not isomorphic to Plurality, but is polynomial for Plurality. This demonstrates that adding the test for a Condorcet winner to give $Condorcet$THENX increases the computational complexity of manipulation over that for the scoring rule X alone.

PROPOSITION 7. *Deciding whether there exists a coalitional manipulation for the combination $Condorcet$THEN$Plurality$ with weighted votes is NP-complete with three or more candidates.*

Proof: We reduce from the number partitioning problem with n integers k_i, $i = 1, \ldots, n$, $\sum_{i=1}^{n} k_i = 2K$. We have n manipulators with the weight k_i each. We have three candidates, a, b and p (who the manipulators wish to make win), $4K$ votes cast by the non-manipulators and $2K$ to be cast by the manipulators. Suppose agents with total weight $2K$ cast (a, b, p) and agents with total weight $2K$ cast (b, a, p). The candidate p is a Condorcet loser as it loses to both a and b. Moreover, as a and b are tied, there is no Condorcet winner. Note that if all manipulators put p in the first position then p wins under plurality. However, the manipulators have to make sure that they also do not make a or b the Condorcet winner. Note that if a (b) gets a higher score than b (a) then a (b) is the Condorcet winner. Therefore, the only way to prevent one of them from becoming the Condorcet winner is to partition the total weight of votes between a and b. Thus, manipulators with a total weight of K have to vote (p, a, b) and the remaining manipulators have to vote (p, b, a). Therefore, there exists a manipulation iff there is a partition with the required sum K. ■

PROPOSITION 8. *With weighted votes and any scoring rule X that is not isomorphic to Plurality, computing a coalition manipulation of $Condorcet$THENX is NP-hard for three or more candidates.*

Proof: Without loss of generality, we consider a scoring rule which gives a score of α_1 for a candidate in 1st place in a vote, α_2 for 2nd place, and 0 for 3rd place. We adapt the reduction used in the proof of Theorem 6 in [7] for the NP-hardness of manipulating any scoring rule that is not isomorphic to Plurality voting. The reduction in [7] is from the number partitioning problem and constructs an election with a weight of $6\alpha_1 K - 2$ votes over the candidates a, b and p (who the manipulators wish to make win). Within these votes, the manipulators have a weight of $2(\alpha_1 + \alpha_2)K$ votes, and the rest are fixed. The number partition problem is to divide a set of integers summing to $2K$ into two equal sums. There is a manipulator of weight $(\alpha + 1 + \alpha_2)k_1$ for every integer k_i in the set being partitioned. We add $6\alpha_1 K - 1$ triples of votes to this election: (a, b, p), (b, p, a), (p, a, b). This has no impact on the differences in the scores between the candidates. However, it creates a Condorcet cycle so that there cannot be a Condorcet winner whatever the manipulators do with their votes. Hence, we must pass to the second round where the winner is decided by the scoring rule X. As in the proof of Theorem 6 in [7], there is a manipulation that makes p the winner of the scoring rule X iff there is a partition into two equal sums. Thus, computing a coalition manipulation of $Condorcet$THENX is NP-hard. ■

It follows immediately that coalition manipulation of Black's procedure, which is $Condorcet$THEN$Borda$ is NP-hard with three or more candidates.

COROLLARY 1. *With weighted votes, computing a coalition manipulation of Black's procedure is NP-hard with three or more candidates.*

5.3 Unweighted votes, general results

As with weighted votes, there is no connection in general between the computational complexity of computing a manipulation of a two stage voting rule with unweighted votes and the computational complexity of computing a manipulation of its parts.

PROPOSITION 9. *There exist voting rules X and Y with the following properties:*

1. *computing manipulations of X, Y and XTHENY are polynomial;*
2. *computing manipulations of X and Y are polynomial but of XTHENY is NP-hard;*
3. *computing a manipulation of X is polynomial and of Y is NP-hard, but of XTHENY is polynomial;*
4. *computing a manipulation of X is polynomial, but of Y and XTHENY are NP-hard;*
5. *computing a manipulation of X is NP-hard, but of Y and XTHENY are polynomial;*
6. *computing a manipulation of X is NP-hard and of Y is polynomial, but of XTHENY is NP-hard;*
7. *computing manipulations of X and Y are NP-hard but of XTHENY is polynomial;*
8. *computing manipulations of X, Y and XTHENY are NP-hard.*

Proof: The NP-hardness results are derived from the NP-hardness of manipulating STV with unweighted votes and a single manipulator [2].

1 Identical examples to the weighted case.

2 Consider the multi-winner voting rule X that eliminates a fixed candidate p, and the rule Y that elects the plurality winner between the candidates that are preferred by at least one agent to p or, if there are no such candidates, the STV winner. Now X is polynomial to manipulate as it ignores the votes. Similarly, Y is polynomial to manipulate since the manipulators should always put the candidate that they wish to win in first place, and p anywhere else in their vote. If all other agents prefer p to any other candidate, then this vote will ensure that the manipulators' preferred candidate wins. On the

other hand, if the other agents prefer one or more candidates to p, then this is the best vote for ensuring the manipulators' preferred candidate is the plurality winner. Now XTHENY is NP-hard to manipulate. We adapt the reduction used in [2] to prove that STV is NP-hard to manipulate by a single manipulator. We simply introduce an additional candidate, p into the voting profile used in this proof.

3-8 Identical examples to the weighted case. ■

5.4 Unweighted votes, specific rules

With unweighted votes, we already know that a number of specific multi-stage voting rules are NP-hard to manipulate including STV [2], Nanson's, Baldwin's [15] and Coombs rules [10]. We can add Black's procedure to this list of voting rules that are NP-hard to manipulate with unweighted votes. Like Borda voting on which it is based, a single manipulator can compute a manipulation of Black's procedure in polynomial time, but coordinating two manipulators makes the problem NP-hard [3, 9].

PROPOSITION 10. *Computing a manipulation of Black's procedure with unweighted votes and two manipulators is NP-hard.*

Proof: We adapt the reduction used in the proof of Theorem 3.1 in [3] for the NP-hardness of manipulating Borda voting. This reduction is from a special case of numerical matching with target sums. It constructs an election with 5 votes, 3 fixed votes and 2 votes of the manipulators over the candidates 1 to m. We add 6 sets of cyclic votes: $(1, 2, \ldots, m-1, m)$, $(2, 3 \ldots, m, 1)$, $\ldots$, $(m-1, m, \ldots, m-3, m-2)$, $(m, 1, \ldots, m-2, m-1)$. This has no impact on the differences in the scores between the candidates. However, it creates a Condorcet cycle so there cannot be a Condorcet winner however the two manipulators vote. Hence, we must pass to the second round where the winner is decided by the Borda rule. As in the proof of Theorem 3.1 in [3], there is a manipulation that makes a chosen candidate the Borda winner iff there is a solution to the numerical matching problem with target sums. Thus, computing a manipulation of $Condorcet$THEN$Borda$, which is Black's procedure, is NP-hard. ■

PROPOSITION 11. *Deciding whether one manipulator can make a candidate win for Black's procedure with unweighted votes is polynomial.*

Proof: Let p be the candidate who the manipulator wishes to win. We give a polynomial time greedy procedure which constructs a successful manipulation if one exists or fails (in which case there is no way to make p win). We put p in first place in the manipulator's vote. This is the best possible thing we can do to ensure p wins. If p is now necessarily the Condorcet winner, we are done; p will win however we complete the vote. Otherwise we consider the remaining positions in the manipulator's vote, starting with the 2nd (and most dangerous) position. We put the least dangerous candidate for Borda in this position that does not create a Condorcet winner. To do this, we consider the remaining candidates in reverse order of their Borda scores. We put in this position the first candidate that is not then necessarily a Condorcet winner or a candidate with a higher Borda score than p. If there is no such candidate, we fail. Otherwise, we move to the next position in the vote and repeat. ■

6. MULTIPLE BALLOTS

So far, we have assumed that agents vote only once. However, the THEN combinator is naturally sequential. We therefore consider the case where agents are allowed to re-vote in each round. For example, in the French presidential elections, voters re-vote in the second stage. Such re-voting increases the potential for manipulation in several different ways. First, there are elections which can only be manipulated when the manipulators vote differently in the two rounds. Of course, all those elections where manipulators can change the result by strategically voting just once remain manipulable. Second, the first round of voting reveals agents' preferences, thereby enabling manipulations to take place that require such knowledge. We discuss this more in the next section. Third, agents can vote strategically in the first round to give their preferred candidate an easier contest in the second round.

If agents re-vote between rounds, we use the term "re-voting" in the description of the voting rule. Hence, plurality with runoff and re-voting is the two stage election rule used in French presidential elections in which, unless there a majority in the first round, plurality is used in the first round to select two candidates to go through to the runoff, and agents then re-vote in the second round to decide the winner of the runoff. The following two results demonstrate that allowing agents to re-vote between rounds can either increase or decrease the computational complexity of computing a strategic vote. The first result also shows that there exist elections where strategic voting with plurality with runoff is not possible unless we permit re-voting between rounds.

PROPOSITION 12. *There exists a class of weighted elections on 3 candidates such that computing a coalition manipulation for plurality with run-off is polynomial when agents vote just once, but it is NP-hard to compute a coalition manipulation when the manipulating agents can re-vote in the run-off.*

Proof: We suppose that only the manipulating agents change their vote in the run-off. Reduction from number partitioning. We consider a bag of integers k_i with sum $2K$ and ask if we can divide them into two bags each with sum K. We construct an election which is not manipulable if agents vote just once (hence, it is polynomial to compute if there is a successful manipulation). However, a coalition of agents, each with vote of weight k_i can construct a successful manipulation of this election when they re-vote between rounds iff there is a perfect partition. There are 3 candidates, a, b and p. The manipulating coalition wish to make p win. The election has $2K$ fixed votes for (a, b, p), $2K$ fixed votes for (b, a, p), K fixed votes for (b, p, a), and $2K$ fixed votes for (p, a, b). We suppose that in the case of a 3-way tie in the first round, b is eliminated. Now, for p to win, b must be eliminated in the first round since $3K$ more agents prefer b to p and the coalition cannot over-turn this majority. Since a and p have $2K$ fixed votes and b has $3K$, this is only possible if we have a 3-way tie in the first round. This requires the manipulating coalition to vote so that there is a weight of K votes for a and for p (which corresponds to a perfect partition). Suppose that this is possible. If the manipulating coalition cannot re-vote in the run-off, then a wins with $5K$ votes against the $4K$ votes for p. However, if all the coalition re-vote in the run-off for p then p wins with $5K$ votes against the $4K$ votes for a. Hence, there is a successful manipulation in which p wins iff there is a perfect partition and the manipulators can re-vote in the run-off. ■

We also show the reverse: we identify a class of elections where plurality with run-off is polynomial to manipulate with re-voting in the run-off, but NP-hard when agents can vote just once.

PROPOSITION 13. *There exists a class of weighted elections on 4 candidates such that computing a coalition manipulation for plurality with run-off is NP-hard when agents vote just once, but it is polynomial to compute a coalition manipulation when agents can re-vote in the run-off.*

Proof: We reduce from number partitioning. We consider a bag of integers k_i with sum $2K$ and ask if we can divide them into two bags each with sum K. We construct an election which is easy to manipulate if agents can re-vote in the run-off, but it requires a solution to the partitioning problem to compute a manipulation if they vote just once. There are 4 candidates, a, b, c and p. The manipulating coalition, each with a weight of $2k_i$ wish to make p win. The election has $4K$ fixed votes for (p, a, b, c), $4K$ fixed votes for (a, b, c, p), $4K - 1$ fixed votes for (b, p, c, a), $2K$ fixed votes for (b, c, p, a), and $6K - 2$ fixed votes for (c, b, p, a). In the first round, both p and a have $4K$ votes, b has $6K - 1$, and c has $6K - 2$. If the manipulators can re-vote in the run-off then it is easy to construct a manipulation. The manipulators cast one vote for c and the rest for p. Suppose this one vote has weight $2k$ (where $k \geq 1$). Then p and c go through to the run-off as p has $8K - 2k$ votes, a has $4K$ votes, b has $6K - 2$ votes and c has $6K - 2 + 2k$ votes. In the run-off, if all the manipulators now vote for p then p wins with $12K - 1$ votes compared to $12K - 2$ votes for c.

On the other hand, if the manipulators cannot re-vote in the run-off, we argue that p cannot win unless a and p enter the run-off, and this is only possible if there is a perfect partition. Suppose b and p go through to the run-off. Then, irrespective of how the manipulators vote, b wins as there are $16K - 3$ fixed votes for b and only $4K$ fixed votes for p. Suppose c and p go through to the run-off. Then c must receive vote of weight $2k$ (where $k \geq 1$) from the manipulators to beat b in the first round. In the run-off, c then has $12K - 2 + 2k$ votes which is more than the $12K - 1 - 2k$ votes for p. Hence, c will win the run-off. Thus, for p to win, we just need to consider the case that p and a enter the run-off. The only way that p and a can enter run-off is if there is a perfect partition and agents in the manipulating coalition corresponding to one partition vote $(a, \ldots)$ whilst the other partition vote $(p, \ldots)$. Then p and a both have $6K$ votes and go through to the run-off. In the run-off, p wins with $18K - 3$ votes compared to the $6K$ votes for a. Hence, p wins iff there is a perfect partition. Thus computing a coalition manipulation is NP-hard when agents cannot re-vote in the run-off. ■

6.1 General results on re-voting

The last two results demonstrates that permitting agents to re-vote between rounds can either increase or decrease the computational complexity of computing a coalition manipulation. We might wonder if the same holds for unweighted votes. We argue here that, in the unweighted case, there is also no connection between the computational complexity of manipulating a two-stage voting rule with and without re-voting. We give two artificial two-stage voting rules where re-voting between rounds either increases or decreases the worst case complexity of computing a single strategic vote.

PROPOSITION 14. *There exists a two-stage voting rule such that computing a strategic vote by a single manipulator is NP-hard when the manipulator cannot re-vote between rounds but polynomial when the manipulator can.*

Proof: (Sketch) We reduce from the 1in3-SAT problem ("does there exist a truth assignment that satisfies exactly one in three literals in each clause?") on positive clauses. The reduction uses votes to represent clauses and a truth assignment. We use positive integers to represent literals, as well as the candidates 0 and -1 to identify the vote represent a truth assignment. By distinguishing between the non-negative integers, 0 and -1, our voting rules are not neutral (that is, are not invariant to the names of the candidates). They can, however, be made neutral but at the expense of greater complexity. For any vote in which 0 is not in first place, the first three non-zero positive candidates represent the positive literals in a clause. For any vote in which 0 is in first place, the non-zero positive candidates before -1 represent the positive literals in the truth assignment which are set to true, and all other positive literals are set to false. X is the rule that elects 0 if no vote represents a truth assignment which satisfies at least one in three literals in each clause represented by the other votes, otherwise it elects all candidates. On the other hand, Y is the rule that elects -1 if there is a vote representing a truth assignment which satisfies at most one in three literals in each clause, otherwise it elects 0. The non-manipulators cast votes that represent the clauses. When the manipulator cannot re-vote between rounds, a manipulator can make -1 win XTHENY iff there is a satisfying 1in3-SAT assignment. On the other hand, computing a manipulation of XTHENY is polynomial when the manipulator can re-vote between rounds. The manipulator simply puts 0 in first place and -1 in last place in the first round, and 0 in first place and -1 in second place in the second round. ■

PROPOSITION 15. *There exists a two-stage voting rule such that computing a strategic vote by a single manipulator is polynomial when the manipulator cannot re-vote between rounds but NP-hard when the manipulator can.*

Proof: We again reduce from 1in3-SAT on positive clauses and uses votes to represent clauses and a truth assignment in the same way as the last proof. As before, our voting rules are not neutral but could be made so with some additional complexity. X is the voting rule that elects 0 if there is any vote with 0 in first place otherwise elects every candidate. On the other hand, Y is the rule that elects -1 if there is a vote representing a truth assignment which satisfies exactly one in three literals in each clause, otherwise it elects 0. As before, the non-manipulators cast votes that represent the clauses. When the manipulator cannot re-vote between rounds, the only possible result of XTHENY is 0. On the other hand, when there is re-voting between rounds, a manipulator can make -1 win iff there is a satisfying 1in3-SAT assignment. The manipulator simply puts 0 in last place in the first round, and in the second round casts a vote corresponding to a satisfying 1in3-SAT truth assignment. ■

6.2 Specific results on re-voting

For commonly occurring two-stage voting rules like plurality with run-off and Black's procedure, we now show that re-voting has no impact on the worst case complexity of computing a strategic vote when we consider manipulating any possible profile of votes. Consider plurality with runoff. With unweighted votes, computing a manipulation of plurality with runoff with and without re-voting is polynomial. With weighted votes and three or more candidates, it is NP-hard to compute a coalition manipulation of plurality with run-off when agents vote just once [7]. With re-voting in the run-off, computing a coalition manipulation remains NP-hard with just three candidates.

PROPOSITION 16. *Computing a weighted coalition manipulation for plurality with run-off is NP-hard with three or more candidates when agents can re-vote in the run-off.*

Proof: We use the same reduction as in the proof of Theorem 9 in [7] which demonstrates that weighted coalition manipulation for plurality with run-off is NP-hard when agents vote just once, and do not re-vote in the run-off. We consider a number partitioning problem of the integers k_i with sum $2K$. There are 3 candidates, a, b and p with $6K - 1$ votes for (b, p, a), and $4K$ votes each for (a, b, p) and (p, a, b). The manipulating coalition wish to make p

win. Each manipulator has a vote of weight $2k_i$. As in [7], if there is a partition, then we can construct a manipulation in which agents corresponding to one partition vote (a, p, b) and to the other partition vote (p, a, b). Both a and p then advance to the run-off where p wins easily without the manipulators needing to re-vote. On the other hand, suppose there is a manipulation, with manipulators potentially re-voting in the run-off. As argued in [7], for p to win, both p and a must go through to the run-off since b will beat p in any run-off whatever the manipulators do. However, this is only possible if we have manipulating votes corresponding to a partition with half putting a in first place, and the other half putting p in first place. ■

We turn next to Black's procedure. Computing a coalition manipulation of Black's procedure with weighted votes remains NP-hard when we permit re-voting.

PROPOSITION 17. *Computing a weighted coalition manipulation of Black's procedure is NP-hard when there are three or more candidates and agents can re-vote in the second round.*

Proof: We use the same reduction as in the proof of Proposition 8. Whatever vote is cast by the manipulators in the first round, there is not a Condorcet winner. Hence all candidates proceed to the second round. In the second round, the agents re-vote and the desired candidate p wins iff the coalition cast new votes corresponding to a perfect partition. Thus computing a coalition manipulation of Black's procedure with re-voting is NP-hard. ■

With unweighted votes, re-voting also has no impact on the worst case complexity of computing a strategic vote.

PROPOSITION 18. *Computing a manipulation of Black's procedure by two manipulators is NP-hard when votes are unweighted and agents can re-vote in the second round, whilst it is polynomial with just one manipulator.*

Proof: With two manipulators, we use the same reduction as in the proof of Proposition 10 where the two manipulators cannot re-vote. With one manipulator, we construct a strategic vote for the first round by putting the desired candidate first, and reverse ordering the remaining candidates according to their Copeland scores. For the second round, we simply use the greedy method for approximating Borda manipulation that puts the desired candidate first, and reverse orders the remaining candidates according to their Borda scores. ■

7. REVEALED PREFERENCES

One of the (potentially strong) assumptions made in much work on (the computational complexity of) manipulation is that the manipulators know the other agents' preferences [8]. There are many situations where this is unrealistic. When we have re-voting, it may be reasonable to suppose agents' preferences have been (partially) revealed by the first round of voting. This introduces new opportunities for manipulation. Consider Black's procedure with re-voting and a manipulator who lacks any knowledge of the other agents' preferences, so votes truthfully in the first round. The following example demonstrates that this manipulator can vote strategically in the second round based on the votes revealed in the first round.

EXAMPLE 1. *Suppose the first round reveals that there are 2 votes for (a, b, p), 2 votes for (b, p, a), 1 vote for (p, a, b), and a single manipulator's truthful vote for (p, b, a). There is no Condorcet winner so all candidates go through to the second round. Without re-voting, b has the highest Borda score in the second round and is the overall winner. On the other hand, suppose the manipulator changes their vote in the second round to (p, a, b) based on the preferences revealed in the first round. Then, assuming the other votes remain the same, the Borda scores of all candidates are equal. If such a 3-way tie is broken in favour of the manipulator, then the manipulator's preferred candidate p now wins.*

It is natural to consider more game theoretic behaviours in such two stage voting rules. Re-voting can be viewed as a finite repeated sequential game so we can use concepts like subgame perfect Nash equilibrium and backward induction to predict how agents will play strategically in each round. An interesting open question is the computational complexity of computing such strategic behaviour. This sort of strategic voting has already received some attention in the literature. For example, Bag, Sabourian and Winter prove that a class of voting rules which use repeated ballots and eliminate one candidate in each round are Condorcet consistent [1]. They illustrate this class with the *weakest link* rule in which the candidate with the fewest ballots in each round is eliminated.

It is also natural to consider iterated voting in multiple stage voting rules. After each round of voting, we might suppose that agents change their vote according to a best response strategy, starting perhaps from a truthful vote. We can also consider the situation where the full preferences of the agents are revealed in each round of voting, as well as the situation where only partially information is revealed like total Borda scores. However, unlike previous studies like [14], candidates are also eliminated in each round.

8. RELATED WORK

As noted earlier, a number of well known voting rules like Black's procedure and Plurality with runoff are instances of this voting schema. However, there exist many other related voting rules. For example, Conitzer and Sandholm [5] studied the impact on the computational complexity of manipulation of adding an initial round of the Cup rule to a voting rule X. This initial round eliminates half the candidates and makes manipulation NP-hard to compute for several voting rule including plurality and Borda. Consider the multi-winner voting rule, $Bisect$ which runs an election between given pairs of candidates, and returns the winning half of the candidates. Then Conitzer and Sandholm's study can be viewed as of the two stage voting rule $Bisect$THENX.

Elkind and Lipmaa [11] extended Conitzer and Sandholm's idea of making manipulation computationally hard to computer by adding a pre-round of the cup rule. They proposed a general technique for combining together any 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.

As mentioned previously, Davies *et al.* have recently considered a general class of voting rules in which a base rule is repeatedly applied to eliminate the least popular candidates [10]. STV, Baldwin's and Coombs rules can all be viewed as instances of this general schema. For example, STV is $eliminate(plurality)$, Baldwin's rule is $eliminate(Borda)$, and (a variant of) Coombs is $eliminate(veto)$. Successively eliminating candidates in this way can often increase the computational complexity of computing a manipulation.

Beside STV, Nanson's, Baldwin's and Coombs rule, a number of other recursively defined rules have been put forwards in the literature. For example, Tideman proposed the Alternative Smith rule [18]. This is recursively defined as

$$SmithSet\text{THEN}(PluralityLoser\text{THEN}AlternativeSmith)$$

Other complex multi-stage rules have also been proposed. For example, [13] has proposed a complex rule that computes the Schwartz choice set, then iteratively applies Copeland's procedure until a fixed point is reached. If several candidates remain at this point, the rule then selects the plurality winners. If there are several such winners, the rule then chooses among then according to the number of second place votes, and so on. If this still does not select a winner, a lottery is used amongst the candidates that remain.

Finally, as mentioned previously, Narodytska *et al.* have also considered a parallel combinator for taking the consensus of two (or more) voting rules [16]. Given two voting rules X and Y, the parallel combinator $X + Y$ computes the winners of X and Y and then recursively applies $X + Y$ to this set. If X and Y are majority consistent (that is, given an election with just two candidates, they both return the majority winner) then $X + Y$ is $(X \text{OR} Y)\text{THEN}Majority$ where $X \text{OR} Y$ returns the union of the winners of X and Y.

9. CONCLUSIONS

We have considered voting rules which have two stages. For example, Black's procedure selects the Condorcet winner if they exist, otherwise in the second stage, it selects the Borda winner. We denoted this as $Condorcet\text{THEN}Borda$. Combining voting rules together in this way can increase their resistance to manipulation. For example, whilst Plurality is polynomial to manipulate with weighted votes, $Condorcet\text{THEN}Plurality$ is NP-hard with three or more candidates and a coalition of manipulators. A combination of voting rules can also inherit computational resistance to manipulation from its parts. For example, we proved that computing a manipulation of Black's procedure, which is the combination $Condorcet\text{THEN}Borda$, is NP-hard with weighted or unweighted votes. We have also considered the impact of allowing agents to re-vote between rounds. Our results are mostly worst case and may not reflect the difficulty of computing a manipulation on average or in practice (see, for instance, [6, 17, 19, 20]). Thus this work should be a first step in understanding the computational complexity of manipulating two stage voting rules. There are many other directions for future work. For instance, it would also be interesting to consider the impact of such two stage voting on other types of control, on bribery and on issues like the computation of possible winners.

10. ACKNOWLEDGEMENTS

NICTA is funded by the Department of Broadband, Communications and the Digital Economy, and the Australian Research Council. The authors are also supported by the Asian Office of Aerospace Research and Development (AOARD 124056).

11. REFERENCES

[1] P.K. Bag, H. Sabourian, and E. Winter. Multi-stage voting, sequential elimination and Condorcet consistency. *Journal of Economic Theory*, 144(3):1278 – 1299, 2009.

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

[3] N. Betzler, R. Niedermeier, and G.J. Woeginger. Unweighted coalitional manipulation under the Borda rule is NP-hard. In *Proc. of the 22nd International Joint Conf. on Artificial Intelligence (IJCAI 2011)*. 2011.

[4] F. Brandt, M. Brill, E. Hemaspaandra, and L.A. Hemaspaandra. Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. In *Proc. of the 24th AAAI Conf. on Artificial Intelligence (AAAI 2010)*. 2010.

[5] V. Conitzer and T. Sandholm. Universal voting protocol tweaks to make manipulation hard. In *Proc. of 18th IJCAI*, pages 781–788. 2003.

[6] V. Conitzer and T. Sandholm. Nonexistence of voting rules that are usually hard to manipulate. In *Proc. of the 21st National Conf. on Artificial Intelligence (AAAI 2006)*, pages 627–634. 2006.

[7] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate. *Journal of the Association for Computing Machinery*, 54, 2007.

[8] V. Conitzer, T. Walsh, and L. Xia. Dominating manipulations in voting with partial information. In *Proc. of the 25th AAAI Conf. on Artificial Intelligence (AAAI 2011)*. AAAI Press, 2011.

[9] J. Davies, G. Katsirelos, N. Narodytska, and T. Walsh. Complexity of and Algorithms for Borda Manipulation. In *Proc. of the 25th AAAI Conf. on Artificial Intelligence (AAAI 2011)*. AAAI Press, 2011.

[10] J. Davies, N. Narodytska, and T. Walsh. Eliminating the weakest link: Making manipulation intractable? In *Proc. of the 26th AAAI Conf. on Artificial Intelligence (AAAI 2012)*. 2012.

[11] E. Elkind and H. Lipmaa. Hybrid voting protocols and hardness of manipulation. In *Proc. of the 16th Annual International Symposium on Algorithms and Computation (ISAAC'05)*, 2005.

[12] P.C. Fishburn. Condorcet social choice functions. *SIAM Journal on Applied Mathematics*, 33(3):469–489, 1977.

[13] C.G. Hoag and G.H. Hallett. *Proportional Representation*. Macmillan, 1926.

[14] O. Lev and J.S. Rosenschein. Convergence of iterative voting. In *The 11th International Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, 2012.

[15] N. Narodytska, T. Walsh, and L. Xia. Manipulation of Nanson's and Baldwin's rules. In *Proc. of the 25th AAAI Conf. on Artificial Intelligence (AAAI 2011)*. AAAI Press, 2011.

[16] N. Narodytska, T. Walsh, and L. Xia. Combining voting rules together. In *Proc. of the 20th European Conf. on Artificial Intelligence (ECAI-2012)*. 2012.

[17] A. D. Procaccia and J. S. Rosenschein. Junta distributions and the average-case complexity of manipulating elections. *Journal of Artificial Intelligence Research*, 28:157–181, 2007.

[18] N. Tideman. *Collective Decisions And Voting: The Potential for Public Choice*. Ashgate, 2006.

[19] T. Walsh. Where are the really hard manipulation problems? The phase transition in manipulating the veto rule. In *Proc. of the 21st International Joint Conf. on Artificial Intelligence (IJCAI-2009)*, pages 324–329. 2009.

[20] T. Walsh. An empirical study of the manipulability of single transferable voting. In *Proc. of the 19th European Conf. on Artificial Intelligence (ECAI-2010)*, pages 257–262. 2010.