On the Computational Hardness of Manipulating Pairwise Voting Rules

Rohit Vaish[*] Neeldhara Misra[†] Shivani Agarwal[‡] Avrim Blum[§]

September 7, 2018

Abstract

Standard voting rules usually assume that the preferences of voters are provided in the form of complete rankings over a fixed set of candidates. This assumption does not hold in applications like recommendation systems where the set of candidates is extremely large and only partial preferences can be elicited. In this paper, we study the problem of strategic manipulation of voting rules that aggregate voter preferences provided in the form of pairwise comparisons between candidates. Our contribution is threefold: First, we show that any pairwise voting rule under which each candidate has some chance of winning is manipulable in principle. Next, we analyze how the computational complexity of manipulation of such rules varies with the structure of the graph induced by the pairs of candidates that the manipulator is allowed to compare and the type of the preference relation. Building on natural connections between the pairwise manipulation and sports elimination problems, we show that manipulating pairwise voting rules can be computationally hard even in the single manipulator setting—a setting where most standard voting rules are known to be easy to manipulate. Finally, we develop a comprehensive picture of the parameterized complexity of the manipulation problem in terms of various natural structural parameters associated with the action space of the manipulator.

1 Introduction

A central result in social choice theory (Arrow et al., 2010) is the Gibbard-Satterthwaite theorem (Gibbard, 1973; Satterthwaite, 1975), which states that any non-dictatorial voting rule over at least three candidates under which each candidate has some chance of winning is susceptible to strategic voting (i.e., is *manipulable*). Given the impossibility suggested by this theorem, there has been substantial work concerning a finer analysis of the situation and finding possible workarounds. A prominent example is the seminal work of Bartholdi III et al. (1989b) who proposed the possibility of using computational hardness as a barrier against the manipulation of voting rules. They argue that although the opportunities for manipulation always exist in principle, there might not exist an efficient general purpose algorithm for finding them in practice.

Since then, a considerable body of work has developed around the computational study of manipulation (Faliszewski and Procaccia, 2010; Walsh, 2011; Brandt et al., 2016). Much of this work models voter preferences as *complete rankings* over a fixed set of candidates. While this is a reasonable choice for certain situations, many large-scale settings prevalent today (e.g., recommendation systems) involve extremely large candidate sets (e.g., movies, products, webpages). It is therefore unreasonable, and often impractical, to ask the users to provide complete

[*]Indian Institute of Science, Bangalore - 560012, India. rohitv@iisc.ac.in
[†]Indian Institute of Technology Gandhinagar, Gandhinagar - 382355, India. mail@neeldhara.com
[‡]University of Pennsylvania, Philadelphia, PA 19104, USA. ashivani@seas.upenn.edu
[§]Toyota Technological Institute at Chicago, Chicago, IL 60637, USA. avrim@ttic.edu

rankings of these candidates. For such settings, it is decidedly more natural to aggregate *partial* preferences such as top-k rankings (Ailon, 2010; Narodytska and Walsh, 2014; Menon and Larson, 2017) or partial orders (Pini et al., 2009) to arrive at an outcome. For a similar reason, some situations might call for relaxing the requirement of *transitivity* among the preferences. Indeed, when candidates are compared using not one but multiple quality parameters, it is natural to permit possibly cyclic preferences (Elkind and Shah, 2014).

In this work, we consider the model of *pairwise preferences*, where each vote is simply a collection of pairwise comparisons between the candidates, with no constraints other than anti-symmetry (i.e., $A \succ B \Rightarrow B \not\succ A$). Voter preferences can therefore be *incomplete* (i.e., not all pairs of candidates are compared by a voter) and can even contain *cycles* (e.g., $A \succ B, B \succ C, C \succ A$). Modeling votes as sets of pairwise comparisons offers substantial generality, and, as a result, there has been growing interest in designing algorithms that elicit pairwise preferences from the users, especially in the machine learning and crowdsourcing communities (Lu and Boutilier, 2011; Jiang et al., 2011; Ailon, 2012; Negahban et al., 2012; Chen et al., 2013; Rajkumar and Agarwal, 2014). Although these studies explore the statistical properties of algorithms for aggregating pairwise preferences, the question of whether these algorithms (or *pairwise voting rules*) are resistant to strategic behavior by the users remains to be answered. The focus of the present work is to address these questions both from axiomatic and computational perspectives.

Our Contributions

1. We show that any pairwise voting rule that is onto (i.e., under which each candidate has some chance of winning) must be *manipulable* (Section 4.1).

2. Given the above impossibility, we study settings under which computational difficulty can provide a worst-case barrier against manipulation (Section 4.2). Specifically, we study the problem of manipulation of pairwise voting rules along the following two dimensions: (a) the *structure of the graph* $\mathcal{A}$ induced by the pairs of candidates that the manipulator is allowed to compare (e.g., tree, bipartite, complete graph) and (b) the *type of preference relation* `pref-type` (e.g., total, acyclic). Our analysis focuses on the *pairwise Borda* (pBorda) voting rule and the Copeland$^\alpha$ family of voting rules, both of which are defined in Section 2.1. Tables 1 to 4 summarize our computational results for these rules. The essence of our results is that *the structure of the induced graph and the type of revealed preferences (and the parameter α in case of* Copeland$^\alpha$*) can shape the complexity landscape in important ways.* We remark that while the manipulation problem in the context of rankings is efficiently solvable for most voting rules (Bartholdi III et al., 1989b), we already encounter hardness results for the relatively simple pBorda and Copeland$^\alpha$ rules in the pairwise preferences model, which we consider an important contrast.[1]

3. Our final set of results (Section 4.3) pertains to a comprehensive classification of the parameterized complexity of the manipulation problem in terms of various parameters of the graph structure such as *maximum degree*, size of *minimum feedback vertex set*, size of *minimum vertex cover*, and several others, as summarized in Table 5. Figure 1 shows how these parameters are related to each other.

2 Preliminaries

This section provides all the relevant definitions and concepts required for the presentation of our results in Section 4.

[1] Even in the context of rankings, however, voting rules such as the second-order Copeland rule (Bartholdi III et al., 1989b) and many elimination-style rules (Bartholdi III and Orlin, 1991; Xia et al., 2009; Davies et al., 2012, 2014) are known to be computationally resistant to manipulation by a single voter.

		pref-type			
		total+acyclic	acyclic	total	unrestricted
$\mathcal{A}$	tree	**P** (Theorem 3)	**P** (Theorem 3)	**P** (Theorem 3)	**P** (Theorem 3)
	complete graph (or clique)	**P** (Theorem 3)	**NP-complete** (Theorem 8)	**NP-complete** (Theorem 8)	**NP-complete** (Theorem 8)
	bipartite graph	**P** (Theorem 3)	**NP-complete** (Theorem 5)	**NP-complete** (Theorem 7)	**NP-complete** (Theorem 5)
	general graph (no restriction)	**P** (Theorem 3)	**NP-complete** (Theorem 5)	**NP-complete** (Theorem 7)	**NP-complete** (Theorem 5)

Table 1: *The complexity of* pBorda-MANIPULATION *under the conditions specified by the corresponding action space $\mathcal{A}$ and type of preference relation* `pref-type` *(refer to Section 2.2 for relevant definitions).*

		pref-type			
		total+acyclic	acyclic	total	unrestricted
$\mathcal{A}$	tree	**P** (Corollary 4)	**P** (Corollary 4)	**P** (Corollary 4)	**P** (Corollary 4)
	complete graph (or clique)	**P** (Corollary 4)	**P** (Theorem 9)	**P** (Theorem 9)	**P** (Theorem 9)
	bipartite graph	**P** (Corollary 4)	**P** (Theorem 9)	**P** (Theorem 9)	**P** (Theorem 9)
	general graph (no restriction)	**P** (Corollary 4)	**P** (Theorem 9)	**P** (Theorem 9)	**P** (Theorem 9)

Table 2: *The complexity of* Copeland^0-MANIPULATION *and* Copeland^1-MANIPULATION.

2.1 Pairwise Preferences and Pairwise Voting Rules

Let $[n] = \{1, 2, \ldots, n\}$ denote the set of *candidates* and $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ denote the set of *voters*. Let $\succ_u \subseteq [n] \times [n]$ denote the binary relation indicating the preferences (or vote) of the voter u, so that $i \succ_u j$ indicates that voter u prefers candidate i over candidate j. For each pair of candidates i, j and each voter u, we can have exactly one of $i \succ_u j$, $j \succ_u i$ or neither (i.e., voter u *skips* the comparison between i and j). We let $\mathcal{R}$ denote the set of all such asymmetric binary relations on $[n]$, and let $\Pi = (\succ_{u_1}, \succ_{u_2}, \ldots, \succ_{u_m}) \in \mathcal{R}^m$ denote the *pairwise preference profile* of the voters.

A *pairwise voting rule* r maps a pairwise preference profile $\Pi \in \cup_{k=1}^{\infty} \mathcal{R}^k$ to a unique candidate $r(\Pi) \in [n]$. Given a preference profile $\Pi \in \mathcal{R}^m$ and a pair of candidates i, j, let $m_{ij}(\Pi)$ denote the number of voters who strictly prefer candidate i over candidate j. That is, $m_{ij}(\Pi) = \sum_{k=1}^{m} [(i \succ_{u_k} j) \in \Pi]$ where $[\cdot]$ denotes the Iverson bracket.[2] A *score-based pairwise voting rule* is any pairwise voting rule r for which there exists a scoring function $\mathbf{s} : \cup_{k=1}^{\infty} \mathcal{R}^k \to \mathbb{R}^n$ such that $r(\Pi)$ is the highest-scoring candidate according to $\mathbf{s}(\Pi)$ under some fixed tie-breaking rule. That is, $r(\Pi) = T\left(\arg\max_i s_i(\Pi)\right)$ for some tie-breaking rule $T : 2^{[n]} \setminus \{\emptyset\} \to [n]$ satisfying $T(S) \in S$ for all non-empty $S \subseteq [n]$. Some examples of score-based pairwise voting rules are the following:

[2]The Iverson bracket $[P]$ equal 1 whenever the predicate P is true, and is 0 otherwise.

$\mathcal{A}$		pref-type: total+acyclic	acyclic	total	unrestricted
	tree	**P** (Corollary 4)	**P** (Corollary 4)	**P** (Corollary 4)	**P** (Corollary 4)
	complete graph (or clique)	**P** (Corollary 4)	**P** (Theorem 9)	**NP-complete** (Theorem 12)	**P** (Theorem 9)
	bipartite graph	**P** (Corollary 4)	**P** (Theorem 9)	**NP-complete** (Theorem 12)	**P** (Theorem 9)
	general graph (no restriction)	**P** (Corollary 4)	**P** (Theorem 9)	**NP-complete** (Theorem 12)	**P** (Theorem 9)

Table 3: *The complexity of* Copeland$^{0.5}$-Manipulation.

$\mathcal{A}$		pref-type: total+acyclic	acyclic	total	unrestricted
	tree	**P** (Corollary 4)	**P** (Corollary 4)	**P** (Corollary 4)	**P** (Corollary 4)
	complete graph (or clique)	**P** (Corollary 4)	**NP-complete** (Theorem 10)	**NP-complete** (Theorem 12)	**NP-complete** (Theorem 10)
	bipartite graph	**P** (Corollary 4)	**NP-complete** (Theorem 10)	**NP-complete** (Theorem 12)	**NP-complete** (Theorem 10)
	general graph (no restriction)	**P** (Corollary 4)	**NP-complete** (Theorem 10)	**NP-complete** (Theorem 12)	**NP-complete** (Theorem 10)

Table 4: *The complexity of* Copeland$^{\alpha}$-Manipulation *for* $\alpha \in \mathbb{Q} \cap (0,1) \setminus \{0.5\}$.

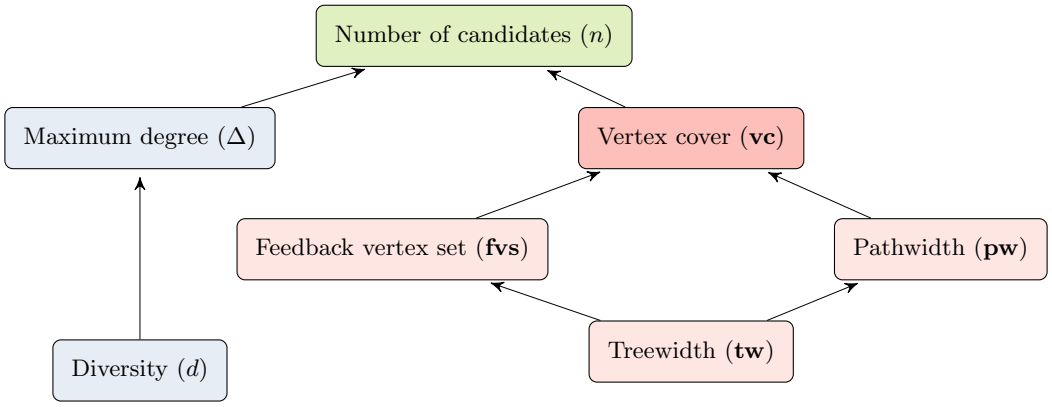
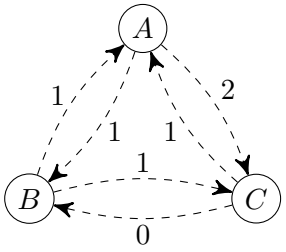


Figure 1: *A Hasse diagram depicting the relationship between the various parameters (refer to Section 2.7 for relevant definitions). Each arrow is directed from a smaller parameter to a larger one. The implications for parameterized tractability (FPT, XP) propagate upwards along the figure in the direction of the arrows, while intractability (W-hardness) propagates in the opposite direction.*

<table>
<tr><td></td><td>Vertex cover</td><td>Pathwidth</td><td>Feedback vertex set</td><td>Treewidth</td><td>∅</td></tr>
<tr><td>Diversity</td><td>FPT
(Theorem 18)</td><td colspan="3">W[1]-hard and XP
(Theorem 15 and Corollary 17)</td><td rowspan="2">para-NP-complete
(Theorem 5)</td></tr>
<tr><td>Max degree</td><td colspan="4">FPT (Corollary 17)</td></tr>
<tr><td>∅</td><td colspan="5">para-NP-complete (Theorem 14)</td></tr>
</table>

Table 5: *Parameterized complexity results for* pBorda-Manipulation *under the conditions specified by the corresponding combination of parameters. The notation ∅ is used to enable the consideration of singleton parameters. Merged cells indicate combined parameters (refer to Section 2.7 for relevant definitions). Notice that the columns 'vertex cover' and 'feedback vertex set' indicate the* sizes *of the minimum vertex cover and the minimum feedback vertex set respectively, and not the sets themselves.*



(a) *An election instance*

	A	B	C
pBorda	$1/2 + 2/3$	$\mathbf{1/2 + 1}$	$1/3 + 0$
Copeland^0	$\mathbf{0+1}$	$0 + 1$	$0 + 0$
Outdegree	$\mathbf{1+2}$	$1 + 1$	$1 + 0$

(b) *Candidate scores under various voting rules*

Figure 2: *This figure illustrates an election instance for which the winning candidate is decided according to the various pairwise voting rules defined in Section 2.1. Subfigure (a) depicts the election instance as a multigraph, where each vertex represents a candidate, and each dashed, directed edge represents a strict pairwise comparison. Thus, for instance, an edge from A to C denotes the pairwise comparison $A \succ C$ by some voter. The numbers alongside the edges denote the number of voters with that preference (e.g., two voters vote $A \succ C$ while one voter votes $C \succ A$). Subfigure (b) lists the scores of the candidates under the* pBorda, Copeland^0 *and Outdegree rules (under the tie-breaking rule $A \succ B \succ C$). The winner's score for each voting rule is highlighted in boldface.*

1. *Pairwise Borda rule* or pBorda (Rajkumar and Agarwal, 2014): The pBorda score of candidate i under the preference profile Π is given by

$$s_i^{\text{pBorda}}(\Pi) = \sum_{j=1}^{n} \frac{m_{ij}(\Pi)}{m_{ij}(\Pi) + m_{ji}(\Pi)}$$

   where we adopt the convention $0/0 = 0$. Note that this rule is equivalent to the standard Borda count rule (Arrow et al., 2010) when all the votes are provided as complete rankings over the set of candidates.

2. *Copeland$^\alpha$ rule* (Faliszewski et al., 2008): The Copeland$^\alpha$ score of candidate i (for any rational $\alpha \in [0, 1]$) under the preference profile Π is given by

$$s_i^{\text{Copeland}^\alpha}(\Pi) = \sum_{j=1}^{n} [m_{ij}(\Pi) > m_{ji}(\Pi)] + \alpha \cdot [m_{ij}(\Pi) = m_{ji}(\Pi)],$$

   where $[\cdot]$ once again denotes the Iverson bracket.

3. *Outdegree rule*: This rule provides another way of generalizing the standard Borda count rule to the domain of pairwise preferences. The Outdegree score of candidate i under the

preference profile Π is given by

$$s_i^{\text{Outdeg}}(\Pi) = \sum_{j=1}^{n} m_{ij}(\Pi).$$

Other examples of pairwise voting rules include PageRank (Brin and Page, 2012), HodgeRank (Jiang et al., 2011), Ranked Pairs and Schulze's rule (Parkes and Xia, 2012). Figure 2 instantiates the pairwise voting rules defined above on a small election instance. Notice that *any* standard voting rule defined over the domain of rankings can in principle be *extended* to the domain of pairwise preferences.

2.2 Manipulation of Pairwise Voting Rules

The focus of this paper is on manipulation problems involving a *single manipulator*, who has *complete information*[3] about the votes of all the other voters (called the *non-manipulators*). Formally, a pairwise voting rule r is said to be *manipulable* if there exists a pair of profiles $\Pi = (\succ_{u_1}, \ldots, \succ_{u_{m-1}}, \succ_{u_m}) \in \mathcal{R}^m$ and $\Pi' = (\succ_{u_1}, \ldots, \succ_{u_{m-1}}, \succ'_{u_m}) \in \mathcal{R}^m$ differing only in the preference of voter u_m such that $r(\Pi') \succ_{u_m} r(\Pi)$. That is, voter u_m (called *the manipulator*) strictly prefers the new outcome over the old one according to its old preference $\succ_{u_m}$.

Given a pairwise voting rule r, we refer to the computational problem faced by the manipulator as r-Manipulation, which is defined below. Our formulation of r-Manipulation problem involves two additional notions that are central to this work—the action space $\mathcal{A}$ and the preference type `pref-type`. The *action space* $\mathcal{A} \subseteq \binom{[n]}{2}$ refers to the pairs of candidates that the manipulator is allowed to compare.[4] Stated differently, the manipulator is not allowed to compare any pair of candidates outside of $\mathcal{A}$. The parameter `pref-type` specifies whether the preferences of the manipulator over $\mathcal{A}$ are required to be *total* (i.e., the manipulator must provide strict comparisons for all candidate pairs in $\mathcal{A}$), *acyclic* (i.e., directed cycles of the form $1 \succ 2, 2 \succ 3, 3 \succ 1$ are not allowed), *total+acyclic* (both total and acyclic) or *unrestricted* (no such restriction).

r-Manipulation

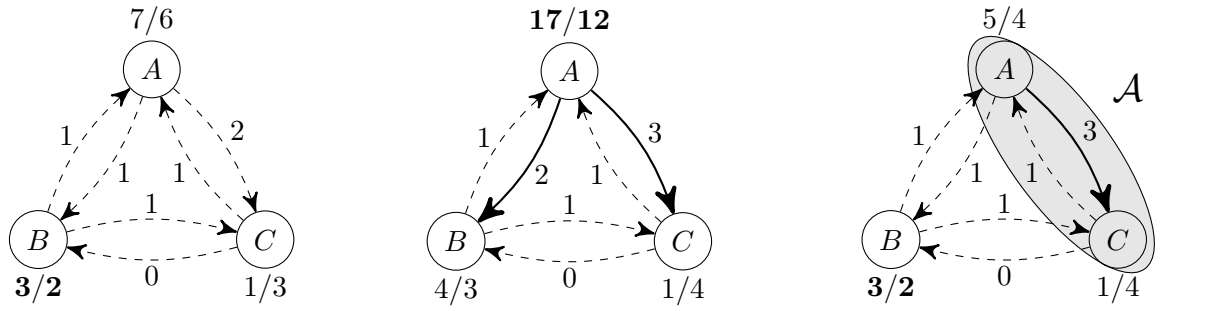
Input:	A tuple $\langle \Pi, i^*, \mathcal{A}, \texttt{pref-type} \rangle$, where $\Pi \in \mathcal{R}^{m-1}$ is the preference profile of the non-manipulators $(u_1, \ldots, u_{m-1})$, $i^* \in [n]$ is the distinguished candidate, $\mathcal{A} \subseteq \binom{[n]}{2}$ is the set of pairwise comparisons that the manipulator is allowed to make, and `pref-type` $\in$ {total+acyclic, total, acyclic, unrestricted} is the preference constraint with respect to $\mathcal{A}$.
Question:	Does there exist a vote $\succ_{u_m}$ over $\mathcal{A}$ satisfying `pref-type` such that $r(\Pi \cup \{\succ_{u_m}\}) = i^*$?

The parameters $\mathcal{A}$ and `pref-type` together capture a rich variety of preference models. For example, when $\mathcal{A}$ = complete graph and `pref-type` = total+acyclic, we recover the standard manipulation problem where the manipulator is required to provide a *complete ranking* of the candidates.[5] When $\mathcal{A}$ = complete graph and `pref-type` = total, the manipulator is required to come up with a *tournament* vote, which generalizes the standard setting by relaxing the

[3] A detailed discussion on the *complete information* assumption can be found in the work of Hemaspaandra et al. (2007). The authors thank an anonymous reviewer for the pointer.

[4] This is not to be confused with the term "strategy space" often used in game theory.

[5] A technical difference from the standard setting is that unlike in the standard setting, we do not require the votes of the non-manipulators to be rankings.



(a) *Before manipulation* **(b)** *After manipulation* **(c)** *After restricted manipulation*

Figure 3: *An illustration of the election instance in Example 1. Subfigure (a) shows the same election instance as in Figure 2, where each vertex of the multigraph represents a candidate, and each dashed, directed edge represents a strict pairwise comparison. The numbers alongside the edges denote the number of non-manipulators with that preference. The* pBorda *scores of the candidates are mentioned next to the corresponding vertices, and the winner's score is highlighted in boldface. Subfigure (b) shows the pairwise comparisons made by the manipulator as solid edges (notice that the count of voters with that preference goes up by one). Subfigure (c) shows the restricted action space of the manipulator* ($\mathcal{A} = \{\{A, C\}\}$) *as shaded in* ***grey****.*

transitivity constraint. Similarly, the setting where $\mathcal{A}$ = complete graph and `pref-type` = transitive (i.e., requiring $i \succ k$ whenever $i \succ j$ and $j \succ k$) corresponds to a *partial order* vote of the manipulator. Finally, when $\mathcal{A}$ = complete graph and `pref-type` = acyclic, we further relax the partial order requirement by allowing the manipulator to skip a comparison between candidates 1 and 3 while allowing for $1 \succ 2$ and $2 \succ 3$.

We refer to the instantiation of r-Manipulation for pBorda rule (respectively Copeland$^\alpha$) as pBorda-Manipulation (respectively Copeland$^\alpha$-Manipulation). We will assume throughout that ties are broken in favor of the manipulator. That is, in order to make the distinguished candidate i^* win the election, the manipulator only needs to ensure that i^* is *one of* the highest scoring candidates.

Example 1 illustrates the role of the action space $\mathcal{A}$ in the manipulation problem.

Example 1 (The role of the action space $\mathcal{A}$)**.** *Consider an election setting with three candidates A, B and C, with the votes of non-manipulators as shown in Figure 3a. The initial* pBorda *scores of these candidates are* 7/6, 3/2 *and* 1/3 *respectively, making B the* pBorda *winner. Let the favorite candidate of the manipulator be A. Observe that if the manipulator votes $\{A \succ B, A \succ C\}$ (as shown in Figure 3b), the new* pBorda *scores will be* 17/12, 4/3 *and* 1/4, *and the manipulator will be able to make A the winner. Thus, the given* pBorda-Manipulation *instance admits a feasible solution when $\mathcal{A} = \{\{A, B\}, \{A, C\}\}$ or $\mathcal{A} = \{\{A, B\}, \{A, C\}, \{B, C\}\}$ and* `pref-type` = unrestricted*. However, if the manipulator is only allowed to compare the candidates A and C (i.e., $\mathcal{A} = \{\{A, C\}\}$), then despite voting in favor of A, the manipulator cannot make A win (Figure 3c). Thus, there is no feasible solution to the given* pBorda-Manipulation *instance when $\mathcal{A} = \{\{A, C\}\}$ and* `pref-type` = unrestricted*. It is easy to see that the above observations continue to hold when* `pref-type` $\in$ {total, acyclic, total+acyclic}*.*

A note on the formulation of action space: The idea of using the action space $\mathcal{A}$ to restrict the vote of the manipulator is a novel contribution of this paper and has not been previously studied to the best of our knowledge. Formulating the manipulation problem in terms of a general action space $\mathcal{A}$ (instead of implicitly working with $\mathcal{A}$ = complete graph) offers two advantages: First, with the benefit of hindsight, we know that by varying the structure

of the action space $\mathcal{A}$, we can cover the entire spectrum of the computational complexity of the manipulation problem—both in terms of classical (Table 1) and parameterized (Table 5) complexity. Second, the concept of action space is also interesting from the viewpoint of *elicitation*. Indeed, as our results demonstrate (Table 1), requiring the voters to compare all $\mathcal{O}(n^2)$ pairs is computationally resistant to manipulation, but might be infeasible in practice due to high elicitation cost. On the other hand, eliciting comparisons over tree-structured pairs requires only $\mathcal{O}(n)$ elicitations but is susceptible to manipulation. Therefore, considering other action spaces such as bipartite graphs allows for "the best of both worlds," i.e., computational resistance to manipulation with small elicitation cost.

2.3 Vote Configuration

We will often use the shorthand $a : b$ (for non-negative integers a, b) for an ordered pair of candidates (i, j) to denote that a voters vote $i \succ j$ while b other voters vote $j \succ i$. We will refer to $a : b$ as the *vote configuration* between i and j. Thus, for instance, in Figure 3a, the candidate pair (A, C) is in $2 : 1$ configuration.

2.4 Score Transfer Property of the pBorda Rule

An intriguing property of the pBorda rule that will be useful in our proofs is the following: For any given pair of candidates (i, j), the increase in pBorda score of i due to a vote $i \succ j$ of the manipulator is equal to the decrease in pBorda score of j. As an example of this, consider the election instance in Figure 3a, consisting entirely of non-manipulators' votes. Suppose the manipulator votes $A \succ C$, resulting in the instance in Figure 3c. The pBorda score of A increases by $\frac{5}{4} - \frac{7}{6} = \frac{1}{12}$, while that of C decreases by $\frac{1}{3} - \frac{1}{4} = \frac{1}{12}$, which are equal.

It is easy to show that this is true in general. Indeed, consider an election instance where the pair (i, j) is in $m_{ij} : m_{ji}$ configuration. Let us assume for simplicity that there is no other candidate in the election. The current pBorda scores of i and j are therefore $\frac{m_{ij}}{m_{ij}+m_{ji}}$ and $\frac{m_{ji}}{m_{ij}+m_{ji}}$ respectively. Let us now add the vote $i \succ j$ of the manipulator. The new pBorda score of i is $\frac{m_{ij}+1}{m_{ij}+m_{ji}+1}$, meaning that its score increases by $\frac{m_{ji}}{(m_{ij}+m_{ji})(m_{ij}+m_{ji}+1)}$. The new pBorda score of j is $\frac{m_{ji}}{m_{ij}+m_{ji}+1}$, meaning that its score decreases by the same amount, i.e., $\frac{m_{ji}}{(m_{ij}+m_{ji})(m_{ij}+m_{ji}+1)}$. This property holds for all candidate pairs for which $m_{ij} + m_{ji} \neq 0$.

Thus, the change in the pBorda scores of the candidates due to the manipulator's vote can be described in terms of a *score transfer*. This is the amount of pBorda score that gets *transferred* from j to i when the manipulator votes $i \succ j$. If the manipulator skips the comparison, the pBorda scores stay unchanged and no score transfer takes place. Define a *score transfer vector* as a tuple of score transfers due to the votes $i \succ j$, skip or $j \succ i$ of the manipulator. Thus, for instance, the score transfer vector for the candidate pair (A, C) in Figure 3a is $\left[\left(+\frac{1}{12}, -\frac{1}{12}\right), (0, 0), \left(-\frac{1}{6}, +\frac{1}{6}\right)\right]$.

Notice that the score transfer property described above is specific to the pBorda rule, and does not hold in general for other pairwise voting rules like Copeland$^\alpha$. For instance, consider a Copeland$^\alpha$ election instance with $\alpha = 0$, where the pair (i, j) is in 1:0 configuration and the manipulator votes $j \succ i$. In this case, the score of i decreases by 1, while the score of j stays the same.

2.5 Excess Scores

The *excess score* of a candidate i refers to the amount by which the score of i exceeds the score of the distinguished candidate i^* in a given election instance. For example, in Figure 3c in Section 2.2, the excess pBorda scores of the candidates B and C (with respect to the distinguished candidate A) are $1/4$ and -1 respectively. Hence, r-Manipulation for a score-based voting

rule r can be restated as deciding whether there exists a vote for the manipulator such that the final excess scores of all candidates are zero or less.

2.6 Parameterized Complexity

A *parameterized problem* is denoted by a pair $(Q, k) \subseteq \Sigma^* \times \mathbb{N}$. The first component Q is a classical language and the second component k is a number (called the *parameter*). Such a problem is called *fixed–parameter tractable* (FPT) if there exists an algorithm that decides it in time $\mathcal{O}(f(k)n^{\mathcal{O}(1)})$ on instances of size n for some computable function $f : \mathbb{N} \to \mathbb{N}$.

Just as NP-hardness is used as evidence that a problem probably is not polynomial-time solvable, there exists a hierarchy of complexity classes above FPT, and showing that a parameterized problem is hard for one of these classes is considered evidence that the problem is unlikely to be fixed-parameter tractable with respect to the corresponding parameter. The main classes in this hierarchy are

$$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[P] \subseteq XP,$$

where a parameterized problem belongs to the class XP if there exists an algorithm for it with running time bounded by $n^{g(k)}$ for some computable function $g : \mathbb{N} \to \mathbb{N}$. A detailed introduction to parameterized complexity theory can be found in the standard texts by Downey and Fellows (1999); Flum and Grohe (2006); Niedermeier (2006) and Cygan et al. (2015).

A parameterized problem is said to be *para-NP-complete* if it is NP-complete even for constant values of the parameter. A classic example of a para-NP-complete problem is Graph Coloring parameterized by the number of colors (Garey and Johnson, 1979)—recall that it is NP-complete to determine if a graph can be properly colored with three colors. Observe that a para-NP-complete problem does not belong to XP unless $P = NP$.

For any pair of parameterized problems A and B, we say that A is (uniformly many:1) *FPT-reducible* to B if there exists an algorithm Φ that transforms an instance (x, k) of A into an instance (x', k') of B such that

- (x, k) is a yes-instance of A if and only if (x', k') is a yes-instance of B,
- $k' \leq g(k)$ for some computable function g, and
- the running time of Φ is $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some computable function $f : \mathbb{N} \to \mathbb{N}$ (where $|x|$ denotes the size of the input).

A convenient way of showing that a problem is W[1]-hard is via an FPT reduction from a known W[1]-hard problem. Hence, in the above definition, if the problem A is known to be W[1]-hard in parameter k and there exists an FPT reduction from A to B, then B is W[1]-hard in the parameter $g(k)$.

A note on the use of combined parameters: When working with multiple parameters $k_1, k_2, \ldots, k_r$ associated with a given problem, we will think of the *combined parameter* k as being given by $k = \kappa(k_1, k_2, \ldots, k_r)$, where $\kappa : \mathbb{N}^r \to \mathbb{N}$ is a fixed, computable function. The standard notions of parameterized running time (such as FPT, XP) extend naturally to this case. For instance, we will say that a parameterized problem is FPT when *simultaneously parameterized* by $k_1, k_2, \ldots, k_r$ if, for some fixed, computable function $\kappa : \mathbb{N}^r \to \mathbb{N}$, there exists an algorithm that decides it in time $\mathcal{O}(f(k)n^{\mathcal{O}(1)})$ on instances of size n, where $f : \mathbb{N} \to \mathbb{N}$ is some computable function and $k = \kappa(k_1, k_2, \ldots, k_r)$. Similarly, we will say that a parameterized problem is in XP when simultaneously parameterized by $k_1, k_2, \ldots, k_r$ if, for some fixed, computable function $\kappa : \mathbb{N}^r \to \mathbb{N}$, there exists an algorithm that decides it in time $\mathcal{O}(n^{g(k)})$ on instances of size n, where $g : \mathbb{N} \to \mathbb{N}$ is some computable function and $k = \kappa(k_1, k_2, \ldots, k_r)$. Finally, for any pair of parameterized problems A and B, where A is parameterized by k and B

is simultaneously parameterized by $k_1, \ldots, k_r$, we will say that A is (uniformly many:1) *FPT-reducible* to B if there exists an algorithm Φ that transforms an instance (x, k) of A into an instance $(x', k_1, \ldots, k_r)$ of B such that

- (x, k) is a yes-instance of A if and only if $(x', k_1, \ldots, k_r)$ is a yes-instance of B,
- $\kappa(k_1, k_2, \ldots, k_r) \leq g(k)$ for some computable functions $\kappa : \mathbb{N}^r \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$, and
- the running time of Φ is $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some computable function $f : \mathbb{N} \to \mathbb{N}$.

2.7 Parameters Used in This Work

We will now define the various parameters used in our parameterized complexity results. We will assume throughout that $G = (V, E)$ denotes a simple undirected graph, unless explicitly stated otherwise.

Maximum degree (Δ): The *maximum degree* of a graph G is the maximum number of edges incident to any vertex of G.

Vertex cover (**vc**): A set of vertices $V' \subseteq V$ is a *vertex cover* of G if for every edge $(u, v) \in E$, either $u \in V'$ or $v \in V'$ or both. Throughout the paper, we will use the phrase "parameterized by vertex cover" to refer to the parameterization with respect to the size of a minimum vertex cover (and not the set itself).

Feedback vertex set (**fvs**): A *feedback vertex set* of a graph is a set of vertices whose removal makes the graph acyclic. Throughout the paper, we will use the phrase "parameterized by feedback vertex set" to refer to the parameterization with respect to the size of a minimum feedback vertex set (and not the set itself).

Tree decomposition: A tree decomposition of a graph G is a tuple $\mathcal{T} = (T, \{B_t\}_{t \in V(T)})$ where T is a tree, $V(T)$ is the vertex set of T, and each node t of T is assigned a set of vertices $B_t \subseteq V$ (called a *bag*) such that the following hold:

1. For each vertex $v \in V$, there exists a node t of T such that $v \in B_t$. In other words, $\cup_{t \in V(T)} B_t = V$.
2. For each edge $(u, v) \in E$, there exists a node t such that $u \in B_t$ and $v \in B_t$.
3. For each vertex $v \in V$, the set of nodes $\{t \in V(T) : v \in B_t\}$ forms a connected subtree of T.

The *width* of a tree decomposition $\mathcal{T} = (T, \{B_t\}_{t \in V(T)})$ equals $\max_{t \in V(T)} |B_t| - 1$ (i.e., size of the largest bag minus one). The *treewidth* (**tw**) of a graph G is the minimum possible width of a tree decomposition of G.

Nice tree decomposition: A tree decomposition $\mathcal{T} = (T, \{B_t\}_{t \in V(T)})$ is *nice* if T is a rooted binary tree (with root node r) and the following conditions hold:

1. The root node and all leaf nodes correspond to empty bags (i.e., $B_r = \emptyset$ and $B_\ell = \emptyset$ for every leaf ℓ of T).
2. Each non-leaf node of T is of one of the following three types:
   - *Introduce node*: a node t with exactly one child t' such that $B_t = B_{t'} \cup \{v\}$ for some vertex $v \in V$. We say that v is *introduced* at node t.
   - *Forget node*: a node t with exactly one child t' such that $B_t = B_{t'} \setminus \{v\}$ for some vertex $v \in V$. We say that v is *forgotten* at node t.
   - *Join node*: a node t with exactly two children t_1, t_2 such that $B_t = B_{t_1} = B_{t_2}$.

The notions of *pathwidth* (**pw**), *path decomposition* and *nice path decomposition* are defined analogously in terms of paths (instead of trees).

We will implicitly assume in all our algorithmic results that the parameter of interest (e.g., size of a *minimum vertex cover* of $\mathcal{A}$ or the treewidth of a *nice tree decomposition* of $\mathcal{A}$) is provided as an input along with the graph $\mathcal{A}$, and our algorithms are designed to work directly on these inputs. This is without loss of generality, since the overall running time in these situations is dominated by the procedures that make use of these inputs and not the procedures that compute these parameters from a given graph (Chen et al., 2010; Bodlaender et al., 2013).

In addition to the above-mentioned parameters, we introduce a new structural parameter called *diversity*, as defined below.

Definition 1. DIVERSITY (d)*:* The *diversity* of the action space $\mathcal{A}$ (with respect to an underlying preference profile Π of non-manipulators' votes) is defined as the maximum number of distinct score transfers that a candidate can experience due to a single pairwise comparison made by the manipulator. As an example, consider the election instance shown in Figure 3a in Section 2.2, and consider the candidate A in particular. Assuming that $\mathcal{A}$ = complete graph, the manipulator is free to compare any of the pairs $\{A, B\}$, $\{B, C\}$ or $\{C, A\}$. If the manipulator compares the pair $\{A, B\}$, then the pBorda-score of candidate A can change by $+1/6$, 0 or $-1/6$ respectively, depending on whether the manipulator votes $A \succ B$, 'skip' or $B \succ A$. This can be concisely represented as a vector $(+1/6, 0, -1/6)$, called the *score transfer vector for A from the pair (A, B)*. Similarly, the score transfer vectors for candidate A from the pairs (A, C) and (B, C) are $(+1/12, 0, -1/6)$ and $(0, 0, 0)$ respectively. Since there are three such distinct vectors, the *diversity* for candidate A is three. The *diversity of an instance* is the maximum diversity for any candidate. Notice that the *diversity* of an election instance can depend crucially on the pairwise voting rule. Indeed, the *diversity* under the pBorda rule can be as large as $n-1$, while the same for Copeland$^\alpha$ is $\mathcal{O}(1)$ due to the limited types of score exchanges permitted under the definition of the Copeland voting rule. It is easy to see that for any pairwise voting rule where a pairwise comparison by the manipulator can only affect the scores of the two candidates involved—examples include pBorda and Copeland$^\alpha$—*diversity* is at most the *maximum degree*, i.e., $d \leq \Delta$.

2.8 Elimination Problem in Sports

The sports elimination problem (Schwartz, 1966) asks the following question: Given the current scores of all teams and the set of games remaining to be played in a sports competition, is it possible for a team i^* to still win the competition? As we will see, this problem turns out to be intimately connected with the manipulation problem described earlier. Formally, let $[N] = \{1, 2, \ldots, N\}$ be the set of N teams and $s_i \in \mathbb{R}$ denote the current score of team $i \in [N]$. Let $\mathcal{G} \subseteq \binom{[N]}{2}$ denote the set of remaining games between pairs of teams (played according to some predefined schedule). The points awarded to each team based on the outcome of the game (e.g., *home win*, *draw* or *away win*) are given by a *scoring system* S, defined as follows:

Definition 2. SCORING SYSTEM: A scoring system S is a tuple of ordered pairs of rational numbers $[(\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots, (\alpha_t, \beta_t)]$ where the subscripts $1, 2, \ldots, t$ correspond to the outcomes of games, and the first and second entries of each pair (denoted by α and β) correspond to the points awarded respectively to the home and away team under that outcome.

Example 2. *A popular example of a scoring system is the one used by several football leagues in Europe. Each national league typically consists of 20 teams, and each pair of teams play two games against each other during the season—one home and one away. The winner in each game is awarded* 3 *points, while the losing team does not get any points. In case of a draw, both teams get a point each. Thus, the European football scoring system is given by* $\mathrm{S}_1 = [(3, 0), (1, 1), (0, 3)]$*, where* $t = 3$*. A variant of the above system can be* $\mathrm{S}_2 = [(3, 0), (1, 2), (0, 3)]$ *where the away*

team gets an extra point under a draw. Another example is the baseball scoring system $\mathrm{S}_3 = [(1,0),(0,1)]$*, which only admits win-loss outcomes (and thus* $t = 2$*).*

Given the vector of current scores $\mathbf{s} = (s_1, s_2, \ldots, s_N)^T$ and the set of games $\mathcal{G}$ to be played between pairs of teams, the question "*can team* i^* *still win the competition?*" can be formalized as the following computational problem called S-Elimination (Kern and Paulusma, 2004):

S-Elimination

Input:	A tuple $\langle \mathbf{s}, i^*, \mathcal{G} \rangle$, where $\mathbf{s} = (s_1, s_2, \ldots, s_N)^T$ is the vector of current scores of the N teams, $i^* \in [N]$ is the distinguished team and $\mathcal{G} \subseteq \binom{[N]}{2}$ is the set of remaining games between the teams.
Question:	Does there exist an assignment of outcomes for the games in $\mathcal{G}$ such that i^* ends up with the (joint) highest overall score among all teams under the scoring system S?

Specifically, Kern and Paulusma (2004) consider a special case of the above problem where the distinguished team i^* has no remaining games (in other words, the score of i^* is frozen). They refer to this variant as *partial sports competition* problem (PSC). In our work, we will use the term S-Elimination to refer to the PSC problem.

Theorem 1 (Kern and Paulusma, 2004)**.** *Let* $\mathrm{S} = [(\alpha_1, \beta_1), \ldots, (\alpha_t, \beta_t)]$ *be a scoring system satisfying* $\alpha_1 > \cdots > \alpha_{t-1} = 1 > \alpha_t = 0$ *and* $0 = \beta_1 < 1 \leq \beta_2 < \cdots < \beta_t$ *for some* $t \in \mathbb{N}$*. Then,* S-Elimination *is efficiently solvable if* $\mathrm{S} = \{(t-1-i, i) : 0 \leq i \leq t-1\}$*. In all other cases, the problem is* NP-complete*.*

Thus, for instance, S-Elimination is efficiently solvable for the baseball scoring system $\mathrm{S}_3 = [(1,0),(0,1)]$, but is computationally intractable for the European football scoring system $\mathrm{S}_1 = [(3,0),(1,1),(0,3)]$, or its variant $\mathrm{S}_2 = [(3,0),(1,2),(0,3)]$.

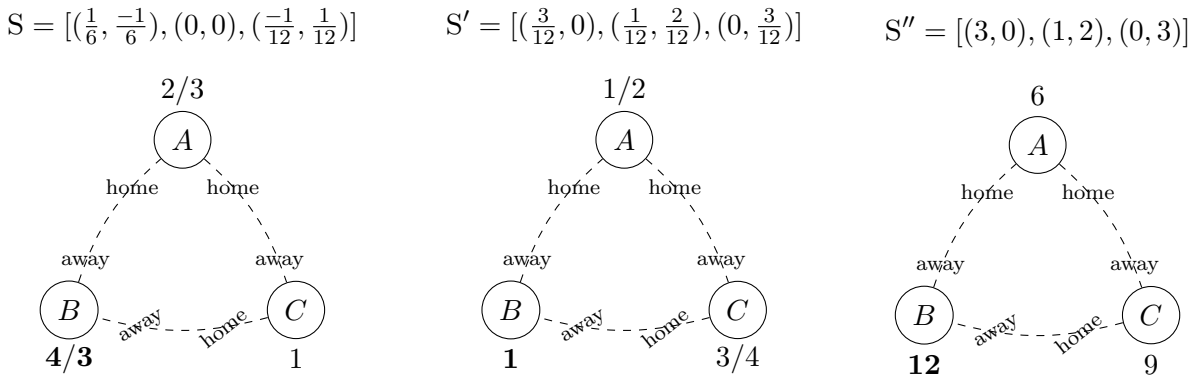
It is worth pointing out that the assumptions about the scoring system S stated in Theorem 1 are without loss of generality since any given instance of S-Elimination can be transformed into an equivalent instance that satisfies these assumptions. This observation follows from the *normalization scheme* of Kern and Paulusma (2004), which we recall below.

2.9 Normalization Scheme

The normalization scheme of Kern and Paulusma (2004) is a set of affine operations that can be used to transform a given instance of S-Elimination into an equivalent instance that satisfies the conditions of Theorem 1 such that the two instances have identical solutions. The scheme consists of the following three steps (see Section 4 of Kern and Paulusma, 2004):

1. *Discarding dominated outcomes*: Given $\mathrm{S} = [(\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots, (\alpha_t, \beta_t)]$ where $\alpha_i, \beta_i \in \mathbb{Q}$ for all $i \in [t]$ for some $t \in \mathbb{N}$, we can assume, without loss of generality, that no ordered pair (α_i, β_i) occurs more than once in S. Similarly, for any two distinct ordered pairs (α_i, β_i) and (α_j, β_j), we can assume, without loss of generality, that either (i) $\alpha_i < \alpha_j$ and $\beta_i > \beta_j$ or (ii) $\alpha_i > \alpha_j$ and $\beta_i < \beta_j$. This is because of the following reason: For any remaining game between teams other than i^*, if $\alpha_i \leq \alpha_j$ and $\beta_i \leq \beta_j$, then the outcome (α_i, β_i) is no worse than the outcome (α_j, β_j) from the viewpoint of i^*, and therefore it is safe to discard the latter. We will also assume, again without loss of generality, that S satisfies $\beta_1 < \beta_2 < \cdots < \beta_t$, which in turn implies that $\alpha_1 > \alpha_2 > \cdots > \alpha_t$.

2. *Translation step*: We now modify the given S-Elimination instance with $\mathrm{S} = [(\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots, (\alpha_t, \beta_t)]$ from step 1 to obtain an instance with



(a) *Given instance* **(b)** *After translation* **(c)** *After scaling*

Figure 4: *An illustration of the application of the normalization scheme. Subfigure (a) shows the given* S-Elimination *instance. Subfigures (b) and (c) show the same instance after the translation and scaling steps respectively. In each case, the score of the winning team is highlighted in boldface.*

$\mathrm{S}' = [(\alpha'_1, \beta'_1), (\alpha'_2, \beta'_2), \ldots, (\alpha'_t, \beta'_t)]$, where $\alpha'_i = \alpha_i - \alpha_t$ and $\beta'_i = \beta_i - \beta_1$ for all $i \in [t]$. That is, we use a *translation* operation to make all entries in S' non-negative. In order to preserve the equivalence of the two instances of S-Elimination before and after the translation, we need to compensate each team for the loss in score that it might incur in the remaining games due to the modified scoring system. We do this by awarding, for each remaining game in $\mathcal{G}$, α_t and β_1 points to each home and away team respectively. That is, for each $i \in [N] \setminus \{i^*\}$, $s'_i = s_i + H_i \cdot \alpha_t + A_i \cdot \beta_1$, where H_i and A_i are the number of games in $\mathcal{G}$ where i is the home and the away team respectively.

3. *Scaling step*: The final step in the normalization scheme consists of *scaling down* the modified scores of all teams $\mathbf{s}'$ and the entries of the modified scoring system S' by α_{t-1} (recall that $\alpha_{t-1} > 0$). This step is without loss of generality, since the new problem can be seen as measuring the score values in the units of α_{t-1}. The new instance of S-Elimination has team scores $\mathbf{s}''$, where $s''_i = s'_i/\alpha_{t-1}$ for each $i \in [N]$, and a new scoring system $\mathrm{S}'' = [(\alpha''_1, 0), (\alpha''_2, \beta''_2), \ldots, (1, \beta''_{t-1}), (0, \beta''_t)]$ where $\alpha''_i = \alpha'_i/\alpha_{t-1}$ for $1 \leq i \leq t-2$ and $\beta''_j = \beta'_j/\alpha_{t-1}$ for $2 \leq j \leq t$. Notice that if $\beta'_2 \geq \alpha_{t-1}$ (i.e., if $\beta''_2 \geq 1$), then S'' already satisfies the conditions of Theorem 1. On the other hand, if $\beta'_2 < \alpha_{t-1}$, then we can repeat the steps 1-3 of the normalization scheme by swapping the *home* and *away* labels for each remaining game and swapping the corresponding scoring system values (i.e., exchanging α_i's with β_i's) while keeping the individual team scores intact.

Example 3 illustrates the above procedure applied to the toy instance in Figure 4.

Example 3. *Consider the* S-Elimination *instance shown in Figure 4a. There are three teams A, B and C with initial scores $2/3$, $4/3$ and 1 respectively, and $\mathrm{S} = [(1/6, -1/6), (0, 0), (-1/12, 1/12)]$. Note that* S *does not satisfy the conditions required by Theorem 1 to begin with. Since* S *does not contain any dominated outcomes, we move to the translation step of the above scheme (step 2). For each remaining game, the home and away teams are awarded $-1/12$ and $-1/6$ points respectively, resulting in the new scoring system $\mathrm{S}' = [(3/12, 0), (1/12, 2/12), (0, 3/12)]$ and the updated team scores, as shown in Figure 4b. Finally, scaling down all teams scores and scoring system entries by $1/12$ (according to step 3) results in the* S-Elimination *instance shown in Figure 4c. The new scoring system $\mathrm{S}'' = [(3, 0), (1, 2), (0, 3)]$ indeed satisfies the conditions in Theorem 1.*

2.10 Some Useful Computational Problems

We will now describe some well-known computational problems that will find use in our reductions.

- *3-dimensional Matching:*

  The problem of 3-D Matching can informally be stated as follows: Suppose we are given the following three sets of equal cardinality: a set of men, a set of women and a set of pets. Suppose we are also given a set of compatible triples of men, women, and pets. Does there exist a set of compatible triples such that each man, woman and pet is chosen in exactly one triple?

3-D Matching	
Input:	A tuple $\langle W, X, Y, R\rangle$, where W, X, Y are disjoint sets each of size q and $R \subseteq W \times X \times Y$ is a set of triples.
Question:	Does there exist a subset $R' \subseteq R$ such that R' covers $W \cup X \cup Y$ exactly (i.e., each $w \in W$, $x \in X$, and $y \in Y$ is present in exactly one triple in R')?

  3-D Matching is known to be NP-complete even when each element of the sets W, X, Y occurs in at most three triples in R (Garey and Johnson, 1979, Problem SP1 on page 221).[6]

- *Capacitated Dominating Set:*

  Given a graph $G = (V, E)$, a set $V' \subseteq V$ is a *Capacitated Dominating Set* if there exists an assignment $f : V \setminus V' \to V'$ that maps each vertex in $V \setminus V'$ to a neighboring vertex in V' such that no vertex in V' is assigned to more neighbors than its capacity.

Capacitated Dominating Set	
Input:	A triple $\langle G, c, k\rangle$, where $G = (V, E)$ is a graph, $c : V \to \mathbb{N}$ is a capacity function for the vertices of G and k is a positive integer.
Question:	Does there exist a set of vertices $V' \subseteq V$ of size at most k such that each vertex $v \in V \setminus V'$ can be assigned to some adjacent vertex $v' \in V'$, and no vertex $v' \in V'$ is assigned more than $c(v')$ vertices?

  Capacitated Dominating Set is known to be W[1]-hard when simultaneously parameterized by the *treewidth* and the solution size k (Dom et al., 2008). In fact, the problem remains W[1]-hard when simultaneously parameterized by the *pathwidth* and the *feedback vertex set*, even on instances with only a constant number of distinct capacities.[7]

[6] A previous version of this paper (Vaish et al., 2016) incorrectly claimed that 3-D Matching continues to be NP-complete even when each element of W, X, Y occurs in exactly two triples in R. We fix this issue in this paper by proving the same results starting from the variant of 3-D Matching where each element of W, X, Y occurs in at most three triples in R—this variant is indeed known to be NP-complete (Garey and Johnson, 1979, Problem SP1 on page 221). The authors thank an anonymous reviewer for pointing this out.

[7] Dom et al. (2008) show W[1]-hardness of Capacitated Dominating Set via a reduction from Multicolored Clique on general graphs. By carrying out their reduction starting instead from Multicolored Clique on *regular* graphs—which is also W[1]-hard (Cygan et al., 2015)—it is easy to observe that the parameters *pathwidth*, *treewidth* and the size of a *minimum feedback vertex set* of the reduced Capacitated Dominating Set instance are all $\mathcal{O}(k^4)$ in size, where k is the solution size.

- *Partition:*

  Given a multiset S of natural numbers, PARTITION asks whether there exists a way to divide S into two disjoint parts with equal sums. For example, given $S = \{1, 1, 1, 2, 2, 3, 3, 5\}$, the sets $S_1 = \{1, 1, 1, 3, 3\}$ and $S_2 = \{2, 2, 5\}$ constitute a valid partition (i.e., $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = S$).

PARTITION	
Input:	A multiset $S = \{a_1, a_2, \ldots, a_n\}$ of n positive integers.
Question:	Does there exist a partition of S into two subsets S_1 and S_2 such that $\sum_{a_i \in S_1} a_i = \sum_{a_j \in S_2} a_j = \frac{1}{2} \sum_{a_k \in S} a_k$?

  PARTITION is known to be NP-complete (Garey and Johnson, 1979, Problem SP12 on page 223). We assume, without loss of generality, that $a_1 \leq a_2 \leq \cdots \leq a_n$.

- *Integer Linear Programming:*

  Given a pair of integer matrices $\langle \mathbf{A}, \mathbf{b} \rangle$ where $\mathbf{A} \in \mathbb{Z}^{p \times q}$ and $\mathbf{b} \in \mathbb{Z}^{q \times 1}$, INTEGER LINEAR PROGRAMMING asks whether there exists $\mathbf{x} \in \mathbb{Z}^{q \times 1}$ satisfying $\mathbf{A} \cdot \mathbf{x} \preccurlyeq \mathbf{b}$. Here $\preccurlyeq$ denotes the 'component-wise less than or equal to' relation.

INTEGER LINEAR PROGRAMMING	
Input:	A pair of matrices $\langle \mathbf{A}, \mathbf{b} \rangle$ where $\mathbf{A} \in \mathbb{Z}^{p \times q}$ and $\mathbf{b} \in \mathbb{Z}^{q \times 1}$ are integer matrices, and p, q are natural numbers.
Question:	Does $\mathbf{A} \cdot \mathbf{x} \preccurlyeq \mathbf{b}$ admit a solution $\mathbf{x} \in \mathbb{Z}^{q \times 1}$?

  INTEGER LINEAR PROGRAMMING is known to be FPT when parameterized by the number of variables q (Lenstra Jr, 1983).

3 Related Work

In this section, we review some of the related literature on other approaches for dealing with incomplete preferences and on the parameterized studies of the manipulation problem.

- *Possible and necessary winners*: A common approach for dealing with incomplete preferences in the computational social choice literature is by modifying the notion of an election winner to accommodate sets of winners. Given an incomplete preference profile and a standard (ranking-based) voting rule, the set of *possible/necessary* winners is the set of all candidates that can win the election under some/every completion of the preferences (Konczak and Lang, 2005; Walsh, 2007; Betzler and Dorn, 2010; Pini et al., 2011; Xia and Conitzer, 2011; Baumeister and Rothe, 2012; Gaspers et al., 2014). Manipulating a voting rule in this context means making a distinguished candidate a possible/necessary winner, given the (incomplete) preferences of the non-manipulators. A distinguishing feature of pairwise voting rules (such as pBorda and Copeland$^\alpha$) is that they deal with incomplete preferences *directly* without the need for extending each incomplete vote.

  Outside the realm of elections, the work of Aziz et al. (2015) on determining possible and necessary winners in *partial tournaments* closely relates to our formulation of the

manipulation problem. A partial tournament refers to an asymmetric directed graph that is not necessarily complete. The problem of finding a possible/necessary winner for such a graph involves checking whether a specific vertex can be made the winner (according to a given tournament solution) under some/every completion of this graph. In our framework, this corresponds to the manipulation problem with a single non-manipulator who compares some of the candidate pairs and skips the rest. The action space of the manipulator is precisely the set of pairs skipped by the non-manipulator. The question of whether the manipulator can come up with a vote that makes a distinguished candidate v win the election corresponds exactly to whether v is a possible winner in the partial tournament. This way, our model strictly generalizes that of partial tournaments. It is also easy to see that *weighted* partial tournaments—which are partially specified multigraphs—correspond to the coalitional manipulation question in our setting.

More specifically, the tournament solutions studied by Aziz et al. (2015) that are of direct relevance to our work are (a) Copeland winners for the unweighted setting, and (b) Borda winners for the weighted setting. For (a), Aziz et al. (2015) show that possible and necessary Copeland-winner problems are efficiently solvable for partial unweighted tournaments. In our framework, this corresponds to the constructive and destructive variants of Copeland^0-Manipulation problem with a single non-manipulator, where the action space of the manipulator is the set of all comparisons that are skipped by the non-manipulator and `pref-type` = total. This is clearly a special case of our results (Table 2), since we show that the tractability results extend to *any* given number of non-manipulators, *any* action space and *any* choice of `pref-type`.

For (b), Aziz et al. (2015) show that the problem of determining possible and necessary Borda-winners is efficiently solvable in the weighted setting. Their definition of a Borda-winner, however, corresponds to a candidate with the highest *outdegree* (i.e., a candidate that wins the maximum number of pairwise comparisons). As a result, a Borda-winner in their model might differ from a pBorda winner (Figure 2 in Section 2.1 is an example of this), which precludes a comparison between the two sets of results.

- *Modifying standard voting rules*: A growing body of work (Baumeister et al., 2012; Narodytska and Walsh, 2014; Fitzsimmons and Hemaspaandra, 2015; Menon and Larson, 2017) has studied modifications of standard voting rules to deal with partial preferences that are specified as either top-truncated ballots or rankings with ties. A top-truncated ballot refers to a partial ranking where each ranked candidate is assumed to be preferred over all the unranked ones. Voting rules that aggregate such preferences were studied by Narodytska and Walsh (2014) in the context of the manipulation problem. Their results were later generalized to a larger class of voting rules (Menon and Larson, 2017), to votes with ties (Fitzsimmons and Hemaspaandra, 2015), and to structured preferences such as single-peaked and nearly single-peaked domains (Menon and Larson, 2017). Nearly all the voting rules used in these studies are *positional*, meaning the score of a candidate depends on its *position* or *rank* in each vote. Such rules, however, cannot be immediately generalized to work with pairwise preferences since the position of a candidate in a vote is often not well-defined. An exception to such rules is the Copeland$^\alpha$ family of voting rules, for which the manipulation problem has been studied in context of weighted votes and a coalition of manipulators (Narodytska and Walsh, 2014; Fitzsimmons and Hemaspaandra, 2015; Menon and Larson, 2017). Even here, a direct comparison with our setting is not possible since these papers study a more general problem than ours (namely, coalitional manipulation with weighted votes) over a strict sub-domain of pairwise preferences (namely, top-truncated ballots).

- *Parameterized complexity of manipulation problems*: Parameterized complexity analysis has proven extremely useful in scrutinizing the computational behavior of a variety of

problems in computational social choice, such as

- *winner determination* (Bartholdi III et al., 1989a; Betzler et al., 2009a; Fellows et al., 2010; Betzler et al., 2010, 2013),
- *manipulation* (Conitzer et al., 2007; Betzler et al., 2011; Yang, 2014; Xia, 2014; Yang, 2015; Yang and Guo, 2016),
- *bribery* (Dorn and Schlotter, 2012; Wang et al., 2013; Bredereck et al., 2016), and
- *possible and necessary winner problems* (Betzler et al., 2009b; Xia and Conitzer, 2011; Baumeister et al., 2012).

We refer the reader to detailed surveys on this topic by Lindner and Rothe (2008) and Betzler et al. (2012).

Among the studies on the parameterized complexity of manipulation of standard voting rules, the works of Faliszewski et al. (2014) and Yang and Guo (2016) are closest to ours in spirit. Specifically, Faliszewski et al. (2014) study the computational complexity of the coalitional manipulation problem for weighted veto elections, when all votes are required to be complete rankings. This problem is known to be NP-complete in general (Hemaspaandra and Hemaspaandra, 2007) and efficiently solvable when preferences are single-peaked (Faliszewski et al., 2009a). Faliszewski et al. (2014) show that the tractability result in fact extends to various generalizations of single-peakedness, such as 'the number of voters that are not single-peaked' (called maverick-voters) or 'how many pairwise swaps away is each vote from being single-peaked'.

Similarly, Yang and Guo (2016) show that unweighted Borda manipulation with two manipulators is efficiently solvable over the domain of single-peaked preferences, although the problem is known to be NP-complete in the general case (Davies et al., 2014; Betzler et al., 2011).[8] This result was generalized by Yang (2015), who showed that the manipulation problems for the Borda and Copeland$^\alpha$ rules by two manipulators are FPT in the parameter *single-peaked width*.

4 Our Results and Techniques

This section presents our axiomatic (Section 4.1), classical complexity-theoretic (Section 4.2) and parameterized complexity-theoretic results (Section 4.3). We will start with axiomatic result, which shows that any *reasonable* pairwise voting rule is susceptible to manipulation. This is followed by our classical complexity-theoretic results, which show that computational complexity can provide a promising worst-case shield against the manipulation of the pBorda and Copeland$^\alpha$ rules. Finally, our parameterized complexity results provide a more refined understanding of such computational protections for the case of the pBorda rule.

4.1 Axiomatic Result

Recall that the Gibbard-Satterthwaite theorem (Gibbard, 1973; Satterthwaite, 1975) shows that any non-dictatorial voting rule over at least three candidates under which each candidate has some chance of winning is manipulable. In its standard form, this result requires the voter preferences over the set of candidates to be complete, transitive and unrestricted (i.e., voters are allowed to pick any possible ranking). Several follow-up works have since then looked into relaxing one or more of these assumptions: For instance, Sen (2001); Aswal et al. (2003); Sato (2010) and Pramanik (2015) relax *unrestrictedness* by studying the minimally rich subsets of

[8] Recall that Bartholdi III et al. (1989b) showed that the problem of unweighted Borda manipulation by a single manipulator is efficiently solvable over the domain of rankings.

the domain of all rankings over which the theorem continues to hold. Similarly, Reffgen (2011) and Pini et al. (2009) relax the *completeness* assumption by studying settings where only top-k choices or partial orders are specified. Extensions of this theorem to voting rules that pick a *set of winners* instead of a unique winner have also been studied (Duggan and Schwartz, 2000; Pini et al., 2009). Our axiomatic result can be seen as belonging to this line of research where we simultaneously relax the assumptions of transitivity and completeness while allowing for unrestrictedness (i.e., voters can pick any set of pairwise preferences).

Specifically, we show that even for the domain of pairwise preferences (a strict superset of the domain of rankings), no *reasonable* voting rule is non-manipulable. We might expect this intuitively—indeed, if r is a non-manipulable pairwise voting rule, then the *projection* of r to the domain of rankings (i.e., a standard voting rule that is consistent with r) will also be non-manipulable. However, the projection of a "non-dictatorial" pairwise voting rule r to the domain of rankings might still turn out to be dictatorial,[9] which makes the desired result elusive.

Theorem 2. *If there are at least three candidates and at least two voters, then any onto pairwise voting rule must be manipulable.*

Proof. The proof proceeds in four steps: Suppose, for the sake of contradiction, that a given onto pairwise voting rule r is non-manipulable.

1. Consider a pairwise preference profile Π_1 where each vote is a directed cycle of the form $n \succ n-1$, $n-1 \succ n-2$, $\ldots$, $2 \succ 1$, $1 \succ n$ (the remaining comparisons can be set arbitrarily). We require $n \geq 3$ for this to be well defined. Let candidate 1 be the winner chosen according to r for this profile (i.e., $r(\Pi_1) = 1$).

   (*Observe that when the true preferences of a voter form a directed cycle, then regardless of the outcome of the voting rule, there is always some other candidate that the voter would prefer over the current choice. For instance, in our construction, every voter prefers candidate* 2 *over the currently chosen candidate* 1.)

2. Starting from the voter u_1, sequentially modify the votes of all the voters $u_1, u_2, \ldots, u_m$ such that in each vote, candidate 2 beats all other candidates and candidate 1 beats everyone except for candidate 2. At each stage of this modification process, candidate 1 must remain the election winner (if candidate 2 wins at any stage, then the swing voter has an incentive to switch to the new vote; if some other candidate $i \neq 1, 2$ wins, then there is incentive for the swing voter to switch back to the old vote). Call this new profile Π_2. Hence $r(\Pi_2) = 1$.

   (*Notice that the profile* Π_1 *constructed in step 1 already highlights an unsatisfactory aspect of the voting rule* r*, which is that the chosen candidate is strictly less preferred to another candidate by every voter. The profile* Π_2 *constructed in this step further highlights this problem, in the sense that now there is a candidate that is unanimously preferred by every voter over* every *other candidate, and yet the voting rule fails to pick this candidate. As we will see in steps 3 and 4, along with ontoness, this anomaly of the voting rule will force the existence of a manipulation.*)

3. Ontoness of r implies that there exists a preference profile Π such that $r(\Pi) = 2$. By an argument similar to step 2 above, the profile Π can be transformed into a profile Π_3 where candidate 2 is preferred over every other candidate by each voter and it continues to be the winner (i.e., $r(\Pi_3) = 2$). Indeed, if at any point during the transformation from Π to Π_3, the outcome changes to a candidate $i \neq 2$, then the swing voter has an incentive to switch back to the old vote.

[9] An example of this is a pairwise voting rule which picks the favorite candidate of a fixed voter (i.e., a *dictator*) whenever *all* voters provide complete rankings, and picks a more 'democratic' option otherwise.

4. Starting from Π_3, sequentially modify the votes to transform it into Π_2. At some point during this process, candidate 2 must lose, providing the incentive for the swing voter to switch back to the old vote, and thus contradicting the non-manipulability of r. □

4.2 Classical Complexity Results

This section presents our results on the computational complexity of manipulation of pBorda (Section 4.2.1) and Copeland$^\alpha$ (Section 4.2.2) voting rules.

4.2.1 Complexity of Manipulating Pairwise Borda

Having established the impossibility result above, we study settings under which computational complexity can provide a worst-case barrier against manipulation in the setting of pairwise preferences. To this end, we build a comprehensive landscape of the computational complexity of pBorda-Manipulation for various combinations of possibilities that arise along the two dimensions mentioned earlier, namely (a) the structure of the *action space* (i.e., $\mathcal{A}$ = tree, bipartite, complete graph), and (b) the *preference type* (i.e., `pref-type` = total, acyclic, total+acyclic, unrestricted) .

Our results show that the manipulation problem turns out to be easy (i.e., polynomial-time solvable) whenever the graph structure is *simple enough* (e.g., $\mathcal{A}$ = a tree), regardless of the preference type. However, as we move toward more complex graph structures like bipartite or complete graphs, requiring either totality or acyclicity (but not both) can lead to computational intractability (i.e., NP-hardness). Stated differently, our results for the pBorda rule show that moving away from the standard framework of rankings by relaxing either completeness or transitivity or both is a promising direction for making the manipulation problem computationally hard.

Our results for pBorda-Manipulation are summarized in Table 1 in Section 1 and are stated as Theorems 3, 5, 7 and 8. We start with the easiness-of-manipulation result.

Theorem 3. pBorda-Manipulation *is efficiently solvable in the following cases: (a) for any choice of* `pref-type` *when* $\mathcal{A}$ *is a tree, and (b) for any choice of* $\mathcal{A}$ *when* `pref-type` = total+acyclic.

Proof. We provide separate proofs for the parts (a) and (b) of the theorem.

Case (a): When $\mathcal{A}$ *induces a tree.*

In this case, a manipulative vote (if it exists) can be efficiently computed by a bottom-up greedy algorithm that starts at the leaves of the tree and forces a *safe* option for the manipulator. For example, if a leaf node has a positive excess score, then it must lose the comparison with its parent in the manipulator's vote (and if this is not enough, then it is legitimate to abort). Similarly, if a leaf has negative excess, then it might as well win the comparison with its parent whenever the score transfer is feasible.

Formally, let $\mathcal{A}$ denote both the set of pairwise comparisons that the manipulator is allowed to make and the tree graph underlying this set. The manipulation algorithm consists of the following two steps:

1. Start from an arbitrary leaf node v in $\mathcal{A}$ (such a node always exists in a tree). Let w denote the parent of v in $\mathcal{A}$. If v can $\{\text{win}, \text{draw}, \text{lose}\}$ the pairwise comparison against w without gaining a positive excess score, in that order, then pick that option. Otherwise return NO.[10]

[10]Note that we can assume, without loss of generality, that the distinguished candidate i^* wins all its pairwise comparisons. That is, any winning vote for the manipulator can be transformed into another (possibly different) winning vote where i^* wins all its pairwise comparisons. This holds for any choice of `pref-type` $\in \{\text{total}, \text{acyclic}, \text{total+acyclic}, \text{unrestricted}\}$. Therefore, i^* is not a part of the action space, and the notion of excess score in Step 1 is well-defined.

2. Remove v and the edge vw from $\mathcal{A}$ and repeat step 1 starting from another leaf node. This process continues until all comparisons in $\mathcal{A}$ have been considered.

We will now discuss the computational complexity and correctness of the above algorithm.

Running Time: The algorithm involves $\mathcal{O}(\log m)$ computation at each edge, thus $\mathcal{O}(|\mathcal{A}| \log m)$ runtime overall.

Correctness: First, we claim that if our algorithm outputs a vote, then it must be valid. Indeed, the excess score constraint for each node (or candidate) is satisfied at each step. Furthermore, the tree structure of the action space ensures that the vote is always acyclic. Finally, Step 1 maintains consistency with the 'total' constraint of `pref-type`.

Next, we will argue that converse, i.e., if there exists a valid vote $\succ_{u_m}$ for the manipulator, then our algorithm must also return a valid vote, possibly different from $\succ_{u_m}$. We will argue this by contradiction. Suppose our algorithm terminates with a NO output. Consider an alternate algorithm that examines pairwise comparisons in the order in which our algorithm examines them, but makes choices consistent with $\succ_{u_m}$. Clearly the alternate algorithm will finish its run (instead of terminating prematurely like our algorithm) and output the valid vote $\succ_{u_m}$. Now consider running the two algorithms in parallel. At each step during the two runs, the excess score of any remaining node in our algorithm can only be less than or equal to the corresponding excess score under the alternate algorithm (this follows from the greedy semantics of our algorithm). Therefore, there is no reason for our algorithm to terminate with a NO output if the alternate algorithm finishes its run on the same input, providing the desired contradiction.

Case (b): When `pref-type` = total+acyclic.

In this case, the manipulator is required to orient every edge of a given undirected graph $\mathcal{A}$ such that (a) there are no directed cycles, and (b) the resulting vote helps the distinguished candidate i^* win the election. Our algorithm for this problem (Algorithm 1) relies on the fact that any directed acyclic graph must admit a topological ordering, and, as a result, there must exist a *source* vertex (i.e., one that wins all its pairwise comparisons in the manipulator's vote). At each step, the algorithm iterates over all the remaining vertices to check if any of them can be made the source. If yes, the algorithm orients all its adjacent edges as outgoing, and proceeds with a smaller graph with this vertex and all its adjacent edges removed. If, on the other hand, no remaining vertex can be made the source, the algorithm terminates with a NO output.

We remark that our algorithm is a straightforward extension of the greedy algorithm of Bartholdi III et al. (1989b) to general action spaces. Indeed, when the action space $\mathcal{A}$ is a complete graph and `pref-type` = total+acyclic, then the manipulator is effectively required to construct a ranking of the candidates (as is the case with the manipulation of standard voting rules), and we recover the greedy procedure of Bartholdi III et al. (1989b).

Running Time: In each iteration, the algorithm makes $\mathcal{O}(n)$ checks for the source vertex, and there are at most $(n-1)$ iterations overall. Each check requires $\mathcal{O}(n \log m)$ computation. Therefore, the algorithm runs in $\mathcal{O}(n^3 \log m)$ time.

Correctness: If Algorithm 1 returns a vote, then it must be valid because (a) all candidate pairs are compared (hence *total*), (b) there are no directed cycles (hence *acyclic*), and (c) all excess score constraints are satisfied at each step of the algorithm. We will now argue that the converse is also true by showing that the algorithm terminates with a NO output only when there does not exist a valid vote. Suppose, for the sake of contradiction, that there exists a valid manipulative vote $\succ_{u_m}$ but Algorithm 1 terminates with a NO output. The only way this happens is if, at the time of termination, none of the remaining candidates could have been made a source vertex without gaining positive excess. Let $S \subseteq [n] \setminus i^*$ denote the set of remaining candidates in $\mathcal{A}$ at the time of termination of the algorithm, and let $\succ_{\text{partial}}$ denote the partial orientation constructed by the algorithm before it terminates. Thus, if $\mathcal{A}_S = \{\{v, w\} \in \mathcal{A} : v \in S \text{ and } w \in S\}$ denotes the restriction of the action space $\mathcal{A}$ to the candidates in S, then $\succ_{\text{partial}}$ corresponds to the orientation (or vote) constructed by the algorithm

Algorithm 1: Greedy Manipulation

input : A tuple $\langle \Pi, i^*, \mathcal{A} \rangle$.
output: A total and acyclic vote $\succ_{u_m} \in \mathcal{R}$ over $\mathcal{A}$.

1 Start by making i^* win all its pairwise comparisons (and remove all the edges oriented this way from $\mathcal{A}$). Update excess scores if required.
2 **while** *some vertex v can be made a source vertex without gaining positive excess* **do**
3 Orient all incident edges of v as outgoing and update excess scores for all vertices.
4 Remove the vertex v and the edges oriented above from further consideration.
5 **if** *all edges in $\mathcal{A}$ have been oriented* **then**
6 **return** YES and output the oriented graph.
7 Repeat with the smaller action space.
8 **return** NO.

for candidate pairs in $\mathcal{A} \setminus \mathcal{A}_S$.

Now consider a different vote $\succ'$ obtained by combining $\succ_{\text{partial}}$ with the vote $\succ_{u_m}$ restricted to the candidate pairs in $\mathcal{A}_S$. In other words, $\succ'$ resembles $\succ_{\text{partial}}$ for candidate pairs in $\mathcal{A} \setminus \mathcal{A}_S$, and $\succ_{u_m}$ for candidate pairs in $\mathcal{A}_S$, and is therefore well-defined.

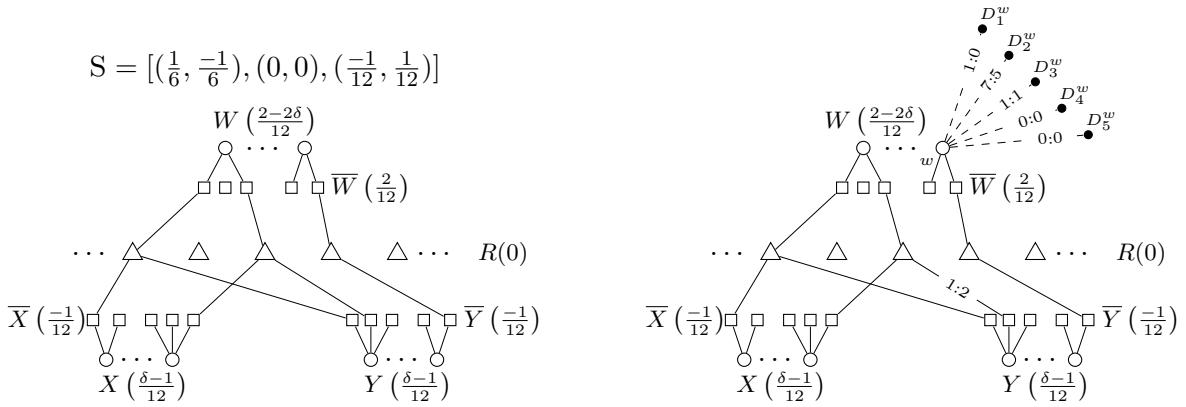
We will now prove a key claim, which, along with the observation made in the next paragraph, will provide the desired contradiction. First, we claim that $\succ'$ *constitutes a valid manipulative vote*. The reason for this is as follows: Any vertex $v \in \mathcal{A}$ is such that either $v \in S$ or $v \notin S$. If $v \notin S$, then, by construction, its excess score under the vote $\succ'$ depends only on the $\succ_{\text{partial}}$ component. This means that v must have been made a source vertex at some stage of the algorithm without it gaining positive excess. Therefore, the excess score constraint for any $v \notin S$ is satisfied under $\succ'$. Now consider the case when $v \in S$. It is easy to observe that the excess score of v under $\succ'$ is at most its excess score under the valid vote $\succ_{u_m}$. Indeed, by construction of $\succ'$, v must lose all its pairwise comparisons in $\mathcal{A} \setminus \mathcal{A}_S$ under $\succ'$, whereas under $\succ_{u_m}$, v can possibly win one or more of the comparisons in $\mathcal{A} \setminus \mathcal{A}_S$, resulting in a possibly greater excess score. Moreover, since $\succ_{u_m}$ is a valid vote, the excess score of v under $\succ_{u_m}$ must be non-positive, and by extension, the same holds for the vote $\succ'$. Thus, the excess score of all vertices in $\mathcal{A}$ under $\succ'$ is zero or less, making it a feasible vote.

The final step in the proof is to observe that the existence of a feasible vote $\succ'$ contradicts the assumption that our algorithm terminates prematurely with a NO output. Indeed, feasibility of $\succ'$ demonstrates that there exists a valid orientation for the remaining vertices in S at the time of termination (namely, $\succ_{u_m}$ restricted to the candidate pairs in $\mathcal{A}_S$), and therefore the check in Line 2 of Algorithm 1 cannot fail. This finishes the proof of correctness of Algorithm 1. □

Remark 1. (Locality Property) The proofs for both parts (a) and (b) of Theorem 3 rely only on a *locality* property of score-based pairwise voting rules. This property requires that adding a pairwise comparison between a pair of candidates only affects the scores of the two candidates involved, while the scores of all the other candidates are unaffected. For this reason, other score-based pairwise voting rules with this locality property such as Copeland$^\alpha$ also admit polynomial time manipulation algorithms in the settings described above (refer to Tables 2 to 4 in Section 1). We state this observation as Corollary 4.

Corollary 4. Copeland$^\alpha$-Manipulation *for all rational $\alpha \in [0, 1]$ is efficiently solvable in the following cases: (a) for any choice of* `pref-type` *when $\mathcal{A}$ is a tree, and (b) for any choice of $\mathcal{A}$ when* `pref-type` = total+acyclic.

We now turn to the results about the computational hardness of manipulation. Our first result states that manipulating the pBorda rule can be computationally intractable, even when



(a) S-Elimination *instance* **(b)** pBorda-Manipulation *instance*

Figure 5: *Subfigure (a) shows the instance of* S-Elimination *used in the proof of Theorem 5. This instance is identical (up to normalization) to the one originally constructed by Kern and Paulusma (2004) in the context of showing NP-completeness of* S-Elimination *via reduction from* 3-D Matching. *The instance consists of seven groups of teams—namely, $W, \overline{W}, R, \overline{X}, \overline{Y}, X$ and Y—represented as circles, squares and triangles. The solid edges stand for the set of games that are still to be played. The scoring system is* $\mathrm{S} = [(\frac{1}{6}, \frac{-1}{6}), (0,0), (\frac{-1}{12}, \frac{1}{12})]$. *All teams in each group have the same score, as mentioned inside parentheses. Subfigure (b) shows the corresponding* pBorda-Manipulation *instance, where the circles, squares and triangles now correspond to the candidates, and the solid edges denote the action space of the manipulator. An edge label "a:b" for an edge connecting the vertices i and j means that i and j are in a:b configuration. The dummy candidates that interact with a candidate $w \in W$ are shown as solid circles, and the dashed edges represent the votes of the non-manipulators for these pairs. Notice that the dashed edges do not belong to the action space $\mathcal{A}$. The excess* pBorda *scores of the candidates in each group are mentioned inside parentheses.*

the action space of the manipulator is a bipartite graph of *maximum degree* three.

Theorem 5. pBorda-Manipulation *is* NP-complete *when* $\mathcal{A} \in \{\text{bipartite, general graph}\}$ *has maximum degree three and* `pref-type` $\in \{\text{acyclic, unrestricted}\}$.

Proof. The problem is clearly in NP. We show NP-hardness via reduction from S-Elimination with $\mathrm{S} = [(1/6, -1/6), (0,0), (-1/12, 1/12)]$. Recall that an instance of S-Elimination is defined by a tuple $\langle \mathbf{s}, i^*, \mathcal{G} \rangle$, where $\mathbf{s} = (s_1, s_2, \ldots, s_N)^T$ is the vector of current scores of the N teams, $i^* \in [N]$ is the distinguished team, and $\mathcal{G} \subseteq \binom{[N]}{2}$ is the set of remaining games between the teams.

Instead of showing a reduction from a general instance of S-Elimination, we will assume that the instance has the following additional properties:

- The S-Elimination instance consists of seven groups of teams comprising $[N]$, namely $W, \overline{W}, R, \overline{X}, \overline{Y}, X$ and Y (represented as circles, squares and triangles in Figure 5a) as well as the distinguished team i^* (not shown in the figure). We also have that $|W| = |X| = |Y|$.
- The set of remaining games $\mathcal{G}$ for each team is as follows:
  - each team in W (respectively X and Y) has at least one and at most three remaining games against the teams in $\overline{W}$ (respectively $\overline{X}$ and $\overline{Y}$),[11]

[11] This assumption, while not necessarily true for the S-Elimination instance originally constructed by Kern

  - each team in $\overline{W}$ (respectively $\overline{X}$ and $\overline{Y}$) has two remaining games—one apiece against the teams in R and W (respectively X and Y),
  - each team in R has three remaining games—one apiece against the teams in $\overline{W}, \overline{X}$ and $\overline{Y}$, and
  - the distinguished team i^* has no remaining games.

- All teams in a group have the same score, and therefore the same excess score. The current scores of all teams in each group are as follows: $W\,((2-2\delta)/12)$, $\overline{W}(2/12)$, $R(0)$, $\overline{X}(-1/12)$, $\overline{Y}(-1/12)$, $X\,((\delta-1)/12)$ and $Y\,((\delta-1)/12)$, where δ denotes the degree function of the graph $\mathcal{G}$. The distinguished team i^* has an initial score of 0.

- The scoring system is $\mathrm{S} = [(\frac{1}{6}, \frac{-1}{6}), (0,0), (\frac{-1}{12}, \frac{1}{12})]$.

- Finally, for each remaining game in $\mathcal{G}$, the team from $\overline{W}, \overline{X}$ or $\overline{Y}$ should be considered the *away* team.

We remark that the above structural assumptions about the original S-Elimination instance are without loss of generality, since the problem continues to be NP-complete even under the assumed conditions. This is simply because the above properties hold for the instance of S-Elimination constructed by Kern and Paulusma (2004) in their proof for showing the NP-completeness of S-Elimination via redution from 3-D Matching.[12] The advantage of these additional assumptions, of course, will be in simplifying the presentation of several of our proofs, while also lending structural insights into the complexity of pBorda-Manipulation problem.

The reduced pBorda-Manipulation instance $\langle \Pi, i^*, \mathcal{A}, \texttt{pref-type} \rangle$ is constructed as follows:

- The *set of candidates* $[n]$ consists of
  - a candidate i for each team $i \in [N]$, including a distinguished candidate i^* corresponding to the distinguished team i^*, and
  - five dummy candidates $D_1^i, \ldots, D_5^i$ for each candidate $i \in [n]$. Thus, there are $n = N + 5 \cdot N = 6N$ candidates overall.

- The *action space* $\mathcal{A}$ of the manipulator corresponds to the set of the remaining games $\mathcal{G}$ between the corresponding pairs of teams.

- The *set of non-manipulators* consists of twelve voters $u_1, \ldots, u_{12}$. Their votes are set up as follows:
  - *votes between candidate pairs* (i,j) *in* $\mathcal{A}$: For each game $(i,j) \in \mathcal{G}$ where i is the home team and j is the away team, the corresponding candidate pair $(i,j) \in \mathcal{A}$ is in a 1:2 configuration,[13]
  - *votes between the distinguished candidate* i^* *and the dummy candidates*: The pairs $(i^*, D_1^{i^*})$, $(i^*, D_2^{i^*})$, $(i^*, D_3^{i^*})$, $(i^*, D_4^{i^*})$ and $(i^*, D_5^{i^*})$ are set up in the configuration 1:0, 5:7, 1:1, 1:1 and 1:1 respectively, and

and Paulusma (2004), can be justified by starting their reduction from the variant of 3-D Matching where each base element occurs in at least one and at most three triples. This variant is known to be NP-complete (Garey and Johnson, 1979, Problem SP1 on page 221).

[12] Strictly speaking, the instance used by Kern and Paulusma (2004) is the *normalized* version of the instance used in our proof (refer to the normalization scheme described in Section 2.9). Nevertheless, we choose to work with the unnormalized instance in the interest of ease of demonstrating the connection with the manipulation problem. We remark that this is without loss of generality, since our unnormalized instance can be obtained from Kern and Paulusma's normalized instance by carrying out the translation and scaling operations in the reverse order.

[13] Recall from Section 2.3 that a vote configuration of 1:2 between a pair of candidates (i,j) means that one voter (say u_1) votes $i \succ_{u_1} j$ while two other voters (say u_2 and u_3) vote $j \succ_{u_2} i$ and $j \succ_{u_3} i$.

	$\mathcal{A}$	D_1	D_2	D_3	D_4	D_5	Excess score
$W_{\delta=1}$	$\frac{1}{3}$	1	$\frac{7}{12}$	$\frac{1}{2}$	$\frac{1}{2}$	0	0
$W_{\delta=2}$	$\frac{2}{3}$	1	$\frac{7}{12}$	$\frac{1}{2}$	0	0	$-\frac{2}{12}$
$W_{\delta=3}$	1	1	$\frac{7}{12}$	0	0	0	$-\frac{4}{12}$
$\overline{W}$	$\frac{4}{3}$	0	$\frac{3}{12}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{2}{12}$
R	1	0	$\frac{5}{12}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	0
$\overline{X} \cup \overline{Y}$	$\frac{4}{3}$	0	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{12}$
$X_{\delta=1} \cup Y_{\delta=1}$	$\frac{1}{3}$	1	$\frac{4}{12}$	$\frac{5}{12}$	$\frac{5}{12}$	$\frac{5}{12}$	0
$X_{\delta=2} \cup Y_{\delta=2}$	$\frac{2}{3}$	1	$\frac{4}{12}$	$\frac{5}{12}$	$\frac{5}{12}$	$\frac{2}{12}$	$\frac{1}{12}$
$X_{\delta=3} \cup Y_{\delta=3}$	1	1	$\frac{4}{12}$	$\frac{5}{12}$	$\frac{2}{12}$	$\frac{2}{12}$	$\frac{2}{12}$
i^*	0	1	$\frac{5}{12}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	0

Table 6: *This table shows the* pBorda *scores acquired by various candidates in our construction in the proof of Theorem 5. Each entry of the table indicates the* pBorda *score acquired by a row candidate due to its interaction with the set of candidates in each column. The rightmost column shows the excess scores.*

- *votes between non-distinguished candidates in $\mathcal{A}$ and the dummy candidates*:
  - for each candidate $w \in W$ such that $\delta(w) = 1$, the pairs (w, D_1^w), (w, D_2^w), (w, D_3^w), (w, D_4^w) and (w, D_5^w) are set up in the configuration 1:0, 7:5, 1:1, 1:1 and 0:0 respectively; for each candidate $w \in W$ such that $\delta(w) = 2$, the pairs (w, D_1^w), (w, D_2^w), (w, D_3^w), (w, D_4^w) and (w, D_5^w) are set up in the configuration 1:0, 7:5, 1:1, 0:0 and 0:0 respectively; for each candidate $w \in W$ such that $\delta(w) = 3$, the pairs (w, D_1^w), (w, D_2^w), (w, D_3^w), (w, D_4^w) and (w, D_5^w) are set up in the configuration 1:0, 7:5, 0:0, 0:0 and 0:0 respectively,
  - for each candidate $\overline{w} \in \overline{W}$, the pairs $(\overline{w}, D_1^{\overline{w}})$, $(\overline{w}, D_2^{\overline{w}})$, $(\overline{w}, D_3^{\overline{w}})$, $(\overline{w}, D_4^{\overline{w}})$ and $(\overline{w}, D_5^{\overline{w}})$ are set up in the configuration 0:0, 3:9, 1:1, 1:1 and 1:1 respectively,
  - for each candidate $r \in R$, the pairs (r, D_1^r), (r, D_2^r), (r, D_3^r), (r, D_4^r) and (r, D_5^r) are set up in the configuration 0:0, 5:7, 1:1, 1:1 and 1:1 respectively,
  - for each candidate $\overline{x} \in \overline{X}$, the pairs $(\overline{x}, D_1^{\overline{x}})$, $(\overline{x}, D_2^{\overline{x}})$, $(\overline{x}, D_3^{\overline{x}})$, $(\overline{x}, D_4^{\overline{x}})$ and $(\overline{x}, D_5^{\overline{x}})$ are set up in the configuration 0:0, 0:0, 1:1, 1:1 and 1:1 respectively (the votes involving the candidates in the set $\overline{Y}$ are defined analogously), and
  - for each candidate $x \in X$ such that $\delta(x) = 1$, the pairs (x, D_1^x), (x, D_2^x), (x, D_3^x), (x, D_4^x) and (x, D_5^x) are set up in the configuration 1:0, 4:8, 5:7, 5:7 and 5:7 respectively; for each candidate $x \in X$ such that $\delta(x) = 2$, the pairs (x, D_1^x), (x, D_2^x), (x, D_3^x), (x, D_4^x) and (x, D_5^x) are set up in the configuration 1:0, 4:8, 5:7, 5:7 and 2:10 respectively; for each candidate $x \in X$ such that $\delta(x) = 3$, the pairs (x, D_1^x), (x, D_2^x), (x, D_3^x), (x, D_4^x) and (x, D_5^x) are set up in the configuration 1:0, 4:8, 5:7, 2:10 and 2:10 respectively (the votes involving the candidates in the set Y are defined analogously).

- Finally, `pref-type` is set to either acyclic or unrestricted.

It is easy to check that the excess pBorda scores of the candidates exactly match the excess scores of the corresponding teams in the sports instance (refer to Table 6). Furthermore, for each candidate pair $(i, j) \in \mathcal{A}$ set up in a 1:2 configuration, the pBorda scores of i and j can change

by $(+1/6, -1/6)$, $(0, 0)$ or $(-1/12, +1/12)$ depending on whether the manipulator votes $i \succ j$, *skip* or $j \succ i$ respectively. This set of score exchanges is exactly the set of scoring outcomes prescribed by S in the S-Elimination instance. Therefore, an assignment of remaining games in the original instance is a winning one if and only if the corresponding vote in the reduced instance is a winning one as well.

It remains to be shown that any winning vote in the reduced instance is *acyclic* without loss of generality. To show this, we will argue that any winning assignment in the original instance is, without loss of generality, acyclic. Indeed, observe that any team $w \in W$ in the S-Elimination instance can win at most one of its remaining games against the teams in $\overline{W}$ without ending up with positive excess. Therefore, we will assume, without loss of generality, that w wins exactly one of its remaining games and plays a draw in the other(s). Notice that this already rules out w from being part of a directed cycle in the eventual assignment. Next, if a team $\overline{w} \in \overline{W}$ loses the game against $w \in W$, then, without loss of generality, it plays a draw with the team in R in its other remaining game, otherwise it loses the latter game. Similarly, any team $r \in R$ that wins the game against the team in $\overline{W}$ must lose both of its remaining games against the teams in $\overline{X}$ and $\overline{Y}$, otherwise it must play a draw in both these games. The latter case rules out the possibility of a directed cycle, so we focus on the former. In this case, each $\overline{x} \in \overline{X}$ (or $\overline{y} \in \overline{Y}$) that wins its game against the team in R can, without loss of generality, play a draw in its other remaining game against the team in X (or Y). In summary, it is without loss of generality that any winning assignment in the original instance is acyclic, implying the same for the reduced election instance.

Finally, notice that the graph $\mathcal{G}$ in the original instance (Figure 5a) is *bipartite* and has *maximum degree* three, implying the same for the action space $\mathcal{A}$ in the reduced instance. Therefore, pBorda-Manipulation is NP-complete even when $\mathcal{A}$ is bipartite and has maximum degree three, and when `pref-type` = acyclic. □

Remark 2. (Parameterized implication) Since the above reduction requires at most 12 non-manipulators, we have that pBorda-Manipulation is para-NP-complete when parameterized by the number of non-manipulators.

Remark 3. (Parameterized implication) Note that the reduced election instance has *maximum degree* $\Delta = 3$, and therefore from Definition 1, *diversity* $d \leq 3$. Hence, pBorda-Manipulation is para-NP-complete when simultaneously parameterized by the *diversity* (d) and *maximum degree* (Δ) parameters.

Remark 4. (Technical observation) Observe that regardless of how the manipulator chooses to vote, the smallest positive excess score that any candidate can achieve in the reduced instance is always at least $1/12$. Thus, pBorda-Manipulation continues to be NP-complete even when (a) the smallest possible positive excess score that any candidate can have is at least $1/12$, and (b) there are at most twelve non-manipulators in the election. This feature of the reduced instance will be useful later in the proof of Theorem 8.

Remark 5. (When `pref-type` = transitive) It can be shown that the above NP-completeness result for `pref-type` = acyclic extends to the case where `pref-type` = transitive. Recall that transitivity requires that $i \succ k$ whenever $i \succ j$ and $j \succ k$. Since the action space in our reduced instance is bipartite (i.e., consists of no odd cycles), the transitivity condition is vacuously satisfied. As a result, pBorda-Manipulation continues to be NP-complete even when $\mathcal{A}$ = bipartite and `pref-type` = transitive. The requirement for transitivity becomes interesting only when the action space consists of cycles of size three, such as when $\mathcal{A}$ is a complete graph. As we will argue later in Remark 8, pBorda-Manipulation remains NP-complete even for this case.

Our next result (Theorem 7) shows that pBorda-Manipulation is NP-complete when the manipulator is not allowed to skip any pairwise comparison in $\mathcal{A}$ (i.e., `pref-type` = total).

Our proof consists of two parts: First, we introduce a generalization of the S-Elimination problem (which we call Mixed-Elimination) and show that the problem is NP-complete via reduction from 3-D Matching (Lemma 6). Next, we show that Mixed-Elimination reduces to pBorda-Manipulation when `pref-type` = total, providing the desired hardness result. We start with a description of Mixed-Elimination.

Mixed-Elimination: Mixed-Elimination is a generalization of S-Elimination for competitions in which different games can be played under different scoring systems. That is, instead of a single scoring system S as in S-Elimination, Mixed-Elimination consists of k different scoring systems $\mathrm{S}_1, \mathrm{S}_2, \ldots, \mathrm{S}_k$ and an assignment function η which maps each game in $\mathcal{G}$ to exactly one of these scoring systems. The question of interest, once again, is: *Can team i^* still win the competition?*

Mixed-Elimination$(\mathrm{S}_1, \mathrm{S}_2, \ldots, \mathrm{S}_k)$

Input: A tuple $\langle \mathbf{s}, i^*, \mathcal{G}, \eta \rangle$, where $\mathbf{s} = (s_1, s_2, \ldots, s_N)^T$ is the vector of current scores of the N teams, $i^* \in [N]$ is the distinguished team, $\mathcal{G} \subseteq \binom{[N]}{2}$ is the set of remaining games between the teams, and η is a function that assigns a scoring system from $\{\mathrm{S}_1, \mathrm{S}_2, \ldots, \mathrm{S}_k\}$ to each game in $\mathcal{G}$.

Question: Does there exist an assignment of outcomes for the games in $\mathcal{G}$ such that i^* ends up with the (joint) highest overall score among all teams when each game is scored according to the scoring system chosen by η?

Analogous to S-Elimination, we will assume for Mixed-Elimination that the distinguished team i^* has no remaining games.

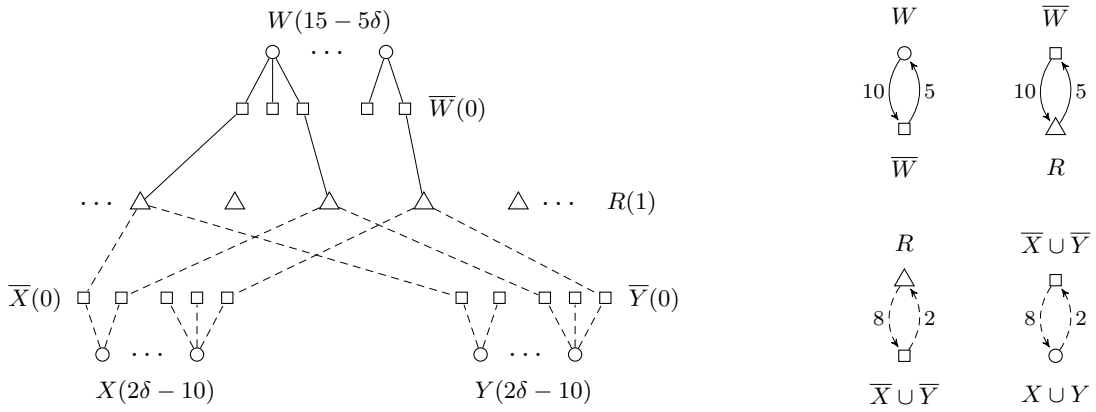
Our proof of Theorem 7 focuses on Mixed-Elimination$(\mathrm{S}_1, \mathrm{S}_2)$ where $\mathrm{S}_1 = [(10, -10), (-5, 5)]$ and $\mathrm{S}_2 = [(8, -8), (-2, 2)]$. That is, each remaining game is scored according to either S_1 or S_2 and no game can end in a draw. We already know from Theorem 1 that the *pure* elimination problems with either of these scoring systems—namely S_1-Elimination or S_2-Elimination—admit polynomial time algorithms.[14] By contrast, the *mixed* elimination problem with these two scoring systems together turns out to be computationally hard (Lemma 6).

Lemma 6. Mixed-Elimination$(\mathrm{S}_1, \mathrm{S}_2)$ *with* $\mathrm{S}_1 = [(10, -10), (-5, 5)]$ *and* $\mathrm{S}_2 = [(8, -8), (-2, 2)]$ *is* NP-complete *even if the set of remaining games $\mathcal{G}$ induces a bipartite graph.*

Proof. Membership in NP is clear. We show a polynomial time reduction from a variant of 3-D Matching where each base element occurs in at most three triples. This variant is known to be NP-complete (Garey and Johnson, 1979, Problem SP1 on page 221).

Our proof follows the template of a similar proof by Kern and Paulusma (2004), who show the hardness of S-Elimination via a reduction from the standard version of 3-D Matching (i.e., they do not make the additional assumption that each base element occurs in at most three triples). Specifically, Kern and Paulusma (2004) represent any instance of 3-D Matching as a *matching graph* (see Figure 6a). Each vertex of the matching graph belongs either to one of the base sets W, X or Y, or the set of triples R, or one of the sets $\overline{W}, \overline{X}, \overline{Y}$ that encode the

[14] This is because any instance of S-Elimination with $\mathrm{S} = \mathrm{S}_1$ (or $\mathrm{S} = \mathrm{S}_2$) can be transformed into an equivalent (efficiently solvable) instance of S-Elimination with $\mathrm{S} = [(1, 0), (0, 1)]$ via the normalization scheme described in Section 2.9.



(a) *Matching graph* **(b)** *Score transfers*

Figure 6: *Subfigure (a) shows the matching graph for a given* 3-D Matching *instance. The base sets W, X and Y and the set of triples R are represented as circles and triangles respectively. Each member of W (respectively X, Y) is connected to at most three members of the set $\overline{W}$ (respectively $\overline{X}, \overline{Y}$), shown as squares. Each such square represents an occurrence of the member of W (or X or Y) in the set R. The excess scores of the teams represented by each set in the* Mixed-Elimination *instance are mentioned in parentheses. The solid and dashed edges denote the games scored according to $\mathrm{S}_1 = [(10, -10), (-5, 5)]$ and $\mathrm{S}_2 = [(8, -8), (-2, 2)]$ respectively. Subfigure (b) shows all possible score transfers between pairs of teams from different sets. Thus, for instance, for a game between a team in W (away) and a team in $\overline{W}$ (home), a home win results in the team in W losing 10 points while the team in $\overline{W}$ gains the same amount. Similarly, an away win results in a transfer of 5 points in the opposite direction.*

occurrences of the elements of the base sets W, X and Y in the set of triples R. Each vertex in W is connected via an edge to each of the k vertices $\overline{w}_1, \ldots, \overline{w}_k$ in $\overline{W}$ that represent the k distinct occurrences of w in the set of triples R. The vertices in the sets X and Y are similarly connected to the vertices in $\overline{X}$ and $\overline{Y}$ respectively. In addition, each vertex $r \in R$ representing a triple $r = (w, x, y)$ is connected to three vertices—one each in $\overline{W}, \overline{X}$ and $\overline{Y}$—that are in turn adjacent to the base elements w, x and y respectively. Under the additional assumption about the variant of 3-D Matching mentioned above, it is easy to see that each base element in the sets W, X and Y is adjacent to at least one and at most three vertices in the sets $\overline{W}$, $\overline{X}$ and $\overline{Y}$ respectively.

The main idea in Kern and Paulusma's proof is to think of the *vertices* and the *edges* in the matching graph respectively as *teams* and the set of *remaining games* between them. We use this idea to set up an instance of Mixed-Elimination$(\mathrm{S}_1, \mathrm{S}_2)$ in which the games in the *upper half* of the matching graph are scored according to $\mathrm{S}_1 = [(10, -10), (-5, 5)]$ and those in the *lower half* are scored according to $\mathrm{S}_2 = [(8, -8), (-2, 2)]$, shown as solid and dashed edges respectively in Figure 6a.

Construction of the reduced instance: Formally, given an instance $\langle W, X, Y, R\rangle$ of 3-D Matching, we construct an instance $\langle \mathbf{s}, i^*, \mathcal{G}, \eta\rangle$ of Mixed-Elimination$(\mathrm{S}_1, \mathrm{S}_2)$ as follows:

- the *set of teams* $[N]$ is the union of the sets $W, X, Y, \overline{W}, \overline{X}, \overline{Y}$ and R (i.e., a team for every vertex in the matching graph) along with the distinguished team i^*,
- the *set of remaining games* $\mathcal{G}$ is precisely the set of all solid and dashed edges in the matching graph,
- the *assignment function* η is such that all games involving teams from $\overline{W}$ are scored according to S_1, while all other games in $\mathcal{G}$ are scored according to S_2. Furthermore, for

any game, a team *higher up* in the matching graph should be considered the *away* team, and

- the *current score* of the team i^* is fixed at 0. The group-wise excess scores of the teams are: $W(15-5\delta)$, $\overline{W}(0)$, $R(1)$, $\overline{X}(0)$, $\overline{Y}(0)$, $X(2\delta-10)$ and $Y(2\delta-10)$.

Equivalence of solutions: ($\Rightarrow$) Suppose there exists a solution $R' \subseteq R$ to the 3-D Matching instance. Then, a winning assignment can be constructed as follows:

- Each team $r \in R'$ where $r = (w, x, y)$ beats the team $\overline{w} \in \overline{W}$ corresponding to the element w, and loses against the teams $\overline{x} \in \overline{X}$ and $\overline{y} \in \overline{X}$ corresponding to the elements x and y respectively. This leaves the team r with an excess of -5.
- Each team $r \in R \setminus R'$ where $r = (w, x, y)$ does the exact opposite by losing against the team $\overline{w} \in \overline{W}$ corresponding to the element w, and winning against the teams $\overline{x} \in \overline{X}$ and $\overline{y} \in \overline{X}$ corresponding to the elements x and y respectively. This leaves the team r with zero excess.
- If a team $\overline{w} \in \overline{W}$ (respectively $\overline{x} \in \overline{X}$ and $\overline{y} \in \overline{Y}$) loses against the team in R, then it wins its other remaining game against $w \in W$ (respectively $x \in X$ and $y \in Y$), otherwise it loses the latter game. This leaves each team in $\overline{W}$, $\overline{X}$ and $\overline{Y}$ with zero excess.

Finally, as a result of the above assignment, each team $w \in W$ ends up losing exactly one and winning all the rest of its games, leaving it with zero excess. A similar observation for the teams in X and Y shows that they end up with zero excess as well. Therefore, the above assignment of outcomes is a winning one for the team i^*.

($\Leftarrow$) Now suppose there exists an assignment of the remaining games that makes the team i^* win. Observe that each team in W can win at most one of its games (while losing all the rest) without gaining positive excess score. We can therefore assume, without loss of generality, that each team in W wins *exactly* one of its games while losing all the others. Next, any team in $\overline{W}$ that loses against the team in W can be, without loss of generality, assumed to win against the team in R, while a team in $\overline{W}$ that wins against the team in W *must* lose against the team in R. Combining the above observations, it can be inferred that among all the triples in R that contain a fixed element $w \in W$, exactly one triple $r \in R$ loses against the team in $\overline{W}$ while all the other triples win their corresponding games against the teams in $\overline{W}$. We will say that r is the unique triple that is *activated* by w.

Continuing the above reasoning for the teams in the set R, it can be observed that any team in R that loses against the team in $\overline{W}$ can be, without loss of generality, assumed to win against the teams in $\overline{X}$ and $\overline{Y}$. In other words, the team corresponding to a triple that is activated by some element of W can be assumed to win against the teams in $\overline{X}$ and $\overline{Y}$. Moreover, any team in R that wins against the team in $\overline{W}$ *must* lose against the teams in $\overline{X}$ and $\overline{Y}$.

By a similar argument as above, any team in $\overline{X}$ (respectively $\overline{Y}$) that loses against the team in R can be, without loss of generality, assumed to win against the team in X (respectively Y), while any team in $\overline{X}$ (respectively $\overline{Y}$) that wins against the team in R must lose against the team in X (respectively Y).

Once again by earlier arguments, any team in X (respectively Y) can be, without loss of generality, assumed to win exactly one of its games while losing all the rest. As before, we will say that for each team $x \in X$ (respectively $y \in Y$), there is a unique triple in R that is activated by x (respectively y). Since any triple that is not activated by some element of W must lose against the teams in $\overline{X}$ and $\overline{Y}$, we have that the triples activated by wins from X and Y must coincide with those activated by W. The set of all activated triples can now be shown to constitute a valid solution to the 3-D Matching instance.

To finish the proof, observe that since the graph $\mathcal{G}$ in the reduced instance is bipartite, we have that Mixed-Elimination(S_1, S_2) is NP-complete even when the set of remaining games induces a bipartite graph, as desired. □

Theorem 7. pBorda-Manipulation *is* NP-complete *when* $\mathcal{A} \in$ {bipartite, general graph} *and* `pref-type` = total.

Proof. (Sketch.) We only provide a sketch of the proof here, since the detailed proof largely follows the template of the proof of Theorem 5. We show a polynomial time reduction from Mixed-Elimination(S_1, S_2) with $S_1 = [(10, -10), (-5, 5)]$ and $S_2 = [(8, -8), (-2, 2)]$, which was shown to be NP-complete in Lemma 6. Just like in the proof of Theorem 5, we match teams with candidates and the set of remaining games with the action space of the manipulator. Furthermore, if a game between two teams is scored according to S_1 (respectively S_2), then the corresponding candidates are set up in a 1:2 (respectively 1:4) configuration. The 'no games drawn' condition translates to `pref-type` = total. Finally, since the set of remaining games in the Mixed-Elimination instance can be assumed to induce a bipartite graph, the same holds for the action space of the manipulator in the reduced instance. □

Remark 6. (The need for Mixed-Elimination) It is worth pointing out that when `pref-type` = total, we cannot reuse the arguments in Theorem 5 to show a reduction from S-Elimination to pBorda-Manipulation. The reason is that for a scoring system S to correspond to the changes in pBorda scores due to the manipulator's vote, it must be of the form $[(\alpha, -\alpha), (0, 0), (-\beta, \beta)]$, where $\alpha, \beta \in \mathbb{N}$. When the manipulator is not allowed to skip any comparison—as is the case with `pref-type` = total—the required form becomes $S = [(\alpha, -\alpha), (-\beta, \beta)]$, which can be reduced to the trivial $[(0, 0)]$ system via the normalization scheme described in Section 2.9. This motivates the need to consider instances with more than one scoring system, as in Mixed-Elimination.

Remark 7. (Technical observation) It is easy to verify that if two teams have non-identical scores in our Mixed-Elimination instance (Lemma 6), then their scores must always differ by at least 1, regardless of the choice of the assignment for the remaining games. For the election instance (Theorem 7), this means that if the scores of any two candidates are distinct, then they must differ by at least $\frac{1}{60}$, regardless of the manipulator's vote. Thus, pBorda-Manipulation continues to be NP-complete even when the smallest possible non-zero difference between the scores of any two candidates is at least $\frac{1}{60}$. This feature of the reduced instance will be useful in the proof of Theorem 8.

Our final result in this section shows that pBorda-Manipulation remains NP-complete even when the manipulator is allowed to compare all pairs of candidates.

Theorem 8. pBorda-Manipulation *is* NP-complete *when* $\mathcal{A}$ = complete graph *and* `pref-type` $\in$ {total, acyclic, unrestricted}.

Proof. The problem is clearly in NP. For any fixed choice of `pref-type` $\in$ {total, acyclic, unrestricted}, we reduce from the corresponding pBorda-Manipulation problem with $\mathcal{A}$ = general graph, which was shown to be NP-complete in Theorems 5 and 7. We present the proof in two parts, namely (a) `pref-type` $\in$ {acyclic, unrestricted} and (b) `pref-type` = total. We start by providing a high-level overview of the reductions.

Informal idea of the reduction: The original and the reduced instances have the same set of candidates. The votes of the non-manipulators between the candidate pairs in $\mathcal{A}$ in the original instance are identical to those for the corresponding candidate pairs in the reduced instance. For candidate pairs outside $\mathcal{A}$ in the original instance that are in $a : b$ configuration, the corresponding candidate pairs in the reduced instance are in $aK : bK$ configuration, where K is a polynomial in n. Since the fraction of voters that prefer one candidate over the other in the original instance is the same as that for the corresponding candidates in the reduced instance, the pBorda scores of the candidates are unchanged across the two instances. *Scaling up* the non-manipulators' votes for the candidate pairs outside $\mathcal{A}$ in the reduced instance means that the additional vote of the manipulator can make very little change in the pBorda scores of these

candidate pairs. This essentially forces the manipulator in the reduced instance to work with the candidate pairs within $\mathcal{A}$, providing the desired equivalence of the two solutions. The formal proof follows.

Case (a): `pref-type` $\in$ {acyclic, unrestricted}

In this case, we reduce from the instance of pBorda-Manipulation constructed in the proof of Theorem 5 (refer to Figure 5b). Specifically, we will use the following two features of this election instance:

- there are at most 12 non-manipulators, and
- the smallest possible positive excess score that a candidate can have, regardless of how the manipulator chooses to vote, is at least $\frac{1}{12}$ (see Remarks 2 and 4).

We start by describing the construction of the reduced instance followed by showing the equivalence of solutions of the two problems.

Construction of the reduced instance: Given an instance $\langle \Pi, i^*, \mathcal{A}, \texttt{pref-type} \rangle$ of pBorda-Manipulation with $\mathcal{A}$ = general graph and `pref-type` $\in$ {acyclic, unrestricted}, we construct an instance $\langle \Pi', i^{*\prime}, \mathcal{A}', \texttt{pref-type}' \rangle$ of pBorda-Manipulation with $\mathcal{A}'$ = complete graph and $\texttt{pref-type}' = \texttt{pref-type}$ as follows:

1. The original and the reduced instances have the same set of candidates. The distinguished candidate $i^{*\prime}$ is the same as i^*.
2. The reduced instance consists of $(m' - 1)$ non-manipulators and one manipulator. We choose $m' = 6K + 1$ where $K = 13n^2 + 1$. The votes of the non-manipulators in the reduced instance are set up as follows:
   - if a candidate pair $(i, j) \in \mathcal{A}$ in the original instance is in $a : b$ configuration, then the corresponding candidate pair in the reduced instance is also in $a : b$ configuration.
   - if a candidate pair $(i, j) \notin \mathcal{A}$ in the original instance is in $a : b$ configuration, then the corresponding candidate pair in the reduced instance is in $aK : bK$ configuration.

In other words, the preferences of the non-manipulators between candidate pairs in $\mathcal{A}$ are unchanged, while those between the candidate pairs outside $\mathcal{A}$ are scaled up by a factor of K in the reduced instance. The pBorda scores of the candidates are preserved in the reduction since they depend only on the *fractional preferences* of voters, which are identical between the two instances. As a consequence, the excess scores of candidates are also unchanged. This reduction is clearly *efficient* since the size of the reduced instance is polynomial in the size of the pBorda-Manipulation input. Indeed, the reduced instance consists of n candidates and $\mathcal{O}(n^2)$ voters each with $\mathcal{O}(n^2)$ preference information.

We will use the notation $\mathcal{A}' \cap \mathcal{A}$ to refer to the set of all candidate pairs (i', j') in the reduced instance such that $(i, j) \in \mathcal{A}$.

Equivalence of solutions: ($\Rightarrow$) Suppose there exists a vote $\succ_{u_m}$ for the manipulator u_m in the original pBorda-Manipulation instance such that i^* is a pBorda winner. Then a winning vote $\succ_{u_{m'}}$ for the manipulator $u_{m'}$ in the reduced instance can be constructed by simply mimicking $\succ_{u_m}$ for the pairwise comparisons in $\mathcal{A}' \cap \mathcal{A}$ and skipping the rest of the comparisons. Since a skipped comparison does not count for any change in pBorda scores, the outcomes for the two instances are identical. Moreover, an acyclic vote in the original instance continues to be acyclic in the reduced instance.

($\Leftarrow$) Now suppose there exists a vote $\succ_{u_{m'}}$ for the manipulator $u_{m'}$ in the reduced instance such that $i^{*\prime}$ is the pBorda winner. We will show that there must exist a vote $\succ_{u_m}$ for the manipulator u_m in the original pBorda-Manipulation instance that makes i^* win.

Suppose, for the sake of contradiction, that there does not exist such a vote $\succ_{u_m}$ in the original instance. In particular, this means that the restriction of the winning vote $\succ_{u_{m'}}$ in the

reduced instance to the pairs in $\mathcal{A}' \cap \mathcal{A}$ does not constitute a winning vote in the original instance. In other words, the manipulator u_m in the original instance cannot succeed by mimicking the winning vote of $u_{m'}$ restricted to $\mathcal{A}' \cap \mathcal{A}$, since such a vote always leaves some candidate in $\mathcal{A}$ with positive excess score. On the other hand, the manipulator $u_{m'}$ in the reduced instance manages to nullify any such excess by using pairwise comparisons from *outside* $\mathcal{A}' \cap \mathcal{A}$.

We will now argue that such a scenario is impossible. Indeed, the largest possible excess score that can be nullified in the reduced instance by the combined contribution of *all* pairwise comparisons from outside $\mathcal{A}' \cap \mathcal{A}$ is $\frac{1}{K+1} \cdot n^2$. This is because there are at most n^2 such pairs, and each pair can help remove an excess of at most $\frac{1}{K+1}$ due to the scaling by K. Since $K > 13n^2$, we have that $\frac{1}{K+1} \cdot n^2 < \frac{1}{13}$. However, as mentioned earlier, the smallest possible positive excess score of any candidate in the original pBorda-Manipulation instance, regardless of how the manipulator chooses to vote, is at least $\frac{1}{12} > \frac{1}{13}$. Hence, the manipulator $u_{m'}$ cannot use pairwise preferences from outside $\mathcal{A}' \cap \mathcal{A}$ to convert a losing vote in the original election into a winning one in the reduced election, providing the desired contradiction.

Case (b): `pref-type` = total

In this case, we reduce from the instance of pBorda-Manipulation constructed in the proof of Theorem 7. We will make use of the observation from Remark 7 that the smallest possible non-zero difference in the pBorda scores of any pair of candidates in the original election instance is, without loss of generality, at least $\frac{1}{60}$. The construction of the reduced instance of pBorda-Manipulation with `pref-type` = total and A' = complete graph is identical to the one described earlier in case (a) (with the exception that this time $K = 60n^2 + 1$), so we move on to showing the equivalence of the two solutions.

Equivalence of solutions: ($\Rightarrow$) Suppose there exists a vote $\succ_{u_m}$ for the manipulator u_m in the original pBorda-Manipulation instance such that i^* is a pBorda-winner. Unlike the previous case, where a winning vote was constructed by mimicking the relevant preferences and skipping the rest of the comparisons, the argument here is slightly more subtle, since the manipulator in the reduced instance can no more choose to skip any pairwise comparisons. Therefore, the manipulator $u_{m'}$ must ensure that no choice of total preferences outside $\mathcal{A}' \cap \mathcal{A}$ in the reduced instance can offset even the smallest non-zero score difference between any two candidates inside $\mathcal{A}' \cap \mathcal{A}$. We know from Remark 7 and the choice of K above that this is indeed true. Hence, a winning vote $\succ_{u_{m'}}$ for the manipulator $u_{m'}$ in the reduced instance can be constructed by simply mimicking the preferences in $\succ_{u_m}$ for comparisons in $\mathcal{A}' \cap \mathcal{A}$ and providing arbitrary preferences for the remaining comparisons. Since the influence of the total preferences from outside $\mathcal{A}' \cap \mathcal{A}$ is essentially superfluous, the two elections have the same outcome.

($\Leftarrow$) Conversely, a winning vote in the reduced instance can be used to construct a winning vote in the original instance, simply by taking the restriction of the former to $\mathcal{A}' \cap \mathcal{A}$. The reasoning is identical to that of case (a). This completes the proof of Theorem 8. □

Remark 8. (When `pref-type` = transitive) Recall that the NP-completeness of pBorda-Manipulation when $\mathcal{A}$ = general graph and `pref-type` = transitive was argued in Remark 5. By starting our reduction in case (a) from an instance of pBorda-Manipulation with `pref-type` = transitive (instead of acyclic), we can, therefore, use similar reasoning to show that pBorda-Manipulation continues to be NP-complete when $\mathcal{A}$ = complete graph and `pref-type` = transitive. This result suggests that the approach of moving away from the standard framework of rankings by relaxing the requirement of comparing all pairs (and assuming only transitivity) can be a useful way of making the pBorda-Manipulation problem computationally hard. This observation is complemented by our result in case (b), which shows that relaxing the transitivity/acyclicity assumption (and only requiring that all candidate pairs are compared) is a similarly promising approach.

4.2.2 Complexity of Manipulating Copeland$^\alpha$

Our results for the complexity of Copeland$^\alpha$-Manipulation are summarized in Tables 2 to 4 in Section 1 and are stated as Theorems 9, 10 and 12. Recall from Corollary 4 in Section 4.2.1 that Copeland$^\alpha$-Manipulation is efficiently solvable for any action space $\mathcal{A}$ when `pref-type` = total+acyclic. This still leaves open the case of `pref-type` = total, acyclic or unrestricted.

For the case `pref-type`= total, Faliszewski et al. (2010) showed that Copeland$^\alpha$-Manipulation for $\mathcal{A}$ = complete graph is polynomial-time solvable for $\alpha \in \{0, 1\}$, and NP-complete for all rational values of $\alpha \in (0, 1) \setminus \{0.5\}$.[15] Notice that this setting corresponds to the votes being *tournaments*.

In Theorem 9, we extend the tractability results of Faliszewski et al. (2010) for $\alpha \in \{0, 1\}$ to *any* choice of the action space $\mathcal{A}$ and `pref-type` $\in$ {total, acyclic, unrestricted} (refer to Table 2 in Section 1). Along with Corollary 4, this result shows that the Copeland$^\alpha$ rule is efficiently manipulable when $\alpha \in \{0, 1\}$, regardless of the choice of $\mathcal{A}$ and `pref-type`.

For Copeland$^{0.5}$-Manipulation, we show tractability for any choice of $\mathcal{A}$ when `pref-type` $\in$ {acyclic, unrestricted} (refer to Table 3). For all remaining choices of α, $\mathcal{A}$ and `pref-type`, we show NP-completeness results in Theorems 10 and 12 (refer to Tables 3 and 4).

Theorem 9. Copeland$^\alpha$-Manipulation *is efficiently solvable in the following cases: (a) for* $\alpha \in \{0, 1\}$ *for any choice of* $\mathcal{A}$ *when* `pref-type` $\in$ {total, acyclic, unrestricted} *and (b) for* $\alpha = 0.5$ *for any choice of* $\mathcal{A}$ *when* `pref-type` $\in$ {acyclic, unrestricted}.

Proof. We start with the observation that the vote of the manipulator between a pair of candidates (i, j) has no effect on the Copeland scores if $|m_{ij} - m_{ji}| > 1$, where m_{ij} and m_{ji} denote the number of non-manipulators who vote $i \succ j$ and $j \succ i$ respectively. Hence, without loss of generality, the manipulator either *skips* such comparisons (if `pref-type` $\in$ {acyclic, unrestricted}) or provides *arbitrary* preferences for such pairs (if `pref-type` = total). Thus, we will assume for the rest of the proof that all candidate pairs (i, j) in $\mathcal{A}$ are of the form $|m_{ij} - m_{ji}| \leq 1$. We will also assume throughout that the distinguished candidate wins all its pairwise comparisons in the manipulator's vote, and therefore its Copeland score is fixed.

We now describe the manipulation algorithms for each setting of α and `pref-type`. These algorithms will differ from each other only in their handling of the two subcases corresponding to $m_{ij} = m_{ji} + 1$ and $m_{ij} = m_{ji}$.

Case (a): Copeland$^\alpha$-Manipulation *for* $\alpha \in \{0, 1\}$ *for any choice of* $\mathcal{A}$ *and* `pref-type` $\in$ {total, acyclic, unrestricted}.

(1) $\alpha = 0$

(i) `pref-type` = unrestricted

In this case, the manipulator would like to create as many *ties* between pairs of non-distinguished candidates as possible (thereby providing $\alpha = 0$ points to both candidates involved). Thus, for any candidate pair (i, j) in $\mathcal{A}$, the manipulator votes $j \succ i$ whenever $m_{ij} = m_{ji} + 1$, and skips the comparison otherwise. Formally,

$$\succ_{u_m} = \bigcup_{(i,j)\in\mathcal{A}} \begin{cases} \{ j \succ i \} & \text{if } m_{ij} = m_{ji} + 1; \\ \text{skip} & \text{otherwise.} \end{cases}$$

[15] Faliszewski et al. (2010) claim in Theorem 5.2 of their paper that Copeland$^{0.5}$-Manipulation with $k \geq 1$ manipulators is also efficiently solvable when $\mathcal{A}$=complete graph and `pref-type` = total. Their proof uses the computational results of Faliszewski et al. (2009b) about a related problem called *microbribery* of Copeland$^\alpha$ elections. This problem was shown by Faliszewski et al. (2009b) to be efficiently solvable for $\alpha \in \{0, 1\}$ but not for $\alpha = 0.5$. Hence, the polynomial time solvability of Copeland$^{0.5}$-Manipulation cannot be directly inferred from the corresponding microbribery version, since the computational complexity of the latter is not known. We fill this gap by showing that Copeland$^{0.5}$-Manipulation is in fact NP-complete when $\mathcal{A} \in$ {bipartite, complete graph} and `pref-type` = total (refer to Theorem 12), implying the same for the corresponding microbribery problem (of which the manipulation problem is a special case).

(ii) `pref-type` = total

In this case, the manipulator faces a situation similar to the one above with the additional constraint that skipping a comparison is not allowed. Recall that in the above case, all comparisons skipped by the manipulator are of the form $m_{ij} = m_{ji}$. For such candidate pairs (i, j), the Copeland scores of i and j can change by (+1,0) or (0,+1) depending on whether the manipulator votes $i \succ j$ or $j \succ i$. The problem of finding a winning vote over such pairs can be reduced to S-Elimination with S = $[(1, 0), (0, 1)]$ in the following straightforward manner: Each candidate i or j as above corresponds to a team, and the Copeland score of i or j is equal to the current score of the team. Moreover, the comparison between the pair (i, j) (where $m_{ij} = m_{ji}$) corresponds to the game to be played between the corresponding teams. We know from Theorem 1 in Section 2.8 that this subproblem is efficiently solvable. Hence, the manipulator's vote consists of comparisons of the form $j \succ i$ whenever $m_{ij} = m_{ji} + 1$ combined with the votes/outcomes resulting from the S-Elimination subroutine.

(iii) `pref-type` = acyclic

In this case, the vote constructed above for the case '`pref-type` = unrestricted' suffices if it contains no directed cycles. Otherwise, we will show that it suffices to run the greedy algorithm (Algorithm 1) from Theorem 3 in Section 4.2.1 over the action space $\mathcal{A}$ restricted to the pairs with $m_{ij} = m_{ji} + 1$ (and skip all other comparisons).

First, observe that for candidate pairs (i, j) where $m_{ij} = m_{ji}$, the Copeland scores of the candidates can change by $(+1, 0)$, $(0, 0)$ or $(0, +1)$ for the comparison $i \succ j$, skip or $j \succ i$ made by the manipulator respectively. Hence it is without loss of generality that the manipulator skips such comparisons.

Next, we focus on the pairs (i, j) with $m_{ij} = m_{ji} + 1$. Let us denote the action space $\mathcal{A}$ restricted to these pairs by $\mathcal{A}^{\text{res}}$. For such pairs, we claim that *if there exists a winning vote for the manipulator with* `pref-type` = acyclic*, then there must also exist a winning vote with* `pref-type` = total+acyclic over $\mathcal{A}^{\text{res}}$. This claim can be proved as follows: Let $\succ^{\text{acyc}}$ be the restriction of a winning acyclic vote to $\mathcal{A}^{\text{res}}$. We will argue that $\succ^{\text{acyc}}$ can be transformed into a total and acyclic vote $\succ^{\text{tot+acyc}}$ such that the latter is also a winning vote. In order to see this, notice that for any $(i, j) \in \mathcal{A}^{\text{res}}$, the Copeland scores of i and j can change by $(0, 0)$, $(0, 0)$ or $(-1, 0)$ depending on whether the manipulator votes $i \succ j$, skip or $j \succ i$ respectively. Therefore, a skipped comparison between a pair of candidates (i, j) in $\succ^{\text{acyc}}$ can be replaced with a strict comparison (either $i \succ j$ or $j \succ i$) without increasing the Copeland score of either i or j. We will now argue that such a replacement does not introduce any new directed cycles in $\succ^{\text{acyc}}$. This is because either the replacement $i \succ j$ suffices, or there must be a directed path in $\succ^{\text{acyc}}$ consisting of a series of comparisons of the form $j \succ j_1, j_1 \succ j_2, \ldots, j_k \succ i$ that forbids the introduction of $i \succ j$. In this case, the replacement $j \succ i$ can be used instead. Note that both $i \succ j$ and $j \succ i$ cannot be forbidden in this manner, since that would imply that $\succ^{\text{acyc}}$ was cyclic to begin with. Thus, any skipped comparison in $\succ^{\text{acyc}}$ can be replaced with a strict comparison, and, as a result, any acyclic winning vote $\succ^{\text{acyc}}$ can be transformed, without loss of generality, into another winning vote $\succ^{\text{tot+acyc}}$ that is total and acyclic. This transformation reduces the search space to solutions with `pref-type` = total+acyclic over $\mathcal{A}^{\text{res}}$, for which the greedy algorithm from Theorem 3 suffices.

(2) $\alpha = 1$

(i) `pref-type` = unrestricted

In this case, the manipulator tries to *avoid ties* between pairs of candidates in $\mathcal{A}$. This is because for any pairwise comparison between the non-distinguished candidates, it is

strictly better for the manipulator to have a clear winner (resulting in exactly one of the two candidates getting 1 point while the other gets 0) instead of both candidates getting a point each because of a tie. Therefore, the manipulator skips all comparisons in $\mathcal{A}$ except the ones that are already tied on the basis of the non-manipulators' votes (i.e., $m_{ij} = m_{ji}$). For such pairs, just like in the previous case, the manipulator faces the S-Elimination subproblem with $\mathrm{S} = [(1,0),(0,1)]$, which is efficiently solvable.

(ii) `pref-type` = total

In this case, the manipulator faces the additional constraint that no comparison can be skipped. Notice that all comparisons skipped by the manipulator for the above case are of the form $m_{ij} = m_{ji} + 1$. For each such pair, the manipulator avoids a tie by simply voting $i \succ j$.

(iii) `pref-type` = acyclic

In this case, the vote constructed above for the case '`pref-type` = unrestricted' suffices if it contains no directed cycles. Otherwise, just like in Copeland^0-Manipulation, it can once again be shown that any valid vote consists entirely of skipped comparisons except for the candidate pairs (i,j) with $m_{ij} = m_{ji}$, for which the manipulator must provide strict comparisons of the form $i \succ j$ or $j \succ i$. As before, such a vote can be efficiently computed by using the greedy algorithm (Algorithm 1) suggested in Theorem 3 as a subroutine, where the input to the algorithm is the action space $\mathcal{A}$ restricted to the candidate pairs with $m_{ij} = m_{ji}$.

Case (b): Copeland$^{0.5}$-Manipulation *when* $\mathcal{A}$ = general graph *and* `pref-type` $\in$ {acyclic, unrestricted}.

(i) `pref-type` = unrestricted

Observe that for each candidate pair (i,j) where $m_{ij} = m_{ji} + 1$, the Copeland scores of the candidates can change by $(0,0)$, $(0,0)$ or $(-0.5,+0.5)$ depending on whether the manipulator votes $i \succ j$, skip or $j \succ i$ respectively. Similarly, for each candidate pair (i,j) where $m_{ij} = m_{ji}$, the Copeland scores of the candidates (i,j) can change by $(+0.5,-0.5)$, $(0,0)$ or $(-0.5,+0.5)$ for the votes $i \succ j$, skip or $j \succ i$ of the manipulator respectively. By using the natural extension of the normalization scheme described in Section 2.9 for the Mixed-Elimination problem, the manipulation problem can be reduced to Mixed-Elimination$(\mathrm{S}_1, \mathrm{S}_2)$ with $\mathrm{S}_1 = [(1,0),(0,1)]$ and $\mathrm{S}_2 = [(2,0),(1,1),(0,2)]$. The latter is efficiently solvable using the max-flow techniques of Kern and Paulusma (2004).

(ii) `pref-type` = acyclic

In this case, the vote constructed above for the case '`pref-type` = unrestricted' suffices if it contains no directed cycles. Otherwise, we will argue that any such feasible but cyclic vote can be transformed into an equivalent acyclic vote by 'ironing out' all directed cycles. Specifically, we will show that the series of strict comparisons in any directed cycle of the form $i_1 \succ i_2$, $i_2 \succ i_3$, $\ldots$, $i_{k-1} \succ i_k$, $i_k \succ i_1$ can be transformed to a series of *skipped* comparisons without affecting the Copeland score of any candidate.

We start by focusing on the candidate pairs (i,j) with $m_{ij} = m_{ji} + 1$. Recall that for such pairs, the Copeland scores of i and j can change by $(0,0)$, $(0,0)$ or $(-0.5,+0.5)$ depending on whether the manipulator votes $i \succ j$, skip or $j \succ i$ respectively. Therefore, any $i \succ j$ comparison in this case can be replaced with an equivalent skip comparison without changing the Copeland score of either i or j. Hence, we can assume, without loss of generality, that any strict comparison in this case is of the form $j \succ i$.

Next, consider the candidate pairs (i,j) with $m_{ij} = m_{ji}$. For such pairs, the Copeland scores of i and j can change by $(+0.5,-0.5)$, $(0,0)$ or $(-0.5,+0.5)$ depending on whether

the manipulator votes $i \succ j$, skip or $j \succ i$ respectively. It is easy to see that there is no way of replacing a strict comparison with an equivalent skip comparison in this case. As a result, strict comparisons of both forms $i \succ j$ and $j \succ i$ are possible for such pairs.

In summary, there can be three kinds of strict comparisons in the manipulator's vote:

(1) $j \succ i$ for pairs (i, j) with $m_{ij} = m_{ji} + 1$,

(2) $j \succ i$ for pairs (i, j) with $m_{ij} = m_{ji}$, and

(3) $i \succ j$ for pairs (i, j) with $m_{ij} = m_{ji}$.

Notice that in each case, replacing a strict comparison $i_1 \succ i_2$ with a skip vote amounts to an increase of 0.5 in the Copeland score of i_2, and a decrease of 0.5 in the Copeland score of i_1. Therefore, for any given directed cycle of the form $i_1 \succ i_2$, $i_2 \succ i_3$, $\ldots$, $i_{k-1} \succ i_k$, $i_k \succ i_1$, replacing each strict comparison with a skip comparison amounts to a simultaneous increase and decrease of 0.5 in the Copeland score of each candidate, implying that the Copeland scores of all the candidates stay the same. Hence, any directed cycle can be ironed out in this manner without loss of generality. □

Our next result shows that Copeland$^\alpha$-Manipulation is NP-complete for all $\alpha \in \mathbb{Q} \cap (0, 1) \setminus \{0.5\}$ when $\mathcal{A}$ is a bipartite or complete graph and `pref-type` $\in$ {acyclic, unrestricted} (see Table 4 in Section 1). We show a reduction from S-Elimination with $\mathrm{S} = [(1/\alpha, 0), (1, 1), (0, 1/\alpha)]$, which is known to be NP-complete for all $\alpha \in \mathbb{Q} \cap (0, 1) \setminus \{0.5\}$ (refer to Theorem 1 in Section 2.8). Our proof closely follows the template of the proof of Theorem 5 for pBorda-Manipulation.

Theorem 10. Copeland$^\alpha$-Manipulation *is* NP-complete *for all* $\alpha \in \mathbb{Q} \cap (0, 1) \setminus \{0.5\}$ *when* $\mathcal{A} \in$ {bipartite, complete, general graph} *and* `pref-type` $\in$ {acyclic, unrestricted}.

Proof. The problem is clearly in NP. Just like in the proof of Theorem 5, we reduce from the S-Elimination instance of Kern and Paulusma (2004) with $\mathrm{S} = [(1/\alpha, 0), (1, 1), (0, 1/\alpha)]$. That is, we once again start from an instance of S-Elimination that consists of seven groups of teams, namely $W, \overline{W}, R, \overline{X}, \overline{Y}, X$ and Y, shown as circles, squares and triangles in Figure 5a. The distinguished team i^* (not shown in the figure) has no remaining games and has an initial score of 0. The current scores of the teams are as follows:

- when $0 < \alpha < \frac{1}{2}$: $W(-1 - (\delta - 1)/\alpha)$, $\overline{W}(-1)$, $R(-\max\{1/\alpha, 3\})$, $\overline{X}(-1 - 1/\alpha)$, $\overline{Y}(-1 - 1/\alpha)$, $X(-1)$, and $Y(-1)$, and
- when $\frac{1}{2} < \alpha < 1$: $W(-1)$, $\overline{W}(-1 - 1/\alpha)$, $R(-\max\{2/\alpha, 3\})$, $\overline{X}(-1)$, $\overline{Y}(-1)$, $X(-1 - (\delta - 1)/\alpha)$, and $Y(-1 - (\delta - 1)/\alpha)$.

In each of the above cases, $\delta(\cdot)$ denotes the degree function of the graph $\mathcal{G}$ corresponding to the set of remaining games. Besides, for each remaining game, the teams from $\overline{W}$, $\overline{X}$ and $\overline{Y}$ should be considered the *away* team. The NP-completeness of S-Elimination for both ranges of α stated above follows directly from Theorem 1 in Section 2.8.

The reduced Copeland$^\alpha$-Manipulation instance $\langle \Pi, i^*, \mathcal{A}, \texttt{pref-type} \rangle$ is constructed as follows:

- The *set of candidates* $[n]$ consists of
  - a candidate i for each team $i \in [N]$ (including a candidate i^* corresponding to the distinguished team i^*), and

– the dummy candidates[16] $D_1, \ldots, D_5$. Hence, there are $n = N + 5$ candidates overall.

- The *action space* $\mathcal{A}$ of the manipulator corresponds to the set of remaining games $\mathcal{G}$.
- The *set of non-manipulators* consists of a single voter, say u_1 (hence $m = 2$). This means that in order to specify the *voting profile*, we only need to describe the vote of u_1, as below (all comparisons that are not explicitly specified should be considered as skipped):
  - when $0 < \alpha \leq \frac{1}{3}$: In this case, the vote of u_1 is given by:
    * $\{D_1\} \succ_{u_1} \{W, \overline{W}, \overline{X}, \overline{Y}, X, Y\}$,
    * $\{i^*, W_{\delta=1}, \overline{W}, X, Y\} \succ_{u_1} \{D_2\}$,
    * $\{D_2\} \succ_{u_1} \{W_{\delta=2}, W_{\delta=3}, R, \overline{X}, \overline{Y}\}$,
    * $\{i^*, W_{\delta=1}, W_{\delta=2}, \overline{W}, R, \overline{X}, \overline{Y}, X, Y\} \succ_{u_1} \{D_3\}$, and
    * $\{D_3\} \succ_{u_1} \{W_{\delta=3}\}$.
  - when $\frac{1}{3} < \alpha < \frac{1}{2}$: In this case, the vote of u_1 is given by:
    * $\{D_1\} \succ_{u_1} \{W, \overline{W}, R, \overline{X}, \overline{Y}, X, Y\}$,
    * $\{i^*, W_{\delta=1}, \overline{W}, R, X, Y\} \succ_{u_1} \{D_2\}$,
    * $\{D_2\} \succ_{u_1} \{W_{\delta=2}, W_{\delta=3}, \overline{X}, \overline{Y}\}$,
    * $\{i^*, W_{\delta=1}, W_{\delta=2}, \overline{W}, R, \overline{X}, \overline{Y}, X, Y\} \succ_{u_1} \{D_3\}$,
    * $\{D_3\} \succ_{u_1} \{W_{\delta=3}\}$, and
    * $\{D_4, D_5\} \succ_{u_1} \{R\}$.
  - when $\frac{1}{2} < \alpha \leq \frac{2}{3}$: In this case, the vote of u_1 is given by:
    * $\{D_1\} \succ_{u_1} \{W, \overline{W}, \overline{X}, \overline{Y}, X, Y\}$,
    * $\{i^*, W, \overline{W}, \overline{X}, \overline{Y}, X_{\delta=1}, Y_{\delta=1}\} \succ_{u_1} \{D_2\}$,
    * $\{D_2\} \succ_{u_1} \{X_{\delta=2}, Y_{\delta=2}, X_{\delta=3}, Y_{\delta=3}, R\}$,
    * $\{i^*, W, \overline{X}, \overline{Y}, X_{\delta=1}, Y_{\delta=1}, X_{\delta=2}, Y_{\delta=2}\} \succ_{u_1} \{D_3\}$, and
    * $\{D_3\} \succ_{u_1} \{\overline{W}, R, X_{\delta=3}, Y_{\delta=3}\}$.
  - when $\frac{2}{3} < \alpha < 1$: In this case, the vote of u_1 is given by:
    * $\{D_1\} \succ_{u_1} \{W, \overline{W}, R, \overline{X}, \overline{Y}, X, Y\}$,
    * $\{i^*, W, \overline{W}, R, \overline{X}, \overline{Y}, X_{\delta=1}, Y_{\delta=1}\} \succ_{u_1} \{D_2\}$,
    * $\{D_2\} \succ_{u_1} \{X_{\delta=2}, Y_{\delta=2}, X_{\delta=3}, Y_{\delta=3}\}$,
    * $\{i^*, W, R, \overline{X}, \overline{Y}, X_{\delta=1}, Y_{\delta=1}, X_{\delta=2}, Y_{\delta=2}\} \succ_{u_1} \{D_3\}$,
    * $\{D_3\} \succ_{u_1} \{\overline{W}, X_{\delta=3}, Y_{\delta=3}\}$, and
    * $\{D_4, D_5\} \succ_{u_1} \{R\}$.
- Finally, `pref-type` is set to either acyclic or unrestricted.

The equivalence of the two solutions follows from the arguments in Theorem 5.

Once again, just like in Theorem 5, the graph $\mathcal{G}$ in the original sports instance (Figure 5a) is *bipartite* and any winning assignment of the remaining games in $\mathcal{G}$ is *acyclic* without loss of generality. As a result, the same holds for the reduced instance as well.

Finally, by using two non-manipulators (instead of one), it is possible to use the same procedure as above to show NP-completeness of Copeland$^\alpha$-Manipulation when $\mathcal{A}$ = complete

[16] In order to ensure that no dummy candidate ever wins the election no matter how the manipulator chooses to vote, we can add $q > \frac{N+4\alpha}{1-\alpha}$ additional dummy candidates, each of which loses to all the candidates representing the teams and 'plays a draw' with all the other dummy candidates. This is because adding q dummy candidates gives each team candidate an additional score of q, while increasing the score of any dummy candidate by at most $\alpha(q+4)$. Overall, therefore, we only require that $q - \alpha(q+4) > N$, which gives the aforementioned bound. We keep the description of the reduced instance simple by avoiding this technical detail.

graph. Indeed, in the above proof, for candidate pairs outside the bipartite graph $\mathcal{A}$, we set up the non-manipulators' votes in a 2:0 configuration in order to nullify the effect that a single manipulator can have. This way, the manipulator is effectively forced to solve the manipulation problem over a bipartite space even when $\mathcal{A}$ is a complete graph. The votes of the non-manipulators involving the dummy candidates can be similarly constructed. □

Our final result (Theorem 12) shows that Copeland$^\alpha$-Manipulation is NP-complete for all $\alpha \in \mathbb{Q} \cap (0,1)$ when the manipulator is required to provide total preferences over a bipartite or complete graph. The reduction is from Mixed-Elimination$(\mathrm{S}_1, \mathrm{S}_2)$ with $\mathrm{S}_1 = [(1,0),(0,1)]$ and $\mathrm{S}_2 = [(1-\alpha,0),(0,\alpha)]$, which is shown to be NP-complete in Lemma 11 for all $\alpha \in \mathbb{Q} \cap (0,1)$ via reduction from 3-D Matching by a proof similar to that of Lemma 6.

Lemma 11. *For all* $\alpha \in \mathbb{Q} \cap (0,1)$, Mixed-Elimination$(\mathrm{S}_1, \mathrm{S}_2)$ *with* $\mathrm{S}_1 = [(1,0),(0,1)]$ *and* $\mathrm{S}_2 = [(1-\alpha,0),(0,\alpha)]$ *where the set of remaining games* $\mathcal{G}$ *induces a bipartite graph is* NP-complete.

Proof. Our proof closely follows that of Lemma 6. Since membership in NP is clear, we proceed to show a polynomial time reduction from a variant of 3-D Matching where each base element occurs in at most three triples. This variant is known to be NP-complete (Garey and Johnson, 1979, Problem SP1 on page 221).

As before, we treat the vertices of the *matching graph* of the given 3-D Matching instance as teams and the edges as the set of remaining games between teams (refer to Figure 6). The games in the upper half of the corresponding sports instance graph are scored according to $\mathrm{S}_1 = [(1,0),(0,1)]$ and those in the lower half are scored according to $\mathrm{S}_2 = [(1-\alpha,0),(0,\alpha)]$, shown as solid and dashed edges in Figure 6.

Construction of the reduced instance: Given an instance $\langle W, X, Y, R\rangle$ of 3-D Matching, we construct an instance $\langle \mathbf{s}, i^*, \mathcal{G}, \eta\rangle$ of Mixed-Elimination$(\mathrm{S}_1, \mathrm{S}_2)$ as follows:

- the *set of teams* $[N]$ is the union of the sets $W, X, Y, \overline{W}, \overline{X}, \overline{Y}, R$ (i.e., one team per element) along with the distinguished team i^*,
- the *set of remaining games* $\mathcal{G}$ is the set of all solid and dashed edges in the matching graph,
- the *assignment function* η is such that all games involving teams from $\overline{W}$ are scored according to S_1, while all other games in $\mathcal{G}$ are scored according to S_2. Furthermore, for any game, a team *higher up* in the matching graph should be considered the *away* team, and
- the *current score* of the team i^* is fixed at 0, without loss of generality.

The group-wise excess scores of all teams are: $W(-1)$, $\overline{W}(-1)$, $R(-\max\{1, 2\alpha\})$, $\overline{X}(-\max\{\alpha, 1-\alpha\})$, $\overline{Y}(-\max\{\alpha, 1-\alpha\})$, $X(-(\delta-1)(1-\alpha))$ and $Y(-(\delta-1)(1-\alpha))$.

The equivalence of solutions follows from arguments similar to those in Lemma 6. Finally, observe that since the matching graph is bipartite, the same holds for the graph $\mathcal{G}$ induced by the set of remaining games. This completes the proof of Lemma 11. □

Theorem 12. Copeland$^\alpha$-Manipulation *for all* $\alpha \in \mathbb{Q} \cap (0,1)$ *when* $\mathcal{A} \in \{$bipartite, complete graph$\}$ *and* `pref-type` = total *is* NP-complete.

Proof. Our proof uses a polynomial time reduction from the instance of Mixed-Elimination$(\mathrm{S}_1, \mathrm{S}_2)$ with $\mathrm{S}_1 = [(1,0),(0,1)]$ and $\mathrm{S}_2 = [(1-\alpha,0),(0,\alpha)]$ which was shown to be NP-complete in Lemma 11 even when the graph induced by the set of remaining games is bipartite. Just like in the proof of Theorem 10, there is a candidate for each team (along with

a constant number of dummy candidates), and the set of remaining games corresponds to the action space of the manipulator. We set `pref-type` = total.

The *set of non-manipulators* consists of a single voter, say u_1 (hence $m = 2$). This means that in order to specify the *voting profile*, we only need to describe the vote of u_1, as below (all comparisons that are not explicitly specified should be considered as skipped):

- $j \succ_{u_1} i$ for any two teams $i \in R$ and $j \in \overline{X} \cup \overline{Y}$ such that $(i, j) \in \mathcal{G}$.
- $i \succ_{u_1} j$ for any two teams $i \in X \cup Y$ and $j \in \overline{X} \cup \overline{Y}$ such that $(i, j) \in \mathcal{G}$.
- The preferences involving the dummy candidates depend on the value of α as follows:
  - when $0 < \alpha < \frac{1}{2}$: In this case, the vote of u_1 is given by:
    - $\{i^*, \overline{X}, \overline{Y}, X_{\delta=1}, Y_{\delta=1}\} \succ_{u_1} \{D_1\}$,
    - $\{D_1\} \succ_{u_1} \{W, \overline{W}, R, X_{\delta=2}, Y_{\delta=2}, X_{\delta=3}, Y_{\delta=3}\}$,
    - $\{D_2, D_3, D_4\} \succ_{u_1} \{i^*, X_{\delta=1}, Y_{\delta=1}\}$,
    - $\{D_3, D_4\} \succ_{u_1} \{W_{\delta=1}\}$,
    - $\{D_4\} \succ_{u_1} \{W_{\delta=2}, \overline{W}, \overline{X}, \overline{Y}, X_{\delta=2}, Y_{\delta=2}\}$,
    - $\{D_5\} \succ_{u_1} \{\overline{X}, \overline{Y}, X_{\delta=2}, Y_{\delta=2}, X_{\delta=3}, Y_{\delta=3}\}$,
    - $\{i^*, W, \overline{W}, R, X_{\delta=1}, Y_{\delta=1}\} \succ_{u_1} \{D_5\}$,
    - $\{i^*, W, \overline{W}, R, \overline{X}, \overline{Y}, X_{\delta=1}, Y_{\delta=1}, X_{\delta=2}, Y_{\delta=2}\} \succ_{u_1} \{D_6\}$,
    - $\{D_6\} \succ_{u_1} \{X_{\delta=3}, Y_{\delta=3}\}$,
    - $\{D_7\} \succ_{u_1} \{i^*, W, \overline{W}, R, \overline{X}, \overline{Y}, X_{\delta=1}, Y_{\delta=1}, X_{\delta=2}, Y_{\delta=2}\}$,
    - $\{i^*, W, \overline{W}, R, \overline{X}, \overline{Y}, X_{\delta=1}, Y_{\delta=1}, X_{\delta=2}, Y_{\delta=2}\} \succ_{u_1} \{D_8\}$, and
    - $\{D_8\} \succ_{u_1} \{X_{\delta=3}, Y_{\delta=3}\}$.
  - when $\frac{1}{2} \leq \alpha < 1$: In this case, the vote of u_1 is given by:
    - $\{i^*, R, \overline{X}, \overline{Y}, X_{\delta=1}, Y_{\delta=1}\} \succ_{u_1} \{D_1\}$,
    - $\{D_1\} \succ_{u_1} \{W, \overline{W}, X_{\delta=2}, Y_{\delta=2}, X_{\delta=3}, Y_{\delta=3}\}$,
    - $\{D_2, D_3, D_4\} \succ_{u_1} \{i^*, \overline{X}, \overline{Y}, X_{\delta=1}, Y_{\delta=1}\}$,
    - $\{D_3, D_4\} \succ_{u_1} \{R, W_{\delta=1}\}$,
    - $\{D_4\} \succ_{u_1} \{W_{\delta=2}, \overline{W}, X_{\delta=2}, Y_{\delta=2}\}$,
    - $\{D_5\} \succ_{u_1} \{X_{\delta=2}, Y_{\delta=2}, X_{\delta=3}, Y_{\delta=3}\}$,
    - $\{i^*, W, \overline{W}, R, \overline{X}, \overline{Y}, X_{\delta=1}, Y_{\delta=1}\} \succ_{u_1} \{D_5\}$,
    - $\{i^*, W, \overline{W}, R, \overline{X}, \overline{Y}, X_{\delta=1}, Y_{\delta=1}, X_{\delta=2}, Y_{\delta=2}\} \succ_{u_1} \{D_6\}$,
    - $\{D_6\} \succ_{u_1} \{X_{\delta=3}, Y_{\delta=3}\}$,
    - $\{D_7\} \succ_{u_1} \{i^*, W, \overline{W}, R, \overline{X}, \overline{Y}, X_{\delta=1}, Y_{\delta=1}, X_{\delta=2}, Y_{\delta=2}\}$,
    - $\{i^*, W, \overline{W}, R, \overline{X}, \overline{Y}, X_{\delta=1}, Y_{\delta=1}, X_{\delta=2}, Y_{\delta=2}\} \succ_{u_1} \{D_8\}$, and
    - $\{D_8\} \succ_{u_1} \{X_{\delta=3}, Y_{\delta=3}\}$.

The equivalence of the solutions follows from the arguments similar to those in Theorem 5. Moreover, since the graph $\mathcal{G}$ in the original sports instance is *bipartite*, the manipulation problem continues to be NP-complete when the action space $\mathcal{A}$ is bipartite. Finally, the above proof can be easily extended to the case $\mathcal{A}$ = complete graph using the techniques described in the proof of Theorem 10. □

4.3 Parameterized Complexity Results

Our classification of the parameterized complexity of pBorda-Manipulation for all combinations of the considered parameters is summarized by Theorem 13 (also refer to Table 5 in Section 1). We remark that we have no explicit restrictions on the type of preference relation in the manipulator's vote (i.e., `pref-type` = unrestricted). Before providing the formal results, we briefly motivate the choice of parameters used in this study.

Motivation for the choice of parameters: In the parameterized studies of computational problems that arise in the context of voting, a commonly used parameter is the *number of candidates*, n (Bartholdi III et al., 1989a; Conitzer et al., 2007; Betzler et al., 2009a; Faliszewski et al., 2009b; Dorn and Schlotter, 2012; Yang, 2014; Xia, 2014). We observe that for any pairwise voting rule that is easy to evaluate, the problem of manipulation by a single manipulator is trivially FPT for this choice of parameter, because even a brute-force search over all possible votes of the manipulator will yield the desired running time (i.e., $\mathcal{O}(3^{n^2})$). The other natural choice of parameter is the *number of voters*. However, we know from Remark 2 that pBorda-Manipulation is NP-complete *even with twelve non-manipulators*. Given the extreme behaviors on the two obvious choices of parameters, we turn to the action space of the manipulator $\mathcal{A}$ and try to understand how the problem complexity is influenced by parameters associated with the structure of $\mathcal{A}$.

We observe that the complexity of pBorda-Manipulation varies considerably with the structure of the action space $\mathcal{A}$. On one hand, we have the tractability result from Theorem 3 which states that pBorda-Manipulation is efficiently solvable when $\mathcal{A}$ is a tree/forest. On the other extreme, we know from Theorem 8 that pBorda-Manipulation can be computationally intractable when the action space $\mathcal{A}$ is the complete graph. Given these two extremes, we follow the *distance from triviality* approach in parameterized analysis (Guo et al., 2004) and consider parameters that measure how *far* the given instance is from the class of tractable instances (i.e., the degree of closeness to being a tree or a forest). This motivates the choice of parameters such as *treewidth* (**tw**), *feedback vertex set* (**fvs**) and *maximum degree* (Δ), and upper/lower bounds on these parameters such as *pathwidth* (**pw**) and *vertex cover* (**vc**) (refer to Section 2.7 for formal definitions). Interestingly, a similar set of parameters was recently used in the parameterized complexity analysis of the closely-related S-Elimination problem (Cechlárová et al., 2016), and studying the influence of these parameters on the complexity of pBorda-Manipulation allows us to compare the complexity landscapes of the two problems (Remark 9).

Overview of our results: Our first set of results shows that the manipulation problem is, somewhat surprisingly, *para-NP-complete* for (a) the *maximum degree* parameter, and (b) *any combination of the parameters in* {**vc**, **fvs**, **pw**, **tw**}. This already establishes a contrast to S-Elimination which, for instance, is in XP when parameterized by the *treewidth* of the graph formed by the set of the remaining games (Cechlárová et al., 2016).

On the tractability side, we show that pBorda-Manipulation is FPT when simultaneously parameterized by *maximum degree* (Δ) and *any combination of the other parameters*. We ask if there is a *natural* parameter that is, in general, smaller than *maximum degree* (Δ), but that can still provide tractability when combined with some of the other structural parameters. We discover an answer in the form of a novel parameter called *diversity* (d), which is a measure of how many different types of score exchanges the manipulator encounters for any candidate (refer to Definition 1). Although pBorda-Manipulation turns out to be NP-complete on graphs with constant *diversity* (d)—in fact, it remains NP-complete even when the sum of *diversity* (d) and *maximum degree* (Δ) is bounded by a constant (Theorem 5)—we obtain an FPT algorithm by combining *diversity* with *vertex cover* (**vc**), and XP algorithms by combining it with any of the other parameters in {**fvs**, **pw**, **tw**}. We do not expect to improve this XP result, as the problem remains W[1]-hard in those cases.

Theorem 13 (Parameterized Complexity)**.** *Let* $\mathcal{P} = \{$***vc***, ***pw***, ***fvs***, ***tw***, $\Delta, d\}$ *denote the set of*

parameters defined with respect to the action space $\mathcal{A}$. Let $\mathcal{X} = \{\boldsymbol{vc}, \boldsymbol{pw}, \boldsymbol{fvs}, \boldsymbol{tw}\}$ and $\mathcal{Y} = \{d, \Delta\}$. Then,

1. *For any $\mathcal{Q} \subseteq \mathcal{X}$ or $\mathcal{Q} \subseteq \mathcal{Y}$,* pBorda-Manipulation *is* NP-complete *even when the sum of the values of all parameters in $\mathcal{Q}$ is bounded by a constant.*

2. *For all $\mathcal{Q} \subseteq \mathcal{P}$,* pBorda-Manipulation *parameterized by $\mathcal{Q}$ is in XP if $\mathcal{Q}$ contains d along with any element of $\mathcal{X}$. Further, the problem is* FPT *if $\mathcal{Q}$ contains Δ along with any element of $\mathcal{X}$, or if it contains both d and $\boldsymbol{vc}$.*

3. *In the case when $\mathcal{Q} \subseteq \mathcal{P}$ does not contain either Δ or $\boldsymbol{vc}$,* pBorda-Manipulation *is* W[1]-hard *when parameterized by $\mathcal{Q}$, even on instances where d is bounded by a constant.*

All the claims made in Theorem 13 can be inferred from two algorithmic results and three reductions, one of which we have already encountered (Theorem 5). We briefly summarize these main results and their implications.

1. We show that pBorda-Manipulation remains NP-complete even for instances where $\mathcal{A}$ has a *vertex cover* of size two (Theorem 14). Since a bound on the size of a *vertex cover* implies a bound on the size of a *feedback vertex set*, *pathwidth* and *treewidth*, we have NP-completeness of pBorda-Manipulation even with respect to all parameters in $\mathcal{X}$ combined. Together with Remark 3, which shows NP-completeness of pBorda-Manipulation on instances of *maximum degree* $\Delta = 3$ (and therefore *diversity* $d \leq 3$), this implies statement 1 of Theorem 13.

2. We use dynamic programming over tree decompositions to show that pBorda-Manipulation is FPT when parameterized by *maximum degree* and *treewidth* (Theorem 16). Since all other parameters in $\mathcal{X}$ are larger than *treewidth*, this gives us an FPT classification when *maximum degree* is combined with any subset of parameters in $\mathcal{X}$. Similarly, we use the result of Lenstra Jr (1983) on Integer Linear Programming being FPT in the number of variables to show that pBorda-Manipulation is FPT when simultaneously parameterized by the *vertex cover* and *diversity* of $\mathcal{A}$ (Theorem 18). These two results together imply statement 2 of Theorem 13.

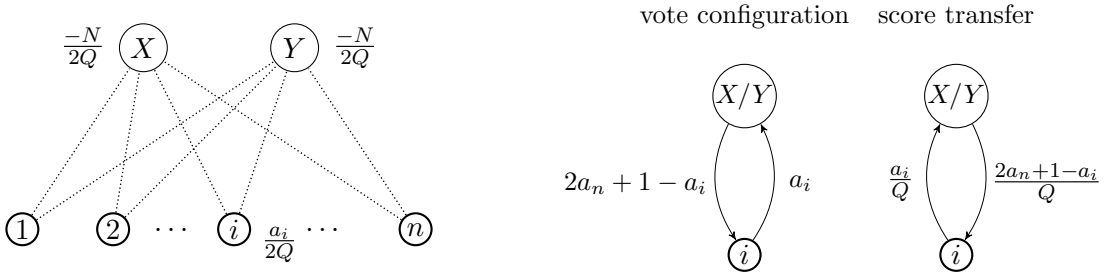
3. Finally, we show that pBorda-Manipulation is W[1]-hard when simultaneously parameterized by the *feedback vertex set* and *pathwidth* of $\mathcal{A}$ via an FPT-reduction from Capacitated Dominating Set (Theorem 15). This proves statement 3 of Theorem 13.

We will now provide formal statements and proofs of the results stated above. Our first result in this section (Theorem 14) shows NP-completeness of pBorda-Manipulation on instances with a *vertex cover* of constant size.

Theorem 14. pBorda-Manipulation *is* NP-complete *when $\mathcal{A}$ is a general graph with a* vertex cover *of size two and* `pref-type` = unrestricted.

Proof. The problem is clearly in NP. We show NP-hardness by reduction from Partition. We start with an informal overview to the proof followed by detailed explanations.

Informal idea of the reduction: Recall from Section 2.10 that an instance of Partition consists of a multiset S of n numbers $a_1 \leq a_2 \leq \cdots \leq a_n$ which sum to N, and the task is to decide if there is a subset whose sum is $N/2$. We will introduce two candidates X and Y that will serve the purpose of a selection gadget (refer to Figure 7a). Further, we have a candidate i corresponding to each number a_i. The action space $\mathcal{A}$ is the complete bipartite graph with $\{X, Y\}$ on one side and the candidates $\{1, 2, \ldots, n\}$ on the other. We now aim to set up the non-manipulators' votes (with the help of additional dummy candidates that are not part of the action space $\mathcal{A}$) such that the following are true:



(a) *Reduced* pBorda-MANIPULATION *instance* **(b)** *Vote configuration and score transfers*

Figure 7: *This figure shows the instance of* pBorda-MANIPULATION *constructed in the proof of Theorem 14. Subfigure (a) shows the action space of the manipulator as dotted lines, along with the corresponding excess scores. Subfigure (b) shows the configuration of the non-manipulators' votes and the resulting score transfers. For each $i \in [n]$, the candidate pairs (X, i) and (Y, i) are both in $(2a_n + 1 - a_i) : a_i$ configuration. As a result, a* pBorda *score of a_i/Q gets transferred from i to X (or Y) if the manipulator votes $X \succ i$ (or $Y \succ i$), while a transfer of $(2a_n + 1 - a_i)/Q$ happens in the opposite direction for the vote $i \succ X$ (or $i \succ Y$).*

- The excess score of both X and Y is $-N/2$ units and that of each i is $a_i/2$ units.
- If the manipulator votes $X \succ i$ (or $Y \succ i$), then a_i units of pBorda score are transferred from i to X (or Y).[17] However, if the manipulator votes $i \succ X$ (or $i \succ Y$), then a large amount of score is transferred from X (or Y) to i. In particular, the transferred amount is greater than a_i, and thus offloading it to the other selector Y (or X) will not suffice. This means that the manipulator never votes $i \succ X$ (or $i \succ Y$) for any i. Therefore, the manipulator is required to distribute the combined excess of all i's, which equals N, between the two selectors X and Y, who can together handle an excess of at most N. This corresponds to solving the given instance of PARTITION.

Construction of the reduced instance: Given an instance $S = \{a_1, a_2, \ldots, a_n\}$ of PARTITION, we construct an instance $\langle \Pi, i^*, \mathcal{A}, \texttt{pref-type} \rangle$ of pBorda-MANIPULATION as follows (refer to Figure 7):

- The *candidate set* consists of
  - the *selector candidates* X and Y, corresponding to the two partitions,
  - a candidate i for each positive integer $a_i \in S$ (call these the *integer* candidates),
  - the *distinguished candidate* i^*, and
  - the *dummy candidates* $D_1, D_2, \ldots, D_{4n}$.

  Hence there are $(2 + n + 1 + 4n) = 5n + 3$ candidates overall.
- The *action space* $\mathcal{A}$ for the manipulator is the complete bipartite graph between the *selectors* and the *integer* candidates. That is, $\mathcal{A} = \big\{\{\cup_{i\in[N]}\{(X, i)\} \bigcup \{\cup_{i\in[N]}\{(Y, i)\}\big\}$. Note that the distinguished candidate i^* and the dummy candidates are not part of this space, and hence their pBorda scores are not affected by the manipulator's vote.
- Set `pref-type` = unrestricted.
- We will now describe the construction of the *voting profile* Π *of the non-manipulators.* The set of voters consists of $2Q$ non-manipulators (where $Q = (2a_n + 1) \cdot (2a_n + 2)$) and one manipulator, hence $m = 2Q + 1$ voters overall.

[17] Refer to Section 2.4 for details on the score transfer property of the pBorda rule.

– The votes of the non-manipulators for candidate pairs in the action space (i.e., between the *selectors* and the *integer* candidates) are set up so as to ensure that the score transfers resulting from the manipulator's vote are $\frac{a_i}{Q}$ (if the manipulator votes $X \succ i$ or $Y \succ i$) and $\frac{2a_n+1-a_i}{Q}$ (if the manipulator votes $i \succ X$ or $i \succ Y$). Thus, for each $i \in [n]$, the candidate pairs (i, X) and (i, Y) are both in $a_i : (2a_n + 1 - a_i)$ configuration, as shown in Figure 7b.

– For each $i \in [n]$, the candidate pair (i^*, D_i) is in $(2a_n + 1 - a_i) : a_i$ configuration while the candidate pair (i^*, D_{n+i}) is in $a_i : (2Q - a_i)$ configuration.

– For each $i \in [n]$, the candidate pair (i, D_{2n+i}) is in $(2a_n + 1 - 3a_i) : 3a_i$ configuration while the candidate pair (i, D_{3n+i}) is in $2a_i : (2Q - 2a_i)$ configuration. Finally, for each $i \in [n]$ and each $k \in [n] \setminus i$, the candidate pair (i, D_{2n+k}) is in $(2a_n + 1 - a_k) : a_k$ configuration while the candidate pair (i, D_{3n+k}) is in $a_k : (2Q - a_k)$ configuration.

This finishes the construction of the election instance. It is easy to check that the excess score of each *integer* candidate i after this construction is $\frac{a_i}{2Q}$, while that of each *selector* is $\frac{-N}{2Q}$. Besides, the scores of the dummy candidates are sufficiently low so that any selector or integer candidate always has a higher pBorda score than any dummy candidate, regardless of how the manipulator chooses to vote. As a result, the winning candidate is always one of the candidates in the action space.

Note that the above reduction is *efficient* since it uses $\mathcal{O}(a_n^2)$ voters and $\mathcal{O}(n)$ candidates. Also note that the *selectors* X, Y form a *vertex cover* of size two, which implies a bound on the *treewidth*, *pathwidth* and *feedback vertex set* (Figure 1 in Section 1). All other vertices in $\mathcal{A}$ have degree two.

Equivalence of solutions: ($\Rightarrow$) Suppose there exists a partition of S into the sets S_1 and S_2. A valid manipulative vote can be constructed from this partition as follows: For each $i \in [n]$, the manipulator votes $X \succ i$ if $a_i \in S_1$ or $Y \succ i$ if $a_i \in S_2$ and skips all other comparisons. The final excess pBorda score of each *integer* candidate i is negative, since $\frac{a_i}{2Q} - \frac{a_i}{Q} < 0$. The final excess score for each of the *selectors* equals 0 due to the partition property, making i^* the pBorda winner.

($\Leftarrow$) Suppose there exists a valid manipulative vote that makes i^* win. We will show that a valid partition can be constructed from such a vote. First, observe that in any winning vote, each *integer* candidate i must lose against at least one of the selectors in order to get rid of its initial excess. This means that the combined pBorda score that gets transferred from the *integer* candidates to the *selectors* is at least $\sum_{a_k \in S} \frac{a_k}{Q}$. Next, observe that no *integer* candidate i can win a pairwise comparison against any of the *selectors*. This is because a win against either X or Y leaves i with an additional score of $\frac{2a_n+1-a_i}{Q}$, which is more than what it can offload to the other selector. This means that no pBorda score gets transferred from the *selectors* to the *integer* candidates. Since the *selectors* can together handle an inflow of at most $\sum_{a_k \in S} \frac{a_k}{Q}$, any valid manipulative vote must have each *integer* candidate i lose against exactly one of the *selectors*, while the other comparison is skipped. A partition can now be naturally inferred from such a vote. This completes the proof of Theorem 14. □

Remark 9. Notice that Theorem 14 provides a complexity theoretic distinction between pBorda-Manipulation and S-Elimination. As mentioned earlier, Cechlárová et al. (2016) showed that S-Elimination is in XP when parameterized by the *treewidth* of the graph formed by the set of remaining games. On the other hand, pBorda-Manipulation is para-NP-complete in the same parameter. Hence, pBorda-Manipulation is necessarily harder than S-Elimination unless $P = NP$.

Our next result establishes the W[1]-hardness of pBorda-Manipulation in terms of the size of a *minimum feedback vertex set* and the *pathwidth* of $\mathcal{A}$, even on instances where the *diversity* of $\mathcal{A}$ is bounded by a constant.

Theorem 15. pBorda-Manipulation *is* W[1]-hard *when simultaneously parameterized by feedback vertex set and pathwidth of* $\mathcal{A}$*, where* $\mathcal{A}$ *is a general graph with constant-sized diversity and* `pref-type` = unrestricted.

Proof. We will show an FPT reduction from Capacitated Dominating Set to pBorda-Manipulation with the following properties:

- the *feedback vertex set* and *pathwidth* parameters of the action space $\mathcal{A}$ of the pBorda-Manipulation instance are polynomially bounded by the corresponding parameters of the original Capacitated Dominating Set instance, and
- the number of distinct capacity values in the Capacitated Dominating Set instance is an upper bound for the *diversity* of the pBorda-Manipulation instance (up to additive constants).

These two properties give us the desired hardness implications, since we know from Section 2.10 that Capacitated Dominating Set is W[1]-hard when simultaneously parameterized by *feedback vertex set* and *pathwidth* (and as a consequence, *treewidth*), even when there are only a constant number of distinct capacity values. We start with an informal overview of the proof.

Informal idea of the reduction: Given an instance $\langle G, c, k\rangle$ of Capacitated Dominating Set with $G = (V, E)$, we construct an instance of pBorda-Manipulation consisting of a *source* candidate X, two *sink* candidates—an *edge sink* Y and *vertex sink* Y'—and a candidate v_i for each vertex $v_i \in V$, called a *vertex* candidate (refer to Figure 8a). For each pair of *vertex* candidates (v_i, v_j) whose corresponding vertices are adjacent in G, we add an *edge* candidate representing the edge (v_i, v_j). The manipulator is allowed to make the following kinds of comparisons:

- each *vertex* candidate can be compared with the *source* and the *vertex sink*, and
- each *edge* candidate can be compared with the *edge sink* and the two *vertex* candidates adjacent to it.

The initial excess score of each *vertex* candidate v_i is $(\phi - 1 - c(v_i))$ units (where $\phi = \max_{v_i \in V} c(v_i)$), while that of the *source* X is a sufficiently large number. The large initial excess score of the *source* forces the manipulator to offload it to at least $(|V| - k)$ *vertex* candidates, thus making each of them *heavy*. Specifically, each *vertex* candidate that helps the *source* shed its excess now has an excess of its own equalling $(B + \phi - 1)$ units, where B is a suitably large number. Let H denote the set of all such *heavy vertex candidates*. Notice that the action space allows each *vertex* candidate to offload its excess to the *vertex sink* and the *edge* candidates adjacent to it. If an *edge* candidate is triggered in this way, then it is forced to offload its own excess to the *edge sink* and the other *vertex* candidate adjacent to it. The score transfers[18] in our reduced instance are set up in a way that makes it necessary for the manipulator to trigger a distinct *edge* candidate for each *heavy vertex* candidate in H. After this step, all candidates in H that were previously *heavy* now become *light* (i.e., with non-positive excess), while a different set of *vertex candidates* (say H') now become *heavy* (i.e., with a positive excess) due to score inflow from *edge* candidates triggered earlier. Notice that at least $(|V| - k)$ *edge* candidates are triggered in this process. By construction, this is also the maximum number of *edge* candidates that can be triggered, since the *edge sink* cannot handle any additional score inflow. Therefore, no other *edge* candidates can be triggered to accommodate the excess of any candidate in H', leaving the manipulator with only the *vertex sink* to work with.

Once again, by construction, the excess score of the *vertex sink* is set up to allow score inflow from at most k *vertex* candidates. Moreover, no vertex candidate is allowed to offload

[18] Refer to Section 2.4 for details on the score transfer property of the pBorda rule.

more than $(\phi - 1)$ units of score to the *vertex sink*. These constraints force the manipulator to ensure the following two conditions:

- no candidate in H' has an excess of more than $(\phi - 1)$ units (otherwise such a candidate will be left with a positive excess with nowhere to offload it to), and
- $|H'| \leq k$.

In other words, the set H' must be such that each candidate in the original *heavy* set H channels an excess of 1 toward some candidate in H' (via an *edge* candidate), while no $v_i \in H'$ receives an inflow of more than $c(v_i)$ units in this process. The set H' must therefore be a capacitated dominating set.

The detailed proof that follows describes the prescription of the non-manipulators' votes that realize the desired configuration of capacities, excess scores and score transfers.

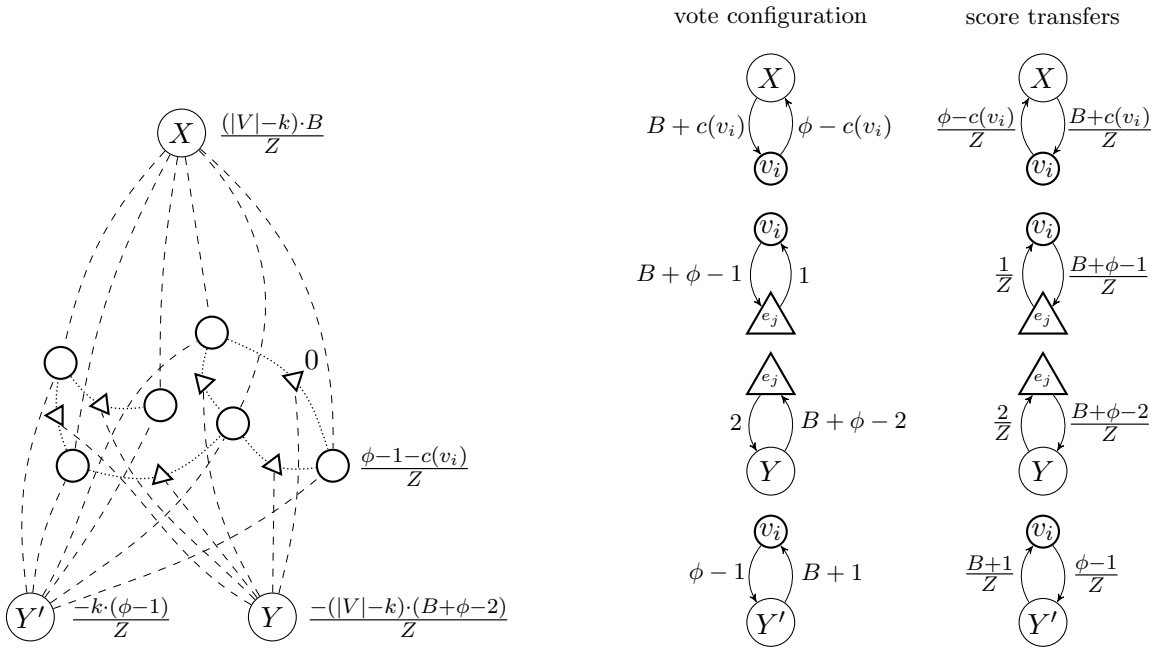
Construction of the reduced instance: Given an instance $\langle G, c, k\rangle$ of Capacitated Dominating Set, we construct an instance $\langle \Pi, i^*, \mathcal{A}, \texttt{pref-type}\rangle$ of pBorda-Manipulation as follows (refer to Figure 8):

- The *set of candidates* consists of
  - the *source* X, the *edge sink* Y and the *vertex sink* Y',
  - a candidate v_i for each vertex v_i in G, called a *vertex* candidate,
  - a candidate e_j for each edge e_j in G, called an *edge* candidate,
  - the *distinguished* candidate i^*,
  - the *dummy* candidates $D_1, D_2, \ldots, D_\ell$. These are further divided into five subsets $D^{(1)}, D^{(2)}, \ldots, D^{(5)}$ of different sizes. Overall, there are $\ell = 3|V| + |E| + 2$ *dummy* candidates.

  Hence, there are $n = 4|V| + 2|E| + 6$ candidates overall.
- The *action space* $\mathcal{A}$ is the union of all unordered pairs of candidates connected by dashed and dotted edges in Figure 8. That is,

$$\mathcal{A} = \big\{\{\cup_{i\in[|V|]}\{X, v_i\}\} \bigcup \{\cup_{j\in[|E|]}\{e_j, Y\}\} \bigcup \{\cup_{i\in[|V|]}\{v_i, Y'\}\} \bigcup \{\cup_{i\in[|V|],j\in[|E|]}\{v_i, e_j\} \text{ where } v_i \text{ is adjacent to } e_j \text{ in } G\}\big\}.$$

  Note that the *distinguished* candidate i^* and the *dummy* candidates are not part of the action space and hence their pBorda scores are not affected by the manipulator's vote.
- Set `pref-type` = unrestricted.
- We will now describe the construction of the *votes of the non-manipulators*. The set of voters consists of Z non-manipulators and one manipulator, hence $m = Z + 1$. Here, $Z = (B + \phi) \cdot (B + \phi + 1)$, $B = |V| + |E| + \sum_{v_i \in V(G)} c(v_i)$ and $\phi = \max_{v_i \in V} c(v_i)$. The scores acquired by the candidates in each step are shown in Table 7.
  - For each $i \in [|V|]$, the candidate pairs (X, v_i) and (Y', v_i) are in $(B+c(v_i)) : (\phi - c(v_i))$ and $(B+1) : (\phi - 1)$ configuration respectively. For each $i \in [|V|]$ and each $j \in [|E|]$ such that v_i is adjacent to e_j in G, the candidate pair (v_i, e_j) is in $(B + \phi - 1) : 1$ configuration. Finally, for each $j \in [|E|]$, the candidate pair (e_j, Y) is in $2 : (B+\phi-2)$ configuration. The vote arrangements and corresponding score transfers after this step are shown in Figure 8b.



(a) *Reduced* pBorda-Manipulation *instance* **(b)** *Vote configuration and score transfers*

Figure 8: *Subfigure (a) shows the reduced* pBorda-Manipulation *instance (excluding the dummy candidates and the distinguished candidate i^*) constructed from the given* Capacitated Dominating Set *instance. The action space $\mathcal{A}$ of the manipulator is marked via dashed and dotted edges, and the excess* pBorda *scores are mentioned next to the corresponding candidates. The* vertex *candidates are shown as circles in the middle layer and the* edge *candidates are shown as triangles. Subfigure (b) shows the configuration of non-manipulators' votes for various candidate-pairs and the resulting scores transfers. For instance, the pair (X, v_i) is in $(B+c(v_i)) : (\phi - c(v_i))$ configuration. As a result, a* pBorda *score of $(\phi - c(v_i))/Z$ gets transferred from v_i to X if the manipulator votes $X \succ v_i$, while a transfer of $(B + c(v_i))/Z$ happens in the opposite direction for the vote $v_i \succ X$.*

- For each $i \in [|E|]$, $(B + \phi - 2)$ voters vote $(X, Y', v_{1:|V|}, e_{1:|E|}, i^*) \succ D_i^{(1)}$ while two voters vote the opposite.[19]
- For each $i \in [|V|]$, $(B + 1)$ voters vote $(e_{1:|E|}, i^*) \succ D_i^{(2)}$ while $(\phi - 1)$ voters vote the opposite. For each $i \in [|V| - k]$, B voters vote $Y \succ D_i^{(2)}$ while ϕ voters vote the opposite. For each $i \in \{|V| - k + 1, \ldots, |V|\}$, $B + 1$ voters vote $Y \succ D_i^{(2)}$ while $(\phi - 1)$ voters vote the opposite. For each $i \in [|V|]$, $(\phi - c(v_i))$ voters vote $X \succ D_i^{(2)}$ while $(B + c(v_i))$ voters vote the opposite. For each $i \in [|V| - \deg(v_i)]$, $(B + \phi - 1)$ voters vote $v_i \succ D_i^{(2)}$ while one voter votes the opposite.
- The candidate pairs $(X, D_1^{(3)})$, $(Y, D_1^{(3)})$, $(Y', D_1^{(3)})$ and $(i^*, D_1^{(3)})$ are in $(2\phi - c(v_i) + 3) : (B - \phi + c(v_i) - 3)$ configuration. For each $i \in [|V|]$, the pairs $(v_i, D_1^{(3)})$ are in $5 : (B+\phi-5)$ configuration. For each $j \in [|E|]$, the pairs $(e_j, D_1^{(3)})$ are in $(2\phi - c(v_i) - 1) : (B - \phi + c(v_i) + 1)$ configuration. The candidate pairs $(X, D_2^{(3)})$, $(Y, D_2^{(3)})$, $(Y', D_2^{(3)})$ and $(i^*, D_2^{(3)})$ are in $\phi : (Z - \phi)$ configuration. For each $i \in [|V|]$, the pairs $(v_i, D_2^{(3)})$ are in $(\phi - c(v_i)) : (Z - \phi + c(v_i))$ configuration. For each $j \in [|E|]$, the pairs $(e_j, D_2^{(3)})$ are in $\phi : (Z - \phi)$ configuration.

[19] We use the shorthand $(X, Y', v_{1:|V|}, e_{1:|E|}, i^*) \succ D_i^{(1)}$ to represent $X \succ D_i^{(1)}$, $Y' \succ D_i^{(1)}$, $v_{1:|V|} \succ D_i^{(1)}$, $e_{1:|E|} \succ D_i^{(1)}$ and $i^* \succ D_i^{(1)}$. Here, $v_{1:|V|} = v_1, \ldots, v_{|V|}$ and $e_{1:|E|} = e_1, \ldots, e_{|E|}$.

– For each $i \in [k]$, $(\phi-1)$ voters vote $(X, Y, e_{1:|E|}, i^*) \succ D_i^{(4)}$ while $(Z-\phi+1)$ voters vote the opposite. For each $i \in [k+1]$, $(\phi-1)$ voters vote $(v_{1:|V|}) \succ D_i^{(4)}$ while $(Z-\phi+1)$ voters vote the opposite. For each $i \in \{k+1, \ldots, |V|\}$, B voters vote $X \succ D_i^{(4)}$ while $(Z-B)$ voters vote the opposite. For each $i \in \{k+1, \ldots, |V|\}$, 3 voters vote $Y \succ D_i^{(4)}$ while $(Z-3)$ voters vote the opposite.

– For each $i \in [|V|]$, $(\phi-1)$ voters vote $(Y, Y', e_{1:|E|}, i^*) \succ D_i^{(5)}$ while $(B+1)$ voters vote the opposite. For each $i \in [|V|-1]$, one voter votes $v_{1:|V|} \succ D_i^{(5)}$ while $(B+\phi-1)$ voters vote the opposite.

This finishes the construction of the election instance. Note that the reduction is *efficient* since it uses $\mathcal{O}((|V|+|E|)^2)$ voters and $\mathcal{O}(|V|+|E|)$ candidates. Also note that the reduction is *parameter preserving*, in that if the *pathwidth* and the *feedback vertex set* of the Capacitated Dominating Set instance are w and t respectively, then the same parameters for the action space $\mathcal{A}$ in the pBorda-Manipulation instance are $\mathcal{O}(w^2)$ and $(t+3)$ respectively.[20] Moreover, since the original Capacitated Dominating Set instance has $\mathcal{O}(1)$ distinct capacity values, there can only be $\mathcal{O}(1)$ distinct score exchanges due to the addition of the manipulator's vote (refer to Figure 8b). Therefore, the *diversity* of the reduced election instance is a constant.

Equivalence of solutions: ($\Rightarrow$) Suppose $S \subseteq V(G)$ is a valid capacitated dominating set, and let $f : V(G) \setminus S \to S$ be the corresponding assignment. Then, a winning vote can be constructed as follows: First, the manipulator triggers score transfers from the *source* X to the *vertex* candidates in $V(G) \setminus S$ by voting $v_i \succ X$ for all $v_i \in V(G) \setminus S$. This brings the excess score of X below zero, and results in an excess of $(B+\phi-1)/Z$ for each $v_i \in V(G) \setminus S$. Next, for each such v_i, the manipulator votes $e_j \succ v_i$ where e_j is the edge connecting v_i and $f(v_i)$ in G. After this step, the excess scores of all candidates in $V(G) \setminus S$ become zero. Each *edge* candidate e_j chosen above now acquires an excess score of $(B+\phi-1)/Z$. For each such e_j, the manipulator votes $Y \succ e_j$ and $f(v_i) \succ e_j$. This results in a score transfer of $1/Z$ and $(B+\phi-2)/Z$ to $f(v_i)$ and Y respectively. The excess score of e_j now becomes zero, while each $f(v_i)$ acquires an excess of $(\phi-1)/Z$, since it experiences score inflows from $c(v_i)$ candidates in $V(G) \setminus S$ of magnitude $1/Z$ each. The excess score of Y also becomes zero. In the final step, the manipulator votes $Y' \succ f(v_i)$ for all $f(v_i) \in S$. This step triggers a score transfer of $(\phi-1)/Z$ from each $f(v_i)$ to the *vertex sink* Y', resulting in a final excess score of zero for all candidates in S as well as for Y', making i^* the pBorda winner.

($\Leftarrow$) Suppose there exists a valid manipulative vote that makes i^* win. Without loss of generality, X must lose against at least $(|V|-k)$ *vertex* candidates in order to offload its excess. Let S' denote the set of *vertex* candidates that X loses to. Hence, $|S'| \geq |V|-k$. Each candidate in S' acquires an excess of $(B+\phi-1)/Z$, which it can offload to the *vertex sink* Y' and the *edge* candidates adjacent to it. It is easy to see that offloading the excess to Y' alone doesn't suffice, and hence, without loss of generality, each candidate in S' must offload its excess to at least one of the *edge* candidates adjacent to it. Notice that at most $(|V|-k)$ *edge* candidates can be triggered in this manner, since each such *edge* candidate is, in turn, forced to offload its own excess by transferring a score of $(B+\phi-2)/Z$ to the *edge sink* Y, and there can only be $(|V|-k)$ such transfers without giving positive excess to Y (it is easy to check that no transfers happen in the reverse direction). As a result, each candidate in S' offloads its excess to *exactly one edge candidate* adjacent to it (which also means that $|S'| = |V|-k$), and ends up with an excess

[20] Notice that if the *pathwidth* of the original instance is w, then the size of each bag in a path decomposition is at most $(w+1)$. In order to obtain a path decomposition for the reduced instance, we will add to each bag (in the path decomposition of the original instance) the *source* and the two *sink* candidates, along with the *edge* candidates corresponding to the edges originally present in that bag. The size of the largest bag in the new path decomposition is at most $(w+1)+3+\binom{w+1}{2}$. Thus, the *pathwidth* of the reduced instance is $\mathcal{O}(w^2)$. Similarly, a *feedback vertex set* of the reduced instance can be obtained by combining the *feedback vertex set* for the original instance with the *source* and the two *sink* vertices, giving a size bound of $(t+3)$.

	Action space	$D^{(1)}$	$D^{(2)}$	$D^{(3)}$	$D^{(4)}$	$D^{(5)}$	Excess score
X	$\sum_{v_i \in V} \frac{B+c(v_i)}{B+\phi}$	$\lvert E\rvert \cdot \frac{B+\phi-2}{B+\phi}$	$\sum_{v_i \in V} \frac{\phi-c(v_i)}{B+\phi}$	$\frac{2\phi-c(v_i)+3}{B+\phi} + \frac{\phi}{Z}$	$k \cdot \frac{\phi-1}{Z} +$ $(\lvert V\rvert - k) \cdot \frac{B}{Z}$	0	$(\lvert V\rvert - k) \cdot \frac{B}{Z}$
Y	$\lvert E\rvert \cdot \frac{B+\phi-2}{B+\phi}$	0	$k \cdot \frac{B+1}{B+\phi} +$ $(\lvert V\rvert - k) \cdot \frac{B}{B+\phi}$	$\frac{2\phi-c(v_i)+3}{B+\phi} + \frac{\phi}{Z}$	$k \cdot \frac{\phi-1}{Z} +$ $(\lvert V\rvert - k) \cdot \frac{3}{Z}$	$\lvert V\rvert \cdot \frac{\phi-1}{B+\phi}$	$-(\lvert V\rvert - k) \cdot \frac{B+\phi-2}{Z}$
Y'	$\lvert V\rvert \cdot \frac{B+1}{B+\phi}$	$\lvert E\rvert \cdot \frac{B+\phi-2}{B+\phi}$	0	$\frac{2\phi-c(v_i)+3}{B+\phi} + \frac{\phi}{Z}$	0	$\lvert V\rvert \cdot \frac{\phi-1}{B+\phi}$	$-k \cdot \frac{\phi-1}{Z}$
v_i	$\frac{\phi-c(v_i)}{B+\phi} + \frac{\phi-1}{B+\phi} +$ $\deg(v_i) \cdot \frac{B+\phi-1}{B+\phi}$	$\lvert E\rvert \cdot \frac{B+\phi-2}{B+\phi}$	$(\lvert V\rvert - \deg(v_i)) \cdot$ $\frac{B+\phi-1}{B+\phi}$	$\frac{5}{B+\phi} + \frac{\phi-c(v_i)}{Z}$	$(k+1) \cdot \frac{\phi-1}{Z}$	$(\lvert V\rvert - 1) \cdot \frac{1}{B+\phi}$	$\frac{\phi-1-c(v_i)}{Z}$
e_j	$\frac{4}{B+\phi}$	$\lvert E\rvert \cdot \frac{B+\phi-2}{B+\phi}$	$\lvert V\rvert \cdot \frac{B+1}{B+\phi}$	$\frac{\phi}{Z} + \frac{2\phi-c(v_i)-1}{B+\phi}$	$k \cdot \frac{\phi-1}{Z}$	$\lvert V\rvert \cdot \frac{\phi-1}{B+\phi}$	0
i^*	0	$\lvert E\rvert \cdot \frac{B+\phi-2}{B+\phi}$	$\lvert V\rvert \cdot \frac{B+1}{B+\phi}$	$\frac{2\phi-c(v_i)+3}{B+\phi} + \frac{\phi}{Z}$	$k \cdot \frac{\phi-1}{Z}$	$\lvert V\rvert \cdot \frac{\phi-1}{B+\phi}$	0

Table 7: *This table shows the* pBorda *scores acquired by various candidates in our construction in the proof of Theorem 15. Each entry of the table indicates the* pBorda *score acquired by a row candidate due to its interaction with the set of candidates in each column. The rightmost column shows the excess scores.*

of zero. This excess is transferred via the *edge* candidate to another *vertex* candidate. Let S denote the set of *vertex* candidates that witness a score inflow from the candidates in S'. It is easy to see that $S \cap S' = \emptyset$ (otherwise some candidate in S' gains a positive excess score with nowhere to offload it to). Notice that the only outlet for the candidates in S to offload their excess to is the *vertex sink* Y' since (a) no *edge* candidate can be triggered anymore, and (b) a reverse transfer to the source X is not possible as there is no remaining outlet for X. As mentioned earlier, the *vertex sink* can handle score transfers from at most k vertex candidates, of magnitude $(\phi - 1)/Z$ each. Therefore, the set S must be such that (a) no candidate in S has an excess score of more than $(\phi - 1)/Z$ after the transfers from S' have happened, and (b) $|S| \leq k$. Putting together the above observations, we get that, without loss of generality, any winning vote requires the manipulator to select a set S of candidates such that $|S| \leq k$, and each candidate in S witnesses a score inflow from at most $c(v_i)$ candidates in S'. Such a set S constitutes a capacitated dominating set. This completes the proof of Theorem 15. □

Remark 10. Cechlárová et al. (2016) have shown that except for scoring systems of the form $\mathrm{S} = \{(i, t-i) : 0 \leq i \leq t\}$ for some $t \in \mathbb{N}$, S-Elimination is W[1]-hard separately in the parameters *feedback vertex set* and *pathwidth*. We know from the proof of Theorem 5 that S-Elimination with $\mathrm{S} = [(3,0),(1,2),(0,3)]$ is a special case of pBorda-Manipulation. Hence, the results of Cechlárová et al. (2016) can be used to infer W[1]-hardness of pBorda-Manipulation in each of these two parameters separately. Our result, however, is stronger since we show a hardness result in the two parameters combined.

Our first algorithmic result is about fixed-parameter tractability of pBorda-Manipulation in the parameters *treewidth* and *maximum degree* of $\mathcal{A}$.

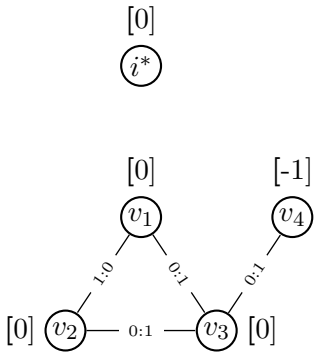
Theorem 16. pBorda-Manipulation *is solvable in time* $\mathcal{O}\big(\Delta^{\mathcal{O}(dw^2)}(n \log m)^{\mathcal{O}(1)}\big)$, *where* Δ, w *and* d *denote the maximum degree, treewidth and diversity of* $\mathcal{A}$ *respectively.*

Proof. Let $\mathcal{T} = (T, \{B_t\}_{t \in V(T)})$ be a nice tree decomposition of $\mathcal{A}$, where T is a rooted binary tree with root node r. Recall from Section 2.7 that a nice tree decomposition has, in addition to the root and leaf nodes, the following three kinds of internal nodes: *introduce*, *forget* and *join*. Our goal for the rest of the proof will be to follow the standard bottom-up dynamic programming (DP) approach for nice tree decompositions (Cygan et al., 2015) by providing explicit DP recurrences for each type of node. Figure 9 shows an example where the main ideas involved in the proof have been instantiated on a toy election instance.

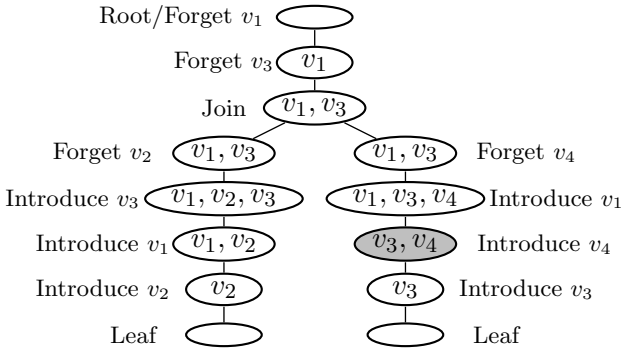
We start with the necessary notation. For any $t \in V(T)$, let $\mathcal{A}_t$ denote the restriction of the space $\mathcal{A}$ to the subgraph induced by the node t and its descendants in T. That is, $\mathcal{A}_t$ is the graph $\mathcal{A}$ restricted to the vertex set $B_t \cup \{\bigcup B_{t'} : t' \in \mathrm{desc}_T(t)\}$. Let $s_i^{\mathrm{NM}} \in \mathbb{R}$ denote the pBorda score of candidate i (corresponding to vertex v_i) resulting from the votes of the non-manipulators. For any candidate i, let $\mathcal{L}_i$ denote the set of all pBorda score values that i can attain due to all possible votes of the manipulator over $\mathcal{A}$. For example, in Figure 9c, the set of all pBorda score values that can be attained by v_4 is $\{-1, -3/2\}$. It can be shown that the size of the set $\mathcal{L}_i$, for any candidate i, is at most Δ^{3d}.[21] Similarly, for each bag B_t, let $S_t \subseteq \bigtimes_{v_i \in B_t} \mathcal{L}_i$ be the set of all tuples of pBorda scores of the vertices in B_t resulting from all possible votes of the manipulator. Thus, $|S_t| \leq \Delta^{3d(w+1)}$ for any $t \in T$. We will call the set S_t the *score set* of node t.

In addition to the score sets, we will require the notion of an *orientation function*, which prescribes an orientation for the edges in a graph (corresponding to a vote of the manipulator), and an *update function*, which specifies the change in pBorda scores of the vertices resulting

[21] This is because a vertex v_i is adjacent to at most Δ edges in $\mathcal{A}$, and there are only d distinct types of score transfer vectors possible across these edges. For each possible score transfer vector, the manipulator has three available choices, amounting to at most $3d$ distinct score exchanges overall. Hence, the total number of distinct score values that v_i can take is at most the number of ways in which Δ identical balls can be thrown into $3d$ bins.



(a) *Action space* $\mathcal{A}$



(b) *A nice tree decomposition of* $\mathcal{A}$

Set of all possible pBorda score values attainable by $v_3 = \{1/2, 0, -1/2, -1\}$
Set of all possible pBorda score values attainable by $v_4 = \{-1, -3/2\}$

Score sets for (v_3, v_4)	$v_3 \succ v_4$	skip	$v_4 \succ v_3$	realized by?
$(1/2, -1)$	0	0	0	Not realizable
$(1/2, -3/2)$	1	0	0	$v_3 \succ v_4$, skip other comparisons
$(0, -1)$	0	1	1	(i) skip all comparisons (ii) $v_4 \succ v_3$, skip others
$(0, -3/2)$	1	0	0	$v_3 \succ v_4, v_2 \succ v_3$, skip others
$(-1/2, -1)$	0	1	1	(i) $v_1 \succ v_3$, skip others (ii) $v_4 \succ v_3, v_1 \succ v_3$, skip others
$(-1/2, -3/2)$	1	0	0	$v_3 \succ v_4, v_1 \succ v_3, v_2 \succ v_3$, skip others
$(-1, -1)$	0	1	1	(i) $v_1 \succ v_3, v_2 \succ v_3$, skip others (ii) $v_4 \succ v_3, v_1 \succ v_3, v_2 \succ v_3$, skip others
$(-1, -3/2)$	0	0	0	Not realizable

(c) *DP table for the bag* $\{v_3, v_4\}$

Figure 9: *Illustrating the main ideas in the proof of Theorem 16 on a toy instance.* *Subfigure (a)* shows the action space $\mathcal{A}$ of the manipulator for an election instance consisting of the candidates $i^*, v_1, \ldots, v_4$. The solid edges indicate the comparisons that the manipulator is allowed to make. The label "$a : b$" for an edge (v_i, v_j) means the pair is in $a : b$ configuration (based on the votes of the non-manipulators). For example, the pair (v_1, v_3) is in $0 : 1$ configuration. The pBorda scores of the candidates are mentioned in square brackets. *Subfigure (b)* shows a nice tree decomposition of $\mathcal{A}$, starting from the leaf nodes at the bottom (denoted by the empty bags) and ending with the root node at the top (also empty). Observe that this decomposition satisfies all the properties stated in Section 2.7. *Subfigure (c)* shows the DP table for the bag $\{v_3, v_4\}$ shaded in gray in Subfigure (b). The rows of this table denote all possible score combinations for the vertices in this bag. The second, third and fourth columns consist of binary indicators denoting whether a particular score tuple can be realized by a vote of the manipulator that is consistent with the orientation specified by the column. Thus, for instance, the score tuple in the first row is not realizable by any vote of the manipulator. This is because the score of v_4 is fixed at -1, which already rules out $v_3 \succ v_4$. Hence, the manipulator must either skip the comparison, or vote $v_4 \succ v_3$, neither of which change the pBorda score of either v_3 or v_4. Next, observe that even if v_3 wins the comparisons against v_1 and v_2, its pBorda score cannot increase. Thus, the largest pBorda score that v_3 can attain in this case is 0. As a result, the score tuple $(1/2, -1)$ is not realizable under any orientation of the edges in the bag $\{v_3, v_4\}$.

from a given orientation. Formally, an orientation function $\phi_G : E(G) \to \{-1, 0, +1\}$ for a graph G assigns an orientation for each edge in $E(G)$. The range $\{-1, 0, +1\}$ encodes the three choices available to the manipulator for each edge. Denote by Φ_G the space of all orientations of the set of edges $E(G)$. For a fixed profile Π of the votes of the non-manipulators, an *update function* $u_G : V(G) \times \Phi_G \to \mathbb{R}$ takes as input a vertex $v \in V(G)$ and an orientation $\phi_G \in \Phi_G$ of graph G, and returns the *change* in the pBorda score of v resulting from the vote ϕ_G of the manipulator. For notational simplicity, we will write ϕ instead of ϕ_G whenever the underlying graph G is clear from the context.

We will now describe the information stored at each node of T (refer to Figure 9c for an example). Essentially, at each node t, the algorithm stores a binary table $\mathbb{T}_t$, where the rows correspond to the score vectors (i.e., elements of the *score set* S_t) and the columns correspond to orientations of the edges inside the corresponding bag B_t. The $(i, j)^{\text{th}}$ entry of this table, namely $\mathbb{T}_t(i, j)$, is 1 if the score vector in row i is *realizable* under the orientation in column j, and is 0 otherwise. A score vector $(s_1, \ldots, s_\tau) \in S_t$ for the vertices $(v_1, \ldots, v_\tau)$ of a bag B_t is said to be *realizable* under an orientation ϕ_{B_t} if there exists an orientation $\phi_{\mathcal{A}_t} \in \Phi_{\mathcal{A}_t}$ of the induced subgraph $\mathcal{A}_t$ for which the following conditions hold:

1. The projection of $\phi_{\mathcal{A}_t}$ onto the bag B_t is precisely ϕ_{B_t}.
2. For each vertex $v_i \in B_t$, the corresponding score value s_i should be attained under $\phi_{\mathcal{A}_t}$. That is,
$$s_i^{\text{NM}} + u_{\mathcal{A}_t}(v_i, \phi_{\mathcal{A}_t}) = s_i.$$
3. For each vertex $v_j \in \mathcal{A}_t \setminus B_t$, the excess score constraint should be satisfied. That is,
$$s_j^{\text{NM}} + u_{\mathcal{A}_t}(v_j, \phi_{\mathcal{A}_t}) \leq s^*,$$
where s^* is the (fixed) pBorda score of the distinguished candidate i^*.

Thus, the largest DP table has at most $\Delta^{3d(w+1)}$ rows and $3^{(w+1)^2}$ columns, hence $\mathcal{O}(\Delta^{dw^2})$ bits in size overall. Note that if an entry in the table $\mathbb{T}_r$ for the root node r is 1, then there must be a score vector (i.e., a row in the table) that is realizable by some vote of the manipulator (i.e., a column) over the entire space $\mathcal{A}$, implying that pBorda-Manipulation has a valid YES instance. This is because the root node in a nice tree decomposition corresponds to an empty set, and thus $\mathcal{A}_r = \mathcal{A}$.

We will now describe the procedure for updating the DP tables for all node types.

1. *Introduce node*: Let t be a node that introduces a vertex v, and let t' be the unique child of t in T. Let $B_{t'} = \{v_1, \ldots, v_\tau\}$ and, as a result, $B_t = \{v_1, \ldots, v_\tau, v\}$. As described above, the rows of $\mathbb{T}_t$ correspond to elements of S_t and the columns correspond to orientations of edges in B_t. Since t is the parent node of t', any orientation ϕ of the edges in B_t induces an orientation $\phi_{B_{t'}}$ of the edges in $B_{t'}$. Thus, by looking at all possible ways in which the edges in B_t adjacent to v can be oriented (call any such orientation ϕ_v) and checking realizability for each such orientation, we can construct the DP table for node t as follows:
$$\mathbb{T}_t((s_1, \ldots, s_\tau, s_v), \phi) = \begin{cases} 1 & \text{if } s_v^{\text{NM}} + u_{B_t}(v, \phi) = s_v \text{ and } \mathbb{T}_{t'}((s'_1, \ldots, s'_\tau), \phi_{B_{t'}}) = 1 \\ & \text{where } s'_i = s_i - u_{B_t}(v_i, \phi) \text{ for all } v_i \in B_t \setminus \{v\} \text{ and} \\ & \phi_{B_{t'}} \cup \phi_v = \phi; \\ 0 & \text{otherwise.} \end{cases}$$
Here, the condition $s_v^{\text{NM}} + u_{B_t}(v, \phi) = s_v$ checks whether v attains the score s_v under the orientation ϕ, while the condition $\mathbb{T}_{t'}((s'_1, \ldots, s'_\tau), \phi_{B_{t'}}) = 1$ checks for realizability of the

score vector $(s'_1, \ldots, s'_\tau)$ under the orientation $\phi_{B_{t'}}$ obtained by restricting ϕ to the child node $B_{t'}$. The time taken in updating each entry of the table $\mathbb{T}_t$ is dominated by the computation of the update function, which is $\mathcal{O}(w \log m)$, since computing the updated score for each candidate pair requires $\mathcal{O}(\log m)$ effort, and there are at most $\mathcal{O}(w)$ such pairs. Therefore, DP update for an introduce node takes $\mathcal{O}(\Delta^{dw^2} w \log m)$ time overall.

2. *Forget node*: Let t be a node that forgets a vertex v, and let t' be the unique child of t in T. Let $B_{t'} = \{v_1, \ldots, v_\tau, v\}$ and, as a result, $B_t = \{v_1, \ldots, v_\tau\}$. The DP updates for the forget node are similar to those for the introduce node described above, since we once again iterate over all possible orientations ϕ_v (this time in $B_{t'}$) and check if the adjusted score vector has an entry with 1 in the child DP table. Thus,

$$\mathbb{T}_t((s_1, \ldots, s_\tau), \phi) = \begin{cases} 1 & \text{if } \; s'_v \leq s^* \text{ and } \mathbb{T}_{t'}((s'_1, \ldots, s'_\tau, s'_v), \phi_{B_{t'}}) = 1 \\ & \text{where } s'_i = s_i + u_{B_{t'}}(v_i, \phi_{B_{t'}}) \text{ for all } v_i \in B_t \\ & \text{and } \phi_{B_{t'}} = \phi \cup \phi_v; \\ 0 & \text{otherwise.} \end{cases}$$

   For a similar reason as before, this step takes $\mathcal{O}(\Delta^{dw^2} w \log m)$ time.

3. *Join node*: Let t be a node that joins the child nodes t_L and t_R such that $B_t = B_{t_L} = B_{t_R} = \{v_1, \ldots, v_\tau\}$. We will once again iterate over all possible orientations of the edges of the bag and check if a pair of score assignments – one for each child bag – add up to the score assignment for the parent node under the given orientation. Thus,

$$\mathbb{T}_t((s_1, \ldots, s_\tau), \phi) = \begin{cases} 1 & \text{if } \; \mathbb{T}_{t_L}((s'_1, \ldots, s'_\tau), \phi) = 1 \text{ and } \mathbb{T}_{t_R}((s''_1, \ldots, s''_\tau), \phi) = 1 \\ & \text{where } s'_i + s''_i = s_i \text{ for all } v_i \in B_t; \\ 0 & \text{otherwise.} \end{cases}$$

   The time taken is $\mathcal{O}(\Delta^{dw^2} \cdot \Delta^{dw^2})$, since we need to scan an entire column of the DP table of one of the child nodes to iterate over all s'_i (or s''_i), which requires $\mathcal{O}(\Delta^{dw^2})$ time per entry of the parent table. Thus, $\mathcal{O}(\Delta^{\mathcal{O}(dw^2)})$ time overall.

Finally, we can assume, without loss of generality, that any nice tree decomposition with treewidth w has at most $\mathcal{O}(w \cdot n)$ nodes (Cygan et al., 2015). Therefore, the overall running time of our algorithm is $\mathcal{O}(\Delta^{\mathcal{O}(dw^2)} \cdot (w \log m) \cdot (w \cdot n))$, or more concisely $\mathcal{O}\big(\Delta^{\mathcal{O}(dw^2)} (n \log m)^{\mathcal{O}(1)}\big)$, as desired. □

Note that since $d \leq \Delta$ for the pBorda rule, the running time above is FPT in Δ and the *treewidth* of $\mathcal{A}$. Moreover, since $\Delta \leq n$, the running time is also in XP with respect to the *diversity* d and *treewidth* w of $\mathcal{A}$. We restate these observations as the following corollary:

Corollary 17. pBorda-Manipulation *is* FPT *when parameterized by maximum degree and treewidth of $\mathcal{A}$, and in XP when parameterized by diversity and treewidth of $\mathcal{A}$.*

Our next algorithmic result pertains to graphs of bounded *vertex cover* and bounded *diversity.*

Theorem 18. pBorda-Manipulation *is solvable in time $\mathcal{O}(f(k,d)(n \log m)^{\mathcal{O}(1)})$, where k and d denote the size of a minimum vertex cover and the diversity of $\mathcal{A}$ respectively, and $f : \mathbb{N}^2 \to \mathbb{N}$ is a computable function.*

Proof. At a high level, the proof proceeds by partitioning the vertices of the independent set of $\mathcal{A}$ into equivalence classes based on their interactions with the vertex cover, and exploiting a size bound on the number of such equivalence classes in the subsequent Integer Linear Programming formulation. Figure 10 illustrates the main ideas involved in the proof via a toy example. As always, we will assume upfront that the distinguished candidate i^* wins all its pairwise comparisons in the manipulator's vote, thereby freezing its pBorda score s^*.

We start with the necessary notation. Let $S \subseteq V(\mathcal{A})$ be a *vertex cover* of $\mathcal{A}$ of size k, and let $I = V(\mathcal{A}) \setminus S$ be the corresponding independent set. We begin by partitioning the vertices of the independent set I into equivalence classes based on their interactions with the *vertex cover* (refer to Figure 10a for an example). In particular, for any subset $T \subseteq S$ of the *vertex cover*, let $I_T \subseteq I$ denote the set of all vertices of the independent set whose neighborhood within $\mathcal{A}$ is exactly the set T.[22] That is, $I_T = \{u \in I \,|\, \mathcal{N}(u) \cap \mathcal{A} = T\}$. Next, given $T = \{v_1, \ldots, v_t\} \subseteq S$ and a vector $E_T = \langle(\alpha_1, \beta_1), \ldots, (\alpha_t, \beta_t)\rangle$ consisting of pairs of non-negative integers (α_i, β_i), define the *equivalence class* $I_{T,E_T} \subseteq I$ as the set of all vertices of the independent set with the following properties:

- each vertex of I_{T,E_T} has the exact same neighborhood in the *vertex cover* (i.e. the set T), and
- for any pair of vertices $v_i \in T$ and $u \in I_{T,E_T}$, the candidate pair (v_i, u) is in $\alpha_i : \beta_i$ configuration with respect to the votes of the non-manipulators.

Thus, given a preference profile Π,

$$I_{T,E_T} = \{u \in I_T \,|\, m_{v_i,u}(\Pi) = \alpha_i \text{ and } m_{u,v_i}(\Pi) = \beta_i \text{ for all } i \in [t]\}.$$

Note that since the *diversity* d of the instance is bounded, any vertex of the independent set must belong to exactly one of at most $2^k \cdot d^k$ equivalence classes. This is because there are 2^k choices for T, and for each such choice, there are at most d^k vectors E_T.

Call a vertex $u \in I_{T,E_T}$ *safe* with respect to a vector $z \in \{-1, 0, 1\}^{|T|}$ if the excess score of u remains zero or less for the following vote of the manipulator: For each $v_i \in T$,

- $u \succ v_i$ if $z(i) = +1$,
- skip the comparison between u and v_i if $z(i) = 0$, and
- $v_i \succ u$ if $z(i) = -1$.

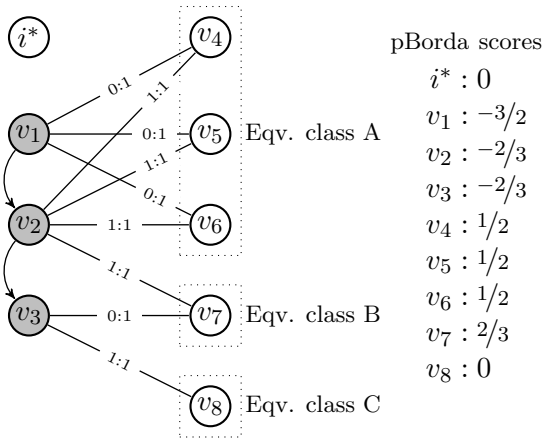
Since the pBorda score of u can only be affected via the comparisons in $\mathcal{A}$ that involve u, fixing the vote vector for such pairs fixes the pBorda score of u. Similarly, define the *safety-set* of a vertex $u \in I_{T,E_T}$ as the set of all vectors $z \in \{-1, 0, 1\}^{|T|}$ with respect to which u is *safe* (refer to Figure 10b for an example).

We say that a vertex $u \in I_{T,E_T}$ *sees* a vote $z \in \{-1, 0, 1\}^{|T|}$ of the manipulator if z is the restriction of the manipulator's vote to the pairwise comparisons in $\mathcal{A}$ involving u. Hence, another way of saying that a vertex u is *safe* is if it *sees* a vector z that keeps its pBorda score below that of i^*.

Given an equivalence class I_{T,E_T} and the safety-set for each $u \in I_{T,E_T}$, define a *safe-subclass* as the set of all vertices in the equivalence class I_{T,E_T} with identical safety-sets. We denote the number of safe-subclasses in I_{T,E_T} by N_{T,E_T}. Thus, $N_{T,E_T} \leq 3^k$, since the number of safe-subclasses within an equivalence class is at most the number of distinct vectors $z \in \{-1, 0, 1\}^{|T|}$ *seen* by any vertex in it, which is at most 3^k.

We will now make a key claim regarding the structure of the solutions: *For any valid solution of* pBorda-Manipulation*, there exists another (possibly different) valid solution where all*

[22] The *neighborhood of a vertex u within $\mathcal{A}$*, denoted by $\mathcal{N}(u) \cap \mathcal{A}$, is the set of all vertices in $V(\mathcal{A})$ that are adjacent to u in $\mathcal{A}$. That is, $\mathcal{N}(u) \cap \mathcal{A} = \{v \in V(\mathcal{A}) : \{u, v\} \in \mathcal{A}\}$.



pBorda scores

$i^* : 0$
$v_1 : -3/2$
$v_2 : -2/3$
$v_3 : -2/3$
$v_4 : 1/2$
$v_5 : 1/2$
$v_6 : 1/2$
$v_7 : 2/3$
$v_8 : 0$

(a) *Action space $\mathcal{A}$ and* pBorda *scores*

Eqv. class A $= \{v_4, v_5, v_6\}$

Safety set $= \{\{-1,-1\},\{-1,0\}\}$
with respect to $T = \{v_1, v_2\}$.

Eqv. class B $= \{v_7\}$

Safety set $= \{\{-1,-1\}\}$
with respect to $T = \{v_2, v_3\}$.

Eqv. class C $= \{v_8\}$

Safety set $= \{\{-1\},\{0\}\}$
with respect to $T = \{v_3\}$.

(b) *Safety sets for the equivalence classes*

Constraints for v_1:
$-3/2 + Z_{\{v_1,v_2\},\langle(0:1,1:1)\rangle,\{-1,-1\},A} \cdot \left(\frac{1}{(0+1)(0+1+1)} \cdot |A|\right) + Z_{\{v_1,v_2\},\langle(0:1,1:1)\rangle,\{-1,0\},A} \cdot (0 \cdot |A|) \leq 0.$

Constraints for v_2:
$-2/3 + Z_{\{v_1,v_2\},\langle(0:1,1:1)\rangle,\{-1,-1\},A} \cdot \left(\frac{1}{(1+1)(1+1+1)} \cdot |A|\right) + Z_{\{v_1,v_2\},\langle(0:1,1:1)\rangle,\{-1,0\},A} \cdot (0 \cdot |A|) +$
$Z_{\{v_2,v_3\},\langle(1:1,0:1)\rangle,\{-1,-1\},B} \cdot \left(\frac{1}{(1+1)(1+1+1)} \cdot |B|\right) \leq 0.$

Constraints for v_3:
$-2/3 + Z_{\{v_2,v_3\},\langle(1:1,0:1)\rangle,\{-1,-1\},B} \cdot \left(\frac{1}{(0+1)(0+1+1)} \cdot |B|\right) + Z_{\{v_3\},\langle(1:1)\rangle,\{-1\},C} \cdot \left(\frac{1}{(1+1)(1+1+1)} \cdot |C|\right) +$
$Z_{\{v_3\},\langle(1:1)\rangle,\{0\},C} \cdot (0 \cdot |C|) \leq 0.$

(c) *Excess score constraints in the ILP formulation*

Figure 10: *Illustrating the main ideas in the proof of Theorem 18 on a toy instance.* *Subfigure (a)* shows the action space $\mathcal{A}$ of the manipulator, consisting of the candidates $v_1, \ldots, v_8$ and the distinguished candidate i^*. Each solid edge indicates a pairwise comparison that the manipulator is allowed to make. The label "$a : b$" for an edge (v_i, v_j) means the pair is in $a : b$ configuration (based on the votes of the non-manipulators). For example, the pair (v_1, v_4) is in $0 : 1$ configuration. The shaded vertices v_1, v_2, v_3 indicate the minimum vertex cover S of $\mathcal{A}$, and the directed edges denote the manipulator's guess $\succ_S$ for the comparisons within S. The table next to the action space shows the pBorda scores of all candidates *after* the manipulator has made the guess $\succ_S$. Notice that all vertices in the set $\{v_4, v_5, v_6\}$ are adjacent to the same subset of S (namely, $\{v_1, v_2\}$), and also have identical interactions with this set (namely, via $0 : 1$ and $1 : 1$ edges). As a result, the set $\{v_4, v_5, v_6\}$ constitutes an equivalence class (denoted by 'Eqv. class A'). Similarly, the singleton sets $\{v_7\}$ and $\{v_8\}$ correspond to the equivalence classes B and C respectively. *Subfigure (b)* specifies the set of votes that are *safe* (i.e., the safety set) for each equivalence class. Thus, for instance, for the equivalence class A, it is safe for each vertex to either lose against v_1 and draw against v_2 (this corresponds to the vote $\{-1, 0\}$), or lose against both v_1 and v_2 (this corresponds to the vote $\{-1, -1\}$). *Subfigure (c)* describes the excess score constraints for all vertices in the vertex cover. In particular, for the vertex v_2, we start from its current pBorda score (which is $-2/3$), and consider all possible changes in its pBorda score that can be brought about by its interactions with the equivalence classes A and B. The former is captured by the binary variables $Z_{\{v_1,v_2\},\langle(0:1,1:1)\rangle,\{-1,-1\},A}$ and $Z_{\{v_1,v_2\},\langle(0:1,1:1)\rangle,\{-1,0\},A}$, while the latter is captured by the binary variable $Z_{\{v_2,v_3\},\langle(1:1,0:1)\rangle,\{-1,-1\},B}$.

vertices inside each safe-subclass see *the same vote vector*. Indeed, for any valid vote $\succ$ and for any given safe-subclass, let z' be the restriction of $\succ$ as *seen* by the vertex with the highest excess score in that safe-subclass (if there are many such vertices, pick one arbitrarily). Then, another valid vote can be constructed as follows: in the original vote $\succ$, replace the pairwise comparisons corresponding to the vote vector currently *seen* by each vertex inside the safe-subclass by z', while keeping the rest of the vote unchanged. The excess score constraints for all the vertices inside the safe-subclass continue to remain satisfied since, by definition of a safe-subclass, the vector z' belongs to the safety-set of all such vertices. Similarly, excess scores constraints for all the other vertices in the independent set continue to remain satisfied since their scores are unaffected by this change. Finally, the excess score constraints for all vertices in the *vertex cover* continue to remain satisfied since they do not gain any additional score in the above process. Therefore, without loss of generality, all vertices inside the same safe-subclass *see* the same vote of the manipulator. This observation is at the heart of our Integer Linear Programming formulation, since it allows us to simply define a variable for each safe-subclass, instead of for each vertex in the independent set. The former is purely a function of d and k, while the latter can be a function of n.

We are now ready to describe our algorithm for pBorda-Manipulation. Our algorithm takes as input an instance of pBorda-Manipulation, namely $\langle \Pi, i^*, \mathcal{A} \rangle$ (we assume `pref-type` = unrestricted), and returns a YES/NO output indicating the existence of a valid manipulative vote (along with a valid vote, if one exists). The algorithm starts by *guessing* the manipulator's vote within the *vertex cover*. For each such guess, the algorithm uses Integer Linear Programming (ILP) to solve separately for each equivalence class, and checks if the combined vote constitutes a valid solution. If yes, the algorithm terminates with a YES output along with the manipulative vote; otherwise the algorithm returns NO. We will first describe the ILP formulation, followed by arguing the correctness of the algorithm and analyzing its running time.

1. *Guessing the solution inside the vertex cover*: We begin by guessing the solution $\succ_S$ restricted to the pairs $\{u, v\} \in \mathcal{A}$ such that $u \in S$ and $v \in S$. Since there are at most $\binom{k}{2}$ such pairs, the total number of choices is at most $3^{\mathcal{O}(k^2)}$. For each such guess $\succ_S$, we obtain a new instance of pBorda-Manipulation, namely $\langle \Pi', i^*, \mathcal{A}' \rangle$, where Π' is a voting profile consisting of the original votes of the non-manipulators combined with the manipulator's vote $\succ_S$ over the *vertex cover*, and $\mathcal{A}'$ represents the restriction of the graph $\mathcal{A}$ to the bipartite subgraph $S \times I$.[23]

2. *Formulating the ILP*: We will now describe the variables and constraints for the ILP.

   Variables: For each subset $T \subseteq S$, each score vector $E_T = \langle (\alpha_1, \beta_1), \ldots, (\alpha_t, \beta_t) \rangle$, each $1 \leq p \leq 3^{|T|}$ and each $1 \leq q \leq N_{T,E_T}$, we define a binary variable $Z_{T,E_T,p,q} \in \{0, 1\}$. Here $Z_{T,E_T,p,q} = 1$ (respectively 0) indicates that given the set T and the score vector E_T (and hence the induced equivalence class I_{T,E_T}), the safe-subclass indexed by q *sees* (respectively *does not see*) the vote vector indexed by p. Thus, there are at most $2^k \cdot d^k \cdot 3^{2k}$ variables overall. Note that the number of variables is purely a function of d and k, and does not depend on the input size n.

   Constraints: Our ILP formulation consists of three types of constraints.

   (a) *Sanity constraints*:

      (i) $Z_{T,E_T,p,q} \in \{0, 1\}$ for all T, E_T, p and q.

[23] Notice that $S \times I$ is bipartite because the manipulator has already made a guess $\succ_S$ about all pairs of vertices in S that are adjacent in $\mathcal{A}$ (hence, these edges are not a part of the new action space $\mathcal{A}'$) and the vertices in I, by definition, cannot be adjacent in $\mathcal{A}$ (or $\mathcal{A}'$).

(ii) For every T, E_T and q, $\sum_p Z_{T,E_T,p,q} = 1$ (requiring that each safe-subclass *sees* exactly one vote vector).

(b) *Excess score constraints for the vertex cover*: For each vertex $v_i \in S$,

$$s_{v_i} + \sum_{T \in T_v} \sum_{E_T} \sum_q \sum_p Z_{T,E_T,p,q} \cdot [p_i = -1] \cdot \tfrac{\beta_i}{(\alpha_i+\beta_i)(\alpha_i+\beta_i+1)} \cdot |q| + Z_{T,E_T,p,q} \cdot [p_i = 0] \cdot 0 + Z_{T,E_T,p,q} \cdot [p_i = +1] \cdot \tfrac{-\alpha_i}{(\alpha_i+\beta_i)(\alpha_i+\beta_i+1)} \cdot |q| \leq s^*,$$

where $[\cdot]$ is the Iverson bracket, s_{v_i} is the pBorda score of v_i due to the votes of the non-manipulators, $T_v = \{T \subseteq S \mid v \in T\}$, p_i is the i^{th} element of the vector indexed by p, and $|q|$ represents the cardinality of the safe-subclass indexed by q. The latter can be efficiently precomputed. Recall that the terms involving α_i and β_i in the above inequality correspond to the score transfers under the pBorda rule (refer to Section 2.4). Figure 10c shows the excess score constraints for a small election instance.

(c) *Excess score constraints for the independent set*: For all T, E_T, p and q,

$$Z_{T,E_T,p,q} \leq Z^{\text{safe}}_{T,E_T,p,q},$$

where $Z^{\text{safe}}_{T,E_T,p,q} \in \{0,1\}$ is a (precomputed) binary indicator which is 1 whenever, given T and E_T, the vector indexed by p belongs to the safety-set of (any vertex in) the safe-subclass indexed by q; and is 0 otherwise.

Running time: As remarked earlier, the number of variables that we have introduced is a function of d and k alone (instead of the input size n). Therefore, from the result of Lenstra Jr (1983), the Integer Linear Programming subroutine (and, as a result, the overall algorithm) is FPT in these parameters.

Correctness: If the above algorithm returns a set of binary outputs $Z_{T,E_T,p,q}$, then a valid manipulative vote can be uniquely constructed from these values given the correctness of the *sanity* and *excess score* constraints. Conversely, if there exists a valid vote, then there also exists a feasible solution to the above ILP. In this case, by the correctness of the ILP solver, we will obtain a (possibly different) feasible output that can, as before, be uniquely transformed into a manipulative vote. This completes the proof of Theorem 18. □

5 Concluding Remarks

We considered the phenomenon of manipulation in the model of pairwise preferences and showed results of both axiomatic and computational flavor. In particular, we provided a complete understanding of how the computational complexity of manipulating the pairwise Borda (pBorda) rule and the Copeland$^\alpha$ family of pairwise voting rules is influenced by the action space of the manipulator and the type of preference relation, up to the point of distinguishing the polynomial time cases from the NP-complete ones.

We also took a closer look at the case when there is no restriction on the nature of the manipulator's vote, and gave a complete parameterized classification based on various natural structural parameters relating to the action space. This involved the introduction of *diversity* as a parameter, which we demonstrated to be useful from an algorithmic perspective.

Our results extend quite naturally to many other closely related problems such as *destructive manipulation* (i.e., making the distinguished candidate lose), or generalizations such as *top-k manipulation* and *bottom-k manipulation* (i.e., ensuring a top-k or bottom-k finish for the distinguished candidate). Furthermore, many of our results rely only on a certain *locality property* of the pairwise voting rule, and turn out to be true for any pairwise voting rule with this property.

The most natural direction for future research would be to understand the complexity of manipulation in the pairwise preference model for other voting rules such as PageRank (Brin and Page, 2012), HodgeRank (Jiang et al., 2011), Ranked Pairs and Schulze's rule (Parkes and Xia, 2012). An ambitious question in this context would be a complete classification of the complexity of manipulation by a single voter in the space of all pairwise voting rules. The question of manipulation by a coalition of voters, both in weighted and unweighted elections, is also interesting.

Acknowledgments

Preliminary versions of this paper have appeared at the 15$^{\text{th}}$ International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016) (Vaish et al., 2016) and the 3$^{\text{rd}}$ Workshop on Exploring Beyond the Worst Case in Computational Social Choice (EXPLORE 2016) (Vaish and Misra, 2016). The authors thank the various anonymous reviewers for their careful reading of earlier drafts of this paper and several helpful suggestions. This work was supported in part by an Indo-US Joint Center grant (SA,AB), a DST INSPIRE Faculty grant [IFA12-ENG-31] (NM), DST Ramanujan Fellowship (SA), and NSF grant CCF-1525971 (AB). Part of this work was done when RV was visiting Carnegie Mellon University as part of a student visit under the Indo-US Joint Center for Advanced Research in Machine Learning, Game Theory and Optimization supported by the Indo-US Science and Technology Forum. RV also thanks Google and Microsoft Research for travel grants to present parts of this work at AAMAS 2016 and EXPLORE 2016.

References

Nir Ailon. Aggregation of Partial Rankings, p-Ratings and Top-m Lists. *Algorithmica*, 57(2): 284–300, 2010. (Cited on page 2)

Nir Ailon. An Active Learning Algorithm for Ranking from Pairwise Preferences with an Almost Optimal Query Complexity. *Journal of Machine Learning Research*, 13(1):137–164, 2012. (Cited on page 2)

Kenneth J Arrow, Amartya Sen, and Kotaro Suzumura. *Handbook of Social Choice & Welfare*, volume 2. Elsevier, 2010. (Cited on pages 1 and 5)

Navin Aswal, Shurojit Chatterji, and Arunava Sen. Dictatorial Domains. *Economic Theory*, 22 (1):45–62, 2003. (Cited on page 17)

Haris Aziz, Markus Brill, Felix Fischer, Paul Harrenstein, Jérôme Lang, and Hans Georg Seedig. Possible and Necessary Winners of Partial Tournaments. *Journal of Artificial Intelligence Research*, 54:493–534, 2015. (Cited on pages 15 and 16)

John J Bartholdi III and James B Orlin. Single Transferable Vote Resists Strategic Voting. *Social Choice and Welfare*, 8(4):341–354, 1991. (Cited on page 2)

John J. Bartholdi III, Craig A. Tovey, and Michael A. Trick. Voting Schemes for Which It Can Be Difficult to Tell Who Won the Election. *Social Choice and Welfare*, 6(2):157–165, 1989a. (Cited on pages 17 and 39)

John J. Bartholdi III, Craig A. Tovey, and Michael A. Trick. The Computational Difficulty of Manipulating an Election. *Social Choice and Welfare*, 6(3):227–241, 1989b. (Cited on pages 1, 2, 17, and 20)

Dorothea Baumeister and Jörg Rothe. Taking the Final Step to a Full Dichotomy of the Possible Winner Problem in Pure Scoring Rules. *Information Processing Letters*, 112(5): 186–190, February 2012. (Cited on page 15)

Dorothea Baumeister, Piotr Faliszewski, Jérôme Lang, and Jörg Rothe. Campaigns for Lazy Voters: Truncated Ballots. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 577–584, 2012. (Cited on pages 16 and 17)

Nadja Betzler and Britta Dorn. Towards a Dichotomy for the Possible Winner Problem in Elections Based on Scoring Rules. *Journal of Computer and System Sciences*, 76(8):812–836, 2010. (Cited on page 15)

Nadja Betzler, Michael R Fellows, Jiong Guo, Rolf Niedermeier, and Frances A Rosamond. Fixed-parameter Algorithms for Kemeny Rankings. *Theoretical Computer Science*, 410(45): 4554–4570, 2009a. (Cited on pages 17 and 39)

Nadja Betzler, Susanne Hemmann, and Rolf Niedermeier. A Multivariate Complexity Analysis of Determining Possible Winners Given Incomplete Votes. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, volume 9, pages 53–58, 2009b. (Cited on page 17)

Nadja Betzler, Jiong Guo, and Rolf Niedermeier. Parameterized Computational Complexity of Dodgson and Young Elections. *Information and Computation*, 208(2):165–177, 2010. (Cited on page 17)

Nadja Betzler, Rolf Niedermeier, and Gerhard J Woeginger. Unweighted Coalitional Manipulation Under the Borda Rule Is NP-Hard. In *Proceedings of the 22th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 11, pages 55–60, 2011. (Cited on page 17)

Nadja Betzler, Robert Bredereck, Jiehua Chen, and Rolf Niedermeier. Studies in Computational Aspects of Voting: A Parameterized Complexity Perspective. In *The Multivariate Algorithmic Revolution and Beyond*, pages 318–363, 2012. (Cited on page 17)

Nadja Betzler, Arkadii Slinko, and Johannes Uhlmann. On the Computation of Fully Proportional Representation. *Journal of Artificial Intelligence Research*, 47:475–519, 2013. (Cited on page 17)

H.L. Bodlaender, P.G. Drange, M.S. Dregi, F.V. Fomin, D. Lokshtanov, and M. Pilipczuk. A $c^k n$ 5-Approximation Algorithm for Treewidth. In *54th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 499–508, 2013. (Cited on page 11)

Felix Brandt, Vincent Conitzer, Ulle Endriss, Ariel D Procaccia, and Jérôme Lang. *Handbook of Computational Social Choice*. Cambridge University Press, 2016. (Cited on page 1)

Robert Bredereck, Jiehua Chen, Piotr Faliszewski, André Nichterlein, and Rolf Niedermeier. Prices Matter for the Parameterized Complexity of Shift Bribery. *Information and Computation*, 251:140–164, 2016. (Cited on page 17)

Sergey Brin and Lawrence Page. Reprint of: The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks*, 56(18):3825–3833, 2012. (Cited on pages 6 and 56)

Katarina Cechlárová, Eva Potpinková, and Ildikó Schlotter. Refining the Complexity of the Sports Elimination Problem. *Discrete Applied Mathematics*, 199:172 – 186, 2016. (Cited on pages 39, 42, and 48)

Jianer Chen, Iyad A Kanj, and Ge Xia. Improved Upper Bounds for Vertex Cover. *Theoretical Computer Science*, 411(40):3736–3756, 2010. (Cited on page 11)

Xi Chen, Paul N Bennett, Kevyn Collins-Thompson, and Eric Horvitz. Pairwise Ranking Aggregation in a Crowdsourced Setting. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 193–202, 2013. (Cited on page 2)

Vincent Conitzer, Tuomas Sandholm, and Jérôme Lang. When are Elections with Few Candidates Hard to Manipulate? *Journal of the ACM*, 54(3):14, 2007. (Cited on pages 17 and 39)

Marek Cygan, Fedor V Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. (Cited on pages 9, 14, 48, and 51)

Jessica Davies, Nina Narodytska, and Toby Walsh. Eliminating the Weakest Link: Making Manipulation Intractable? In *Proceedings of the 26th Conference on Artificial Intelligence (AAAI)*, pages 1333–1339, 2012. (Cited on page 2)

Jessica Davies, George Katsirelos, Nina Narodytska, Toby Walsh, and Lirong Xia. Complexity of and Algorithms for the Manipulation of Borda, Nanson's and Baldwin's Voting Rules. *Artificial Intelligence*, 217:20–42, 2014. (Cited on pages 2 and 17)

Michael Dom, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. Capacitated Domination and Covering: A Parameterized Perspective. In *Parameterized and Exact Computation*, pages 78–90. 2008. (Cited on page 14)

Britta Dorn and Ildikó Schlotter. Multivariate Complexity Analysis of Swap Bribery. *Algorithmica*, 64(1):126–151, 2012. (Cited on pages 17 and 39)

R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, New York, 1999. (Cited on page 9)

John Duggan and Thomas Schwartz. Strategic Manipulability Without Resoluteness or Shared Beliefs: Gibbard-Satterthwaite Generalized. *Social Choice and Welfare*, 17(1):85–93, 2000. (Cited on page 18)

Edith Elkind and Nisarg Shah. Electing the Most Probable Without Eliminating the Irrational: Voting Over Intransitive Domains. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 182–191, 2014. (Cited on page 2)

Piotr Faliszewski and Ariel D.D Procaccia. AI's War on Manipulation: Are We Winning? *The AI Magazine*, 31(4):53–64, 2010. (Cited on page 1)

Piotr Faliszewski, Edith Hemaspaandra, and Henning Schnoor. Copeland Voting: Ties Matter. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 983–990, 2008. (Cited on page 5)

Piotr Faliszewski, Edith Hemaspaandra, Lane A Hemaspaandra, and Jörg Rothe. The Shield that Never Was: Societies with Single-Peaked Preferences are More Open to Manipulation and Control. In *Proceedings of the 12th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, pages 118–127, 2009a. (Cited on page 17)

Piotr Faliszewski, Edith Hemaspaandra, Lane A Hemaspaandra, and Jörg Rothe. Llull and Copeland Voting Computationally Resist Bribery and Constructive Control. *Journal of Artificial Intelligence Research*, 35:275–341, 2009b. (Cited on pages 32 and 39)

Piotr Faliszewski, Edith Hemaspaandra, and Henning Schnoor. Manipulation of Copeland Elections. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 367–374, 2010. (Cited on page 32)

Piotr Faliszewski, Edith Hemaspaandra, and Lane A Hemaspaandra. The Complexity of Manipulative Attacks in Nearly Single-Peaked Electorates. *Artificial Intelligence*, 207:69–99, 2014. (Cited on page 17)

Michael R Fellows, Bart MP Jansen, Daniel Lokshtanov, Frances A Rosamond, and Saket Saurabh. Determining the Winner of a Dodgson Election is Hard. In *30th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 459–468, 2010. (Cited on page 17)

Zack Fitzsimmons and Edith Hemaspaandra. Complexity of Manipulative Actions When Voting with Ties. In *International Conference on Algorithmic Decision Theory (ADT)*, pages 103–119. 2015. (Cited on page 16)

Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, 2006. (Cited on page 9)

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979. (Cited on pages 9, 14, 15, 23, 26, and 37)

Serge Gaspers, Victor Naroditskiy, Nina Narodytska, and Toby Walsh. Possible and Necessary Winner Problem in Social Polls. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, pages 613–620, 2014. (Cited on page 15)

Allan Gibbard. Manipulation of Voting Schemes: A General Result. *Econometrica*, 41(4): 587–601, 1973. (Cited on pages 1 and 17)

Jiong Guo, Falk Hüffner, and Rolf Niedermeier. A Structural View on Parameterizing Problems: Distance from Triviality. In *Parameterized and Exact Computation*, pages 162–173. 2004. (Cited on page 39)

Edith Hemaspaandra and Lane A Hemaspaandra. Dichotomy for Voting Systems. *Journal of Computer and System Sciences*, 73(1):73–83, 2007. (Cited on page 17)

Edith Hemaspaandra, Lane A Hemaspaandra, and Jörg Rothe. Anyone but him: The Complexity of Precluding an Alternative. *Artificial Intelligence*, 171(5):255–285, 2007. (Cited on page 6)

Xiaoye Jiang, Lek-Heng Lim, Yuan Yao, and Yinyu Ye. Statistical Ranking and Combinatorial Hodge Theory. *Mathematical Programming*, 127(1):203–244, 2011. (Cited on pages 2, 6, and 56)

Walter Kern and Daniël Paulusma. The Computational Complexity of the Elimination Problem in Generalized Sports Competitions. *Discrete Optimization*, 1(2):205–214, 2004. (Cited on pages 12, 22, 23, 26, 34, and 35)

Kathrin Konczak and Jérôme Lang. Voting Procedures with Incomplete Preferences. In *Proceedings of IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling*, volume 20, 2005. (Cited on page 15)

Hendrik W Lenstra Jr. Integer Programming with a Fixed Number of Variables. *Mathematics of Operations Research*, 8(4):538–548, 1983. (Cited on pages 15, 40, and 55)

Claudia Lindner and Jörg Rothe. Fixed-Parameter Tractability and Parameterized Complexity, Applied to Problems From Computational Social Choice. In *Mathematical Programming Glossary*. INFORMS Computing Society, 2008. (Cited on page 17)

Tyler Lu and Craig Boutilier. Learning Mallows Models with Pairwise Preferences. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 145–152, 2011. (Cited on page 2)

Vijay Menon and Kate Larson. Computational Aspects of Strategic Behaviour in Elections with Top-truncated Ballots. *Autonomous Agents and Multi-Agent Systems*, 31(6):1506–1547, 2017. (Cited on pages 2 and 16)

Nina Narodytska and Toby Walsh. The Computational Impact of Partial Votes on Strategic Voting. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI)*, pages 657–662, 2014. (Cited on pages 2 and 16)

Sahand Negahban, Sewoong Oh, and Devavrat Shah. Iterative Ranking from Pair-wise Comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2474–2482, 2012. (Cited on page 2)

Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. (Cited on page 9)

David C Parkes and Lirong Xia. A Complexity-of-Strategic-Behavior Comparison Between Schulze's Rule and Ranked Pairs. In *Proceedings of the 26th Conference on Artificial Intelligence (AAAI)*, pages 1429–1435, 2012. (Cited on pages 6 and 56)

M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Incompleteness and Incomparability in Preference Aggregation: Complexity results. *Artificial Intelligence*, 175(7–8):1272–1289, May 2011. (Cited on page 15)

Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. Aggregating Partially Ordered Preferences. *Journal of Logic and Computation*, 19(3):475–502, June 2009. (Cited on pages 2 and 18)

Anup Pramanik. Further Results on Dictatorial Domains. *Social Choice and Welfare*, 45(2): 379–398, 2015. (Cited on page 17)

Arun Rajkumar and Shivani Agarwal. A Statistical Convergence Perspective of Algorithms for Rank Aggregation from Pairwise Data. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 118–126, 2014. (Cited on pages 2 and 5)

Alexander Reffgen. Generalizing the Gibbard–Satterthwaite Theorem: Partial Preferences, the Degree of Manipulation, and Multi-valuedness. *Social Choice and Welfare*, 37(1):39–59, 2011. (Cited on page 18)

Shin Sato. Circular Domains. *Review of Economic Design*, 14(3-4):331–342, 2010. (Cited on page 17)

M.A. Satterthwaite. Strategy-proofness and Arrow's Conditions: Existence and Correspondence Theorems for Voting Procedures and Social Welfare Functions. *Journal of Economic Theory*, 10(2):187–217, 1975. (Cited on pages 1 and 17)

Benjamin L Schwartz. Possible Winners in Partially Completed Tournaments. *SIAM Review*, 8(3):302–308, 1966. (Cited on page 11)

Arunava Sen. Another Direct Proof of the Gibbard–Satterthwaite Theorem. *Economics Letters*, 70(3):381–385, 2001. (Cited on page 17)

Rohit Vaish and Neeldhara Misra. On the Parameterized Complexity of Manipulating Pairwise Voting Rules. In *Proceedings of the 3rd Workshop on Exploring Beyond the Worst Case in Computational Social Choice (EXPLORE)*, 2016. (Cited on page 56)

Rohit Vaish, Neeldhara Misra, Shivani Agarwal, and Avrim Blum. On the Computational Hardness of Manipulating Pairwise Voting Rules. In *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 358–367, 2016. (Cited on pages 14 and 56)

Toby Walsh. Uncertainty in Preference Elicitation and Aggregation. In *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI)*, volume 7, pages 3–8, 2007. (Cited on page 15)

Toby Walsh. Where are the Hard Manipulation Problems? *Journal of Artificial Intelligence Research*, 42:1–29, 2011. (Cited on page 1)

Jianxin Wang, Min Yang, Jiong Guo, Qilong Feng, and Jianer Chen. Parameterized Complexity of Control and Bribery for d-approval Elections. In *Combinatorial Optimization and Applications*, pages 260–271. 2013. (Cited on page 17)

Lirong Xia. Fixed-Parameter Tractability of Integer Generalized Scoring Rules. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, pages 1599–1600, 2014. (Cited on pages 17 and 39)

Lirong Xia and Vincent Conitzer. Determining Possible and Necessary Winners under Common Voting Rules Given Partial Orders. *Journal of Artificial Intelligence Research*, 41:25–67, 2011. (Cited on pages 15 and 17)

Lirong Xia, Michael Zuckerman, Ariel D Procaccia, Vincent Conitzer, and Jeffrey S Rosenschein. Complexity of Unweighted Coalitional Manipulation under Some Common Voting Rules. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, volume 9, pages 348–352, 2009. (Cited on page 2)

Yongjie Yang. Election Attacks with Few Candidates. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI)*, pages 1131–1132, 2014. (Cited on pages 17 and 39)

Yongjie Yang. Manipulation with Bounded Single-Peaked Width: A Parameterized Study. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 77–85, 2015. (Cited on page 17)

Yongjie Yang and Jiong Guo. Exact Algorithms for Weighted and Unweighted Borda Manipulation Problems. *Theoretical Computer Science*, 622:79–89, 2016. (Cited on page 17)