

# Controlling Delegations in Liquid Democracy

AAAI Track

Shiri Alouf-Heffetz  
Ben-Gurion University  
Beer Sheva, Israel  
shirihe@post.bgu.ac.il

Tanmay Inamdar  
Indian Institute Of Technology  
Jodhpur  
Jodhpur, India  
taninamdar@gmail.com

Pallavi Jain  
Indian Institute Of Technology  
Jodhpur  
Jodhpur, India  
pallavijain.t.cms@gmail.com

Nimrod Talmon  
Ben-Gurion University  
Beer Sheva, Israel  
talmonn@bgu.ac.il

Yash More Hiren  
Indian Institute Of Technology  
Gandhinagar  
Gandhinagar, India  
yash.mh@iitgn.ac.in

## ABSTRACT

In liquid democracy, agents can either vote directly or delegate their vote to a different agent of their choice. This results in a power structure in which certain agents possess more voting weight than others. As a result, it opens up certain possibilities of vote manipulation, including control and bribery, that do not exist in standard voting scenarios of direct democracy. Here we formalize a certain kind of election control – in which an external agent may change certain delegation arcs – and study the computational complexity of the corresponding combinatorial problem.

## KEYWORDS

Computational social choice; proxy voting; liquid democracy; manipulation; computational complexity; parameterized complexity

### ACM Reference Format:

Shiri Alouf-Heffetz, Tanmay Inamdar, Pallavi Jain, Nimrod Talmon, and Yash More Hiren. 2024. Controlling Delegations in Liquid Democracy: AAAI Track. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

## 1 INTRODUCTION

Liquid democracy is an innovative approach to democratic governance that can be thought of as a middle-point between direct and representative democracy. It has attracted significant attention both from practitioners and from academics, mainly as it offers greater flexibility to voters: essentially, voters participating in liquid democracy can not only choose how to fill their ballot, but they can choose instead of filling their ballot to delegate their vote to another voter of their choice.

More elaborately, unlike traditional models where agents either directly participate in the decision-making process by explicitly

casting their vote – as in direct democracy – or elect representatives to make decisions on their behalf – as in representative democracy, liquid democracy offers a hybrid system that allows for both direct and indirect participation. At its core, liquid democracy enables individuals to delegate their voting power to trusted proxies while retaining the option to vote directly on specific issues. This fluidity of participation holds the potential to enhance democratic engagement, facilitate more informed decision-making, and address the limitations of existing democratic systems.

One of the key advantages of liquid democracy indeed lies in the added flexibility it provides to voters. By allowing individuals to delegate their voting power to trusted proxies, liquid democracy empowers citizens to actively participate in decision-making even when they are unable to devote extensive time and effort to every issue. This flexibility enables individuals to delegate their votes to experts or representatives they trust on specific subjects, while still retaining the ability to directly vote on matters that are of particular importance to them. Importantly, this has the potential of increasing the quality of the decision making process.

Like any democratic system, however, liquid democracy is not immune to potential challenges, including the risk of voter manipulation. Given the fluid nature of delegation, there is the natural concern that influential or malicious actors could exert undue influence by strategically delegating votes or exploiting the trust placed in them as proxies. Such manipulation could undermine the principles of fairness and equality that are fundamental to the democratic process. The issue of the vulnerability of liquid democracy to certain different forms of manipulation has yet to receive sufficient attention from the research community.

Here we consider several forms of manipulation in the context of liquid democracy. In particular, we concentrate on the computational complexity of successfully conducting these manipulations. In particular, we are interested in the situation in which some external agent tries to rig the result of the election by redirecting few delegation arcs.

We define a corresponding combinatorial problem and study its computational complexity. We observe cases that allow for polynomial-time algorithms, cases that allow for XP algorithms, parameterized (approximation) algorithms, as well as cases for which

Authors are ordered in the alphabetical order of their last names.



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

*Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).*

no efficient algorithms exist (assuming certain complexity theoretic hypothesis, such as  $P \neq NP$ ).

We wish to stress that liquid democracy is currently utilized in practical settings for decisions of varying significance, from political realms like the German Pirate Party to financial contexts in various blockchain ventures. Its appeal lies both in its potential to enhance democratic participation as well as its promise to improve the quality of the decision making itself. Thus, we argue that, recognizing the limitations and vulnerability of liquid democracy to external forms of influence – such as the ones we study in the current paper – is essential for a better understanding and usage of liquid democracy.

## 2 RELATED WORK

By now, liquid democracy has attracted significant attention from the research community. In particular, it is studied from a political science perspective [4, 20]; from an algorithmic point of view [7, 10, 19]; from a game-theoretic point of view [3, 12, 22]; and from an agenda aiming at pushing its boundaries by increasing the expressiveness power that is granted to voters [6, 18].

Here we study election control in liquid democracy; in this context, we mention the extensive survey on control and bribery [14]. Indeed, due to their importance, the problems of control and bribery are studied thoroughly [2, 5, 21].

To the best of our knowledge, however, the topic of election control and bribery in the context of liquid democracy was not yet studied. We do mention the work of [22] and its follow up paper [9], in which agents may behave strategically. Note, however, that we are interested not in vote manipulation (by the community members themselves) but in election control (by an external agent). Thus, from this point of view, our work extends the vast literature on control and bribery – that is currently applied to settings such as single-winner elections and multiwinner elections – to the setting of liquid democracy. And, as the most distinguishing feature of liquid democracy is its usage of vote delegations, we study the natural control action of altering the delegations themselves.

Another related line of research focuses on opinion diffusion in social networks. Following vote delegation chains in liquid democracy mirrors the process of tracking preferences in opinion diffusion. Notably, [1, 13] provided insights into opinion propagation using a voting rule similar to our approach for resolving multi-delegations.

Finally, note that the type of liquid democracy we consider here is not only the standard type in which a voter may delegate their vote to a single other voter but also a version in which a voter may delegate to several other voters, in which case the votes of this set of voters is aggregated to compute the vote of the delegating voter. In this context, we are closer to more advanced types of liquid democracy [8, 17].

## 3 CONTROL BY REDIRECTING ARCS

Next, we first position our work within the landscape of control problems for liquid democracy; and then describe our formal model.

### 3.1 Liquid Democracy and Election Control

Generally speaking, when considering election control in liquid democracy, there are several orthogonal factors to take into account:

- The ballot type: e.g., whether ballots are binary, plurality, approval, ordinal, cumulative.
- The delegations type: e.g., whether delegations are transitive, coarse-grained or fine-grained [6, 18], whether several delegations are possible for a single agent.
- The actions at the disposal of the controlling agent: e.g., whether the controlling agent can add edges, remove edges, redirect edges, change the ballots of some voters.
- The goal of the controlling agent: e.g., whether the goal is to make some predefined preferred candidate win the election.

Following the taxonomy above, it is worthwhile to note that here we consider: approval ballots; transitive coarse-grained ballots with possibly more than one delegate to each agent; a controlling agent that can only redirect edges and whose goal is to make some predefined preferred candidate a winner of the election after the controlling actions and the unraveling of the (partially modified) delegations.

### 3.2 Formal Model

We describe our formal model. Our formal model contains the following ingredients:

- A set of voters  $V = \{v_1, \dots, v_n\}$ .
- A *delegation graph*  $G = (V, E)$ , which is a directed graph where the voters are the vertices. If the delegation graph has an out-degree of at most one, then we sometimes refer to the situation as a “single-delegation” setting; while, in general, it is a “multi-delegation” setting. A voter  $v_i$  with out-degree 0 is said to be an *active voter*, while a voter  $v_i$  with out-degree strictly positive is said to be a *passive voter*.
- A cost function  $cost: E(G) \rightarrow \mathbb{N}$ . The cost of an arc in the delegation graph  $G$  is the cost of redirecting this arc.
- We consider approval elections, so there is a (global) set of candidates  $C = \{c_1, \dots, c_m\}$  and each active voter  $v_i$  corresponds to a ballot  $b_i \subseteq C$ .
- An *unraveling function*  $\mathcal{R}$  that takes several ballots and returns a ballot; it is used in the following way: the ballot of a passive voter delegation to active voters with ballots  $b_1, \dots, b_z$  is unravelled to the ballot  $\mathcal{R}(b_1, \dots, b_z)$ , as explained next.) Note that, throughout the paper we consider several rules  $\mathcal{R}$  as the unravelling function as described below. In particular, we consider the following rules:
  - **The union function:** here,  $\mathcal{R}$  returns the union of the ballots it gets as input:  $\mathcal{R}(b_1, \dots, b_z) = \cup_{z \in [z]} b_z$ .
  - **Approval function:** an approval function takes a ballot and returns the set of candidates approved by the maximum number of voters.
  - **GreedyMRC function:** Under GreedyMRC, we return a set of candidates obtained by the following method: we start with an empty set and perform a sequence of iterations, where in each iteration we (a) add to the current set, a set of candidates  $S$  approved by the largest number of voters (all the candidates in  $S$  are approved by the same number of voters) and (b) remove the voters that approve candidates in  $S$  from further consideration.

Note that the choice of unraveling function is only relevant for multiple delegation.

- Given a delegation graph  $G$  and an unraveling function  $\mathcal{R}$ , we can define the *unraveled vote* of each passive voter by following its delegations transitively; in particular, a passive voter  $v_i$  with out-arcs to  $u_1, u_2, u_3$  – say, all of which are active voters – will be assigned the unraveled ballot  $b_i = \mathcal{R}(u_1, u_2, u_3)$ ; this happens transitively (i.e., at each level). For simplicity, assume that there are no cycles. From real-world applications of liquid democracy, in particular in the context of the LiquidFeedback platform, we know that usually the number of cycles is rather limited in practice. Let  $b_i$  be the ballot of  $v_i$  if  $v_i$  is an active voter and its unraveled vote if  $v_i$  is a passive voter.
- A *voting rule*  $\mathcal{W}$  that takes a set of (possibly some of which are unraveled) ballots and returns a candidate  $c^* \in C$  as the winner of the election. Note that, throughout the paper we consider  $\mathcal{W}$  as the **Approval voting rule**, which returns the name of the candidate with the highest number of votes; we discuss other options in the Outlook section towards the end of the paper.
- An external agent – referred to as *the controller* – who has a predefined budget of  $k$  and who can change a set of arcs, say  $S$ , whose total cost is at most  $k$  and whose goal is to make some predefined preferred candidate  $c^*$  a winner of the election where we first redirect the arcs in  $S$ , then unravel all delegations using  $\mathcal{R}$ , then apply  $\mathcal{W}$  on the (direct and unraveled) ballots, and then see whether  $c^*$  is in the winning committee  $w$ .

By redirecting an arc we mean taking an arc  $(u, v)$ , removing it from the graph, and inserting a different arc  $(u, v')$ .

So, formally, the problem we are considering in this paper can be defined as follows:

**CONSTRUCTIVE CONTROL BY REDIRECTING ARCS (CCRA)**

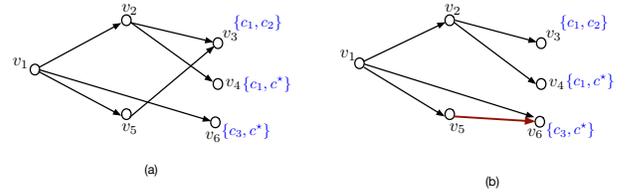
**Input:** A delegation graph  $G = (V, E)$  with costs function  $cost$  with approval ballots over a set of candidates  $C$ , an unraveling function  $\mathcal{R}$ , a voting rule  $\mathcal{W}$ , a preferred candidate  $c^*$ , and a budget  $k$ .

**Question:** Does there exist a delegation graph  $G'$  that is achieved from  $G$  by performing redirections within the budget  $k$ , such that, after unraveling all of the delegations of  $G'$  using  $\mathcal{R}$ ,  $c^*$  is the unique winner of the resulting election using  $\mathcal{W}$ ?

We denote the number of delegation as  $\#delegations$  and the number of approvals as  $\#approvals$ .

The following example illustrates our model.

**Example 1.** Consider the delegation graph in Figure 1 on the voter set  $\{v_1, v_2, v_3, v_4, v_5, v_6\}$  and the candidate set  $\{c_1, c_2, c_3, c^*\}$ , where  $c^*$  is the preferred candidate. Let the cost of every arc be 1. Consider union function as the unraveling function  $\mathcal{R}$  and approval voting as the voting rule  $\mathcal{W}$ . In Figure 1(a), since  $v_2$  delegates to  $v_3$  and  $v_4$ , the vote of  $v_2$  is  $\{c_1, c_2, c^*\}$ . Similarly, the vote of  $v_5$  is  $\{c_1, c_2\}$  and the vote of  $v_1$  is  $\{c_1, c_2, c_3, c^*\}$ . Since  $c_1$  is approved by the maximum number of voters,  $c_1$  is the winner in the delegation graph in Figure 1(a). We



**Figure 1: (a) Input delegation graph, (b) Delegation graph after redirecting arc  $(v_5, v_3)$  to  $(v_5, v_6)$**

redirect the arc  $(v_5, v_3)$  to  $(v_5, v_6)$  as depicted in Figure 1(b). Now,  $c^*$  is approved by 5 voters (all but  $v_3$  approves  $c^*$ ) and no other candidate is approved by 5 voters. Thus,  $c^*$  is the unique winner in the delegation graph in Figure 1(b).

Table 1 and Table 2 summarise the classical and parameterized complexity of our problem, respectively.

**REMARK 1.** Note that  $\mathcal{W}$  can be any single-winner voting rule operating on approval ballots. Similarly,  $\mathcal{R}$  can be any committee selection rule with variable number of winners (usually referred to as a VNW rule) [15]. As such, it means that our model is general enough to apply any VNW rule to its unravelling procedure and any single-winner voting rule as its winner selection procedure.

**REMARK 2.** In many works on liquid democracy there is an underlying social graph connecting the voters and voters can only delegate to their corresponding friends. Note that, in our setting, a voter can delegate to any other voter; so, put differently, we study the special case in which the underlying social graph is a clique. Since we study a special case, our hardness results hold also for the general case with an arbitrary underlying social graph.

All the missing proofs are in supplementary. We assume the familiarity with the parameterized and approximation algorithms. Their definitions are discussed in the supplementary.

## 4 HARDNESS RESULTS

We begin with our results regarding the computational hardness of the CCRA problem. Both our NP-hardness results are due to the polynomial-time reduction from the VERTEX COVER problem in cubic graphs and the W[2]-hardness results are due to the same reduction from the HITTING SET problem.

In the VERTEX COVER problem in cubic graphs, given a cubic graph  $G = (V, E)$  (the degree of every vertex is three), and an integer  $\tilde{k}$ , the goal is to find a set  $S \subseteq V(G)$  of size at most  $\tilde{k}$  such that at least one of the endpoint of every edge is in  $S$ . The problem is known to be NP-hard [16]. In the HITTING SET problem, given a universe  $U$ , a family,  $\mathcal{F}$ , of subsets of  $U$ , and an integer  $\tilde{k}$ , the goal is to find a set  $S \subseteq U$  such that for every set  $F \in \mathcal{F}$ ,  $S \cap F \neq \emptyset$ . Note that HITTING SET is a generalisation of the VERTEX COVER problem, and it is known to be W[2]-hard with respect to the parameter  $\tilde{k}$  [11].

We begin with our first hardness result.

**Theorem 1.** CCRA is NP-hard when  $\mathcal{R}$  is union function or approval function or GreedyMRC function even when

- (1) every vertex in the delegation graph satisfies one of the following conditions:

**Table 1: Computational Complexity of CCRA**

#delegations=1		#delegations $\geq 1$	
#approvals = 1	#approvals $\geq 1$	#approvals = 1	#approvals $\geq 1$
P(Theorem 3)	NP-complete(Theorem 2)	NP-complete(Theorem 1)	NP-complete(Theorem 1)

**Table 2: Parameterized Complexity of CCRA. Here,  $\delta^+(G)$  is the maximum out-degree of a vertex in  $G$ ,  $\delta^-(G)$  is the maximum in-degree of a vertex in  $G$ , and  $\lambda(G)$  is the maximum length of a path in the delegation graph  $G$ . ? denote that the complexity is open.**

Parameters	#delegations=1	#delegations $\geq 1$	
	#approvals $\geq 1$	#approvals = 1	#approvals $\geq 1$
#voters ( $n$ )	FPT(Theorem 4)	FPT(Theorem 4)	FPT(Theorem 4)
#active voters ( $t$ )	XP(Theorem 5), FPT-AS <sup>a</sup> (Theorem 6)	?	?
#candidates ( $m$ )	XP, FPT-AS <sup>b</sup> (Corollary 1)	?	?
$\delta^+(G) + \delta^-(G) + \#\text{approvals}$	paraNP-hard(Theorem 2)	paraNP-hard(Theorem 1)	paraNP-hard(Theorem 1)
$\delta^+(G) + \lambda(G) + \#\text{approvals}$	paraNP-hard(Theorem 2)	paraNP-hard(Theorem 1)	paraNP-hard(Theorem 1)
$k + \#\text{redirections}$	W[2]-hard <sup>c</sup> (Theorem 2)	W[2]-hard <sup>d</sup> (Theorem 1)	W[2]-hard <sup>e</sup> (Theorem 1)

<sup>a</sup> The parameter is  $t + \epsilon$ .

<sup>b</sup> The parameter is  $m + \epsilon$ .

<sup>a,b</sup> For both the results, we assume that there is an active voter who either approves only  $c^*$  or any subset of candidates that excludes  $c^*$ .

<sup>d</sup> The result holds even when  $\delta^+(G) + \delta^-(G) + \#\text{approvals}$  is constant or  $\lambda(G) + \#\text{approvals}$  is constant.

<sup>e</sup> The result holds even when  $\delta^-(G) + \#\text{approvals}$  is constant or  $\lambda(G) + \#\text{approvals}$  is constant.

- (a) *out-degree is at most three, the maximum length of a path in the delegation graph is at most two, and the cost of arcs belong to  $\{1, 2\}$*
- (b) *out-degree is at most two and in-degree is at most two*
- (2) *size of approval set is one*
- (3) *only one voter approves  $c^*$*

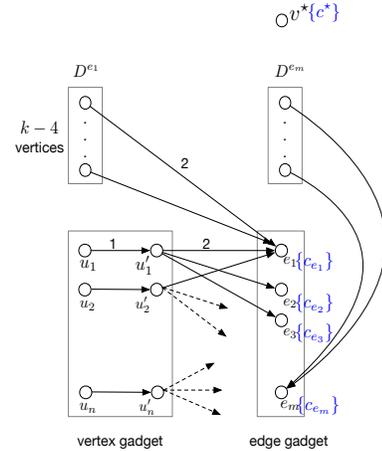
Furthermore, it is W[2]-hard with respect to  $k + \#\text{redirections}$  with all the above constraints except that in 1(a) out-degree is not bounded by a constant.

**PROOF.** We focus the proof for the union function. The proof is exactly the same for other two unraveling functions as well since in our gadget the set of candidates obtained by all these unraveling functions is same.

We give a polynomial time reduction from the VERTEX COVER problem in cubic graphs. Let  $(G, \tilde{k})$  be an instance of VERTEX COVER such that  $G$  is a cubic graph. We construct an instance of CCRA that satisfies constraints 1(a), 2, and 3 in the theorem statement. We will discuss later the required modifications for constraint 1(b). The construction is as follows.

- For every edge  $e \in E(G)$ , we have a candidate  $c_e$ . Additionally, we have a candidate  $c^*$  (our preferred candidate).
- For every vertex  $v \in V(G)$ , we have two passive voters  $v$  and  $v'$ , and  $v$  delegates to  $v'$ . The cost of this arc is 1.
- For every edge  $e \in E(G)$ , we have an active voter  $e$  who approves the candidate  $c_e$ . If  $u$  is an endpoint of  $e$ , then  $u'$  delegates to  $e$  and the cost of this arc is 2.
- We add a special voter  $v^*$  who approves the candidate  $c^*$ .
- For every edge  $e \in E(G)$ , we add a set of dummy voters  $D^e = \{d_1^e, \dots, d_{k-5}^e\}$ . Each dummy voter in  $D^e$  delegates to the active voter  $e$  and the cost of this arc is 2.
- The budget  $k$  is  $\tilde{k}$ .

Let  $H$  be the delegation graph that we constructed. Note that in this graph the score of  $c^*$  is 0 and  $k$  for every other candidate. Figure 2 illustrates the construction.



**Figure 2: Illustration of NP-hardness claimed in Theorem 1 with constraints 1(a), 2, and 3. The dotted arcs are to illustrate that the out-degree of every  $u'$  is three. Here,  $e_1 = u_1u_2$  is an edge in  $G$ . An approved candidate by an active voter is written in the blue color next to the voter name. The number on the top of an edge is the cost of redirection.**

Next, we prove the correctness. In the forward direction, suppose that  $S$  is a solution to  $(G, k)$ . Without loss of generality, we assume that  $|S| = k$ . If  $v \in S$ , then we redirect the arc  $(v, v')$  to  $(v, v^*)$ . Since the cost of arc  $(v, v')$  is 1 and  $|S| = k$ , the total cost of redirection is  $k$ . Now, we prove that  $c^*$  is the unique winner in the delegation graph  $H^*$  obtained after redirections. For every vertex  $v \in S$ , we

have an arc  $(v, v^*)$  in  $H^*$ . Furthermore, since  $v$  is the source vertex (vertex with in-degree zero) in the delegation graph, the vote of  $c^*$  is  $k$ . For every edge  $e(=uv) \in E(G)$ ,  $u'$  and  $v'$  delegates to  $e$ . Since either  $u$  or  $v$  is in  $S$ , the vote of candidate  $c_e$  decreases by at least 1, hence, the score of  $c_e$  in  $H^*$  is at most  $k-1$  ( $k-5$  votes are due to dummy voters). Thus,  $c^*$  is the unique winner in  $H$ .

In the reverse direction, suppose that  $H^*$  is the delegation graph obtained after redirecting arcs within the budget such that  $c^*$  is the unique winner. Let  $S$  be the set of arcs that are redirected in  $H$  to obtain  $H^*$ . We can safely assume that all the arcs are redirected to  $v^*$  because if an arc is redirected to some other voter say  $v$ , then instead of  $v$ , redirecting it to  $v^*$  increases the score of  $c^*$  and decrease the score of other voters. We modify  $S$  a bit as follows. Suppose that an arc  $(d, e) \in S$ , and  $(u', e) \notin S$ , where  $u'$  is a voter corresponding to the vertex  $u \in V(G)$ , which is an endpoint of the edge  $e \in E(G)$ . Then, delete  $(d, e)$  from  $S$  and add  $(u', e)$  in  $S$ . Note that the cost of both the arcs is same, so there the total cost remain same. Redirection of arc  $(d, e)$  decreases the score of the candidate  $c_e$  by 1 and increase the score of  $c^*$  by 1. However, redirection of arc  $(u', e)$  decreases the score of the candidate  $c_e$  by at least 1 and increase the score of  $c^*$  by at least 1 ("at least" because we might have already redirected arc  $(u, e)$ ). Thus,  $S$  is still a solution. We can safely assume that if an arc  $(u', e)$  is redirected, then  $(u, u')$  is not redirected as it only contributes in the cost, but does not increase/decrease the score.

We construct a set  $S' \subseteq V(G)$  as follows: if an edge incident to  $u'$  ( $(u, u')$  or  $(u', e)$ ) is in  $S$ , add the vertex  $u \in V(G)$  in  $S'$ . Next, we prove that  $S'$  is a vertex cover of  $G$  of size at most  $k$ . Let  $\alpha, \beta, \gamma$  be the number of arcs in  $S$ , that are incident on dummy voters, arcs of type  $(u', e)$ , and arcs of type  $(u, u')$ . Thus, the cost of redirection is  $2\alpha + 2\beta + \gamma$  and the score of  $c^*$  is  $\alpha + 2\beta + \gamma$ .

**Claim 1.** *By redirecting arcs in  $S$ , the score of every candidate  $c_e$ , where  $e \in E(G)$ , reduces by at least 1.*

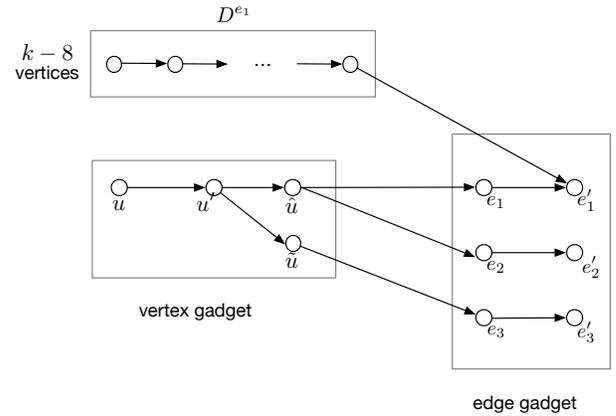
**PROOF.** Note that in the delegation graph  $H$ , the score of every candidate  $c_e$ , where  $e \in E(G)$ , is  $k$  and the score of  $c^*$  is 0. If there exists a candidate whose score is  $k$  in the delegation graph  $H^*$ , then the score of  $c^*$  is at least  $k+1$ . Thus,  $\alpha + 2\beta + \gamma > k$ . Hence,  $2\alpha + 2\beta + \gamma > k$ , a contradiction to the fact that the cost of redirection is at most  $k$ .  $\square$

Due to Claim 1 and the modified  $S$ , we know that for every voter  $e$ , either  $(u', e) \in S$  or  $(u, u') \in S$  (not both), where  $u$  is an endpoint of edge  $e \in E(G)$ . Thus, due to the construction of  $S'$ , for every edge  $e \in E(G)$ , at least one of its endpoint is in  $S'$ . We next claim that the size of  $S'$  is at most  $k$ . For every vertex  $u \in V(G)$ , either  $(u, u') \in S$  or  $(u', e)$ , not both. Thus,  $|S'| = \beta + \gamma$ . Since  $2\alpha + 2\beta + \gamma \leq k$ , it follows that  $\beta + \gamma \leq k$ . Hence,  $|S'| \leq k$ .

This completes the proof of Theorem 1 with constraints 1(a), 2, and 3.

Next, we mention the required modification to decrease the in-degree and out-degree of the delegation graph. Figure 3 illustrates these modifications. Now, every vertex  $u \in V(G)$ , we add four passive voters  $u, u', \hat{u}$ , and  $\tilde{u}$ ;  $u$  delegates to  $u'$  as earlier, and  $u'$  delegates to  $\hat{u}$  and  $\tilde{u}$ . For every edge  $e \in E(G)$ , we have a passive voter  $e$  and an active voter  $e'$  who votes for the candidate  $c_e$ . The passive vote  $e$  delegates to  $e'$ . Let  $e_1, e_2, e_3$  be the edges incident to

the vertex  $u$  in  $G$ . Then,  $\hat{u}$  delegates to  $e_1$  and  $e_2$ , and  $\tilde{u}$  delegates to  $e_3$ . In particular,  $\hat{u}$  delegates to two passive voters corresponding to two edges incident to  $u$  in  $G$ , and  $\tilde{u}$  delegates to the passive voter corresponding to the third edge incident to  $u$  in  $G$ . Note that the score of  $c_e$  in  $H$  was  $k$ , where  $e \in E(G)$ . Thus, to meet the same score, we add  $k-8$  dummy passive voters in  $D^e$ . Let  $D^e = \{d_1^e, \dots, d_{k-8}^e\}$ . Then, for every  $i \in [k-7]$ ,  $d_i^e$  delegates to  $d_{i+1}^e$ , and  $d_{k-8}^e$  delegates to  $e'$ . The budget is same as earlier, i.e.,  $k = \tilde{k}$ . To ensure that the arc  $(e, e')$  is not redirected, we set the cost of this arc as  $k+1$ , for every  $e \in E(G)$ . Furthermore, we set the cost of arc  $(u, u')$  as 1 and the remaining arcs as  $k+1$ . The correctness is same as earlier. Note, that now we can only redirect the arcs of type  $(u, u')$  within the budget.



**Figure 3: Modifications in Figure 2 in the vertex gadget, edge gadget, and dummy vertices for NP-hardness of Theorem 1 with constraint 1(b). Here,  $u$  is an endpoint of  $e$  in  $G$ .**

Instead of VERTEX COVER, if we give the same reduction from the HITTING SET problem, we obtain the claimed  $W[2]$ -hardness.  $\square$

Since the problem is NP-hard even for two delegations, next we study the problem for single delegation. Unfortunately, we again have a negative result.

**Theorem 2.** *CCRA is NP-hard when  $\mathcal{R}$  is union function or approval function or GreedyMRC function,  $\#delegations = 1$  even when*

- (1) *the delegation graph satisfies one of the following conditions:*
  - (a) *the maximum length of a path in the delegation graph is at most one and the cost of all the arcs is 1*
  - (b) *every vertex in the delegation graph has in-degree at most one*
- (2) *size of approval set is at most three*
- (3) *only one voter approves  $c^*$*

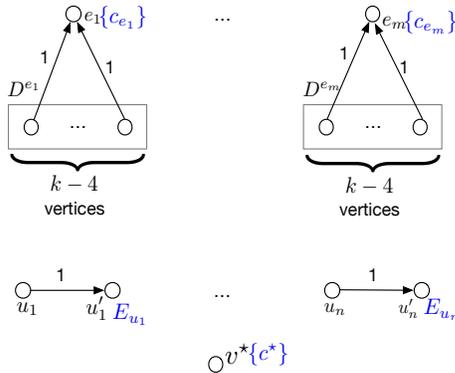
*Furthermore, it is  $W[2]$ -hard with respect to  $k + \#redirections$  even when  $\#delegations = 1$  and condition 1 and 3 holds.*

**PROOF.** We focus the proof for the union function. The proof is exactly the same for other two unraveling functions as well since in our gadget the set of candidates obtained by all these unraveling functions is same.

We give a polynomial time reduction from the VERTEX COVER problem in cubic graphs. Let  $(G, \tilde{k})$  be an instance of VERTEX COVER such that  $G$  is a cubic graph. The idea is similar to the one used to prove Theorem 1. Here,  $u$  delegates to  $u'$  and  $u'$  approves all the candidates corresponding to the edges incident on  $u$ . Dummy voters in  $D^e$  delegate to the voter corresponding to the edge  $e$ , who approves  $c_e$ . Figure 4 illustrates the construction. Next, we present the construction, in detail. We first construct an instance of CCRA that satisfies constraints 1(a), 2, and 3 in the theorem statement. We will discuss later the required modifications for constraint 1(b). The construction is as follows.

- For every edge  $e \in E(G)$ , we have a candidate  $c_e$ . Additionally, we have a candidate  $c^*$  who is our favorite candidate. For every  $u \in V(G)$ , let  $E_u$  be the set of candidates corresponding to the edges incident to  $u$  in  $G$ .
- For every vertex  $u \in V(G)$ , we have a passive voter  $u$  and an active voter  $u'$ . The voter  $u$  delegates to  $u'$  and the voter  $u'$  approves all the candidates in  $E_u$ .
- For every edge  $e \in E(G)$ , we have an active voter  $e$  who approves the candidate  $c_e$ .
- For every edge  $e \in E(G)$ , we add a set of dummy voters  $D^e = \{d_1^e, \dots, d_{k-4}^e\}$ . Each dummy voter in  $D^e$  delegates to the active voter  $e$  and the cost of this arc is 1.
- We add a special voter  $v^*$  who approves the candidate  $c^*$ .
- The cost of all the arcs is 1.
- The budget  $k$  is  $\tilde{k}$ .

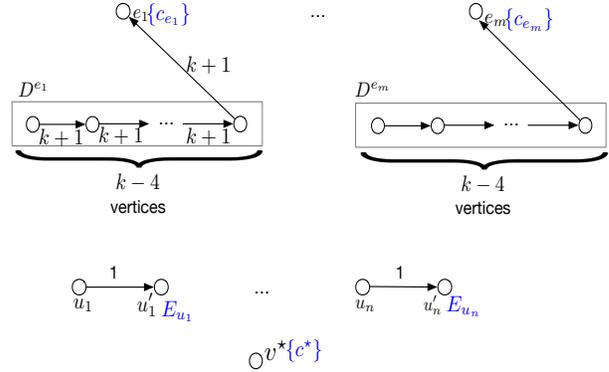
Let  $H$  be the delegation graph that we constructed. Note that in this graph the score of  $c^*$  is 0 and  $k$  for every other candidate.



**Figure 4: Illustration of NP-hardness claimed in Theorem 2.**  $E_u$  is the set of candidates corresponding to the edges incident to  $u$  in  $G$ . The set of approved candidates by an active voter is written in the blue color next to the voter name. The number on an edge is the cost of redirection.

To prove the result for the constraint 1(b), 2, and 3, we do the same modification as in Theorem 1. Note that the only active voters  $e$ , for every  $e \in E(G)$  has large in-degree. All the other vertices have in-degree (out-degree) at most 1. Thus, to decrease the out-degree of  $e$ , we add a path on the dummy voters. That is, for every  $i \in [k-5]$ ,  $e \in E(G)$ ,  $d_i^e$  delegates to  $d_{i+1}^e$  and  $d_{k-4}^e$  delegates to the voter  $e$ . Figure 5 illustrates the modification. The cost of arc of type

$(u, u')$  is 1 and for the remaining arcs, it is  $k+1$ . The correctness is same as earlier. Now, we can only redirect the arcs of type  $(u, u')$ .



**Figure 5: Modifications in Figure 5 for NP-hardness of Theorem 1 with constraint 1(b).**

Instead of VERTEX COVER, again if we give the same reduction from HITTING SET, we obtain the claimed  $W[2]$ -hardness with respect to  $k + \#$ redirections, but the size of approval set is no longer a constant. □

The complexity of CCRA when  $\#$ delegations = 1 and  $\#$ approvals = 2 elude us so far.

## 5 ALGORITHMIC RESULTS

In this section, we design exact, parameterized, and approximation algorithms for CCRA in various settings.

*Preprocessing.* We perform the following simple preprocessing step on the given CCRA instance. We partition the active voters into equivalence classes based on their approval sets, i.e., for each subset  $C' \subseteq C$  of candidates, let  $S_{C'}$  be the set of active voters whose approval set is equal to  $C'$ . Then, for each  $C'$  such that  $S_{C'}$  is non-empty, we add a *virtual* active voter  $v_{C'}$ —who does not have an actual vote, i.e., does not add to the resulting scores— and add arcs from each original active voter  $v \in S_{C'}$  of cost  $\infty$ . It is easy to see that this results in an equivalent instance of CCRA wherein the number of active voters is upper bounded by the number of subsets  $C' \subseteq C$  such that  $S_{C'} \neq \emptyset$ , which is at most  $2^m$ . In the following algorithms, we work with the instances of CCRA preprocessed in this manner.

### 5.1 Polynomial-Time Algorithm

In the following theorem, we give a polynomial-time algorithm for CCRA in the single-delegation, single-approval setting.

**Theorem 3.** *CCRA is polynomial-time solvable when  $\#$ delegations = 1 and  $\#$ approvals = 1.*

*PROOF.* Due to preprocessing and  $\#$ approvals = 1, we know that each active voter votes for a distinct candidate and  $t = m$ . Let us arbitrarily number the active voters as  $v_0, v_1, v_2, \dots, v_t$  such that each  $v_i$  votes for candidate  $c_i$ , where  $c_t = c^*$ , our preferred

candidate. Since  $\#delegations = 1$ , the set of vertices currently voting for  $c_i$ , by the virtue of the active voter  $v_i$  forms a tree, say  $T_i$ . First, we prove the following lemma that is used as a subroutine in the main algorithm.

**Lemma 1.** *There exists a polynomial-time algorithm that returns, for each  $1 \leq i \leq t$ , an array  $A_i$  of length  $|V(T_i)| + 1$ , such that for each  $0 \leq j \leq |V(T_i)|$ ,  $A_i[j]$  is equal to the minimum-cost of a subset of arcs from  $T_i$ , whose redirection yields at least  $j$  votes for  $c^*$ . One can also compute in polynomial-time, as the corresponding subsets of arcs realizing the minimum cost.*

**PROOF.** We perform a bottom-up dynamic programming on the  $T_i$ , which we root at the active voter  $v_i$ . Thus, every arc  $(u, v)$  is directed from a child  $u$  to its parent  $v$ . For each vertex  $v$  with parent  $w$ , we associate two arrays, namely  $A_v$  and  $B_v$  – since the root  $v_i$  does not have a parent, we only define  $B_{v_i}$  for it, which is also the final output of the algorithm.

Let  $\ell_v$  denote the number of nodes in the subtree rooted at  $v$  (denoted by  $T_v$ ), then  $A_v$  has length  $\ell_v + 1$  and  $B_v$  has length  $\ell_v$ , these two arrays represent the computational primitives for the dynamic steps in the algorithm. For  $0 \leq j \leq \ell_v - 1$ ,  $B_v[j]$  is the minimum cost of arcs from  $E(T_v)$ , whose redirection yields at least  $j$  votes from  $V(T_v) \setminus \{v\}$ ; whereas for  $0 \leq j \leq \ell_v$ ,  $A_v[j]$  is the minimum cost of arcs from  $E(T_v) \cup \{(v, w)\}$ , whose redirection yields at least  $j$  votes from  $V(T_v)$ .

**Leaf case.** If  $u$  is a leaf, then  $B_u$  is an array of length 1, with  $B_u[0] := 0$ .

**Computing  $A_v$  given  $B_v$ .** Let  $v$  be a vertex (other than the root  $v_i$ ) with parent  $w$ . Suppose we have already correctly computed the array  $B_w$  of length  $\ell = |V(T_w)|$ . Then, we define  $A_v[\ell] = cost((v, w))$ , since redirection of edge  $(v, w)$  is the only way to obtain  $\ell$  votes. For all other  $0 \leq j \leq \ell - 1$ , define

$$A_v[j] = \min \{B_w[j], cost((v, w))\}. \quad (1)$$

**Computing  $B_v$  given the arrays  $A$  for the children.** Consider a vertex  $v$  with children  $u_1, u_2, \dots, u_r$  (numbered arbitrarily), and for each  $1 \leq q \leq r$ , let  $e_q = (u_q, v)$  be the edge between  $v$  and  $q$ -th child. Suppose that we have computed the arrays  $A_{u_q}$  for every  $u_q$ . For simplicity, we denote  $A_{u_q}$  by  $A_q$ . We compute the array  $B_v$  by iteratively merging the arrays for the children, as follows. First, we merge  $A_{\leq 1} := A_1$  and  $A_2$  to compute  $B_{\leq 2}$  (discussed next). Then, we merge  $A_{\leq 2}$  with  $A_3$  to compute  $A_{\leq 3}$ , and so on. The final output  $B_v$  is equal to  $A_{\leq t}$  computed in this manner.

Now we discuss how to compute  $A_{\leq y+1}$ , given an array  $A_{\leq y}$  of length  $\ell'_y + 1$ , where  $\ell'_y = |\bigcup_{1 \leq q \leq y} V(T_{u_q})|$ , and an array  $A_{y+1}$  of length  $\ell_{y+1} + 1$ . This is done as follows. For each  $0 \leq j \leq \ell'_y + \ell_{y+1} + 1$ , define

$$A_{\leq y+1}[j] = \min_{0 \leq x \leq \min\{\ell'_y, \ell_{y+1}\}} \{A_{\leq y}[x] + A_{y+1}[j - x]\} \quad (2)$$

The correctness of the dynamic program is argued in the supplementary.  $\square$

We use the algorithm in Lemma 1 to compute the arrays  $A_i[\cdot]$  for each  $0 \leq i < t$  (since  $v_t$  votes for  $c^*$ , we will not redirect any arcs out of tree  $T_t$ ). We use these arrays to perform another level of dynamic programming, to solve CCRA optimally. Our DP table is

parameterized by a tuple,  $(i, k, x)$ , where  $0 \leq i < t$ , and  $0 \leq x, k \leq n$ , and the corresponding entry  $T[i, k, x]$  is equal to the minimum cost of arcs redirected from  $T_1 \cup \dots \cup T_i$  to get at least  $k$  votes for  $c^*$  (i.e., the arcs will be redirected into  $v_t$ ), and leaving at most  $x$  votes for the first  $i$  candidates,  $c_1, \dots, c_i$ .

For the base case, we define  $T[0, k, x] = 0$  iff  $k = x = 0$  and  $T[0, k, x] = +\infty$  otherwise. Suppose we have already computed all the entries of the form  $T[i - 1, k, x]$ . Let  $l_i = |V(T_i)| + 1$  denote the number of voters in the  $i$ -th tree. We compute the entries in the  $i$ -th row using the following recurrence.

$$T[i, k, x] = \min_{l_i - x \leq j \leq l_i} \{A_i[j] + T[i - 1, k - j, x]\} \quad (3)$$

The correctness of this recurrence is argued in the supplementary. The final output of the algorithm is the minimum value  $T[t, k, x]$  over all  $0 \leq x < k \leq n$ .  $\square$

## 5.2 Parameterized Algorithms

We design the *fixed-parameter tractable* (FPT) algorithms with respect to the parameters, number of vertices ( $n$ ), the number of active voters ( $t$ ), and number of candidates ( $m$ ). First, we have the following result.

**Theorem 4.** *CCRA is FPT w.r.t. the parameter  $n$ .*

**PROOF.** The proof is by enumerating solutions and returning a minimum-cost solution found wherein our preferred candidate  $c^*$  is the unique winner. Note that the number of redirected arcs in any solution is at most  $n^2$ , and for each redirected arc, there are at most  $n$  choices for the new destination. Thus, the number of solutions is upper bounded by  $2^{n^2} \cdot n^{n^2} = 2^{O(n^2 \log n)}$ . In the single-delegation scenario (i.e.,  $\#delegations = 1$ ), note that the number of edges in the graph is upper bounded by  $n$ , and thus the running time of the algorithm improves to  $2^{O(n \log n)}$ .  $\square$

Now we turn to CCRA when  $\#delegations = 1$  and  $\#approvals \geq 1$ , and first design an XP algorithm for CCRA parameterized by the number of active voters (Theorem 5). Subsequently, we show that in the special case of this setting, one can obtain an FPT approximation scheme (FPT-AS) for CCRA parameterized by the number of active voters (Theorem 6), i.e., a  $(1 + \epsilon)$ -approximation for the minimum cost that runs in time  $f(t, \epsilon) \cdot (m + n)^{O(1)}$  where  $t$  is the number of active voters.

**Theorem 5.** *CCRA is XP with respect to  $t$ , the number of active voters when  $\#delegations = 1$ .*

**PROOF SKETCH.** For each active voter  $v_i$ ,  $1 \leq i \leq t$ , let  $T_i$  be the corresponding tree and let  $l_i = |V(T_i)|$  denote the number of voters in the tree.

Fix  $1 \leq i \leq t$ . We generalize the dynamic programming algorithm from Lemma 1 to first compute the answers to the subproblems of the following form. Let  $S = \{v_1, v_2, \dots, v_{t'}\}$  be a multiset of size  $1 \leq t' \leq t$  (with repetitions) such that  $v_j \in \{1, 2, \dots, l_i\}$ . Then, let  $T[i, S]$  denote the minimum cost of deleting a subset of arcs  $A$  from  $T_i$  such that,  $A$  can be decomposed into  $A_1 \uplus A_2 \uplus \dots \uplus A_{t'}$ , where the total number of votes obtained from the set  $A_j$  is at least  $v_j$ . In the supplementary, we generalize the algorithm from Lemma 1, and show that table can be computed in time  $n^{O(t)}$ .

Suppose that we have computed the tables  $T[i, S]$  for each  $i$  and each multiset  $S$ . Now, fix an optimal solution. For each  $1 \leq i \neq j \leq t$ , let  $n(i, j)$  denote the number of votes redirected by the optimal solution from  $T_i$  into  $v_j$ . We guess the numbers  $n(i, j)$  – note that the number of guesses is upper bounded by  $n^{t^2}$ . Fix one such guess. For each  $1 \leq i \leq t$ , define the multiset  $S_i = \{n(i, j) : 1 \leq j \leq t, j \neq i\}$ , and we look up the optimal cost of redirecting a subset of arcs from  $T_i$  to get the votes as given in  $S_i$  using the entry  $T[i, S_i]$ . The total cost corresponding to this guess is defined as  $\sum_{i=1}^t T[i, S_i]$ . Finally, we return the minimum solution found over all guesses. It follows that the algorithm runs in time  $n^{O(t^2)}$  and returns an optimal solution.  $\square$

Now we consider the special case of CCRA in the single delegation, multi-approval setting, where our preferred candidate  $c^*$  appears *separately* from the rest of the candidates. More specifically, we assume that there exists an active voter, say  $v_t$ , whose approval set is equal to  $\{c^*\}$ , and the approval sets of other active voters do not contain  $c^*$  – note that they may be arbitrary subsets of  $C \setminus \{c^*\}$ . At an intuitive level, this setting is easier to handle due to the fact that, in an optimal solution, each redirected edge is redirected into  $v_t$ . We proceed to the following theorem that gives an FPT approximation scheme (FPT-AS).

**Theorem 6.** *CCRA with #delegations = 1 and #approvals  $\geq 1$ , admits an FPT-AS parameterized by  $t$  and  $\epsilon$  in the special setting, where  $c^*$  appears separate from the rest of the candidates. That is, in this setting, for any  $0 \leq \epsilon \leq 1$ , there exists an algorithm that runs in time  $(t/\epsilon)^{O(t)} \cdot (m+n)^{O(1)}$  and returns a solution of cost at most  $(1+\epsilon)$  times the optimal redirection cost.*

**PROOF.** First, by iterating over all edges, we “guess” the most expensive edge in the optimal solution. Consider one such guess corresponding to edge  $e$  with cost  $w$ . In the following, we describe the algorithm assuming this guess is correct, i.e.,  $w$  is indeed the weight of the most expensive edge in the optimal solution. Then, we want to find a solution that only contains edges of cost at most  $w$ , which implies that *OPT*, the cost of the optimal solution is at least  $w$  and at most  $n \cdot w$  (recall that #delegations = 1, so the total number of edges in  $G$  is at most  $n$ ).

Next, for each active voter  $v_i \neq v_1$ , let  $w_i$  denote the total cost of the edges redirected from the tree  $T_i$  into  $v_1$  in the optimal solution. Note that  $\sum_i w_i = \text{OPT}$ . Let  $w'_i = \max\{w_i, \epsilon w/2n\}$ . Note that  $\epsilon w/n \leq w'_i \leq wn$ , and  $\sum_{i=2}^t w'_i \leq (1+\epsilon/2) \cdot \text{OPT}$ , where *OPT* denotes the cost of an optimal solution. Next, let  $p_i = \lceil \log_{1+\epsilon}(w'_i) \rceil$ , i.e.,  $(1+\epsilon/3)^{p_i}$  is the smallest power of  $1+\epsilon$  that is at least  $w'_i$ . Note that for each  $2 \leq i \leq t$ ,  $\log_{1+\epsilon/3}(\epsilon w/2n) \leq p_i \leq \log_{1+\epsilon/3}(nw) + 1$ , which implies that all the value of  $p_i$  lies in the interval of range  $r = \log_{1+\epsilon/3}\left(\frac{nw}{\epsilon w/2n}\right) = O\left(\frac{\log n}{\epsilon^2}\right)$ .

Next, we guess the value of  $p_i$  for each  $2 \leq i \leq t$  – note that the number of guesses is at most  $r^{t-1}$ . Let  $p'_2, p'_3, \dots, p'_t$  be the guessed values. Then, we use the polynomial-time algorithm in Lemma 1 to find, for each  $2 \leq i \leq t$ , the largest number of votes that can be obtained from the tree  $T_i$  using cost at most  $(1+\epsilon/3)^{p'_i}$ , and redirect such edges to  $v_1$ . After doing this for all  $2 \leq i \leq t$ , we check whether  $c^*$  is the unique winning candidate. We return the minimum-cost solution found over all guesses for  $w$  and all guesses

for  $p'_i$ . Note that when we correctly guess the values of  $p_i$ , it holds  $w_i \leq (1+\epsilon/3)^{p_i}$ , thus, the maximum number of votes from  $T_i$  that can be obtained using a budget of  $(1+\epsilon/3)^{p_i}$  is at least the number of votes that can be obtained using a budget of  $w_i$ . Thus, in the iteration corresponding to the correct guess, when each guessed value is indeed equal to  $p'_i$ , the number of votes obtained for  $c^*$  is no fewer than that in the optimal solution (since from each tree  $T_i$ , the number of votes obtained by a solution of cost  $(1+\epsilon/3)^{p'_i}$  is no fewer than that of cost  $w_i$ ). Furthermore, the cost of the combined solution thus obtained, is at most  $(1+\epsilon/2) \cdot (1+\epsilon/3) \leq (1+\epsilon)$  times the cost of an optimal solution.

Finally, we argue about the running time, which is dominated by the number of guesses for the values of  $p_i$ . The total number of guesses is at most  $\left(\frac{\log n}{\epsilon}\right)^{O(t)}$ . Now we consider two cases. If

$t \leq \frac{\log n}{\log \log n}$ , then the previous quantity is at most  $\frac{1}{\epsilon^{O(t)}} \cdot n^{O(1)}$ .

Otherwise, if  $t > \frac{\log n}{\log \log n}$ , then  $\left(\frac{\log n}{\epsilon}\right)^{O(t)} \leq \left(\frac{t}{\epsilon}\right)^{O(t)}$ . Thus, in either case, we can upper bound the number of guesses, and in turn the running time of the algorithm by  $(t/\epsilon)^{O(t)} \cdot (m+n)^{O(1)}$ .  $\square$

Recall that our preprocessing step bounds the number of active voters  $t$  by the number of distinct approval sets, which is at most  $2^m$ . Thus, Theorem 5 and Theorem 6 immediately result in the following corollary.

**Corollary 1.** *Consider CCRA when #delegations = 1 and #approvals  $\geq 1$ . In this setting, the problem is XP w.r.t.  $m$  (the number of candidates), and admits an FPT-AS parameterized by  $m$  and  $\epsilon$  in the special case when  $c^*$  appears separate from the rest of the candidates.*

## 6 OUTLOOK

As liquid democracy is gaining more attention in different applications – as well as more real-world usage, including in high-stakes ones – it is important to study different aspects of it, including the possibility of external agents controlling and rigging elections performed using liquid democracy. In this context, here we considered a particular form of election control for liquid democracy by redirecting arcs. Below we describe few future research directions that we view as important and promising: (1) A natural expansion of our work may be to consider not only single-winner elections but also multiwinner elections; and, related, to consider other unraveling functions  $\mathcal{R}$  and other voting rules  $\mathcal{W}$ . (2) A different generalization of our work is to consider general underlying social graphs, as described in Remark 2. (3) An immediate future research direction is to consider other forms of election control and bribery, e.g. do not redirect arcs but change the delegation graph in other ways. (4) Another future research direction would be to study the type of control we study here but from a more practical point of view, e.g., by performing computer-based simulations to estimate the feasibility of successfully controlling a real world election by redirecting liquid democracy arcs.

## REFERENCES

- [1] Mohammad Abouei Mehrizi and Gianlorenzo D'Angelo. 2020. Multi-winner election control via social influence. In *Proceedings of SIROCCO '20*. 331–348.
- [2] John J Bartholdi III, Craig A Tovey, and Michael A Trick. 1992. How hard is it to control an election? *Mathematical and Computer Modelling* 16, 8-9 (1992), 27–40.
- [3] Daan Bloembergen, Davide Grossi, and Martin Lackner. 2019. On rational delegations in liquid democracy. In *Proceedings of AAAI '19*, Vol. 33. 1796–1803.
- [4] Christian Blum and Christina Isabel Zuber. 2016. Liquid democracy: Potentials, problems, and perspectives. *Journal of political philosophy* 24, 2 (2016), 162–182.
- [5] Robert Brederick, Piotr Faliszewski, Rolf Niedermeier, and Nimrod Talmon. 2016. Complexity of shift bribery in committee elections. In *Proceedings of AAAI '16*, Vol. 30.
- [6] Markus Brill and Nimrod Talmon. 2018. Pairwise Liquid Democracy. In *Proceedings of IJCAI '18*, Vol. 18. 137–143.
- [7] Ioannis Caragiannis and Evi Micha. 2019. A Contribution to the Critique of Liquid Democracy. In *Proceedings of IJCAI '19*. 116–122.
- [8] Rachael Colley, Umberto Grandi, and Arianna Novaro. 2022. Unravelling multi-agent ranked delegations. *Autonomous Agents and Multi-Agent Systems* 36, 1 (2022), 9.
- [9] Gianlorenzo d'Angelo, Esmaeil Delfaraz, and Hugo Gilbert. 2022. Computation and bribery of voting power in delegative simple games. In *Proceedings of AAMAS '22*, Vol. 1. 336–344.
- [10] Palash Dey, Arnab Maiti, and Amaty Sharma. 2021. On parameterized complexity of liquid democracy. In *Proceedings of CALDAM '21*. 83–94.
- [11] Rodney G. Downey and Michael R. Fellows. 1995. Fixed-Parameter Tractability and Completeness I: Basic Results. *SIAM Journal of Computing* 24, 4 (1995), 873–921.
- [12] Bruno Escoffier, Hugo Gilbert, and Adèle Pass-Lanneau. 2019. The convergence of iterative delegations in liquid democracy in a social network. In *Proceedings of SAGT '19*. 284–297.
- [13] Piotr Faliszewski, Rica Gonen, Martin Koutecký, and Nimrod Talmon. 2022. Opinion diffusion and campaigning on society graphs. *Journal of Logic and Computation* 32, 6 (2022), 1162–1194.
- [14] Piotr Faliszewski and Jörg Rothe. 2016. Control and Bribery in Voting. In *Handbook of Computational Social Choice*. Cambridge University Press, 146–168.
- [15] Piotr Faliszewski, Arkadii Slinko, and Nimrod Talmon. 2020. Multiwinner rules with variable number of winners. In *Proceedings of ECAI '20*. IOS Press, 67–74.
- [16] M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. 1974. Some Simplified NP-Complete Problems. In *Proceedings of STOC '74*. ACM, 47–63.
- [17] Paul Gözl, Anson Kahng, Simon Mackenzie, and Ariel D Procaccia. 2021. The fluid mechanics of liquid democracy. *ACM Transactions on Economics and Computation* 9, 4 (2021), 1–39.
- [18] Pallavi Jain, Krzysztof Sornat, and Nimrod Talmon. 2022. Preserving consistency for liquid knapsack voting. In *Proceedings of ECAI '22*. 221–238.
- [19] Anson Kahng, Simon Mackenzie, and Ariel Procaccia. 2021. Liquid democracy: An algorithmic perspective. *Journal of Artificial Intelligence Research* 70 (2021), 1223–1252.
- [20] Alois Paulin. 2020. An overview of ten years of liquid democracy research. In *DGO '20*. 116–121.
- [21] Ariel D Procaccia, Jeffrey S Rosenschein, and Aviv Zohar. 2007. Multi-Winner Elections: Complexity of Manipulation, Control and Winner-Determination.. In *Proceedings of IJCAI '07*, Vol. 7. 1476–1481.
- [22] Yuzhe Zhang and Davide Grossi. 2021. Power in liquid democracy. In *Proceedings AAAI '21*, Vol. 35. 5822–5830.