Exact Algorithms for Weighted and Unweighted Borda Manipulation Problems

Yongjie Yang
Universität des Saarlandes
Saarbrücken, Germany
yyongjie@mmci.uni-saarland.de

Jiong Guo
Universität des Saarlandes
Saarbrücken, Germany
jguo@mmci.uni-saarland.de

1. INTRODUCTION
Voting systems have many applications in a variety of areas, including political election, web spam reduction, multi-agent planning, etc. The Borda system, proposed by Jean-Charles de Borda in 1781, is one of the most significant voting systems. It is the prototype of scoring systems and many other voting systems. The Borda system has been used for selecting presidential election candidates in some of the Pacific island countries such as Nauru and Kiribati. It also has been shown that the Borda system is a powerful technique for pattern recognition.

Certain issues which have been attracting much attention in voting systems are the strategic behaviors, e.g., one or more than one voter to influence the outcome of the elections by doing some tricks. By the celebrated Gibbard-Satterthwaite Theorem [14, 19], every reasonable voting system with at least three candidates can be attacked by the voters with providing insincere votes. However, from the viewpoint of complexity theory, if it is \( \mathcal{NP} \)-hard to determine how to influence the election, one may give up his attacking to the election. From this point, computational complexity could be a reasonable way to protect elections from being attacked. The first study in this direction was conducted by Bartholdi et al. in their seminal paper [16]. Since then, researches on computational complexity of strategic behaviors of voting systems have been opened up (See [15, 20, 9, 6, 2] for more details). Recently, many \( \mathcal{NP} \)-hard strategic behavior problems have been extensively studied from the view point of exact, exponential-time algorithms, for instance, manipulations [5], bribery problems [8], control problems [12], etc. For more recent development in this direction, we refer to the excellent survey by Betzler et al. [1]. We focus on deriving exact combinatorial algorithms for weighted and unweighted Borda manipulation problems, both of which have been proved \( \mathcal{NP} \)-hard [2, 6].

1.1 Preliminaries
A multiset \( S := \{s_1, s_2, ..., s_{|S|}\} \) is a generalization of a set where objects of \( S \) are allowed to appear more than one time in \( S \), that is, \( s_i = s_j \) is allowed for \( i \neq j \). An element of \( S \) is one copy of some object. We use \( s \in S \) to denote that \( s \) is an element of \( S \). The cardinality of \( S \) denoted by \( |S| \) is the number of elements contained in \( S \). For example, the cardinality of the multiset \( \{1,1,2,3,3\} \) is 6. For two multisets \( A \) and \( B \), we use \( A \uplus B \) to denote the multiset containing all elements in \( A \) and \( B \). For example, for \( A := \{1,1,2,3,3,4\} \) and \( B := \{1,2,3\} \), \( A \uplus B \) := \{1,1,1,2,2,2,3,3,3,3,4\}.

Normally, a voting system can be specified by a set \( C \)
of candidates, a multiset $\Pi_V := \{\pi_{v_1}, \pi_{v_2}, \ldots, \pi_{v_n}\}$ of votes casted by a corresponding set $V := \{v_1, v_2, \ldots, v_n\}$ of voters ($\pi_{v_i}$ is casted by $v_i$), and a voting protocol which maps the election $(C, \Pi_V, V)$ to a candidate $w \in C$ which we call the winner. Each vote $\pi_v \in \Pi_V$ is defined by a bijection $\pi_v : C \rightarrow [\#(C)]$ (in some other literature, a vote is defined as a linear order over the candidates), where $[n]$ denotes the set $\{1, 2, \ldots, n\}$. The position of a candidate $c \in \pi_v$ is the value of $\pi_v(c)$. We say a voter $v$ placing a candidate $c$ in his/her $x$-th position or a voter $v$ fixing his/her $x$-th position by the candidate $c$ if $\pi_v(c) = x$. The candidate placed in the highest, that is, the $|C|$-th, position in $\pi_v$ is called the most preferred candidate of $v$, the candidate placed in the second-highest, that is, the $|C| - 1$-th position in $\pi_v$ is called the second preferred candidate of $v$, and so on.

In the following, we use $m$ to denote the number of candidates. A Borda protocol can be defined by a vector $(m - 1, m - 2, \ldots, 0)$. Each voter contributes $m - 1$ score to his/her most preferred candidate, $m - 2$ score to his/her second preferred candidate, and so on. The winner is a candidate who has the highest total score. Here, we break a tie randomly, that is, if there is more than one candidate having the highest score, the winner will be chosen randomly from these candidates. In a weighted Borda system, each voter $v$ is associated with a non-negative integer weight $f(v)$ and contributes $f(v) \cdot (m - 1)$ score to his/her most preferred candidate, $f(v) \cdot (m - 2)$ score to his/her second preferred candidate, and so on. Accordingly, a candidate having the highest total score wins the election. The unweighted Borda system is the special case of the weighted Borda system where each voter has the unit weight of 1.

For a candidate $c$ and a voter $v$, we use $SC_v(c)$ to denote the score of $c$ which is contributed by $v$, that is, $SC_v(c) := f(v) \cdot (\pi_v(c) - 1)$ (in an unweighted Borda system, $SC_v(c) := \pi_v(c) - 1$). Let $SC_V(c)$ denote the total score of $c$ contributed by voters in $V$, that is, $SC_V(c) := \sum_{v \in V} SC_v(c)$.

In the setting of manipulators, we have, in addition to $V$, a set $V'$ of voters which are called manipulators. The manipulators form a coalition and desire to coordinate their votes to make a distinguished candidate win the new election with votes in $\Pi'_V \cup \Pi_V$, where $\Pi'_V$ is the multiset of votes casted by the manipulators. The formal definitions of the problems studied in this paper are as follows.

Unweighted Borda Manipulation (UBM)

Input: An election $(C \cup \{p\}, \Pi_V, V)$ where $p$ is not the winner, and a set $V'$ of $t$ manipulators.

Question: Can the manipulators cast their votes $\Pi'_V$ in such a way that $p$ wins the election $(C \cup \{p\}, \Pi'_V \cup \Pi_V, V \cup V')$?

Weighted Borda Manipulation (WBM)

Input: A weighted election $(C \cup \{p\}, \Pi_V, V, f : V \rightarrow \mathbb{N})$ where $p$ is not the winner, a set $V'$ of $t$ manipulators, and a weight function $f : V \rightarrow \mathbb{N}$.

Question: Can the manipulators cast their votes $\Pi'_V$ in such a way that $p$ wins the weighted election $(C \cup \{p\}, \Pi'_V \cup \Pi_V, V \cup V', f : V \cup V' \rightarrow \mathbb{N})$, where $f(v) = f_1(v)$ if $v \in V$ and $f(v) = f_2(v)$ otherwise.

Since we break ties randomly, in order to make $p$ the winner, the manipulators must assure that after the manipulation the distinguished candidate $p$ becomes the only candidate who has the highest total score among all candidates.

1.2 Related Work and Our Contribution

As one of the most prominent voting systems, complexity of strategic behaviors under the Borda system has been intensively studied. It is known that many types of bribery and control behaviors under the unweighted Borda system are $\mathcal{NP}$-hard. For manipulation behaviors, WBM is $\mathcal{NP}$-hard even when the election contains only three candidates [4]. Bartholdi et al. [6] showed that both UBM and WBM in the case of only one manipulator are polynomial-time solvable. The complexity of UBM in the case of more than one manipulator remained open for many years, until very recently it was proved $\mathcal{NP}$-hard even when restricted to the case of only two manipulators [2-4]. Heuristic and approximation algorithms for UBM have been studied in the literature [23, 6]. It is worthy to mention that Zuckerman et al. [22] showed that UBM admits an approximation algorithm which can output a success manipulation with $t + 1$ manipulators whenever the given instance has a success manipulation with $t$ manipulators. By applying the integer linear programming (ILP) technique, UBM can be solved exactly with a very high computational complexity $O^*(m^{O(m)})$ [2]. However, no purely combinatorial exact algorithm seems known for these problems. In particular, Betzler et al. [2] posed as an open problem whether UBM can be solved exactly with a running time single-exponentially depending on $m$ in the case of constant number of manipulators.

We propose two algorithms solving WBM and UBM in $O^*((m2^m)^{t+1})$ time and $O^*(2^m)$ time, respectively, where $t$ is the number of manipulators and $m$ is the number of candidates. Both algorithms are based on dynamic programming techniques. Our results imply that both WBM and UBM can be solved in time single exponentially on $m$ in the case of constant number of manipulators, answering the open question in [2]. Additionally, we improve the running time of the ILP-based algorithms for UBM to $O^*(2^{m^2 log m})$. The key here is to transfer UBM to an integer linear programming with $m^2$ variables. Furthermore, we study polynomial-time algorithms for UBM in the case of at most two manipulators under single-peaked elections (the definition of single-peaked election is introduced in Sec. [4]. Due to lack of space, some proofs are deferred to the long version.

2. ALGORITHM FOR WEIGHTED BORDA MANIPULATION

In this section, we present an exact combinatorial algorithm for WBM. The following observation is clearly true.

Observation 1. Every true-instance of WBM has a solution where each manipulator places the distinguished candidate $p$ in his/her highest position.

Let $(C \cup \{p\}, \Pi_V, V, f_1, f_2, t)$ be the given instance. Due to Observation 1, there must be a solution $\Pi'_V$ with $SC_{V \cup V'}(p) := SC_V(p) + \sum_{v \in V'} f(v') \cdot |C|$ if this instance is true. Therefore, to make $p$ the winner, $SC_{V'}(c) \leq g(c)$ should be satisfied for all $c \in C$, where $g(c) := SC_V(p) + \sum_{v \in V'} f(v') \cdot |C| - SC_V(c) - 1$. The value of $g(c)$ is called the capacity of $c$. Meanwhile, if in the given instance there is a candidate $c$ with $g(c) < 0$, then the given instance must be a false-instance. Therefore, in the following, we assume that the given instance contains no candidate $c$ with $g(c) < 0$. Based on these, we can reformulate WBM as follows:
Reformulation of WBM

Input: A set $C$ of candidates associated with a capacity function $g : C \rightarrow \mathbb{N}$, and a multiset $F := \{f_1, f_2, ..., f_t\}$ of non-negative integers.

Question: Is there a multiset $\Pi := \{\pi_1, \pi_2, ..., \pi_t\}$ of bijections mapping from $C$ to $|[C]|$ such that $\sum_{i=1}^t f_i \cdot (\pi_i(c) - 1) \leq g(c)$ holds for all $c \in C$?

Here, the bijection $\pi_i$ corresponds to the vote casted by the $i$-th manipulator and $f_i \in F$ corresponds to the weight of the $i$-th manipulator (suppose that a fixed ordering over the manipulators is given).

Our algorithm is based on a dynamic programming method which is associated with a boolean dynamic table defined as $DT(C, Z_1, Z_2, ..., Z_l)$, where $C \subseteq C$ is a subset of candidates, $Z_i \subseteq [|[C]|]$ and $|C| = |Z_i|$ for all $i \in [t]$. Here, each $Z_i$ encodes the positions that are occupied by the candidates of $C$ in the vote casted by the $i$-th manipulator. The entry $DT(C, Z_1, Z_2, ..., Z_l) = 1$ means that there is a multiset $\Pi = \{\pi_1, \pi_2, ..., \pi_t\}$ of bijections mapping from $C$ to $|[C]|$ such that for each $i \in [t]$, $\bigcup_{c \in C} \Pi(c) = Z_i$, and, for every candidate $c \in C$, $c$ is “safe” under $\Pi$. Here, we say a candidate $c$ is safe under $\Pi$, if $\sum_{i=1}^t f_i \cdot (\pi_i(c) - 1) \leq g(c)$. Intuitively, $DT(C, Z_1, Z_2, ..., Z_l) = 1$ means that we can place all candidates of $C$ in the positions encoded by $Z_i$ for all $i \in [t]$ without exceeding the capacity of any $c \in C$. Clearly, a given instance of WBM is a true-instance if and only if $DT(C, Z_1, Z_2, ..., Z_l) = 1$. The algorithm is as follows:

Initialization: For all $c \in C$ and $z_1, z_2, ..., z_l \in [|[C]|]$, if $\sum_{i=1}^t f_i \cdot (\pi_i(c) - 1) \leq g(c)$, then $DT(\{c\}, \{z_1\}, \{z_2\}, ..., \{z_l\}) = 1$; otherwise, $DT(\{c\}, \{z_1\}, \{z_2\}, ..., \{z_l\}) = 0$.

Updating: For each $l$ from 2 to $|[C]|$, we update the entries $DT(C, Z_1, Z_2, ..., Z_l)$ with $|C| = |Z_1| = |Z_2| = ... = |Z_l| = l$ as follows: if $\exists c \in C$ and $\exists z_i \in Z_i$ for all $i \in [l]$ such that $DT(C \setminus \{c\}, Z_i \setminus \{z_i\}, Z_j \setminus \{z_j\}, ..., Z_k \setminus \{z_k\}) = 1$ and $DT(\{c\}, \{z_1\}, \{z_2\}, ..., \{z_l\}) = 1$, then $DT(C, Z_1, Z_2, ..., Z_l) = 1$, otherwise, $DT(C, Z_1, Z_2, ..., Z_l) = 0$.

Theorem 2. WBM is solvable in $O^\ast((|[C]|^2 |C| + 1)^{t+1})$ time.

Proof. We consider the above algorithm for WBM. In the initialization, we check whether $\sum_{i=1}^t f_i \cdot (\pi_i(c) - 1) \leq g(c)$ for each candidate $c \in C$ and each encoded position $z_i \in [|[C]|]$ for each $i \in [t]$. Since there are $|C|$ many candidates and $|[C]|$ many positions to be considered for each $z_i$, the running time of the initialization is bounded by $O^\ast(|C|^t)$. In the recurrence, we compute $DT(C, Z_1, Z_2, ..., Z_l)$ for all $C \subseteq C$ and all $Z_1 \subseteq [|[C]|]$, $Z_2 \subseteq [|[C]|]$, ..., $Z_l \subseteq [|[C]|]$ with $|C| = |Z_1| = |Z_2| = ... = |Z_l| = l$, where $2 \leq l \leq m$. To compute each of them, we consider all possibilities of $c \in C$ and $z_1 \in Z_1, z_2 \in Z_2, ..., z_l \in Z_l$. For each possibility, we further check whether $DT(C \setminus \{c\}, Z_i \setminus \{z_i\}, Z_j \setminus \{z_j\}, ..., Z_k \setminus \{z_k\}) = 1$ and $DT(\{c\}, \{z_1\}, \{z_2\}, ..., \{z_l\}) = 1$. Since there are at most $|C|^{t+1}$ such possibilities, and there are at most $2^{(t+1)|C|}$ entries needed to be computed, we arrive at the total running time of $O^\ast((|[C]|^2 |C|)^{t+1})$.

The correctness directly follows from the meaning of the dynamic table we defined. □

In [2], Betzler et al. posed as an open question whether UBM in case of two manipulators can be solved in less than $O^\ast(|C|!)$ time. By Theorem 2, we can answer this question affirmatively.

Corollary 3. WBM (UBM is a special case of WBM) in case of two manipulators can be solved in $O^\ast(2^{|C|})$ time.

3. ALGORITHM FOR UNWEIGHTED BORDA MANIPULATION

In this section, we study the UBM problem. Recall that UBM is a special case of WBM where all voters have the same unit weight. However, compared to the weighted version, when we compute $SC\Pi(c)$ for a candidate $c$, it is irrelevant which manipulators placed $c$ in the $j$-th positions. The decisive factor is the number of manipulators placing $c$ in the $j$-th positions. This leads to the following approach where we firstly reduce UBM to a matrix problem and then solve this matrix problem by a dynamic programming technique, resulting in a better running time than in Corollary 3.

Firstly, the matrix problem is defined as follows.

Filling Magic Matrix (FMM)

Input: A multiset $g := \{g_1, g_2, ..., g_m\}$ of non-negative integers and an integer $t > 0$.

Question: Is there an $m \times m$ matrix $M$ with non-negative integers such that $\sum_{i=1}^{m} (j - 1) \cdot M[i][j] \leq g_i$ and $\sum_{j=1}^{m} M[i][j] = t$, and $\forall j \in [m]$, $\sum_{i=1}^{m} M[i][j] = t$?

In the following, we present an algorithm for FMM. The algorithm is based on a dynamic programming method associated with a boolean dynamic table $DT(l, T)$, where $l \in [m]$ and $T := \{T_j \in \mathbb{N} | j \in [m], T_j \leq t\}$ is a multiset. The entry $DT(l, T) = 1$ means that there is an $m \times m$ matrix $M$ such that: (1) $\sum_{j=1}^{m} M[i][j] = t$ for all $i \in [t]$; (2) $\sum_{i=1}^{m} M[i][j] = T_j$ for all $j \in [m]$; and (3) $\sum_{i=1}^{m} (j - 1) \cdot M[i][j] \leq g_i$ for all $i \in [t]$. It is clear that a given instance of FMM is a true-instance if and only if $DT(m, T[m]) = 1$, where $T[m]$ is the multiset containing $m$ copies of $t$. The algorithm for solving FMM is as follows.

Initialization: For each possible multisets $T := \{T_j \in \mathbb{N} | j \in [m], T_j \leq t\}$, if $\sum_{j=1}^{m} T_j = t$ and $\sum_{j=1}^{m} (j - 1) \cdot T_j \leq g_i$, then set $DT(1, T) = 1$; otherwise, set $DT(1, T) = 0$.

Updating: For each $l$ from 2 to $m$ and each possible multiset $T := \{T_j \in \mathbb{N} | j \in [m], T_j \leq t\}$, if there is a multiset $T' := \{T_j \in \mathbb{N} | j \in [m], T'_j \leq T_j\}$ such that $DT(l - 1, T') = 1$, $\sum_{j=1}^{m} (T_j - T'_j) = t$ and $\sum_{j=1}^{m} (j - 1) \cdot (T_j - T'_j) \leq g_i$, then set $DT(l, T) = 1$; otherwise, set $DT(l, T) = 0$.

Lemma 4. FMM is solvable in $O^\ast(2^{m^2})$ time.

Proof. In the initialization, we consider all possible multisets $T := \{T_j \in \mathbb{N} | j \in [m], T_j \leq t\}$ with $\sum_{j=1}^{m} T_j = t$ and $\sum_{j=1}^{m} (j - 1) \cdot T_j \leq g_i$. Since $T$ has at most $m^2$ possibilities, the running time of the initialization is bounded by $O^\ast(m^2)$. In the recurrence, we use a loop indicated by a variable $l$ with $2 \leq l \leq m$ to update $DT(l, T)$. In each loop we compute the values of the entries $DT(l, T)$ for all multisets $T := \{T_j \in \mathbb{N} | j \in [m], T_j \leq t\}$. To compute each of the entries, we check whether $DT(l - 1, T') = 1$, $\sum_{j=1}^{m} (T_j - T'_j) = t$ and $\sum_{j=1}^{m} (j - 1) \cdot (T_j - T'_j) \leq g_i$, for all possible multisets $T' := \{T'_j \in \mathbb{N} | j \in [m], T'_j \leq T_j\}$. Since there are at most $m^2$ possible multisets $T'$, the time to compute each $DT(l, T)$ is bounded by $O^\ast(m^2)$. Since $T$ has at most $m^2$ possibilities, there are at most $m^2$ entries needed to be computed in each loop, implying a total time of $O^\ast(m^4)$.
We now come to show how to solve UBM via FMM. A partial vote is a partial injection \( \pi : C \cup \{ p \} \rightarrow |C \cup \{ p \}| \) which maps a subset \( C \subseteq C \cup \{ p \} \) onto \( |C \cup \{ p \}| \) such that for any two distinct \( a_1, a_2 \in C \), \( \pi(a_1) \neq \pi(a_2) \). Here, \( C \) is the domain and \( \{ \{ a \} \mid a \in C \} \) is the codomain of \( \pi \). A position not in the codomain is called a free position. For simplicity, we define \( \pi(c) = i \) for \( c \notin C \).

LEMMA 5. UBM can be reduced to FMM in polynomial time.

PROOF SKETCH. Let \( \Gamma := ((C \cup \{ p \}, \Pi_V, \mathcal{V}, \mathcal{V}', t)) \) be an instance of UBM. By Observation 1, we know that if \( \Gamma \) is a true-instance there must be a solution \( \Pi_V \) such that each manipulator places \( p \) in his/her highest position. We assume that \( SC_V(p) + t \geq |C| - SC_V(c) - 1 \geq 0 \) for all \( c \in C \) as discussed in Section 2. Let \( (c_1, c_2, \ldots, c_{|C|}) \) be any arbitrary ordering of \( C \). We construct an instance \( \Gamma' := (t, g) \) of FMM where \( t \) has the same value as in \( \Gamma \) and \( g := \{ g_1, g_2, \ldots, g_{|C|} \} \) with \( g_i = SC_V(p) + t \cdot |C| - SC_V(c_i) - 1 \) for all \( i \in |C| \). It is clear that the construction takes polynomial time. In the following, we prove that \( \Gamma \) is a true-instance if and only if \( \Gamma' \) is a true-instance.

\[ \Rightarrow \text{: Given a solution } \Pi_V \text{ for } \Gamma, \text{ we can get a solution for } \Gamma' \text{ by setting } M[i][j] = \{(\pi \in \Pi_V \mid \pi(c_i) = j)\}, \text{ where } (\pi \in \Pi_V \mid \pi(c_i) = j) \text{ is the multiset containing all votes } \pi \in \Pi_V \text{ with } \pi(c_i) = j \text{. By the above construction, the correctness of } M \text{ is easy to verify.} \]

\[ \Leftarrow \text{: Let a } |C| \times |C| \text{ matrix } M \text{ be a solution for } \Gamma' := (t, g := \{g_1, g_2, \ldots, g_{|C|}\}). \text{ Then, a solution for } \Gamma, \text{ where there are exactly } M[i][j] \text{ manipulators who place } c_i \text{ in the } j \text{-th positions, can be constructed by the following polynomial algorithm. For simplicity, for a candidate } c_i \text{ and an integer } j \text{ with } 1 \leq j \leq |C|, c_i \sim j \text{ means that there are less than } M[i][j] \text{ manipulators who have already placed } c_i \text{ in the } j \text{-th position. For two partial votes } \pi \text{ and } \pi' \text{ and two candidates } c \text{ and } c', \text{ } (\pi, c) \leftrightarrow (\pi', c') \text{ means to switch the position of } c \text{ in } \pi \text{ and the position of } c' \text{ in } \pi' \text{ that is, if } \pi(c) = j, \pi'(c) = j' \text{ then, after } (\pi, c) \leftrightarrow (\pi', c'), \pi(c) = j, \pi'(c) = j'. \]
We now analyze the running time. The algorithm has three loops. The “for” loop in Step 3 has $|C|$ rounds. The “while” loop in Step 3.1 has at most $t$ rounds since each execution of the loop fixes a free position for some $\pi_i$. Due to Lemma 5, the “while” loop in Step 3.1.2.1 takes polynomial time. Summingy all above, the running time of the algorithm is polynomially in $t$ and $|C|$, which complete the proof.

Due to Lemmas 4 and 5, we get the following theorem.

**Theorem 7.** **UBM can be solved in $O^*(t^{2|C|})$ time.**

Next we show that FMM can be solved by an integer linear programming (ILP) based algorithm. The ILP contains $m^2$ variables $x_{ij}$ for $i, j \in [m]$ and, subject to the following restrictions:

$$
\begin{align*}
\sum_{i=1}^{m} x_{ij} &= t \text{ for all } j \in [m] \\
\sum_{j=1}^{m} x_{ij} &= t \text{ for all } i \in [m] \\
\sum_{j=1}^{m} (j - 1) \cdot x_{ij} &\leq g_i \text{ for all } i \in [m] \\
x_{ij} &\geq 0 \text{ for all } i, j \in [m]
\end{align*}
$$

where $t \in \mathbb{N}$ and $g := \{g_1, g_2, \ldots, g_m\}$ with $g_i \in \mathbb{N}$ for all $i \in [m]$ are input.

H. W. Lenstra [18] proposed an $O^*(\zeta^{O(C)})$-time algorithm for solving ILP with $\zeta$ variables. The running time was then improved by R. Kannan [17].

**Lemma 8.** An ILP problem with $\zeta$ variables can be solved in $O^*(\zeta^{O(C)})$ time.

Due to Lemmas 9 and 8, we have the following theorem.

**Theorem 9.** **UBM admits an algorithm with running time $O^*(2^{2|C|^2} \log |C|)$.**

### 4. BORDA MANIPULATION UNDER SINGLE-PEAKED ELECTIONS

The single-peaked election was introduced by D. Black in 1948 [3]. The complexity of many voting problems under single-peaked elections has been studied in the literature [11][1][13]. It turns out that many $\mathcal{NP}$-hard problems become polynomial-time solvable when restricted to single-peaked elections [1][13].

Given a set $C$ of candidates and a bijection $\mathcal{L} : C \to [|C|]$, we say that a vote $\pi \in \mathcal{L}$ is **coincident** with $\mathcal{L}$ if and only if for any three distinct candidates $a, b, c \in C$ with $\mathcal{L}(a) < \mathcal{L}(b) < \mathcal{L}(c)$ or $\mathcal{L}(c) < \mathcal{L}(b) < \mathcal{L}(a)$, $\pi(a) > \pi(b)$ implies $\pi(c) > \pi(a)$. The candidate having the highest value of $\pi(v(\cdot))$ is called the peak of $\pi_v$ with respect to $\mathcal{L}$. An election $(C, \Pi_v, \mathcal{L})$ is a single-peaked election if there exists a bijection $\mathcal{L} : C \to [|C|]$ such that all votes of $\Pi_v$ are co-incident with $\mathcal{L}$. We call such a bijection $\mathcal{L}$ a **harmonious order** of $(C, \Pi_v, \mathcal{L})$. See Fig. 2 for a concrete example of single-peaked elections.

It has been shown in [10] that one can test whether a given election is a single-peaked election in polynomial time. Moreover, a harmonious order can be found in polynomial time if the given election is a single-peaked election.

![Figure 2](image-url)

**Figure 2:** This figure shows a visual representation of a single-peaked election which contains five candidates $A, B, C, D, E$ and three votes $\pi_{v_b}$, $\pi_{v_c}$, and $\pi_{v_d}$ defined by $\pi_{v_b}(A) = 5$, $\pi_{v_b}(B) = 4$, $\pi_{v_b}(C) = 3$, $\pi_{v_b}(D) = 2$, and $\pi_{v_b}(E) = 1$; $\pi_{v_c}(A) = 5$, $\pi_{v_c}(B) = 4$, $\pi_{v_c}(C) = 3$, $\pi_{v_c}(D) = 2$, and $\pi_{v_c}(E) = 1$; $\pi_{v_d}(A) = 5$, $\pi_{v_d}(B) = 4$, $\pi_{v_d}(C) = 3$, $\pi_{v_d}(D) = 2$, and $\pi_{v_d}(E) = 1$.

For a manipulation problem under single-peaked elections, we are given a single-peaked election $(C \cup \{p\}, \Pi_v, \mathcal{L})$, a harmonious order $\mathcal{L}$, and a set $\mathcal{V'}$ of manipulators. We are asked whether the manipulators can cast their votes $\Pi_{v'}$ in such a way that all their votes are coincident with $\mathcal{L}$ and $p$ wins the election $(C \cup \{p\}, \Pi_v \cup \Pi_{v'}, \mathcal{V} \cup \mathcal{V'})$. In the following, let UBMISP and UBMISP denote the problems of UBM with only one manipulator and with exactly two manipulators under single-peaked elections, respectively.

It is known that the unweighted Borda manipulation problem is polynomial-time solvable with one manipulator [16] but becomes $\mathcal{NP}$-hard with two manipulators [2][6]. Here, we show that this problem with two manipulators can be solved in polynomial time in single-peaked elections.

**Theorem 10.** **Both UBMISP and UBMISP are polynomial-time solvable.**

All remaining parts of this section are devoted to prove Theorem 10.

To this end, let $(C \cup \{p\}, \Pi_v, \mathcal{L})$ be the given single-peaked election and $\mathcal{L}$ be a harmonious order of $(C \cup \{p\}, \Pi_v, \mathcal{L})$. Let $(i, i, i, i, p, r_1, r_2, ..., r_5)$ be an ordering of $C \cup \{p\}$ with $\mathcal{L}(i_1) < \mathcal{L}(i_2) < \mathcal{L}(i_3)$ and $\mathcal{L}(p) < \mathcal{L}(r_1) < \mathcal{L}(r_2) < \mathcal{L}(r_3)$, and let $\mathcal{L}_L := \{i, i, i, i, i\}$ and $\mathcal{L}_R := \{r_1, r_2, ..., r_5\}$.

For two partial votes $\pi_1$ with domain $C_1$ and $\pi_2$ with domain $C_2$, we say $\pi_1$ and $\pi_2$ are *comparable* if for every $c \in C_1 \cap C_2$, $\pi_1(c) = \pi_2(c)$. Furthermore, for such comparable partial votes $\pi_1$ and $\pi_2$, let $\pi_1 \sqcap \pi_2$ denote the partial vote $\pi$ with domain $C_1 \cup C_2$ such that $\pi(c) = \pi_1(c)$ for all $c \in C_1$ and $\pi(c) = \pi_2(c)$ for all $c \in C_2$.

For a partial vote $\pi$ with domain $C$ and a vote $\pi'$, we say $\pi$ is extendable to $\pi'$ if for any $a \in C$, $\pi(a) = \pi'(a)$.

The definition of single-peaked elections directly implies the following observation.
Lemma 13. For two bijections $L : C \cup \{p\} \rightarrow |C \cup \{p\}|$ and $\pi : C \cup \{p\} \rightarrow |C \cup \{p\}|$, $\pi$ is coincident with $L$ if and only if for any integer $j$ with $1 \leq j \leq |C|$, all candidates in the set $\{c \in C \cup \{p\} \mid \pi(c) \geq j\}$ lie consecutively in $L$.

The following observation directly follows from the above lemma.

Observation 14. Let $\pi$ be a vote with $p$ as the peak. $L$ is the harmonious order of the given election and $C_L, C_R$ are defined as above. Then, $\pi$ is coincident with $L$ if and only if for all $l_i, l_j \in C_L$ (resp. $r_j, r_j' \in C_R$), $i < j'$ (resp. $j < j'$) implies $\pi(l_i) > \pi(l_i)$ (resp. $\pi(r_j) > \pi(r_j')$).

4.1 Algorithm for UBM1SP

For two distinct candidates $c, c' \in C \cup \{p\}$, we say $c$ and $c'$ are neighbors if $|L(c) - L(c')| = 1$. Let $N(c)$ denote the set of neighbors of $c$. Clearly, every candidate has at most two neighbors. A block is a subset of candidates lying consecutively in $L$. For a block $S \subseteq C \cup \{p\}$, let $N(S) := \{c \in (C \cup \{p\}) \mid S \neq c \in N(c')\}$. It is easy to verify that $|N(S)| \leq 2$ for every block $S$.

Note that the polynomial-time algorithm for the general Borda manipulation problem with one manipulator proposed in [16] cannot be directly used for solving UBM1SP, since UBM1SP requires that the manipulator’s vote should be single-peaked. Our polynomial-time algorithm for UBM1SP is a slightly modified version of the algorithm in [16]. Basically, the algorithm places candidates one-by-one in the positions of the manipulator’s vote, from the highest to the lowest. The currently highest, unoccupied position of the vote is called the “next free position”. The details of the algorithm is as follows: (1) Place $p$ in the highest position; (2) Set $S = \{p\}$, where the block $S$ is used to store all candidates which have been placed in some positions by the manipulator; (3) If none of $N(S)$ can be “safely” placed in the next free position, then return “No”. A candidate $c$ can be safely placed in the $j$-th position of the manipulator if $SC(V(c) + j - 1 < SC(V(p) + |C|)$. The final score of $p$ is clearly $SC(V(p) + |C|)$. (4) Otherwise, place a candidate $c \in N(S)$, which can be safely placed in the next free position, in the next free position and set $S := S \cup \{c\}$. If $S = C \cup \{p\}$, return “Yes”, otherwise, go back to step (3).

Clearly, the above algorithm needs $O(m)$ time, where $m$ is the number of candidates: Each iteration extends the block $S$ by adding one new candidate to $S$. Since $S$ can be extended at most $|C|$ times and each iteration takes constant time, the total time is bounded by $O(m)$. The correctness of the above algorithm is easy to verify by Lemma 12 and Lemma 13.

4.2 Algorithm for UBM2SP

The algorithm for UBM2SP is similar to the one for UBM1SP: Greedily fill free positions of both manipulators until no free position remains or no candidate can be safely placed in a next free position.

Main Idea: Let $\pi_1$ and $\pi_2$ be the votes of the two manipulators $v_1$ and $v_2$. We use $S_1$ and $S_2$ to denote the sets of candidates that are already placed in some positions of $\pi_1$ and $\pi_2$, respectively. At the beginning, $S_1 = S_2 = \{p\}$ and we then iteratively extend $S_1$ and $S_2$ by placing candidates in $\pi_1$ and $\pi_2$. After each iteration, we will assure the invariant that $S_1 = S_2 = S$ is a block of $L$. In each iteration,

Figure 3: Case 2 in the proof of Lemma 12
The left-hand shows the original vote $\pi_0$ with $\pi_0(l_2) > \pi_0(l_3) > \pi_0(l_4) > \pi_0(r_1) > \pi_0(l_4)$ and is extendable to the original vote $\pi_0$. The right-hand shows the relaxed vote $\pi' \cup \pi''$ with $\pi_0(p) > \pi_0(l_1) > \pi_0(l_2) > \pi_0(l_4) > \pi_0(r_1) > \pi_0(l_4)$ and is extendable to the original vote $\pi_0$. Here $C_{small} = \{l_4, l_5, l_6, r_1, r_2, r_3\}$, $\pi'$ has domain $\{p, l_1, l_2, l_3\}$ and codomain $\{10, 9, 8, 7\}$, and $\pi''$ has domain $\{r_1, l_4, l_5, r_2, r_3, l_6\}$ and codomain $\{1, 2, 3, 4, 5, 6\}$.

Observation 11. Let $L$ be a harmonious order, $\pi_0$ be a vote which is coincident with $L$ and $\pi$ be the peak of $\pi_0$, then for any $c_1, c_2$ with $L(c_1) < L(c_2) < L(\pi)$ or $L(c) < L(c_2) < L(\pi_1)$, $\pi(c_1) < \pi(c_2) < \pi(c_0)$.

Lemma 12. Every true-instance of UBM under single-peaked elections has a solution where every manipulator places the distinguished candidate $p$ in his/her highest position.

Proof. Assume that $\Pi_1 \cup \Pi_2$ is a solution that contradicts the claim of the lemma, and $v$ is a manipulator who did not place $p$ in his/her highest position. Without loss of generality, assume that $C_R = \emptyset$. Let $a := |C_L|$ and $b := |C_L|$. Due to Observation 11, $\pi_0(p) > \pi_0(l_1) > \pi_0(l_2) > \pi_0(r_1) > \pi_0(r_2) > \ldots \pi_0(r_3)$ (2). We consider two cases. The first case is that for any $l \in C_L$, $\pi_0(l) > \pi_0(p)$. In this case, we can create a new solution by recasting $\pi_0$ with $\pi_0(p) > \pi_0(l_1) > \pi_0(l_2) > \pi_0(l_4) > \pi_0(r_1) > \pi_0(r_2) > \ldots \pi_0(r_3)$ (2). Let $C_{small} = \{l_4, l_5, l_6, r_1, r_2, r_3\}$. We can get a new solution by recasting $\pi_0$, where $\pi' \cup \pi''$ is the partial vote which has domain $\{C \cup \{p\}\} \setminus C_{small}$ and codomain $\{C \cup \{p\}\} \setminus C_{small} + 1$ such that $\pi_0(p) > \pi_0(l_1) > \pi_0(l_4) > \pi_0(r_1) > \pi_0(r_2) > \pi_0(r_3)$. The second case is that there is a $l \in C_L$ with $\pi_0(l) < \pi_0(p)$. In this case, there must be a $a \in [a]$ such that $\pi_0(l_1) > \pi_0(p)$ for all $j \in [a - 1]$ and $\pi_0(l) < \pi_0(p)$ for all $a \geq j > z$. Let $C_{small} = \{l_4, l_5, l_6, r_1, r_2, r_3\}$ and is extendable to the original vote $\pi_0$. Fig. 3 illustrates such a case.

For a subset $C$ of candidates and a bijection $\Lambda : C \cup \{p\} \rightarrow |C \cup \{p\}|$, we say that the candidates in $C$ lie consecutively in $\Lambda$ if and only if there are no $c_1, c_2 \in C$ and $c \in (C \cup \{p\}) \setminus C$ such that $L(c_1) < L(c) < L(c_2)$. If $C$ contains only two candidates $c_1$ and $c_2$, then we say $c_1$ and $c_2$ lie consecutively in $\Lambda$ if $|L(c_1) - L(c_2)| = 1$.

The following lemma has been proven in [21] (Theorem 1).
define $S = S_1 = S_2$ and let $l$ and $r$ be the two neighbors of $S$ in $L$. In order to keep $\pi_1$ and $\pi_2$ single-peaked, there are only two cases to consider: First, both next free positions of $\pi_1$ and $\pi_2$ are occupied by one of $l$ and $r$; Second, $l$ is placed in the next free position of one of $\pi_1$ and $\pi_2$, and $r$ is placed in the next free position of the other. If we have $c \in \{l, r\}$ satisfying $SCV(c) + 2\alpha < SCV(p) + 2|C|$, where $\alpha$ is the score contributed by one of $\pi_1$ and $\pi_2$ with placing $c$ in its next free position, then we prefer the first case, set $S_1 := S_1 \cup \{c\}$ and $S_2 := S_2 \cup \{c\}$, and proceed with the next iteration. Otherwise, we check whether the second case is “safe”, that is, $SCV(r) + \alpha < SCV(p) + 2|C|$ and $SCV(l) + \alpha < SCV(p) + 2|C|$. If not, then the given instance is a false-instance; otherwise, $S_1 := S_1 \cup \{l\}$ and $S_2 := S_2 \cup \{r\}$. Next, we have to restore the invariant that $S_1 = S_2$. Observe that $S_1, S_2$, and $S_1 \cap S_2$ all form blocks in $L$. Moreover, $S_1 \setminus S_2$ and $S_2 \setminus S_1$ form two blocks and both are neighbors to $S_1 \cap S_2$. We apply here another iteration to consider the candidates in $S_1 \setminus S_2$ (or $S_2 \setminus S_1$) one-by-one. For each of them in $S_1 \setminus S_2$ (or $S_2 \setminus S_1$), we place it in the highest safe position in $\pi_2$ (or $\pi_1$) and fill the “gaps” that are consecutive free positions with candidates not in $S_1 \cup S_2$. More details refer to Step 2.4 of the algorithm ALGO-UBM2SP.

For simplicity, we define some new notations. For a candidate $c$ and an integer $s$ with $1 \leq s \leq |C|$, we use $c \rightarrow s$ or $c \rightarrow s$ to denote that $c$ can (or cannot) be safely placed in the $(|C| + 1 - s)$-th position of some manipulator. We also use $c \rightarrow \{s_1, s_2\}$ or $c \nrightarrow \{s_1, s_2\}$ to denote that $c$ can (or cannot) be safely placed in the $(|C| + 1 - s_1)$-th position of one manipulator and in the $(|C| + 1 - s_2)$-th position of the other, simultaneously. For a block $S$, corresponding to a manipulator $v_i$, with $i = 1, 2$ and a candidate $c \in N(S_i)$, we use $extend(S_i, c)$ to denote the following operations: (1) place $c$ in the next free position of $\pi_i$, that is, set $\pi_i(c) = |C| + 1 - |S_i|$; and (2) extend $S_i$ with $S_i := S_i \cup \{c\}$.

The algorithm ALGO-UBM2SP for UBM2SP:

Step 1 Both manipulators place $p$ in their highest positions. Set $S_1 = S_2 = \{p\}$.

Step 2 While $S_1 \neq S_2 \neq \emptyset \cup \{p\}$, do let $S = S_1$.

Step 2.1 If $\exists c \in N(S)$ with $c \rightarrow \{|S_1|, |S_2|\}$, then extend($S_1, c$) and extend($S_2, c$).

Step 2.2 Else if $|N(S)| = 1$ and $N(S) \nrightarrow \{|S_1|, |S_2|\}$, then return “No”;

* Comments: In the case $N(S) = \{c\}$, $c$ is the only candidate which can be placed in the next free positions without destroying the single-peakedness. Thus, if $c \nrightarrow \{|S_1|, |S_2|\}$, return “No”.*

Step 2.3 Else, let $c, c' \in N(S)$. If $c \rightarrow |S_1|$ and $c' \rightarrow |S_2|$, then extend($S_1, c$) and extend($S_2, c'$), otherwise, return “No”.

Step 2.4 While $S_1 \neq S_2$, do

Step 2.4.1 Let $c$ be any candidate in $N(S_1 \cap S_2)$ that has already been placed in a position by only one manipulator $v \in \{v_1, v_2\}$.

Step 2.4.2 While $c \nrightarrow |S_1|$, where $z = 1$ if $v = v_2$ and $z = 2$ if $v = v_1$, do

Step 2.4.2.1 If $N(S_1) \setminus \{c\} = \emptyset$, return “No”;

Step 2.4.2.2 Else, let $c' = N(S_1) \setminus \{c\}$; if $c' \rightarrow |S_2|$, then extend($S_2, c'$); otherwise, return “No”;

End While 2.4

Step 2.4.3 extend($S_2, c$).

End While 2.4

End While 2

Step 3 Return “Yes”.

To show the correctness of the algorithm, we need the following lemma. For a vote $\pi$ and two integers $x \leq x'$ with $x, x' \in [|C| + 1)$, let $C_s(x, x') := \{c \in C \cup \{p\} \mid x \leq \pi(c) \leq x'\}$. Let

$$C(L, -c) := \begin{cases} C_L & \text{if } c \in C_R \\ C_R & \text{if } c \in C_L \end{cases}$$

Lemma 15. Let $\{\pi_1, \pi_2\}$ be a solution for UBM2SP, and $c \in C$ be a candidate with $\pi_1(c) = x$ and $\pi_2(c) = y$. If there are two integers $x', y'$ such that (1) $x' > x, y' > y$; (2) $SCV(c) + x' + y' - 2 < SCV(p) + 2|C|$; (3) $C_s(x, x') \subseteq C(L, -c), C_{\pi_2}(y + 1, y') \subseteq C(L, -c)$, then, the following two votes $\pi_1', \pi_2'$ with

$$\pi_1'(c') = \begin{cases} \pi_1(c') & \text{if } \pi_1(c') > x' \text{ or } \pi_1(c') < x \\ x' & \text{if } c' = c \\ j & \text{if } x < \pi_1(c') = j + 1 \leq x' \end{cases}$$

$$\pi_2'(c') = \begin{cases} \pi_2(c') & \text{if } \pi_2(c') > y' \text{ or } \pi_2(c') < y \\ y' & \text{if } c' = c \\ j & \text{if } y < \pi_2(c') = j + 1 \leq y' \end{cases}$$

must be another solution.

Proof. We can transfer $\pi_1$ to $\pi_1'$ by continually switching the position of $c$ and the position of the candidate $c'$ with $\pi_1(c') = \pi_1(c) + 1$, until $c$ is moved to the $x'$-th position. Since $c' \in C(L, -c)$, due to Observation 14 each switching will not change a vote which is coincident with $L$ to a vote which is not coincident with $L$. By the same way, we can transfer $\pi_2$ to $\pi_2'$ such that $\pi_2'$ is coincident with $L$. Since $SCV(c) + x' + y' - 2 < SCV(c) + 2|C|$, and each switch does not increase the total scores of any other candidates, $\{\pi_1', \pi_2'\}$ must be a solution.

We now come to show the correctness of the algorithm. We say two partial votes $\pi_1, \pi_2$ with domain $S_1, S_2$ with $|S_1| \leq |S_2|$, respectively, are extendable to a solution, if there is a solution $\{\pi_1, \pi_2\}$ such that $\{\pi_1^{(-1)}(j), \pi_2^{(-1)}(j)\} = \{\pi_1^{-1}(j), \pi_2^{-1}(j)\}$ for all $|C| + 2 - |S_1| \leq j \leq |C| + 1$ and $\pi_2^{(-1)}(j) \in \{\pi_1^{-1}(j), \pi_2^{-1}(j)\}$ for all $|C| + 2 - |S_2| \leq j < |C| + 2 - |S_1|$. (Note that $\pi_2^{-1}$ is the inverse of $\pi_2$)

Lemma 16. ALGO-UBM2SP solves UBM2SP correctly in $O(m)$ time, where $m$ is the number of candidates.

Proof Sketch. Firstly, we consider the correctness. In each step, the algorithm extends $S_1$ (or $S_2$) by adding a candidate from $N(S_1)$ (or $N(S_2)$) to $S_1$ (or $S_2$), which makes $S_1$ (or $S_2$) always be a block. Due to Lemma 15 if the algorithm returns “Yes”, the two votes $\pi_1$ and $\pi_2$ must be coincident with $L$. We need the following observations.

Observation 17. If at some point $S_1 = S_2 = S$, $\pi_1, \pi_2$ are extendable to a solution, and a candidate $c \in N(S)$ satisfies $c \rightarrow \{|S|, |S|\}$, then the partial votes $\pi_1$ with $\pi_1(c') = \pi_1(c')$ for $c' \in S$ and $\pi_1(c') = |C| + 1 - |S|$ and $\pi_2$ with $\pi_2(c') = \pi_2(c')$ for $c' \in S$ and $\pi_2(c') = |C| + 1 - |S|$ are extendable to a solution.
Observation 18. If at some point \( S_1 = S_2 = S \) with \( N(S) = \{ l, r \} \), \( \pi_1, \pi_2 \) are extendable to a solution, no candidate \( c \in N(S) \) satisfies \( c \rightarrow \{ |S|, |S| \} \), and \( l \rightarrow |S| \) and \( r \rightarrow |S| \), then the partial votes \( \pi'_1 \) with \( \pi'_1(c') = \pi_1(c') \) for \( c' \in S \) and \( \pi'_2(l) = |S| + 1 - |S| \) and \( \pi'_2 \) with \( \pi'_2(c') = \pi_2(c') \) for \( c' \in S \) and \( \pi'_2(r) = |S| + 1 - |S| \) are extendable to a solution.

Observation 19. If at some point two partial votes \( \pi_1 \) and \( \pi_2 \) with domains \( S_1 \) and \( S_2 \), respectively, are extendable to a solution and there is a candidate \( c \) which has been placed in some position in \( \pi_1 \) (but has not been placed by the other vote, that is, \( \pi_2 \rightarrow \{ |S| \} \)), then, the partial votes \( \pi_1 \) and \( \pi'_2 \) with \( \pi'_2(c') = \pi_2(c') \) for \( c' \in S_2 \) and \( \pi'_1(l) = |S| + 1 - |S| \) and \( \pi'_1 \) with \( \pi'_1(c') = \pi_1(c') \) for \( c' \in S_1 \) and \( \pi'_1(r) = |S| + 1 - |S| \) are extendable to a solution.

The correctness of the above observations follows from Lemma 15.

Due to Lemma 12, Step 1 is correct. Then, we consider the while loop in Step 2. Due to Observation 17, if Step 2.1 was executed and the given instance is a true-instance, then, the partial votes after the execution of Step 2.1 must be extendable to a solution. Otherwise, if Step 2.1 was not executed, and \( N(S) \) contains only one candidate, then the given instance must be a false-instance as described in the comments of Step 2.2. Finally, if the \( \text{extend} \) operations in Step 2.3 was executed, then, due to Observation 17, these operations are correct. If Step 2.3 returns “No”, then the given instance must be a false-instance. The reason for this is that, due to Lemma 13 and Observation 17, if the given instance is a true-instance, then \( \pi_1, \pi_2 \) must be extendable to a solution and thus, the next free positions must be occupied by \( N(S) \). If the conditions of Steps 2.1 to 2.3 are not satisfied, the next free positions can not be safely placed, then, we can return “No”. After the execution of \( \text{extend} \) operations in Step 2.3, \( S_1 \neq S_2 \), and the algorithm then goes to the while loop in Step 2.4. Since \( S_1 \cap S_2 \neq \emptyset \) (at least \( p \in S_1 \cap S_2 \) and both \( S_1 \) and \( S_2 \) are blocks (this is easy to check by the definition of \( \text{extend} \)), there must be at least one candidate \( c \in N(S_1 \cap S_2) \) which has already been placed in a position by a manipulator \( v \in \{ v_1, v_2 \} \). Due to Observation 17, if the given instance is a true-instance, then we can place \( c \) in the position \( y = \min \{ SCV(p) + 2|C| - SCV_1(c) - \pi_1(c) + 1, |C| + 1 - |S_1| \} \), where \( i = 1, z = 2 \) if \( v = v_1 \) and \( i = 2, z = 1 \) if \( v = v_2 \), to get partial votes which are extendable to a solution. Due to Lemma 13, all free positions higher than the \( g \)-th position can only fixed by candidates from \( C(L, \cdots) \setminus S_2 \), and each of these free positions is fixed by an unique candidate from \( C(L, \cdots) \setminus S_1 \). These are exactly what the while loop in 2.4.2 does: fixing the free positions which are higher than the \( g \)-th position one-by-one. After the loop, Step 2.4.3 places \( c \) in the \( g \)-th position. However, if \( C(L, \cdots) \setminus S_j \) is less than the number of free positions which are higher than the \( g \)-th position, then due to Lemma 13 the given instance must be a false-instance. Due to the above analysis, each case of Steps 2.1 to 2.4 correctly extends the partial votes to two new partial votes which are also extendable to a solution, or correctly returns “No”.

To analyze the running time, we need to consider how many times Steps 2.1, 2.3, 2.4.1, 2.4.2, 2.4.3 are executed. For simplicity, let \( n_1 \) denote the number of times that Step 2 is executed throughout the algorithm. It is clear that \( n_{2.4.1} = n_{2.4.3} \). Since each execution of Steps 2.1, 2.3, 2.4.2, 2.4.3 extends at least one of \( S_1 \) and \( S_2 \) by adding a new candidate to the block or terminates the algorithm, and both \( S_1 \) and \( S_2 \) can be extended at most 2|C| times, we conclude that \( n_{2.1} + n_{2.3} + n_{2.4.1} + n_{2.4.2} + n_{2.4.3} = O(|C|) \). Since each execution of these Steps needs constant time\(^3\), we arrive at a total running time of \( O(|C|) \).

5. CONCLUSION AND FUTURE WORK

In this paper, we initiate the study of exact combinatorial algorithms for Borda manipulation problems. We propose two exact combinatorial algorithms with running times \( O((m - 2)^t) \) and \( O((i^m) \) for WBM and UBM, respectively, where \( t \) is the number of manipulators and \( m \) is the number of candidates in the given election. These results answer an open problem posed by Betzler et al. [2]. In addition, we present an integer linear programming based algorithm with running time \( O(2^{3m-log^3}) \) for UBM. Finally, we study UBM under single-peaked elections and propose polynomial-time algorithms for UBM in case of no more than two manipulators. We mention here that all our algorithms can be used with slight modifications for (positional) scoring systems.

One future direction could be to improve the presented combinatorial algorithms. Parameterized complexity has been proved a powerful tool to handle \( NP \)-hard problems. As showed here and in [2], UBM is fixed-parameter tractable with respect to the number \( m \) of candidates. A challenging work is to improve the running time to \( O(2^{3m-log^3}) \), or even \( O(e^m) \) with \( e \) being a constant. Furthermore, the complexity of UBM in the case of more than two manipulators under single-peaked elections remains open.

6. REFERENCES

[1] N. Betzler, R. Bredereck, J. Chen, and R. Niedermeier. Studies in computational aspects of voting - a parameterized complexity perspective. In The Multivariate Algorithmic Revolution and Beyond, pages 318–363, 2012.
[2] N. Betzler, R. Niedermeier, and G. J. Woeginger. Unweighted coalitional manipulation under the Borda rule is \( NP \)-hard. In IICAI, pages 55–60, 2011.
[3] D. Black. On the rationale of group decision-making. Journal of Political Economy, 56:23–34, 1948.
[4] F. Brandt, M. Brill, E. Hemaspaandra, and L. A. Hemaspaandra. Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorate. In AAAI, pages 715–722, 2010.
[5] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? J. ACM, 54(3):1–33, 2007.
[6] J. Davies, G. Katsirelos, N. Narodytska, and T. Walsh. Complexity of and algorithms for Borda manipulation. In AAAI, pages 657–662, 2011.
[7] J. de Borda. Mémoire sur les élections par scrutin. Des Sciences, pages 657–665, 1781.
[8] B. Dorn and I. Schlotter. Multivariate complexity analysis of swap bribery. Algorithmica, 64(1):126–151, 2012.

\(^3\)To calculate \( N(S_1 \cap S_2) \) in Step 2.4.1 in constant time, we need some proper date structure like double-linked queue.
[9] Gábor Erdélyi, Lena Piras, and Jörg Rothe. The complexity of voter partition in bucklin and fallback voting: solving three open problems. In AAMAS, pages 837–844, 2011.

[10] B. Escoffier, J. Lang, and M. Öztürk. Single-peaked consistency and its complexity. In ECAI, pages 366–370, 2008.

[11] P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. The complexity of manipulative attacks in nearly single-peaked electorates. In TARK, pages 228–237, 2011.

[12] P. Faliszewski, E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. Llull and copeland voting computationally resist bribery and constructive control. J. Artif. Intell. Res. (JAIR), 35:275–341, 2009.

[13] P. Faliszewski, E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. The shield that never was: Societies with single-peaked preferences are more open to manipulation and control. Inf. Comput., 209(2):89–107, 2011.

[14] A. Gibbard. Manipulation of voting schemes: A general result. Econometrica, 41(4):587–601, 1973.

[15] Egor Ianovski, Lan Yu, Edith Elkind, and Mark C. Wilson. The complexity of safe manipulation under scoring rules. In IJCAI, pages 246–251, 2011.

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

[17] R. Kannan. Minkowski’s convex body theory and integer programming. Mathematics of Operations Research, 12:415–440, 1987.

[18] H. W. Lenstra. Integer programming with a fixed number of variables. Mathematics of Operations Research, 8(4):538–548, 1983.

[19] M. Satterthwaite. Strategy-proofness and arrow’s conditions: Existence and correspondence theorems for voting procedures and social welfare functions. Journal of Economic Theory.

[20] Svetlana Obraztsova and Edith Elkind. On the complexity of voting manipulation under randomized tie-breaking. In IJCAI, pages 319–324, 2011.

[21] M. A. Trick. Recognizing single-peaked preferences on a tree. Mathematical Social Sciences, 17(3):329–334, 1989.

[22] T. Walsh. Uncertainty in preference elicitation and aggregation. In AAAI, pages 3–8, 2007.

[23] M. Zuckerman, A. D. Procaccia, and J. S. Rosenschein. Algorithms for the coalitional manipulation problem. Artif. Intell., 173(2):392–412, 2009.