Efficiently Negative: Complexity and Approximations of Targeted Negative Campaigning

AVISHAI ZAGOURY*, Bar-Ilan University, Israel
ORGAD KELLER, Google Research, Israel
AVINATAN HASSIDIM, Bar-Ilan University; Google Research, Israel
NOAM HAZON, Ariel University, Israel

Given the ubiquity of negative campaigning in recent political elections, we find it important to study its properties from a theoretical computational perspective. To this end, we present a model where elections can be manipulated by convincing voters to demote specific non-favored candidates, and study its properties in the classic setting of scoring rules.

When the goal is constructive (making a preferred candidate win), we prove that finding such a demotion strategy is easy for Plurality and Veto, while generally hard for t-approval and Borda. We also provide a $\min(t, m-t)$-factor approximation for t-approval for every $t \in \{1, \ldots, m-1\}$ (where m is the number of candidates), and a 3-factor approximation algorithm for Borda. Interestingly enough—following recent trends in political science that show that the effectiveness of negative campaigning depends on the type of candidate and demographic—when assigning varying prices to different possible demotion operations, we are able to provide inapproximability results.

When the goal is destructive (making the leading opponent lose), we show that the problem is easy for a broad class of scoring rules and provide an FPTAS for the general case.

JAIR Associate Editor: Piotr Skowron

JAIR Reference Format:
Avishai Zagoury, Orgad Keller, Avinatan Hassidim, and Noam Hazon. 2026. Efficiently Negative: Complexity and Approximations of Targeted Negative Campaigning. *Journal of Artificial Intelligence Research* 86, Article 20 (July 2026), 25 pages. DOI: 10.1613/jair.1.21371

1 Introduction

Recent years have seen negative campaigning becoming ubiquitous in elections (Mattes and Redlawsk 2014). For instance, according to studies by the Wesleyan Media Project (WMP)—which monitors the content and volume of political advertising in the United States (Franklin Fowler and Ridout 2014)—in the U.S. Congress elections of 2010, 2012, and 2014, more than 50% of ads were negative in nature—even when not including contrast ads which compare a favored candidate to his or her opponent. In the 2012 U.S. presidential election—according to an analysis by The Washington Post (Andrews et al. 2012)—one candidate's campaign had spent 91% of its $492 million budget on negative ads, and the other candidate's campaign had spent 85% of its $404 million budget on negative ads.

Negative campaigning is not limited to the two-opponent setting; famously Ralph Nader—during the 2000 U.S. presidential election—referred to Bush and Gore as "Tweedledee and Tweedledum" (Greenhouse 2000). Negative

*Corresponding Author.

Authors' Contact Information: Avishai Zagoury, ORCID: 0000-0002-8476-0351, avishaizag@gmail.com, Bar-Ilan University, Israel; Orgad Keller, ORCID: 0000-0001-8827-429X, orgad@google.com, Google Research, Israel; Avinatan Hassidim, ORCID: 0000-0002-3855-344X, avinatanh@gmail.com, Bar-Ilan University; Google Research, Israel; Noam Hazon, ORCID: 0000-0001-9910-0762, noamh@ariel.ac.il, Ariel University, Israel.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 Copyright held by the owner/author(s).
DOI: 10.1613/jair.1.21371

campaigning is also not a novel approach and its importance traces back to antiquity; in a 64 BC letter by Quintus Tullius Cicero (Cicero 2012), to his brother Marcus, running for the consul of Rome, Cicero recommends to smear his opponents.

Haselmayer (2019) argues about some of the advantages of negativity in campaigns: first, it may convince voters to refrain from voting for an opponent even if it will not make them support the candidate favored by the campaign manager; second, it relies on the concept of "negativity bias" from cognitive psychology: that people tend to give more weight to negative information; and third, the perceived "newsworthiness" of negative facts or stories among journalists tends to be higher, and thus attracts more attention from media outlets. Haselmayer also shows that research about the potential effect of negative campaigning has become widespread as well. Starting as an occasional topic in the 1990s, research on negative campaigning by political scientists has increased substantially in recent years.

In recent years, targeted advertising has emerged (Johnson 2013): many platforms allow the advertisers to deliver a user-specific content, based on user-specific traits, interests or preferences. It has been shown that targeted advertising is an efficient and effective manner of communication, in which the advertiser benefits from a more efficient campaign and a better use of its advertising budget (Iyer et al. 2005). Combining targeted advertising with negative campaigning can thus be a very useful approach (Wong 2020). Even though the effectiveness of targeted negative campaigning was demonstrated in practice, to the best of our knowledge, it has not been studied from a computational perspective.

In this work, we study targeted negative campaigning from a computational perspective by modeling it as a unique variant of Bribery. In our variant, a campaign manager can direct funds for targeting specific demographics (or—in Bribery jargon—pay voters) in order to demote any opponent. Our model—termed TNC (for *targeted negative campaign*) is constructed in such a way to ensure consistency with the properties and effectiveness of targeted negative campaigning in practice: roughly, it makes demotions cheap while effectively making promotions expensive. We contrast this with Shift Bribery (Elkind, Faliszewski, and Slinko 2009) where only promotions of the preferred candidate are allowed. We also prove that our model cannot be framed within the Swap Bribery framework (of the same paper). While we consider both constructive and destructive settings, our model should not be confused with the destructive variants of existing models. A full discussion is provided later.

1.1 Our Contributions

We first prove that for $t \geq 3$, t-approval-TNC is NP-hard. We also show that the problem can be approximated within a factor of $\min(t, m-t)$ in polynomial time, even when each voter has a different price. We obtain this by defining a reduction to min-cost flow which carefully utilizes some of the problem properties. The same algorithm can compute the exact solution for the Plurality and Veto scoring rules.

We then show that Borda-TNC is NP-hard as well, and provide a 3-multiplicative approximation to the problem. We also show that if we introduce prices that are a function of both the voter and candidate, then Borda-TNC cannot be approximated within a factor of $(1-\epsilon)\ln(m/2-1)$ unless P = NP, for any $\epsilon > 0$, where m is the number of candidates.

We continue to discuss the destructive variant. We provide exact solutions for scoring rules if at least one of the following conditions hold: (a) the scoring rule's score values are integers polynomially-bounded in the input size or (b) voter prices are integers polynomially-bounded in the input size. The latter case trivially implies that if all voters have the same price, then the problem has an exact solution. In addition, if neither of these conditions hold, then we show how to turn this algorithm into an FPTAS.

This paper extends a recent conference paper by the authors (Zagoury et al. 2021). We explicitly mention the following major differences between the two papers. For t-approval-TNC approximation, the conference version

constrained t to be fixed, and under this assumption showed a t-factor approximation. That algorithm—while also relying on min-cost flow—involved the enumeration of all the size-t candidate subsets, which was the reason for constraining t to be fixed; in this work, the algorithm is redesigned in order to cleverly avoid this enumeration by carefully utilizing other properties of the problem, thus simultaneously extending (non-fixed-t) and improving (better approximation factor) the result. For the destructive variants, in the original work we provided only an exact solution only for case (a) above; as mentioned this is substantially extended in the current work.

1.2 Related Work

The Bribery problem was originally formulated by Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra (2009, 2006), and has been studied extensively in recent years, usually in the context of hardness (Faliszewski and Rothe 2016) but also in the context of approximation (Faliszewski 2008; Keller et al. 2019). Several types of bribery problems were studied; in Swap Bribery (Elkind and Faliszewski 2010; Elkind, Faliszewski, and Slinko 2009), we pay for swapping two adjacently ranked candidates within a single vote (where price is a function of the bribed voter and the candidates swapped). Shift Bribery is the Swap Bribery variant where only changes promoting a preferred candidate p are allowed; it has received significant attention (Bredereck, J. Chen, et al. 2016; Bredereck, Faliszewski, Niedermeier, et al. 2021, 2016; Faliszewski, Manurangsi, et al. 2021; Maushagen, Neveling, Rothe, and A. Selker 2018; Schlotter et al. 2017). In addition, many voting models were considered, e.g., truncated ballots and partial information (Baumeister et al. 2012; Briskorn et al. 2016), random sample voting (Tao, L. Chen, L. Xu, Shi, et al. 2025), soft constraints (Pini et al. 2013), CP-nets (Dorn and Krüger 2016; Mattei et al. 2012), stochastic influence (Tao, L. Chen, L. Xu, S. Xu, et al. 2023), safe bribery (Karia et al. 2023), iterative voting systems (Maushagen, Neveling, Rothe, and A.-K. Selker 2022), sequential elections (E. Hemaspaandra et al. 2022), and multi-winner elections (Barr et al. 2025; Bredereck, Faliszewski, Furdyna, et al. 2025; Kusek et al. 2023).

Our work is mostly related to Shift Bribery, as Shift Bribery is concerned with promoting the preferred candidate, where we may demote *any* candidate. Shift Bribery was shown to be in P for t-approval and NP-hard for Borda (Elkind, Faliszewski, and Slinko 2009), yet approximable within a $(1+\epsilon)$-factor for any scoring rule (Faliszewski, Manurangsi, et al. 2021). For Copeland, a hardness of approximation result was shown in the same work. In contrast, Swap Bribery was shown to be NP-hard to approximate up to an arbitrary factor for a large list of voting rules, including t-approval for $t \geq 2$, Borda, Copeland, and Maximin; (Elkind and Faliszewski 2010).

In most of the aforementioned works the goal is to make a specific candidate win the election. In contrast, destructive Bribery variants, where the goal is to prevent a candidate from winning, were studied by Faliszewski, E. Hemaspaandra, L. A. Hemaspaandra, and Rothe (2009), and under the name Margin Of Victory (Cary 2011; Dey and Narahari 2015; Magrino et al. 2011; Xia 2012). Other destructive variants of note are Destructive Swap Bribery (Shiryaev et al. 2013) and Destructive Shift Bribery (Kaczmarczyk and Faliszewski 2019). A detailed comparison between our work and existing models is provided later.

We also note that some computational aspects of negativity in elections were previously studied in the context of social networks (Castiglioni et al. 2020; Mehrizi et al. 2019).

2 Preliminaries

An *election* is a pair $E = (C, V)$ such that $C = \{c_1, c_2, \ldots, c_m\}$ is a set of *candidates*, and $V = (v_1, v_2, \ldots, v_n)$ is the *preference profile*, that is, a list of *preference orders* for a set $N = \{1, \ldots, n\}$ of voters, where a preference order v_ℓ is a linear order of the candidates according to ℓ's preferences. We sometimes refer to a preference order v_ℓ as a function such that $v_\ell(c)$ is the rank of candidate c in v_ℓ and $v_\ell^{-1}(j)$ is the candidate ranked j-th in v_ℓ. A rank of 1 means that the candidate is preferred to any other candidate by ℓ. We also use $c \succ_\ell c'$ to indicate that c is preferred to c' by voter ℓ.

A *scoring rule* for m candidates is described by a vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ of non-negative numbers such that $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_m \geq 0$. This rule is applied as follows: each voter ℓ awards every candidate c_i a score according to its rank $v_\ell(c_i)$ i.e., $\alpha_{v_\ell(c_i)}$. A candidate's final score is the sum of the points awarded to him. The *winner set* is the set of all the candidates with the highest final score; we use the *co-winner assumption* where a candidate is considered a winner if he is included in the winner set, and a loser otherwise. Prominent examples of scoring rules are Borda, for which $\boldsymbol{\alpha} = (m-1, m-2, \ldots, 1, 0)$ and t-approval for which $\boldsymbol{\alpha} = (\mathbf{1}^t; \mathbf{0}^{m-t})$ where $t \in \{1, \ldots, m-1\}$ and for $b \in \{0, 1\}$, $\mathbf{b}^k$ is b concatenated k times. Plurality is the specific case of 1-approval, and Veto is the case in which $\boldsymbol{\alpha} = (\mathbf{1}^{m-1}; 0)$. In this paper we only consider $t \in \{1, 2, \ldots, m-1\}$ as the cases of $t \in \{0, m\}$ are trivial. We also mention the non-preferential *range voting* (RV) rule, where voters award a score between 0 and $m-1$ to each candidate, and scores may be repeated.

2.1 Notation

We denote the initial score—before any demotion operations were performed—of a candidate c as $\sigma(c)$. The margin between two candidates c and c' is denoted as $\text{diff}(c, c') = \sigma(c) - \sigma(c')$ (and can be negative). We sometimes use $s(c)$ to denote a candidate's *final* score.

For a candidate set C, we let $\overrightarrow{C}$ denote the sequence containing the elements of C in some arbitrary yet predetermined order. We use $\overleftarrow{C}$ to denote the reverse of $\overrightarrow{C}$. For any subset $C' \subseteq C$, we let $\overrightarrow{C'}$ (resp. $\overleftarrow{C'}$) denote the sub-sequence of $\overrightarrow{C}$ (resp. $\overleftarrow{C}$) containing only the elements of C'. Given a candidate set C', we use $\binom{C'}{t}$ to denote the collection of all size-t subsets of C'. We let $\mathbb{1}[\cdot]$ denote the indicator function where $\mathbb{1}_{C'}(c)$ is a shorthand for $\mathbb{1}[c \in C']$. We also let $[t]$ denote the set $\{1, \ldots, t\}$.

2.2 Problem Definitions

We define the following problems:

2.2.1 Targeted Negative Campaigning. Targeted negative campaign (TNC) is defined as follows. Given an election $E = (C, V)$, and a preferred candidate $p \in C$, the goal is to make p win by finding a minimum-cost sequence Q of demotion operations, where each such operation is a tuple (ℓ, c_i, δ) with the meaning that c_i is demoted δ positions in (the current) v_ℓ. The operations in Q are performed sequentially, by the order they appear in Q. In the unpriced model, the cost of Q—denoted $\pi(Q)$—is the number of operations in Q. We also mention priced variants where the price of each operation (ℓ, c_i, δ) is a function of either ℓ, or both ℓ and c_i. With a slight abuse of notation, we use π to also denote this price function and thus in the former case $\pi(Q) = \sum_{(\ell, c_i, \delta) \in Q} \pi(\ell)$ and in the latter $\pi(Q) = \sum_{(\ell, c_i, \delta) \in Q} \pi(\ell, c_i)$.

Destructive Targeted Negative Campaigning. We also discuss a destructive version of TNC named *destructive targeted negative campaigning* (DTNC), in which our goal is to find a minimum-cost sequence Q of demotion operations that will prevent the currently leading candidate $d \in C$ from winning.

We also provide formal definitions of the following problems, which will be used by our algorithms and proofs.

2.2.2 Set Cover. Given an instance $I = (U, \mathcal{S})$ of Set Cover, where $U = \{u_1, \ldots, u_{\bar{n}}\}$ is the ground-set and $\mathcal{S} = \{S_1, \ldots, S_{\bar{m}}\}$ is a collection of subsets of U, the goal is to find a minimal collection $\mathcal{S}' \subseteq \mathcal{S}$ such that $\bigcup_{S \in \mathcal{S}'} = U$.

2.2.3 Min Cost Flow. Given a directed graph $G = (V, E)$ where each edge $e \in E$ has a capacity $\gamma(e)$ and a cost $a(e)$ (both are non-negative), two specified vertices s and d, and a value T, the goal is to find a flow function $f : E \to \mathbb{R}_0^+$ representing the flow from s to d, subject to (Edmonds and Karp 1972):

- the capacity constraint that $f(e) \leq \gamma(e)$ for each $e \in E$;

- the flow conservation constraint that $\sum_{u \in N^{\text{in}}(v)} f(u,v) = \sum_{u \in N^{\text{out}}(v)} f(v,u)$ for each $v \in V \setminus \{s,d\}$, where $N^{\text{in}}(v)$ and $N^{\text{out}}(v)$ are v's in- and out-neighborhood, respectively;
- the flow value is $|f| = \sum_{u \in N^{\text{out}}(s)} f(s,u) = T$;
- its total cost $\text{TC}(f) = \sum_{e \in E} a(e) f(e)$ is minimal.

The minimum cost flow problem with integer capacities and costs can be solved in polynomial time using algorithms based on augmenting paths, such as those presented by (Edmonds and Karp 1972). These algorithms guarantee an integral flow solution due to the integer-valued augmentation process.

3 Our Model: Discussion and Comparisons

In this section we discuss our model in light of the characteristics of modern-day negative campaigning, and contrast it with existing models.

Our model is constructed in such a way to ensure consistency with the effectiveness of targeted negative campaigning in practice. Specifically, recent practical trends in targeted negative campaigning allow large-scale fine-tuning of ads according to the views of a targeted voter—in one case even supporting 218,000 ad variants; (Wong 2020). Therefore, if a voter has several topics she might care about in the context of a political candidate—possibly with different levels of importance—the choice of topic for an ad provides a way not only to affect the sentiment towards a candidate, but also to control its intensity, or *level*.

We identify this control over the level of negativity with allowing the campaign manager to control the number of positions a candidate is demoted by (hereafter, the *demotion level*) when affecting a voter (as opposed to, for example, always demoting a candidate to become last). Moreover, we allow *any* demotion level.

This fine-grained demotion level is also similar to the inherent nuances in other models of manipulation and bribery under scoring rules: demoting a candidate promotes some candidates below him; as such, the campaign manager has to apply discretion when choosing the demotion level, in order to limit this effect.

We note that in some settings, a finer-grained control over the demotion level is not even required, e.g., for scoring rules that have blocks of same-score positions—like approval-based rules—where we only care about a demotion that will result in a strictly lower score for the candidate.

While we sometimes allow voters to have varying prices for demotions—possibly on a per-candidate basis—in our model the demotion level does not affect the price. This is motivated by arguing that the price is paid for a single ad exposure (or click through), and it is the content of the ad (as discussed above)—and not the number of exposures—that changes the voter's mind. It thus leads to the following observation: by having e.g., a unit price for any demotion level, *demoting* a candidate by δ positions will cost a unit, but *promoting* him by δ positions will cost δ (as it translates to δ demotions), and this is consistent with the effectiveness of negativity argued by political science researchers (Haselmayer 2019). We contrast this with Shift Bribery where only promotions of the preferred candidate are allowed.

Interestingly enough, while the Swap Bribery model is a very general model of campaign management, our model cannot be framed within its framework. We discuss this in the next subsection.

3.1 An Axiomatic Approach

In this section we formally detail the general properties and characteristics that we wanted our model to support in order to capture the essence of negative campaigning. We later contrast them with those of Swap Bribery; most importantly, we show that even a careful choice of a price function in the Swap Bribery setting cannot capture two such properties simultaneously. We do not even require the simplifying assumption—as in the rest of the paper—that demotion of a candidate c by a voter ℓ does not depend on its level δ to show this.

Let $v_\ell^{c\uparrow\delta}$ be the ballot accepted by promoting c by δ positions in v_ℓ. Likewise, let $v_\ell^{c\downarrow\delta}$ be the ballot accepted by demoting c by δ positions in v_ℓ. Notice both are well-defined only if δ is *valid*, i.e, the new rank of c, $v_\ell(c) + \delta$

(resp. $v_\ell(c) - \delta$) remains in the range $[1, m]$. Let $\text{cost}_\ell(v_\ell^{c\uparrow\delta})$ the cost of transforming ballot v_ℓ to $v_\ell^{c\uparrow\delta}$ when bribing ℓ. This is, both in our model and in Swap Bribery respectively, the minimum cost of a valid sequence of allowed operations transforming v_ℓ to $v_\ell^{c\uparrow\delta}$.

Now consider the following two properties:

(1) Demotion is strictly cheaper than promotion: for any ℓ, c, it holds that $\text{cost}_\ell(v_\ell^{c\uparrow\delta}) > \text{cost}_\ell(v_\ell^{c\downarrow\delta})$ for any *valid* $\delta \geq 2$.
(2) Demotion price is candidate-agnostic: for any ℓ and two candidates c, c', $\text{cost}_\ell(v_\ell^{c\downarrow\delta}) = \text{cost}_\ell(v_\ell^{c'\downarrow\delta})$ for any δ which is valid for both c and c'.

Property 1 tries to capture the effectiveness of TNC discussed above. Property 2 is a reasonable simplifying assumption which in many cases—though not all—we would want to have. We show that TNC can satisfy these two conditions simultaneously, for example, in our simple variant having unit-price demotion for any candidate. However, any model satisfying these two conditions cannot be "embedded" into a Swap Bribery scheme, in other words, given a general election $E = (C, V)$ in TNC, we cannot always define a Swap Bribery price function $\pi_\ell^{\text{SB}}(\cdot,\cdot)$ such that $\text{cost}_\ell(v'_\ell)$ in TNC, for any new ballot v'_ℓ, is maintained when switching to Swap Bribery with π_ℓ^{SB} as the price function. We show this by demonstrating that Swap Bribery cannot generally adhere to these two conditions simultaneously, as shown by the following lemma:

Proposition 1. *For a general election $E = (C, V)$, Swap Bribery cannot satisfy these two conditions simultaneously.*

Proof. Let $\pi_\ell^{\text{SB}}(\cdot,\cdot)$ be ℓ's Swap Bribery price function. We first show that Item 2 entails that for any two candidates c and c', it holds that $\pi_\ell^{\text{SB}}(c, c') = f(|v_\ell(c) - v_\ell(c')|)$ for some function f, i.e., the swap price has to be a function of the difference between the two candidates' positions in the original ranking. This can be shown by an induction on $\delta = |v_\ell(c) - v_\ell(c')|$; for $\delta = 1$, it follows directly from Item 2. For $\delta = 2, 3, \ldots$, assuming w.l.o.g. that c is the candidate ranked closer to the top compared to c', then $\text{cost}_\ell(v_\ell^{c\downarrow\delta}) = \text{cost}_\ell(v_\ell^{c\downarrow\delta-1}) + \pi_\ell^{\text{SB}}(c, c')$ which by Item 2, forces $\pi_\ell^{\text{SB}}(c, c')$ to have the same value for any two candidates c, c' sharing the same $v_\ell(c) - v_\ell(c') = \delta$ value.

Now, Fix a voter ℓ and notice that demoting the first—and possibly only—candidate being demoted by ℓ by δ positions costs $\text{cost}_\ell(v_\ell^{c\downarrow\delta}) = \sum_{\delta'=1}^{\delta} f(\delta')$, as we have to swap the candidate by the candidates below her in an increasing order of distance. However, the same goes for promotion, i.e., $\text{cost}_\ell(v_\ell^{c\uparrow\delta}) = \sum_{\delta'=1}^{\delta} f(\delta')$ as well. This directly contradicts Item 1. □

As mentioned, we consider both constructive (where the goal is to make a preferred candidate win) and destructive (where the goal is to prevent the leading opponent from winning) settings. Specifically, our constructive variant should not be confused with the destructive variants of existing models: we allow "negative" operations, but our goal is still "positive" (or constructive): making p win.

Our destructive variants have some resemblance to Destructive Shift Bribery of Kaczmarczyk and Faliszewski (2019). In Destructive Shift Bribery, only operations directly demoting d are allowed. In contrast, our model allows the demotion of any candidate c, for example, in order to strengthen another candidate c' with the goal of making the currently leading candidate d lose. As a result, the cost of the strategy might be radically different. Consider for example the "all-or-nothing" pricing model discussed by Kaczmarczyk and Faliszewski (2019) which is similar to our assumption that a demotion has the same price regardless of its level. Then the fact that we can demote any candidate (and not just d) can change the overall cost from $\Omega(n)$ to 1, as shown by the following theorem.

Proposition 2. *There is an infinite family of election instances under "all-or-nothing" pricing such that DTNC costs* 1*, but Destructive Shift Bribery costs* $\Omega(n) = \Omega(m)$.

Proof. Let $N = [n]$, $C = \{c_1, c_2, \ldots, c_{n-2}, a, b, d\}$ (so that $|C| = |N| + 1$), and fix the scoring rule $\boldsymbol{\alpha} = (2n^2, 3n, 0, \ldots, 0)$. We define the following ballots:

$$\begin{aligned} v_i &= c_i \succ d \succ \overrightarrow{C \setminus \{c_i, d\}} && \forall i \in [n-2]\,; \\ v_{n-1} &= a \succ d \succ \overrightarrow{C \setminus \{a, d\}}\,; \\ v_n &= b \succ a \succ \overrightarrow{C \setminus \{b, a\}}\,. \end{aligned}$$

The scores are as follows: $\sigma(d) = 3n^2 - 3n$, $\sigma(a) = 2n^2 + 3n$, $\sigma(b) = 2n^2$, and for every $i \in [n-2]$, $\sigma(c_i) = 2n^2$. When $n > 6$, d wins. While in Destructive Shift Bribery we will need $\Omega(n)$ operations to make d lose (as each operation can make him lose only $O(n)$ points), in our model a single operation—that is, $(n, b, 1)$—is enough to effectively promote a, award him a final score of $4n^2$, and make him win. □

3.2 Ethical Considerations

In our work, we provide polynomial time algorithms for optimal targeted negative campaigning, and it is worth clarifying the ethical implications of our work. We are, of course, not advocating negative campaigning, and we believe that emphasizing one's strengths is morally preferable to focusing on an opponent's weaknesses. However, negative campaigning is a real and impactful phenomenon, which warrants study from a computational perspective. As such, we are interested in analyzing the vulnerability of various voting rules to manipulation. In addition, the model of targeted negative campaigning has various socially beneficial interpretations, similar to the positive interpretations of Bribery, as discussed by Faliszewski and Rothe (2016) and Faliszewski, Manurangsi, et al. (2021)).

In the margin-of-victory interpretation, we consider destructive negative campaigning and view the minimum cost that will prevent the currently leading candidate from winning as an indication of how decisive the election result was. If this value is high, then it is unlikely that the election was adversely influenced. However, if this number is low, it indicates that the result could have been easily manipulated, warranting closer scrutiny of the election process. Since negative campaigning is (arguably) more tempting than traditional campaigning, measuring the margin of victory using the negative campaigning model may provide a more accurate assessment of an election's vulnerability.

Destructive negative campaigning can also serve as a tool to evaluate the robustness of an election. In this context, we are not trying to defend an election from a manipulating agent who might have affected some voters. Instead, we are assuming that the voters might have made some mistakes on their own. In addition, voters might prioritize their most preferred candidates, ranking them accurately at the top, while ranking other candidates arbitrarily or carelessly. Understanding whether such voter errors or inattention could have influenced the election results provides valuable insights on the robustness of the election. Note that a voter's error or careless ranking of candidates is better modeled as negative campaigning, as voters are more likely to notice if they have mistakenly promoted a candidate they do not favor.

Finally, the minimum cost that makes a candidate the winner can be interpreted as a measure of the candidate's success. That is, even if a candidate does not win, it is still helpful to measure how close the candidate was to victory, and one possible such measure is to use the cost of the negative campaigning.

4 t-Approval

In this section, we will prove the NP-hardness of t-approval-TNC for any fixed $t \geq 3$. We then introduce a $\min(t, m-t)$-factor approximation for the problem for every t, which also supports the case where an operation price is a function of the voter.

4.1 NP-Hardness

We claim the following:

Theorem 1. *For any fixed $t \geq 3$, t-approval-TNC is* NP*-hard.*

To show this, we will focus on the following variant of the *exact cover by 3-sets* (X3C) problem; for some positive integer q, its input is a set $U = \{u_1, \ldots, u_{3q}\}$ of items, and a collection $\mathcal{S} = \{S_1, \ldots, S_{3q}\}$ of subsets of U, where each subset has size 3 and where *each item appears in exactly* 3 *subsets in* $\mathcal{S}$ (this explains why the size of $\mathcal{S}$ is $3q$ as well). The goal is to find an exact cover $\mathcal{S}' \subseteq \mathcal{S}$ of U, i.e., a subset $\mathcal{S}'$ such that each item is covered exactly once. Notice that it necessarily holds that $|\mathcal{S}'| = q$. This specific variant was analyzed by Gonzalez (1985).

It is sufficient to prove the following, focusing on the decision version of the problem:

Lemma 1. *For any fixed $t \geq 3$, given an input consisting of a t-approval-TNC instance and a value q, determining whether there exists a solution with at most tq demotion operations is* NP*-hard.*

We will prove this for $t = 3$; the extension for any fixed $t \geq 4$ is easy with simple adjustments.

Proof. It is easy to see that the problem is in NP. We will show that it is NP-hard by a reduction from X3C with the same parameter q. Given an instance $I = (U, \mathcal{S})$ of X3C, we build the reduction as follows.

We define the candidate set $C = \{p\} \cup U \cup D$, where D is a set of "filler" candidates to be defined immediately; its sole purpose is to guarantee that each of the members of D appears as one of the first three candidates in a vote *at most once*. D is defined as follows: $D = D^S \cup D^u$ where

$$D^S = \left\{ d^S_{i,\beta} \;\middle|\; (i,\beta) \in [3q] \times \{1,2\} \right\};$$
$$D^u = \left\{ d^u_{j,z,\beta} \;\middle|\; (j,z,\beta) \in [3q] \times [q-2] \times \{1,2\} \right\}.$$

We now move to define the votes. For each $S_i \in \mathcal{S}$ we create a vote v_i with the following preference order:

$$v_i = \overrightarrow{S_i} \succ \overrightarrow{\{d^S_{i,1}, d^S_{i,2}\}} \succ p \succ \overrightarrow{C \setminus (\{d^S_{i,1}, d^S_{i,2}, p\} \cup S_i)}.$$

We will refer to the votes in $V_1 = \{v_i\}_{i=1}^{3q}$ as the *primary votes*; these are the votes that the algorithm will have an incentive to bribe: we will later show that they are the only votes with a potential to make p win by bribing them. Notice that such a vote v_i awards a point to each candidate in S_i, and has p ranked in the sixth place (meaning that we will need 3 bribery operations in order to make p gain a point and to make all the items in S_i lose a point simultaneously).

Before adding more voters, notice that at this point we have exactly $|\mathcal{S}| = 3q$ voters, and as such each candidate in U has score exactly 3; each candidate in D^S has score exactly 1. p has score 0.

We will now add more voters with the goal that for every $u_j \in U$, its score will be exactly $\sigma(u_j) = q + 1$. This is achievable by defining a generic "filler" vote $\hat{v}^j_i$ such that:

$$\hat{v}^j_z = \overrightarrow{\{u_j, d^u_{j,z,1}, d^u_{j,z,2}\}} \succ \overrightarrow{C \setminus \{u_j, d^u_{j,z,1}, d^u_{j,z,2}, p\}} \succ p.$$

Then, for each u_j, we add $q - 2$ "copies" of $\hat{v}^j_z$ iterating on z, i.e., we add the votes $\hat{v}^j_1, \hat{v}^j_2, \ldots, \hat{v}^j_{q-2}$. Notice that in the filler votes p is ranked last so that the "filler" voters will not be considered for bribing, as it would have a small impact.

To sum up, the current scores are as follows: each $u_j \in U$ now has a score of exactly $q + 1$; each candidate in $D^S \cup D^u$ has score exactly 1; p has score 0. Note that $|C| = O(q^2)$ and $|V| = O(q^2)$ where the size of the input of X3C is $\Omega(q)$, i.e., the construction is polynomial in the original input size.

Claim 1. *There exists an exact cover to the given X3C instance if and only if there exists a solution to the above TNC with* $3q$ *demotion operations.*

Proof. As follows.

If. Assume that there is a solution with $3q$ demotion operations to our instance of TNC. The only option for increasing the score of p by a point using at most 3 operations is by demoting the first three candidates within a single vote in V_1 (p is never in the fourth nor the fifth place and is in the sixth place only for the *primary voters*). Therefore, as a results of $3q$ demotions, p's score can be at most q. Now, in order for p to win, each u_j had to lose at least one point (because every one of them was initially beating p by $q+1$ points). Given that we have bribed q voters from V_1, and that this bribery led each candidate in U to lose at least a single point, we arrive at the conclusion that $\mathcal{S}' = \{S_i \mid v_i \text{ was bribed}\}$ is a cover of size q. However, as the number of demotions, i.e., $3q$, equals $|U|$, each candidate had to be demoted exactly once, making $\mathcal{S}'$ an *exact* cover.

Only if. Suppose that there is an exact size-q cover $\mathcal{S}'$ to the X3C instance and that we would want to find a solution to our constructed instance of TNC. For every primary vote v_i such that $S_i \in \mathcal{S}'$ we will demote the highest 3 candidates in v_i, which are exactly the candidates of S_i, to be consecutively last (the final order among them is not important). In each such bribed vote v_i, p will go up 3 places and therefore will be awarded a point.

As we have bribed exactly q voters, p's score is now q. Because we have done 3 demotion operations for each one of these q bribed voters, we have made in total $3q$ operations. Now, because we know that $\bigcup_{S\in\mathcal{S}'} S = U$ and $\sum_{S\in\mathcal{S}'}|S| = |U| = 3q$ (as $\mathcal{S}'$ is an exact cover), this means that each candidate $u_j \in U$ had lost exactly 1 point and therefore u_j's final score is exactly q. We arrive at the conclusion that for all $c \in C$, it now holds that $\mathrm{diff}(p,c) \geq 0$ and p wins. □

We have found that X3C can be reduced in a polynomial time to TNC. □

4.2 Veto

Moving away from the case of a fixed t, and focusing on the *unpriced* variant, we note that *Veto* can be solved in polynomial time by iterating over every voter ℓ for which $v_\ell(p) = m$, and greedily demoting the candidate having the highest overall score to the bottom of the list. This is repeated until p wins. This strategy is always the most efficient: it increases $\mathrm{diff}(p,c)$ by 1 for every non-preferred candidate c, besides the leading one c', for which $\mathrm{diff}(p,c')$ is increased by 2. Interestingly, this method relies on the co-winner assumption: if we get to the point where no voter ℓ for which $v_\ell(p) = m$ exists, p must already be winning. In the unique-winner assumption this is clearly not the case and the solution is more involved. The approximation algorithm of the next section will also provide an exact solution for Veto, which holds also when there is also a price which is a function of the voter. It can also be easily modified to hold in the unique-winner assumption.

4.3 Approximation

In this section we provide a τ-factor approximation for t-approval for every t, where $\tau = \min(t, m-t)$, which also supports the case where an operation price is a function of the voter. The algorithm is outlined as follows. Let $b^\star$ denote the final score of p under the optimal demotion strategy making him win. (Notice that a strategy where p wins has to exist under the co-winner assumption: trivially, we can always bribe all voters not approving p, which will make all voters approve of p, granting her the maximum possible number of points. This makes $b^\star$ well-defined) In the following, since we do not know what $b^\star$ is, we iteratively try all possible guesses $b \in \{\sigma(p), \ldots, n\}$ for $b^\star$, and choose the one resulting in the cheapest final solution. (Notice that the number of iterations of this loop is bounded by n, as the maximum score of a candidate in t-approval is n as well. It is thus polynomial in the input size) In each such iteration, we apply a procedure based on constructing a flow

network $G(b) = G(b; C, V, p, \pi)$ which reduces our t-approval-TNC instance to an instance of the *minimal cost flow* problem,[1] and run the *minimal cost flow* algorithm on it with a desired flow value tn (Edmonds and Karp 1972). If it failed to find such a flow we finish the current iteration and try a next value of b; otherwise we can extract a demotion strategy from it. The exact procedure is detailed in Section 4.3.1. We note that while extracting a strategy is fairly easy, analyzing the approximation factor will be notably more involved; we will devote Section 4.3.2 for the analysis.

The reduction will utilize an observation based on the following lemma.

Lemma 2. *For a size-t candidate subset $S \subseteq C$, let $c_{\ell,S} = \arg\max_{c \in S} v_\ell(c)$ be the least preferred candidate from S w.r.t. v_ℓ. The minimal price for making the set of candidates in S become the top t candidates in v_ℓ is $h_\ell(c_{\ell,S})$ where $h_\ell(c) = \pi(\ell) \cdot (v_\ell(c) - t)$.*

Proof. $c_{\ell,S}$ is the least-preferred candidate in S w.r.t. v_ℓ. The number of candidates not in S who are preferred to him in v_ℓ is exactly $v_\ell(c_{\ell,S}) - t$. The cost of demoting each one of these candidates is $\pi(\ell)$, therefore, the minimal price is $\pi(\ell) \cdot (v_\ell(c_{\ell,S}) - t)$. □

Corollary 1. *Given a voter ℓ, two size-t candidate subsets S and S' have the same cost for making their (respective) candidates ranked as the top t candidates by ℓ, if and only if they share their least preferred candidate w.r.t. v_ℓ.*

The above corollary effectively defines an equivalence relation $\sim_\ell$ on the collection $\binom{C}{t}$ of all size-t candidate subsets with the quotient set (i.e., the collection of all equivalence classes) $\binom{C}{t}/\sim_\ell = \{\mathcal{Z}_\ell(c) \mid c \in C, v_\ell(c) \geq t\}$, where $\mathcal{Z}_\ell(c) = \{S \in \binom{C}{t} \mid \arg\max_{c' \in S} v_\ell(c') = c\}$ is the collection of all size-t subsets having candidate c as their least preferred candidate w.r.t. v_ℓ.

4.3.1 Main Iteration Logic. Let $A_\ell = \{c \in C \mid v_\ell(c) \leq t\}$ be the subset of candidates who ℓ currently approves of, likewise $D_\ell = C \setminus A_\ell$ is the subset of candidates who ℓ currently disapproves of. Given a guess b, we construct a priced flow network $G(b)$ in which units of flow represent awarded points, as follows. The vertex set $U = \{s, d, r\} \cup N \cup U_1 \cup C$ consists of the following "layers" (visualized in Figure 1).

- N (resp. C) represents the voter set (resp. candidate set); a flow unit passing through a vertex $\ell \in N$ (resp. $c \in C$) represents a point awarded by ℓ (resp. awarded to c).
- $U_1 = \{u_{c,\ell} \mid c \in C, \ell \in N\}$. A flow unit passing through a vertex $u_{c,\ell}$ represents a point awarded by ℓ to the candidate c.
- s and d are the *source* and *destination* vertices. r represents all the candidates other than p.

The edge set $E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5 \cup E_6 \cup E_7$ consists of the following, where $\gamma(u, v)$ is the capacity of an edge (u, v) and $a(u, v)$ is its cost. When discussing the different edge types, we provide some partial intuition as to their role, however, the role will be made clear in Section 4.3.2.

- $E_1 = \{(s, \ell) \mid \ell \in N\}$ with all capacities equal to t, guarantees that each voter has at most t points to distribute.
- $E_2 = \{(\ell, u_{c,\ell}) \mid \ell \in N, c \in A_\ell\}$ with all capacities equal to 1. Recall that the candidates in A_ℓ are the ones currently awarded a point by ℓ.
- $E_3 = \{(\ell, u_{c,\ell}) \mid \ell \in N, c \in D_\ell\}$ with all capacities equal to 1. For each edge $(\ell, u_{c,\ell}) \in E_3$, we let $a(\ell, u_{c,\ell}) = h_\ell(c)$ where $h_\ell(c) = \pi(\ell) \cdot (v_\ell(c) - t)$. These edges thus represent $\mathcal{Z}_\ell(c)$ for $c \in D_\ell$.
- $E_4 = \{(u_{c,\ell}, c) \mid c \in A_\ell, \ell \in N\}$ with all capacities equal to 1, guarantees that each candidate in A_ℓ can receive at most one point from a specific voter.
- $E_5 = \{(u_{c,\ell}, c) \mid c \in D_\ell, \ell \in N\}$ with all capacities equal to 1, guarantees that each candidate in D_ℓ can receive at most one point from a specific voter.

[1] As C, V, p, π are clear from context, we omit them and hereafter use $G(b)$ to denote the network.

Algorithm 1: Approximation for t-approval-TNC

input: an election (C, V), a price function $\pi\colon N \to \mathbb{R}_0^+$, and a candidate p.

for $b \leftarrow \sigma(p)$ **to** n **do**
  Construct $G(b) = G(b; C, V, p, \pi)$. Find its min-cost flow (of value tn) f, if exists.
  if f *not found* **then continue**
  Construct a strategy Q_b based on the following operations:
  foreach $\ell \in N$ **do**
    $A_\ell \leftarrow \{c \in C \mid v_\ell(c) \leq t\}$ `/* candidates ℓ currently approves of */`
    $F_\ell \leftarrow \{c \in C \mid f(u_{c,\ell}, c) = 1\}$ `/* candidates ℓ should approve of */`
    if $F_\ell \neq A_\ell$ **then**
      $c'_\ell \leftarrow \arg\max_{c \in F_\ell} v_\ell(c)$
      Add to Q_b the operations demoting the candidates $\{c \notin F_\ell \mid v_\ell(c) < v_\ell(c'_\ell)\}$ to be ranked consecutively last by ℓ.
return the strategy Q_b, for $b \in \{\sigma(p), \ldots, n\}$, minimizing $\pi(Q_b)$.

- $E_6 = \{(c, r) \mid c \in C \setminus \{p\}\}$, with all capacities equal to b guarantees that each candidate will receive at most b points (where b will be p's final score).
- $E_7 = \{(p, d), (r, d)\}$, where $\gamma(p, d) = b$ and $\gamma(r, d) = tn - b$. By later setting the desired flow value to tn, these edges will be saturated, and p's score will be exactly b.

Excluding edges in E_3, all edge costs are 0. Finally we let $G(b) = (U, E, \gamma, a, s, d)$ denote the instance of the min-cost flow we have constructed. An example of a sub-graph of $G(b)$ is shown in Figure 1.

After constructing $G(b)$, we run the *minimal cost flow* algorithm on it with a desired flow value tn (Edmonds and Karp 1972). If it failed to find such a flow then we continue to the next iteration; otherwise, denote the resulting network flow as f. For each voter ℓ, let $F_\ell = \{c \mid f(u_{c,\ell}, c) = 1\}$ be the set of candidates receiving a unit flow from ℓ, and let $c'_\ell = \arg\max_{c \in F_\ell} v_\ell(c)$. We will later show that $|F_\ell| = t$. Based on f, we compute the following demotion strategy Q: we perform the necessary demotions such that for each ℓ for which $F_\ell \neq A_\ell$ (i.e., the set of candidates receiving a unit flow from ℓ differs from the set of candidates ℓ originally approved of), the candidates in F_ℓ become the top t candidates ranked by ℓ; this is achieved by demoting all candidates $c \notin F_\ell$ for which $v_\ell(c) < v_\ell(c'_\ell)$ to the bottom of v_ℓ.

As mentioned, our approximation algorithm performs the above procedure for each $b \in \{\sigma(p), \ldots, n\}$. As discussed above, each such trial either fails or finds a candidate demotion strategy. Finally, our algorithm returns the strategy Q having the minimum cost out of all strategies computed during the trials for the various b's. Pseudo-code is provided in Algorithm 1.

4.3.2 Analysis. In the following we analyze our algorithm's correctness and approximation factor.

Lemma 3. *In Algorithm 1, if during an iteration b a minimum-cost flow (of value tn) f is found, then the algorithm constructs valid strategy Q_b making p win.*

Proof. We first show that if a minimum-cost flow f was found, then a valid strategy Q_b is constructed. Focus on a single voter ℓ. The incoming flow to the vertex ℓ is capped at t; since $|f| = tn$, indeed the edge (s, ℓ) is saturated and exactly t units of flow pass through ℓ. As these t units of flow have to pass through the vertices $\{u_{c,\ell} \mid c \in C\}$, the incoming flow to each such vertex is capped at 1, and the flow on an edge is integral, there are *exactly* t candidates c such that $f(u_{c,\ell}, c) = 1$, thus $|F_\ell| = t$. In words, for each ℓ, the flow f allows us to determine exactly t candidates that should be ranked first by ℓ.

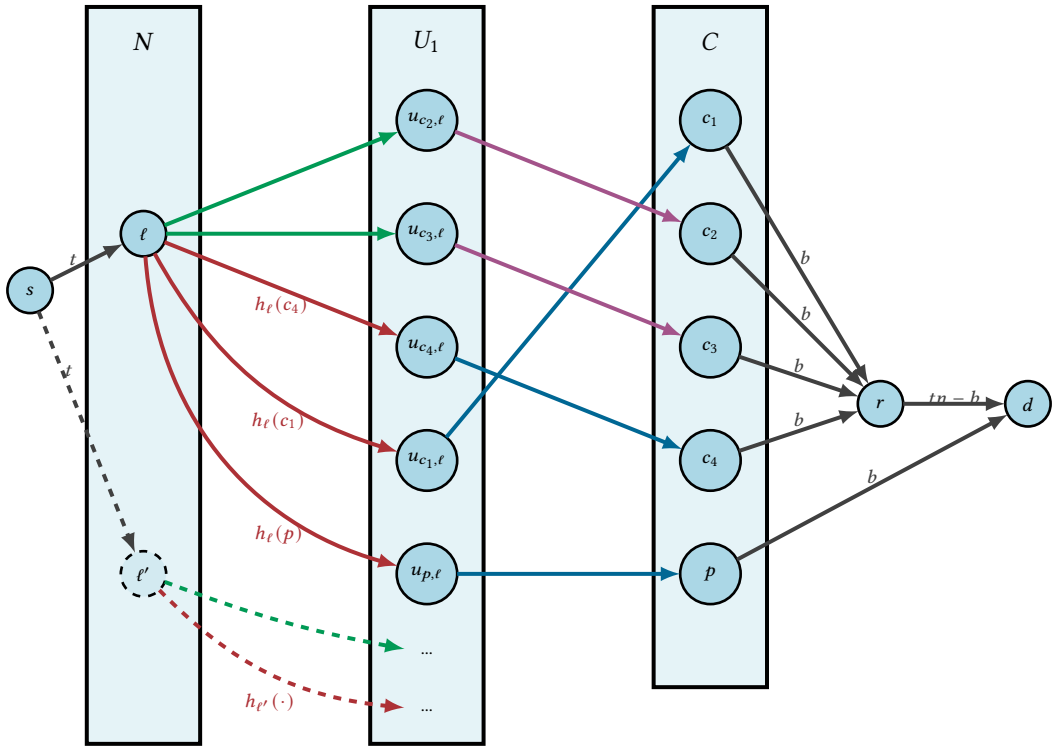

Fig. 1. A sub-graph of $G(b)$ for 2-approval, containing the source node s, a single voter node ℓ, and the nodes reachable from ℓ (but throwing away other voter nodes and some of the nodes reachable from them). Assume that ℓ's preferences are $c_2 \succ c_3 \succ c_4 \succ c_1 \succ p$. When indicated, the edge labels above the edge are their capacity. Where edges have a second label (below the edge), it is their cost. All other edge capacities are 1 and costs are 0. Node layers are indicated by rectangles. Nodes in U_1 are organized top-to-bottom according to v_ℓ, for clarity. Edges from E_2 are indicated in green. Edges from E_3 are indicated in red. Edges from E_4 are indicated in purple. Edges from E_5 are indicated in blue. Best viewed in color.

We now show that this strategy makes p win. Q_b was constructed such that c will be approved by ℓ if $f(u_{c,\ell}, c) = 1$. As $f(c, r) = \sum_{\ell \in N} f(u_{c,\ell}, c)$, the values $f(c, r)$ equal the number of points that candidate $c \in C \setminus \{p\}$ is awarded, and $f(p, d) = \sum_{\ell \in N} f(u_{p,\ell}, p)$ equals the number of points p is awarded in Q_b. Since $|f| = tn$, the edge (p, d) has to be saturated thus $f(p, d) = b$. In addition, since $f(c, r) \leq b$ for all $c \in C \setminus \{p\}$ (by the capacity constraints), p is necessarily winning with a score of b. □

Let $W \subset U \setminus \{s, d\}$ be a vertex subset such that no path exists between two vertices in W. With a slight abuse of notation, for a flow f, let $f(W)$ denote either the *total incoming flow* to the vertices of W or *total outgoing flow* from the vertices of W (as the two values equal, the choice of which is insignificant). Formally, $f(W) = \sum_{u \in U} \sum_{w \in W} f(u, w) = \sum_{u \in U} \sum_{w \in W} f(w, u)$ (for this purpose, the value $f(u, v)$ when $(u, v) \notin E$ is always 0).

Lemma 4. *Let f be a flow in the graph. For every voter ℓ, it holds that $f(\{u_{c,\ell} \mid c \in D_\ell\}) \leq \min(t, m - t) = \tau$.*

Proof. The only vertex transferring flow to $\{u_{c,\ell} \mid c \in D_\ell\}$ is ℓ. Since the incoming capacity of ℓ is at most t, the total incoming flow to the vertices in $\{u_{c,\ell} \mid c \in D_\ell\}$ is at most t as well. Since every vertex in $\{u_{c,\ell} \mid c \in D_\ell\}$

has an outgoing capacity of 1, and $|\{u_{c,\ell} \mid c \in D_\ell\}| = |D_\ell| = m - t$, the outgoing flow from $\{u_{c,\ell} \mid c \in D_\ell\}$ is at most $m - t$. □

Let $Q^\star$ be the optimal demotion strategy (which, as mentioned, is guaranteed to exist). Recall that $b^\star$ is the final score of p under $Q^\star$, and that $\tau = \min(t, m - t)$.

Lemma 5. *There exists a potential flow $f^\star$ in $G(b^\star)$ where $|f^\star| = tn$ and $\mathrm{TC}(f^\star) \leq \tau \cdot \pi(Q^\star)$.*

Proof. We will construct $f^\star$ in $G(b^\star)$ as follows. For each ℓ, let $F_\ell^\star$ be the candidates ℓ approves of following the demotion operations in $Q^\star$. We also let $A_\ell^\star = F_\ell^\star \cap A_\ell$ be the candidates in $F_\ell^\star$ who ℓ originally approved of, and likewise $D_\ell^\star = F_\ell^\star \cap D_\ell$ are the candidates in $F_\ell^\star$ who ℓ originally disapproved of. The main idea is that a unit flow going to a candidate $c \in A_\ell^\star$ can flow directly (and for free) on the path $\ell \to u_{c,\ell} \to c$, but that a unit flow for a candidate $c \in D_\ell^\star$ (if such exists; otherwise, $A_\ell^\star = F_\ell^\star$ and ℓ was not "bribed") will be flowing on the path $\ell \to u_{c,\ell} \to c$ for a price; specifically, some flow would have to pass on the path $\ell \to u_{c_\ell^\star,\ell} \to c_\ell^\star$, , where $c_\ell^\star = \arg\max_{c \in F_\ell^\star} v_\ell(c)$ is the least preferred candidate in $F_\ell^\star$. Specifically, it will flow through the edge $(\ell, u_{c_\ell^\star,\ell})$, incurring a cost of $h_\ell(c_\ell^\star)$ in the process.

Putting this intuition formally, let $s^\star(c)$ be the final score of c in $Q^\star$. The flow is defined, for every ℓ, as follows:

$$
\begin{aligned}
f^\star(s,\ell) &= t & f^\star(\ell, u_{c,\ell}) &= \mathbb{1}_{F_\ell^\star}(c), \quad \forall c \\
f^\star(u_{c,\ell}, c) &= \mathbb{1}_{F_\ell^\star}(c), \quad \forall c & f^\star(c, r) &= s^\star(c), \quad \forall c \\
f^\star(r,d) &= tn - b^\star & f^\star(p,d) &= b^\star \,.
\end{aligned}
$$

The above covers both the cases where ℓ was either bribed or not. All other edges will have flow value 0.

Let $N^\star \subseteq N$ be the set of voters bribed in $Q^\star$. It can be easily verified that all the flow conditions are satisfied, that $|f^\star| = f^\star(r,d) + f^\star(p,d) = tn$, and that

$$
\begin{aligned}
\mathrm{TC}(f^\star) &= \sum_{\ell \in N^\star} \sum_{c \in D_\ell^\star} a(\ell, u_{c,\ell}) f^\star(\ell, u_{c,\ell}) && (1)\\
&\leq \sum_{\ell \in N^\star} a(\ell, u_{c_\ell^\star,\ell}) \sum_{c \in D_\ell^\star} f^\star(\ell, u_{c,\ell}) && (2)\\
&= \sum_{\ell \in N^\star} a(\ell, u_{c_\ell^\star,\ell}) f^\star(\{u_{c,\ell} \mid c \in D_\ell^\star\}) && (3)\\
&\leq \tau \cdot \sum_{\ell \in N^\star} h_\ell(c_\ell^\star) && (4)\\
&= \tau \cdot \pi(Q^\star) \,, && (5)
\end{aligned}
$$

where Inequality (4) follows from Lemma 4 and Inequality (5) follows from Lemma 2. □

Recall that for each voter ℓ, $F_\ell = \{c \mid f(u_{c,\ell}, c) = 1\}$ is the set of candidates receiving a unit flow from ℓ, and that $c'_\ell = \arg\max_{c \in F_\ell} v_\ell(c)$. Following that, we let $\bar{A}_\ell = A_\ell \cap F_\ell$ and $\bar{D}_\ell = D_\ell \cap F_\ell$.

In the following lemmas, we will argue that Algorithm 1 returns a demotion strategy Q for which $\pi(Q) \leq \tau \cdot \pi(Q^\star)$. The following lemmas reason about iterations b for which a flow in $G(b)$ was found at Algorithm 1 of Algorithm 1, and thus $|f| = tn$. Notice that at least one such "successful" $b \in \{\sigma(p), \ldots, n\}$ has to exist, and that this is a direct corollary of Lemma 5.

For a flow f, let $\mathrm{TC}_\ell(f) = \sum_{c \in D_\ell} a(\ell, u_{c,\ell}) f(\ell, u_{c,\ell})$. Notice that $\mathrm{TC}_\ell(f)$ is the part of $\mathrm{TC}(f)$ that is "attributed" to ℓ, i.e., results from flow incoming from vertex ℓ.

Lemma 6. *If $F_\ell = A_\ell$, then $\mathrm{TC}_\ell(f) = 0$. Otherwise, $\mathrm{TC}_\ell(f) \geq h_\ell(c'_\ell)$.*

Proof. If $F_\ell = A_\ell$, the flow outgoing from ℓ does not pass through vertices of $\{u_{c,\ell} \mid c \in D_\ell\}$ and is thus free of cost.

If $F_\ell \neq A_\ell$, then since $F_\ell = \{c \mid f(u_{c,\ell}, c) = 1\}$ and $c'_\ell \in F_\ell$, the incoming flow to $u_{c'_\ell,\ell}$ is also 1 thus $f(\ell, u_{c'_\ell,\ell}) = 1$. Then:

$$\begin{aligned} h_\ell(c'_\ell) &= a(\ell, u_{c'_\ell,\ell}) && \text{(by definition of } a\text{)} \\ &= a(\ell, u_{c'_\ell,\ell}) \cdot f(\ell, u_{c'_\ell,\ell}) && \text{(by } f(\ell, u_{c'_\ell,\ell}) = 1\text{)} \\ &\leq \sum_{c \in D_\ell} a(\ell, u_{c,\ell}) \cdot f(\ell, u_{c,\ell}) \\ &= \mathrm{TC}_\ell(f)\ . \end{aligned}$$

This completes the proof. □

Lemma 7. *Algorithm 1 returns a valid solution Q making p win, and $\pi(Q) \leq \tau \cdot \pi(Q^\star)$.*

Proof. As proven in Lemma 5, when $b = b^\star$ there exists a maximal flow tn in $G(b^\star)$ and therefore our algorithm will find such a flow. By Lemma 3, in this iteration we will find a valid strategy Q_b making p win. It is therefore enough to show that $\pi(Q_b) \leq \tau \cdot \pi(Q^\star)$.

As a result of Lemma 2 the price paid for bribing a voter ℓ is exactly $h_\ell(c'_\ell)$. Recall $f^\star$, the flow induced by the optimal strategy as detailed in Lemma 5. Let $N' \subseteq N$ be the set of voters bribed in Q_b. Since $\pi(Q_b) = \sum_{\ell \in N'} h_\ell(c'_\ell)$ we get that

$$\begin{aligned} \pi(Q_b) &= \sum_{\ell \in N'} h_\ell(c'_\ell) && \text{(by Lemma 2)} \\ &\leq \sum_{\ell \in N'} \mathrm{TC}_\ell(f) && \text{(by Lemma 6)} \\ &= \mathrm{TC}(f) \\ &\leq \mathrm{TC}(f^\star) && (6) \\ &\leq \tau \cdot \pi(Q^\star) && \text{(by Lemma 5)}\ , \end{aligned}$$

where eq. (6) follows from the minimality of f. □

As a direct corollary, we obtain the following theorem:

Theorem 2. *Algorithm 1 is a $\min\{t, m-t\}$-approximation algorithm for t-approval-TNC.*

Corollary 2. *With a price function of type $\pi\colon N \to \mathbb{R}_0^+$, Plurality-TNC and Veto-TNC can be solved exactly using the above algorithm.*

Remark. The above algorithm can easily be modified to support the unique-winner assumption: by defining the capacity of the edge set $E_6 = \{(c, r) \mid c \in C \setminus \{p\}\}$ to be $b - 1$ (instead of the previous b) we *guarantee* that p, who is "forced" by the reduction to be awarded a final score of b, will win over all other candidates. However, notice that under this assumption a winning strategy might not at all exist; in this case no flow will be found in any of the iterations of our algorithm. An easy example for having no winning strategy is the setting of Veto when $n < m - 1$: even if we get to a point where p is receives a point from *each* of the voters, there has to exist another candidate with the same property.

5 Borda

In this section we will show NP-hardness and a 3-multiplicative approximation for Borda.

5.1 NP-Hardness

Theorem 3. *Borda-TNC is* NP*-hard.*

To show this, it is sufficient to prove the following, focusing on the decision version of the problem:

Lemma 8. *Given a value k, determining whether there exists a solution to Borda-TNC with at most k demotions is* NP*-hard.*

Proof. Recall that $\bar{n}$ and $\bar{m}$ are the number of items and the number of subsets in an instance of Set Cover, respectively. Given an instance $I = (U, \mathcal{S})$ of Set Cover and a desired cover size $k \leq \bar{m}$, we construct a reduction as follows.

We define the candidate set $C = U \cup D \cup \{p, a, b\}$, where $D = \{d_0, \ldots, d_{\bar{n}-2}\}$ is a set of $\bar{n}-1$ filler candidates, p is the preferred candidate and a, b are two additional candidates. We let $D_{<q} = \{d_0, \ldots, d_{q-1}\}$ and $D_{\geq q} = \{d_q, \ldots, d_{\bar{n}-2}\}$ for any $q \in \{0, \ldots, \bar{n}-2\}$.

The preference profile is defined as follows. For each set $S_i \in \mathcal{S}$, we define two preference orders v_i^1, v_i^2, such that

$$v_i^1 = \overrightarrow{D_{\geq |S_i|}} \succ \overrightarrow{S_i} \succ a \succ b \succ \overrightarrow{U \setminus S_i} \succ \overrightarrow{D_{<|S_i|}} \succ p \,; \tag{7}$$

$$v_i^2 = p \succ \overleftarrow{D_{<|S_i|}} \succ \overleftarrow{U \setminus S_i} \succ b \succ a \succ \overleftarrow{S_i} \succ \overleftarrow{D_{\geq |S_i|}} \,. \tag{8}$$

In addition, we define the following two preference orders:

$$\bar{v} = \overrightarrow{D} \succ a \succ b \succ \overrightarrow{U} \succ p \,; \tag{9}$$

$$\hat{v} = p \succ \overleftarrow{U} \succ a \succ b \succ \overleftarrow{D} \,. \tag{10}$$

Finally, we define the following two preference orders:

$$\bar{v}' = \overrightarrow{D} \succ a \succ \overrightarrow{U} \succ b \succ p \,; \tag{11}$$

$$\hat{v}' = p \succ \overleftarrow{U} \succ a \succ b \succ \overleftarrow{D} \,. \tag{12}$$

We let

$$V = \left\{v_i^1, v_i^2\right\}_{i=1}^{\bar{m}} \cup \left\{\bar{v}_j, \hat{v}_j\right\}_{j=1}^{k(\bar{n}+3)-1} \cup \{\bar{v}', \hat{v}'\} \,,$$

where each $\bar{v}_j$ (resp. $\hat{v}_j$) is a copy of $\bar{v}$ (resp. $\hat{v}$).[2] Let $E = (C, V)$ be the resulting election.

Lemma 9. *For E, it holds that* $\mathrm{diff}(a, p) = k(\bar{n}+3)$*, that* $\mathrm{diff}(p, b) = k(\bar{n}+3) + \bar{n}$*, and that* $\mathrm{diff}(p, d) = 0$ *for every $d \in D$. In addition,* $\mathrm{diff}(u_1, p) = \cdots = \mathrm{diff}(u_{\bar{n}}, p) = 1$.

Proof. Assume that we start with an empty preference profile, first add the votes $\{v_i^1, v_i^2\}_{i=1}^{\bar{m}}$, then the votes $\{\bar{v}_j, \hat{v}_j\}_{j=1}^{k(\bar{n}+3)-1}$ and finally $\{\bar{v}', \hat{v}'\}$.

When the preference profile is empty, obviously $\mathrm{diff}(c, c') = 0$ for every $c, c' \in C$. Adding $\{v_i^1, v_i^2\}_{i=1}^{\bar{m}}$ preserves these relative scores as v_i^1 and v_i^2 are the same preference order, only that one is reversed w.r.t. the other. Therefore, $\mathrm{diff}(c, c')$ is still 0 for every $c, c' \in C$.

Each pair of votes in $\{\bar{v}_j, \hat{v}_j\}_{j=1}^{k(\bar{n}+3)-1}$ adds an additional point to a on the expense of b—as $a \succ b$ in *both* $\bar{v}$ and $\hat{v}$—while preserving the relative scores of all other candidates. Thus $\mathrm{diff}(c, c') = 0$ for every $c, c' \notin \{a, b\}$, but

[2] While V is a list of preference orders, with a slight abuse of notation, we have defined it using set operations.

$\text{diff}(a,c) = k(\bar{n}+3) - 1$ if $c \neq b$ and $\text{diff}(b,c) = -k(\bar{n}+3) + 1$ if $c \neq a$. Obviously, $\text{diff}(a,b) = 2k(\bar{n}+3) - 2$. Finally, the last pair $\{\bar{v}', \hat{v}'\}$ adds a point to each of $\{a\} \cup U$ on the expense of b. □

Lemma 10. *Let Q be a winning demotion strategy for E with k demotion operations. Then all bribed voters have votes of types v_i^1, $\bar{v}$, or $\bar{v}'$, and in each of them a was demoted by $\bar{n}+2$ positions.*

Proof. Let $s(c)$ be the score of candidate c after all the demotion operations. Since we have k demotion operations, $s(p)$ will be at most $\sigma(p)+k$. To make p indeed win we require that $s(a) \leq s(p)$, thus a should lose *at least* $k(\bar{n}+2)$ points. However, notice that for each voter, a can be pushed down at most $\bar{n}+2$ positions. Indeed, only for votes of types v_i^1, $\bar{v}$, and $\bar{v}'$, a can be demoted by $\bar{n}+2$ positions (for all other voters, a can be demoted by at most $\bar{n}+1$ positions). Thus, to reach the desired decrease in score, only voters of types $v_i^1, \bar{v}, \bar{v}'$ can be bribed, each once, and in each operation a will be demoted exactly $\bar{n}+2$ positions. As a result, $s(a) = s(p) = \sigma(p)+k$. □

We are now ready to complete the proof of Lemma 8 by showing that E has a winning strategy with k operations if and only if U can be covered by k subsets of $\mathcal{S}$.

If. Let $\mathcal{S}'$ be a valid k-cover. For each $S_i \in \mathcal{S}'$, bribe v_i^1 and move a to the last position in v_ℓ. Notice that now $s(p) = s(a) = \sigma(p)+k$. Now focus on the effect of bribing a single voter v_i^1: it does not change the value $\text{diff}(p,u)$ for each $u \in U \setminus S_i$, but increases $\text{diff}(p,u)$ by 1 for each $u \in S_i$. Since $\mathcal{S}'$ is a proper cover, it means that according to our scheme, for each $u \in U$ there exists at least one bribed voter v_i^1 who increases $\text{diff}(p,u)$ by 1 (this is a voter v_i^1 for which $S_i \in \mathcal{S}'$ and $u \in S_i$), thus at the end of the bribery process, $\text{diff}(p,u) \geq 0$. We have just showed that $s(p) \geq s(u)$ for each $u \in U$, and that $s(p) \geq s(a)$. As for the remaining candidates in $D \cup \{b\}$, notice that at first they were ranked equally or less than p, and that each time one of them was promoted, p was promoted as well, and therefore $\text{diff}(p,c)$ does not decrease for each $c \in D \cup \{b\}$. We conclude that as a result of this bribery scheme, p now wins.

Only if. Assume that p can be made to win by bribing k voters, and consider the corresponding demotion strategy. By Lemma 10, in all bribed votes a was pushed down $\bar{n}+2$ positions. Now observe only the bribed voters of type v_i^1. Since for each $u \in U$ it held before that $\text{diff}(p,u) = -1$, but now $\text{diff}(p,u) \geq 0$, we bribed at least one voter v_i^1 such that $u \in S_i$, which allowed $\text{diff}(p,u)$ to increase by a point. Therefore, the collection $\mathcal{S}' = \{S_i \in \mathcal{S} \mid v_i^1 \text{ is bribed}\}$ constitutes a valid cover. Since $|\mathcal{S}'| \leq k$ we are done. □

5.2 Approximation

The main idea behind our approximation is noticing that the hardness of TNC under Borda stems from the fact that when we demote a candidate by δ positions, then δ candidates receive a point. Now assume that we ignore this issue for a moment. This is equivalent to range voting (RV) where each voter awards a score between 0 and $m-1$ to each candidate. Here, demoting a candidate—e.g., by having a voter decrease her awarded score by δ points—does not have any consequence on the score of other candidates.

We start by reducing our instance into a corresponding RV instance $\bar{E}$: we naturally translate a ranking in the j-th place of a candidate by a voter to awarding him a score of $\alpha_j = m - j$ by the voter. In the reduced instance, we look for the smallest k for which we can make sure—using at most k demotion operations—that all candidate scores do not exceed a bound $T = \sigma(p) + k$. We refer to TNC under RV with this goal as *bounded* TNC under RV. We denote the function that—given a RV instance, and the values k and T—either returns a sequence of at most k demotion operations or fails—as $\text{BTNCRV}(\bar{E}, k, T)$.

Computing BTNCRV is easily achievable by a greedy algorithm which iteratively: (a) identifies a *violating candidate* for which the score exceeds T; (b) finds the voter who awards him the maximal number of points; and (c) performs a demotion operation, decreasing the score given by this voter to the candidate to 0. The algorithm for BTNCRV is illustrated as Algorithm 2.

Algorithm 2: BTNCRV$((C, \bar{V}), k, T)$

/* Recall that $\bar{V} = (\bar{v}_\ell)_\ell$ where $\bar{v}_\ell = \{(c, \alpha_{v_\ell(c)}) \mid c \in C \setminus \{p\}\}$. */

$Q \leftarrow \emptyset$
foreach $c \in C$ **do**
  $S_c \leftarrow \{(\ell, \delta) \mid (c, \delta) \in \bar{v}_\ell,\ \ell = 1, \ldots, n\}$
  $s_c \leftarrow \sigma(c)$
for $i \leftarrow 1$ **to** k **do**
  $C' \leftarrow \{c \in C \mid s_c > T\}$
  if $C' = \emptyset$ **then**
    break
  Pick an arbitrary $c \in C'$.
  $(\ell', \delta') \leftarrow \arg\max_{(\ell,\delta) \in S_c} \delta$
  $Q \leftarrow Q \cup \{(\ell', c, \delta')\}$
  $S_c \leftarrow S_c \setminus \{(\ell', \delta')\}$
  $s_c \leftarrow s_c - \delta'$
$C' \leftarrow \{c \in C \mid s_c > T\}$
if $C' = \emptyset$ **then return** Q **else fail**

Algorithm 3: Approximation for Borda-TNC

input: an election (C, V), and a candidate p.
$\bar{V} = (\bar{v}_\ell)_\ell$ where $\bar{v}_\ell = \{(c, \alpha_{v_\ell(c)}) \mid c \in C \setminus \{p\}\}$
Let $\bar{E} = (C \setminus \{p\}, \bar{V})$
Let k' be the minimum k for which BTNCRV$(\bar{E}, k, \sigma(p) + k)$ does not fail
$Q \leftarrow$ BTNCRV$(\bar{E}, k', \sigma(p) + k')$
foreach $(\ell, c, \delta) \in Q$ **do**
  Demote c in v_ℓ by δ positions.
while *p is not winning* **do**
  Let $\ell \in N, c \in C \setminus \{p\}$ such that c is ranked immediately before p in v_ℓ.
  Demote c in v_ℓ by one position.
return the sequence of demotion operations performed.

After finding the smallest k for which BTNCRV$(\bar{E}, k, \sigma(p)+k)$ succeeds, we take the resulting demotion strategy and apply it on the original Borda instance. Unfortunately, these operations now do have their consequences, and candidate scores might increase as a result of the demotion operations. However, we will prove that not by too much. In any case, p might be losing. We address this by repeating the following procedure: we find a voter who currently has p at any position but the top one, and then swap p and the candidate ranked above him by this voter. We shall repeat this step until p is winning. The overall algorithm is described in Algorithm 3. A similar 2-stage logic was used by Elkind, Faliszewski, and Slinko (2009) in their 2-approximation for Shift Bribery: both algorithm first promote p using "coarser" actions, i.e. actions which might affect several other candidates. Then "finer" operations—swaps—are performed in order to promote p.

Lemma 11. *Algorithm 2 finds a solution to bounded TNC under range voting (*BTNCRV*).*

Proof. For RV, a demotion of a candidate by a voter does not have any consequences on other voters and candidates. Therefore, a simple greedy procedure of repeatedly identifying and demoting a violating candidate is sufficient. □

Lemma 12. *Let E be a preferential election under Borda, and let $\bar{E}$ be its corresponding election under RV, as constructed in Algorithm 3 of Algorithm 3. Then:*

- *A sequence of operations Q for range voting can be applied on the Borda instance, only that in each operation $(\ell, c, \delta) \in Q$, δ now pertains to the number of positions c is demoted by in v_ℓ (for RV δ was the decrease in points).*
- *A sequence of operations Q applicable on E can be modified into a sequence of operations $f(Q)$ applicable on $\bar{E}$ such that the final score of each candidate in the RV setting will be at most his final score in the Borda setting.*

Proof. For the first item, assume that we apply each operation in Q sequentially, in parallel on the two elections. In Borda, an operation (ℓ, c, δ) might have side effects on other candidates, however the value $v_\ell(c')$ for any candidate $c' \neq c$ can only decrease. As such, if a later applied operation is of the type (ℓ, c', δ'), this means that the score currently awarded to c' by ℓ in the Borda instance is at least δ', and thus c''s rank is at most $m - \delta'$, meaning that the operation can be safely applied.

For the second item, simply replace each operation $(\ell, c, \delta) \in Q$ with an operation (ℓ, c, δ') where δ' is the score currently awarded to c by ℓ in the RV instance (so that following the operation, the score awarded to c by ℓ is 0). □

Let $k^\star$ be the optimal number of demotion operations required to make p win, and let k' be the value from Algorithm 3 of Algorithm 3.

Lemma 13. $\text{BTNCRV}(\bar{E}, k^\star, \sigma(p) + k^\star)$ *does not fail. In particular, this implies that $k' \leq k^\star$.*

Proof. For TNC under Borda, if p can be made to win by at most $k^\star$ demotion operations, then p's final score $s(p)$ is at most $\sigma(p) + k^\star$. In addition, following these operations, each other candidate score is at most $s(p) \leq \sigma(p) + k^\star$. Let $Q^\star$ be the demotion strategy applied by an optimal strategy on E. Assume we apply the strategy $f(Q^\star)$ as defined by Lemma 12 on $\bar{E}$. Since here when we demote a candidate, other candidate scores do not increase, each candidate's final score is at most his corresponding Borda final score. As such, the sequence $f(Q^\star)$ is a valid solution for $\text{BTNCRV}(\bar{E}, k^\star, \sigma(p) + k^\star)$. □

Lemma 14. *The loop in Algorithm 3 of Algorithm 3 will run at most $2k'$ times.*

Proof. Let $s'(c)$ be the score of a candidate c right before Algorithm 3 of Algorithm 3. The demotion strategy Q found in Algorithm 3 guarantees that under RV, each candidate's score will be at most $\sigma(p) + k'$. In contrast, when applying Q on the original Borda instance E (which is possible by Lemma 12), each candidate might be awarded one additional point for each operation. As $\pi(Q) \leq k'$, this means that the score of each candidate might increase by additional k' points compared to their final RV score. Thus $s'(c) \leq \sigma(p) + 2k'$ for each $c \in C \setminus \{p\}$. As $s'(p) \geq \sigma(p)$, at most $2k'$ swaps promoting p are enough to make him win. □

Theorem 4. *Algorithm 3 is a 3-approximation algorithm for TNC under Borda.*

Proof. As $k' \leq k^\star$ (by Lemma 13), it is sufficient to show that Algorithm 3 performs at most $3k'$ operations. To see that, notice that $\pi(Q) \leq k'$ and that the loop of Algorithm 3 of Algorithm 3 runs at most $2k'$ times (by Lemma 14), where each iteration involves a single demotion operation. □

We now demonstrate that allowing candidate-specific prices drastically alters the complexity landscape

5.3 Inapproximability Results

What if the price of a demotion is a function of both the bribed voter and the demoted candidate? Indeed, assigning prices for different demographic-candidate pairs is consistent with the observation that the effect of negative campaigns changes with different demographic and targeted candidate combinations. Some demographics were shown to be more tolerant to negativity in campaigns, while others might pose a risk for a backlash against the favored candidate. In addition, the effectiveness of such attacks was shown to be dependent on candidate properties, such as ethnicity, gender, and whether the candidate is incumbent or challenging (Fridkin and Kenney 2011).

Interestingly enough, when the price function is of type $\hat{\pi}\colon N \times C \to \mathbb{R}_0^+$ ($\hat{\pi}(\ell, c)$ is the price for demoting c in v_ℓ), TNC for Borda is hard to approximate within some ratio:

Theorem 5. *With the price function $\hat{\pi}\colon N \times C \to \mathbb{R}_0^+$, for every constant $\epsilon > 0$, TNC for Borda cannot be approximated within $(1-\epsilon)\ln(m/2-1)$ in polynomial time unless* P = NP.

Proof. Recall that $\bar{n}$ and $\bar{m}$ are the number of items and the number of subsets in an instance of Set Cover, respectively. Let f be the reduction from Set Cover described in the proof of Lemma 8, adjusted such that in $f(I,k)$, for each voter ℓ with a vote in $\{v_i^1\}_{i=1}^{\bar{m}}$, $\pi(\ell, a) = 1$ and $\pi(\ell, c) = k\ln(m/2-1)$ for every $c \neq a$. In addition, every voter ℓ with vote in $V \setminus \{v_i^1\}_{i=1}^{\bar{m}}$, $\pi(\ell, c) = k\ln(m/2-1)$ for every $c \in C$.

Let us fix $\epsilon > 0$. Assume by contradiction that there exists a polynomial-time $(1-\epsilon)\ln(m/2-1)$-approximation algorithm $\mathcal{A}$ for TNC with prices $\hat{\pi}$. Assume that the optimal set-cover has size k. In this case, we know that a strategy of price k exists, by an argument very similar to the completeness argument in Lemma 8. In this case, $\mathcal{A}(f(I,k))$ will find a strategy of price of at most $(1-\epsilon)k\ln(m/2-1)$ in polynomial time.

Now focus on the strategy returned by $\mathcal{A}(f(I,k))$. By the overall price paid being at most $(1-\epsilon)k\ln(m/2-1)$, we know that only voters having votes of type v_i^1 were bribed, and that in each such operation, a was demoted. We can assume w.l.o.g. that in each operation, a was demoted to be ranked last, otherwise we can modify the operation so that this will be the case; although this modification might award a point to some additional candidates, since it will also award an additional point to p, then $\mathrm{diff}(p,c)$ for any $c \in C$ can only maintain its value or increase. Specifically, after applying these modifications p is still winning and no price increase was made.

We again reach a point where for all bribed voters a was pushed down $\bar{n}+2$ positions. Now observe only the bribed voters having votes of type v_i^1. Since for each $u \in U$ it held before that $\mathrm{diff}(p,u) = -1$, but now $\mathrm{diff}(p,u) \geq 0$, we bribed at least one voter having a vote of type v_i^1 such that $u \in S_i$, which allowed $\mathrm{diff}(p,u)$ to increase by a point. Therefore, the collection $\mathcal{S}'' = \{S_i \in \mathcal{S} \mid v_i^1 \text{ is bribed in } \mathcal{A}(f(I,k))\}$ constitutes a valid cover and $|\mathcal{S}''| \leq (1-\epsilon)k\ln(m/2-1) = (1-\epsilon)k\ln\bar{n}$.

We can now relax the assumption that we know what is the optimal set cover size k, by considering the following procedure for Set Cover: for $k = 1, \ldots, \bar{n}$, we compute $f(I,k)$ and apply $\mathcal{A}$ on the result. If the returned strategy involved price at most $(1-\epsilon)k\ln(m/2-1)$, halt and extract the set cover from the bribery strategy as described above. Notice that this procedure will have to halt and succeed at the iteration where k is indeed the optimal set cover size (if not even before). The procedure we have just described is a $(1-\epsilon)\ln\bar{n}$-approximation to Set Cover (as $m = 2\bar{n}+2$). However, this contradicts the inapproximability of Set Cover within a $(1-\epsilon)\ln\bar{n}$ factor, unless P = NP, as shown by Dinur and Steurer (2014). □

6 Destructive Variants

In DTNC, the destructive variant, our goal it to make the currently leading candidate $d \in C$ lose (by making sure that he will not be in the winner set). To this end we introduce algorithms for scoring rules, applicable even when voters have prices $\pi(\ell)$ for each voter ℓ. We study three cases: (a) where each score value of the scoring rule is

representable as an $O(\log(nm))$-bit integer (notice that the "natural" input size is $\Theta(nm)$); (b) where each voter price is representable as an $O(\log(nm))$-bit integer; (c) without any restriction on the scores and prices.

While the latter setting is the most general, we note that assuming that the scores are representable as $O(\log(nm))$-bit integers is very natural: it is trivially true for Plurality and Veto, t-approval, Borda and truncated variants thereof. Even Dowdall, for which $\alpha_j = 1/j$, can be modified to conform to this assumption, by noticing that the smallest difference between any two score values in $\boldsymbol{\alpha}$ is at least m^{-2}. Thus, by rounding each score to the nearest multiple of e.g., $1/(2nm^2)$, a candidate's final score would change by at most $1/(2m^2)$ and the order induced by candidates' final scores will be unchanged. At this point, the modified scores can be normalized to become $O(\log(nm))$-bit integers. A scoring rule not satisfying our assumption is, for example, exponential-Borda (Put and Faliszewski 2016), for which $\boldsymbol{\alpha} = (2^{m-1}, 2^{m-2}, \ldots, 2^1, 2^0)$.

Each of the algorithms will rely on an invocation of a min-Knapsack variant, discussed in the following.

6.1 Min-Knapsack Variants

In the classic 0-1-max-Knaspack problem, we are given a set of items $U = \{x_1, \ldots, x_t\}$, where each item x_i has an associated weight $w(x_i)$ and value $v(x_i)$, and a weight bound W. We should find a subset of items S maximizing the overall value $\sum_{x\in S} v(x)$ such that $\sum_{x\in S} w(x) \leq W$. The min-Knapsack variant switches the roles of values and weights: instead of the given weight bound W, we are given a value bound L. This time, our task is to minimize $\sum_{x\in S} w(x)$ such that $\sum_{x\in S} v(x) \geq L$.

Patel et al. (2021) studied a family of Knapsack generalizations collectively referred to as *fair Knapsack*. Specifically, we shall utilize one of their variants called *min-Knapsack with a bound on the number of items*—denoted as BN^{min} in their work. It is defined as follows. Let the collection $\mathcal{X} = \{X_1, \ldots, X_n\}$ be a partition of a set of items U, where for each ℓ, it holds that $X_\ell = \{x_{\ell,1}, \ldots, x_{\ell,|X_\ell|}\} \subseteq U$. Again, each such item has a weight $w(x_{\ell,j}) \geq 0$ and a value $v(x_{\ell,j}) \geq 0$. We are also given two bounds a_ℓ, b_ℓ for each $\ell \in [n]$, and a value bound L. The goal is to find a set of items S that minimizes $\sum_{x_{\ell,j}\in S} w(x_{\ell,j})$ while $\sum_{x_{\ell,j}\in S} v(x_{\ell,j}) \geq L$, which also satisfies the constraint that $a_\ell \leq |S \cap X_\ell| \leq b_\ell$ for each ℓ. In other words, in this setting there are constrains on the number of items that can be chosen from each X_ℓ. We note that in our use-case, we will always have $a_\ell = 0$ and $b_\ell = 1$, and therefore we will usually omit the references to a_ℓ and b_ℓ.

In their work, they provide a *fully-polynomial approximation scheme* (FPTAS) for the problem: for an arbitrary $\epsilon > 0$, they show that it can be approximated within $(1+\epsilon)$, in time polynomial in the input size and $1/\epsilon$.

Theorem 6 (Patel et al. (2021)). *BN^{min} has an FPTAS.*

Patel et al. (2021) do not assume anything on the nature of neither weights nor values; however, we are also interested in the case where either the weights or the values are polynomially-bounded integers (or can be normalized to become such). We therefore provide the following:

Lemma 15. *If weights are integers bounded by a polynomial in the input size, then BN^{min} can be solved exactly in polynomial time.*

Proof. Essentially, the lemma can be obtained by removing the step where the weights are scaled-down from their algorithm. However, if we prefer to show this while treating their algorithm as a black-box, then by choosing a small enough ϵ we can "force" their algorithm to return an exact solution, yet remain polynomial. This can be obtained for example by choosing $\epsilon = 1/\sum_{x\in U} w(x)$. In that case, letting $w^\star$ be the weight of the optimal solution, the additive part of the approximation becomes $\epsilon w^\star < 1$ and by the integrality assumption, the algorithm has to return an optimal solution. As in our case, $1/\epsilon$ is still polynomial, the lemma follows. □

Lemma 16. *If values are integers bounded by a polynomial in the input size, then BN^{min} can be solved exactly in polynomial time.*

Proof. We can assume w.l.o.g. that L is also an integer and that $L \leq \sum_{x \in U} v(x)$ (in the former case, it can be rounded-down to the nearest integer; in the latter case, trivially there is no solution). Let $f(k, i)$ represent the minimal weight needed in order to have a value of at least k when only choosing from $X_1, \ldots, X_i$. We can compute $f(L, n)$ (our objective) with dynamic programming using the following recursion: $f(k, i) = \min_{j=0}^{|X_i|}(f(k - v(x_{i,j}), i-1) + w(x_{i,j}))$ where $x_{i,0}$ is a placeholder item having zero weight and value representing not taking any item from X_i. The edge cases are as follows: $f(0, 0) = 0$; for every $k > 0$, $f(k, 0) = \infty$; and for each $k < 0, i \in [n]$, $f(k, i) = 0$.

Notice that filling the table is polynomial in n and L; since $L \leq \sum_{x \in U} v(x)$ and the values are polynomially-bounded integers, the lemma follows. □

6.2 Algorithm

It is sufficient that one candidate beats d, therefore, we can loop over each candidate c and find the minimal-cost strategy making c beat d. Assume we do so, and let c be such candidate. Let $Q^\star$ be the optimal strategy (unknown to us) making c beat d. Fix ℓ be a voter bribed in $Q^\star$ and let $C' \neq \emptyset$ be the candidates demoted by ℓ in $Q^\star$. The following two lemmas apply for ℓ:

Lemma 17. *The following can be assumed w.r.t. $Q^\star$ w.l.o.g.:*

(1) *If $d \notin C'$, then $c \succ_\ell d$ and the candidates in C' were ranked consecutively, immediately before c. Overall, as a result of the operations, c is promoted by $|C'|$ positions.*
(2) *If $d \in C'$, then d was the first to be demoted, and was demoted to be last. Following the demotion of d, the candidates in $C' \setminus \{d\}$ were ranked consecutively, immediately before c. Overall, as a result of the operations, c is promoted by $|C'| - 1 + \mathbb{1}[d \succ_\ell c]$ positions.*

Proof. The lemma essentially splits the bribed voters into two cases: the ones who demoted d and the ones who did not.

For the first type, assume that it holds that $d \succ_\ell c$. Let $c' \in C'$ be a voter demoted by ℓ. Then we can demote d instead of c' and have an even greater effect on $\mathrm{diff}(d, c)$. However, this would place ℓ in the second type of voters. It follows that we can w.l.o.g. assume that if ℓ is of the first type, then $c \succ_\ell d$. Notice that there is no point in demoting a candidate $c' \neq d$ if $c' \prec_\ell c$: d will not lose any points and c will not gain any points. Thus, we are only interested in demoting candidates ranked *before* c. Then, if we demote any $|C'|$ candidates ranked before c, the choice of which candidates is insignificant; in any case c be promoted by $|C'|$ positions and we can thus assume that we always demote the $|C'|$ candidates who were ranked *consecutively, immediately* before c.

For the second type, note that in our model—where all demotion levels are allowed and have the same price—we can always change the order of operations pertaining to ℓ and get to the same end result. Thus we can assume that d was the first to be demoted. Then, we always gain possibly more by demoting d to be last, without any added cost. As a result of demoting d, if before he was demoted it held that $d \succ_\ell c$, then c will also be promoted by 1 position.

Now, assume that we have just demoted d. The choice of which $|C' \setminus \{d\}|$ candidates to demote now follows the same logic in the above case: we can assume we demote the $|C' \setminus \{d\}|$ candidates who are now ranked *consecutively, immediately* before c, and that c is promoted $|C' \setminus \{d\}| = |C'| - 1$ *additional* positions as a result of this. □

Lemma 18. *Let $\Delta_\ell(C')$ be the change in $\mathrm{diff}(d, c)$ as a result of the demotion of C' by ℓ. If $d \notin C'$ then $\Delta_\ell(C') = \alpha_{v_\ell(c)-|C'|} - \alpha_{v_\ell(c)}$. Otherwise*

$$\Delta_\ell(C') = (\alpha_{v_\ell(d)} - \alpha_m) + (\alpha_{v_\ell(c)-|C'|+1-\mathbb{1}[d \succ_\ell c]} - \alpha_{v_\ell(c)}) .$$

Algorithm 4: Approximation algorithm for making c beat d with minimum cost

input: an election (C, V), a price function $\pi\colon N \to \mathbb{R}_0^+$, and candidates c and d.

`/* Constructing a BN^min instance: */`

$L \leftarrow \text{diff}(d, c)$ `/* diff(d, c) is the original score difference */`

Let w, v be two new, currently empty, mappings.

foreach $\ell \in N$ **do**

  Let C_ℓ be the collection of all sets C' of candidates such that the candidates in C' are ranked consecutively, immediately before c, including the empty set.

  $\bar{C}_\ell \leftarrow C_\ell \cup \{C' \cup \{d\} \mid C' \in C_\ell\}$

  $X_\ell \leftarrow \{(\ell, C') \mid C' \in \bar{C}_\ell\}$

  foreach $C' \in X_\ell$ **do**

    $w((\ell, C')) \leftarrow \pi(\ell) \cdot |C'|$

    $v((\ell, C')) \leftarrow \Delta_\ell(C')$

$U \leftarrow \bigcup_{\ell=1}^{n} X_\ell, \mathcal{X} \leftarrow \{X_1, \ldots, X_n\}$

Solve the $\text{BN}^{\min}$ instance $(U, \mathcal{X}, w, v, L)$. Let $S = [x_\ell]_\ell$ be the solution, where x_ℓ are the candidates chosen to be demoted by ℓ.

return the following strategy:

  foreach $\ell \in N$ **do**

    If $d \in x_\ell$, demote d to be last in v_ℓ. Demote the candidates in $x_\ell \setminus \{d\}$ to be ranked consecutively, immediately after c in v_ℓ.

Proof. The first case follows directly from the previous lemma. For the second, we claim that $\Delta_\ell(C') = (\alpha_{v_\ell(d)} - \alpha_m) + (\alpha_{v_\ell(c)-|C'|+1-\mathbb{1}[d \succ_\ell c]} - \alpha_{v_\ell(c)})$. As a result of d's demotion, $\text{diff}(d, c)$ increases by $\alpha_{v_\ell(d)} - \alpha_m$ points (the first term). The second term follows directly from the previous lemma. □

For each voter ℓ, we do not know whether ℓ is bribed and if so, which type of bribed voter will he be. As such, the previous two lemmas show us that if a voter ℓ is bribed, we only need to cover a polynomial number of options for the identity of the set of candidates demoted by ℓ: this is, the collection of all subsets of candidates which form a consecutive ranking immediately before c, plus the possible addition of d. The algorithm follows this logic: let C_ℓ be the collection of all nonempty sets C' of candidates such that the candidates in C' are ranked *consecutively*, *immediately* before c. For example, if $v_\ell = c_3 \succ c_9 \succ d \succ c_5 \succ c \succ \cdots$, then $C_\ell = \{\{c_5\}, \{c_5, d\}, \{c_5, d, c_9\}, \{c_5, d, c_9, c_3\}\}$. Notice that some sets in C_ℓ might include d. Now let $\bar{C}_\ell = C_\ell \cup \{C' \cup \{d\} \mid C' \in C_\ell\}$. In words, $\bar{C}_\ell$ includes all sets in C_ℓ; for such sets C' which do not include d, a copy of the set—this time including d—is added.

We can now find a sequence of operations by a reduction to $\text{BN}^{\min}$. We let the bound L equal $\text{diff}(d, c)$. Focus on a voter ℓ, we define the item-set $X_\ell = \{(\ell, C') \mid C' \in \bar{C}_\ell\}$, where the item (ℓ, C') represents the event that ℓ demotes C'. For each item (ℓ, C'), we define $w((\ell, C')) = \pi(\ell) \cdot |C'|$, and—following lemma 18—$v((\ell, C')) = \Delta_\ell(C')$. Each item thus corresponds to a potential set of candidates to be demoted by ℓ, covering a sufficient subset of cases according to Lemma 17; the price of an item (ℓ, C') is simply the per-voter price $\pi(\ell)$ multiplied by the number of candidates in C', and the value if the potential change in $\text{diff}(d, c)$. The solution to this $\text{BN}^{\min}$ instance thus directly describes which candidates to demote by each voter ℓ. For each such voter, as mentioned, d—if demoted—will always be demoted to be last. Other candidates will be demoted to be ranked consecutively, immediately following c. The procedures above are summarized as Algorithm 4.

Our main results for this section now follow according to the assumptions made on either the scores or prices, and dictate the choice of the $\text{BN}^{\min}$ algorithm to be used:

Theorem 7. *The DTNC problem with prices which are a function of the voters can be solved in polynomial time, if at least one of the following holds:*

(1) *For the given scoring rule, each score can be represented as an $O(\log(nm))$-bit integer.*
(2) *Each voter price can be represented as an $O(\log(nm))$-bit integer.*

Note that while the first item is a requirement on the scoring rule, the second applies to any scoring rule.

Proof. Follows from the above discussion, with the choice of Lemma 16 for the two Knapsack invocations in the former, and the choice of Lemma 15 for the two Knapsack invocations of the latter. □

Theorem 8. *For every scoring rule, the DTNC problem with prices which are a function of the voters has an FPTAS.*

Proof. Follows from the above discussion, with the choice of Theorem 6 for the two Knapsack invocations. □

7 Conclusions

In this work we studied a de facto standard in political campaigns: targeted negative campaigning. While being so widespread—as far as we know—it was not modeled or studied from a computational perspective[3]. Our goal was to define a model capturing a realistic—yet relatively simple and appealing—setting. Several interesting insights arise from this work and we detail them in the following. Some of them might pave way for future work.

A motivation for our model was the intuition that Swap Bribery's pricing scheme tends to be too granular: Swap Bribery models bribery and campaigning in a very "local", swap-oriented level where the demotion price depends on the identity of the candidates being swapped, and usually will also depend on the desired demotion level. We contrasted this with the more "global" pricing of demotions in our model: while demoting a candidate to become last (capturing that the candidate "went out of favor" with the voter) may consist of several swaps in Swap Bribery, it is essentially a single logical sequence of swap operations, and thus charged accordingly by our model. As discussed in Section 3, when trying to model this more global setting in Swap Bribery, the ability to achieve price asymmetry between promotions and demotions is violated.

With respect to the hardness of bribery, our results show that our model is a sweet-spot between Swap Bribery and Shift Bribery—which models positive campaigns. As such our results for both the unpriced and priced variants are somewhere between the $(1+\epsilon)$ Shift Bribery approximation and the general inapproximability of Swap Bribery for many voting rules.

We provided results for both t-approval and Borda, yet we feel that there is headroom for finding stronger approximation guarantees or lower bounds for our problems. Mainly, the dependency on t in the approximation factor for t-Approval seems like a natural opportunity for improvement.

Another goal would be to better understand the complexity of t-approval-TNC when $t = 2$ and when t is not fixed; as discussed, our hardness proof covers the fixed $t \geq 3$ case, and our approximation algorithm returns an *exact* solution when $t = 1$. The $t = 2$ case thus remains unclear.

For Borda, a gap exists between the approximability of the case where each demotion has a unit price and is independent of the voter involved, and the hardness-of-approximation result when the price is a function of both the voter and the candidate demoted. This naturally leaves room for better understanding the feasibility of approximation when price is of the form $\pi\colon N \to \mathbb{R}_0^+$, i.e., voter-dependent.

We also note that another natural venue for research is for more general scoring rules results (for example, dichotomy results for hardness-of-approximation) and also other voting rules, such as Copeland and Maximin.

[3]While Swap Bribery can be used to model negative settings, it has the limitations detailed in Proposition 1.

Acknowledgments

This research has been partly supported by the Israel Science Foundation under grant 1092/24.

References

W. Andrews, D. Keating, and K. Yourish. 2012. *Mad Money: TV Ads in the 2012 Presidential Campaign*. https://www.washingtonpost.com/wp-srv/special/politics/track-presidential-campaign-ads-2012/. Accessed: 2020-08-11. (2012).

H. Barr, Y. Aumann, and S. Kraus. 2025. "Bribery for Coalitions in Parliamentary Elections." *CoRR*, abs/2503.14707. arXiv: 2503.14707. doi:10.48550/ARXIV.2503.14707.

D. Baumeister, P. Faliszewski, J. Lang, and J. Rothe. 2012. "Campaigns for Lazy Voters: Truncated Ballots." In: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS, 577–584.

R. Bredereck, J. Chen, P. Faliszewski, A. Nichterlein, and R. Niedermeier. 2016. "Prices Matter for the Parameterized Complexity of Shift Bribery." *Inf. Comput.*, 251, 140–164.

R. Bredereck, P. Faliszewski, M. Furdyna, A. Kaczmarczyk, J. Kaczmarek, M. Lackner, C. Laußmann, J. Rothe, and T. Seeger. 2025. "How to tamper with a Parliament: Strategic campaigns in apportionment elections." *Journal of Computer and System Sciences*, 155, 103700.

R. Bredereck, P. Faliszewski, R. Niedermeier, and N. Talmon. 2021. "Complexity of Shift Bribery in Committee Elections." *ACM Trans. Comput. Theory*, 13, 3, 20:1–20:25. doi:10.1145/3470647.

R. Bredereck, P. Faliszewski, R. Niedermeier, and N. Talmon. 2016. "Large-Scale Election Campaigns: Combinatorial Shift Bribery." *J. Artif. Intell. Res.*, 55, 603–652.

D. Briskorn, G. Erdélyi, and C. Reger. 2016. "Bribery in k-Approval and k-Veto Under Partial Information: (Extended Abstract)." In: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems*. ACM, 1299–1300.

D. Cary. 2011. "Estimating the Margin of Victory for Instant-Runoff Voting." In: *Proceedings of the 2011 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections*. USENIX Association.

M. Castiglioni, D. Ferraioli, and N. Gatti. 2020. "Election Control in Social Networks via Edge Addition or Removal." In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. AAAI Press, 1878–1885.

Q. T. Cicero. 2012. *How to win an election: An ancient guide for modern politicians*. Translated by Philip Freeman. Princeton University Press.

P. Dey and Y. Narahari. 2015. "Estimating the Margin of Victory of an Election Using Sampling." In: *Proceedings of the 24th International Joint Conference on Artificial Intelligence*. AAAI Press, 1120–1126.

I. Dinur and D. Steurer. 2014. "Analytical Approach to Parallel Repetition." In: *Proceedings of the 46th Symposium on Theory of Computing*. ACM, 624–633.

B. Dorn and D. Krüger. 2016. "On the Hardness of Bribery Variants in Voting with CP-Nets." *Annals of Mathematics and Artificial Intelligence*, 77, 3-4, 251–279.

J. Edmonds and R. M. Karp. 1972. "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems." *Journal of the ACM (JACM)*, 19, 2, 248–264.

E. Elkind and P. Faliszewski. 2010. "Approximation Algorithms for Campaign Management." In: *Proceedings of Internet and Network Economics - 6th International Workshop* (Lecture Notes in Computer Science). Vol. 6484. Springer, 473–482.

E. Elkind, P. Faliszewski, and A. M. Slinko. 2009. "Swap Bribery." In: *Proceedings of Algorithmic Game Theory, 2nd International Symposium* (Lecture Notes in Computer Science). Vol. 5814. Springer, 299–310.

P. Faliszewski. 2008. "Nonuniform Bribery." In: *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS, 1569–1572.

P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. 2009. "How Hard Is Bribery in Elections?" *J. Artif. Intell. Res.*, 35, 485–532.

P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. 2006. "The Complexity of Bribery in Elections." In: *Proceedings of the 21st National Conference on Artificial Intelligence*. AAAI Press, 641–646.

P. Faliszewski, E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. 2009. "Llull and Copeland Voting Computationally Resist Bribery and Constructive Control." *J. Artif. Intell. Res.*, 35, 275–341.

P. Faliszewski, P. Manurangsi, and K. Sornat. 2021. "Approximation and hardness of Shift-Bribery." *Artif. Intell.*, 298, 103520. doi:10.1016/J.ARTINT.2021.103520.

P. Faliszewski and J. Rothe. 2016. "Control and Bribery in Voting." In: *Handbook of Computational Social Choice*. Cambridge University Press, 146–168.

E. Franklin Fowler and T. N. Ridout. 2014. "Political Advertising in 2014: The Year of the Outside Group." *The Forum*, 12, 4, 663–684.

K. L. Fridkin and P. Kenney. 2011. "Variability in Citizens' Reactions to Different Types of Negative Campaigns." *American Journal of Political Science*, 55, 2, 307–325.

T. F. Gonzalez. 1985. "Clustering to Minimize the Maximum Intercluster Distance." *Theor. Comput. Sci.*, 38, 293–306. doi:10.1016/0304-3975(85)90224-5.

S. Greenhouse. 2000. *THE 2000 CAMPAIGN: THE GREEN PARTY; Nader, in Harlem, Attacks Gore and Bush With Gusto*. https://www.nytimes.com/2000/11/07/us/2000-campaign-green-party-nader-harlem-attacks-gore-bush-with-gusto.html. Accessed: 2025-02-03. (2000).

M. Haselmayer. 2019. "Negative Campaigning and its Consequences: A Review and a Look Ahead." *French Politics*, 17, 355–372.

E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. 2022. "The complexity of online bribery in sequential elections." *Journal of Computer and System Sciences*, 127, 66–90.

G. Iyer, D. Soberman, and J. M. Villas-Boas. 2005. "The Targeting of Advertising." *Marketing Science*, 24, 3, 461–476.

J. P. Johnson. 2013. "Targeted Advertising and Advertising Avoidance." *The RAND Journal of Economics*, 44, 1, 128–144.

A. Kaczmarczyk and P. Faliszewski. 2019. "Algorithms for Destructive Shift Bribery." *Autonomous Agents and Multi-Agent Systems*, 33, 275–297.

N. Karia, F. Mallick, and P. Dey. 2023. "How hard is safe bribery?" *Theoretical Computer Science*, 979, 114156.

O. Keller, A. Hassidim, and N. Hazon. 2019. "Approximating Weighted and Priced Bribery in Scoring Rules." *J. Artif. Intell. Res.*, 66, 1057–1098.

B. Kusek, R. Bredereck, P. Faliszewski, A. Kaczmarczyk, and D. Knop. 2023. "Bribery Can Get Harder in Structured Multiwinner Approval Election." In: *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 1725–1733.

T. R. Magrino, R. L. Rivest, and E. Shen. 2011. "Computing the Margin of Victory in IRV Elections." In: *Proceedings of the 2011 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections*. USENIX Association.

N. Mattei, M. S. Pini, K. B. Venable, and F. Rossi. 2012. "Bribery in Voting over Combinatorial Domains is Easy." In: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS, 1407–1408.

K. Mattes and D. P. Redlawsk. 2014. *The positive case for negative campaigning*. University of Chicago Press.

C. Maushagen, M. Neveling, J. Rothe, and A.-K. Selker. 2022. "Complexity of shift bribery for iterative voting rules." *Annals of Mathematics and Artificial Intelligence*, 90, 10, 1017–1054.

C. Maushagen, M. Neveling, J. Rothe, and A. Selker. 2018. "Complexity of Shift Bribery in Iterative Elections." In: *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS / ACM, 1567–1575.

M. A. Mehrizi, F. Corò, E. Cruciani, G. D'Angelo, and S. Ponziani. 2019. "Models and Algorithms for Election Control through Influence Maximization." In: *Proceedings of the 20th Italian Conference on Theoretical Computer Science* (CEUR Workshop Proceedings). Vol. 2504. CEUR-WS.org, 97–103.

D. Patel, A. Khan, and A. Louis. 2021. "Group Fairness for Knapsack Problems." In: *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems*. ACM, 1001–1009. https://dl.acm.org/doi/10.5555/3463952.3464069.

M. S. Pini, F. Rossi, and K. B. Venable. 2013. "Bribery in Voting With Soft Constraints." In: *Proceedings of the 27th AAAI Conference on Artificial Intelligence*. AAAI Press, 803–809.

T. Put and P. Faliszewski. 2016. "The Complexity of Voter Control and Shift Bribery Under Parliament Choosing Rules." *Trans. Comput. Collect. Intell.*, 23, 29–50.

I. Schlotter, P. Faliszewski, and E. Elkind. 2017. "Campaign Management Under Approval-Driven Voting Rules." *Algorithmica*, 77, 1, 84–115.

D. Shiryaev, L. Yu, and E. Elkind. 2013. "On Elections with Robust Winners." In: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS, 415–422.

L. Tao, L. Chen, L. Xu, W. Shi, M. M. U. Zaman, and A. Sunny. 2025. "Bribery in elections with randomly selected voters: Hardness and algorithm." *Theoretical Computer Science*, 1036, 115150.

L. Tao, L. Chen, L. Xu, S. Xu, Z. Gao, and W. Shi. 2023. "Electoral manipulation via influence: probabilistic model." *Autonomous Agents and Multi-Agent Systems*, 37, 1, 18.

J. C. Wong. 2020. *One Year Inside Trump's Monumental Facebook Campaign*. https://www.theguardian.com/us-news/2020/jan/28/donald-trump-facebook-ad-campaign-2020-election. Accessed: 2023-06-02. (2020).

L. Xia. 2012. "Computing the margin of victory for various voting rules." In: *Proceedings of the 13th ACM Conference on Electronic Commerce*. ACM, 982–999.

A. Zagoury, O. Keller, A. Hassidim, and N. Hazon. 2021. "Targeted Negative Campaigning: Complexity and Approximations." In: *Thirty-Fifth AAAI Conference on Artificial Intelligence*. AAAI Press, 5768–5778. https://ojs.aaai.org/index.php/AAAI/article/view/16723.

Received 15 December 2025; accepted 05 February 2026