Strategic Voting in Negotiating Teams

Leora Schmerler
Department of Computer Science
Ariel University
Israel
leorabach@gmail.com

Noam Hazon
Department of Computer Science
Ariel University
Israel
noamh@ariel.ac.il

ABSTRACT

A negotiating team is a group of two or more agents who join together as a single negotiating party because they share a common goal related to the negotiation. Since a negotiating team is composed of several stakeholders, represented as a single negotiating party, there is need for a voting rule for the team to reach decisions. In this paper, we investigate the problem of strategic voting in the context of negotiating teams. Specifically, we present a polynomial time algorithm that finds a manipulation for a single voter when using a positional scoring rule. We show that the problem is still tractable when there is a coalition of manipulators that uses an approval-based rule. The coalitional manipulation problem becomes computationally hard when using Borda, but we provide a polynomial time algorithm with the following guarantee: given a manipulable instance with k manipulators, the algorithm finds a successful manipulation with at most one additional manipulator. Our results hold for both constructive and destructive manipulations.

KEYWORDS

Voting, negotiation, manipulation

ACM Reference Format:
Leora Schmerler and Noam Hazon. 2021. Strategic Voting in Negotiating Teams. In *The 3rd Games, Agents, and Incentives Workshop (GAIW 2021). Held as part of the Workshops at the 20th International Conference on Autonomous Agents and Multiagent Systems., London, UK, May 2021*, IFAAMAS, 12 pages.

1 INTRODUCTION

Voting is a common way to combine the preferences of several agents in order to reach a consensus. While being prevalent in human societies, it has also played a major rule in multi-agent systems for applied tasks such as multi-agent planning [16] or aggregating search results from the web [14]. In its essence, a voting process consists of several voters along with their ranking of the candidates, and a voting rule, which needs to decide on a winning candidate or on a winning ranking of the candidates.

Another common mechanism for reaching an agreement among several agents is a negotiation [19]. In a negotiation there is a dialogue between several agents in order to reach an agreement that is beneficial for all of them. Extensive work has been invested in developing negotiation protocols for many settings, but bilateral negotiations, where there are only two negotiating parties, is the most common type of negotiations [6]. Many works have focused on the case where each negotiating party represents a single agent. However, there are many cases in which a negotiating party represents more than one individual.

For example (motivated by [25]), consider an agricultural cooperative that negotiates with the government. Even though the members of the cooperative have a common goal, they may have different preferences regarding the prohibition of importing products, government supervision of prices, insect control, tax concessions, etc. As another example, consider the government of the United Kingdom that negotiates with the European Union (EU) regarding withdrawal from the EU (i.e., the Brexit). The members of the EU have similar interests and objectives, and thus they are considered a single party in the negotiation process. Nevertheless, the EU is composed of different countries, and they may have different preferences regarding sovereignty, migrants and welfare benefits, economic governance, competitiveness, etc. These situations are denoted by social scientists as negotiating teams, in which *a group of two or more interdependent persons join together as a single negotiating party because their similar interests and objectives relate to the negotiation* [7].

Since a negotiating team comprises several stakeholders represented as a single negotiating party, there is need for a coordination mechanism, and a voting rule is a natural candidate. Ideally, the voters report their true preferences so that the voting rule will be able to choose the most appropriate outcome. However, as shown by Gibbard (1973) and Satterthwaite (1975), every reasonable voting rule with at least 3 candidates is prone to strategic voting. That is, voters might benefit from reporting rankings different from their true ones. Clearly, this problem of manipulation also exists in a negotiating team. For example, suppose that there is a EU council committee that negotiates with the UK on agricultural and fishery policies. The committee may decide that the UK will be excluded from the agricultural policy due to Brexit, or the UK will still be included. Similarly, the committee may decide that the fishery policy no longer applies to the UK or include the UK. Therefore, there are 4 possible outcomes, denoted by o_1, o_2, o_3 and o_4. Now, suppose that Germany prefers o_1 over o_2, o_2 over o_3, and o_3 over o_4. We may assume that the preferences of the UK government are publicly known, and it is also possible that Germany, which currently holds the presidency of the EU council, is familiar with the preferences of the other EU council members. Since the negotiation protocol usually is also known, Germany might be able to reason that o_3 is the negotiation result, but if Germany will vote strategically and misreport its preferences then o_2 will be the negotiating result. To the best of our knowledge, the analysis of manipulation in the context of negotiating teams has not been investigated to date.

In this paper, we investigate manipulation in the context of negotiating teams. We assume that there is a negotiation process between two parties. One of the parties is a negotiating team, and the team uses a voting rule to reach a decision regarding its negotiation strategy. Specifically, the negotiating team uses a positional scoring rule as a *social welfare function (SWF)*, which outputs a complete preference order. This preference order represents the negotiating party, and is the input in the negotiation process. We thus assume that there is a negotiation protocol that can work with ordinal preferences. We use the *Voting by Alternating Offers and Vetoes (VAOV)* protocol [1], since it is intuitive, easy to understand, and the negotiation result is Pareto optimal. Moreover, Erlich et al. (2018) have shown that we can identify the negotiation result of the VAOV protocol if both parties follow a sub-game perfect equilibrium with an intuitive procedure.

We analyze two types of manipulation, constructive and destructive. We begin by studying constructive manipulation by a single voter, where there is a single manipulator that would like to manipulate the election so that a preferred candidate will be elected as a result of the negotiation. We show that, surprisingly, placing the preferred candidate in the highest position in the manipulative vote is not always the optimal strategy, unlike in the traditional constructive manipulation of scoring rules, and we provide a polynomial time algorithm to find a manipulation (or decide that such a manipulation does not exist). We then analyze the constructive coalitional manipulation problem, where several voters collude and coordinate their votes so that an agreed candidate will be the negotiation result. We show that this problem is still tractable for approval-based rules, but it becomes computationally hard for Borda. However, we provide a polynomial time algorithm for the coalitional manipulation of Borda with the following guarantee: given a manipulable instance with k manipulators, the algorithm finds a successful manipulation with at most one additional manipulator. Finally, we show that our hardness result and algorithms can be adapted for destructive manipulation problems, where the goal of the manipulation is to prevent a candidate from being the result of the negotiation.

The contribution of this work is twofold. First, it provides an analysis of a voting manipulation in the context of negotiating teams, a problem that has not been investigated to date. Our analysis also emphasizes the importance of analyzing voting rules within an actual context, because it leads to new insights and a deeper understanding of the voting rules. Second, our work concerns the manipulation of SWF, which has been scarcely investigated.

2 PRELIMINARIES

We assume that there is a set of outcomes, O, $|O| = m$ and a set of voters $V = \{1, ..., n\}$. Each voter i is represented by her preference p_i, which is a total order over O. We write $o >_{p_i} o'$ to denote that outcome o is preferred over outcome o' according to p_i. The position of outcome o in preference p_i, denoted by $pos(o, p_i)$, is the number of outcomes that o is preferred over them in p_i. That is, the most preferred outcome is in position $m-1$ and the least preferred outcome is in position 0. We also refer to the outcomes of O as candidates, and to the total orders over O as votes.

A preference profile is a vector $\vec{p} = (p_1, p_2, ..., p_n)$. In our setting we are interested in a *resolute social welfare function*, which is a mapping of the set of all preference profiles to a single strict preference order. A scoring vector for m candidates is $\vec{s} = (s_{m-1} \dots, s_0)$, where every s_i is a real number and $s_{m-1} \geq \dots \geq s_0 \geq 0$. A scoring vector essentially defines a voting rule for m candidates: each voter awards s_i points to the candidate in position i. Then, when using the rule as a SWF, the candidate with the highest aggregated score is ranked first, the candidate with the second highest score is ranked second, etc. Since ties are possible, we assume that a lexicographical tie-breaking rule is used. We study *positional scoring rules*, where each rule in this family applies an appropriate scoring vector for each number of candidates. That is, a scoring rule is represented by a function f such that for each $m \in \mathbb{N}$, $f(m) = (s^m_{m-1} \dots, s^m_0)$ is a scoring vector for m candidates. Some of our results hold only for x-approval rules, in which $f(m) = (1, \dots, 1, 0, \dots, 0)$, where the number of 1's is x. Note that the well-known *Plurality* rule (where each voter awards one point to her favorite candidate) is 1-approval and *Veto* rule (where each voter awards one point to all the candidates, except for the least preferred one) is $(m$-1)-approval and they are thus both x-approval rules. We also analyze the Borda rule, where each voter awards the candidate a score that equals the candidate's position, i.e., $f(m) = (m-1, m-2, \dots, 1, 0)$. In general, we denote the resulting social welfare function $\mathcal{F}$.

In the negotiation process we assume there are two parties: t is the negotiating team, which comprises a set of voters, and there is another party. The parties negotiate over the set of outcomes O, and their preferences are also total orders over O. However, since t is a negotiating team that comprises several stakeholders, the preference order of t, p_t, is determined by the social welfare function over the preference profile of the members of t, that is, $p_t = \mathcal{F}(\vec{p})$. We denote by p_o the preference order of the other party.

We assume that negotiating parties use the *Voting by Alternating Offers and Vetoes (VAOV)* protocol [1], which is a negotiation protocol that works with ordinal preferences. The protocol works as follows. Let p_1 be the party that initiates the negotiation and let p_2 be the other party. At round 1, party p_1 offers an outcome $o \in O$ to p_2. If p_2 accepts, the negotiation terminates successfully with o as the result of the negotiation. Otherwise, party p_2 offers an outcome $o' \in O \setminus \{o\}$. If p_1 accepts, the negotiation terminates successfully with o' as the result of the negotiation. Otherwise, p_1 offers an outcome $o'' \in O \setminus \{o, o'\}$ to p_2, and so on. If no offer was accepted until round m then the last available outcome is accepted in the last round as the result of the negotiation. We further assume that the negotiating parties are rational and each party has full information on the other party's preferences. Therefore, the parties will follow a sub-game perfect equilibrium (SPE) during the negotiation. Anbarci (1993) showed that if both parties follow an SPE the negotiation result will be unique. We can thus call this outcome the SPE result.

In some negotiation settings the parties are cooperative. There are also settings where there is a central authority that can force the parties to offer specific outcomes in a specific order. In both cases it is common to use a bargaining rule, which is a function that assigns each negotiation instance a subset of the outcomes that is considered the result of the negotiation. One such bargaining rule is the *Rational Compromise (RC)* bargaining rule [21]. Let $A^j_{(p_t)} = \{$the j most preferred outcomes in p_t}. $A^j_{(p_o)}$ is defined similarly for p_o. *RC* is computed as follows:

(1) Let $j = 1$
(2) If $|A^j_{(p_t)} \cap A^j_{(p_o)}| > 0$ then return $A^j_{(p_t)} \cap A^j_{(p_o)}$.
(3) Else, $j \leftarrow j + 1$ and go to line 2.

Note that the *RC* bargaining rule is equivalent to Bucklin voting with two voters and no tie-breaking mechanism. An important finding of [17] shows that the negotiation result of the VAOV protocol if both parties follow an SPE (i.e., the SPE result) is always part of the set returned by *RC* rule. We use this connection between *RC* and the VAOV negotiation protocol whenever we need to identify the SPE result. Specifically, if *RC* returns one outcome, this is also the SPE result. If *RC* returns two outcomes then the SPE result depends on the number of outcomes and on the party that initiates the negotiation. We assume that the decision of which party initiates the negotiation is not always known in advance. Therefore, we write $\mathcal{N}$ to denote the mapping from two preference orders, p_t and p_o, to the possible SPE results. That is, $\mathcal{N}$ equals the set returned by *RC*.

3 CONSTRUCTIVE MANIPULATION BY A SINGLE VOTER

We begin by studying the problem of constructive manipulation by a single voter. We assume that a manipulator v' would like to manipulate the election so that a preferred candidate p will be the SPE result, and that the SPE result will not depend on the identity of the party that initiates the negotiation. Therefore, we require that $\mathcal{N}$ consists of exactly the preferred candidate. The Constructive Manipulation in the context of Negotiations (C-MaNego) is defined as follows:

Definition 3.1 (C-MaNego). We are given social welfare function $\mathcal{F}$, a preference profile $\vec{p}$ of voters on the negotiating team t, the preference of the other party p_o, a specific manipulator v', and a preferred candidate $p \in O$. We are asked whether a preference order $p_{v'}$ exists for the manipulator v' such that $\mathcal{N}(\mathcal{F}(\vec{p} \cup p_{v'}), p_o) = \{p\}$.

We first observe that manipulation problems in the context of negotiations are inherently different from the traditional voting manipulation problems. First, in voting manipulation there is one set of voters in which their preferences are the inputs of the voting rule. The manipulator only needs to take these preferences into account when she decides on her manipulative vote. In our case there are two stages: in the first stage there is a set of voters and in the second stage there are two negotiating parties, and the manipulator needs to consider the preferences of all of these agents when she decides on her manipulative vote. In addition, unlike constructive manipulation in many voting rules, placing the preferred candidate p in the highest position in the manipulative vote is not always the optimal strategy. Indeed, the following example describes a scenario where there is no manipulation where p is placed in the highest position but a manipulation is possible if p is placed in the second highest position. Assume that p_o is the following preference order: $p_o = b \succ p \succ a \succ c$. There is one manipulator v', and $\vec{p}$ comprises 4 voters with the following preferences: $p \succ c \succ a \succ b, p \succ b \succ a \succ c$, $b \succ p \succ a \succ c$, $b \succ a \succ c \succ p$. Assume that we use the Borda rule, and thus the voters of $\vec{p}$ give the following scores: b gets 8 points, p gets 8 points, a gets 5 points and c gets 3 points. Since we assume that the tie-breaking rule is a lexicographical order, $p_t = b \succ p \succ a \succ c$. In order to find a successful manipulation v' needs to make sure that b will not be in the two highest positions in $\mathcal{F}(\vec{p} \cup p_{v'})$. Now, if the manipulator places p in the highest position then p gets 11 points. Then, placing the other candidates in every possible order results in b in the second highest positions in $\mathcal{F}(\vec{p} \cup p_{v'})$. Alternatively, if v' votes as follows: $a \succ p \succ c \succ b$, then p gets 10 points, a and b get 8 points, and c gets 4 points; thus $\mathcal{F}(\vec{p} \cup p_{v'}) = p \succ a \succ b \succ c$. Now the SPE result is p.

We now present a polynomial time algorithm for C-MaNego with any scoring rule. Let p^a be the order that the algorithm finds (i.e., p^a is a possible $p_{v'}$), and let $p^a_t = \mathcal{F}(\vec{p} \cup p^a)$. Note that during the algorithm we use $\mathcal{F}(\vec{p} \cup p^a)$, where p^a is not a complete preference order. In these situations we assume that all of the candidates that are not in p^a get a score of 0 from p^a. Given i, let H be the i most preferred outcomes in p_t that do not belong to $A^i_{(p_o)}$. If p is not in H we replace the least preferred outcome in H with p.

Our algorithm works as follows. It uses the connection between *RC* and the negotiation protocol to identify the SPE result. Clearly, if the position of p in p_o is less than $\lceil m/2 \rceil$ then for any possible p^a *RC* does not return p. Therefore, there is no manipulation and the algorithm returns false (lines 1-2). Otherwise, we use the variable i to indicate the iteration number in which *RC* terminates. Thus, the algorithm iterates over the values of i from 1 to $\lceil m/2 \rceil$ (line 3). For any given i, we need to make sure that no outcome from $A^i_{(p_o)}$ will be placed in the i highest positions in p^a_t. Consequently, the algorithm places the outcomes from H in the highest positions and they receive the highest scores. Moreover, the outcomes are placed in a reverse order (with regards to their order in p_t) to ensure that even the least preferred outcome in H will receive a score that is as high as possible (in order to be included in the highest positions in p^a_t). Then, the algorithm places the remaining outcomes, denoted C, so that they will not prevent p from being the negotiation result (lines 6-14). Specifically, the algorithm places a candidate from C in the highest available position in p^a (line 9). If p is still the negotiation result, then the algorithm updates the set C (line 11) and proceeds to place another candidate from C in the next highest available position in p^a (line 12). If there is a position in p^a where no candidate from C can be placed (while keeping p the negotiation result), then there is no manipulation where *RC* terminates in iteration i and the algorithm proceeds to the next iteration (line 13).

THEOREM 3.2. *Algorithm 1 correctly decides the C-MaNego problem with any positional scoring rule in polynomial time.*

PROOF. Clearly, the algorithm runs in polynomial time since there are three loops, where each loop iterates at most m times. In addition, if the algorithm successfully constructs a manipulation order, p will be the negotiation result. We need to show that if an order that makes p the negotiation result exists, then our algorithm will find such an order. Assume that we have a manipulative vote, p^m, that makes p the negotiation result, and let $p^m_t = \mathcal{F}(\vec{p} \cup p^m)$. Thus, $\mathcal{N}(p^m_t, p_o) = \{p\}$. In addition, given a set H let $L = \{\ell | \exists h \in H \text{ s.t. } h \prec_{p_t} \ell\}$ and $R = \{o | o \in O, o \notin H \text{ and } o \notin L\}$.

We show that Algorithm 1 returns p^a in line 14, when i equals the iteration in which *RC* terminates given p^m_t and p_o. There are two possible cases to consider:

ALGORITHM 1: Constructive manipulation by a single voter

```
if $pos(p, p_o) < \lceil m/2 \rceil$ then
    return false
for $i = 1$ to $\lceil m/2 \rceil$ do
    define $H$ according to the given $i$
    $p^a \leftarrow H$ in a reverse order of the positions in $p_t$
    $C \leftarrow \{O \setminus H\}$
    for $j = |C| - 1$ to $0$ do
        foreach $c \in C$ do
            place $c$ in $p^a$ such that $pos(c, p^a) = j$
            if $\mathcal{N}(\mathcal{F}(\vec{p} \cup p^a), p_o) = \{p\}$ then
                remove $c$ from $C$
                exit the foreach loop and goto line 7
        break the loop and goto line 3
    return $p^a$    /* $C$ is empty */
return false
```

- $A^i_{(p^a)} = A^i_{(p^m)}$: according to Algorithm 1, $A^i_{(p^a)} = H$, and since $A^i_{(p^a)} = A^i_{(p^m)}, A^i_{(p^m)} = H$. By definition, $\forall r \in R$ and $\forall h \in H$, $r \prec_{p_t} h$ and $r \prec_{p^m} h$. Since we use a scoring rule, $\forall r \in R$ and $\forall h \in H$, $r \prec_{p^m_t} h$. Since p^m is a successful manipulation and *RC* terminates at iteration i, then $\forall \ell \in L$ where $\ell \in A^i_{(p_o)}$, $\ell \notin A^i_{(p^m_t)}$. For any other $\ell \in L$ we know that $p \prec_{p_t} \ell$ and for any $h \in H \setminus \{p\}$, $\ell \prec_{p_t} h$. Since p^m is a successful manipulation and *RC* terminates at iteration i, $p \in A^i_{(p^m_t)}$. Overall, $A^i_{(p^m_t)} = H$.

We first assume that all the candidates that are not in H get a score of 0 from p^a, and we show that $A^i_{(p^a_t)} = H$. For any $h \in H$, if $pos(h, p^a) \geq pos(h, p^m)$ then $pos(h, p^a_t) \geq pos(h, p^m_t)$. Otherwise, let $h \in H$ be a candidate such that $pos(h, p^a) < pos(h, p^m)$ and let $s = pos(h, p^a)$. There are $m - s - 1$ candidates from H above h in p^a. According to the pigeonhole principle, there is at least one candidate, denoted h', that is placed in p^m at position s or lower. That is, $pos(h', p^m) \leq pos(h, p^a)$. By the algorithm construction, all of the candidates that are ranked higher than h in p^a are ranked lower than h in p_t. That is, $h' \prec_{p_t} h$. However, $h' \in A^i_{(p^m_t)}$ and thus $h \in A^i_{(p^a_t)}$. Overall, $A^i_{(p^a_t)} = H$. Note that the previous argument shows an even stronger claim. Given two candidates c, c' and two corresponding preference orders x, x', we write $(c, x) \geq (c', x')$ when the score of c in x is greater than the score of c' in x', or when these scores are equal but either $c = c'$ or c is preferred over c' according to the lexicographical tie-breaking rule. Now, let $h^a \in H$ be the candidate that has the lowest position in p^a_t. Similarly, let $h^m \in H$ be the candidate that has the lowest position in p^m_t. By the algorithm construction, $(h^a, p^a_t) \geq (h^m, p^m_t)$.

We now show that Algorithm 1 (lines 7-13) can assign scores to all the candidates that are not in H such that p^a is a full preference order and it is a successful manipulation. Assume by contradiction that Algorithm 1 does not succeed in fully constructing p^a. That is, the loop is terminated in line 13 and at this stage $C \neq \emptyset$. Let $c \in C$ be the most preferred candidate of C according to p^m. Assume that we complete the preference order p^a by placing c in the highest position that is available in p^a, and by placing the other candidates from C arbitrarily. Since $A^i_{(p^m_t)} = H$, then $pos(h^m, p^m_t) > pos(c, p^m_t)$. That is, $(h^m, p^m_t) \geq (c, p^m_t)$. In addition, we showed that $(h^a, p^a_t) \geq (h^m, p^m_t)$. Finally, since $C \subseteq \{o \in O : pos(o, p^m) \leq pos(c, p^m)\}$, then $pos(c, p^m) \geq pos(c, p^a)$. Therefore, $(h^a, p^a_t) \geq (c, p^a_t)$. That is, $pos(h^a, p^a_t) > pos(c, p^a_t)$. Consequently, if we place c in the highest position that is available in p^a then $\mathcal{N}(\mathcal{F}(\vec{p} \cup p^a), p_o) = \{p\}$. However, we assume that Algorithm 1 terminates in line 13 without fully constructing p^a. That is, for every $c \in C$, if we place c in the highest position that is available in p^a, then $\mathcal{N}(\mathcal{F}(\vec{p} \cup p^a), p_o) \neq \{p\}$, a contradiction.

- $A^i_{(p^a)} \neq A^i_{(p^m)}$: let $p^{m\prime}$ be the manipulation p^m with the following changes: each $r \in A^i_{(p^m_t)} \setminus H$ is replaced with a candidate $h_r \in H \setminus A^i_{(p^m_t)}$. That is, $pos(r, p^{m\prime}) = pos(h_r, p^m)$ and $pos(h_r, p^{m\prime}) = pos(r, p^m)$. Since p^m is a successful manipulation, if $r \in A^i_{(p^m_t)} \setminus H$ then $r \notin A^i_{(p_o)}$. Thus, by the definition of H, $\forall r \in A^i_{(p^m_t)} \setminus H$ and $\forall h \in H \setminus A^i_{(p^m_t)}, pos(r, p_t) < pos(h, p_t)$. Therefore, since each $r \in A^i_{(p^m_t)} \setminus H$ is ranked in the highest i positions in p^m_t, then h_r is ranked in the highest i positions in $p^{m\prime}_t$. Similarly, since each h_r is not ranked in the highest i positions in p^m_t, then r is not ranked in the highest i positions in $p^{m\prime}_t$. That is, $h_r \in A^i_{(p^{m\prime}_t)}$ and $r \notin A^i_{(p^{m\prime}_t)}$, and thus, $H = A^i_{(p^{m\prime}_t)}$. Let $p^{m\prime\prime}$ be the manipulation $p^{m\prime}$ with the following changes: each $r \in A^i_{(p^{m\prime})} \setminus H$ is replaced with a candidate $h_r \in H \setminus A^i_{(p^{m\prime})}$. That is, $A^i_{(p^{m\prime\prime})} = H$. Note that $c \notin A^i_{(p^{m\prime}_t)}$ for every $c \in \{O \setminus H\}$, and therefore $c \notin A^i_{(p^{m\prime\prime}_t)}$. Thus, $A^i_{(p^{m\prime\prime}_t)} = H$. That is, $p^{m\prime\prime}$ is a successful manipulation, and $A^i_{(p^{m\prime\prime})} = A^i_{(p^a)}$. This brings us back to the first case we already considered and showed that p^a is a successful manipulation. □

4 CONSTRUCTIVE COALITIONAL MANIPULATION

We now consider the problem of constructive manipulation by a coalition of voters. That is, several manipulators, denoted by M, might decide to collude and coordinate their votes in such a way that an agreed candidate p will be the SPE result. The constructive coalitional manipulation problem is defined as follows:

Definition 4.1 (CC-MaNego). Given a social welfare function $\mathcal{F}$, a preference profile $\vec{p}$ of the voters of negotiating team t, the preference of the other party p_o, a number of manipulators k, and a preferred candidate $p \in O$, we check whether a preference profile $\vec{p}_M$ for the manipulators exists such that $\mathcal{N}(\mathcal{F}(\vec{p} \cup \vec{p}_M), p_o) = \{p\}$.

We show that CC-MaNego can be decided in polynomial time for any x-approval rule using Algorithm 2, which works as follows. Similarly to Algorithm 1, the algorithm iterates over the possible values of i, where i indicates the iteration number in which *RC* terminates. For any given i, the algorithm iterates over the number of manipulators and determines their votes (Lines 6-20). We refer to each of these iterations as a *stage* of the algorithm. In each stage, a vote of one manipulator is determined, denoted by p^a. We begin with an empty set of votes, $\vec{p}_M$. Then, the algorithm places the outcomes from H in the highest positions in p^a. The outcomes are placed in a reverse order, with regards to their order in $\mathcal{F}(\vec{p} \cup \vec{p}_M)$. Similarly, the algorithm places all the other outcomes in the lowest positions in p^a and the outcomes are placed in a reverse order, with regards to their order in $\mathcal{F}(\vec{p} \cup \vec{p}_M)$. Note that the set H does not change throughout

ALGORITHM 2: Coalitional manipulation

```
if $pos(p, p_o) < \lceil m/2 \rceil$ then
    return false
for $i = 1$ to $\lceil m/2 \rceil$ do
    define $H$ according to the given $i$
    $\vec{p}_M \leftarrow []$
    for $\ell = 1$ to $|M|$ do
        $p^a \leftarrow$ empty preference order
        $C \leftarrow H$
        for $j = 1$ to $|H|$ do
            $c \leftarrow$ the least preferred outcome from $C$ under $\mathcal{F}(\vec{p} \cup \vec{p}_M)$
            place $c$ in $p^a$ such that $pos(c, p^a) = m - j$
            $j \leftarrow j + 1$
            remove $c$ from $C$
        $C \leftarrow \{O \setminus H\}$
        for $j = 1$ to $|\{O \setminus H\}|$ do
            $c \leftarrow$ the most preferred outcome from $C$ under $\mathcal{F}(\vec{p} \cup \vec{p}_M)$
            place $c$ in $p^a$ such that $pos(c, p^a) = j - 1$
            $j \leftarrow j + 1$
            remove $c$ from $C$
        add $p^a$ to $\vec{p}_M$
    if $\mathcal{N}(\mathcal{F}(\vec{p} \cup \vec{p}_M), p_o) = \{p\}$ then
        return $\vec{p}_M$
return false
```

the algorithm's stages. However, the order of the outcomes in H and $O \setminus H$ according to $\mathcal{F}(\vec{p} \cup \vec{p}_M)$ may change when we update $\vec{p}_M$, which implies that the order in which we place the outcomes from H and $O \setminus H$ in p^a may differ from one vote to another.

THEOREM 4.2. *Algorithm 2 correctly decides the CC-MaNego problem with x-approval rule in polynomial time.*

In order to prove Theorem 4.2 we use the following definitions. Recall that $k = |M|$. Let $s_\ell(c)$ be the score of candidate c in $\mathcal{F}(\vec{p} \cup \vec{p}_M)$ after stage ℓ. Note that $\forall s$ $U^s \neq U^{s-1}$ and $D^s \neq D^{s-1}$, and s does not necessarily equal k.

We begin by proving some Lemmas that are necessary for the proof of Theorem 4.2.

LEMMA 4.3.
(1) *The candidates in U are placed in each stage l, $1 \leq l \leq k$ in the $|U|$ highest positions.*
(2) *The candidates in D are placed in each stage l, $1 \leq l \leq k$ in the $|D|$ lowest positions.*

PROOF. Assume by contradiction that there exists some $h \in H \setminus U$ that was placed in some stage in one of the first $|U|$ places, then there exists some $u \in U$ that was placed below h at this stage. Let $s \geq 0$ such that $u \in U^s$. By definition, $h \in U^{s+1}$ and thus $h \in U$, which is a contradiction to the choice of h. The proof for the set D is similar. □

LEMMA 4.4. *If $\forall u \in U$ in each stage j, u receives 1 point, then $|U| = 1$. Similarly, if $\forall d \in D$ in each stage j, d receives 0 points, then $|D| = 1$.*

PROOF. Let $u_0 = U^0$. Assume to contradiction that there exists some $u \in U$ such that $u \neq u_0$. By definition of the set U, after some stage, u was positioned lower than u_0. But u and u_0 gained in each stage 1 point, and therefore u could not have bean positioned lower than u_0 in any stage. Contradiction to the existence of such a candidate u. The proof for the set D is similar. □

LEMMA 4.5. *For all $u_1, u_2 \in U$, $|s_k(u_1) - s_k(u_2)| \leq 1$. Similarly, for all $d_1, d_2 \in D$, $|s_k(d_1) - s_k(d_2)| \leq 1$.*

PROOF. Recall that $U^0 = \arg\min_{h \in H} pos(h, p_t)$, and for each $s = 1, 2, ...$, $U^s = U^{s-1} \cup \{u : u$ was ranked above some $u' \in U^{s-1}$ in some stage l, $1 \leq l < k$, but u was ranked below some $u' \in U^{s-1}$ in stage $l + 1\}$. In addition, $U = \bigcup_{0 \leq s} U^s$. We prove by induction on the index s, that for all $u_1, u_2 \in U$, $|s_k(u_1) - s_k(u_2)| \leq 1$. In the base case, when $s = 0$, $|U^0| = 1$ and thus the inequality trivially holds. Assume that for $s - 1$, for all $u_1, u_2 \in U^{s-1}$, $|s_l(u_1) - s_l(u_2)| \leq 1$. We show that $u_1, u_2 \in U^s$, $|s_{l+1}(u_1) - s_{l+1}(u_2)| \leq 1$.

There are several possible cases:

(1) $u_1, u_2 \in U^{s-1}$ and $\sigma_{l+1}(u_1) = \sigma_{l+1}(u_2)$. Following the induction assumption, $|s_{l+1}(u_1) - s_{l+1}(u_2)| \leq 1$.
(2) $u_1, u_2 \in U^{s-1}$ but $\sigma_{l+1}(u_1) \neq \sigma_{l+1}(u_2)$ and $s_l(u_2) = s_l(u_1)$. By the definition of x-approval, $|s_{l+1}(u_1) - s_{l+1}(u_2)| = 1$.
(3) $u_1, u_2 \in U^{s-1}$ but $\sigma_{l+1}(u_1) \neq \sigma_{l+1}(u_2)$ and, with out loss of generality, $s_l(u_2) > s_l(u_1)$. By the algorithm construction, $\sigma_{l+1}(u_2) = 0$ and $\sigma_{l+1}(u_1) = 1$ and thus by the induction assumption, $s_{l+1}(u_2) = s_{l+1}(u_1)$.
(4) With out loss of generality, $u_1 \in U^{s-1}$ and $u_2 \notin U^{s-1}$. In addition, $\sigma_{l+1}(u_1) = \sigma_{l+1}(u_2)$. By the definition of U^{s-1}, $s_l(u_2) \geq s_l(u_1)$, and since $\sigma_{l+1}(u_1) = \sigma_{l+1}(u_2)$ then $s_{l+1}(u_2) \geq s_{l+1}(u_1)$. Since $u_2 \in U^s$, then $\exists u \in U^{s-1}$ such the $s_{l+1}(u) \geq s_{l+1}(u_2)$. According to cases 1, 2 or 3, $s_{l+1}(u) - s_{l+1}(u_1) \leq 1$. Combining the inequalities we get that $s_{l+1}(u_2) - s_{l+1}(u_1) \leq 1$.
(5) With out loss of generality, $u_1 \in U^{s-1}$ and $u_2 \notin U^{s-1}$. In addition, $\sigma_{l+1}(u_1) \neq \sigma_{l+1}(u_2)$. By the definition of U^{s-1}, $s_l(u_2) \geq s_l(u_1)$. In addition, by the algorithm construction $\sigma_{l+1}(u_1) = 1$ and $\sigma_{l+1}(u_2) = 0$. There are two possible cases:
    - $s_{l+1}(u_2) < s_{l+1}(u_1)$. That is, $s_l(u_2) = s_l(u_1)$ and thus $s_{l+1}(u_1) - s_{l+1}(u_2) = 1$
    - $s_{l+1}(u_2) \geq s_{l+1}(u_1)$. Since $u_2 \in U^s$, then $\exists u \in U^{s-1}$ such that $s_{l+1}(u) \geq s_{l+1}(u_2)$. According to cases 1, 2 or 3, $s_{l+1}(u) - s_{l+1}(u_1) \leq 1$. Combining the inequalities we get that $s_{l+1}(u_2) - s_{l+1}(u_1) \leq 1$.
(6) $u_1, u_2 \notin U^{s-1}$. Since $u_1, u_2 \in U^s$ then $\exists u'_1, u'_2 \in U^{s-1}$ such the $s_{l+1}(u'_1) \geq s_{l+1}(u_1)$ and $s_{l+1}(u'_2) \geq s_{l+1}(u_2)$. According to cases 4 or 5, $s_{l+1}(u'_1) - s_{l+1}(u_1) \leq 1$ and $s_{l+1}(u'_2) - s_{l+1}(u_2) \leq 1$. Let $u = \arg\max\{u'_1, u'_2\}$. Then, $s_{l+1}(u) - s_{l+1}(u_1) \leq 1$ and $s_{l+1}(u) - s_{l+1}(u_2) \leq 1$, and thus $|s_{l+1}(u_2) - s_{l+1}(u_1)| \leq 1$.

The proof for the set D is similar. □

LEMMA 4.6. *For all $u_1, u_2 \in U$, if $s_k(u_1) = s_k(u_2) + 1$ then u_2 is preferred over u_1 according to the lexicographical tie-breaking rule. Similarly, for all $d_1, d_2 \in D$, if $s_k(d_1) = s_k(d_2) + 1$ then d_2 is preferred over d_1 according to the lexicographical tie-breaking rule.*

PROOF. Recall that $U^0 = \arg\min_{h \in H} pos(h, p_t)$, and for each $s = 1, 2, ...$, $U^s = U^{s-1} \cup \{u : u$ was ranked above some $u' \in U^{s-1}$ in some stage l, $1 \leq l < k$, but u was ranked below some $u' \in U^{s-1}$ in stage $l + 1\}$. In addition, $U = \bigcup_{0 \leq s} U^s$. We prove by induction on the index s, that for all $u_1, u_2 \in U$, if $s_k(u_1) = s_k(u_2) + 1$

then u_2 is preferred over u_1 according to the lexicographical tie-breaking rule. In the base case, when $s = 0$, $|U^0| = 1$ and thus the claim trivially holds. Assume that for $s-1$, for all $u_1, u_2 \in U^{s-1}$, if $s_l(u_1) = s_l(u_2) + 1$ then u_2 is preferred over u_1 according to the lexicographical tie-breaking rule. We show that $\forall u_1, u_2 \in U^s$, if $s_{l+1}(u_1) = s_{l+1}(u_2) + 1$ then u_2 is preferred over u_1 according to the lexicographical tie-breaking rule. There are several possible cases:

(1) $u_1, u_2 \in U^{s-1}$ and $\sigma_{l+1}(u_1) = \sigma_{l+1}(u_2)$. Following the induction assumption, since $s_l(u_1) = s_l(u_2) + 1$ then u_2 is preferred over u_1 according to the lexicographical tie-breaking rule. Now, $s_{l+1}(u_1) = s_{l+1}(u_2) + 1$ and u_2 is preferred over u_1 according to the lexicographical tie-breaking rule.

(2) $u_1, u_2 \in U^{s-1}$ and $\sigma_{l+1}(u_1) \neq \sigma_{l+1}(u_2)$. Since $s_{l+1}(u_1) = s_{l+1}(u_2) + 1$ and $\sigma_{l+1}(u_1) \neq \sigma_{l+1}(u_2)$ then $s_l(u_1) = s_l(u_2)$. Since $s_{l+1}(u_1) = s_{l+1}(u_2)+1$ then $\sigma_{l+1}(u_1) = 1$ and $\sigma_{l+1}(u_2) = 0$. That is, u_2 is preferred over u_1 according to the lexicographical tie-breaking rule (by the algorithm construction).

(3) $u_1 \in U^{s-1}$, $u_2 \notin U^{s-1}$. That is, u_1 was ranked lower than u_2 in stage l. However, $s_{l+1}(u_1) = s_{l+1}(u_2) + 1$, and thus $\sigma_{l+1}(u_2) = 0$ and $\sigma_{l+1}(u_1) = 1$, and so $s_l(u_1) = s_l(u_2)$. Since u_1 was ranked lower than u_2 in stage l then it must be that u_2 is preferred over u_1 according to the lexicographical tie-breaking rule.

(4) $u_1 \notin U^{s-1}$, $u_2 \in U^{s-1}$. Since $u_1 \in U^s$, then $\exists u \in U^{s-1}$ such that u is ranked higher than u_1 in stage $l+1$. Thus, $\sigma_{l+1}(u_1) = 0$ and $\sigma_{l+1}(u) = 1$. Since $u, u_1, u_2 \in U^s$, $s_{l+1}(u_1) = s_{l+1}(u_2) + 1$ and u is ranked higher than u_1 in stage $l+1$, by Lemma 4.5, $s_{l+1}(u_1) = s_{l+1}(u)$. Since u is ranked higher than u_1 in stage $l+1$, u is preferred over u_1 according to the lexicographical tie-breaking rule. There are two possible cases:
   - $s_l(u) = s_l(u_2)$. That is, $\sigma_{l+1}(u_2) = 0$ and $\sigma_{l+1}(u) = 1$. Thus, u_2 is preferred over u according to the lexicographical tie-breaking rule, by the algorithm construction. Therefore, u_2 is preferred over u_1 according to the lexicographical tie-breaking rule.
   - $s_l(u) = s_l(u_2) + 1$. Following the induction assumption, since $s_l(u) = s_l(u_2) + 1$ then u_2 is preferred over u according to the lexicographical tie-breaking rule. Therefore, u_2 is preferred over u_1 according to the lexicographical tie-breaking rule.

(5) $u_1, u_2 \notin U^{s-1}$. Since $u_1, u_2 \in U^s$, by definition of U^s, $\exists u \in U^{s-1}$ such that u is ranked higher than u_1 and u_2 in stage $l+1$, but u was ranked lower than u_1 and u_2 in stage l. Thus, $\sigma_{l+1}(u_1) = \sigma_{l+1}(u_2) = 0$ and $\sigma_{l+1}(u) = 1$. Since $s_{l+1}(u_1) = s_{l+1}(u_2)+1$ and $\sigma_{l+1}(u_1) = \sigma_{l+1}(u_2)$, then $s_l(u_1) = s_l(u_2)+1$. Therefore, $s_l(u) = s_l(u_2)$ and $s_{l+1}(u) = s_{l+1}(u_1)$. That is, u is preferred over u_1 according to the lexicographical tie-breaking order. In addition, since $s_l(u) = s_l(u_2)$ and $u \in U^{s-1}$ and $u_2 \notin U^{s-1}$, then u_2 is preferred over u according to the lexicographic tie-breaking order. Thus, u_2 is preferred over u_1 according to the lexicographical tie-breaking rule.

The proof for the set D is similar. □

LEMMA 4.7. *The set $|U| = 1$ or $|D| = 1$.*

PROOF. Recall that $|H| = i$, $U^0 = \{u_0\}$ where $u_0 = \arg\min_{h \in H} pos(h, p_t)$ and $D^0 = \{d_0\}$ where $d_0 = \arg\max_{d \in \{O \setminus H\}} pos(d, p_t)$. Since we use x-approval, if $x \geq i$ then by Lemma 4.3 all of the candidates of H get a score of 1 in each stage. Therefore, there is no candidate from H that is ranked lower than u_0, and thus $U = U^0$. On the other hand, if $x < i$ then by Lemma 4.3 all of the candidates of $\{O \setminus H\}$ get a score of 0 in each stage. Therefore, there is no candidate from D that is ranked higher than d_0, and thus $D = D^0$. □

LEMMA 4.8. *Let $d* \in D$ such that $\forall d \in D$, $d \neq d*$ and $d* \succ_{p_t^a} d$. Similarly, let $u* \in U$ such that $\forall u \in U$, $u \neq u*$ and $u \succ_{p_t^a} u*$. If $u* \succ_{p_t^a} d*$, and there are k manipulators then there is a manipulation that makes p the SPE result, and Algorithm 2 will find it.*

PROOF. First we show that for all $o \in \{O \setminus (H \cup D)\}$ and $\forall d \in D$, it holds that $d \succ_{p_t^a} o$. Assume by contradiction that there exists $o \in \{O \setminus (H \cup D)\}$ and $d \in D$, such that $o \succ_{p_t^a} d$. Then, by the definition of D, $o \in D$, which is a contradiction to the choice of o. Let $h* \in H$ such that $\forall h \in H$, $h \neq h*$ and $h \succ_{p_t^a} h*$. Now, if $u* \succ_{p_t^a} d*$, then for all $o \in O \setminus H$, $o \prec_{p_t^a} u* = h*$ (otherwise, $h*$ would have been part of U). Therefore, $A^i_{(\mathcal{F}(\vec{p} \cup \vec{p}_M))} = H$, and by definition of H, $A^i_{(p_o)} \cap H = \{p\}$. That is, the algorithm finds a manipulation that makes p the SPE result. □

LEMMA 4.9. *Let $q(U)$ and $q(D)$ be the average score of candidates in U and D after k stages, respectively. That is, $q(U) = \frac{1}{|U|}\sum_{u \in U} s_k(u)$, $q(D) = \frac{1}{|D|}\sum_{d \in D} s_k(d)$. Let $d* \in D$ such that $\forall d \in D$, $d \neq d*$ and $d* \succ_{p_t^a} d$. Similarly, let $u* \in U$ such that $\forall u \in U$, $u \neq u*$ and $u \succ_{p_t^a} u*$. If $d* \succ_{p_t^a} u*$, and there are k manipulators then there is no manipulation that makes p the SPE result, and the algorithm will return false.*

PROOF. Assume that there is a successful manipulation $\vec{p}_m$ with k manipulators. Let i be the corresponding iteration in which RC returns p. Let $p_t^m = \mathcal{F}(\vec{p} \cup \vec{p}_m)$, and let $s_k^m(c)$ be the score of a candidate c in p_t^m. Since Algorithm 2 (as proved in Lemma 4.3) places all the outcomes $u \in U$ at the $|U|$ highest positions and the outcomes $d \in D$ at the $|D|$ lowest positions, then:

$$q(U) = \frac{1}{|U|}(\sum_{u \in U} s_0(u) + k \cdot \min\{x, |U|\}) \geq \qquad (1)$$

$$\frac{1}{|U|}(\sum_{u \in U} s_k^m(u)) =: q^m(U)$$

$$q(D) = \frac{1}{|D|}(\sum_{d \in D} s_0(d) + k \cdot (\max\{|D|, m-x\} - (m-x))) \leq \qquad (2)$$

$$\frac{1}{|D|}\sum_{d \in D} s_k^m(d) =: q^m(D)$$

Since $d* \succ_{p_t^a} u*$, then ,by Lemma 4.5, $\lceil q(D) \rceil \geq \lfloor q(U) \rfloor$. Combining the equations we get $\lceil q^m(D) \rceil \geq \lfloor q^m(U) \rfloor$. Since $q^m(D)$ and $q^m(U)$ are averages then there is at least one $u \in U$ and one $d \in D$, such that $s_k^m(d) = \lceil q^m(D) \rceil$ and $\lfloor q^m(U) \rfloor = s_k^m(u)$. Therefore, $s_k^m(d) \geq s_k^m(u)$. If $s_k^m(d) > s_k^m(u)$, then $d \succ_{p_t^m} u$. Otherwise, $s_k^m(d) = s_k^m(u)$, and we show that $\exists d' \in D$ such that $d' \succ_{p_t^m} u$. According to Lemma 4.7 either $|U| = 1$ or $|D| = 1$, so assume that $|U| = 1$ and thus $u = u*$. There are three possible cases:

(1) The algorithm assigns a score of 0 to all of the candidates in D. In this case, according to Lemma 4.4, $|D| = 1$. That is, $d = d*$. Since $s_k^m(d) = s_k^m(u)$ then $s_k(d) = s_k(u)$ (according to Equations 1,2). Since $d \succ_{p_t^a} u$, then $d \succ_{p_t^m} u$.

(2) The algorithm assigns a score of 0 to some of the candidates in D, and $\forall d' \in D\ s_k(d') \geq s_k(u)$. If $\exists d'' \in D$ such that $s_k(d'') > s_k(u)$ then $d'' \succ_{p_t^m} u$. Otherwise, $s_k(d) = s_k(d*)$. Since $s_k^m(d*) = s_k^m(u)$ then $s_k(d*) = s_k(u)$ (according to Equations 1,2). Since $d* \succ_{p_t^a} u$, then $d* \succ_{p_t^m} u$.

(3) The algorithm assigns a score of 0 to some of the candidates in D, but $\exists d' \in D\ s_k(d') < s_k(u)$. If $\exists d'' \in D$ such that $s_k^m(d'') > s_k^m(u)$, then $d'' \succ_{p_t^m} u$. Otherwise, let y be the number of candidates from D that have the score of $s_k(u)$ according to p_t^a. By Equations 1,2, there are at least y candidates $d'' \in D$ such that $s_k^m(d'') = s_k^m(u)$. By Lemma 4.6, there is at least one candidate that, $d'' \in D$ such that $s_k^m(d'') = s_k^m(u)$ and $d'' \succ_{p_t^m} u$.

Since $\min_{u \in U} s_k(u) = \min_{h \in H} s_k(h)$ and $|H| = i$ then there exists $d \in D$ such that $d \in A^i_{(p_t^m)}$. Let $D' = \{d \in \{O \setminus H\} : d \in A^i_{(p_t^m)}\}$ and let $H' = \{h \in H : h \notin A^i_{(p_t^m)}\}$, where $D' = \{d_1, \ldots, d_w\}$ and $H' = \{h_1, \ldots, h_w\}$. Now, we switch between the candidates from D' and the candidates from H' in $\vec{p}_m$. That is, given a preference order $p^a \in \vec{p}_m$ let $p^{a'} \leftarrow p^a$ and then for all $1 \leq j \leq w$, h_j is placed in $p^{a'}$ in $pos(d_j, p^a)$ and d_j is placed in $p^{a'}$ in $pos(h_j, p^a)$, for $d_j \in D'$ and $h_j \in H'$. Let $\vec{p}_{m'}$ be the manipulation where for each $p^a \in \vec{p}_m, p^{a'} \in \vec{p}_{m'}$, let $p_t^{m'} = \mathcal{F}(\vec{p} \cup \vec{p}_{m'})$, and let $q^{m'}(U)$ and $q^{m'}(D)$ be the average scores of candidates from U and D, respectively, in $p_t^{m'}$. Clearly, Equations 1- 2 hold for $q^{m'}(U)$ and $q^{m'}(D)$ as well. That is, $\lceil q^{m'}(D) \rceil \geq \lfloor q^{m'}(U) \rfloor$ and so, $\exists d \in D$ and $u \in U$ such that $d \succ_{p_t^{m'}} u$. On the other hand, for all $h \in H \setminus \{p\}$ and $d \in D'$, $pos(h, p_t) > pos(d, p_t)$, by the definition of H, and $p \in A^i_{(p_t^{m'})}$. Therefore, $A^i_{(p_t^{m'})} = H$, and since $U \subseteq H$ then $\lfloor q^{m'}(U) \rfloor \geq \lceil q^{m'}(D) \rceil$, and so, $\forall d \in D$ and $\forall u \in U$, $u \succ_{p_t^{m'}} d$, a contradiction. Therefore, there is no manipulation that makes p the SPE result. The proof for the case where $|D| = 1$ is similar. Finally, if Algorithm 2 returns $\vec{p}_M$ then it is a successful manipulation. Thus, if there is no manipulation the algorithm will return false. □

PROOF OF THEOREM 4.2. Clearly, Algorithm 2 runs in polynomial time. According to Lemma 4.8, if $u* \succ_{p_t^a} d*$, then there is a manipulation that makes p the SPE result, and Algorithm 2 will find it. On the other hand, according to Lemma 4.9 if $d* \succ_{p_t^a} u*$ then there is no manipulation that makes p the SPE result (with k manipulators), and the algorithm will return false. Thus, Algorithm 2 correctly decides the CC-MaNego problem with x-approval. □

Unlike with the family of x-approval rules, CC-MaNego is computationally hard with Borda. The reduction is from the Permutation Sum problem [27]. Due to space constraints, the full proofs of this theorem and all subsequent theorems are provided in the appendix.

THEOREM 4.10. *CC-MaNego is NP-Complete with Borda.*

Even though CC-MaNego with Borda is *NP*-complete, it is still desirable to have an efficient heuristic algorithm that finds a successful coalitional manipulation. We now show that Algorithm 2 is such a heuristic, and show its theoretical guarantee. Specifically, the algorithm is guaranteed to find a coalitional manipulation in many instances, and we characterize the instances in which it may fail. Formally,

THEOREM 4.11. *Given an instance of CC-MaNego with Borda,*

(1) *If there is no preference profile making p the negotiation result, then Algorithm 2 will return false.*
(2) *If a preference profile making p the negotiation result exists, then for the same instance with one additional manipulator, Algorithm 2 will return a preference profile that makes p the negotiation result.*

That is, Algorithm 2 will succeed on any given instance such that the same instance but with one less manipulator is manipulable. Thus, it can be viewed as a 1-additive approximation algorithm (this approximate sense was introduced by [28] when analyzing Borda as a social choice function (SCF)).

PROOF (SKETCH). Interestingly, this proof is in the same vein as the proof of Theorem 4.2, and we again use the sets U and D. However, the proof here is more involved. Let $s_\ell(c)$ be the score of candidate c in $\mathcal{F}(\vec{p} \cup \vec{p}_M)$ after stage ℓ. We first show that the sets of scores $\{s_{k-1}(u) : u \in U\}$ and $\{s_{k-1}(d) : d \in D\}$ are 1-dense, which is the following:

Definition 4.12 (due to [28]). A finite non-empty set of integers B is called 1-dense if when sorting the set in a non-increasing order $b_1 \geq b_2 \geq \cdots \geq b_i$ (such that $\{b1, \ldots, b_i\} = B$), $\forall j, 1 \leq j \leq i-1$, $b_{j+1} \geq b_j - 1$ holds.

Let $q(U)$ and $q(D)$ be the average score of candidates in U and D, respectively, after $k-1$ stages. We show that $q(U) \leq \min_{u \in U}\{s_k(u)\} - m + |U|$, and similarly, $\max_{d \in D}\{s_k(d)\} \leq q(D) + |D| - 1$. That is, we bound the distance between the minimal score in U (the maximal score in D) after stage k and the average score in U (D) after stage $k-1$. Now, suppose that there is a successful manipulation for Borda with $k-1$ manipulators. We prove that the average score of the candidates in U according to the manipulation is not higher than $q(U)$, and the average score of the candidates in D according to the manipulation in not lower than $q(D)$. Therefore, it is possible to show that by adding one additional manipulator the algorithm will find a successful manipulation. □

5 DESTRUCTIVE MANIPULATION

In this section we study the *destructive* manipulation problem, where the goal of the manipulation is to prevent an outcome from being the SPE result. We begin with the destructive variant of manipulation by a single voter.

Definition 5.1 (D-MaNego). We are given a social welfare function $\mathcal{F}$, a preference profile $\vec{p}$ of the voters of negotiating team t, the preference of the other party p_o, a specific manipulator v', and a disliked candidate $e \in O$. We are asked whether a preference order $p_{v'}$ exists for the manipulator v' such that $e \notin \mathcal{N}(\mathcal{F}(\vec{p} \cup p_{v'}), p_o)$.

Recall that C-MaNego is in P for any scoring rule, but this does not immediately imply that D-MaNego is also in P. Indeed, it is possible to run Algorithm 1 for each candidate $o \neq e$. However, since Algorithm 1 returns a manipulation only when $\mathcal{N}(\mathcal{F}(\vec{p} \cup p_{v'}), p_o) = \{p\}$, it does not find a solution where $\mathcal{N}(\mathcal{F}(\vec{p} \cup p_{v'}), p_o) = \{o, o'\}$, where both $o, o' \neq e$, which is a possible solution for D-MaNego. Nevertheless, we can use a slightly modified version of Algorithm 1 for D-MaNego.

THEOREM 5.2. *D-MaNego with any positional scoring rule can be decided in polynomial time.*

We now continue with the destructive coalitional manipulation problem, where several manipulators might decide to collude and coordinate their votes in such a way that an agreed candidate e will not be the SPE result. The problem is defined as follows:

Definition 5.3 (DC-MaNego). Given a social welfare function $\mathcal{F}$, a preference profile $\vec{p}$ of the voters of negotiating team t, the preference of the other party p_o, a number of manipulators k, and a disliked candidate $e \in O$, we check whether a preference profile $\vec{p}_M$ exists for the manipulators such that $e \notin \mathcal{N}(\mathcal{F}(\vec{p} \cup \vec{p}_M), p_o)$.

Similar to C-MaNego, we show that a slightly modified version of Algorithm 2 decides DC-MaNego with any x-approval rule.

THEOREM 5.4. *DC-MaNego with any x-approval rule can be decided in polynomial time.*

Unfortunately, DC-MaNego with Borda is computationally hard. Note that this result is surprising, since the destructive coalitional manipulation problem when using Borda as an SCF is in P [9].

THEOREM 5.5. *DC-MaNego with Borda is NP-Complete.*

Finally, similar to CC-MaNego, we show that the modified Algorithm 2 is an efficient heuristic algorithm that finds a successful destructive manipulation, and we guarantee the same approximation. That is, the algorithm succeeds in finding a destructive manipulation for any given instance such that success for the same instance with one less manipulator is possible.

THEOREM 5.6. *There is a* 1*-additive approximation algorithm for DC-MaNego with Borda.*

6 RELATED WORK

The computational analysis of voting manipulation was initially performed by [3], and [2], who investigated constructive manipulation by a single voter. Following these pioneer works, many researchers have investigated the computational complexity of manipulation, and studied different types of manipulation with different voting rules in varied settings. We refer the reader to the survey provided by [18], and more recent survey by [10]. All of the works that are surveyed in these papers analyze the manipulation of voting rules as *social choice functions*, that is, the voting rules are used to output one winning candidate (or a set of tied winning candidates). In our work we investigate manipulation of a resolute SWF, i.e., it outputs a complete preference order of the candidates.

There are very few papers that investigate the manipulation of SWFs. This is possibly since the opportunities for manipulation are not well-defined without additional assumptions. That is, since the output of a SWF is an order, and voters do not report their preferences over all possible orderings, some assumptions have to be made on how the voters compare possible orders. Indeed, the first work that directly deals with the manipulation of SWF was by [5], who assumed that a voter prefers one order over another if the former is closer to her own preferences than the latter according to the Kemeny distance, and mainly presented impossibility results. Bossert and Sprumont [4] assumed that a voter prefers one order over another if the former is strictly between the latter and the voter's own preferences. Built on this definition their work studies three classes of SWF that are not prone to manipulation (i.e., strategy-proof). Dogan and Lainé [13] characterized the conditions to be imposed on SWFs so that if we extend the preferences of the voters to preferences over orders in specific ways the SWFs will not be prone to manipulation. Our work also investigates the manipulation of SWF, but we analyze the SWF in the specific context of a negotiation. Therefore, unlike all of the above works, the preferences of the manipulators are well-defined and no additional assumptions are needed.

Our work is also connected to committee elections or multi-winner elections, where manipulation of scoring rules has been considered [22, 24]. However, in committee election we are given the size of the committee as an input. In our setting the output of the voting rule (i.e., the ranking) essentially determines the point in which RC terminates. Using the model of committee election in our setting we can say that the ranking determines the size of the committee. That is, each possible manipulation determines not only the position of each candidate but also the size of the committee.

The work that is closest to ours is the paper by [25], which involves the use of voting rules for the decision process of a negotiating team, i.e., the same basic scenario that we consider. The paper presents several strategies they developed, which use some specific, tailored-made, voting rules, and experimentally analyzes them in different environments. Our work analyzes voting in the context of a negotiation from a theoretical perspective. We formally define the general problem, show polynomial time algorithms for some cases, and provide hardness results and approximations for others.

Finally, we note that in our setting there is a SWF, which outputs an order over the candidates, and this order is used as an input for the negotiation process. In Section 2 we note that there is a connection between the sub-game perfect equilibrium of the negotiation and the Bucklin voting rule. Therefore, our setting is also related to a multi-stage voting. Several variants of multi-stage voting have been considered [8, 12, 15, 23]. All of these works did not consider the case of SWF in the first round, as we do. More importantly, in all of these works the set of voters remains the same throughout the application of the voting rules. In our case the set of the voters in the first stage is different from the set in the second stage. In the first stage the voters are the agents in the negotiating team, and they use a scoring rule as a SWF. In the second stage there are only two voters, which are the negotiating parties, and they use an equivalent of Bucklin on their full preference orders.

7 CONCLUSION

In this paper we analyze the problem of strategic voting in the context of negotiating teams. Specifically, a scoring rule is used as a SWF, which outputs an order over the candidates that is used as an input in the negotiation process with the VAOV protocol. We show that the single manipulation problem is in P with this two stage procedure, and the coalitional manipulation is also in P for any x-approval rule. The problem of coalitional manipulation becomes hard when using Borda, but we provide an algorithm that can be viewed as a 1-additive approximation for this case. Interestingly, our complexity results hold both for constructive and destructive manipulations, unlike the problems of manipulation when using Borda as an SCF. Note also that our algorithms are quite general. Algorithm 1 provides a solution with any scoring rule. Algorithm 2 solves the coalitional manipulation problem with any x-approval and it is also an efficient approximation with Borda.

ACKNOWLEDGMENTS

This research was supported in part by the Ministry of Science, Technology & Space, Israel.

REFERENCES

[1] Nejat Anbarci. 1993. Noncooperative foundations of the area monotonic solution. *The Quarterly Journal of Economics* 108, 1 (1993), 245–258.

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

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

[4] Walter Bossert and Yves Sprumont. 2014. Strategy-proof preference aggregation: Possibilities and characterizations. *Games and Economic Behavior* 85 (2014), 109–126.

[5] Walter Bossert and Ton Storcken. 1992. Strategy-proofness of social welfare functions: the use of the Kemeny distance between preference orderings. *Social Choice and Welfare* 9, 4 (1992), 345–360.

[6] Steven J Brams. 2003. *Negotiation Games: Applying Game Theory to Bargaining and Arbitration*. Vol. 2. Psychology Press.

[7] Susan Brodt and Leigh Thompson. 2001. Negotiating teams: A levels of analysis approach. *Group Dynamics: Theory, Research, and Practice* 5, 3 (2001), 208–219.

[8] Vincent Conitzer and Tuomas Sandholm. 2003. Universal Voting Protocol Tweaks to Make Manipulation Hard. In *Proceedings of IJCAI*. 781–788.

[9] Vincent Conitzer, Tuomas Sandholm, and Jérôme Lang. 2007. When are elections with few candidates hard to manipulate? *Journal of the ACM (JACM)* 54, 3 (2007), 14.

[10] Vincent Conitzer and Toby Walsh. 2016. Barriers to Manipulation in Voting. In *Handbook of Computational Social Choice*, Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia (Eds.). 127–145.

[11] Jessica Davies, George Katsirelos, Nina Narodytska, and Toby Walsh. 2011. Complexity of and algorithms for Borda manipulation. In *Proceedings of AAAI*. 657–662.

[12] Jessica Davies, Nina Narodytska, and Toby Walsh. 2012. Eliminating the weakest link: Making manipulation intractable?. In *Proceedings of AAAI*. 1333–1339.

[13] Onur Dogan and Jean Lainé. 2016. Strategic Manipulation of Social Welfare Functions via Strict Preference Extensions. In *The 13th Meeting of the Society for Social Choice and Welfare*. 199.

[14] Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. 2001. Rank aggregation methods for the web. In *Proceedings of WWW*. 613–622.

[15] Edith Elkind and Helger Lipmaa. 2005. Hybrid voting protocols and hardness of manipulation. In *ISAAC*. 206–215.

[16] Eithan Ephrati, Jeffrey S Rosenschein, et al. 1993. Multi-agent planning as a dynamic search for social consensus. In *Proceedings of IJCAI*. 423–429.

[17] Sefi Erlich, Noam Hazon, and Sarit Kraus. 2018. Negotiation strategies for agents with ordinal preferences. In *Proceedings of IJCAI*. 210–218.

[18] Piotr Faliszewski and Ariel D Procaccia. 2010. AI's war on manipulation: Are we winning? *AI Magazine* 31, 4 (2010), 53–64.

[19] Shaheen Fatima, Sarit Kraus, and Michael Wooldridge. 2014. *Principles of automated negotiation*. Cambridge University Press.

[20] Allan Gibbard. 1973. Manipulation of voting schemes: a general result. *Econometrica* 41, 4 (1973), 587–601.

[21] Özgür Kıbrıs and Murat R Sertel. 2007. Bargaining over a finite set of alternatives. *Social Choice and Welfare* 28, 3 (2007), 421–437.

[22] Reshef Meir, Ariel D Procaccia, Jeffrey S Rosenschein, and Aviv Zohar. 2008. Complexity of strategic behavior in multi-winner elections. *Journal of Artificial Intelligence Research* 33 (2008), 149–178.

[23] Nina Narodytska and Toby Walsh. 2013. Manipulating two stage voting rules. In *Proceedings of AAMAS*. 423–430.

[24] Svetlana Obraztsova, Yair Zick, and Edith Elkind. 2013. On manipulation in multi-winner elections based on scoring rules. In *Proceedings of AAMAS*. 359–366.

[25] Víctor Sánchez-Anguix, Vicente Botti, Vicente Julián, and Ana García-Fornes. 2011. Analyzing intra-team strategies for agent-based negotiation teams. In *Proceedings of AAMAS*. 929–936.

[26] Mark Allen Satterthwaite. 1975. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of economic theory* 10, 2 (1975), 187–217.

[27] Wenci Yu, Han Hoogeveen, and Jan Karel Lenstra. 2004. Minimizing makespan in a two-machine flow shop with delays and unit-time operations is NP-hard. *Journal of Scheduling* 7, 5 (2004), 333–348.

[28] Michael Zuckerman, Ariel D Procaccia, and Jeffrey S Rosenschein. 2009. Algorithms for the coalitional manipulation problem. *Artificial Intelligence* 173, 2 (2009), 392–412.

8 APPENDIX

8.1 Proof of Theorem 4.10

Let $p_t^M = \mathcal{F}(\vec{p} \cup \vec{p}_M)$. Clearly, the CC-MaNego problem is in *NP*. The proof of the *NP*- hardness is by a reduction from a specialized permutation problem that is *NP*- complete [27].

Definition 8.1 (Permutation Sum). Given n integers $X_1 \leq \ldots \leq X_n$ where $\sum_{i=1}^{n} X_i = n(n+1)$, do two permutations σ and π of 1 to n exist such that $\sigma(i) + \pi(i) = X_i$ for all $1 \leq i \leq n$? [11]

Given an instance of the Permutation Sum we build an instance of the CC-MaNego problem as follows. There are $2n+4$ outcomes: $x_1, \ldots, x_n$, which correspond to the integers $X_1, \ldots, X_n$, $y_1, \ldots, y_{n+1}$ and three outcomes a, b and c. By Lemma 1 from [11], we can construct an election in which the non-manipulators cast votes such that:

$$p_t = (y_1, \ldots, y_{n+1}, b, x_1, \ldots, x_n, a, c),$$

and the corresponding scores are:

$$(4n+7+C, \ldots, 4n+7+C, 4n+6+C,$$
$$4n+6+C-X_1, \ldots, 4n+6+C-X_n, C, z),$$

where C is a constant and $z \leq C$. The preference order of p_o is as follows:

$$p_o = (x_1, \ldots, x_n, b, a, c, y_1, \ldots, y_{n+1})$$

We show that two manipulators can make the outcome a the SPE result iff the Permutation Sum problem has a solution.

($\Leftarrow$) Suppose we have two permutations σ and π of 1 to n such that $\sigma(i) + \pi(i) = X_i$. Let σ^{-1} be the inverse function of σ, i.e., $i = \sigma^{-1}(x)$ if $x = \sigma(i)$. We define $\pi^{-1}(x)$ similarly. We construct the following two manipulative votes:

$$(a, y_1, \ldots, y_{n+1}, c, x_{\sigma^{-1}(n)}, \ldots, x_{\sigma^{-1}(1)}, b)$$
$$(a, y_1, \ldots, y_{n+1}, c, x_{\pi^{-1}(n)}, \ldots, x_{\pi^{-1}(1)}, b)$$

Since the permutation sum satisfies $\sigma(i) + \pi(i) = X_i$ and $z \leq C$, the preference profile $p_t^M = \mathcal{F}(\vec{p} \cup \vec{p}_M)$ is:

$$(y_1, y_2 \ldots, y_{n+1}, a, b, x_1, \ldots, x_n, c),$$

since the corresponding scores are:

$$(4n+7+C+2(2n+2), 4n+7+C+2(2n+1), \ldots,$$
$$4n+7+C+2(n+2), 4n+6+C, 4n+6+C,$$
$$4n+6+C-X_1+X_1, \ldots, 4n+6+C-X_n+X_n, 2(n+1)+z).$$

Therefore, $\mathcal{N}(p_t^M, p_o) = \{a\}$.

($\Rightarrow$) Assume we have a successful manipulation. Such manipulation must ensure that all of the candidates $x_1, \ldots, x_n$, and b are not placed in the $n+2$ highest positions in p_t^M, but $pos(a, p_t^M) \geq n+2$. That is, to ensure that outcome a is ranked higher than outcome b, both manipulators have to place a in the highest position in their preferences, and b in the lowest position in their preferences. Thus, the score of outcome a in p_t^M will be $4n+6+C$. Let $\sigma(i)$ be a function that determines the score where the first manipulator assigned to outcome x_i. $\pi(i)$ is defined similarly for the second manipulator. Since the manipulation is successful, for every i, $1 \leq i \leq n$,

$$4n+6+C-X_i+\sigma(i)+\pi(i) \leq 4n+6+C,$$

and thus,

$$\sigma(i)+\pi(i) \leq X_i.$$

Since $\sum_{i=1}^{n} X_i = n(n+1)$,

$$\sum_{i=1}^{n} \sigma(i) + \pi(i) \leq n(n+1).$$

On the other hand, since b is placed in the lowest position by both manipulators,

$$\sum_{i=1}^{n} \sigma(i) \geq \frac{n(n+1)}{2}$$

and

$$\sum_{i=1}^{n} \pi(i) \geq \frac{n(n+1)}{2}.$$

Therefore, $\sum_{i=1}^{n} \sigma(i) + \pi(i) = n(n+1)$, and $\sum_{i=1}^{n} \sigma(i) = \sum_{i=1}^{n} \pi(i) = \frac{n(n+1)}{2}$. That is, σ and π are permutations of 1 to n. Moreover, since there is no slack in the inequalities,

$$\sigma(i) + \pi(i) = X_i.$$

That is, there is a solution to the permutation sum problem.

8.2 proof of Theorem 4.11

We begin by proving some Lemmas that are necessary for the proof of Theorem 4.11.

LEMMA 8.2. *If* $\max_{d \in D}\{s_k(d)\} < \min_{u \in U}\{s_k(u)\}$, *and there are k manipulators then there is a manipulation that makes p the SPE result, and Algorithm 2 will find it.*

PROOF. First we show that for all $o \in \{O \setminus (H \cup D)\}$, it holds that $s_k(o) \leq min_{d \in D}\{s_k(d)\}$. Assume by contradiction that there exists $o \in \{O \setminus (H \cup D)\}$ and $d \in D$, such that $s_k(o) > s_k(d)$. Then, by the definition of D, $o \in D$, which is a contradiction to the choice of o. Now, if $\max_{d \in D}\{s_k(d)\} < \min_{u \in U}\{s_k(u)\}$, then for all $o \in O \setminus H$, $s_k(o) < \min_{u \in U}\{s_k(u)\} = \min_{h \in H}\{s_k(h)\}$ (otherwise, such $h \in H$ would have been part of U). Therefore, $A^i_{(\mathcal{F}(\vec{p} \cup \vec{p}_M))} = H$, and by definition of H, $A^i_{(p_o)} \cap H = p$. That is, the algorithm finds a manipulation that makes p the SPE result. □

LEMMA 8.3. *Let* $q(U)$ *and* $q(D)$ *be the average score of candidates in U and D after* $k-1$ *stages, respectively. That is,* $q(U) = \frac{1}{|U|}\sum_{u \in U} s_{k-1}(u)$, $q(D) = \frac{1}{|D|}\sum_{d \in D} s_{k-1}(d)$. *If* $q(U) < q(D)$, *and there are* $k-1$ *manipulators then there is no manipulation that makes p the SPE result, and the algorithm will return false.*

PROOF. Assume that there is a successful manipulation $\vec{p}_m$ with $k-1$ manipulators. Let i be the corresponding iteration in which *RC* returns p. Let $p_t^m = \mathcal{F}(\vec{p} \cup \vec{p}_m)$, and let $s_{k-1}^m(c)$ be the Borda score of a candidate c in p_t^m. Since Algorithm 2 (as proved in Lemma 4.3) places all the outcomes $u \in U$ at the $|U|$ highest positions and the outcomes $d \in D$ at the $|D|$ lowest positions, then:

$$q(U) = \frac{1}{|U|}(\sum_{u \in U} s_0(u) + \sum_{j=1}^{k-1}\sum_{i=0}^{|U|-1} m - |U| + i) \geq \quad (3)$$

$$\frac{1}{|U|}(\sum_{u \in U} s_{k-1}^m(u)) =: q^m(U)$$

$$q(D) = \frac{1}{|D|}(\sum_{d \in D} s_0(d) + \sum_{j=1}^{k-1}\sum_{i=0}^{|D|-1} i) \leq \quad (4)$$

$$\frac{1}{|D|}\sum_{d\in D} s^m_{k-1}(d) =: q^m(D)$$

Combining the equations we get $q^m(D) > q^m(U)$. Since $q^m(D)$ and $q^m(U)$ are averages then there is at least one $u \in U$ and one $d \in D$, such that $s^m_{k-1}(d) \geq q^m(D)$ and $q^m(U) \geq s^m_{k-1}(u)$. Therefore, $s^m_{k-1}(d) > s^m_{k-1}(u)$. Since $\min_{u\in U} s_{k-1}(u) = \min_{h\in H} s_{k-1}(h)$ and $|H| = i$ then there exists $d \in D$ such that $d \in A^i_{(p^m_t)}$. Let $D' = \{d \in \{O \setminus H\} : d \in A^i_{(p^m_t)}\}$ and let $H' = \{h \in H : h \notin A^i_{(p^m_t)}\}$, where $D' = \{d_1, \ldots, d_w\}$ and $H' = \{h_1, \ldots, h_w\}$. Now, we switch between the candidates from D' and the candidates from H' in $\vec{p}_m$. That is, given a preference order $p^a \in \vec{p}_m$ let $p^{a\prime} \leftarrow p^a$ and then for all $1 \leq j \leq w$, h_j is placed in $p^{a\prime}$ in $pos(d_j, p^a)$ and d_j is placed in $p^{a\prime}$ in $pos(h_j, p^a)$, for $d_j \in D'$ and $h_j \in H'$. Let $\vec{p}_{m'}$ be the manipulation where for each $p^a \in \vec{p}_m, p^{a\prime} \in \vec{p}_{m'}$, let $p^{m\prime}_t = \mathcal{F}(\vec{p} \cup \vec{p}_{m'})$, and let $q^{m\prime}(U)$ and $q^{m\prime}(D)$ be the average scores of candidates from U and D, respectively, in $p^{m\prime}_t$. Clearly, Equations 3- 4 hold for $q^{m\prime}(U)$ and $q^{m\prime}(D)$ as well. That is, $q^{m\prime}(D) > q^{m\prime}(U)$. On the other hand, for all $h \in H \setminus \{p\}$ and $d \in D'$, $pos(h, p_t) > pos(d, p_t)$, by the definition of H, and $p \in A^i_{(p^{m\prime}_t)}$. Therefore, $A^i_{(p^{m\prime}_t)} = H$, and since $U \subseteq H$ then $q^{m\prime}(U) \geq q^{m\prime}(D)$, a contradiction. Therefore, there is no manipulation that makes p the SPE result. Clearly, if Algorithm 2 returns $\vec{p}_M$ then it is a successful manipulation. Thus, if there is no manipulation the algorithm will return false. □

LEMMA 8.4. *$q(U) \leq \min_{u\in U}\{s_k(u)\} - m + |U|$, and similarly, $\max_{d\in D}\{s_k(d)\} \leq q(D) + |D| - 1$.*

PROOF. We reuse definition 4.12 and the following lemma for the proof.

LEMMA 8.5. *Let U, D be as before. Then the sets $\{s_{k-1}(u) : u \in U\}$ and $\{s_{k-1}(d) : d \in D\}$ are 1-dense.*

PROOF. Zuckerman et al. [28] define a set of candidates G_W, and show that the scores of the candidates in G_W are 1-dense (Lemma 3.12 in [28]). Even though our definition of the set D is slightly different, the set of scores $\{s_{k-1}(d) : d \in D\}$ is essentially identical to the set of scores of the candidates in G_W. Thus, it is 1-dense. The proof for the set $\{s_{k-1}(u) : u \in U\}$ is similar. □

We can now prove the lemma.

Sort the members of U by their scores after stage $k-1$ in a decreasing order, i.e., $U = \{u_1, \ldots, u_{|U|}\}$ such that for all $1 \leq t \leq |U| - 1$, $s_{k-1}(u_{t+1}) \leq s_{k-1}(u_t)$. Thus, $u_1 = \arg\max_{u\in U}\{s_{k-1}(u)\}$. Denote for $1 \leq t \leq |U|$, $g_t = s_{k-1}(u_t) + m - |U| + t - 1$, and let $G = \{g_1, \ldots, g_{|U|}\}$. Note that according to Algorithm 2, for any t, $s_k(u_t) = g_t$. Now, since the set $\{s_{k-1}(u) : u \in U\}$ is 1-dense (according to Lemma 8.5) then for any $1 \leq t \leq |U| - 1$, $g_t \leq g_{t+1}$. Thus, for any $t > 1$, $g_1 \leq g_t$. That is, $g_1 = \min_{u\in U}\{s_k(u)\} = s_{k-1}(u_1) + m - |U| = \max_{u\in U}\{s_{k-1}(u)\} + m - |U|$. Clearly, $q(U) \leq \max_{u\in U}\{s_{k-1}(u)\}$ since $q(U)$ is an average score, which is always less than or equal to the maximum score, and thus $q(U) \leq \min_{u\in U}\{s_k(u)\} - m + |U|$.

Similarly, sort the members of D by their scores after stage $k-1$ in an increasing order, i.e., $D = \{d_1, \ldots, d_{|D|}\}$ such that for all $1 \leq t \leq |D| - 1$, $s_{k-1}(d_t) \leq s_{k-1}(d_{t+1})$. Thus, $d_1 = \arg\min_{d\in D}\{s_{k-1}(d)\}$. Denote for $1 \leq t \leq |D|$, $g_t = s_{k-1}(d_t) + |D| - t$, and let $G = \{g_1, \ldots, g_{|D|}\}$. Note that according to Algorithm 2, for any t, $s_k(d_t) = g_t$. Now, since the set $\{s_{k-1}(d) : d \in D\}$ is 1-dense (according to Lemma 8.5) then for any $1 \leq t \leq |D| - 1$, $g_t \geq g_{t+1}$. Thus, for any $t > 1$, $g_1 \geq g_t$. That is, $g_1 = \max_{d\in D}\{s_k(d)\} = s_{k-1}(d_1) + |D| - 1 = \min_{d\in D}\{s_{k-1}(d)\} + |D| - 1$. Clearly, $q(D) \geq \min_{d\in D}\{s_{k-1}(d)\}$, and thus $\max_{d\in D}\{s_k(d)\} \leq q(D) + |D| - 1$. □

Now we can prove our main theorem.

PROOF OF THEOREM 4.11. Clearly, if the algorithm returns a preference profile $\vec{p}_M$, then it is a successful manipulation that will make p the SPE result. Suppose that a preference profile exists that makes p the SPE result with $k-1$ manipulators. By Lemma 8.4, $\max_{d\in D}\{s_k(d)\} \leq q(D) + |D| - 1$. Since for the given instance a successful manipulation exists, then by Lemma 8.3, $q(D) + |D| - 1 \leq q(U) + |D| - 1$. By Lemma 8.4, $q(U) + |D| - 1 \leq \min_{u\in U}\{s_k(u)\} - m + |U| + |D| - 1$. Since $|U| + |D| \leq m$, $\min_{u\in U}\{s_k(u)\} - m + |U| + |D| - 1 < \min_{u\in U}\{s_k(u)\}$. Overall, $\max_{d\in D}\{s_k(d)\} < \min_{u\in U}\{s_k(u)\}$, and by Lemma 8.2 the algorithm will find a preference profile that will make p the negotiation result with k manipulators. □

8.3 proof of Theorem 5.2

We use Algorithm 1 with the following changes. We change lines 1-2 to check whether e cannot be the SPE result. That is, if $pos(e, p_o) < \lfloor m/2 \rfloor$ the algorithm returns true, since every preference order $p_{o'}$ is a successful manipulation. In addition, we define H as follows. Given i, let $p^* \neq e$ be the most preferred outcome in p_t that belongs to $A^i_{(p_o)}$. The set H is composed of p^* and the other $i - 1$ most preferred outcomes in p_t that are not e. This definition of H is to ensure that there will be at least one outcome from $A^i_{(p_o)}$ in the i highest positions in p^a_t while e will not be in the i highest positions in p^a_t. Finally, we place the remaining outcomes so that they will not make e the negotiation result. Therefore, we change the condition in line 10 to check whether $e \notin \mathcal{N}(\mathcal{F}(\vec{p} \cup p^a), p_o)$. Following these changes the proof of correctness is similar to the proof of Theorem 3.2. In essence, whenever the proof of C-MaNego shows that $p \in A^i_{(p^a_t)}$ we can show in the setting of D-MaNego that $e \notin A^i_{(p^a_t)}$ but that there is another outcome $o \in A^i_{(p^a_t)}$ and $o \in A^i_{(p_o)}$.

8.4 proof of Theorem 5.4

We use Algorithm 2, and change it in the same way that we change Algorithm 1 in the proof of Theorem 5.2. Specifically, in lines 1-2 we return true if $pos(e, p_o) < \lfloor m/2 \rfloor$, the set H is composed of p^* and the other $i - 1$ most preferred outcomes in p_t that are not e, and in line 21 we check if $e \notin \mathcal{N}(\mathcal{F}(\vec{p} \cup \vec{p}_M), p_o)$. Following these changes the proof of correctness is similar to the proof of Theorem 4.2. Specifically, Lemmas 4.3, 4.4, 4.5, 4.6 and 4.7 still hold in the DC-MaNego setting. The proofs of Lemmas 4.8 and 4.9 are slightly changed, where instead of the claim that $A^i_{(p_o)} \cap H = p$, we use the claim that $e \notin A^i_{(p_o)} \cap H$ and $A^i_{(p_o)} \cap H$ is not empty.

8.5 proof of Theorem 5.5

Clearly, the DC-MaNego problem is in NP. The proof of the NP-hardness is by a reduction from the Permutation Sum (definition 8.1).

Given an instance of the Permutation Sum we built an instance of the DC-MaNego problem as follows. There are $n+4$ outcomes: $x_1,\dots,x_n$, which correspond to the integers $X_1,\dots,X_n$, b_1, b_2 and two outcomes d and e. By Lemma 1 from [11], we can construct an election in which the non-manipulators cast votes such that:

$$p_t = (e, d, x_1, \dots, x_n, b_1, b_2),$$

and the corresponding scores are:

$$(4n+13+C, 2n+6+C, 2n+6+C-X_1, \dots, 2n+6+C-X_n,$$
$$C, y),$$

where C is a constant and $y < C$. The preference order of p_o is as follows:

$$p_o = (b_1, b_2, e, x_1, \dots, x_n, d)$$

We show that two manipulators can prevent the outcome e from being the SPE result iff the Permutation Sum problem has a solution.

($\Leftarrow$) Suppose we have two permutations σ and π of 1 to n such that $\sigma(i) + \pi(i) = X_i$. Let σ^{-1} be the inverse function of σ, i.e., $i = \sigma^{-1}(x)$ if $x = \sigma(i)$. We define $\pi^{-1}(x)$ similarly. We construct the following two manipulative votes:

$$(b_1, b_2, e, x_{\sigma^{-1}(n)}, \dots, x_{\sigma^{-1}(1)}, d)$$
$$(b_1, b_2, e, x_{\pi^{-1}(n)}, \dots, x_{\pi^{-1}(1)}, d)$$

Since the permutation sum satisfies $\sigma(i) + \pi(i) = X_i$ and $y < C$, the preference profile $p_t^M = \mathcal{F}(\vec{p} \cup \vec{p}_M)$ is:

$$(e, b_1, d, x_1, \dots, x_n, b_2)$$

since the corresponding scores are:

$$(4n+13+C+2(n+1), 2n+6+C, 2n+6+C, 2n+6+C, \dots,$$
$$2n+6+C, y+2(n+2)).$$

Therefore, $\mathcal{N}(p_t^M, p_o) = b_1 \neq e$.

($\Rightarrow$) Assume we have a successful manipulation. Clearly, every outcome $o \in \{x_1, \dots, x_n, d\}$ cannot be the SPE result since $e \succ_{p_o} o$. In addition, b_2 cannot be the SPE result, since in every possible manipulation $d \succ_{p_t^M} b_2$. Thus, outcome b_1 is the SPE result. Now, in every possible manipulation e is placed in the highest position in p_t^M due to its score in p_t. In addition, since b_1 is the SPE result it must be in the second highest position in p_t^M. Therefore, both manipulators have to place b_1 in the highest position in their preferences, and d in the lowest position in their preferences. Let $\sigma(i)$ be a function that determines the score that the first manipulator assigned to outcome x_i. $\pi(i)$ is defined similarly for the second manipulator. Since the manipulation is successful,

$$2n+6+C-X_i+\sigma(i)+\pi(i) \leq 2n+6+C,$$

and thus,

$$\sigma(i)+\pi(i) \leq X_i.$$

Since $\sum_{i=1}^n X_i = n(n+1)$,

$$\sum_{i=1}^n \sigma(i)+\pi(i) \leq n(n+1).$$

On the other hand, since d is placed in the lowest position by both manipulators,

$$\sum_{i=1}^n \sigma(i) \geq \frac{n(n+1)}{2}$$

and

$$\sum_{i=1}^n \pi(i) \geq \frac{n(n+1)}{2}.$$

Therefore, $\sum_{i=1}^n \sigma(i)+\pi(i) = n(n+1)$, and $\sum_{i=1}^n \sigma(i) = \sum_{i=1}^n \pi(i) = \frac{n(n+1)}{2}$. That is, σ and π are permutations of 1 to n. Moreover, since there is no slack in the inequalities,

$$\sigma(i)+\pi(i) = X_i.$$

Namely, there is a solution to the permutation sum problem.

8.6 proof of Theorem 5.6

We use Algorithm 2, and change it in the same way that we change Algorithm 1 in the proof of Theorem 5.2. Specifically, in lines 1- 2 we return true if $pos(e, p_o) < \lfloor m/2 \rfloor$, the set H is composed of p^* and the other $i-1$ most preferred outcomes in p_t that are not e, and in line 21 we check if $e \notin \mathcal{N}(\mathcal{F}(\vec{p} \cup \vec{p}_M), p_o)$. Following these changes the proof of correctness and the approximation is guaranteed similar to the proof of Theorem 4.11. Specifically, Lemmas 4.3, 8.5 and 8.4 still hold in the DC-MaNego setting. The proofs of Lemmas 8.2 and 8.3 are slightly changed, where instead of the claim that $A^i_{(p_o)} \cap H = p$, we use the claim that $e \notin A^i_{(p_o)} \cap H$ and $A^i_{(p_o)} \cap H$ is not empty.