

# Targeted Negative Campaigning: Complexity and Approximations

Avishai Zagoury,<sup>1</sup> Orgad Keller,<sup>2</sup> Avinatan Hassidim,<sup>1,2</sup> Noam Hazon<sup>3</sup>

<sup>1</sup>Department of Computer Science, Bar-Ilan University, Israel

<sup>2</sup>Google Research

<sup>3</sup>Department of Computer Science, Ariel University, Israel

avishaizag@gmail.com, orgad@google.com, avinatanh@gmail.com, noamh@ariel.ac.il

## Abstract

Given the ubiquity of negative campaigning in recent political elections, we find it important to study its properties from a 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  $t$ -factor approximation for  $t$ -approval for every fixed  $t$ , 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.

## Introduction

Recent years have seen negative campaigning becoming ubiquitous in elections (?). For instance, according to studies by the Wesleyan Media Project (WMP)—which monitors the content and volume of political advertising in the United States (?)—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 favoured candidate to his or her opponent. In the 2012 U.S. presidential election—according to an analysis by The Washington Post (?)—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 a novel approach and its importance traces back to antiquity; in a 64 BC letter by Quintus Tullius Cicero (?), to his brother Marcus, running for the consul of Rome, he writes: “*It also wouldn’t hurt to remind them of what scoundrels your opponents are and to smear these men at every opportunity [...]*”

Haselmayer (?) 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 (?). That is, many platforms allow the advertisers to deliver a user-specific content, based on the 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 (?). Combining targeted advertising with negative campaigning can thus be a very useful approach (?). 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 (?) 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.

**Our Contributions.** We prove that for  $t \geq 3$ ,  $t$ -approval-TNC is NP-hard, but that if  $t$  is fixed, it can be approximated in polynomial-time within a factor of  $t$ , even when each voter has a different price. The same algorithm can compute the exact solution for the plurality scoring rule. 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  $(1 - \epsilon) \ln(m/2 - 1)$  unless  $P = NP$ , where  $m$  is the number of candidates. We also introduce a destructive variant and provide an exact solution for a wide array of scoring rules, even with a varying price per voter.

**Related Work.** The BRIBERY problem was originally formulated by ??, and has been studied extensively in recent years, usually in the context of hardness (see a survey by ?) but also in the context of approximation (??). Several types of bribery problems were studied; in SWAP BRIBERY (??), 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 (?????). In addition, many voting models were considered, e.g., truncated ballots and partial information (??), soft constraints (?), and CP-nets (??).

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 (?), yet approximable within a  $(1 + \epsilon)$ -factor for any scoring rule (?). 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; ?).

In most of the above mentioned 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 ?, and under the name MARGIN OF VICTORY (????). Other destructive variants of note are DESTRUCTIVE SWAP BRIBERY (?) and DESTRUCTIVE SHIFT BRIBERY (?). A detailed comparison between our work and existing models is provided later.

Some computational aspects of negativity in elections were previously studied in the context of social networks (??).

## Preliminaries

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

indicate that  $c$  is preferred to  $c'$  by voter  $\ell$ .

A *scoring rule* for  $m$  candidates is described by a vector  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_m)$  of non-negative numbers such that  $\alpha_1 \geq \alpha_2 \geq \dots \geq \alpha_m \geq 0$ . This rule is applied as follows: each voter  $\ell$  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  $\alpha = (m-1, m-2, \dots, 1, 0)$  and  $t$ -approval for which  $\alpha = (\mathbf{1}^t; \mathbf{0}^{m-t})$  where 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  $\alpha = (\mathbf{1}^{m-1}; 0)$ .

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 can be repeated.

**Notation.** We denote the initial score 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  $\vec{C}$  denote the sequence containing the elements of  $C$  in some arbitrary yet predetermined order. We use  $\overleftarrow{C}$  to denote the reverse of  $\vec{C}$ . For any subset  $C' \subseteq C$ , we let  $\vec{C}'$  (resp.  $\overleftarrow{C}'$ ) denote the sub-sequence of  $\vec{C}$  (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  $C'$ , whose size is  $\binom{|C'|}{t}$ . We let  $\mathbf{1}[\cdot]$  denote the indicator function where  $\mathbf{1}_{C'}(c)$  is a shorthand for  $\mathbf{1}[c \in C']$ . We also let  $[t]$  denote the set  $\{1, \dots, t\}$ .

We define the following problems:

**TNC.** 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  $(\ell, c_i, \delta)$  with the meaning that  $c_i$  is demoted  $\delta$  positions in (the current)  $v_\ell$ . The operations in  $Q$  are performed sequentially, in order. 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  $(\ell, c_i, \delta)$  is a function of either  $\ell$ , or both  $\ell$  and  $c_i$ . With a slight abuse of notation, we use  $\pi$  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  $\pi(Q) = \sum_{(\ell, c_i, \delta) \in Q} \pi(\ell, c_i)$ .

**DTNC.** We also discuss a destructive version of TNC named DTNC (Destructive Targeted Negative Campaign), 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.

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

3SET COVER is the variant where  $|S| = 3$  for all  $S \in \mathcal{S}$ . Both problems are NP-hard (?).<sup>1</sup>

**MIN COST FLOW.** Given a graph  $G = (V, E)$  where each edge  $e \in E$  has a capacity  $\gamma(e)$  and a cost  $a(e)$ , two specified vertices  $s$  and  $d$ , and a value  $T$ , the goal is to find a flow  $f: E \rightarrow \mathbb{R}_0^+$  from  $s$  to  $d$  subject to the capacity constraints  $f(e) \leq \gamma(e)$  for each  $e \in E$ , such that the flow value is  $|f| = \sum_{u \in N(s)} f(s, u) = T$  (where  $N(s)$  are  $s$ 's neighbors) and its total cost  $\mathbf{A}(f) = \sum_{e \in E} a(e)f(e)$  is minimal (?). It is known that integral capacities lead to integral edge flow values.

## 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; ?). Therefore, if a voter has several topics she might care about in the context of a political candidate—possibly in 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  $\delta$  positions will cost a unit, but *promoting* him by  $\delta$  positions will cost  $\delta$  (as it translates

<sup>1</sup>In their definition of 3SET COVER, all subsets have size at most 3. The reduction to our definition is easy using a padding argument.

to  $\delta$  demotions), and this is consistent with the effectiveness of negativity argued by political science researchers (?). 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. This is shown by the following theorem.

**Theorem 1.** SWAP BRIBERY does not generalize TNC.

*Proof.* Assume a TNC instance with a unit price for any demotion operation, and assume by contradiction that it can be modeled under SWAP BRIBERY. Now consider a preference order  $v_\ell = a \succ b \succ c$ . Let  $\pi_\ell(\cdot, \cdot)$  be  $\ell$ 's SWAP BRIBERY price function. Since demoting  $a$  by one position costs 1, then  $\pi_\ell(a, b) = 1$ . Similarly, since demoting  $b$  by one position costs 1, then  $\pi_\ell(b, c) = 1$ . However, since in our instance any demotion has a unit price, demoting  $a$  all the way down has price  $\pi_\ell(a, b) + \pi_\ell(a, c) = 1$  and thus  $\pi_\ell(a, c) = 0$ . However, under the SWAP BRIBERY price function we have just built, promoting  $c$  to the top position will cost  $\pi_\ell(b, c) + \pi_\ell(a, c) = 1$ . However, as it involves two demotion operations (for  $a$  and  $b$ ), under the TNC framework the price should be 2—a contradiction.  $\square$

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 ?. However, our model allows the demotion of a candidate  $c$  in order to strengthen another candidate  $c'$  with the goal of making the currently leading candidate  $d$  lose. In contrast, in DESTRUCTIVE SHIFT BRIBERY, only operations directly demoting  $d$  are allowed. As a result, the cost of the strategy might be radically different. Consider for example the “all-or-nothing” pricing model discussed by ? 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.

**Theorem 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  $N = [n]$ ,  $C = \{c_1, c_2, \dots, c_{n-2}, a, b, d\}$  (so that  $|C| = |N| + 1$ ), and fix the scoring rule  $\alpha = (2n^2, 3n, 0, \dots, 0)$ . 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.  $\square$

### $t$ -Approval

In this section, for any fixed  $t \geq 3$ , we will show NP-hardness and a constant-factor approximation for  $t$ -approval.

#### NP-Hardness

We claim the following:

**Theorem 3.** *Determining whether there exists a solution with at most  $k$  demotion operations to  $t$ -approval-TNC is NP-hard for every fixed  $t \geq 3$ .*

We provide a proof sketch for the case  $t = 3$  (the other cases are similar) and defer the complete proof to the full version of the paper. The idea is to reduce a 3SET COVER instance with a cover size  $k$  to a 3-approval-TNC instance with  $3k$  allowed demotion operations as follows. For every  $S \in \mathcal{S}$ , we define a voter who ranks the elements of  $S$  at the 3 top positions, then two unique dummy candidates, and then  $p$ . We refer to these voters as the main voters. We also add many ‘filler’ voters and more dummy candidates until (a)  $\sigma(p) = \bar{m}$ ; (b) for each  $u \in U$ ,  $\sigma(u) = \sigma(p) + k + 1$ , and (c) there exist a set  $D'$  of  $4k$  dummy candidates where each of them has the score  $\sigma(p) + k$ . Then, the only way to make sure  $p$  does not lose to the candidates in  $D'$  is by having  $p$  gain exactly  $k$  points. However this is the maximum  $p$  can gain and this can only be achieved by bribing exactly  $k$  main voters, each of them 3 times: demoting the top 3 candidates, thus having  $p$  take the third place in each. However, at this point some candidates in  $U$  might still have a point margin over  $p$ —unless the main voters that we have bribed exactly correspond to a cover of the 3SET COVER instance—in which case each candidate  $u \in U$  now has a final score  $s(u) \leq \bar{m} + k = s(p)$ .

Moving away from the case of a fixed  $t$ , we note that *Veto* can be solved in polynomial time by iterating over every voter  $\ell$  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  $\text{diff}(p, c)$  by 1 for every non-preferred candidate  $c$ , besides the leading one  $c'$ , for which  $\text{diff}(p, c')$  is increased by 2. Interestingly, this method relies on the co-winner assumption: if we get to the point where no voter  $\ell$  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.

#### Approximation

In this section we discuss a  $t$ -factor approximation for  $t$ -approval for every fixed  $t$ .

For any set  $S \in \binom{C}{t}$ , we let  $\bar{\pi}_\ell(S) = \max_{c \in S} v_\ell(c) - t$ . We can assume w.l.o.g. that we know the value  $b$ , the final score of  $p$  in the optimal demotion strategy (since we can

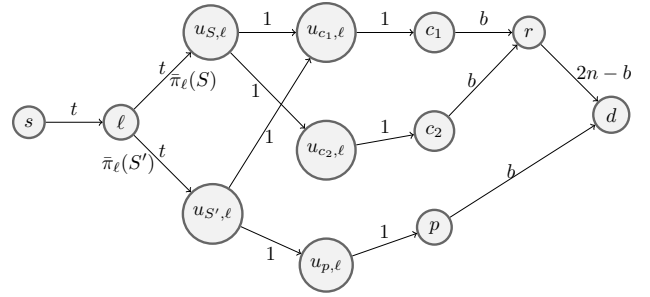


Figure 1: A sub-graph of  $G(b)$  for 2-approval. Assume that  $S = \{c_1, c_2\}$  and  $S' = \{c_1, p\}$ . Edge labels are their capacity. Where edges have a second label, it is their cost. All other edge costs are zero.

try all the possible values for  $b$  and choose the one resulting in the cheapest final solution). We shall construct a priced flow network  $G(b)$  in which units of flow represent awarded points, as follows. The vertex set  $U = \{s, d, r\} \cup N \cup U_1 \cup U_2 \cup C$  is comprised of the following ‘layers’ (visualized in fig. 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  $\ell$  (resp. awarded to  $c$ ).
- $U_1 = \{u_{S,\ell} \mid S \in \binom{C}{t}, \ell \in N\}$ ; a flow unit passing through a vertex  $u_{S,\ell} \in U_1$  represents a point awarded by  $\ell$  to a candidate in  $S$  (the choice of which candidate will be immediately discussed).
- $U_2 = \{u_{c,\ell} \mid c \in C, \ell \in N\} \in U_2$ ; a flow unit passing through a vertex  $u_{c,\ell}$  represents a point awarded by  $\ell$  to the candidate  $c$ .
- $s$  and  $d$  are the source and destination vertices.  $r$  represents any candidate other than  $p$ .

The edge set  $E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5 \cup E_6$  is comprised of the following, where  $\gamma(u, v)$  is the capacity of an edge  $(u, v)$  and  $a(u, v)$  is its cost.

- $E_1 = \{(s, \ell) \mid \ell \in N\}$  with all capacities equal to  $t$ , guarantees that each voter has at most  $t$  points to distribute.
- $E_2 = \{(\ell, u_{S,\ell}) \mid \ell \in N, S \in \binom{C}{t}\}$  with capacities equal to  $t$ , will be discussed later.
- $E_3 = \{(u_{S,\ell}, u_{c,\ell}) \mid \ell \in N, c \in S\}$  with all capacities equal to 1, guarantees that each candidate can receive at most one point from a subset he is a member of.
- $E_4 = \{(u_{c,\ell}, c) \mid c \in C, \ell \in N\}$  with all capacities equal to 1, guarantees that each candidate can receive at most one point from a specific voter.
- $E_5 = \{(c, r) \mid c \in C \setminus \{p\}\}$ , with all capacities equal to  $b$  guarantees that each candidate will receive at most  $b$  points (where  $b$  will be  $p$ ’s final score).
- $E_6 = \{(p, d), (r, d)\}$ , 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  $b$ .

For each edge  $(\ell, u_{S,\ell}) \in E_2$ , we let  $a(\ell, u_{S,\ell}) = \bar{\pi}_\ell(S)$ . All other edge costs are 0. Finally we let  $G(b) = (U, E, \gamma, a)$ . An example of a sub-graph of  $G(b)$  is shown in fig. 1.

The main procedure is the following. Given  $b$ , construct  $G(b)$ , and run a *minimal cost flow* algorithm on it with a desired flow value  $tn$  (?). If it failed to find such a flow then return a *fail* status; otherwise, denote the resulting network flow as  $f$ . We shall modify  $f$  to become another flow  $f'$  as follows. For each voter  $\ell$ , let  $S'_\ell = \{c \mid f(u_{c,\ell}, c) = 1\}$ . We will prove that  $|S'_\ell| = t$ . We define  $f'(\ell, u_{S'_\ell,\ell}) = t$ ,  $f'(u_{S'_\ell,\ell}, u_{c,\ell}) = \mathbf{1}_{S'_\ell}(c)$ ,  $f'(\ell, u_{S,\ell}) = 0$  for all  $S \neq S'_\ell$ , and  $f'(u_{S,\ell}, u_{c,\ell}) = 0$  for all  $S \neq S'_\ell$ . Based on  $f'$ , we compute the following demotion strategy  $Q$ : perform the necessary demotions such that for each  $\ell$ , the candidates in  $S'_\ell$  become the top  $t$  candidates in  $v_\ell$ ; this is done by demoting all candidates  $c' \notin S'_\ell$  for which  $v_\ell(c') < \max_{c \in S'_\ell} v_\ell(c)$  to the bottom of  $v_\ell$ .

Our approximation algorithm, denoted  $\mathcal{M}$ , repeats the above procedure for each  $b \in [n]$ , and returns the strategy  $Q$  having the minimum cost out of all strategies computed.

Let  $Q^*$  be the optimal demotion strategy. In the following lemmas, we will argue that the above algorithm returns a demotion strategy  $Q$  for which  $\pi(Q) \leq t\pi(Q^*)$ .

**Lemma 1.** *For a size- $t$  set  $S$ , the minimal price for making all the candidate in  $S$  become the top  $t$  candidates in  $v_\ell$  is  $\bar{\pi}_\ell(S) = \max_{c \in S} v_\ell(c) - t$ .*

*Proof.* Let  $c' = \arg \max_{c \in S} v_\ell(c)$  be the bottom-most candidate in  $S$  w.r.t.  $v_\ell$ . The number of candidates not in  $S$  who are ranked before him in  $v_\ell$  is exactly  $v_\ell(c') - t$ .  $\square$

Let  $b^*$  be the final score of  $p$  under  $Q^*$ .

**Lemma 2.** *There exists a potential flow  $f^*$  in  $G(b^*)$  where  $|f^*| = tn$  and  $\mathbf{A}(f^*) = t \cdot \pi(Q^*)$ .*

*Proof.* We define  $f^*$  in  $G(b^*)$  as follows. For each  $\ell$ , let  $S'_\ell$  be the candidates  $\ell$  approves of following the demotion operations in  $Q^*$ , and let  $s^*(c)$  be the resulting score of  $c$ . Set  $f^*(\ell, u_{S'_\ell,\ell}) = t$ ,  $f^*(u_{S'_\ell,\ell}, u_{c,\ell}) = \mathbf{1}_{S'_\ell}(c)$ ,  $f^*(u_{c,\ell}, c) = \mathbf{1}_{S'_\ell}(c)$ ,  $f^*(c, r) = s^*(c)$ ,  $f^*(r, d) = \sum_{c \in C \setminus \{p\}} s^*(c) = tn - b^*$ ,  $f(p, d) = b^*$ . All other edges will have 0 flow. It can be easily verified that all the flow conditions are satisfied, that  $|f^*| = f(r, d) + f(p, d) = tn$ , and that  $\mathbf{A}(f^*) = \sum_\ell a(\ell, u_{S'_\ell,\ell}) f^*(\ell, u_{S'_\ell,\ell}) = t \cdot \sum_\ell a(\ell, u_{S'_\ell,\ell}) = t \cdot \sum_\ell \bar{\pi}_\ell(S'_\ell) = t \cdot \pi(Q^*)$ .  $\square$

The following lemmas assume that the algorithm did not fail on  $G(b)$ , and thus  $|f| = tn$ .

**Lemma 3.** *For each voter  $\ell$ ,  $|S'_\ell| = t$ .  $f'$  is well-defined, is a valid flow, and  $|f'| = tn$ .*

*Proof.* We have that  $|f| = tn$ . As every voter  $\ell$  has an incoming capacity of  $t$ , each voter transfers exactly  $t$  flow units to candidates. Fix a voter  $\ell$ . Since  $\gamma(u_{c,\ell}, c) = 1$  for each  $c$ , there must be exactly  $t$  candidates such that each receives a unit flow from  $\ell$ . Therefore  $|S'_\ell| = t$  and  $u_{S'_\ell,\ell}$  is a node in  $G(b)$ . When defining  $f'$ , after identifying the set  $S'_\ell$ , the algorithm simply reroutes the  $t$  flow units to the the candidates in  $S'_\ell$  through  $u_{S'_\ell,\ell}$ , thus the flow value is maintained.  $\square$

**Lemma 4.** *It holds that  $\mathbf{A}(f') \leq t \cdot \mathbf{A}(f)$ .*

*Proof.* Fix a voter  $\ell$ , and let  $R = \{S \mid f(\ell, u_{S,\ell}) \geq 1\}$ . Now consider the candidate set  $C' = \bigcup_{S \in R} S$  and the set  $S_{\max} = \arg \max_{S \in R} \bar{\pi}_\ell(S)$ . Let  $c'$  be the candidate  $c \in C'$  maximizing  $v_\ell(c)$  and notice that  $c' \in S_{\max}$ . Now consider  $S'_\ell$  as defined by the algorithm and notice that  $S'_\ell \subseteq C'$  by the flow properties: a unit of flow which reaches a candidate  $c$  from  $\ell$  must pass through some node  $u_{S,\ell}$  such that  $c \in S$ . Therefore—applying Lemma 1— $\bar{\pi}_\ell(S') \leq \bar{\pi}_\ell(S_{\max})$ .

It follows that  $a(\ell, u_{S'_\ell,\ell}) f'(\ell, u_{S'_\ell,\ell}) = t \cdot a(\ell, u_{S'_\ell,\ell}) \leq t \cdot a(\ell, u_{S_{\max},\ell}) \leq t \cdot \sum_{S \in R} a(\ell, u_{S,\ell}) f(\ell, u_{S,\ell})$ . The lemma follows by summing the last inequality over all voters  $\ell \in N$ , and recalling that for each edge  $e \notin E_2$ ,  $a(e) = 0$ .  $\square$

**Theorem 4.** *Algorithm  $\mathcal{M}$  returns a valid solution  $Q$  making  $p$  win, and  $\pi(Q) \leq t\pi(Q^*)$ .*

*Proof.* It is enough to show that in the iteration where  $b = b^*$ , we find a solution  $Q'$  such that  $\pi(Q') \leq t\pi(Q^*)$ .

As proven in Lemma 2, when  $b = b^*$  there exists a maximal flow  $tn$  in  $G(b^*)$  and therefore our algorithm will find such a flow. Consider  $f$  and  $f'$ , the min-cost flow and the constructed flow in this iteration. Also consider  $f^*$ , the flow induced by the optimal strategy as detailed in Lemma 2.

It holds that  $|f'| = tn$  and  $\mathbf{A}(f') \leq t \cdot \mathbf{A}(f) \leq t \cdot \mathbf{A}(f^*)$ . Since  $|f'| = tn$ , then  $f'(p, d) = b$ . Since  $f'(c, r) \leq b$  for all  $c \in C \setminus \{p\}$ ,  $p$  is necessarily winning.

For the flow  $f'$ , it holds that

$$\begin{aligned} \mathbf{A}(f') &= \sum_{\ell \in N} a(\ell, u_{S'_\ell,\ell}) f'(\ell, u_{S'_\ell,\ell}) \\ &= t \cdot \sum_{\ell \in N} a(\ell, u_{S'_\ell,\ell}) \\ &= t \cdot \sum_{\ell \in N} \bar{\pi}_\ell(S'_\ell) \\ &= t \cdot \pi(Q'). \end{aligned}$$

We obtain that  $\pi(Q') = \mathbf{A}(f')/t \leq \mathbf{A}(f^*) = t \cdot \pi(Q^*)$ , where the inequality follows from Lemma 4 and the final equality from Lemma 2.  $\square$

We make two important remarks: (a) our algorithm can be easily extended to support prices that are a function of the voters (by re-defining  $a(\ell, u_{S,\ell})$  to be also multiplied by the voter's price); and (b) our algorithm provides an exact solution in the specific case of Plurality (as  $t = 1$ ).

## Borda

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

### NP-Hardness

**Theorem 5.** *Given a value  $k$ , determining whether there exists a solution to Borda-TNC with at most  $k$  demotions is NP-hard.*

*Proof.* Given an instance  $I = (U, \mathcal{S})$  of SET COVER and a desired cover size  $k \leq \bar{m}$ , we define a reduction as follows.

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

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

$$v_i^1 = \overrightarrow{D_{\geq |S_i|}} \succ \overrightarrow{S_i} \succ a \succ b \succ \overrightarrow{U \setminus S_i} \succ \overrightarrow{D_{< |S_i|}} \succ p \quad (1)$$

$$v_i^2 = p \succ \overleftarrow{D_{< |S_i|}} \succ \overleftarrow{U \setminus S_i} \succ b \succ a \succ \overleftarrow{S_i} \succ \overleftarrow{D_{\geq |S_i|}}. \quad (2)$$

In addition, we define the following two preference orders:

$$\bar{v} = \overrightarrow{D} \succ a \succ b \succ \overrightarrow{U} \succ p \quad (3)$$

$$\hat{v} = p \succ \overleftarrow{U} \succ a \succ b \succ \overleftarrow{D}. \quad (4)$$

Finally, we define the following two preference orders:

$$\bar{v}' = \overrightarrow{D} \succ a \succ \overrightarrow{U} \succ b \succ p \quad (5)$$

$$\hat{v}' = p \succ \overleftarrow{U} \succ a \succ b \succ \overleftarrow{D}. \quad (6)$$

We let

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

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

**Lemma 5.** *For  $E$ , it holds that  $\text{diff}(a, p) = k(\bar{n} + 3)$ , that  $\text{diff}(p, b) = k(\bar{n} + 3) + \bar{n}$ , and that  $\text{diff}(p, d) = 0$  for every  $d \in D$ . In addition,  $\text{diff}(u_1, p) = \dots = \text{diff}(u_{\bar{n}}, p) = 1$ .*

By summing the points awarded to each candidate. To see that more easily, notice that each pair of preference orders defined above awards  $m - 1$  points to each candidate, unless the following event occurs: whenever  $c \succ c'$  in *both* of the pair's preference orders, it effectively means that a point is 'transferred' from  $c'$  to  $c$ .

**Lemma 6.** *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 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 voters of types  $v_i^1, \bar{v}, \bar{v}'$ ,  $a$  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,  $s(a) = s(p) = \sigma(p) + k$ .  $\square$

<sup>2</sup>While  $V$  is a list of preference orders, with a slight abuse of notation, we have defined it using set operations.

We are now ready to complete the proof of Theorem 5 by showing that  $E$  has a winning strategy with  $k$  operations if and only if  $U$  can be covered by  $k$  subsets of  $\mathcal{S}$ .

*Completeness:* 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  $v_i^1$ . 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  $u$  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.

*Soundness:* Assume that  $p$  can be made to win by bribing  $k$  voters, and consider the corresponding demotion strategy. By Lemma 6, 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.  $\square$

## 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  $\delta$  positions, then  $\delta$  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  $\delta$  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  $\text{BTNCRV}$  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  $\text{BTNCRV}$  is illustrated as Algorithm 1.

---

**Algorithm 1:** BTNCRV( $((C, \bar{V}), k, T)$ )

---

```
1  $Q \leftarrow \emptyset$ 
2 foreach  $c \in C$  do
3    $S_c \leftarrow \{(\ell, \delta) \mid (c, \delta) \in \bar{v}_\ell, \ell = 1, \dots, n\}$ 
4    $s_c \leftarrow \sigma(c)$ 
5 for  $i \leftarrow 1$  to  $k$  do
6    $C' \leftarrow \{c \in C \mid s_c > T\}$ 
7   if  $C' = \emptyset$  then
8     break
9   Pick an arbitrary  $c \in C'$ .
10   $(\ell', \delta') \leftarrow \arg \max_{(\ell, \delta) \in S_c} \delta$ 
11   $Q \leftarrow Q \cup \{(\ell', c, \delta')\}$ 
12   $S_c \leftarrow S_c \setminus \{(\ell', \delta')\}$ 
13   $s_c \leftarrow s_c - \delta'$ 
14   $C' \leftarrow \{c \in C \mid s_c > T\}$ 
15 if  $C' = \emptyset$  then return  $Q$  else fail
```

---

---

**Algorithm 2:** TNC for Borda

---

```
1  $\bar{V} = (\bar{v}_\ell)_\ell$  where  $\bar{v}_\ell = \{(c, \alpha_{v_\ell(c)}) \mid c \in C \setminus \{p\}\}$ 
2 Let  $\bar{E} = (C \setminus \{p\}, \bar{V})$ 
3 Let  $k'$  be the minimum  $k$  for which
   BTNCRV( $\bar{E}, k, \sigma(p) + k$ ) does not fail
4  $Q \leftarrow \text{BTNCRV}(\bar{E}, k', \sigma(p) + k')$ 
5 foreach  $(\ell, c, \delta) \in Q$  do
6   Demote  $c$  in  $v_\ell$  by  $\delta$  positions.
7 while  $p$  is not winning do
8   Let  $\ell \in N, c \in C \setminus \{p\}$  such that  $c$  is ranked
   immediately before  $p$  in  $v_\ell$ .
9   Demote  $c$  in  $v_\ell$  by one position.
10 return the sequence of demotion operations
    performed.
```

---

After finding the smallest  $k$  for which  $\text{BTNCRV}(\bar{E}, k, \sigma(p) + k)$  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 2.

**Lemma 7.** *Algorithm 1 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.  $\square$

**Lemma 8.** *Let  $E$  be a preferential election under Borda, and let  $\bar{E}$  be its corresponding election under RV, as constructed in Line 2 of Algorithm 2. Then:*

- A sequence of operations  $Q$  for range voting can be applied on the Borda instance, only that in each operation  $(\ell, c, \delta) \in Q$ ,  $\delta$  now pertains to the number of positions  $c$  is demoted by in  $v_\ell$  (for RV  $\delta$  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  $(\ell, c, \delta)$  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  $(\ell, c', \delta')$ , this means that the score currently awarded to  $c'$  by  $\ell$  in the Borda instance is at least  $\delta'$ , 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  $(\ell, c, \delta')$  where  $\delta'$  is the score currently awarded to  $c$  by  $\ell$  in the RV instance (so that following the operation, the score awarded to  $c$  by  $\ell$  is 0).  $\square$

Let  $k^*$  be the optimal number of demotion operations required to make  $p$  win, and let  $k'$  be the value from Line 3 of Algorithm 2.

**Lemma 9.** *BTNCRV( $\bar{E}, k^*, \sigma(p) + k^*$ ) does not fail. In particular, this implies that  $k' \leq k^*$ .*

*Proof.* For TNC under Borda, if  $p$  can be made to win by at most  $k^*$  demotion operations, then  $p$ 's final score  $s(p)$  is at most  $\sigma(p) + k^*$ . In addition, following these operations, each other candidate score is at most  $s(p) \leq \sigma(p) + k^*$ . Let  $Q^*$  be the demotion strategy applied by an optimal strategy on  $E$ . Assume we apply the strategy  $f(Q^*)$  as defined by Lemma 8 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  $f(Q^*)$  is a valid solution for  $\text{BTNCRV}(\bar{E}, k^*, \sigma(p) + k^*)$ .  $\square$

**Lemma 10.** *The loop in Line 7 of Algorithm 2 will run at most  $2k'$  times.*

*Proof.* Let  $s'(c)$  be the score of a candidate  $c$  right before Line 7 of Algorithm 2. The demotion strategy  $Q$  found in Algorithm 2 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 8), 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.  $\square$

**Theorem 6.** *Algorithm 2 is a 3-approximation algorithm for TNC under Borda.*

*Proof.* As  $k' \leq k^*$  (by Lemma 9), it is sufficient to show that Algorithm 2 performs at most  $3k'$  operations. To see that, notice that  $\pi(Q) \leq k'$  and that the loop of Line 7 of Algorithm 2 runs at most  $2k'$  times (by Lemma 10), where each iteration involves a single demotion operation.  $\square$

## 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 (?).

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

**Theorem 7.** *With the price function  $\hat{\pi}: N \times C \rightarrow \mathbb{R}_0^+$ , 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.* Let  $f$  be the reduction from SET COVER described in the proof of Theorem 5, adjusted such that in  $f(I, k)$ , for each voter  $\ell$  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  $\ell$  with vote in  $V \setminus \{v_i^1\}_{i=1}^{\bar{m}}$ ,  $\pi(\ell, c) = k \ln(m/2 - 1)$  for every  $c \in C$ .

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 we know what is the optimal set cover size  $k$ . In this case, we know that a strategy of price  $k$  exists, by an argument very similar to the completeness argument in Theorem 5.

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  $\text{diff}(p, c)$  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  $\text{diff}(p, u) = -1$ , but now  $\text{diff}(p, u) \geq 0$ , we bribed at least one voter having a vote of type  $v_i^1$  such that  $u \in S_i$ , which allowed  $\text{diff}(p, u)$  to

increase by a point. Therefore, the collection  $S'' = \{S_i \in \mathcal{S} \mid v_i^1 \text{ is bribed in } \mathcal{A}(f(I, k))\}$  constitutes a valid cover and  $|S''| \leq (1 - \epsilon)k \ln(m/2 - 1) = (1 - \epsilon)k \ln \bar{n}$ .

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  $k = 1, \dots, \bar{n}$ , 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. 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 (?).  $\square$

## Destructive Variants

For the destructive variant DTNC our goal it to make the currently leading candidate  $d \in C$  lose (by making sure he will not be in the winner set). To this end we introduce a polynomial algorithm for scoring rules, applicable even when voters have prices  $\pi(\ell)$  for each voter  $\ell$ . We assume that each score value of the scoring rule is representable as an  $O(\log(nm))$ -bit integer (notice that the ‘natural’ input size is  $\Theta(nm)$ ). This is a very natural assumption: 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 two score values in  $\alpha$  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 (?), for which  $\alpha = (2^{m-1}, 2^{m-2}, \dots, 2^1, 2^0)$ .

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  $Q^*$  be the optimal strategy (unknown to us) making  $c$  beat  $d$ . We can split  $Q^*$  to two sets of operations,  $A$  and  $B$ , such that  $A$  are all operations demoting  $d$  and  $B$  are all operation demoting a candidate other than  $d$ .

**Lemma 11.** *W.l.o.g., all the following hold for  $Q^*$ :*

1. *All operations in  $A$  were performed before all operations in  $B$ .*
2. *All operations in  $A$  demoted  $d$  to be ranked last.*
3. *All operations in  $B$  involved demoting a candidate ranked immediately before  $c$  to be immediately after  $c$ .*

*Let  $\ell$  be a voter bribed during the execution of  $B$ , let  $C'$  be the candidates demoted by  $\ell$  within  $B$ , and let  $t$  be the point in time following the operations in  $A$  and before those of  $B$ .*

4. *At time  $t$ ,  $c \succ_\ell d$ .*
5. *At time  $t$ , the candidates in  $C'$  were ranked consecutively immediately before  $c$ .*



*Proof.* The first three claims are straightforward. For the fourth claim, assume that following the operations in  $A$  it holds that  $d \succ_\ell c$ .  $\ell$  was not bribed in  $A$  because  $d$  is above  $c$ . Let  $c'$  be the voter demoted in this operation. Then we can demote  $d$  instead of  $c'$  and have an even greater effect on  $\text{diff}(d, c)$ . However, in that case this operation can be placed as part of  $A$ . The fifth claim follows from the fact that if we demote  $j$  candidates ranked before  $c$ , the choice of which candidates is unimportant; in any case  $c$  will gain  $\alpha_{v_\ell(c)-j} - \alpha_{v_\ell(c)}$  points.  $\square$

Let  $s$  be the change in  $\text{diff}(d, c)$  as a result of applying  $A$ , and notice that  $0 \leq s \leq \text{diff}(d, c) + \alpha_1$  (in the edge-case where  $B$  is empty, it is possible that the final operation in  $A$  makes the change exceed  $\text{diff}(d, c)$ ). We do not know what  $A$  is, but we can exhaustively try all possibly values for  $s$ . Given the correct guess for  $s$ , we can then find  $A$  using the following reduction to the 0-1-Knapsack problem.

For each voter  $\ell$  we create an item  $\ell$  with a weight  $w(\ell) = \pi(\ell)$  and a value  $v(\ell) = (\alpha_{v_\ell(d)} - \alpha_m) + 1[d \succ_\ell c] \cdot (\alpha_{v_\ell(c)-1} - \alpha_{v_\ell(c)})$ . This value represents the demotion of  $d$  to be last, such that  $d$  loses  $(\alpha_{v_\ell(d)} - \alpha_m)$  points and  $c$  possibly gains  $(\alpha_{v_\ell(c)-1} - \alpha_{v_\ell(c)})$  points (if  $d$  was previously above him).

Our goal is to find the minimal total weight of items needed in order to obtain a value of at least  $s$ . Fortunately, Knapsack has an algorithm which is pseudo-polynomial in the values (?),<sup>3</sup> as our values can be represented as integers bounded by a polynomial in the input size, this Knapsack instance is polynomial-time solvable.

Once we have found  $A$  for a guess of  $s$ , before we continue to finding  $B$ , we will require another version of the Knapsack problem: let the sets  $X_1, \dots, X_n$  be a partition of a set of items, where each  $X_\ell = \{x_{\ell,1}, \dots, x_{\ell,|X_\ell|}\}$ . Each such item has a weight  $w(x_{\ell,j}) \geq 0$  and a value  $v(x_{\ell,j}) \geq 0$ . Given a positive value  $L$  the goal is to construct 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  $|S \cap X_\ell| \in \{0, 1\}$  for each  $\ell$ . This problem is a specific case of GROUP FAIRNESS KNAPSACK (GFK), studied by ?. For completeness, we provide an algorithm for our case in the following.

**Lemma 12.** *GFK can be solved in time polynomial in the input size and pseudo-polynomial in  $L$ .*

*Proof.* Let  $f(k, i)$  represent the minimal weight needed in order to have a value of at least  $k$  when only choosing from  $X_1, \dots, X_i$ . We can compute  $f(L, n)$  (our objective) with dynamic programming using the following recursion:  $f(k, i) = \min_{0 \leq j \leq |X_i|} (f(k - v(x_{i,j}), i - 1) + w(x_{i,j}))$  where  $x_{i,0}$  is a placeholder item having zero weight and value representing not taking any item from the set. The edge cases are as follows:  $f(0, 0) = 0$ ; for every  $k > 0$ ,  $f(k, 0) = \infty$ ; and for each  $k < 0$ ,  $i \in [n]$ ,  $f(k, i) = 0$ .  $\square$

We can now find  $B$  (or an equivalent sequence of operations) by a reduction to GFK: For every voter  $\ell$  create a set

$X_\ell$  containing  $v_\ell(c) - 1$  items  $x_{\ell,1}, \dots, x_{\ell,v_\ell(c)-1}$ , where for every  $j \in [v_\ell(c) - 1]$  we define  $w(x_{\ell,j}) = j \cdot \pi(\ell)$  and  $v(x_{\ell,j}) = \alpha_{v_\ell(c)-j} - \alpha_{v_\ell(c)}$ ; the item  $x_{\ell,j}$  represents the option of making  $c$  gain  $\alpha_{v_\ell(c)-j} - \alpha_{v_\ell(c)}$  points by demoting the  $j$  candidates which are currently consecutively ranked immediately before  $c$ . We choose  $L = \text{diff}(d, c) - s + 1$  (i.e.,  $L$  is the difference between  $d$  and  $c$  after the operations in  $A$  were executed, plus an additional point to make sure that  $d$  is not in the winner set).

**Theorem 8.** *For every scoring rule, in which each score can be represented as an  $O(\log(nm))$ -bit integer, the DTNC problem with prices which are a function of the voters can be solved in polynomial time.*

*Proof.* Follows from the above discussion.  $\square$

## Conclusions

The contribution of this work is twofold: first, we studied the de facto standard of political campaigns: targeted negative campaigning. While being so widespread—as far as we know—it was not studied computationally. Second, our results show that it is a sweet-spot between SHIFT BRIBERY—which models positive campaigns—and SWAP BRIBERY which we feel is too granular: it models bribery and campaigning in a ‘local’, swap-oriented level, instead of the more global effect of our demotion operations. 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 mention some directions for future research: (a) to better understand the complexity of  $t$ -approval-TNC when  $t = 2$ , or when  $t$  is not fixed, and of Borda with a price of the form  $\pi: N \rightarrow \mathbb{R}_0^+$ ; (b) to research other voting rules that are not necessary scoring rules, such as Copeland and Maximin.

Accusamus iste odio eos sed dolores voluptatibus pos-simus saepe, molestiae officiis expedita ipsa incidunt esse sit corrupti sed reprehenderit pariat, expedita facere ut adipisci reiciendis odio, et cupiditate aliquid fugiat, id ut velit. Mollitia placeat accusamus sunt beatae earum veniam preferendis, dolorum neque modi voluptatibus dolore molestiae quae doloremque, est ipsum ea sed quibusdam vel voluptatibus eveniet qui, libero cum rem nam velit inventore deleniti ipsa voluptate earum mollitia asperiores, neque laborum libero ratione earum quos consectetur labore distinctio quibusdam. Dolore numquam possimus sit perspiciatis vel voluptate quaerat labore et eaque pariat, asperiores obcaecati consectetur, repellendus recusandae quas aliquam nostrum nam explicabo, sint incidunt voluptatum quos eius nihil maxime assumenda sed, vitae enim vero consequuntur eligendi aliquid expedita mollitia? Eos recusandae fugiat quis architecto eius, aperiam voluptatum distinctio reprehenderit atque fugiat, nam delectus preferendis vel eveniet placeat libero, similique laudantium quas illum aut rem sed, qui eligendi placeat eos totam sit quae voluptatum? In tempora iste dolorem fugiat, aliquam maxime odit in doloribus perspiciatis quis ipsam, doloremque perspiciatis esse est fugiat sint inventore explicabo? Tenetur beatae sit nisi, eos ve-

<sup>3</sup>There is also an algorithm pseudo-polynomial in the weights.

niam obcaecati, accusantium accusamus fuga quas vel, a  
maxime inventore deleniti nisi modi eos?