THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 1 Capitole

Cotutelle internationale:

Présentée et soutenue par

Rachael COLLEY

Le 26 juin 2023

Expressive and Rational Delegations in Voting

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Laurent PERRUSSEL et Umberto GRANDI

Jury
M. Richard BOOTH, Rapporteur
M. Bruno ESCOFFIER, Rapporteur
Mme Leila AMGOUD, Examinatrice
Mme Maria POLUKAROV, Examinatrice
M. Laurent PERRUSSEL, Directeur de thèse
M. Umberto GRANDI, Co-directeur de thèse

Abstract

Through the digitisation of democracies, more complex voting methods increase the frequency of decisions and expand citizen input on a wider range of topics. As a result, digital democracy has the potential to deliver better outcomes, higher voter turnout and more satisfied citizens. However, despite the increasing use of artificial intelligence in daily life, trust in the digitisation of large-scale political decisions is not uniformly embraced. Trust in the digitisation of collective decisions is being built with the creation of new digital democracy platforms. As a result, the public is more willing to give their opinions, albeit for decisions with low impact, e.g., deciding on a time to meet, how a local council should spend funding and responding to opinion polls online.

This thesis develops the theoretical research underpinning the implementation of digital democracy, particularly models that incorporate the connections between agents, thus furthering their ability to express their opinions and do so more rationally. Specifically, we explore models of delegative democracy and opinion diffusion where the connections between agents represent trust and influence.

In delegative democracy models, trust is represented by one agent delegating their vote to another, who then votes on their behalf. The most well-known models of delegative democracy are proxy voting and liquid democracy. In the former, voters choose which representative from a predefined set will vote on their behalf. In the latter, the delegations can be transitive. When agents receive delegations, they are free to vote or delegate. If they vote directly on the issue, they do so with their own vote and any they have received via delegations. If they delegate, they pass their vote and any received delegations to another agent.

We extend liquid democracy by studying more complex delegations and elections with multiple interconnected issues. Here, we are interested in the properties of our procedures that resolve delegations. In liquid democracy, we introduce a new voting power measure, showing how an agent's position within a social network can impact their power in deciding an outcome. We also study liquid democracy from a more realistic perspective, where delegations are intended to last for multiple elections. Such models paired with a digital platform can balance the benefits and drawbacks of direct democracy and representative democracy. Consequently, they can directly choose the representative who will vote on their behalf and update their representative instantly. Making delegations more rational and expressive gives the voters even more control over how their vote should be determined, thus increasing the representation of the model.

We also study a related model where the connections between agents represent influence, modelling agents who can change each other's opinions. We study an extension of opinion diffusion where the agents' opinions update with respect to a Boolean function instead of a quota rule. Hence, the way in which agents' opinions are updated in a more fine-grained manner. We make the opinion diffusion model more expressive by connecting it to the field of Boolean networks, examining convergence, maximising opinions via asynchronous updates and linking multi-agent delegations to opinion diffusion.

This thesis contributes to a deeper understanding of digital democracy's theoretical foundations, focusing on models of delegative democracy and opinion diffusion. In particular, we study how models of delegative democracy can be made more expressive and rational.

Resumé

L’utilisation de la technologie permet de développer des méthodes de vote plus complexes, d’augmenter la fréquence des décisions et d’élargir la participation des citoyens. La démocratie numérique possède donc le potentiel de fournir de meilleurs résultats, une participation électorale plus élevée et des citoyens plus satisfaits. Cependant, malgré l’utilisation croissante de l’intelligence artificielle dans la vie quotidienne, la numérisation des décisions politiques à grande échelle n’est pas uniformément acceptée. La confiance du public dans cette numérisation se construit à mesure que des décisions collectives sont prises à travers de nouvelles plateformes de démocratie numérique. Par conséquent, le public est plus disposé à donner son avis, même pour des décisions ayant peu d’impact telles que planifier un rendez-vous, décider comment un conseil local devrait dépenser des fonds ou répondre à des sondages en ligne.

Cette thèse développe des recherches théoriques étayant l’implémentation de la démocratie numérique, notamment les modèles qui intègrent les connexions sociales entre les agents permettant leur capacité de partager ses opinions d’une manière plus expressive et rationnelle. Plus précisément, nous explorons des modèles de démocratie délégative et de diffusion de l’opinion où la confiance et l’influence sont représentées par les connexions entre les agents.

Dans les modèles de démocratie délégative, la confiance est représentée par la délégation de la voix d’un agent à un autre qui votera en son nom. Les modèles de démocratie délégative les plus connus sont le vote par procuration et la démocratie liquide. Dans ce premier, les électeurs choisissent le représentant qui votera en leur nom dans un ensemble prédéfini. Dans ce dernier, les délégations peuvent être transitives : lorsqu’un agent reçoit des délégations, il est libre de voter ou de déléguer à nouveau. S’il vote directement sur la question, il le fait avec sa propre voix ainsi que celles qui lui ont été déléguées. S’il délègue, il transmet sa voix et les délégations reçues à un autre agent.

L’une de mes contributions à la démocratie délégative a été d’étendre la démocratie liquide en étudiant des délégations et des élections plus complexes avec plusieurs questions interconnectées. En démocratie liquide, nous avons introduit une nouvelle mesure de pouvoir qui montre comment la position d’un agent dans un réseau social peut avoir un impact sur son pouvoir de décision. Nous étudions également la démocratie liquide sous un angle plus réaliste, où les délégations sont destinées à durer plusieurs élections. Ces modèles associés à une plateforme numérique sont à mi-chemin entre la démocratie directe et la démocratie représentative, où le choix d’un représentant peut être mis à jour instantanément et son vote décidé directement. En rendant les délégations plus rationnelles et plus expressives, cela permet aux électeurs de mieux contrôler la manière dont leur vote doit être déterminé, ce qui accroît la représentativité du modèle.

Nous considérons également un modèle où les connexions entre les agents représentent l'influence sociale. Les agents peuvent ainsi changer les opinions des autres. Ma contribution consiste à donner une extension de la diffusion d'opinion où les opinions des agents sont mises à jour à partir d'une fonction booléenne. Nous étendons le modèle de diffusion d'opinion en le reliant au domaine des réseaux booléens, en examinant sa convergence, en maximisant les opinions à travers des mises à jour asynchrones et en reliant les délégations multi-agents à la diffusion d'opinion.

Dans l'ensemble, cette thèse contribue à une compréhension plus profonde des fondements théoriques de la démocratie numérique, en se concentrant sur les modèles de démocratie délégative et de diffusion d'opinion. Nous étudions notamment la manière dont les modèles de diffusion de l'opinion peuvent être plus expressifs et rationnels que dans la démocratie liquide.

Acknowledgments

This thesis was only possible with the support of many people.

I would first like to thank my supervisor, Umberto Grandi. His care, support and patience throughout my PhD have made my time in Toulouse both formative and enjoyable, and I couldn't be more grateful.

Next, I thank Laurent Perrussel, Arianna Navora, my collaborators, and my wider academic circle for their consistent encouragement and openness to teaching me their expertise.

I am grateful to my thesis committee for giving their time to assess my work as well as for their constructive comments, especially to my reviewers, Bruno Escoffier and Richard Booth.

Finally, I would like to thank my friends, family and partner for their support in my personal life (as well as a bit of proofreading!), making this thesis possible.

Contents

Chapter 1

Introduction

Digital technology is present in almost every aspect of modern life, from socialising to organising medical appointments to managing our finances. By incorporating digital technology into our lives to such an extent, we have not only learnt to trust each of its applications but to embrace them. One area of modern life in which we have avoided digitisation, or at least been more hesitant to adapt to, is in our collective decision-making processes [Mancini, 2014]. We see that by means of electing public officials, many democracies have not evolved and remain in line with the public and social infrastructure currently in place, e.g., the Westminster system in the UK was first developed in the medieval period and has only had minor updates since The Parliament Act 1911, itself a mild reform of the same structure in place since 1801 [Grant, 2009]. Thus, although digital democracy could now be easily implemented at scale, there is unease from the public and decision-makers to digitise our large-scale collective decisions, such as national elections. Although this aversion to digital democracy comes from many different lines of reasoning, the unease towards its adoption often stems from a lack of trust in such systems. For example, an absence of clear accountability of the system, suspicion in the credibility of the outcome, and doubts in its ability to maintain the one person, one vote principle [Rana et al., 2015].

This lack of trust is not unfounded. There have been issues in many attempted implementations of new voting methods. For instance, an early step away from traditional paper ballots in national elections was the introduction of mechanical voting machines [Alvarez and Hall, 2010]. The mechanisation of the voting process promised a more robust, accurate and straightforward counting method with less chance of human corruption and error when counting. Nonetheless, one of the most infamous accounts of the use of mechanical voting machines is that of the faulty tabulation machines during the 2000 presidential elections in the USA [Mebane, 2004]. Here, the reliance on a simple device that punched a hole in a ballot led to 50,000 unintended spoiled votes without which, Mebane [2004] has argued, the election result may well have been different. Circumstances like this make the public wary of changing the methods by which we make crucial decisions.

Even with such a lack of public trust, more technologically advanced solutions are being created and implemented, and in turn, there are more issues of trust and fairness being raised. Staying in the context of the USA, the introduction of e-voting terminals in polling stations was coupled with concerns about their usability, as adopting the machines could exclude portions of society or misrepresent their views [Bederson et al., 2003]. Due to worries about these systems, there has been a breadth of academic research to comprehend the impacts of their introduction. For instance, Buldas and Mägi [2007] studied practical security measures between different e-voting systems, which were deemed vastly different in their ability to be secure and reliable, even though their underlying technologies were extremely similar. Hence, the study surrounding systems that have been put into practice needs to be comprehensive and thoroughly explore the impacts and implications of the systems. We call this *responsive research*, and it reassures the public and builds their trust in these implemented systems. Responsive research in this domain inspects the implementations of digital democracy systems, determining their successes and, in the case of failures, looking to improve the systems so that their behaviour aligns with their purposes.

Much of the research into digital democracy tools is not responsive but is instead forward-looking. Instead of asking what are the impacts of the systems already in place, some question how incorporating technology in collective decision-making can revolutionise its methodology, accessibility, and its place in the broader functioning of society. This *exploratory research* allows the public to see how they can benefit from new voting systems.

These benefits could include making the voting process more flexible and practical for the voters, such as allowing online voting rather than being restricted to a geographical location. For example, although voting to take place online in Estonian elections has not drastically increased the total voter turnout, there has been a clear increase in the percentage of voters using it [Ehin et al., 2022]. Thus, there is a trend that voters are consistently making use of the flexibility of the system.

Another vast area of exploratory research in digital democracy tools is to theoretically examine voting models and their decision-making protocol from a more structural perspective. This could mean creating and advocating for more complex aggregation functions that, although they have beneficial properties, would require a level of computational power that would exceed that capable by hand, especially on a large scale.[1] Some advocate further still, wanting digital democracy to be used to test the boundaries of collective decision-making in society. A prominent proponent of the reinvention of our democratic system is Mancini [2014], who argues in favour of creating a new democratic process to match all other aspects of our now digitised lives.

This thesis examines both this forward-looking and responsive research, studying the exploration of new voting models aided in their implementation by digital democracy tools. In particular, the thesis focuses on voting models that include *delegations*. The use of delegations in voting models makes the process more representative and gives voters more control over how their vote is determined, balancing

[1]For example, Tideman [1995] argued that the voting mechanism of *single transferable vote* (STV) gives a more defensible outcome than the plurality rules, yet countries such as Ireland have had problems with STV when determining the outcome by hand. In a recent election in Ireland to select their members of the EU Parliament, the results were delayed by 28 days due to the need to recount ballots. Following this, some, such as Doyle [2019], called for implementing e-voting systems in future elections.

possible drawbacks in a less transparent and trustworthy system. Two early proponents of this are Miller [1969] and Tullock [1967], who argued for using digital tools to realise a model of proxy voting in place of a congressional model. Here, any voter could vote directly on a motion being passed via a computer in their home or use the same system to choose their representative, who, in turn, would vote on behalf of themself as well as those who chose them as a representative. Although these representatives must be somewhat predetermined, Miller envisioned that not only those acting as politicians would be representatives. In addition, Miller raises questions about security in the voting process, voter fraud, and how technology could counteract this, describing the use of a key given to each voter, allowing them to access their ballot via their computer.

This research provides a vision to make the democratic process more engaging and involved via the use of technology. It is now referred to by the umbrella term of interactive democracy, to follow the terminology of Brill [2018]. In this work, Brill [2018] creates a picture of what can be built under the remit of interactive democracy, highlighting many disparate research directions and finding dynamic uses for such models. Our focus on delegative democracy falls under this umbrella term.

In models of delegative democracy, the connection between voters allows them to rely on one another to find better outcomes. The creation of models of delegative democracy is born from balancing the drawbacks of representative democracy and direct democracy. Models of representative democracy generalise the views of the many into a few representatives who each vote on many decisions on behalf of their community. This can leave voters having their views under- or un-represented, reducing the variety of opinions present in the final voter [Bogdanor, 1997]. This is especially concerning when representatives are part of the bill creation process [Chamberlin and Courant, 1983]. Direct democracy achieves the maximum representation of individual opinions but requires a lot of effort from every member of a community to be sufficiently informed on every issue being voted on, which could leave the turnout rate of elections low if voters do not vote on issues on which they are uninformed [Lutz, 2007]. Delegating allows voters far more choice in who represents them by creating a direct link between the voters and the representatives. Crucially, this direct link is not guaranteed in representative democracy, wherein voters may vote for a candidate who is ultimately not selected to be their representative. Moreover, delegations do not require voters to be informed on every issue; instead, they can entrust certain decisions to another agent. Liquid democracy furthers this still, as an agent chosen as a delegate may decide to delegate their vote and any voters they have received to any other voter as their delegate. Without the guarantee of your delegate voting directly, as in proxy voting, delegations are resolved transitively. This means that a direct vote votes on behalf of those who delegate to them both directly and indirectly through other delegators. Thus, the interpretation of the transitive nature of delegations in liquid democracy is that if you trust your delegate, you should also trust their choice of ballot. However, in some cases, transitive delegations may result in delegation cycles, i.e., a path of delegations which does not lead to a direct voter.

Delegative democracy has been studied from many different angles with different techniques and purposes, but there is consistently a clear overlap in research disciplines. Political scientists and philosophers tackle justifying delegative democracy at a systemic level, examining how it could be an alternative to representative democracy. Economists interested in the subfield of social choice theory consider the aspects of liquid democracy surrounding the mechanism's ability to find an outcome that represents the collective's preferences. Computer scientists investigate delegative democracy from both applied and theoretical perspectives. The applied questions focus on implementing delegative democracy while ensuring that critical democratic principles are upheld, such as vote secrecy and security. In contrast, this thesis is mainly concerned with theoretical questions, specifically the algorithmic questions raised when creating delegative voting models.

This thesis contributes to the field of computational social choice [Brandt et al., 2016]. Therefore, while our methods and techniques are drawn from theoretical computer science, we apply them to social choice theory. Hence, we are interested in creating algorithms to solve problems in models of delegative democracy, such as resolving delegations. We are interested in questions such as: *do the algorithms always terminate? how quickly do they terminate? which procedure returns better outcomes? what is a better result in this model?*

From the classical models of proxy voting and liquid democracy, we are interested in broadening these models to make delegations more rational and expressive, contributing to exploratory and responsive research.

Delegative Democracy Case Study. We exemplify this organic cycle between responsive and exploratory research by discussing delegative democracy, where the form of delegations in these voting models has evolved:

- an initial idea from Dodgson [1884] would allow candidates to redistribute their vote share via delegations to other candidates in multi-winner elections;
- in models of *proxy voting*, voters delegate to one of the voting representatives who vote on their behalf;
- in *liquid democracy* any voter can decide to vote directly on the issue or delegate to any other agent; here the delegations are transitive and resolved by finding a direct vote on their outgoing delegation path.

Such exploration of the possibility of delegative democracy models led to the creation of voting platforms implementing delegations (see Paulin [2020] for an overview). One of these platforms is LiquidFeedback [Behrens et al., 2014], a software for digital voting via a model of liquid democracy (as well as many other features, such as deliberation). Implementing what was only a theoretical model has fed back into the research community in two ways. First, in exploring the extensions of liquid democracy, we can either move towards more realistic situations, such as liquid democracy elections with multiple interconnected issues or change the model to benefit the voters with respect to certain norms. For instance, we can avoid cycles [Brill et al., 2022] or remove the concentration of power [Gölz et al., 2018]. Second, the implementation of LiquidFeedback has affected the study of liquid democracy in that there is also responsive research based on the behaviour exhibited on the platform. For example, the empirical analysis of the voter behaviour including the distribution of voting power [Kling et al., 2015] or studying how votes can intentionally create delegation cycles (see Behrens et al. [2022] for the inspiration for the study given in Section 5.3 of this thesis). △

This thesis is also concerned with a related model of opinion diffusion. Opinion diffusion studies the impact of micro-level interactions between agents on the macrolevel properties of the collective's opinions. This relates to how delegations play a role in our communities. Shapiro and Talmon [2022] advocate for a more holistic view of the term 'digital democracy', arguing that it should also explore ways of making digital society democratic, like our own societies. When considering what would make a democratic digital community, they argue that a vital pillar would ensure that communication between the community's personal digital devices is possible. Thus, opinion diffusion can model forms of deliberative or delegative democracy where the mechanism has not been developed to incorporate it. Shapiro [2022] argues that liquid democracy can be constitutional or grassroots, where the delegations occur either formally inside or outside the mechanism, respectively. The connection between delegations and influence was pointed out by Christoff and Grossi [2017a]. Thus, opinion diffusion has a strong connection to digital democracy with grassroots delegations.

1.1 Context

This section puts the contribution of this thesis into context by addressing some of the related research areas and giving an overview of the previous work it builds on, mainly in the field of computational social choice. We first provide an overview of the broader research on social choice on social networks, and then we move specifically to the context of delegative democracy. In the applications we study, the edges in a social network represent a possible connection between agents. For example, edges represent delegations from one agent to another, showing that the first voter wants the second to vote on their behalf. For opinion diffusion, an edge in a social network means that one agent can, in part, change the opinion of another.

The study of social networks in (computational) social choice has increased as the community moves towards more realistic models. A group rarely makes collective decisions without information about each other's opinions, and they will communicate during the lead-up to the decision being taken. Following the survey of Grandi [2017], most of the related research on social networks can be split into *i)* the effect of social networks on collective decisions and *ii)* the creation of social choice mechanisms utilising social networks. Models of delegative democracy fall into the second category, as the mechanism uses the social network, i.e., the delegations among the agents, to find a collective decision.

We give an example to demonstrate how the inclusion of social networks has made the study of computational social choice more realistic. Control and bribery are well studied in computational social choice [Faliszewski and Rothe, 2016], where control refers to some central actor having the power to alter the structure of an election, and bribery refers to the coercion of some agents to vote or behave in a certain way to impact the outcome. Control in district voting has been extended to social networks by modelling the change of districts where a social network connects voters so that geographical distance can be measured in the model [Bervoets and Merlin, 2012]. As for bribery, one area that has benefitted from incorporating social networks is election campaigning on graphs [Faliszewski et al., 2022]. This connects the study of bribing agents to change their opinions, which are then diffused in the social network and alter the election outcome. Hence, including social networks in the study of bribery has made the research more sophisticated and closer to realistic human interactions.

Delegative Democracy In delegative voting, an agent's vote (or corresponding voting power) can be passed to another voter or candidate. Dodgson [1884] was the first to mention the introduction of delegations to the voting process. However, this was in the context of multi-winner elections, where a candidate could strategically delegate some proportion of their votes to another candidate. The following stream of research relating to incorporating delegations into the voting models was the introduction of proxies. This differs from representative democracy, where representatives are fixed, to a more flexible system where voters can choose their representative and change it at any time [Tullock, 1967, Miller, 1969]. This last point of changing a delegation at any time is now referred to as *instant recall*. It is a central pillar in the justification for liquid democracy from political philosophers and scientists [Blum and Zuber, 2016, Valsangiacomo, 2022]. Although it is not entirely clear when research into introducing delegations into voting models started again after a few instances in the 1960s, the notion of re-delegations was given by Ford [2002], coining the term delegative democracy. The ability to re-delegate a vote is one of the core principles described by Ford. This property of re-delegation is now referred to as transitive delegations in the literature of liquid democracy. Since the creation of liquid democracy, a lot of research stemming from both models of proxy voting and liquid democracy has been undertaken to analyse these models and explore extensions for various purposes. This is especially true since the first implementations of delegations of digital democracy platforms (see Paulin [2020] for an overview).

Much research has been conducted on models of classical liquid democracy[2] and its extensions. A central research area on classical liquid democracy has been study-

[2] By classical liquid democracy, we mean the model where voters can choose to vote directly on a single issue or delegate to a voter, where delegations are transitive. We introduce this term to avoid confusion throughout this thesis, as much of the literature refers to both classical liquid democracy and its various extensions as liquid democracy.

ing delegation cycles; the focus on this problem stemmed from Christoff and Grossi [2017a] to the best of our knowledge. The initial problem of delegation cycles has led to many more models extending classical liquid democracy by introducing more delegation options, either allowing agents submitting a subset of acceptable delegates [Gölz et al., 2018, Dey et al., 2021, Markakis and Papasotiropoulos, 2021], ranked delegations in order of importance [Kotsialou and Riley, 2020, Brill et al., 2022], or delegations using multiple agents' votes [Degrave, 2014]. A question which many researchers have tried to solve is the effectiveness of models of liquid democracy in their ability to uncover some ground truth in comparison to direct democracy [Kahng et al., 2021, Revel et al., 2022b, Caragiannis and Micha, 2019, Armstrong and Larson, 2021, Becker et al., 2021]. This has also been followed up with human experiments analysing how they used delegations, for example in a company [Hardt and Lopes, 2015]. Moreover, some have studied how accurately a group can use delegations to uncover ground truths [Revel et al., 2022a] or if voters rely on delegations over abstentions [Campbell et al., 2022]. Although many studies examining the effectiveness of liquid democracy use direct democracy as a benchmark, Ade et al. [2022] has studied the connection between liquid democracy and representative democracy. Revel et al. [2023] gives a more general model in which many different methods to find representative bodies can be compared. Finally, some are more interested in the rationality of the agents when considering delegations rather than the effectiveness of the model itself [Bloembergen et al., 2019, Noel et al., 2021].

There has been much theoretical work on how models of liquid democracy can be implemented in practice. For example, in examining how vote secrecy can be upheld in a liquid democracy platform [Nejadgholi et al., 2021], or how liquid democracy could be implemented on a blockchain-based voting system [Anwar ul Hassan et al., 2022].

Proxy Voting Although much attention has been focused on liquid democracy, proxy voting has remained a central topic in delegative democracy. From the perspective of political scientists, proxy voting has been characterised to show its benefits with respect to STV [Alger, 2006] and analysed via its axioms to justify its ability to be representative [Green-Armytage, 2015]. Proxy voting has also been studied in its ability to find equilibrium in the models [Cohensius et al., 2017], be manipulated by the proxies and delegators [Bielous and Meir, 2022], quantify how representative a model of proxy voting can be [Anshelevich et al., 2021], as well as the *a priori* voting power of proxies and delegators [Colley et al., 2023a,c]. Moreover, further extensions of proxy voting have also been studied, such as the model given by Abramowitz and Mattei [2019], where voters assign weights to their representatives for each issue.

1.2 Outline of Thesis and Contribution

This thesis contributes to the field of computational social choice by expanding upon models of collective decision-making by incorporating social networks into the mechanism. This breaks down into two research areas, namely models of delegative democracy and models of opinion diffusion. The majority of this thesis is dedicated to the use of delegations in voting models. We want to create and analyse models with expressive and rational delegations. A key contribution of this thesis is extending the notion of a delegation to be more expressive. This means that the voters can have more freedom in how they can give their delegations, meaning that voters can say precisely how their vote should be determined. By rational delegations, we refer to two (connected) ideas from computational social choice. The first is a game-theoretic sense of rationality where delegations are given in such a way that it benefits the delegator. The second is regarding rationality constraints, ensuring that delegations do not go against the sense of rationality provided by the setting, such as giving consistent pairwise comparisons over alternatives or reflecting that voters choose at least one of the options.

Chapter 2 investigates how the classic model of liquid democracy can be extended to give the agents more expressive power, and it does so in two ways. First, agents are allowed to submit multiple ranked delegations. Thus, there are more ways for the delegations to be resolved when their first delegation leads to problems such as delegation cycles. The second extension allows for more complex delegations to be given by the agents, thus allowing a vote of a delegating agent to be determined by a function which uses the votes of a delegator's delegatees. This enables the agents to express how exactly their vote should be determined, with the onus on the model creators to accommodate the delegator's desires. From creating this new framework, we study six algorithmic procedures to resolve the delegations. Each of these output a collection of votes resembling a standard voting profile (i.e., when there is a binary decision, all agents have a vote for or against the issue). Given that there can be obstacles in resolving delegations using every agent's first delegation, these procedures each aim at resolving these issues in a different manner. For example, with respect to minimising the ranked delegation used globally, via an egalitarian approach, or by iteratively adding high-priority votes. The study of these procedures includes their computational complexity, whether they produce Pareto-optimal outcomes and a full picture of how and when they should be chosen for an election.

Chapter 3 addresses a different extension of classical liquid democracy. Instead of studying new forms of delegations, we look at generalising the classical model to contain multiple issues. If all of these issues are independent, this model corresponds to multiple elections using the classical liquid democracy model implemented in parallel. We study liquid democracy on multiple interconnected issues where the resolved delegations must remain consistent with some constraint representing rationality in the given setting. This builds on two existing models of liquid democracy on multiple issues. In the first model, issues represent pairwise comparisons over alternatives; thus, rationality must reflect a consistent preference ordering over the alternatives [Brill and Talmon, 2018]. The second model is a version of liquid democracy where the issues being voted upon are projects to be funded by a budget; here, rationality reflects that the projects the agents accept through voting directly on them or via delegations must respect the budget constraint [Jain et al., 2022]. Instead, we look at a model that generalises both of those previously mentioned in that we consider a set of any binary issues by any constraint (under a few restrictions such that it is of a certain form to keep tractability in the procedures that follow). Hence, we are not restricted to a specific domain. We contribute to the literature by creating and analysing three procedures and rephrasing a fourth procedure from the literature in our model that resolves the delegations. As resolving the delegations straightforwardly could lead to inconsistencies, each procedure ensures that the resulting votes are consistent with the setting's constraint. Many procedures that resolve delegations try to do so optimally for some objective. For example, they try to minimise the number of delegation changes to the profile in resolving delegation consistently. This chapter not only studies procedures looking to minimise the number of changes to regain consistency but also provides procedures which make changes to the delegations and votes with respect to the priorities additionally elicited from the agents. We again study the procedures' computational complexity, showing that our priority procedures are polynomial-time solvable while the optimisation procedures are intractable. We further compare these procedures, showing that the tractable priority procedures do not approximate their intractable counterparts well. Moreover, we compare the procedures based on how well they respect the agents' priorities over their delegations. Lastly, we return to the knapsack voting setting to compare our procedures on budget constraints.

Chapter 4 introduces a model of opinion diffusion via Boolean networks. Here, agents update their (binary) opinion according to a Boolean function that we assume is compactly represented as a propositional formula. This is in contrast to the classical threshold models, where opinions update depending on the proportion of an agent's influencers having differing opinions. Hence, our model extends the functions updating the opinions to be more expressive. Moreover, as Boolean networks are a well-studied mathematical model used extensively in biology, there is a solid foundation for analysing it through the lens of opinion diffusion. Therefore, we focus on extending the known results from the threshold models to our new model of Boolean opinion diffusion, where update functions are agent-specific Boolean functions. Notably, we analyse the computational complexity of deciding if opinions converge from a given initial point, the existence of an asynchronous update that maximises the global agreement among the agents in polynomial time, and we explore connections with delegative voting. Finally, we profit from the wealth of research on Boolean networks in the literature as many results can be translated into our model, therefore expanding the known results of our model.

Chapter 5 returns to classical liquid democracy and highlights two new ways to analyse the model. The first, given in Section 5.2, recreates the well-studied Banzhaf index in liquid democracy. This index measures the likelihood of a voter changing the collective decision by changing their vote, given that the voter is indifferent among the options that they cast in their ballot. Although studied in many different voting contexts, we adapt it to liquid democracy, where an underlying graph represents the possible delegations a voter can choose from. We prove that, in general, computing the index is a #P-hard problem. The second part of the chapter, Section 5.3, introduces a game-theoretical model that analyses liquid democracy from a temporal viewpoint, which has often been overlooked. Here, delegations are not given for single use but are set up to be used multiple times over many different issues to ensure that votes are counted when a voter cannot vote directly. Thus, we introduce a model such that voters have some probability of voting directly, and if not, they rely on their delegation. The main question of our model is finding a stable profile of the voters' delegations so that each voter is happy with their delegation and the expected payoff this delegation will return. We give some initial observations of the model and outline our future research directions.

Chapter 6 concludes this thesis's main streams of research, giving possible future directions for the study of delegations and diffusion for the broader context of digital democracy. Moreover, we introduce two other streams of research of interest that pertain to digital democracy. Concretely, we study the introduction of a new aggregation method that returns a ranking of the divisive issues rather than the collectively agreed upon issues. This new metric has already been implemented on digital democracy platforms, giving participants a well-rounded view of the results. The second area we briefly introduce is weighted judgment aggregation as an overarching framework that can capture many *collective combinatorial optimisation* (CCO) problems, such as participatory budgeting or collective scheduling. This gives weighted judgment aggregation an engineering flavour as we show the CCO domains should be seen as *sister* problems and that many of their aggregation rules are actually well-studied rules from weighted judgment aggregation.

1.3 Published Work

- Chapter 2 is based on:

  Rachael Colley, Umberto Grandi, and Arianna Novaro. Smart voting. In *Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, 2020

  Rachael Colley, Umberto Grandi, and Arianna Novaro. Unravelling multi-agent ranked delegations. *Autonomous Agents and Multi-Agent Systems*, 36 (1):9, 2022

- Chapter 3 is based on:

  Rachael Colley and Umberto Grandi. Preserving consistency in multi-issue liquid democracy. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI)*, 2022a

- Chapter 4 is based on:

  Rachael Colley and Umberto Grandi. The spread of opinions via boolean networks. In *Proceedings of the Multi-Agent Systems: Nineteenth European Conference (EUMAS)*, 2022b

- Chapter 5 is split into two thematic sections. Section 5.2 is based on:

  Rachael Colley, Théo Delemazure, and Hugo Gilbert. Taking delegations seriously: Measuring a priori voting power (Extended abstract). In *Proceedings of the Twenty-Second International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2023c

  Rachael Colley, Théo Delemazure, and Hugo Gilbert. Measuring a priori voting power in liquid democracy. *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)*, 2023b

  Section 5.3 is based on unpublished work.

- Section 6 is the conclusion and it raises other directions of work carried out during my thesis. These are based on:

  Rachael Colley, Umberto Grandi, César Hidalgo, Mariana Macedo, and Carlos Navarrete. Measuring and controlling divisiveness in rank aggregation. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, 2023d

  Linus Boes, Rachael Colley, Umberto Grandi, Jérôme Lang, and Arianna Novaro. Collective discrete optimisation as judgment aggregation. *arXiv preprint arXiv:2112.00574*, 2021

Chapter 2

Multi-Agent Ranked Delegations

2.1 Introduction

This chapter explores the use of complex delegations in voting models to make the voting process more expressive for the voters. This allows voters to decide precisely *how* their vote should be determined via multi-agent ranked delegations. Multi-agent delegations allow a single agent's vote to be determined by the votes of a group of agents via a delegation function. This gives the voters more flexibility as they can rely on a group of trusted voters rather than one individual. We also allow ballots to be more expressive by allowing ranked delegations. Ranked delegations are useful as they will be used when a vote cannot be determined, for example, to avoid delegation cycles. We compare classical liquid democracy and our model in Figure 2.1. On the right-hand side of Figure 2.1, we see a profile with multi-agent ranked delegations. Agent A gives a first delegation representing the majority opinion of the other agents and a backup delegation to B.

We allow delegations to be general functions which take the voter's delegates' votes as the input to the function. For delegation functions to be as general as possible, we model delegation functions as Boolean functions. This provides a very flexible model, giving a general tool to the voters that they can use as much or as little as they desire. We also study different delegation functions, for example,

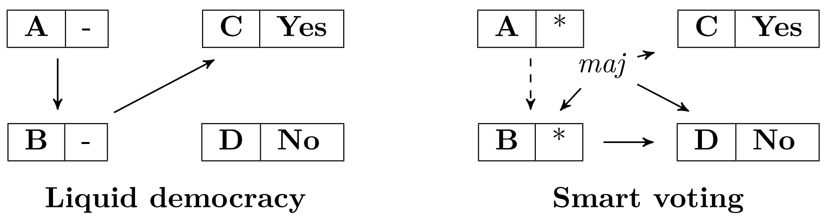


Figure 2.1: A profile of binary votes in liquid democracy on the left: voter A delegates her vote to B, who in turn delegates to C, who casts a direct vote in favour, unlike D, who casts a direct vote against. On the right, a profile of smart ballots: voter A wants her vote to coincide with the majority outcome of B, C, and D's votes, and in case this leads to a delegation cycle, she gives a single delegation to B. Voter B delegates to D, who casts a direct vote against, while C votes in favour. Voters A and B abstain (*) as their final backup option.

delegations decided by a quota rule or a simple delegation to a single agent (i.e., classical liquid democracy). Note that both of these delegation functions can be expressed as Boolean functions.

It should be noted that, even in the simplest model of delegative voting with transitive delegations, it would be unreasonable to ask for delegations to be checked and verified without the aid of computation on a large scale, especially in terms of verifying to the voters how their vote was determined. With the promise of digital democracy platforms, such expressive delegations seem more attainable in practice. One direction for implementing this work comes from the idea that technology, such as blockchain technology, could be the solution to keep elections safe from interference in digital democracy settings. In particular, we called our model *smart voting* [Colley et al., 2020] due to the connection between smart contracts and secure, verifiable voting, as pointed out by Dhillon et al. [2019].

2.1.1 Contribution

This chapter contributes to the creation and analysis of an expressive model of delegative democracy. In Section 2.2, we introduce our model of multi-agent ranked delegations, which we call *smart voting*. Our *smart ballots* allow agents to be more expressive with their delegations than in other models of delegative democracy. Starting from these complex ballots, an *unravelling procedure* returns a standard voting profile from which a collective decision is taken. We introduce six unravelling procedures: MinSum minimises the preference levels globally used for the agents; MinMax minimises the maximum preference level used in the unravelling; the remaining four procedures use a greedy approach to guarantee tractability. We then introduce two restricted languages for smart ballots on binary issues: Bool is our general language, where delegations are arbitrary Boolean functions expressed in complete DNF. A second language of interest is Liquid, which allows for ranked single-agent delegations.

In Section 2.4, we prove our main results: the decision problems needed to compute MinSum and MinMax on the Bool language are NP-complete. However, they are polynomial-time solvable for Liquid ballots. Moreover, we prove that our four greedy unravelling procedures always terminate on valid smart ballots in polynomial time.

In Section 2.5, we compare our unravelling procedures, showing that all six can give outcomes which differ from each other and the procedures from Kotsialou and Riley [2020]. However, the four greedy procedures and MinSum coincide on classical (unranked) liquid democracy ballots. We also study the axioms of cast-participation, guru-participation and a notion of Pareto dominance. Finally, we conclude in Section 2.6.

2.1.2 Related Work

This section complements the introduction and related work outlined in Chapter 1 by giving an overview of the related work extending classical liquid democracy by allowing for multi-agent or ranked delegations.

Multi-Agent Delegations Our first generalisation is to allow voters to express delegations involving many other agents. In the same spirit, the model by Degrave [2014] allows an agent's delegation to be split among possibly many delegates. For example, a voter delegating to three agents could give half their vote to one delegate and a quarter to the other two delegates. Abramowitz and Mattei [2019] give a similar model for proxy voting, where agents assign weights to a fixed set of representatives for each issue: agents can thus choose to spread their vote over many representatives.

In liquid democracy with multiple issues, some models allow delegations to different agents. However, these models focus on one single-agent delegation per issue. The first was on pairwise liquid democracy on ordinal elections [Brill and Talmon, 2018], where ballots are partial rankings over alternatives which can be completed by delegating the decision on distinct pairs of alternatives to a delegate. Note that an agent can delegate to multiple delegates, each on a different pair of alternatives. There have been other models of fine-grained, issue-wise delegations to other voting domains, such as liquid knapsack voting [Jain et al., 2022], and on binary issues connected by Boolean formulas (see Chapter 3 based on Colley and Grandi [2022a]).

Another model in which multiple agent delegations have been studied on a single issue is from Gölz et al. [2018]. They address the problem of a few agents holding a lot of power by studying the fluid mechanics of liquid democracy. Their model balances influence in a delegation graph like liquid in a vessel. Here, agents submit a list of multiple acceptable delegates, and the mechanism balances the voting power of the direct voters.

Ranked Delegations The second generalisation is to allow for ranked delegations to avoid cycles (as in our case) or to avoid the delegation to an abstaining agent. Ford [2002] introduces the notion of *delegation chains*, which we call *ranked delegations*, where an agent submits an ordering of many delegates, from the most trusted to one who still represents the agent, but less than the delegates coming before.

The model proposed by Kotsialou and Riley [2020] includes ranked delegations that avoid delegating to an abstaining agent and delegation cycles. They propose two procedures, breadth-first and depth-first, which find an outcome similar to what we suggest, yet they do not allow delegations to abstaining agents. Brill et al. [2022] follow a similar approach and provide an axiomatic analysis of the delegation rules that return direct votes from ranked delegations. They also perform experiments on synthetic and real-world data.

Kahng et al. [2018, 2021] suggest an alternative method of removing delegation cycles by assigning a competency level to agents and forbidding them to delegate to someone with a lower competency level than themselves. Boldi et al. [2011] propose *viscous democracy*, a model of liquid democracy that can be extended to include ranked delegations, where they use a dampening factor to ensure shorter chains of delegations. Behrens and Swierczek [2015] propose a model of ranked liquid democracy and seven desirable properties for a liquid democracy system, showing that no system can satisfy all of them.

Other lines of research have used the idea of ranks over delegations without incorporating them into the mechanism. For example, Escoffier et al. [2019] introduce a model with iterative delegations, where voters update their ballots iteratively, as a best response dynamic, to find a set of delegations where no voter wants to deviate from their ballot to receive a better ultimate delegate. Given that every voter has a preference ordering over their ultimate delegates, they showed that, in general, a stable Nash equilibrium does not necessarily exist. Yet, in an extension of their work, Escoffier et al. [2020] showed that equilibria exist under certain conditions over the voters' preferences.

2.2 Smart Voting

We formally introduce our delegative voting model, where delegations can be ranked and involve multiple agents. Before doing so, it is worth outlining the four main stages of a smart voting election.

1. Each voter creates their smart ballot, which could be restricted to be of a certain form, and sends it to the central authority;

2. The central authority then checks if the received ballots are valid and abide by the restrictions;

3. The possibly complex ballots involving multi-agent ranked delegations then need to be turned in the votes of the agents, i.e., by resolving or *unravelling* the delegations and the potential cycles;

4. The resolved votes (all in the issue's domain of alternatives) are then aggregated to find a collective decision.

Note that a decentralised system could replace the central authority when the election is held via smart contracts. In this case, steps 2, 3 and 4 are carried out by each member of the decentralised system, with an additional verification step.

This work studies steps 1 to 3 of such an election, as the fourth step can be done with the desired aggregation rule.

We let $\mathcal{N}$ be a set of n agents, and $\mathcal{I}$ is the set of issues which will be voted on. Each issue has a set of possible choices of its outcome, called alternatives, and we denote the domain of the alternatives for an issue $i \in \mathcal{I}$ as $\mathcal{D}(i)$. We mainly consider binary issues, either with or without abstentions, unless stated otherwise, $\mathcal{D}(i) = \{0, 1\}$ or $\mathcal{D}(i) = \{0, 1, *\}$. We consider the issues in $\mathcal{I}$ to be independent of each other, and therefore, we can consider each issue in isolation. Thus, we study the model as if there were just a single issue. Collective decisions on multiple issues would be determined in parallel.

2.2.1 Smart Ballots

We next introduce smart ballots into the model. These ballots differ from classical liquid democracy ballots in that they allow for ranked delegations and more complex delegation functions possibly involving multiple agents. In the former, an ordering of priority is given over the delegations and a lower priority delegation can be used when the higher prioritised delegations cannot be determined, for example, because of a delegation cycle. In the latter, delegations are no longer thought of as passing a vote transitively between agents but rather allowing the agent's vote to be determined by the votes of their delegates with respect to their chosen function. Each agent $a \in \mathcal{N}$ expresses how they want their vote to be determined on an issue $i \in \mathcal{I}$ through a smart ballot B_{ai}, which is defined as follows:

Definition 2.1 (Smart ballot)**.** A *smart ballot* of agent a on an issue $i \in \mathcal{I}$ is an ordering $((S^1, F^1) > \cdots > (S^k, F^k) > x)$ where $k \geq 0$ and for $h \leq k$ we have that $S^h \subseteq \mathcal{N}$ is a set of agents, $F^h : \mathcal{D}(i)^{S^h} \to \mathcal{D}(i)$ is a resolute aggregation function and $x \in \mathcal{D}(i)$ is an alternative.

As we focus on a single issue $i \in \mathcal{I}$, we drop the index i from our notation due to the independence of issues in this model. Thus, $\mathcal{D}(i)$ becomes $\mathcal{D}$ and B_{ai} becomes B_a.

We see that the smart ballots, as defined in Definition 2.1, give voters the ability to prioritise their desired delegations, i.e., the k domain-function pairs (S, F), which are followed by a direct vote in the domain of alternatives for the issue $x \in \mathcal{D}$. The addition of a direct vote as the final entry in the priority order ensures that delegation cycles can always be resolved. This may seem like a large ask upon the agent, yet this could be the status quo option in a binary vote (if one exists) or an abstention when allowed. Observe that when $k = 0$, the agent has decided to vote directly on the issue without delegating; we will refer to these agents as direct voters.

Delegation functions F can represent many ways of delegating, such as: the identity function representing a simple delegation; an aggregation rule such as quota rules; or a Boolean function expressed as a formula. When F is a Boolean function, it is built from the standard logical connectives (i.e., $\neg, \vee, \wedge, \cdots$) and the variables are the agents, i.e., $\{a \mid a \in \mathcal{N}\}$ which represent the vote determined for that agent. Note that when F represents a Boolean formula φ from a pair (S, F) then S must be such that the variables of φ are in S, i.e., $Var(\varphi) \subseteq S$.

When computing a function F on an incomplete input, we use the notion of *necessary winners* [Konczak and Lang, 2005]. For example, consider an agent $a \in \mathcal{N}$

having a delegation $Maj(b, c, d)$—i.e., the majority over the votes of b, c and d—on a binary issue for which b and c have a direct vote for 1, while there is no vote of d yet. We can still compute $Maj(b, c, d)$ without the vote of d since there is already a majority for 1.

We extend Definition 2.1 to give *valid* smart ballots, which now have some additional desired properties.

Definition 2.2 (Valid smart ballot)**.** A *valid smart ballot* of agent a is a smart ballot B_a such that for all $1 \leq s \neq t \leq k$, we have that (i) if $S^s \cap S^t \neq \emptyset$ then F^s is not equivalent to F^t, and (ii) $a \notin S^s$.

Condition *(i)* ensures that agents cannot manipulate by submitting equivalent formulas multiple times. We want to avoid this in order to keep the priority level used between the agents somewhat uniform. For example, an agent cannot submit one delegation function of a and another as $a \wedge a$, as these two functions are logically equivalent. The second condition *(ii)* is that an agent cannot include themself in the domain of the function, thus not allowing immediate delegation cycles. We now give examples of possible smart ballots to demonstrate their range of forms:

Example 2.1. Six agents $\mathcal{N} = \{a, b, c, d, e, f\}$ face the decision of whether to order dinner from one of two new restaurants. Let 1 denote the choice of the first restaurant and 0 denote the choice of the second one, and assume that agents can also abstain, i.e., $\mathcal{D} = \{1, 0, *\}$.

We now present some valid smart ballots that agent a could submit:

(i) $B_a = (1)$

This smart ballot represents the direct vote of a for the first restaurant.

(ii) $B_a = ((\{b, c, d, e, f\}, RMaj) > 0)$

Here, agent a wants their vote to be the *relative majority* $RMaj$ (i.e., plurality between 0 and 1, with $*$ in case of a tie) of the choices of the agents b, c, d, e, f. They will vote for the second restaurant if this first delegation of a cannot be determined.

(iii) $B_a = ((\{d\}, id) > (\{e\}, id) > *)$

Here, a's first preference is to delegate to agent d (id indicates the identity function); if this causes a delegation cycle, then a chooses to delegate to e; and if this also causes a cycle, then a abstains from the vote.

Suppose now that we have a binary issue[1] with domain $\mathcal{D} = \{0, 1\}$ where 1 represents ordering takeaway while 0 represents cooking at home.

[1]Note that in order to use Boolean functions to express a delegation, the domain of the alternatives for the issue must be a Boolean algebra. We restrict this to a two-element Boolean algebra, namely $\{0, 1\}$.

(iv) $B_a = ((\{b, f\}, b \vee f) > (\{b, c, e\}, (c \wedge b) \vee (\neg e \wedge b)) > 1)$

In this case, a's first choice is to delegate using the Boolean function $b \vee f$, i.e., a will vote for takeaway if either b or f wants to; if this creates a delegation cycle, then a will vote to try the takeaway if both b and c also want to, or b wants to while e prefers to cook at home. If this still causes a cycle, a votes to try the new takeaway restaurant.

(v) $B_a = ((\{b, c, f\}, \mathit{Maj}) > (\{b, c, e\}, (c \wedge b) \vee (\neg e \wedge b)) > 1)$

This smart ballot differs from *(iv)* only in that a's first preference is to have their vote coincide with the majority outcome from agents b, c and f.

One can easily check that all the ballots given in Example 2.1 are valid as per Definition 2.2. $\triangle$

Each linear order of delegations (plus the backup vote) in a smart ballot indicates a preference over possible delegations. We write B_a^h to indicate the h^{th} *preference level* given by agent a in their smart ballot B_a. Hence, we have $B_a^h = (S_a^h, F_a^h)$ when the h^{th} preference level is a delegation, or $B_a^h = x$ with $x \in \mathcal{D}$ when the h^{th} preference level is a direct vote. In Example 2.1, e.g., $B_a^2 = (\{e\}, id)$ for ballot *(iii)*. We refer to a collection of smart ballots, one per agent, as a smart profile, denoted with $\boldsymbol{B}$. A *valid (smart) profile* is a smart profile where each smart ballot is valid, according to Definition 2.2.

2.2.2 Language Restrictions of Smart Ballots

Given our general definition of valid smart ballots, as given by Definition 2.2, we may still want to restrict the language of delegations further for a given election. For example, the community may want the election only to contain simple delegations or only a maximum number of levels in the priority. Thus, we let $\mathcal{L}$ be a language to restrict the form of the delegations, saying that a delegation function F belongs to a language $\mathcal{L}$ if it abides by its restrictions. Let $\mathcal{L}[k]$ denote the language restricting the delegations to belong to $\mathcal{L}$ as well as the additional constraint that ballots can have at most k delegations in their ordering. Let $\mathcal{L}_*$ denote a language $\mathcal{L}$ restricting the final direct vote to be an abstention $*$ when an agent has delegations, as in ballot *(iii)* in Example 2.1.

The main language we use throughout the chapter is Bool, which restricts delegations to be expressed as Boolean functions. To do this, we introduce some logical notions. First, restrict the formulas to be propositional formulas on a binary domain; thus, abstentions are not allowed in this domain.[2] Second, we insist that the formulas are *contingent*—i.e., neither a tautology nor a contradiction— to avoid a direct vote in disguise for the issue (in the case of a tautology) or against the issue (in the case of a contradiction), as they would always evaluate to true (respectively,

[2]Note that an extension of Bool could be introduced on the domain $\{0, 1, *\}$ in which evaluating the delegation functions under three values has been specified properly, such that the functions are well-defined and resolute.

to false). Third, the formulas must be expressed in *disjunctive normal form* (DNF), i.e., they are written as a disjunction of *cubes*, where a cube is a conjunction of *literals* (and a literal is a variable or its negation). Finally, a cube C is an *implicant* of formula φ if $C \vDash \varphi$, and C a *prime implicant* of φ if C is an implicant of φ and for all other $C' \vDash \varphi$, we have that $C' \nvDash C$. Intuitively, prime implicants are the minimal partial assignments to make a formula true. A *complete* DNF uniquely represents a DNF listing all of its prime implicants.

Although this list of restrictions may seem hard to fulfil for human users, we envision that the smart voting model would use a digital democracy platform. In doing so, practitioners could create a user-friendly interface to help smart ballots in a complex form be created. Besides, digital smart voting platforms could have a pre-processing step which takes the ballots containing Boolean functions and turns them into formulas in complete DNF. This could use well-known techniques such as the *consensus method* or *variable depletion* (see the textbook by Crama and Hammer [2011] for further details).

We now give the formal definition of Bool, the delegation language using Boolean functions.

Definition 2.3 (Bool)**.** A smart ballot B_a for an agent a on a binary issue is in the language Bool if every F_a^h in B_a is a contingent propositional formula in complete DNF.

Note that Bool is very expressive and can encapsulate many other types of delegations we are interested in. For example, simple delegations to a single agent can be represented as an atomic variable of their delegate's name. In Example 2.1, ballot (v) does not belong to language Bool as $B_a^1 = \mathit{Maj}$ is not a Boolean formula; however, ballot (iv) belongs to Bool. Note that the formula $(b \wedge c) \vee (b \wedge \neg c) \vee f$, which is equivalent to the formula used at the first preference level of *(iv)*, would not be in Bool as it is not complete. Moreover, smart ballots (i) and (iv) in Example 2.1 belong to the language Bool[2].

For the language Bool, we often write $\varphi_a^{\texttt{lev}}$ instead of $F_a^{\texttt{lev}}$.

We insist that Boolean delegation functions are in complete DNF because of the following property: checking if there exists a necessary winner of a formula in complete DNF can be done in polynomial time.

Proposition 2.1. *Deciding if a formula in a Bool ballot has a necessary winner on a partial truth assignment can be done in polynomial time.*

Proof. Observe that the necessary winner for a formula under a partial truth assignment being 1 (respectively, 0) means that the formula is true (respectively, false) in all possible extensions of the partial assignment. We first need to prove the following two claims under a partial truth assignment:

1. The necessary winner of a complete DNF formula is 1 if and only if every literal of at least one cube of the formula is true.

2. The necessary winner of a complete DNF formula is 0 if and only if every cube of the formula is made false by at least one literal.

Both of these claims can be verified in polynomial time by reading the formula and the partial truth assignment. Thus, if they are true, a necessary winner can be found in polynomial time.

For the right-to-left direction of claim (1), assume that one cube of the formula is true. As the formula is a complete DNF, each cube represents one of its prime implicants. By definition, if a prime implicant is made true, so is the formula.

For the left-to-right direction of claim (1), assume that the complete DNF formula φ is made true by some partial truth assignment X. We create a cube C from the partial assignment, where if a variable x is true (respectively, false) in X, then x (respectively, $\neg x$) is a literal in C. As C is built from a partial truth assignment making φ true, we have that $C \vDash \varphi$ and thus C is an implicant of φ. Then, either C is a prime implicant of φ or there exists a prime implicant C' of φ, such that $C' \vDash C$, where C' contains a subset of literals in C. As φ is a complete DNF, in either case, there is a cube of φ made true by X (i.e., either C or C').

For the right-to-left direction of claim (2), if all cubes in the formula are made false, then the formula is also necessarily false (i.e., the necessary winner is 0).

For the left-to-right direction of claim (2), assume that a complete DNF φ evaluates to false under a partial truth assignment X. Yet, assume for a contradiction that there exists a cube C of φ that does not evaluate to false under X. As C is not false, then either C is true under X (yielding a contradiction, as φ would be true), or there are some variables $v \in Var(C)$ without a truth value in X and the remaining literals are made true. We can then extend X for each such $v \in Var(C)$ such that the literal of v in C is made true. As φ is a complete DNF, the cube C would be true—as no cube can contain contradictions (e.g., x and $\neg x$). Thus, the formula φ would be true, and we have reached a contradiction. Finally, checking that each literal of at least one cube is true (or that every cube is made false by at least one literal) can be done by simply inspecting the formula with the partial truth assignment and thus in polynomial time. □

A further advantage of having delegations expressed in complete DNF is that we can check whether a ballot is valid in polynomial time, as a tautology in complete DNF is $\top$, a contradiction is $\bot$, and to check if two complete DNF formulas are equivalent it suffices to see if the lists of their prime implicants are the same. In previous iterations of our work on smart voting [Colley et al., 2020], we did not include this further stipulation that Bool ballots must be in complete DNF. Instead, the validity of the ballots was studied when delegations were contingent Boolean formulas in DNF. We prove that checking validity must be NP-complete when there are at most k delegations on ballots with delegations built from Boolean formulas in DNF.

Theorem 2.1. *Checking if a ballot restricted to be in $\mathcal{L}[k]$ is valid is an NP-complete problem, for $k \geq 1$ and $\mathcal{L}[k]$ allows for at most k delegations which are* *contingent Boolean formulas in DNF on a binary domain.*

Proof. For membership, observe that for a ballot B to be valid with respect to $\mathcal{L}$, the following properties need to be verified (for $1 \leq h, \ell \leq k$), that can either be checked in polynomial time by reading B or require a (polynomial) certificate. The properties that can be checked in polynomial time are:

- there are less than k delegations in the ballot of the form (S_a^h, φ_a^h);
- there is one direct vote in $\{0,1\}$ as final preference;
- each φ_a^h is in DNF;
- each S_a^h is such that $a \notin S_a^h$;
- and each φ_a^h is such that $Var(\varphi_a^h) \subseteq S_a^h \subseteq \mathcal{N}$.

For the final three properties, we have to guess certificates for:

- at most k certificates to check that each φ_a^h is not a tautology, we do this by checking that $\neg\varphi_a^h$ is satisfiable;
- at most k certificate to check that each φ_a^h is not a contradiction, i.e., by checking that φ_a^h is satisfiable;
- at most $\frac{k}{2}(k-1)$ certificates to check that for all φ_a^h and φ_a^ℓ such that $h \neq \ell$, φ_a^h and φ_a^ℓ are not logically equivalent (i.e., $\neg(\varphi_a^h \leftrightarrow \varphi_a^\ell)$ is satisfiable).

All this requires at most $\frac{k}{2}(k+3)$ certificates for constant k. Thus, the problem of checking if a ballot is valid for $\mathcal{L}[k]$ is in NP.

For hardness, we reduce from the NP-complete problem DNF-FALSIFIABLE,[3] whose input is a formula φ in DNF. We create an instance of our problem. Let $\mathcal{N} = Var(\varphi) \cup \{a, b\}$, for fresh variables a and b, $\mathcal{D} = \{0,1\}$ and $B_a = ((\mathcal{N} \backslash \{a\}, \varphi \vee b) > 1)$. We now show that DNF-FALSIFIABLE has a positive answer if and only if our ballot is valid for the language $\mathcal{L}[k]$.

Assume that φ is falsifiable. By the construction of B_a, we only need to check that $\varphi \vee b$ is neither a contradiction nor a tautology: this follows from φ being falsifiable and b being a fresh variable. That is, $\varphi \vee b$ can be made false when b is false and by the truth assignment that falsifies φ (which exists by assumption). $\varphi \vee b$ can be made true when b is true. Therefore, B_a is a valid ballot for $\mathcal{L}[k]$. Next, assume that φ is not falsifiable, i.e., φ is a tautology. Thus, the delegation $\varphi \vee b$ is also a tautology; hence, B_a is not valid. Therefore, the problem of checking if a ballot is valid for language $\mathcal{L}[k]$ (where each of the delegations is a contingent Boolean formula in DNF) is NP-complete. □

[3]Since SAT-CNF is NP-hard, by the duality principle DNF-FALSIFIABLE is also NP-hard; for membership in NP it suffices to verify a falsifying truth assignment in polynomial time.

The previous result shows that by insisting that delegations are in complete DNF, the complexity of checking for necessary winners and ballot validity has been reduced to done in polynomial time. In the former case, this will become particularly important, as in the next section, we will start defining our unravelling procedures which resolve delegations. Here, checking for necessary winners in polynomial time will become useful.

We next introduce a language to restrict the ballots of the agents to be in *ranked* liquid democracy, where delegations must be to a single agent.

Definition 2.4 (Liquid)**.** Smart ballot B_a for agent a belongs to Liquid if every delegating B_a^h is of the form $(\{b\}, id)$ for $b \in \mathcal{N} \setminus \{a\}$ and id the identity function.

In many models of ranked liquid democracy, such as in the models suggested by Brill et al. [2022] and Kotsialou and Riley [2020], the final backup vote must be an abstention. We denote this language as Liquid$_*$.

For instance, in Example 2.1, ballot (*iii*) belongs to the language Liquid$_*$ as well as Liquid[2]. Note that checking if a ballot is valid for Liquid is a tractable problem as it suffices to check that all delegation functions use id and that no one delegates to themselves or the same agent multiple times.

2.3 Unravelling Procedures

From the definitions of the acceptable ballots in the model, we define the third step of the election, taking the complex ballots and turning them into a profile of votes in the domain of the issue $\mathcal{D}^n$. We do this via an unravelling procedure.

Definition 2.5 (Unravelling procedure)**.** An *unravelling procedure* $\mathcal{U}$ for the agents in $\mathcal{N}$ is any computable function

$$\mathcal{U} : (B_1 \times \cdots \times B_n) \to \mathcal{D}^n.$$

Thus, an unravelling procedure $\mathcal{U}$ takes a smart profile $\boldsymbol{B}$ and returns a voting profile in $\mathcal{D}^n$. When $\mathcal{U}$ and $\boldsymbol{B}$ are clear from context, we often write just X to denote an outcome of an unravelling procedure: i.e., $X \in \mathcal{U}(\boldsymbol{B})$ for $X \in \mathcal{D}^n$.

After an unravelling procedure returns a profile of direct votes, the agents may want to know how their smart ballot was unravelled: i.e., which preference level of their ballot was used to compute their direct vote. For this purpose, we introduce the notion of a *certificate.*

Definition 2.6 (Certificate)**.** A certificate $\mathbf{c} \in \mathbb{N}^n$ for profile $\boldsymbol{B}$ is a vector where, for all $a \in \mathcal{N}$ such that $B_a = (B_a^1 > \cdots > B_a^{k_a})$, the entry $\mathbf{c}_a \in [1, k_a]$ corresponds to a preference level for agent a.

Within the class of all possible certificates of Definition 2.6, we are interested in those that satisfy the following property: a certificate is *consistent* if there is an ordering of the agents such that each agent's vote can be determined using the preference level in the certificate, given the votes of the agents that come prior in the order.[4]

Definition 2.7 (Consistent certificate)**.** Given a profile $\boldsymbol{B}$ of valid ballots, a certificate $\mathbf{c}$ is *consistent* if there exists an ordering $\boldsymbol{\sigma} : \mathcal{N} \to \mathcal{N}$ of the agents which, starting from vector $X^0 = \{\Delta\}^n$ with placeholder values Δ for all agents, iteratively constructs an outcome vector of direct votes $X \in \mathcal{D}^n$ as follows, for $\sigma(a) = z \in [1, n]$:

$$X_a^z = \begin{cases} B_a^{\mathbf{c}_a}, & \text{if } B_a^{\mathbf{c}_a} \in \mathcal{D} \\ F_a^{\mathbf{c}_a}(X^{z-1} \restriction_{S_a^{\mathbf{c}_a}}), & \text{otherwise} \end{cases}$$

where X_a represents a's entry in X, we let $X \restriction_S = \Pi_{s \in S} X_s$, and note that $F_a^{\mathbf{c}_a}$ returns the necessary winner when the input is incomplete when possible.

We let $\mathcal{C}(\boldsymbol{B})$ be the set of all consistent certificates of a profile $\boldsymbol{B}$, and the a^{th} entry of $\mathbf{c}$ corresponds to $B_a^{\mathbf{c}_a}$ being used by the unravelling procedure. Moreover, when certificates can determine outcomes of unravelling procedures, we use $\mathcal{C}_{\mathcal{U}}(\boldsymbol{B})$ to denote all consistent certificates given by the unravelling procedure $\mathcal{U}$. We show next that each consistent certificate $\mathbf{c}$ has a unique corresponding outcome vector $X_{\mathbf{c}}$ of direct votes.

Note that checking if a certificate is consistent can be done in polynomial time (the proof of which can be found in the proof of Lemma 2.1).

Proposition 2.2. *If a consistent certificate $\mathbf{c}$ can be given by two orderings σ and σ' of the agents (as per Definition 2.7), then the orderings yield the same outcome $X_{\mathbf{c}} \in \mathcal{D}^n$.*

Proof. Consider an arbitrary profile $\boldsymbol{B}$ and a consistent certificate $\mathbf{c} \in \mathcal{C}(\boldsymbol{B})$. Assume for a contradiction that $\mathbf{c}$ can yield two distinct vectors of direct votes $X \neq X'$, which are given by two orderings σ and σ' of $\mathcal{N}$, respectively. To reach a contradiction, we show by induction on the ordering σ that for each agent $a \in \mathcal{N}$, we have $X_a = X'_a$.

For the base case, consider agent $a \in \mathcal{N}$ such that $\sigma(a) = 1$. As a's vote was added to $X_{\mathbf{c}}$ without any other vote, $\mathbf{c}_a$ must refer to a direct vote. Therefore, the direct vote of a will be added to X and X' (although it may be that $\sigma'(a) \neq 1$). Thus, $X_a = X'_a$.

Our inductive hypothesis assumes that for some k, for all agents $b \in \mathcal{N}$ (where $\sigma(b) \leq k$), it is the case that $X_b = X'_b$. We show that for agent d such that $\sigma(d) = k+1$, we have $X_d = X'_d$. In case $B_d^{\mathbf{c}_d}$ is a direct vote, the same reasoning applies as in the base case. Else, by definition, we have a necessary winner for $F_d^{\mathbf{c}_d}(X \restriction_{S_d^{\mathbf{c}_d}}) = X_d$. If $X_d \neq X'_d$, then $F_d^{\mathbf{c}_d}(X \restriction_{S_d^{\mathbf{c}_d}}) \neq F_d^{\mathbf{c}_d}(X' \restriction_{S_d^{\mathbf{c}_d}})$ and $X \restriction_{S_d^{\mathbf{c}_d}} \neq X' \restriction_{S_d^{\mathbf{c}_d}}$. Hence, an entry differs in the two vectors, which contradicts our inductive hypothesis. Then, $X_d = X'_d$. As $X_a = X'_a$ for all $a \in \mathcal{N}$, we have that $X = X'$. Hence, a consistent certificate $\mathbf{c}$ gives a unique outcome $X_{\mathbf{c}}$. □

[4]When ballots are restricted to single-agent delegations, an unravelling procedure is a confluent sequence rule, defined by Brill et al. [2022], if it always gives outcomes with consistent certificates.

Note that a certificate leads to a *delegation cycle* exactly when it is not consistent, i.e., there is a cycle of dependencies when determining the votes of the agents within the delegation cycle, and therefore, there is no possible ordering σ in which the agents' votes can be added.

Finally, we define the *rank* of a certificate $\mathbf{c}$ as the sum of the preference levels used. Given profile $\boldsymbol{B}$, the $\texttt{rank}$ of a certificate $\mathbf{c} \in \mathcal{C}(\boldsymbol{B})$ is $\texttt{rank}(\mathbf{c}) := \sum_{a \in \mathcal{N}} \mathbf{c}_a$. The minimum possible value of $\texttt{rank}$ for an unravelling is n, i.e., when all the agents have their first preference level used in the unravelling. Thus, if a profile contains a delegation cycle at the first preference level, the certificate $\mathbf{c} = (1, \cdots, 1)$ is not consistent.

2.3.1 Optimal Unravellings

We first inspect our optimal unravelling procedures, which find consistent certificates $\mathbf{c}$ that minimise some criteria. The first procedure tries to find a certificate that minimises the total sum of the entries $\texttt{rank}(\mathbf{c})$. The second follows an egalitarian approach and tries to find a certificate whose maximal entry is minimal.

Our first procedure, MinSum, is *optimal* with respect to the $\texttt{rank}$: it returns all outcome vectors which can be obtained by a consistent certificate minimising the sum of preference levels used for the agents.

Definition 2.8 (MinSum)**.** For a given profile $\boldsymbol{B}$, the MinSum unravelling procedure is defined as:

$$\textsc{MinSum}(\boldsymbol{B}) := \{X_{\mathbf{c}} \mid \mathbf{c} \in \operatorname*{arg\,min}_{\mathbf{c} \in \mathcal{C}(\boldsymbol{B})} \texttt{rank}(\mathbf{c})\}.$$

Hence, MinSum returns all vectors of direct votes $X_{\mathbf{c}}$ whose consistent certificate $\mathbf{c}$ minimises the value of $\texttt{rank}(\mathbf{c})$. Intuitively, more trusted agents are being chosen as delegates by minimising the agents' preference levels used globally. Next, we give examples of consistent certificates and the outcomes of the MinSum procedure.

Example 2.2. Consider a binary issue with domain $\mathcal{D} = \{0, 1\}$ and five agents $\mathcal{N} = \{a, b, c, d, e\}$, whose ballots form the profile $\boldsymbol{B}$, shown schematically in Figure 2.2 as both a table and a delegation graph. Note that there is a cycle in the delegation graph when considering the first preference delegations (the solid lines or B^1_x). We see that agent a needs the vote of c to compute their own vote (given that b votes in favour), agent c delegates to d, agent d delegates to e, and agent e delegates to a. Thus, the certificate $\mathbf{c} = (1, 1, 1, 1, 1)$ leads to a delegation cycle and $\mathbf{c} \notin \mathcal{C}(\boldsymbol{B})$. As a result, a consistent certificate for this profile will have a $\texttt{rank}$ of at least 6.

Consider the certificate $\mathbf{c}' = (1, 1, 2, 1, 1)$, where only c has their second preference used: $\mathbf{c}'$ is consistent, as shown by the ordering $\sigma = (b, c, a, e, d)$. As $\texttt{rank}(1, 1, 2, 1, 1) = 6$, the corresponding outcome $X_{\mathbf{c}'} = (0, 1, 0, 0, 0)$ is an outcome of MinSum($\boldsymbol{B}$). The consistent certificate $\mathbf{c}'' = (1, 1, 1, 2, 1)$ gives $X_{\mathbf{c}''} = (1, 1, 1, 1, 1)$ and as $\texttt{rank}(\mathbf{c}'') = 6$, it also is an outcome of MinSum($\boldsymbol{B}$). Since there

	B_x^1	B_x^2	B_x^3
a	$(\{b,c\}, b \wedge c)$	$(\{d\}, d)$	1
b	1	-	-
c	$(\{d\}, d)$	0	-
d	$(\{e\}, e)$	1	-
e	$(\{a\}, a)$	$(\{b\}, b)$	0

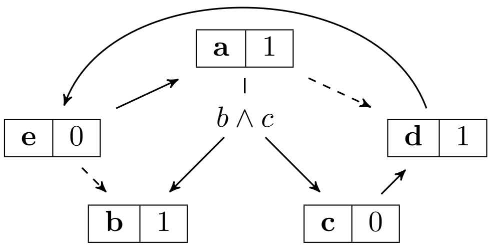


Figure 2.2: A table and graphical depiction of the $\boldsymbol{B}$ from Example 2.2. On the left-hand side, we have a table where each row represents the ballot for each of the agents $\mathcal{N} = \{a, b, c, d, e\}$, while the columns separate the different preference levels of the agents' ballots. On the right-hand side, we display the same profile as a graph, where the solid lines represent first preferences and the dashed lines represent second preferences. The final preference for a direct vote is next to the agent's name.

can be multiple certificates minimising the total rank (yielding distinct vectors of direct votes X), we see that the MinSum unravelling procedure is not resolute. △

While MinSum maximises the *global* satisfaction of the agents, there can be a large disparity in the selected delegation levels used from the perspective of the individual voters. Our second optimal procedure MinMax is motivated by an *egalitarian* approach, finding outcomes whose certificate minimises the maximum delegation level used among all agents.

Definition 2.9 (MinMax)**.** Given profile $\boldsymbol{B}$, the MinMax unravelling procedure returns the following vectors of direct votes:

$$\textsc{MinMax}(\boldsymbol{B}) := \{X_{\mathbf{c}} \mid \mathbf{c} \in \operatorname*{arg\,min}_{\mathbf{c} \in \mathcal{C}(\boldsymbol{B})} \max(\mathbf{c})\}.$$

Example 2.3. Consider a binary issue and 26 agents $\mathcal{N} = \{a, \ldots, z\}$. Let the profile $\boldsymbol{B}$ be such that the smart ballot of agent a is:

$$B_a = (\mathcal{N}\backslash\{a\}, \bigvee_{\substack{x\in\\ \mathcal{N}\backslash\{a\}}} x) > (\mathcal{N}\backslash\{a,b\}, \bigvee_{\substack{x\in\\ \mathcal{N}\backslash\{a,b\}}} x) > (\mathcal{N}\backslash\{a,b,c\}, \bigvee_{\substack{x\in\\ \mathcal{N}\backslash\{a,b,c\}}} x) > 1,$$

and for each agent $x \in \mathcal{N} \setminus \{a\}$ let $B_x = ((\{a\}, a) > 0)$ be their smart ballot.

MinMax($\boldsymbol{B}$) returns the outcomes obtained via the following three certificates $\mathbf{c} = (1, 2, \ldots, 2)$, $\mathbf{c}' = (2, \ldots, 2)$, $\mathbf{c}'' = (2, 1, 2, \ldots, 2)$. Observe that $\max(\mathbf{c}) = \max(\mathbf{c}') = \max(\mathbf{c}'') = 2$, even though $\texttt{rank}(\mathbf{c}) = \texttt{rank}(\mathbf{c}'') = 51$ and $\texttt{rank}(\mathbf{c}') = 52$. The outcome of MinSum($\boldsymbol{B}$) has certificate $\mathbf{c}''' = (4, 1, \ldots, 1)$ and $\texttt{rank}(\mathbf{c}''') = 29$; however, this is not an outcome of MinMax, since $\max(\mathbf{c}''') = 4$. △

Algorithm 1 General unravelling procedure Unravel

```
1: Input: $\boldsymbol{B}$
2: $X := (\Delta, \ldots, \Delta)$          ▷ vector for direct votes initialised with placeholders $\Delta$
3: while $X \notin \mathcal{D}^n$ do
4:     lev := 1                               ▷ reset preference level counter lev to 1
5:     $Y := X$                                ▷ store a copy of $X$ to compute changes
6:     while $X = Y$ do
7:         procedure Update(#) with $\# \in \{\mathbf{U}, \mathbf{RU}, \mathbf{DU}, \mathbf{DRU}\}$
8:         lev := lev + 1
9: return $X$                                  ▷ output a vector of direct votes when complete
```

A disadvantage of MinMax is that it may return a large number of tied outcomes for some profiles, as we shall see in the profile given in Table 2.2.

2.3.2 Greedy Unravellings

In Section 2.4, we prove that computing an outcome of MinSum and MinMax is generally not computationally tractable. This motivates us to introduce four unravelling procedures with a greedy approach. They each find consistent certificates by using the lowest possible delegation level in the ballots while keeping the process tractable.

Algorithm 1 outlines our *general unravelling procedure* Unravel. The input is a smart profile $\boldsymbol{B}$, and the procedure initialises a vector X with placeholders Δ for each agent $a \in \mathcal{N}$. The outcome X is returned when each agent has a vote in $\mathcal{D}$, i.e., $X \in \mathcal{D}^n$. A counter `lev` is always reset to 1 to return to the first preference level of the agents. An additional vector Y is used to help with intermediate computations.

In line 7 a subroutine using an *update procedure* is executed.[5] Given a partial profile of direct votes and placeholders Δ, as well as a preference level `lev`, the Update procedure searches for a direct vote or a vote that can be computed via necessary winners (depending on which Update is used) at the `lev`th preference level in the profile. If this is not possible, Unravel moves to level `lev` + 1 (line 8).

	no random voter selection	*random voter selection*
no direct vote priority	**U**	**RU**
direct vote priority	**DU**	**DRU**

The four update procedures that can be called in Algorithm 1 are defined by the presence or absence of two properties. The first is *direct vote priority* (D): an update procedure prioritises *direct votes* over those that can be computed from the current vector Y of votes (i.e., votes found through delegations). The second is *random voter selection* (R): an update procedure *randomly* selects, when possible, a single agent whose direct or computable vote can be added to X. We thus obtain

[5]In the following, we simply write Unravel(#), with $\# \in \{\mathbf{U}, \mathbf{DU}, \mathbf{RU}, \mathbf{DRU}\}$, to indicate the Unravel algorithm using Update procedure #.

Algorithm 2 UPDATE(**U**)

```
1: for $a \in \mathcal{N}$ such that $x_a = \Delta$ do
2:     if $B_a^{\texttt{lev}} \in \mathcal{D}$ then          ▷ add a's vote if a has a direct vote at lev
3:         $x_a := B_a^{\texttt{lev}}$
4:     else if $F_a^{\texttt{lev}}(Y_{\restriction S_a^{\texttt{lev}}}) \in \mathcal{D}$ then
5:         $x_a := F_a^{\texttt{lev}}(Y_{\restriction S_a^{\texttt{lev}}})$          ▷ add a's vote if a has a computable vote at lev
```

the following procedures: basic update (**U**), update with direct vote priority (**DU**), update with random voter selection (**RU**), update with both direct vote priority and random voter selection (**DRU**).

The UPDATE(**U**) procedure[6] in Algorithm 2 updates the vector X with the direct votes for those agents who currently do not have one (line 1), if their preference at `lev` is a direct vote (line 3) or it can be computed from the votes in Y (line 5).

Algorithm 3 UPDATE(**DU**)

```
1: for $a \in \mathcal{N}$ such that $x_a = \Delta$ do
2:     if $B_a^{\texttt{lev}} \in \mathcal{D}$ then          ▷ add all direct votes
3:         $x_a := B_a^{\texttt{lev}}$
4: if $Y = X$ then          ▷ if no direct votes are added to X
5:     for $a \in \mathcal{N}$ such that $x_a = \Delta$ do
6:         if $F_a^{\texttt{lev}}(Y_{\restriction S_a^{\texttt{lev}}}) \in \mathcal{D}$ then          ▷ find and add computable votes to X
7:             $x_a := F_a^{\texttt{lev}}(Y_{\restriction S_a^{\texttt{lev}}})$
```

In Algorithm 3, UPDATE(**DU**) first adds the direct votes from preference level `lev` to X for those agents without a vote in X (line 2). If there are no direct voters at `lev` (line 4), then the procedure tries to add computable votes (line 6).

Algorithm 4 UPDATE(**RU**)

```
1: $P := \emptyset$          ▷ initialise an empty set
2: for $a \in \mathcal{N}$ such that $x_a = \Delta$ do
3:     if $B_a^{\texttt{lev}} \in \mathcal{D}$ or $F_a^{\texttt{lev}}(Y_{\restriction S_a^{\texttt{lev}}}) \in \mathcal{D}$ then
4:         $P := P \cup \{a\}$          ▷ add voters to P if their vote can be determined
5: if $P \neq \emptyset$ then          ▷ there are direct or computable votes in P
6:     select $b$ from $P$ uniformly at random
7:     if $B_b^{\texttt{lev}} \in \mathcal{D}$ then          ▷ If the randomly chosen agent has a direct vote
8:         $x_b := B_b^{\texttt{lev}}$
9:     else if $F_b^{\texttt{lev}}(Y_{\restriction S_b^{\texttt{lev}}}) \in \mathcal{D}$ then          ▷ If random agent has a computable vote
10:        $x_b := F_b^{\texttt{lev}}(Y_{\restriction S_b^{\texttt{lev}}})$
```

The UPDATE(**RU**) procedure has the random voter selection property (Algorithm 4): at line 1 an empty set P is initialised to store agents with either a direct

[6]Unless otherwise specified, in case the condition in an **if** statement fails, our programs skip to the next step. Also, recall that $Y_{\restriction S}$ denotes the restriction of vector Y to the elements in set S.

Algorithm 5 Update(**DRU**)

```
1: $P, P' := \emptyset$                                    ▷ initialise two empty sets
2: for $a \in \mathcal{N}$ such that $x_a = \Delta$ do
3:     if $B_a^{\texttt{lev}} \in \mathcal{D}$ then           ▷ add agents with direct votes at lev to P
4:         $P := P \cup \{a\}$
5:     else if $F_a^{\texttt{lev}}(Y \restriction_{S_a^{\texttt{lev}}}) \in \mathcal{D}$ then   ▷ add agents with computable votes at lev to P'
6:         $P' := P' \cup \{a\}$
7: if $P \neq \emptyset$ then                              ▷ if there are agents with direct votes
8:     select $b$ from $P$ uniformly at random
9:     $x_b := B_b^{\texttt{lev}}$          ▷ add only the randomly selected voter's direct vote to X
10: else if $P' \neq \emptyset$ then                       ▷ if there are some computable votes
11:     select $b$ from $P'$ uniformly at random
12:     $x_b := F_b^{\texttt{lev}}(Y \restriction_{S_b^{\texttt{lev}}})$     ▷ add only the randomly selected voter's computable vote to X
```

vote or a computable vote at `lev` (line 3). If the set P is non-empty, one agent is randomly selected and their direct or computable vote is added to X.

Lastly, Algorithm 5 presents Update(**DRU**), which has both properties. At `lev`, the procedure adds agents with direct votes to P (line 3) and agents with computable votes to P'. If P is non-empty, an agent is selected from P, and their direct vote is added to X (line 9). Otherwise, if P is empty and P' is not, an agent is selected from P' and their computable vote is added to X (line 12). If both P and P' are empty, no votes are added to X and the procedure terminates.

We now give an example of the execution of these four unravelling procedures:

Example 2.4. For a binary issue with $\mathcal{D} = \{0, 1\}$ consider agents $\mathcal{N} = \{a, \dots, f\}$. In Figure 2.3, we show the agents' Bool ballots and the profiles delegation structure are represented schematically. Observe that $\boldsymbol{B}$ abides by Definition 2.2 and is thus a valid profile. We now illustrate our four greedy unravelling procedures.

	B_x^1	B_x^2	B_x^3
a	$(\{b, c, d\}, (b \wedge c) \vee (b \wedge d)$	$(\{e\}, e)$	1
b	1	-	-
c	0	-	-
d	$(\{e\}, e)$	0	-
e	$(\{f\}, f)$	1	-
f	$(\{a\}, a)$	$(\{b\}, b)$	1

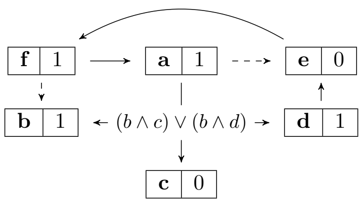


Figure 2.3: Representation of the ballots (left) and the delegation structure (right) of the agents in $\boldsymbol{B}$ from Example 2.4. In the graph on the right, a solid line indicates the first preference for delegation, a dashed line represents the second, and the final preference (a direct vote in $\{0, 1\}$) is written next to the agents' names.

Unravel(U) At `lev` $= 1$, the procedure adds the direct votes of b and c to X. Thus, we have $X = (\Delta, 1, 0, \Delta, \Delta, \Delta)$. Then, the algorithm cannot find a direct or computable vote at `lev` $= 1$, so it moves to `lev` $= 2$ where it uses Y to add the direct votes of d and e, as well as f's vote that is computable from X by copying b, giving $X = (\Delta, 1, 0, 0, 1, 1)$. As no other update is possible at this level, the algorithm sets `lev` $= 1$, and it computes a's vote, yielding $X = (0, 1, 0, 0, 1, 1)$, with $\mathbf{c} = (1, 1, 1, 2, 2, 2)$.

Unravel(DU) As with Unravel(**U**), the direct votes of b and c are added initially to $X = (\Delta, 1, 0, \Delta, \Delta, \Delta)$, and then the algorithm moves to `lev` $= 2$ as no votes at `lev` $= 1$ can be added. Unlike Unravel(**U**), the procedure Unravel(**DU**) adds only the direct votes of d and e, giving $X = (\Delta, 1, 0, 0, 1, \Delta)$. Returning to `lev` $= 1$, a's vote can be computed from the votes of b, c and d, giving $X = (0, 1, 0, 0, 1, \Delta)$. Finally, at `lev` $= 1$, f's delegation is computable (a delegation to a) and is added to X, thus giving $X = (0, 1, 0, 0, 1, 0)$, with certificate $\mathbf{c} = (1, 1, 1, 2, 2, 1)$.

Unravel(RU) First, the direct votes of b and c are added, each in a separate iteration, giving $X = (\Delta, 1, 0, \Delta, \Delta, \Delta)$. Then, the algorithm moves to `lev` $= 2$, where it chooses a single vote at random to add to X from the agents d, e and f.

- If d's direct vote was chosen, then $X = (\Delta, 1, 0, 0, \Delta, \Delta)$. Unravel(**RU**) returns to `lev` $= 1$ where a's vote can be computed, giving $X = (0, 1, 0, 0, \Delta, \Delta)$. Following this, at `lev` $= 1$ the computable vote of f and then e can be added giving $X = (0, 1, 0, 0, 0, 0)$ with certificate $\mathbf{c} = (1, 1, 1, 2, 1, 1)$.

- If e's vote was chosen, then $X = (\Delta, 1, 0, \Delta, 1, \Delta)$. Then at `lev` $= 1$, d's vote can be computed from e's, giving $X = (\Delta, 1, 0, 1, 1, \Delta)$. At `lev` $= 1$, A's computable vote can now be added and then F's computable vote can be added from A's, giving $X = (1, 1, 0, 1, 1, 1)$ with certificate $\mathbf{c} = (1, 1, 1, 1, 2, 1)$.

- If the computable vote of f was added, then $X = (\Delta, 1, 0, \Delta, \Delta, 1)$. At `lev` $=$ 1, e's vote can be computed from f's, and then d's from e's, giving $X = (\Delta, 1, 0, 1, 1, 1)$. Then, at `lev` $= 1$, a's vote can be computed from b, c and d's, yielding $X = (1, 1, 0, 1, 1, 1)$, whose certificate is $\mathbf{c} = (1, 1, 1, 1, 1, 2)$.

Unravel(DRU) This procedure behaves in the same manner as Unravel(**RU**) in the example, with the only exception being that Unravel(**DRU**) chooses randomly only between the direct votes of d and e at the iteration where Unravel(**RU**) can also choose to select the computable delegation of agent f.

In this example, MinMax would return outcomes all certificates $\mathbf{c}$ where $\max(\mathbf{c}) = 2$. This would include, e.g., $\mathbf{c} = (1, 1, 1, 2, 1, 1)$, which is also returned by MinSum, but also $\mathbf{c}' = (2, 1, 1, 2, 2, 2)$ and many more. △

2.4 Computational Complexity of Unravellings

In this section, we study the computational complexity of our unravelling procedures when focusing on our general language Bool. First, we study how hard it is to unravel a smart profile under a given procedure. We begin with MinSum and MinMax, showing that an associated decision problem, respectively BoundedMinSum and BoundedMinMax, are NP-complete. However, a solution can be found in polynomial time when smart ballots are restricted to Liquid. Unlike MinSum and MinMax, we show that our greedy procedures, Unravel(#) with $\# \in \{\mathbf{U}, \mathbf{DU}, \mathbf{RU}, \mathbf{DRU}\}$, always terminate in a polynomial number of time steps.

2.4.1 Computational Complexity of MinSum

In this section, we study the computational complexity of finding MinSum outcomes when ballots are restricted to either the Bool or Liquid language, finding the problem to be NP-complete in the former case and tractable in the latter.

We begin by studying the decision problem BoundedMinSum rather than the search problem of finding all solutions to MinSum.

BoundedMinSum	
Given:	A smart profile $\boldsymbol{B}$, such that every ballot is valid and restricted to be in Bool$[k]$, and $M \in \mathbb{N}$
Question:	Does there exist a consistent certificate $\mathbf{c}$ that unravels $\boldsymbol{B}$ such that $\texttt{rank}(\mathbf{c}) \leq M$?

Repeatedly using BoundedMinSum for different values of M gives us the minimum value of M, which equates to the `rank` of the certificates found by MinSum. A modified version of BoundedMinSum using partial certificates would allow us to compute an outcome of MinSum. Both problems are harder than BoundedMinSum, which we now show to be NP-complete.

Lemma 2.1. *BoundedMinSum is in NP.*

Proof. Recall that BoundedMinSum is defined on Bool profiles. We prove membership in NP by showing that a witness can be checked in polynomial time. Our witness is the certificate vector $\mathbf{c} \in \mathbb{N}^n$, such that $\mathbf{c}_i$ represents the preference level of agent $i \in \mathcal{N}$ when unravelling the profile.

First, we check that $\mathbf{c}$ abides by Definition 2.6 and is, in fact, a certificate for the profile: that is, for each $i \in \mathcal{N}$, $\mathbf{c}_i$ corresponds to a preference level in B_i. To do this, we need to read the certificate and the profile, taking a polynomial number of time steps. Next, we check that $\mathbf{c}$ is consistent with respect to Definition 2.7: we first find the direct voters $B_i^{\mathbf{c}_i} \in \{0, 1\}$ from the certificate and the profile, and we add them to a set D. We construct the outcome vector $X \in \{\Delta\}^n$ and append the entry $X_i = B_i^{\mathbf{c}_i}$ for these direct voters, which can be done in polynomial time.

Then, we check if any necessary winners can be computed from D: for each agent $i \in \mathcal{N} \setminus D$ such that there exists a $j \in D$ such that $j \in S_i^{\mathbf{c}_i}$, we check if we can compute a necessary winner of $F_i^{\mathbf{c}_i}$ given $X \restriction_{S_i^{\mathbf{c}_i} \cap D}$. If so, we add $i \in D$ and let $X_i = F_i^{c_i}(X \restriction_{S_i^{c_i} \cap D})$. Since all delegation functions are in complete DNF, we can check for a necessary winner in polynomial time by Proposition 2.1. Since at least one agent gives a direct vote, we must check at most $n-1$ agents' delegation functions for a necessary winner in the first round. If the certificate $\mathbf{c}$ is consistent, at least one agent is added in each round. Therefore, we have to do at most $\sum_{k=1}^{n-1} k = \frac{(n-1)n}{2}$ polynomial checks if, in the worst case, only a single agent is found in each round. Finally, we check in polynomial time that $\sum_{i \in \mathcal{N}} \mathbf{c}_i \leq M$. All steps can be done in polynomial time, showing that BoundedMinSum is in NP. □

We now show that BoundedMinSum is NP-hard by giving a reduction from Feedback Vertex Set (FVS), a problem shown by Karp [1972] to be NP-complete. The input of FVS is a directed irreflexive graph $G = (V, E)$ and a positive integer k,[7]. It asks if there is a subset $X \subseteq V$ with $|X| \leq k$ such that when all vertices of X and their adjacent edges are deleted from G, the remaining graph is cycle-free.

Lemma 2.2. *BoundedMinSum is NP-hard.*

Proof. Recall that BoundedMinSum is defined for the language of complete DNFs. We prove the claim by reducing from Feedback Vertex Set (FVS). Given an instance (G, k) of FVS, let an instance of BoundedMinSum be such that $\mathcal{N} = V$, $M = |V| + k$, and for each vertex-agent $v \in V$ their ballot B_v is constructed as follows, we let $O_v = \{u \in V \mid (v, u) \in E\}$ be the set of outgoing edges of vertex v in G their ballot is then:

$$B_v = (O_v, \bigwedge_{u \in O_v} u) > 1.$$

The first delegation of each agent v is a conjunction of positive literals (hence, a formula in complete DNF), each representing one of the outgoing edges from v in graph G. Then, the backup vote for 1 represents the removal of the vertex v in the FVS problem. For the agents $v \in V$ without any outgoing edges ($O_v = \emptyset$), their ballot is $B_v = 1$.

To show the correctness of our reduction, we first prove the following claim: a graph G is acyclic if and only if $\mathbf{c} = \{1\}^n$ is a consistent certificate for the profile $\boldsymbol{B}$ given by the translation above.

For the left-to-right direction, we prove the contrapositive: assume that the certificate $\mathbf{c} = \{1\}^n$ is not consistent for $\boldsymbol{B}$. Therefore, there is no ordering of the agents such that all their votes can be iteratively added using the previously added votes. This means there is a delegation cycle between the formulas at the first

[7]The formulation by Karp [1972] is on directed graphs G which allow for reflexive edges. However, our sub-problem is also NP-complete. A reduction can be given where the constructed graph G' adds a dummy node a' for each node a that had a reflexive edge in G and the edges (a, a') and (a', a).

preference level of some agents in $\boldsymbol{B}$, as at least two agents require each others' votes to determine their own. By the construction of $\boldsymbol{B}$, the literals in the formulas represent the outgoing edges in G, and the graph G is not acyclic.

For the right-to-left direction, let $\mathbf{c} = \{1\}^n$ be a consistent certificate for $\boldsymbol{B}$. First, note that all nodes in G representing non-delegating agents in $\boldsymbol{B}$ have no outgoing edges and are not in a cycle in G. Second, for each delegating agent $v \in V$, their first preference delegation, $\bigwedge_{u \in O_v} u$, can be determined. Since every voter can only be assigned a vote of 1 (their direct vote), the truth value of the formula $\bigwedge_{u \in O_v} u$ can only be determined when all delegations have been resolved. Thus, every maximal path in G starting from a node v ends in a node without any outgoing edges (a node representing a direct voter). Therefore, G is acyclic.

We now prove the reduction using the previous claim. First, assume that a subset X exists such that $|X| \leq k$ and the resulting graph has no cycles: we want to show that $\texttt{rank}(\boldsymbol{B}) \leq M = |V| + k$. If all agents in X receive their second preference, then all remaining agents in $\mathcal{N} \setminus X$ get their first preference. Since this subset is acyclic, it is also consistent (given our claim above), and the addition of direct voters does not impact the consistency of a certificate. The rank of this unravelling is $|V| + |X| \leq |V| + k$ and therefore, $\texttt{rank}(\boldsymbol{B}) \leq M = |V| + k$.

Next, we show that if $\texttt{rank}(\boldsymbol{B}) \leq M = |V| + k$, a subset X exists, such that $|X| \leq k$ and the remainder of G without X is cycle-free. We let $\mathbf{c}$ be the certificate of unravelling $\boldsymbol{B}$ such that the rank is less than or equal to $|V|+k$. From $\mathbf{c}$, we build $X = \{u \mid \texttt{rank}(\mathbf{c}_u) = 2\}$. We remove the agents in X from the profile, both their ballots and any mention of them in delegations. Since $\texttt{rank}(\boldsymbol{B}) \leq M = |V| + k$, it must be the case that $|X| \leq k$. Thus, the restriction of the certificate to $v \in \mathcal{N} \setminus X$ must mean that they each have $\mathbf{c}_v = 1$ by the construction of the profile and X. We can now use the claim above to state that the resulting graph with nodes $V \setminus X$ is acyclic. Therefore, BoundedMinSum is NP-hard. $\square$

Theorem 2.2. *BoundedMinSum is NP-complete.*

Remark 2.1. The reduction in the proof of Lemma 2.2 does not use negated literals in the ballots. Thus, BoundedMinSum would still be NP-complete if delegations were restricted further to contingent complete DNF formulas with only positive literals. By similar reasoning, it would also be NP-complete if Bool was restricted to contain only formulas built with conjunctions, i.e., cubes.

The proof of our next result uses *Edmonds' algorithm* [Edmonds, 1967].[8] This algorithm finds, for a given weighted directed graph, a minimum *arborescence tree*, i.e., a directed rooted tree minimising the sum of the weights of the edges in the tree.[9]

Edmond's algorithm takes as input a (pre-processed[10]) weighted directed graph

[8]Also independently suggested by Chu [1965] and Bock [1971].

[9]For undirected graphs, the corresponding problem is that of finding a minimum spanning tree.

[10]The algorithm removes all incoming edges to the root r, and in case of parallel edges (i.e., edges between the same nodes, in the same direction), it removes them all except the edge with minimum weight.

$D = (V, E, w)$ and a root $r \in V$, where V is a set of vertices, E is a set of edges, and w is a vector of the edges' weights. At each step, the algorithm picks a vertex $v \in V \setminus \{r\}$ that does not yet have an incoming edge in the arborescence tree. It selects the incoming edge of v with the minimum weight. After each edge has been added, the algorithm checks if a cycle has formed: if that is the case, the nodes involved in the cycle are *contracted* to a single node v_C, creating a new directed graph D'. The algorithm continues until the contracted graph is a directed spanning tree. Finally, all of the contractions are expanded.

The cycles are contracted as follows: Given a set of nodes C involved in a cycle, we let $V' = (V \setminus C) \cup \{v_C\}$, for a new node v_C representing the cycle. In case there is an edge entering the cycle but not currently involved in it, i.e., $e_{uv} \in E$ for $u \notin C$ and $v \in C$, we let $e_{uv_C} \in E'$ be such that $w(e_{uv_C}) = w(e_{uv}) - w(e_{wv})$ where e_{wv} is the lowest weighted incoming edge of v (the weight of e_{uv_C} corresponds to the incoming weight to the cycle, minus the lowest weighted incoming weight to node v in the cycle). In case there is an edge exiting the cycle, i.e., $e_{vu} \in E$ for $v \in C$ and $u \notin C$, we let $e_{v_C u} \in E'$ have weight $w(e_{v_C u}) = w(e_{vu})$. All edges whose nodes are not involved with the cycle C, including their weights, remain unchanged.

Theorem 2.3. *An outcome in* MinSum$(\boldsymbol{B})$ *on a profile* $\boldsymbol{B}$ *in* Liquid *can be found in* $\mathcal{O}(n(d+n))$ *time, where* d *represents the number of delegations in* $\boldsymbol{B}$*.*

Proof. The idea of the proof is to create a graph on which to apply Edmonds' algorithm Edmonds [1967]. For a profile $\boldsymbol{B}$ of Liquid ballots, we construct a directed graph $D = (V, E)$, where $V = \mathcal{N} \cup \{r\}$ and r is a fresh node. For the edges in E, we let $e_{ji} \in E$ if $B_i^k = (\{j\}, j)$ for some k, i.e., we add an edge from j to i if i was delegating to j at i's k^{th} preference level. Furthermore, we add an edge $e_{ri} \in E$ for all $i \in \mathcal{N}$, representing the final direct vote of each voter. The weight of each edge is always given by its preference level in the ballot: if $B_i^k \in \mathcal{D}$ then $w(e_{ri}) = k$, and if $B_i^k = (\{j\}, j)$ then $w(e_{ji}) = k$.

Edmonds' algorithm returns the arborescence tree $A = (V, E')$ rooted at r in time $\mathcal{O}(|V| \times |E|)$, minimising its weight $w(A) = \sum_{e_{ij} \in E'} w(e_{ij})$. By applying Edmonds' algorithm to the graph D above, we find an unravelling of $\boldsymbol{B}$, whose certificate vector $\mathbf{c}$ minimises $\texttt{rank}(\mathbf{c})$. Moreover, since it returns a tree which includes every node, there are no delegation cycles and every agent has exactly one of their preference levels used. As a result, the arborescence tree corresponds to a consistent certificate for unravelling the profile with a minimal `rank`.

Thus, we can find a solution of MinSum in $\mathcal{O}(|V| \times |E|) = \mathcal{O}((n+1) \times (d+n))$ time steps, since $|V| = n+1$ (all the agents plus the vertex r), and $|E| = d+n$, where d represents the number of delegations in $\boldsymbol{B}$. When simplifying the bound, this can be done in $\mathcal{O}(n(d+n))$ time steps. □

We can thus find an optimal unravelling of a Liquid smart profile in a polynomial number of time steps. Furthermore, as Edmonds' algorithm is recursive, we are guaranteed that it will terminate, giving an optimal unravelling provided that there is some tie-breaking rule when there are many optimal unravellings. We now illustrate in an example the application of Edmonds' algorithm in the proof of Theorem 2.3.

Example 2.5. Edmonds' algorithm can be applied to a Liquid smart profile $\boldsymbol{B}$, as in the proof of Theorem 2.3, to get a MinSum outcome. Consider the profile $\boldsymbol{B}$ in Table 2.1. The directed graph $D = (V, E, w)$ has nodes $V = \{a, b, c, d, e, r\}$, edges $E = \{(ra), (cb), (ab), (rb), (dc), (ec), (rc), (bd), (ed), (rd), (re)\}$, and weights $w(ra) = w(cb) = w(dc) = w(bd) = w(re) = 1$, $w(ab) = w(ec) = w(ed) = 2$, and $w(rb) = w(rc) = w(rd) = 3$. The graph D is shown on the left of Figure 2.4, with solid, dashed, and dotted lines representing first, second, and third preference levels in the ballots, respectively.

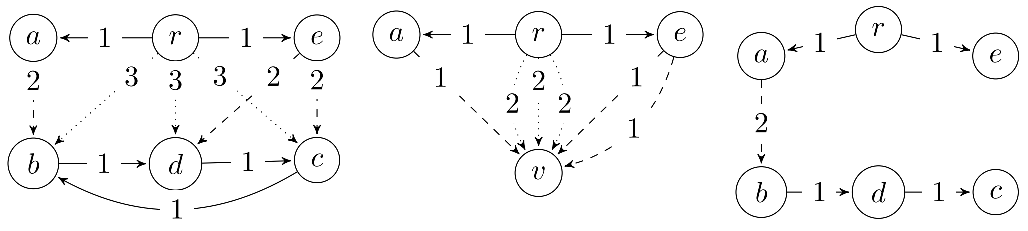
	B_x^1	B_x^2	B_x^3
a	1	-	-
b	$(\{c\}, c)$	$(\{a\}, a)$	$*$
c	$(\{d\}, d)$	$(\{e\}, e)$	$*$
d	$(\{b\}, b)$	$(\{e\}, e)$	$*$
e	0	-	-

Table 2.1: The Liquid profile that is unravelled via Edmonds' algorithm in Example 2.5, and by the algorithm introduced in Theorem 2.5 in Example 2.7.

In Figure 2.4, we see at the bottom of D that there is a cycle among first preference delegations of nodes b, c and d. Edmonds' algorithm contracts this cycle to a single vertex v, creating a second directed graph $D' = (V', E', w)$, shown in the centre of Figure 2.4. The nodes of D' are $V' = \{a, v, e, r\}$; while for the edges E', we keep (ra) and (re) but we alter the edges coming into and out of the cycle. However, note that there are only incoming edges to the cycle: $(ab), (rb), (rd), (rc), (ed), (ec)$. Thus, we add to E' only edges coming into v, taking into account the lowest weighted incoming edge to each node in the cycle.

For the edge $(rb) \in E$, we thus have an edge $(rv) \in E'$ whose weight is computed as $w(rv) = w(rb) - w(xb)$ where $w(xb)$ is the weight of the lowest incoming edge of b, e.g., (cb), which has weight $w(cb) = 1$. Thus, $w(rv) = 3 - 1 = 2$, and analogously for (rc) and (rd). For the edge $(ed) \in E$, we have an edge $(ev) \in E'$ whose weight is $w(ev) = w(ed) - w(xd) = 2 - 1 = 1$, and similarly for (ec). Finally, for $(ab) \in E$, we have an edge $(av) \in E'$ with weight $w(av) = w(ab) - w(xb) = 1$.

Since there are no cycles in D', we can find an arborescence tree of D' rooted at r with edges $(ra), (re)$ and then either (av) or (ev), as they both have the lowest weight of 1. Suppose that (av) is chosen. This represents the delegation from b to a with weight 2. By choosing this edge, a will be followed by b in the arborescence tree—this unravelling is shown on the right-hand side of Figure 2.4. From here, the unravelling continues until all of the vertices in the cycle have been chosen, giving the edges (bd) and (dc). Alternatively, the algorithm could have chosen the edges (ec) or (ed) instead of (ab): all of these unravellings are optimal, with a total weight of 6. △



Directed graph D **Directed graph D'** **Choice of edge (ab)**

Figure 2.4: Three stages of unravelling the Liquid profile $\boldsymbol{B}$ from Table 2.1 by using Edmonds' algorithm. The directed graph D (left) represents the initial profile. In D' (centre), the nodes b, c and d are contracted into v, as they were in a cycle in D. The arborescence tree (right) is the output where the edge (ab) was chosen to break the tie, and it corresponds to an outcome of MinSum on $\boldsymbol{B}$.

2.4.2 Computational Complexity of MinMax

We now study the computational complexity of the MinMax rule, showing that, like MinSum, it is NP-hard for the language Bool and polynomial-time solvable for Liquid. We begin by studying the problem BoundedMinMax.

BoundedMinMax	
Given:	A valid smart profile $\boldsymbol{B}$ restricted to Bool and $M \in \mathbb{N}$
Question:	Does there exist a consistent certificate $\mathbf{c}$ that unravels $\boldsymbol{B}$ such that $\max_{a \in \mathcal{N}}(\mathbf{c}) \leq M$?

We first show membership in NP and then NP-hardness.

Lemma 2.3. *BoundedMinMax is in NP.*

Proof. Recall that BoundedMinMax is defined on Bool profiles. To prove membership in NP, we can check in polynomial time that a certificate vector $\mathbf{c}$ abides by Definition 2.6 and is consistent, as we did for Lemma 2.1. Then, we need to check that all entries in the certificate are less than or equal to the constant M, which can be done in polynomial time. □

Lemma 2.4. *BoundedMinMax is NP-hard.*

Proof. Recall that BoundedMinMax is defined on Bool ballots, where delegations are expressed in complete DNFs. We reduce from the NP-complete problem CNF-Sat which takes as input a formula φ in CNF—i.e., a conjunction of *clauses* (disjunctions of literals)—and it then asks if there exists a satisfying assignment for φ.

For a given formula φ in CNF, let $C = \{c \mid c \text{ is a clause of } \varphi\}$ be a set of variables, each $c \in C$ representing one of the clauses of φ. We now construct an instance of BoundedMinSum where $M = 2$ and the set of agents is $\mathcal{N} = \{x, y\} \cup C \cup Var(\varphi)$, with x and y being fresh variables. The ballots are defined as follows:

- $B_x = (1)$,
- $B_v = ((\{x\}, x) > 0)$ for all $v \in Var(\varphi)$,
- $B_y = ((\{x\} \cup C, x \wedge \bigwedge_{c \in C} c) > (C, \bigwedge_{c \in C} c) > 1)$,
- $B_c = ((\{y\}, y) > (\{y\} \cup Var(c), y \vee \bigvee_{l \in c} l) > 1)$ for all $c \in C$, where $l \in c$ represents the literal l of clause c. If c contains a variable and its negation (i.e., c is a tautology), we remove the second-level delegation in the ballot.

Note that each delegation is a complete DNF since it is either a cube or a clause. Moreover, note that each delegation function is a contingent formula.

First, assume that φ is satisfiable. Therefore, there exists a satisfying truth assignment which can be given as a vector $X \in \{0, 1\}^{|Var(\varphi)|}$. We then build a consistent certificate $\mathbf{c}$ where each agent $v \in Var(\varphi)$ gets their first preference delegation $\mathbf{c}_v = 1$ if in the satisfying truth assignment of φ the variable v is true $X_v = 1$, or their second preference $\mathbf{c}_v = 2$ if v is false in the truth assignment $X_v = 0$. The truth assignment X makes every clause of φ true, and thus, each $c \in C$ true. Therefore, at least one literal in $y \vee \bigvee_{l \in c} l$ is made true, making the whole formula true. Thus, agents $c \in C$ cannot receive higher than their second preference ($\mathbf{c}_c \leq 2$). Agent y can receive their first preference ($\mathbf{c}_y = 1$), given that we can determine the vote of each $c \in C$. Finally, agent x receives their first preference ($\mathbf{c}_x = 1$). Therefore, if φ is satisfiable, there is an unravelling such that every agent receives at most their second preference level in the ballot.

Next, assume that φ is not satisfiable. Therefore, no truth assignment satisfies φ. For a contradiction, we assume that there exists an unravelling with a consistent certificate $\mathbf{c}$ such that $\max(\mathbf{c}) \leq 2$. By constructing the profile, we see that $\mathbf{c}_x = 1$ and for all $v \in Var(\varphi)$ that $\mathbf{c}_v \leq 2$, as they do not have higher than a second preference level in their ballots. Furthermore, by assumption, it must be the case that either $\mathbf{c}_y = 1$ or $\mathbf{c}_y = 2$. If $\mathbf{c}_y = 1$, then either all clauses $c \in C$ evaluate to 1 or there exists a $c \in C$ whose vote is 0, given that x votes for the issue (1). In the latter case, this has to come from c's first or second preference. It cannot be c's first delegation, as c's vote determines y's vote. Thus, the unravelling would not be consistent. Their second preference can only be 0 if all of the literals of c and y are false, which cannot be determined without the vote of y; thus, we have reached a contradiction. However, if each $c \in C$ has a vote of 1, this entails that each $c \in C$ can be made true (using the second preference delegation). Therefore, φ is satisfiable, or the third preference of $c \in C$ has been used. Hence, $\max(\mathbf{c}) > 2$ if φ is unsatisfiable, reaching a contradiction. The same reasoning holds for $\mathbf{c}_y = 2$. Therefore, BoundedMinMax is NP-hard. □

We now give an example of the translation given in Lemma 2.4 to exemplify the reduction.

Example 2.6. We show the translation of the formula $\varphi = i \wedge (j \vee \neg k)$. This translation gives the set of agents $\mathcal{N} = \{x, y, i, j, k, c_1, c_2\}$ with the following ballots (depicted in Figure 2.5).

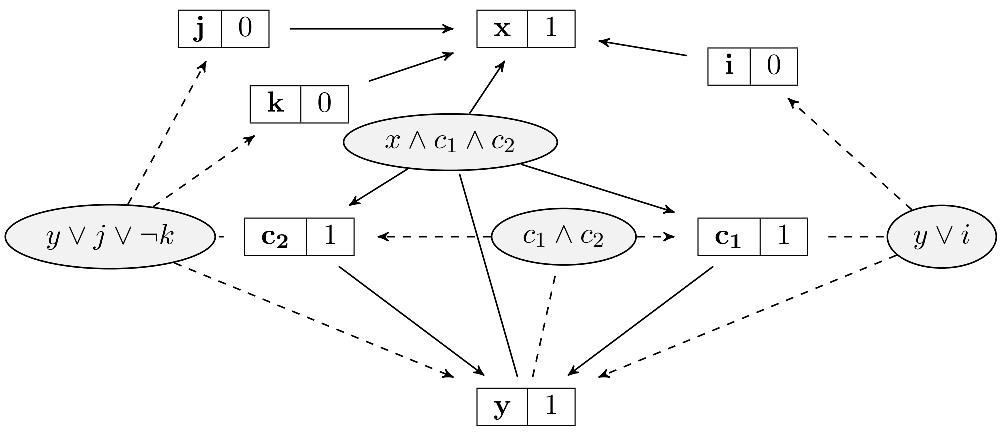


Figure 2.5: An example of the reduction used in Lemma 2.4 from CNF-SAT on the formula $\varphi = i \wedge (j \vee \neg k)$. This diagram depicts the agents' ballots. In each box, we see the agent's name alongside their final direct backup vote. In each ellipse, we give the multi-agent delegation function of an agent. Moreover, the solid lines represent the first delegation, while the dashed line represents a second preference delegation.

- $B_x = 1$;
- for each $a \in \{i, j, k\}$, $B_a = x > 0$;
- $B_y = (x \wedge c_1 \wedge c_2) > (c_1 \wedge c_2) > 1$;
- $B_{c_1} = y > y \vee i > 1$;
- $B_{c_2} = y > (y \vee j \vee \neg k) > 1$.

△

Lemmas 2.3 and 2.4 together give us the following:

Theorem 2.4. *BOUNDEDMINMAX is NP-complete.*

Therefore, as our bounded decision problem is intractable for our most general language, a natural question to ask next is whether there are restrictions on the ballots such that MINMAX can be solved in polynomial time. As with MINSUM, we show that MINMAX can tractably find solutions on LIQUID ballots.

Theorem 2.5. *An outcome of MINMAX on a profile $\boldsymbol{B}$ in LIQUID can be found in time $\mathcal{O}(n^2\ell^2)$, where ℓ is the highest preference level of any agent in the profile.*

Proof. We provide an algorithm to find a MINMAX outcome by transforming the profile $\boldsymbol{B}$ into a directed graph and then finding an arborescence tree.

We construct a directed graph $G = (V, E, w)$ where $V = \mathcal{N} \cup \{r\}$, where r is a fresh node that roots our tree. The set of edges E will reflect the single-agent delegations as before, where each edge has a weight representing its preference level.

We iteratively build the graph with respect to the edges' weights and check if there is an arborescence tree at each iteration. Thus, we start with $E = \emptyset$. Starting from $\texttt{lev} = 1$ until $\texttt{lev} = \ell$, where ℓ is the maximum preference level given by any agent in $\boldsymbol{B}$, the following procedure is executed:

1. Add to the current set E an edge e_{ij} if $B_i^{\texttt{lev}} = (\{j\}, j)$, and an edge e_{ri} if $B_i^{\texttt{lev}} \in \mathcal{D}$. Namely, $E := E \cup \{e_{ji} \mid B_i^{\texttt{lev}} = (\{j\}, j)\} \cup \{e_{ri} \mid B_i^{\texttt{lev}} \in \mathcal{D}\}$. In both cases, let $w(e_{ij}) = w(e_{ri}) = \texttt{lev}$;

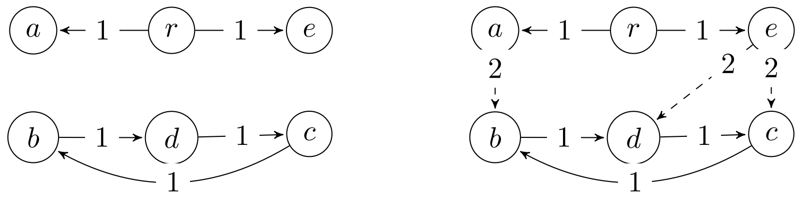
2. Check in $\mathcal{O}(|V| + |E|)$ time (see, e.g., Cormen et al. [2009], pg. 606) if there is a path from r to a via the edges in the current E, for each $a \in \mathcal{N}$ (hence, this step has to be repeated n times, for each $a \in \mathcal{N}$). Then, if r is connected to all $a \in \mathcal{N}$, exit the loop; otherwise, if there is some $a \in \mathcal{N}$ not connected to r, let $\texttt{lev} := \texttt{lev} + 1$.

After the execution of this iterative procedure, we obtain a graph G where all nodes are connected to r. To find a solution, we then find *any* arborescence tree from E rooted at r in $\mathcal{O}(|V| + |E|)$ time [Kozen, 2012, pg. 19]. The consistent certificate $\mathbf{c}$ giving an outcome of MinMax is then given by $\mathbf{c}_a$ being the incoming weight of a in the arborescence tree, for all $a \in \mathcal{N}$. Intuitively, we obtain a MinMax outcome since the root r represents direct votes. We can determine an agent's vote if there is a path from r to them. Since the edges are added iteratively, we know that a path does not exist for a lower maximum preference level.

As all agents give a backup vote, $e_{ri} \in E$ can be eventually added for each $i \in \mathcal{N}$. Thus, the algorithm always terminates. Since the loop iterates at most ℓ times, and each time it makes n checks, each bounded by $\mathcal{O}(|V| + |E|)$, overall it takes at most $\mathcal{O}(n\ell(|V| + |E|))$. Since $|V| = n + 1$ and $|E| \leq n\ell$, the time bound is $\mathcal{O}(n\ell(n + 1 + n\ell))$. In $\mathcal{O}(|V| + |E|) = \mathcal{O}(n + 1 + n\ell)$ an arborescence tree is found. Thus, a solution can be found in $\mathcal{O}((n\ell + 1)(n + 1 + n\ell))$ time steps, which can be simplified to $\mathcal{O}(n^2\ell^2)$. □

We now show an example applying the algorithm from the proof of Theorem 2.5 to find a MinMax outcome on a Liquid profile.

Example 2.7. Consider the Liquid profile in Table 2.1. To find a solution to MinMax, we first construct the directed graph $D_1 = (V, E_1, w)$, shown in Figure 2.6 (left), where $V = \{a, b, c, d, e, r\}$ and E_1 are the edges added when considering $\texttt{lev} = 1$. Since the nodes b, d and c are not connected to the root r in D_1, we set $\texttt{lev} = 2$ and we create the graph $D_2 = (V, E_2, w)$, shown in Figure 2.6 (right). The set E_2 contains edges representing the first and second preference levels. Since in D_2, there is at least one path from r to every other node, we search for an arborescence tree that represents a MinMax outcome, e.g., via a depth-first algorithm. One such tree has edges $\{(ra), (re), (ab), (bd), (dc)\}$. △



Directed graph D_1 **Directed graph D_2**

Figure 2.6: Application of the algorithm in the proof of Theorem 2.5 to the Liquid profile $\boldsymbol{B}$ from Table 2.1. The directed graph D_1 (left) shows the algorithm's first iteration using only the agents' first preference levels. As there is no path from r to every other node in D_1, the algorithm moves to the second iteration, constructing D_2 (right), which shows the agents' first and second preference levels. Since D_2 is connected, the algorithm terminates.

2.4.3 Computational Complexity of the Greedy Unravellings

We now turn to the complexity of the greedy unravelings. In this section, we show that Unravel(#) always terminates when paired with any update procedure $\# \in \{\mathbf{U}, \mathbf{DU}, \mathbf{RU}, \mathbf{DRU}\}$, given a valid profile. Moreover, we show that they always terminate polynomial time.

Proposition 2.3. *Algorithms Unravel(#) with $\# \in \{\mathbf{U}, \mathbf{DU}, \mathbf{RU}, \mathbf{DRU}\}$ always terminate on a valid smart profile $\boldsymbol{B}$.*

Proof. Let $\boldsymbol{B}$ be a valid smart profile for n agents. For the sake of a contradiction, assume that Unravel(#) given in Algorithm 1 does not terminate on a valid profile $\boldsymbol{B}$. Hence, Unravel cannot exit the **while** loop from either line 6, due to no direct votes being computable at any preference level, or from line 3, due to $X \notin \mathcal{D}^n$.

Consider a case where Unravel cannot terminate due to a cycle involving the **while** loop from line 6. Let $A = \{a \in \mathcal{N} \mid x_a = \Delta\}$ be the set of agents whose votes have not been computed due to a delegation cycle. As $\boldsymbol{B}$ is a valid smart profile, we know that for all $a \in A$, B_a has a finite number of preference levels[11] and the final preference must be a direct vote. In each of the update procedures ($\mathbf{U}, \mathbf{DU}, \mathbf{RU}$ and $\mathbf{DRU}$), after a finite number of loops, we reach a direct vote of an agent in A. Each update procedure adds at least one direct vote to the vector of direct votes X at this point, breaking this cycle. Moreover, no procedure replaces a vote in X with Δ or with any value not in $\mathcal{D}$.

Therefore, if the algorithm does not terminate, it must be due to the **while** loop at line 3. This can only happen while $X \notin \mathcal{D}^n$. However, as we can exit the cycle from line 6, the algorithm always changes some $x_a = \Delta$ to a vote in $\mathcal{D}$. Thus, after a finite number of iterations, $X \in \mathcal{D}^n$ and Unravel terminates. □

[11]Recall that since both $\mathcal{D}$ and the possible sets of delegates are finite, and since all functions given in an agent's valid ballot must differ, the possible number of functions must also be finite.

Next, we show that our unravelling procedures terminate in polynomial time on Bool ballots. Recall that the delegations in a Bool ballot are contingent Boolean functions φ expressed in complete DNF. Thus, the size of the input $\boldsymbol{B}$ is in $\mathcal{O}(\max_p(\boldsymbol{B}) \cdot n \cdot \max_\varphi(\boldsymbol{B}))$, where $\max_p(\boldsymbol{B})$ is the highest preference level of any ballot in $\boldsymbol{B}$ and $\max_\varphi(\boldsymbol{B})$ is the maximum length of any formula in $\boldsymbol{B}$.

Proposition 2.4. *Unravel($\#$) for $\# \in \{\mathbf{U}, \mathbf{DU}, \mathbf{RU}, \mathbf{DRU}\}$ terminates in at most $\mathcal{O}(n^2 \cdot \max_p(\boldsymbol{B}) \cdot \max_\varphi(\boldsymbol{B}))$ time steps, on a valid smart profile $\boldsymbol{B}$ in Bool.*

Proof. The **while** loop from line 3 in Unravel (see Algorithm 1) can be repeated at most n times (when a single vote is added to X at each iteration). Moreover, the **while** loop from line 6 can be repeated at most $\max_p(\boldsymbol{B})$ times when all smart ballots are of the same length and no vote is computable in the first $\max_p(\boldsymbol{B}) - 1$ iterations.

The following is executed at most $n \cdot \max_p(\boldsymbol{B})$ times. Unravel($\#$) checks or each agent a when $x_a = \Delta$ (at most $n-1$) if either $B_a^{\texttt{lev}} \in \mathcal{D}$ or $\varphi_a^{\texttt{lev}}$ has a necessary winner (depending on the update procedure used). As each $\varphi_a^{\texttt{lev}}$ is a complete DNF, to verify if it has a necessary winner, we check if either *(i)* all literals of a cube of $\varphi_a^{\texttt{lev}}$ are made true by $X \restriction_{S_a^{\texttt{lev}}}$, or *(ii)* one literal in each cube is made false by $X \restriction_{S_a^{\texttt{lev}}}$, returning a direct vote of 1 or 0, respectively, as described in Proposition 2.1.

The use of Unravel($\#$) takes at most $\mathcal{O}(n \cdot 2 \max_\varphi(\boldsymbol{B}))$ steps, which is equivalent to $\mathcal{O}(n \cdot \max_\varphi(\boldsymbol{B}))$ steps. Thus, Unravel($\#$) with $\# \in \{\mathbf{U}, \mathbf{DU}, \mathbf{RU}, \mathbf{DRU}\}$ yields a vector X of direct votes in $\mathcal{O}(n^2 \cdot \max_p(\boldsymbol{B}) \cdot \max_\varphi(\boldsymbol{B}))$ time steps. □

2.5 Comparing the Unravelling Procedures

In this section, we complement the results of Section 2.4, which analysed the computational complexity of our unravelling procedures by further distinguishing our unravelling procedures. This aims to better understand when a procedure would be preferable to another.

2.5.1 Restrictions Yielding Distinct or Identical Outcomes

We study here under which restrictions on the language of the ballots the outcomes of our unravelling procedures coincide or differ. First, we show that all unravelling procedures defined in Subsections 2.3.2 and 2.3.1 can give different outcomes, even when the ballots are restricted to Liquid.

Proposition 2.5. *The unravellings MinSum, MinMax, and Unravel($\#$), with $\# \in \{\mathbf{U}, \mathbf{DU}, \mathbf{RU}, \mathbf{DRU}\}$ can give different certificates and outcomes on the same smart profile $\boldsymbol{B}$ of Liquid ballots.*

Proof. Consider the Liquid profile $\boldsymbol{B}$ for the domain $\mathcal{D} = \{1, 0\}$ and the set of agents $\mathcal{N} = \{a, b, c, d\}$ presented in Table 2.2 (left). The outcomes of the unravelling procedures and their certificates are also shown in Table 2.2 (right). We do not show the outcomes of MinMax, as it returns all consistent certificates such that

	B^1_x	B^2_x	B^3_x	B^4_x
a	$(\{b\}, b)$	$(\{c\}, c)$	$(\{d\}, d)$	1
b	$(\{a\}, a)$	$(\{c\}, c)$	0	-
c	$(\{a\}, a)$	$(\{b\}, b)$	1	-
d	$(\{a\}, a)$	1	-	-

Procedure	Outcome	Certificate
U	$(1, 0, 1, 1)$	$(3, 3, 3, 2)$
DU	$(0, 0, 1, 1)$	$(1, 3, 3, 2)$
RU	$(1, 1, 1, 1)$	$(3, 1, 1, 2)$
	$(0, 0, 0, 1)$	$(1, 3, 1, 2)$
	$(1, 1, 1, 1)$	$(2, 1, 3, 2)$
	$(1, 1, 1, 1)$	$(1, 2, 3, 2)$
DRU	$(0, 0, 0, 1)$	$(1, 3, 1, 2)$
	$(1, 1, 1, 1)$	$(2, 1, 3, 2)$
	$(1, 1, 1, 1)$	$(1, 2, 3, 2)$
MinSum	$(0, 0, 0, 0)$	$(1, 3, 1, 1)$

Table 2.2: On the left, we show the profile $\boldsymbol{B}$ used in the proof of Proposition 2.5. On the right, the table shows the outcomes and certificates of unravelling profile $\boldsymbol{B}$ with the procedures Unravel(**U**), Unravel(**DU**), Unravel(**RU**), Unravel(**DRU**), and MinSum.

no entry is greater than 3; e.g., it will also include the certificate $\mathbf{c} = (3, 3, 2, 2)$ giving the outcome $(1, 0, 0, 1)$. Since the latter is not an outcome of the other procedures, MinMax differs from those. Moreover, while procedures Unravel(**RU**) and Unravel(**DRU**) give the same outcomes $(1, 1, 1, 1)$ and $(0, 0, 0, 1)$, they are returned at different rates. □

Given the previous result, there are restrictions on the ballots where the outcomes of the procedures coincide. We show that the outcome is the same for all our unravelling procedures, except for MinMax, when considering only Liquid$[1]_*$ ballots.

Proposition 2.6. *If $\boldsymbol{B} \in$ Liquid$[1]_*$, the procedures MinSum and Unravel($\#$) for $\# \in \{\mathbf{U}, \mathbf{DU}, \mathbf{RU}, \mathbf{DRU}\}$ give the same outcome X, although their certificates may differ.*

Proof. Unravel(**U**) and Unravel(**DU**) act in an identical manner for Liquid$[1]_*$ ballots. They first add all non-delegating agents' votes to X. Then, they iteratively unravel the first preference delegations of all agents not in a delegation cycle. Once no more votes can be added from the first preference level, i.e., there are agents in a delegation cycle, the remaining agents are assigned their second choice, an abstention $*$.

Unravel(**RU**) picks one agent at a time from the first preference level who either gives a direct vote or their delegate has a vote in X. When no more agents are available at the first preference level, the remaining agents are in delegation cycles. Moving to the second preference level, one of these agents is added with an abstention. Consequently, everyone caught in this delegation cycle also receives abstentions from the agent who was picked at random. This is repeated until all delegation cycles have been resolved and all agents have a vote in X.

Unravel(**DRU**) first adds the direct votes of the agents who do not delegate one by one. Then, it does the same for delegating agents whose delegate already has a vote in X. Once no more agents can be added with their first preference, the procedure adds a single random agent with an abstention (from their second preference). Then it continues as Unravel(**RU**) until all delegation cycles are resolved.

Finally, MinSum returns all outcomes that minimise the total rank. Therefore, all agents receive their first preference, except for a single agent from each delegation cycle, as in the previous unravellings. Observe that on any profile $\boldsymbol{B}$ in Liquid$[1]_*$, we have $\mathcal{C}_{\text{MinSum}}(\boldsymbol{B}) = \mathcal{C}_{\text{Unravel}(\mathbf{RU})}(\boldsymbol{B}) = \mathcal{C}_{\text{Unravel}(\mathbf{DRU})}(\boldsymbol{B})$. □

Remark 2.2. All our six unravelling procedures have the certificate $\mathbf{c} = \{1\}^n$ on Liquid$[1]_*$ profiles with no delegation cycles. However, if there are delegation cycles, MinMax returns many outcomes—including the one whose certificate gives all delegating agents their second preference ($*$), regardless of whether they are in a delegation cycle. Furthermore, Proposition 2.6 does not hold for Liquid[1], where backup votes are not restricted to $*$, as the tie-breaking affects the outcome X.

Remark 2.3. The breadth-first and depth-first rules by Kotsialou and Riley [2020] differ from all six of our unravellings. Consider $\mathcal{N} = \{a, b, c\}$, an issue with domain $\mathcal{D} = \{1, 0, *\}$, and agents' ballots as follows: $B_a = ((\{b\}, b) > (\{c\}, c) > *)$, $B_b = (*)$, and $B_c = (1)$. Our six unravelling procedures would return the outcome $(*, *, 1)$ with certificate $\mathbf{c} = (1, 1, 1)$, whereas the breadth-first and depth-first procedures would return the outcome $(1, *, 1)$ with certificate $\mathbf{c} = (2, 1, 1)$.

Next, we inspect the connection between our two procedures with the random voter selection property. We show that all possible outcomes of Unravel(**DRU**) are also possible outcomes of Unravel(**RU**), as the set of certificates of the former is a subset of the set of certificates of the latter.

Proposition 2.7. *The set of consistent certificates of unravelling $\boldsymbol{B}$ using Unravel(**DRU**) are also consistent certificates of Unravel(**RU**), $\mathcal{C}_{\text{Unravel}(\mathbf{RU})}(\boldsymbol{B}) \subseteq \mathcal{C}_{\text{Unravel}(\mathbf{DRU})}(\boldsymbol{B})$ for any valid smart profile $\boldsymbol{B}$.*

Proof. At any iteration of Unravel(**RU**), there is a random choice between agents, including direct voters. If there are direct voters, Unravel(**DRU**) has a subset of the choices of Unravel(**RU**) (and hence the potential outcomes and certificates). If there are no direct voters at an iteration, the potential delegation chosen at this step is the same for Unravel(**RU**) and Unravel(**DRU**). Thus, all certificates of Unravel(**DRU**) are also certificates of Unravel(**RU**). □

2.5.2 Participation Axioms

In this subsection, we study two properties of resolute unravelling procedures, focusing on a binary domain with abstentions $\mathcal{D} = \{0, 1, *\}$. Both properties were proposed by Kotsialou and Riley [2020] and focus on a voter's incentive to participate in the election, either by voting directly or by delegating, in line with the classical participation axiom from social choice (see, e.g., Moulin [1988]).

We assume that an agent a expressing a direct vote for $x \in \{0,1\}$ prefers the outcome x over both $1-x$ and an abstention. We denote this by $x >_a 1-x$ and $x >_a *$, respectively. Furthermore, we focus on resolute rules to directly compare the breadth-first and depth-first procedures to our own, as the participation axioms were originally constructed to study these procedures.

First, we distinguish between the unravelling procedures being *resolute*, as our greedy procedures, or *irresolute*, as our optimal procedures. A unique outcome is returned by the former, and by the latter, possibly many tied outcomes are returned. Although throughout the paper we present *all* outcomes of the greedy procedures with random voter selection **RU** and **DRU** (see, e.g., the outcomes displayed in Table 2.2), these procedures are resolute as defined in Algorithms 4 and 5, respectively.

Definition 2.10 (Cast-Participation)**.** A resolute voting rule r and a resolute unravelling procedure $\mathcal{U}$ satisfy *cast-participation* if for any valid smart profile $\boldsymbol{B}$ and any agent $a \in \mathcal{N}$ such that $B_a \in \mathcal{D} \setminus \{*\}$ we have that for any $B'_a \neq B_a$ that

$$r(\mathcal{U}(\boldsymbol{B})) \geq_a r(\mathcal{U}(\boldsymbol{B}_{-a}, B'_a))$$

where $\boldsymbol{B}_{-a}$ is equal to $\boldsymbol{B}$ without a's ballot. We require the inequality to hold for any possible outcome of $\mathcal{U}$ for randomised procedures.

Cast-participation implies that agents who vote directly are incentivised to do so rather than express any other ballot. In order to prove if a pair of an unravelling procedure and an aggregation rule satisfies such a participation axiom, we need some further notation. Let the set of voters *influenced* by a voter a in a profile $\boldsymbol{B}$ using a resolute deterministic unravelling procedure $\mathcal{U}$ be $I^{\mathcal{U}}(\boldsymbol{B}, a) = \{b \mid a \in S_b^k$ for $\mathcal{U}(\boldsymbol{B}) = X_{\mathbf{c}}$ with $\mathbf{c} \in \mathcal{C}(\boldsymbol{B})$ and $\mathbf{c}_b = k\}$. Further, let $I_*^{\mathcal{U}}(\boldsymbol{B}, a) = I^{\mathcal{U}}(\boldsymbol{B}, a) \cup \{c \mid c \in I^{\mathcal{U}}(\boldsymbol{B}, b) \wedge b \in I^{\mathcal{U}}(\boldsymbol{B}, a)\} \cup \ldots$ be the voters who are influenced by a both *directly* and *indirectly*.

Given the domain $\mathcal{D} = \{0,1,*\}$, we consider two rules. The *majority rule* (Maj) returns the alternative from the domain that has more than $n/2$ votes for it, and $*$ otherwise. The *relative majority rule* (RMaj) returns the plurality outcome in $\mathcal{D} \setminus \{*\}$, and if there is a tie, it returns $*$. A voting rule r on the domain $\{0,1,*\}^n$ satisfies *monotonicity* if for any profile X if $r(X) = x$ with $x \in \{0,1\}$ then $r(X_{+x}) = x$, where X_{+x} is obtained from X by making one voter switch from either an initial vote of $1-x$ to either x or $*$, or from an initial vote of $*$ to x. Observe that both Maj and RMaj satisfy monotonicity. Due to this definition, we can now show the following:[12]

[12]Note that Definition 2.10 slightly differs from the one given in previous work Colley et al. [2020], and thus Theorem 2.6 does not hold for **RU** or **DRU**: a counterexample can be constructed exploiting the fact that an agent may prefer the outcome of one random iteration of the procedure to another.

Theorem 2.6. *Any monotonic rule r with* Unravel($\#$) *for* $\# \in \{\mathbf{U}, \mathbf{DU}\}$ *satisfies cast-participation for* Liquid$_*$ *with domain* $\mathcal{D} = \{0, 1, *\}$.

Proof. Without loss of generality, assume that for agent $a \in \mathcal{N}$, we have $B_a = (1)$. To falsify cast-participation, we need to find a profile $\boldsymbol{B}$ with $r(\text{Unravel}(\#)(\boldsymbol{B})) = 0$ or $*$, and a ballot B'_a such that $r(\text{Unravel}(\#)(\boldsymbol{B}_{-a}, B'_a)) = 1$, for $\# \in \{\mathbf{U}, \mathbf{DU}\}$.

First, observe that all voters $c \in I^{\#}_*(\boldsymbol{B}, a)$ vote for 1 in $\boldsymbol{B}$, since the language is restricted to single-agent delegations. Now, if $B'_a = 0$ or $*$ (or they delegate to some agent who is assigned these votes), then by monotonicity, the result of $\boldsymbol{B}'$ will remain as 0 or $*$. Moreover, all $c \notin I^{\#}_*(\boldsymbol{B}, a)$ do not change their vote from $\boldsymbol{B}$ to $\boldsymbol{B}'$, whether B'_a is a direct vote or a possibly ranked delegation. Therefore, the final votes of $\boldsymbol{B}'$ can be obtained from those of $\boldsymbol{B}$ by switching 1s to 0s or $*$s. Thus, this contradicts the monotonicity assumption of rule r. $\square$

Remark 2.4. Theorem 2.6 does not hold for Bool ballots. Consider the counterexample with agents $\mathcal{N} = \{a, b, c\}$ voting on an issue with domain $\mathcal{D} = \{0, 1\}$ having ballots $B_a = (1)$, $B_b = ((\{a\}, \neg a) > 0)$ and $B_c = ((\{a\}, \neg a) > 0)$. Our greedy unravellings would return the outcome $Maj(1, 0, 0) = 0$. However, if $B'_a = 0$ then $Maj(0, 1, 1) = 1$. Thus, agent a strictly prefers to submit a ballot that is not a direct vote for their preferred alternative.

We now focus on the incentive a voter has to receive and accept delegations, building on the definition of the guru-participation property from Kotsialou and Riley [2020].

Definition 2.11 (Guru-participation)**.** A voting rule r and a resolute unravelling procedure $\mathcal{U}$ satisfy *guru-participation* if and only if for all profiles $\boldsymbol{B}$ and all agents $a \in \mathcal{N}$ such that $B_a = (x)$ with $x \in \mathcal{D} \setminus \{*\}$ we have that for any $b \in I^{\#}_*(\boldsymbol{B}, a)$

$$r(\mathcal{U}(\boldsymbol{B})) \geq_a r(\mathcal{U}(\boldsymbol{B}_{-b}, (*)))$$

where $\boldsymbol{B}_{-b}$ is $\boldsymbol{B}$ without b's ballot. We require the inequality to hold for any possible outcome of $\mathcal{U}$ for randomised procedures.

All four greedy unravellings do not satisfy this property when considering the final aggregation rule RMaj.

Theorem 2.7. *RMaj and* Unravel($\#$) *for* $\# \in \{\mathbf{U}, \mathbf{DU}, \mathbf{RU}, \mathbf{DRU}\}$ *do not satisfy guru-participation for* Liquid$_*$ *with domain* $\mathcal{D} = \{0, 1, *\}$.

Proof. Consider a smart profile $\boldsymbol{B}$, as shown on the left-hand-side of Table 2.3, and profile $\boldsymbol{B}' = (\boldsymbol{B}_{-b}, (*))$ obtained from $\boldsymbol{B}$ by switching b's vote to $B'_b = (*)$. The outcomes of the four procedures are shown on the right-hand side of Table 2.3.

By applying Unravel($\mathbf{U}$) and Unravel($\mathbf{DU}$), agent a prefers the outcome of $\boldsymbol{B}'$ to that of $\boldsymbol{B}$, since RMaj$(X^1) = *$ and RMaj$(X^2) = 1$. The outcome of $\boldsymbol{B}'$ when considering Unravel($\mathbf{RU}$) and Unravel($\mathbf{DRU}$) is RMaj$(X^2) = 1$. However, the outcome of $\boldsymbol{B}$ can be either RMaj$(X^4) =$ RMaj$(X^5) = 0$ or RMaj$(X^3) = 1$. Hence,

	B_x^1	B_x^2	B_x^3
a	1	-	-
b	$(\{c\}, id)$	$(\{a\}, id)$	$*$
c	$(\{d\}, id)$	$(\{f\}, id)$	$*$
d	$(\{b\}, id)$	$(\{f\}, id)$	$*$
e	1	-	-
f	0	-	-

#	$\boldsymbol{B}$	$\boldsymbol{B}'$
U/ **DU**	$X^1 = (1,1,0,0,1,0)$	$X^2 = (1,*,*,*,1,0)$
RU/ **DRU**	$X^3 = (1,1,1,1,1,0)$ $X^4 = (1,0,0,0,1,0)$ $X^5 = (1,0,0,0,1,0)$	$X^2 = (1,*,*,*,1,0)$

Table 2.3: A Liquid$_*$ profile $\boldsymbol{B}$ (on the left) and the outcomes of Unravel(**U**), Unravel(**DU**), Unravel(**RU**) and Unravel(**DRU**) on the profiles $\boldsymbol{B}$ and $\boldsymbol{B}'$ (on the right), where $\boldsymbol{B}' = (\boldsymbol{B}_{-b}, (*))$ is obtained from $\boldsymbol{B}$ by switching b's vote to $B'_b = (*)$.

when the random choice of **RU** or **DRU** leads to X^4 or X^5, agent a strictly prefers the outcome RMaj(X^2) to the outcome RMaj(X^4) and RMaj(X^5). Therefore, the inequality does not hold for any outcome of the randomised procedures. □

2.5.3 Pareto Dominance and Optimality

We now focus on comparing the outcomes of our unravelling procedures in terms of *Pareto dominance* and *Pareto optimality.* We show that none of our procedures Pareto dominate another on all profiles. However, we prove that all outcomes of MinSum are Pareto optimal with respect to all outcomes with consistent certificates.

A certificate $\mathbf{c}$ *weakly Pareto dominates* another certificate $\mathbf{c}'$ if for every $i \in \mathcal{N}$, we have that $\mathbf{c}_i \leq \mathbf{c}'_i$. We say that the unravelling procedure $\mathcal{U}$ weakly Pareto dominates another unravelling procedure $\mathcal{U}'$ if for any valid profile $\boldsymbol{B}$, all (possible) certificates $\mathbf{c}$ corresponding to outcomes of $\mathcal{U}(\boldsymbol{B})$ weakly Pareto dominate all the (possible) certificates $\mathbf{c}$ corresponding to the outcomes of $\mathcal{U}'(\boldsymbol{B})$. Note that the possibility of multiple certificates arises not only for irresolute procedures but also from different executions of the random procedures Unravel(**RU**) and Unravel(**DRU**).

Example 2.8. Consider the example given in Table 2.2. The certificate of the outcome of Unravel(**U**) is $\mathbf{c}^{\mathbf{U}} = (3,3,3,2)$, and that of the outcome of Unravel(**DU**) is $\mathbf{c}^{\mathbf{DU}} = (1,3,3,2)$. Thus, since $\mathbf{c}^{\mathbf{DU}}$ weakly Pareto dominates $\mathbf{c}^{\mathbf{U}}$, given that each entry of $\mathbf{c}^{\mathbf{DU}}$ is less than or equal to the corresponding entry in $\mathbf{c}^{\mathbf{U}}$. Moreover, since there is an outcome of Unravel(**RU**) with certificate $\mathbf{c}^{\mathbf{RU}} = (3,1,1,2)$, neither $\mathbf{c}^{\mathbf{DU}}$ weakly Pareto dominates $\mathbf{c}^{\mathbf{RU}}$ (as $\mathbf{c}_a^{\mathbf{DU}} < \mathbf{c}_a^{\mathbf{RU}}$ for the first agent a) nor vice-versa (as $\mathbf{c}_b^{\mathbf{DU}} > \mathbf{c}_b^{\mathbf{RU}}$ for the second agent b). △

We now give an example that shows when an unravelling procedure is weakly dominated by another.

Example 2.9. Consider the unravelling procedure $\mathcal{U}_d$ that returns every voter's final backup direct vote. Thus, $\mathcal{U}_d(\boldsymbol{B})$ returns the outcome with certificate $\mathbf{c}^d$, wherein

$\mathbf{c}_i^d$ is maximal for each $i \in \mathcal{N}$. Each of our six unravelling procedures weakly Pareto dominates $\mathcal{U}_d$, as they always produce a consistent certificate, which cannot have a higher entry than $\mathbf{c}_i^d$, for each agent $i \in \mathcal{N}$. △

When comparing our greedy procedures, one might think that the procedure with the additional properties (R and D) should be chosen over those without. However, the following example provides a profile where Unravel(**DU**), Unravel(**RU**) and Unravel(**DRU**) do not weakly Pareto dominate Unravel(**U**), and thus, they do not weakly Pareto dominate Unravel(**U**) in general.

	B_x^1	B_x^2	B_x^3
a	$(\{b,e\}, b \vee e)$	$(\{c,e\}, c \vee e)$	0
b	$(\{c,e\}, c \vee e)$	$(\{a,e\}, a \vee e)$	0
c	$(\{a,e\}, a \vee e)$	$(\{b,e\}, b \vee e)$	0
d	1	-	-
e	$(\{f\}, f)$	$(\{d\}, d)$	0
f	$(\{e\}, e)$	0	-

Table 2.4: A profile $\boldsymbol{B}$ showing that Unravel(**U**) is not dominated in general by Unravel(**DU**), Unravel(**RU**) or Unravel(**DRU**).

Example 2.10. Take agents $\mathcal{N} = \{a, b, c, d, e, f\}$, whose ballots are shown in Table 2.4. On this profile, Unravel(**U**) gives the outcome $X_{\mathbf{c}} = (1,1,1,1,1,0)$, where $\mathbf{c} = (1,1,1,1,2,2)$ and Unravel(**DU**) gives $X_{\mathbf{c}'} = (0,0,0,1,0,0)$, with certificate $\mathbf{c}' = (3,3,3,1,1,2)$. Thus, $\mathbf{c}$ does not weakly Pareto dominate $\mathbf{c}'$ as agent e's entries in the certificates are such that $\mathbf{c}_e < \mathbf{c}'_e$. It is also not the case that $\mathbf{c}'$ weakly Pareto dominates $\mathbf{c}$ as for some agents, for example, agent a, we have that $\mathbf{c}_a > \mathbf{c}'_a$. Hence, Unravel(**DU**) does not weakly Pareto dominate Unravel(**U**) or vice-versa.

Furthermore, a possible outcome of Unravel(**RU**) is $X_{\mathbf{c}''} = (0,0,0,1,0,0)$ where $\mathbf{c}'' = (3,1,1,1,1,2)$—the random choices pick f first and then a. Again, $\mathbf{c}''$ does not weakly Pareto dominate $\mathbf{c}$, as $\mathbf{c}_a > \mathbf{c}''_a$. Therefore, Unravel(**RU**) does not weakly Pareto dominate Unravel(**U**), and as $X_{\mathbf{c}''}$ is also an outcome of Unravel(**DRU**), Unravel(**DRU**) does not weakly Pareto dominate Unravel(**U**) as well. △

Proposition 2.8. *None of the four greedy unravelling procedures* Unravel(#) *for* $\# \in \{\mathbf{U}, \mathbf{DU}, \mathbf{RU}, \mathbf{DRU}\}$ *weakly Pareto dominates another greedy procedure.*

Proof. Example 2.10 shows that Unravel(**U**) is not weakly Pareto dominated by Unravel(#) for $\# \in \{\mathbf{DU}, \mathbf{RU}, \mathbf{DRU}\}$. Then, in Table 2.2 the outcome of Unravel(**U**) is weakly Pareto dominated by the outcomes of Unravel(#) for $\# \in \{\mathbf{DU}, \mathbf{RU}, \mathbf{DRU}\}$ and therefore, Unravel(**U**) does not weakly Pareto dominate the other greedy procedures.

From Table 2.2, we can also conclude that Unravel(**DRU**) does not weakly Pareto dominate Unravel(**DU**), and vice-versa. The outcome of Unravel(**DRU**) with certificate $\mathbf{c} = (2, 1, 3, 2)$ does not weakly Pareto dominate the outcome of Unravel(**DU**), having certificate $\mathbf{c}' = (1, 3, 3, 2)$, as $\mathbf{c}_a > \mathbf{c}'_a$. For the other direction, as $\mathbf{c}'_b > \mathbf{c}_b$ Unravel(**DU**) does not always weakly Pareto dominate Unravel(**DRU**). Since the outcome with $\mathbf{c} = (2, 1, 3, 2)$ is also possible for Unravel(**RU**), Unravel(**RU**) is not guaranteed to weakly Pareto dominate Unravel(**DU**) and vice-versa.

Finally, Unravel(**DRU**) and Unravel(**RU**) do not weakly Pareto dominate one another as the certificates of the former are a subset of the latter (Proposition 2.7). □

For irresolute procedures, by checking whether an unravelling procedure on any profile always has an outcome whose certificate weakly Pareto dominates the certificates of all the outcomes of another procedure, we find the following negative results:

- The certificates $\mathbf{c} = (4, 1, \ldots, 1)$ of MinSum and $\mathbf{c}' = (1, 2, \ldots, 2)$ of MinMax from Example 2.3 shows that neither $\mathbf{c}$ weakly Pareto dominates $\mathbf{c}'$ nor vice-versa. Therefore, neither MinMax nor MinSum dominates the other.
- From Table 2.2, we see that Unravel(**U**) does not weakly Pareto dominate MinSum or MinMax in general. From Example 2.3 we see that MinSum does not weakly Pareto dominate Unravel(**U**) in general.

Finally, we introduce the notion of *Pareto optimality*, which defines all those consistent certificates that are not Pareto dominated by any other consistent certificate.

Definition 2.12. A consistent certificate $\mathbf{c}$ of $\boldsymbol{B}$ is *Pareto optimal* with respect to all of the consistent certificates $\mathcal{C}(\boldsymbol{B})$ if there exists no $\mathbf{c}' \in \mathcal{C}(\boldsymbol{B})$ with $\mathbf{c}' \neq \mathbf{c}$, such that $\mathbf{c}'$ weakly Pareto dominates $\mathbf{c}$.

The following proposition corresponds to the well-known fact that maximising the average of a vector leads to a Pareto optimal vector, but not vice-versa.

Proposition 2.9. *The certificate $\mathbf{c}$ for any outcome $X_{\mathbf{c}} \in$ MinSum$(\boldsymbol{B})$ is Pareto optimal for $\mathcal{C}(\boldsymbol{B})$, for any valid profile $\boldsymbol{B}$.*

Proof. Take an arbitrary valid smart profile $\boldsymbol{B}$, and arbitrary $X_{\mathbf{c}} \in$ MinSum$(\boldsymbol{B})$. For the sake of a contradiction, assume that $\mathbf{c}$ is not Pareto optimal for $\mathcal{C}(\boldsymbol{B})$. Hence, there exists a $\mathbf{c}' \in \mathcal{C}(\boldsymbol{B}) \setminus \{\mathbf{c}\}$ such that $\mathbf{c}'$ weakly Pareto dominates $\mathbf{c}$. Therefore, for all $i \in \mathcal{N}$, we get $\mathbf{c}'_i \leq \mathbf{c}_i$, which gives us that $\sum\limits_{i \in \mathcal{N}} \mathbf{c}'_i \leq \sum\limits_{i \in \mathcal{N}} \mathbf{c}_i$. Furthermore, since $\mathbf{c} \neq \mathbf{c}'$ and $\mathbf{c}'$ weakly Pareto dominates $\mathbf{c}$, there exists an agent $j \in \mathcal{N}$ such that $\mathbf{c}'_j < \mathbf{c}_j$, and thus $\sum\limits_{i \in \mathcal{N}} \mathbf{c}'_i < \sum\limits_{i \in \mathcal{N}} \mathbf{c}_i$. Since $\sum\limits_{i \in \mathcal{N}} \mathbf{c}_i$ is not minimal, we have $X_{\mathbf{c}} \notin$ MinSum$(\boldsymbol{B})$, and we have reached a contradiction. □

The opposite direction of Proposition 2.9 does not hold. In Example 2.3, the MinSum procedure does not return the Pareto optimal certificate $\mathbf{c} = (1, 2, \ldots, 2)$. Moreover, the other unravelling procedures are not guaranteed to produce outcomes with Pareto optimal certificates, as there exist outcomes of each of them whose certificates are weakly Pareto dominated by some other consistent certificate, as seen in previous examples.

2.5.4 Discussion on the Choice of Unravelling Procedure

In this chapter, we have provided six unravelling procedures and have given results that should guide a user of this model as to which procedure to choose. In what follows, we provide a summary and a discussion of these results.

The main distinction between the optimal and greedy procedures is that finding an outcome with a greedy procedure is a tractable problem, whereas even checking if an outcome of an optimal procedure exists under a given bound on the optimised score is an NP-complete problem for the general language Bool (where the delegations are contingent formulas expressed in complete DNF). Although we acknowledge that the improvements in the performance of Sat-solvers make the intractability BoundedMinMax and BoundedMinSum less concerning, the associated search problem of computing the outcomes of the unravelling remains, in principle, even harder. Moreover, if the model were to be used on a large scale, then even the use of efficient Sat-solvers may not be efficient enough when the input size of the problem is large, i.e., the profile size increases with the number of agents. Hence, the greedy procedures are desirable when tractability is key.

Proposition 2.9 shows that the certificates of the outcomes of MinSum are Pareto optimal and thus are never dominated by outcomes found by a consistent certificate. In contrast, MinMax cannot make this guarantee. Although MinMax may return outcomes that are not Pareto optimal, it can provide more egalitarian outcomes. In Example 2.3, the outcome with the lowest `rank` relies on the fourth preference of agent a being chosen: while it is still a trusted delegate, the agent may be less confident in them than in their three previous delegations.

Furthermore, as MinSum and MinMax are irresolute, they would have to be paired with a tie-breaking mechanism to select a single outcome from the possibly many that they produce. In contrast, the greedy procedures are not only, in general, quicker than the optimal procedures, but they are also resolute.

With profiles of Liquid$[1]_*$ ballots, MinSum and the greedy procedures return the same outcome vector, and therefore, they can be used interchangeably. For profiles of Liquid ballots, there should be a preference for MinSum or MinMax, since an outcome can be found in polynomial time (Theorems 2.3 and 2.5). The choice between these two procedures should be determined by whether the situation would benefit more from Pareto optimality or egalitarian properties. However, these procedures rely on tie-breaking, which could bring up issues of fairness in the certificates.

As the participation axioms do not differentiate the greedy procedures, the properties they are defined on (i.e., direct vote priority and random voter selection) are the clearest way to compare them. Random voter selection should be used when a lottery is acceptable, yet it should be avoided when it would be unfair to give a worse preference level to just some agents. Direct vote priority should be used when a direct vote from an agent is preferred to a delegation, perhaps in situations that could benefit from a level of expertise on the issue, and to ensure shorter delegation chains.

Given the above discussion, one may think that UNRAVEL(**DRU**) gives the best outcomes overall. However, we have proved that no greedy procedure is guaranteed to Pareto dominate another (Proposition 2.8). Thus, the notion of Pareto dominance does not distinguish between greedy procedures.

To summarise, greedy procedures should be preferred when finding outcomes need to be done tractably, except in the special case of LIQUID ballots, for which this problem is polynomial for all proposed rules. The MINSUM procedure should be used when outcomes need to be Pareto optimal and MINMAX should be used when an egalitarian approach is required. When using the greedy procedures, the choice between them should be determined by whether the situation asks for either random voter selection or direct vote priority properties.

2.6 Conclusion and Future Work

In this chapter, we proposed a model of multi-agent ranked delegations in voting, which generalises the standard model of liquid democracy in two aspects, both making the ballots in the model more expressive.

First, delegations can involve many agents instead of a single agent who determines their vote. We introduced a general language named BOOL, in which delegations are expressed as contingent propositional formulas in complete DNF. We emphasise that although agents may not want to use the full expressivity of the language, they are free to use it as much as they desire, and many natural delegation types are captured by it. For example, both liquid democracy delegations and delegations using threshold rules can be expressed in BOOL. More elaborate voting models, such as smart voting, can be used to implement voting systems digitally and utilise computer technology. Moreover, other areas of AI could help the voters use the full expressiveness of the delegations in smart voting. For instance, it may not be possible for most community members to create Boolean functions as delegations; however, using a natural language processing model could aid them with this. Thus, it does not seem too far-fetched that voters would trust a delegation formula returned by a large language model prompted by text from the voter to create a complex delegation. In this case, the flexibility of a language such as BOOL could be utilised.

The second way our model is more expressive is by allowing ranked delegations. As transitive delegations can lead to delegation cycles among the agents' most preferred delegates, the linear order of trusted delegations given by the agents can be used to break these cycles.

Our main contribution is the definition and study of six unravelling procedures: two optimal procedures looking to find outcomes that minimise some given criterion and four that greedily find outcomes. They each take a profile of smart ballots and return a standard voting profile. We show that all of the procedures can give different certificates and outcomes (Proposition 2.5) and that they differ from the breadth-first and depth-first procedures of Kotsialou and Riley [2020] (Remark 2.3). Moreover, we show that all of the procedures, except MinMax, coincide on Liquid$[1]_*$ ballots, i.e., classical liquid democracy ballots with a single delegation per agent. The certificates of the outcomes of MinSum are Pareto optimal with respect to the set of consistent certificates (Proposition 2.9), while greedy procedures do not Pareto dominate one another (Proposition 2.8). Our main results show that decision variants of MinSum and MinMax are NP-complete problems (Theorems 2.2 and 2.4) over the general language Bool. Still, they become tractable when ballots are restricted to Liquid, the language of ranked liquid democracy (Theorems 2.3 and 2.5). Finally, we prove that our four greedy unravelling procedures always terminate (Proposition 2.3) and do so in a polynomial number of time steps for general Bool ballots (Proposition 2.4).

Future Work Since this work was conducted, some of its open problems have been studied. Tyrovolas [2022] gave more complexity results on the procedures. One interesting extension they explored was the combination of our procedures MinSum and MinMax, returning the outcome of MinMax, which had the lowest `rank`, showing that the bounded decision version of this problem is also NP-hard. Moreover, preliminary experimental investigations from Kulesza [2022] have already been into our six unravelling procedures. They conducted an online experiment by creating an online voting platform to analyse the truth-tracking power of liquid democracy with respect to the different unravelling procedures. On their online platform, they focused on the language Liquid.

There is still a lot of analysis to be done on the model. One possibility would be to look at the voting power of the agents in the model. Zhang and Grossi [2022] studied a version of the Banzhaf index in liquid democracy given a delegation graph. In Chapter 5, we will study the a priori voting power of simple liquid democracy. Yet neither approach has been explored on voting models with more complex delegations. Moreover, another extension of this work would study multi-agent ranked delegations on multiple interconnected issues, in line with Chapter 3.

Finally, full implementation of the smart voting model would be an exciting line of research, following the preliminary work of Kulesza [2022]. This would mean creating an online voting platform allowing voters to use full Bool ballots. A major challenge of this line of research would be how to effectively present such a voting model to voters in a user-friendly way. From such a platform, studying cycles and using delegations would be extremely fruitful in guiding future research needed to implement models.

Chapter 3

Preserving Consistency in Multi-Issue Liquid Democracy

3.1 Introduction

This chapter takes on a different aspect of extending delegative democracy from classical liquid democracy. Unlike in the previous chapter, we are no longer interested in extending the delegations to consider more complex ones. Instead, we want to extend the model to account for multiple interconnected issues. Our model builds on those of Brill and Talmon [2018] and Jain et al. [2022]. Their models study liquid democracy with many interconnected issues on specific domains in which the consistency of every agent's ballot must be upheld. Brill and Talmon [2018] studies liquid democracy where each issue is a pairwise comparison between two alternatives. After resolving the delegations, every agent's ballot must contain a complete strict ordering over the alternatives and the resulting preferences cannot contain cycles. Jain et al. [2022] studied a different setting, namely liquid knapsack voting, where the issues are projects to be accepted and their cost must be within the budget. In this model, every voter's ballot must not exceed the budget after resolving the delegations. We study a more general model, considering general constraints connecting the multiple issues and deciding what is rational in each setting. This area of research explores the idea that through liquid democracy any collective decision mechanism can be "liquidised" — to borrow a term from Brill and Talmon [2018], i.e., to include transitive delegations into a voting model.

In each of these models, delegations are given on particular issues. Therefore, an agent can delegate one decision on an issue to one agent while delegating on a different issue to a different agent. Hence, after delegations have been resolved, the returned set of votes may not be rational in this setting, i.e., not consistent with the rationality constraint.

Example 3.1. Consider a group of agents who are voting on a set of issues that reflect projects to be accepted. These two projects could be: building a local park p and a fence being built around the park f. The agents' rationality in this setting could be that a fence cannot be built around the park if it has not been built. Hence, the rationality constraint may insist that every agent's final vote is consistent with the following formula: $f \to p$ (if the fence is accepted, then the park must also be accepted). Agent A may decide that agent B is the best suited to decide if the park is built and delegate to them on issue p. However, agent A may believe that agent C is the best delegate to decide on the fence f. Thus, given A's delegations, their vote will be inconsistent with the rationality constraint as B is against the park being built while C is for the fence being built. $\triangle$

In many of the models of liquid democracy, we run into consistency problems. One strategy to solve these problems is to use algorithmic techniques to solve the problem optimally with respect to some parameters. In the previous work on preserving consistency in multi-issue liquid democracy, this parameter is the number of changes made to the original profile to ensure that the resulting profile is consistent. Optimisation-based approaches proposed in previous work run into high computational costs, and approximation algorithms may be hard to defend in a social choice setting where fairness criteria and explanatory power of the outcome are of primary importance.

We propose to elicit a voter's priorities over the issues on which they are delegating and feed them to tractable rules that maintain ballot consistency. Such an elicitation can easily be performed, for example, using the order in which voters enter their delegations in the voting platform, a global order of priority such as by the cost of implementing the projects, or a personal priority order given directly from the voters. In Example 3.1, A may decide that the issue of the park being built has a higher priority than the fence being built. We show that using priority ordering over the issues in the procedure can lead to outcomes being found tractably.

3.1.1 Contribution

The main contribution of this chapter is extending the modelling of liquid democracy on multiple interconnected issues and analysing how inconsistencies in the agents' ballots should be resolved. This chapter is based on the work of Colley and Grandi [2022a]. We describe this model in Section 3.2. Section 3.3 then introduces our two procedures that minimise the required changes so that every agent's resulting votes are consistent. One of these procedures makes minimal changes to the profile such that when the delegations are resolved, the resulting votes are consistent; this procedure was previously studied in specific settings. The second procedure is novel. It resolves all the delegations and then directly changes the votes to regain consistency. We show that these procedures are, unsurprisingly, intractable. Given their intractability, we then search for different ways in which we can regain consistency. In Section 3.4, we give our solution to elicit the agents' priorities over the issues and use them to find consistent votes tractably and in a principled way. We give two procedures that use the agents' priorities, built with the same distinction between the minimisation procedures, with one changing the profile and the other changing the votes after resolving delegations. Next, we analyse the four procedures, showing that the priority procedures do not approximate their minimisation counterparts well (Section 3.4.2) and compare the procedures on how well they respect the priorities (Section 3.4.3). Section 3.5 focuses on knapsack constraints, as these constraints have the property that a procedure would never turn an issue's rejection into an acceptance. Thus, we study how the procedures affect the number of acceptances in the final votes after inconsistencies have been resolved. Section 3.6 extends the original work [Colley and Grandi, 2022a]. It explores a more general version of this model where either delegation cycles are allowed or every agent can have their own personal rationality constraint.

3.1.2 Related Work

The seminal work of Christoff and Grossi [2017a] already considered a "liquidised" version of binary aggregation on interdependent issues. This chapter builds on this, generalising and unifying the approach of Jain et al. [2021] and Brill and Talmon [2018], where the former pertains to approved projects respecting a budget and the latter to accepted pairwise comparisons being transitive. More precisely, we study the problem of liquid democracy over multiple binary issues where the final opinions must satisfy a constraint. Another model considering a liquid version of preference aggregation is that of Harding [2022]. However, this work guides agents in choosing a proxy that is consistent with their views. Although we allow for multiple delegations, these should be thought of as parallel delegations rather than a single delegation involving multiple agents on a single issue. For example, the smart voting model introduced in Chapter 2, in ranked liquid democracy [Kotsialou and Riley, 2020, Brill et al., 2022], spreading fractional power among delegates [Degrave, 2014], or submitting a subset of acceptable delegates [Dey et al., 2021, Gölz et al., 2018]. As already observed by Christoff and Grossi [2017a] liquid democracy relates strongly to opinion diffusion, where a delegation can be interpreted as an influence link between agents. Constraints in opinion diffusion have been considered by Friedkin et al. [2016] for real-valued beliefs and by Botan et al. [2019] for majoritarian updates on binary issues, yet the connection between the models will be explored more in Chapter 4.

3.2 The Model

We model a group of agents $\mathcal{N} = \{1, \cdots, n\}$ making a collective decision on a set of issues $\mathcal{I} = \{1, \cdots, m\}$, where the domain of each issue $j \in \mathcal{I}$ is $\mathcal{D}(j)$. Without loss of generality, we will consider binary issues, thus $\mathcal{D}(j) = \{0, 1\}$ for all $j \in \mathcal{I}$.[1] Each agent $i \in \mathcal{N}$ submits a ballot $B_i \in \Pi_{j\in\mathcal{I}}(\mathcal{D}(j) \cup \mathcal{N}\backslash\{i\})$, which specifies for each issue $j \in \mathcal{I}$ if the agent gives a direct vote in $\mathcal{D}(j)$ or a delegation to any other agent in $\mathcal{N}\backslash\{i\}$. A profile of ballots is denoted by $\boldsymbol{B} = (B_1, \cdots, B_n)$ and the ballot of agent i on issue j in profile $\boldsymbol{B}$ is denoted as B_{ij}. We let $\hat{\boldsymbol{B}}$ be an $n \times m$ matrix containing only the direct votes of the agents in profile $\boldsymbol{B}$, where $\hat{B}_{ij} = B_{ij}$ if $B_{ij} \in \{0, 1\}$ and $\hat{B}_{ij} = \Delta$, otherwise. We let $\boldsymbol{X} \in \{0, 1\}^{n \times m}$ denote a profile of votes.

Example 3.2. Consider two agents $\mathcal{N} = \{C, D\}$, two issues $\mathcal{I} = \{j, k\}$, and a profile $\boldsymbol{B}$ composed of ballots $B_C = (1, D)$ and $B_D = (C, 1)$, where C delegates to D

[1]Non-binary issues can be expressed in the binary setting by letting each alternative become a binary issue where the constraint permits only one alternative per issue to be chosen.

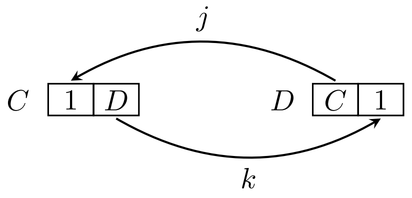


Figure 3.1: The delegation graph $G_{\boldsymbol{B}}$ of the profile given in Example 3.2.

on issue k, and D delegates to C on issue j. The direct votes of the agents are $\hat{\boldsymbol{B}} = ((1, \Delta), (\Delta, 1))$. $\triangle$

We assume there are no delegation cycles on any given issue in the first part of the chapter (we remove this assumption in Section 3.6 and examine its effect on the model). To clarify this, we associate a labelled delegation graph $G_{\boldsymbol{B}} = (V, E)$ with each profile $\boldsymbol{B}$, where $V = \mathcal{N}$ and edge $(C, D, j) \in E$ if C delegates to D on issue j in $\boldsymbol{B}$ (where j is the label of the edge). See Figure 3.1 for a profile of ballots given in Example 3.2. We assume that $G_{\boldsymbol{B}}$ is acyclic for any issue, i.e., for each $(C, D, j) \in E$, there is no j-path from C returning to C. Notice that in Figure 3.1 that there is a cycle in the edges, i.e., an edge from C to D and another from D to C, there is no cycle for a given issue. This would be the case if C's delegation to D were on issue j instead. This allows us to first focus on preserving consistency without making assumptions on how delegation cycles are broken, similarly to the previous work [Brill and Talmon, 2018, Jain et al., 2021].

Thanks to the assumption of acyclicity, we can define a *canonical profile of votes* $\boldsymbol{X}^{\boldsymbol{B}} \in \{0, 1\}^{n \times m}$ for every $\boldsymbol{B}$, where all delegating voters are assigned the vote of the direct voter at the end of the delegation chain (sometimes known as a *guru* or an *ultimate delegate*). Given a delegation graph $G_{\boldsymbol{B}}$, for each $i \in \mathcal{N}$ and $j \in \mathcal{I}$, we find the endpoint $k \in \mathcal{N}$ of the longest outgoing j-path from i and let $X^{\boldsymbol{B}}_{ij} = \hat{B}_{kj}$.

We assume that the issues in $\mathcal{I}$ are interconnected, i.e., that the votes of the agents have to respect a given constraint or set of constraints Γ. For instance, in knapsack voting, agents have to respect a given budget in the set of approved projects, or in preference aggregation, the approved comparisons are assumed to be transitive.

If $X_i \in \{0, 1\}^m$, we denote with $X_i \models \Gamma$ that agent i's vote satisfies the constraint Γ. If $\boldsymbol{X}$ is a profile of votes, we write $\boldsymbol{X} \models \Gamma$ when $X_i \models \Gamma$ for all $i \in \mathcal{N}$. The set of all consistent votes is denoted by $\mathcal{X}_\Gamma = \{X \mid X \models \Gamma\}$, assuming that Γ is not a contradiction, i.e. $\mathcal{X}_\Gamma \neq \emptyset$. We allow for any representation of Γ, with the only restriction being that checking if a partial vote can be completed to a consistent one should be feasible in polynomial time.

For instance, budget constraints, logical formulas in complete DNF, and the transitivity of preferences would all be acceptable constraints. Constraints expressed as arbitrary logical formulas do not, as the problem of completing partial evaluations is equivalent to SAT. Formally, if $\overline{X} \in \{0, 1, \Delta\}^m$ is a partial vote, we write $\overline{X} \approx \Gamma$ if there exists a complete vote $X \in \{0, 1\}^m$ such that $\overline{X} \subseteq X$ and

$X \models \Gamma$.[2] We assume that every agent's (partial) direct votes can be completed, i.e., for all $i \in \mathcal{N}$ we assume $\hat{B}_i \approx \Gamma$.

In line with previous work, we say that a profile of ballots $\boldsymbol{B}$ is *consistent* if $\boldsymbol{X}^{\boldsymbol{B}} \models \Gamma$.[3] From our assumptions that delegations are acyclic and the membership problem of Γ is tractable, we obtain that:

Remark 3.1. To check if a profile $\boldsymbol{B}$ is consistent is polynomial-time solvable.

As profiles may not be consistent, we define consistent delegation rules that take a profile of ballots $\boldsymbol{B}$ and return a profile of consistent votes, preserving the voters' direct votes:

Definition 3.1. Given a constraint Γ, a *consistent delegation rule* $\mathcal{F}$ takes a profile of ballots $\boldsymbol{B}$ and returns a profile of votes such that $\mathcal{F}(\boldsymbol{B}) \models \Gamma$ and $\hat{\boldsymbol{B}} \subseteq \mathcal{F}(\boldsymbol{B})$.

Example 3.3. Consider the same setting as Example 3.2, with the constraint that j and k cannot both be accepted. Thus, Γ can be represented as $\neg j \vee \neg k$ (a logical formula in complete DNF) and $\mathcal{X}_\Gamma = \{(0,0),(0,1),(1,0)\}$. Following the agents' delegations in $\boldsymbol{B}$, the canonical profile of votes is $\boldsymbol{X}^{\boldsymbol{B}} = ((1,1),(1,1))$. Thus, $\boldsymbol{B}$ is not consistent. Observe that any consistent delegation rule would give the outcome $((1,0),(0,1))$ (where the direct votes are not changed). $\triangle$

3.3 Minimal Changes to Ballots and Votes

In this section, we introduce two consistent delegation rules that preserve vote consistency by making minimal changes, showing that associated decision problems are NP-complete.

The first approach is to modify the profile of ballots $\boldsymbol{B}$ by ignoring a minimal number of delegations that conflict with the constraint, replacing them with a direct vote. This rule has been defined in previous work on multi-issue liquid democracy for the specific settings of knapsack voting [Jain et al., 2021] and preference aggregation [Brill and Talmon, 2018]. Given a profile $\boldsymbol{B}$, the *minimal delegation change rule* `MDC` finds all consistent profiles $\boldsymbol{B}'$ that replace a minimal number of delegations with direct votes and whose canonical profile of votes $\boldsymbol{X}^{\boldsymbol{B}'}$ is consistent with Γ:

$$\texttt{MDC}(\boldsymbol{B}) = \{\boldsymbol{X}^{\boldsymbol{B}'} | \boldsymbol{B}' \in \underset{\{\boldsymbol{B}' | \hat{\boldsymbol{B}} \subseteq \boldsymbol{X}^{\boldsymbol{B}'} \text{ \& } \boldsymbol{X}^{\boldsymbol{B}'} \models \Gamma\}}{\arg\max} \sum_{i \in \mathcal{N}} |\boldsymbol{B}_i \cap \boldsymbol{B}'_i|\}$$

We show that the following decision problem associated with `MDC` is NP-complete.

[2]Here we slightly abuse the subset notation, for a partial assignment $\overline{X} \in \{0,1,\Delta\}^m$ and assignment $X \in \{0,1\}^m$, we say $\overline{X} \subseteq X$ when X is found by only changing Δs in $\overline{X}$ to 0 or 1.

[3]Brill and Talmon [2018] also define a weaker notion of consistency. Both are equivalent to our definition of acyclic profiles.

MinimalDelChange	
Given:	a profile $\boldsymbol{B}$, constraint Γ and integer $k \geq 0$
Question:	is there a consistent profile $\boldsymbol{B}'$ found by changing at most k delegations of $\boldsymbol{B}$ to direct votes?

Proposition 3.1. *MinimalDelChange is NP-complete.*

Proof. A sub-problem of MinimalDelChange is the consistent knapsack voting problem CKV from Jain et al. [2021] (they also consider acyclic delegation graphs, as CKV removes cycles by replacing every delegation in a cycle with a direct vote against the issue). CKV is an NP-complete problem, as shown by Jain et al. [2021] (Theorem 1), implying that MinimalDelChange is NP-hard. To see that MinimalDelChange is in NP, consider a certificate that lists agents and issues, indicating delegations to be changed to direct votes to obtain $\boldsymbol{B}'$ from $\boldsymbol{B}$. First, we check that the list has at most k entries. Then we check whether $\boldsymbol{B}'$ is consistent, which can be done in polynomial time by Remark 3.1. □

The second approach, loosely inspired by related literature on judgment aggregation (see, e.g., Lang et al. [2011]), makes minimal changes to the canonical profile of votes $\boldsymbol{X}^{\boldsymbol{B}}$ directly, but only on issues where the voter expressed a delegation. The *minimal vote change rule* `MVC` is defined as follows:

$$\texttt{MVC}(\boldsymbol{B}) = \underset{\{\boldsymbol{X} \mid \hat{\boldsymbol{B}} \subseteq \boldsymbol{X} \text{ and } \boldsymbol{X} \models \Gamma\}}{\arg\max} \sum_{i \in \mathcal{N}} |X_i \cap X_i^{\boldsymbol{B}}|$$

In line with Proposition 3.1, we now show that the following decision problem associated with `MVC` is NP-complete.

MinimalVoteChange	
Given:	A profile $\boldsymbol{B}$, constraint Γ, and integer $k \geq 0$
Question:	Is there an $\boldsymbol{X} \models \Gamma$ such that $\hat{\boldsymbol{B}} \subseteq \boldsymbol{X}$ and $\boldsymbol{X}$ is found by changing at most k votes in $\boldsymbol{X}^{\boldsymbol{B}}$?

Proposition 3.2. *MinimalVoteChange is NP-complete.*

Proof. For membership in NP, we take a certificate that lists pairs of agents and issues to be changed in $\boldsymbol{X}^{\boldsymbol{B}}$ to define $\boldsymbol{X}'$. We then check in polynomial time that the list has at most k entries and that $X_i' \models \Gamma$ for each $i \in \mathcal{N}$.

To show NP-hardness, we reduce from the NP-complete problem independent set IndSet [Karp, 1972]. The input of IndSet is a graph $G = (V, E)$ and integer $t \geq 0$. It asks if there is a $V' \subseteq V$ with $|V'| \geq t$ such that no edge in E has both ends in V'. Given an instance of IndSet, consider a set of agents $\mathcal{N} = \{A_v \mid \text{for all } v \in V\} \cup \{A\}$, issues $\mathcal{I} = \{I_v \mid v \in V\}$, and bound $k = |V| - t$. Next, our constraint only allows for independent sets: $\Gamma_{\text{IND}} = \{(I_u \to \neg I_v) \wedge (I_v \to \neg I_u) \mid (u, v) \in E\}$. First, observe that we can check in polynomial time if a partial assignment can be completed to one satisfying Γ_{IND}, as we only need to check that the vertices accepted in the partial assignment are an independent set. Finally, consider a profile of ballots $\boldsymbol{B}$ such that agent A delegates to agent A_v on issue I_v for all $v \in V$, and each A_v only approves of I_v, rejecting all other issues.

Assume there is an independent set $V' \subseteq V$ with $|V'| \geq t$. Given that $X_A^{\boldsymbol{B}}=(1,\cdots,1)$, consider X_A containing votes for the issues I_v with $v \in V'$ and against the remaining issues. The resulting profile of votes is consistent: for each $v \in V$, X_{A_v} is consistent as only one issue is accepted, and $X_A \models \Gamma_{\text{IND}}$ as X_A reflects an independent set. Thus, we also have a solution to our problem as the number of votes changed is $|V| - |V'| \leq |V| - t = k$. Next, assume that there is no independent set of size at least t, and suppose $t' < t$ is the size of the largest independent set of G. Observe that to make $X_A^{\boldsymbol{B}}$ consistent, at least $|V| - t'$ votes need to be reverted. As $|V|-t' > |V|-t = k$, there is no solution to our problem either. Hence, MinimalVoteChange is NP-hard, thus NP-complete. □

3.4 Eliciting and Applying Priorities over Issues

To provide tractable and principled rules to preserve vote consistency, we propose to elicit from the voters their priorities over the issues they choose to delegate on—this could be, e.g., the order in which the voter expressed their delegations. In doing so, we discard a fixed parameter tractability analysis of `MDC` and `MVC` since Jain et al. [2021] have already provided extensive negative results for this approach.[4]

Example 3.4. Consider $\mathcal{N} = \{C, D, E\}$ voting on $\mathcal{I} = \{j, k\}$ where at most one issue can be chosen, i.e., $\Gamma = \neg j \vee \neg k$. C and D vote directly as such $B_C = (1, 0)$ and $B_D = (0, 1)$, whereas E delegates on both issues as such $B_E = (C, D)$. One of E's delegations will be ignored as $X_E^{\boldsymbol{B}} = (1, 1)$ is inconsistent. Any consistent delegation rule must decide which delegation to ignore, returning either $(1, 0)$ or $(0, 1)$. If E prioritises issue j, they should prefer $(1, 0)$ as their final votes. △

We assume that voters specify a total ordering over the issues on which they express a delegation. To simplify the presentation, we assume these orderings are on the whole set of issues $\mathcal{I}$ (the two assumptions are equivalent since consistent delegation rules never change direct votes). We denote this order by $\prec_i$ for each agent $i \in \mathcal{N}$ and write $\prec_i (k)$ for the issue with the k^{th} highest priority in $\prec_i$.

In parallel with the minimisation rules presented in Section 3.3, we present two approaches based on changing delegations or votes. First, the *priority delegation changing rule* (`PDC`) described in Algorithm 6 iteratively alters an agent's delegation to a direct vote if its addition is inconsistent. Second, the *priority vote changing rule* (`PVC`) described in Algorithm 7 iteratively adds consistent votes from $\boldsymbol{X^B}$ following the agents' order of priorities over the issues. We now define some notation used in

[4]Jain et al. [2021, Theorem 1] show that their version of `MDC` is tractable only under restrictive conditions such as when voters delegate on at most one issue. Moreover, their problem is NP-complete even when restricting many parameters to be small constants, such as the number of issues, the number of delegations, the issues having equal weights, and only one issue being accepted.

Algorithm 6 Priority delegation changing (PDC)

```
1: Input: $\boldsymbol{B}$, $\prec_i$ for all $i \in \mathcal{N}$
2: $t = 0$                                                  ▷ Set the counter to 0
3: Find $\boldsymbol{X}^0 = \hat{\boldsymbol{B}}$              ▷ Let $\boldsymbol{X}^0$ be the direct votes of $\boldsymbol{B}$
4: while $\boldsymbol{X}^t \notin \{0,1\}^{n \times m}$ do     ▷ loop until all agents have voters on each issue
5:     $\boldsymbol{X}^{t+1} := \boldsymbol{X}^t$
6:     for $i \in \mathcal{N}$ and $k \in [1, m]$ do           ▷ for every agent and priority level
7:         $j := \prec_i (k)$                                   ▷ let $j$ be $i$'s kth priority issue
8:         if $\texttt{vote}(\boldsymbol{X}^t, i, j) = \Delta$ and $\texttt{vote}(\boldsymbol{X}^t, B_{ij}, j) \in \{0,1\}$ then   ▷ if no vote is recorded for $i$ on $j$ but $i$'s delegate does
9:             if $(\texttt{bal}(\boldsymbol{X}^{t+1}, i)_{-j}, \texttt{vote}(\boldsymbol{X}^t, B_{ij}, j)) \approx \Gamma$ then $\texttt{vote}(\boldsymbol{X}^{t+1}, i, j) := \texttt{vote}(\boldsymbol{X}^t, B_{ij}, j)$   ▷ update the vote if it is consistent
10:            else $\texttt{vote}(\boldsymbol{X}^{t+1}, i, j) := 1 - \texttt{vote}(\boldsymbol{X}^t, B_{ij}, j)$   ▷ else update the opposite vote
11: $\boldsymbol{X} := \boldsymbol{X}^t$
```

the algorithms. Recall that $\hat{\boldsymbol{B}}$ denotes an $n \times m$ matrix containing only the direct votes of $\boldsymbol{B}$. We let $\texttt{bal}(\boldsymbol{X}, i)$ denote the vector of votes recorded for agent i in matrix $\boldsymbol{X}$ and $\texttt{vote}(\boldsymbol{X}, i, j)$ denote the vote recorded in $\boldsymbol{X}$ for agent i on issue j. Furthermore, (X_{-j}, x_j) denotes the vector X with the entry for issue j appended with a new entry x_j.

Algorithm 6 inputs the profile $\boldsymbol{B}$ and the agents' priorities over the issues $\prec_i$ and then sets the counter t to 0. On line 4, the algorithm starts a `while`-loop that continues until every agent has a vote in $\{0, 1\}$ for every issue. Each iteration of the `while`-loop checks whether an update can be made for each agent following their priority order over the issues. In the current $\boldsymbol{X}^t$, if i does not have a vote for j and their delegate does (on line 8), we check if adding their delegate's vote is consistent. If so we update X_i^{t+1} with the consistent vote (line 9). Otherwise, we add the opposite vote (line 10). Note that the update to $\boldsymbol{X}^{t+1}$ is simultaneous as it uses information from $\boldsymbol{X}^t$. Hence, the order of agents in the `for`-loop does not affect the outcome.

Algorithm 7 Priority vote changing (PVC)

```
1: Input: $\boldsymbol{B}$, $\prec_i$ for all $i \in \mathcal{N}$
2: Find $\boldsymbol{X}^0 = \hat{\boldsymbol{B}}$
3: Find $\boldsymbol{X}^{\boldsymbol{B}}$
4: for $i \in \mathcal{N}$ and $k \in [1, m]$ do               ▷ for every agent and priority level
5:     $j := \prec_i (k)$                                       ▷ let $j$ be $i$'s kth priority issue
6:     if $\texttt{vote}(\boldsymbol{X}^0, i, j) = \Delta$ then   ▷ if $i$ delegates on issue $j$
7:         if $(\texttt{bal}(\boldsymbol{X}^0, i)_{-j}, \texttt{vote}(\boldsymbol{X}^0, X^{\boldsymbol{B}}_{ij}, j)) \approx \Gamma$ then $\texttt{vote}(\boldsymbol{X}^0, i, j) := \texttt{vote}(\boldsymbol{X}^0, X^{\boldsymbol{B}}_{ij}, i)$   ▷ update the vote if it is consistent
8:         else $\texttt{vote}(\boldsymbol{X}^0, i, j) := 1 - \texttt{vote}(\boldsymbol{X}^0, X^{\boldsymbol{B}}_{ij}, j)$   ▷ else update the opposite vote
9: $\boldsymbol{X} := \boldsymbol{X}^0$
```

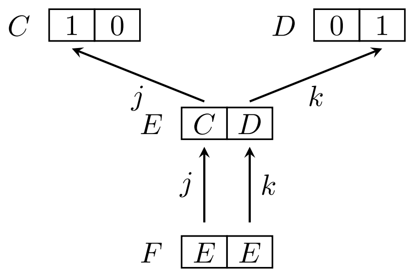


Figure 3.2: The delegation graph $G_{\boldsymbol{B}}$ of the profile given in Example 3.5.

Algorithm 7 describes PVC, it inputs the profile $\boldsymbol{B}$ and the agents' priorities. It then finds $\hat{\boldsymbol{B}}$ and the canonical profile of votes $\boldsymbol{X}^{\boldsymbol{B}}$. For each agent $i \in \mathcal{N}$ it checks in the priority order $\prec_i$ (line 4) if they have a vote recorded in $\boldsymbol{X}^0$ (line 6). If the addition of $X_{ij}^{\boldsymbol{B}}$ is consistent, we update X_i^0 with it (line 7) If $X_{ij}^{\boldsymbol{B}}$ is inconsistent with their current vote, we update the vector with the opposite (line 8).

Example 3.5. Consider agents $\mathcal{N} = \{C, D, E, F\}$ and issues $\mathcal{I} = \{j, k\}$ where at most one issue can be accepted, i.e., $\Gamma = \neg j \vee \neg k$. Consider profile $\boldsymbol{B}$ in which agents C and D vote directly, $B_C = (1, 0)$ and $B_D = (0, 1)$, whereas the remaining agents delegate as such: $B_E = (C, D)$ and $B_F = (E, E)$ (as depicted in Figure 3.2). Agent E has the priority $j \succ_E k$ while F has the priority $k \succ_F j$. PDC first considers delegations of the highest priority issue. The top priority issue for E is j and their delegate has a vote for j in $\boldsymbol{X}^0$. As E currently has no votes in $\boldsymbol{X}^0$ we let $X_{Ej}^1 = 1$. Then, the delegation on the second priority issue k is not consistent. Thus, $X_{Ek}^1 = 0$. As in $\boldsymbol{X}^0$, E does not have any votes and F's votes can only be added in the second iteration. Thus, $X_F^2 = (1, 0)$ and $\texttt{PDC}(\boldsymbol{B}) = ((1, 0), (0, 1), (1, 0), (1, 0))$. PVC first computes the canonical profile of votes $\boldsymbol{X}^{\boldsymbol{B}} = ((1, 0), (0, 1), (1, 1), (1, 1))$, where $X_E^{\boldsymbol{B}}$ and $X_F^{\boldsymbol{B}}$ are inconsistent. For similar reasons as when using PDC, PVC then changes E's vote to $X_E^0 = (1, 0)$. However, for F, the algorithm first tries $X_{Fk}^0 = 1$ in accordance with F's priorities, which is consistent, entailing that $X_{Fj}^0 = 0$. Thus, $\texttt{PVC}(\boldsymbol{B}) = ((1, 0), (0, 1), (1, 0), (0, 1))$. △

3.4.1 Complexity of PDC and PVC

In the absence of constraints, it is easy to see that all four rules output the canonical profile of votes associated with the profile.

Remark 3.2. When $\Gamma = \top$, PDC, PVC, MDC and MVC give the same outcome, $\boldsymbol{X}^{\boldsymbol{B}}$.

We next show that PDC and PVC always terminate with consistent profiles of votes.

Proposition 3.3. *For any acyclic profile $\boldsymbol{B}$ and constraint Γ, PDC and PVC terminate, PDC($\boldsymbol{B}$) $\models \Gamma$ and PVC($\boldsymbol{B}$) $\models \Gamma$.*

Proof. We start from PDC. Due to the `while`-loop on line 4, if Algorithm 6 terminates, it does so with $\boldsymbol{X} \in \{0, 1\}^{n \times m}$. We assume for a contradiction that Algorithm 6 does not terminate, and therefore, $X_{ij}^t = \Delta$ for all $t > T$ after some iteration $T \in \mathbb{N}$ and for some $i \in \mathcal{N}$ and $j \in \mathcal{I}$. Note that $B_{ij} \notin \{0,1\}$ as the algorithm does not change any direct votes to Δ. Therefore, B_{ij} is a delegation, and since votes are propagated following delegations, we can infer that also $X^t_{B_{ij}j} = \Delta$. By the acyclicity of $\boldsymbol{B}$, the delegation chain on issue j starting at agent i must end at a direct voter for j, whose vote will then eventually reach voter i, against our assumption. Next, we show that every agent's votes are consistent with Γ. Recall that the initial direct votes are consistent, i.e., $\hat{B}_i \not\approx \Gamma$ for all $i \in \mathcal{N}$. Each time a vote is added, we check if its addition allows for consistent completion; if not, the opposite is added. As $\mathcal{X}_\Gamma \neq \emptyset$, the process is guaranteed to give a consistent completion of votes.

Next, we show the same for PVC. Algorithm 7 first finds $\boldsymbol{X}^{\boldsymbol{B}}$. It then inspects the cells of $\boldsymbol{X}^0$ with a for-loop on line 4, cycling through every agent and issue (in order of their priorities), after which it will terminate. In line 6, it inspects every entry of $\boldsymbol{X}^0$ without a vote $X^0_{ij} = \Delta$ and then updates it to a vote in $\{0,1\}$ from line 7 to line 8.

Thus, the algorithm always returns $\boldsymbol{X} \in \{0,1\}^{n\times m}$. We next show that the agents' votes are consistent. The algorithm starts with $\hat{\boldsymbol{B}}$, which by assumption is such that $\hat{B}_i \not\approx \Gamma$ for all $i \in \mathcal{N}$. It then iteratively adds votes from $\boldsymbol{X}^{\boldsymbol{B}}$ if they can be completed to a consistent set of votes, otherwise, the opposite vote is added. This will always lead to votes $X^0_i \models \Gamma$. Therefore, PVC always terminates with $\boldsymbol{X}^{n\times m}$ and $X_i \models \Gamma$ for all $i \in \mathcal{N}$. □

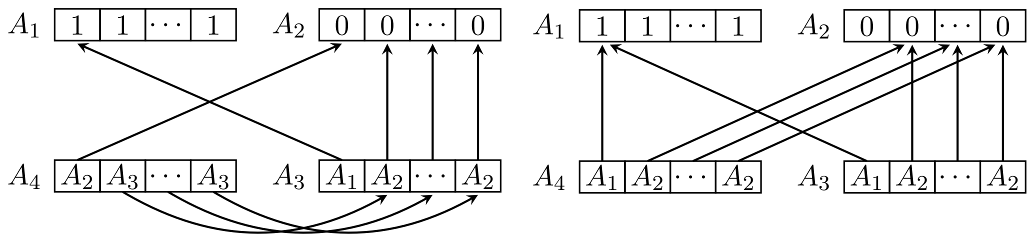
We show that both rules run in polynomial time. Recall that n is the number of voters, m the number of issues, and let ℓ be the time to check if a partial vote has a Γ-consistent completion (we assumed that ℓ is polynomial in n and m).

Proposition 3.4. *PDC terminates in $\mathcal{O}(nm(n+\ell))$ time.*

Proof. In each iteration of while-loop on line 4, Algorithm 6 first checks if $\boldsymbol{X}^t \in \{0,1\}^{n\times m}$ in $\mathcal{O}(nm)$ time. Since each issue has at least one direct voter, in the first iteration of the while-loop the for-loop on line 6 checks at most $m(n-1)$ delegations, to see if their delegate has a vote in $\boldsymbol{X}^0$, each taking constant time. As the profile is acyclic, at least one vote for each issue is added in each iteration. Thus, in iteration t, the for-loop checks at most $m(n-t)$ delegations. Therefore, Algorithm 6 requires at most $\sum_{t=1}^{n} nm + (n-t)m \in \mathcal{O}(mn^2)$ time to check the while-loop condition and if a voter's delegate has a direct vote. Furthermore, once for each delegation, we check if the addition of their delegate's vote has a Γ-consistent completion, in total taking at most $m(n-1)\ell$ steps. Hence, Algorithm 6 terminates in $\mathcal{O}(nm(n+\ell))$ time. □

Proposition 3.5. *PVC terminates in $\mathcal{O}(nm(n+\ell))$ time.*

Proof. Algorithm 7 first finds $\boldsymbol{X}^{\boldsymbol{B}}$ in $\mathcal{O}(n^2m)$ time, where for each issue and each agent it takes at most $\mathcal{O}(n)$ time to unravel the delegation chain until a direct voter is found. For each delegation in $\boldsymbol{B}$, the for-loop on line 4 will check, if the addition of the vote found in $\boldsymbol{X}^{\boldsymbol{B}}$ along with the votes already in $\boldsymbol{X}^0$ can be completed to



(a) Profile from Proposition 3.6 (b) Profile from Proposition 3.7

Figure 3.3: The profiles from (a) Proposition 3.6 and (b) Proposition 3.7 when $\mathcal{N} = \{A_1, A_2, A_3, A_4\}$ and $|\mathcal{I}| = m$.

comply with Γ (following the order given by the priorities). As there are at most $m(n-1)$ delegations and each completion check takes ℓ steps, this can be done in $\mathcal{O}(nm\ell)$ time. Summing the two figures, we obtain that Algorithm 7 gives an outcome in $\mathcal{O}(nm(n+\ell))$ time. □

3.4.2 Approximation Bounds

Next, we assess whether `PDC` and `PVC` are good polynomial approximations of `MDC` and `MVC`, giving a negative response. This is unsurprising, as `PDC` and `PVC` aim to respect priorities rather than minimise changes. By a slight abuse of notation, we define $\texttt{flip}(\mathcal{F}(\boldsymbol{B}))$ to count the number of changes $\mathcal{F}$ makes when finding a consistent outcome on profile $\boldsymbol{B}$. This corresponds to delegations in $\boldsymbol{B}$ being ignored by `PDC` and `MDC`, or direct votes being reverted in $\boldsymbol{X}^{\boldsymbol{B}}$ by `PVC` and `MVC`. An example of the profiles used in each of the following proofs can be seen in Figure 3.3.

Proposition 3.6. *For each $n \geq 3$ and $m \geq 2$, there is a profile $\boldsymbol{B}$ such that $\texttt{flip}(\texttt{PDC}(\boldsymbol{B})) = (n-2) \times (m-1) \times \texttt{flip}(\texttt{MDC}(\boldsymbol{B}))$.*

Proof. Consider $\mathcal{N} = \{A_1, \cdots, A_n\}$ and $\mathcal{I} = \{i_1, \cdots, i_m\}$, and $\mathcal{X}_\Gamma = \{(0, \cdots, 0), (1, \cdots, 1)\}$. Consider profile $\boldsymbol{B}$ where $B_{A_1} = (1, \cdots, 1)$, $B_{A_2} = (0, \cdots, 0)$, and for the remaining agents let $B_{A_t} = (A_{t-2}, A_{t-1}, \cdots, A_{t-1})$ for $t \in [3, n]$, each with priorities $i_x \succ_{A_t} i_y$, if $x < y$. Replacing the delegation of A_3 on issue i_1 to a direct vote of 0 gives a consistent profile, $\texttt{flip}(\texttt{MDC}(\boldsymbol{B})) = 1$. However, `PDC` returns the vote $X_{A_t} = (1, \cdots, 1)$ when t is odd and $X_{A_t} = (0, \cdots, 0)$ when t is even. Thus, for the $n-2$ delegating agents, A_t with $t \geq 2$, all of their delegations are flipped for the issues in $\mathcal{I} \backslash \{i_1\}$. Thus, $\texttt{flip}(\texttt{PDC}(\boldsymbol{B})) = (n-2)(m-1)$, and $\texttt{flip}(\texttt{PDC}(\boldsymbol{B})) = (n-2) \times (m-1) \times \texttt{flip}(\texttt{MDC}(\boldsymbol{B}))$. □

Proposition 3.7. *For each $n \geq 3$ and $m \geq 2$, there is a profile $\boldsymbol{B}$ such that $\texttt{flip}(\texttt{PVC}(\boldsymbol{B})) = (m-1) \times \texttt{flip}(\texttt{MVC}(\boldsymbol{B}))$.*

Proof. Consider $\mathcal{N} = \{A_1, \cdots, A_n\}$ and $\mathcal{I} = \{i_1, \cdots, i_m\}$, and $\mathcal{X}_\Gamma = \{(0, \cdots, 0), (1, \cdots, 1)\}$. The ballots of $\boldsymbol{B}$ are as follows: $B_{A_1} = (1, \cdots, 1)$, $B_{A_2} = (0, \cdots, 0)$, and the remaining agents have the ballot $B_{A_t} = (A_1, A_2, \cdots, A_2)$ for $t \in [3, n]$, with the priorities $i_x \succ_{A_t} i_y$ for every $x < y$.

The canonical profile $\boldsymbol{X}^{\boldsymbol{B}}$ is inconsistent since $X_{A_t} = (1, 0, \cdots, 0)$ for $t > 2$. `MVC` alters the delegated vote of such voters on issue i_1, hence $\texttt{flip}(\texttt{MVC}(\boldsymbol{B})) = n - 2$. `PVC` instead reverts all delegated votes except for the one on issue i_1, hence $\texttt{flip}(\texttt{PVC}(\boldsymbol{B})) = (n - 2) \times (m - 1)$. Therefore, $\texttt{flip}(\texttt{PVC}(\boldsymbol{B})) = (m - 1) \times \texttt{flip}(\texttt{MVC}(\boldsymbol{B}))$. □

Propositions 3.7 and 3.6 show us that the priority procedures can make many more changes to the canonical profile of votes than the minimisation procedures. Thus, we can see that the priority procedures are not approximating their minimisation counterparts.

3.4.3 Comparing Rules on Priorities

In this section, we compare the four procedures that find consistent votes with respect to the agents' priorities over the issues. This section aims to check if our priority procedures respect the priorities elicited from the voters. We use the $\texttt{top}^i$ function which identifies agent i's highest priority issue from a subset of issues according to $\prec_i$.

We first define the sets of issues on which a rule $\mathcal{F}$ does not respect agent i's delegations, either with respect to their delegate (*del*):

$$del^i_{\mathcal{F}}(\boldsymbol{B}) = \{j \mid j \in \mathcal{I},\ B_{ij} \notin \{0, 1\} \text{ and } \mathcal{F}(\boldsymbol{B})_{ij} \neq \mathcal{F}(\boldsymbol{B})_{B_{ij}j}\},$$

or with respect to the canonical profile of vote (*dir*):

$$dir^i_{\mathcal{F}}(\boldsymbol{B}) = \{j \mid j \in \mathcal{I} \text{ and } X^{\boldsymbol{B}}_{ij} \neq \mathcal{F}(\boldsymbol{B})_{ij}\}.$$

The set $del^i_{\mathcal{F}}(\boldsymbol{B})$ identifies the issues on which the votes of agent i in $\mathcal{F}(\boldsymbol{B})$ differ from their delegate's vote, whereas $dir^i_{\mathcal{F}}(\boldsymbol{B})$ identifies the issues on which the votes of agent i in the outcome $\mathcal{F}(\boldsymbol{B})$ differ to their votes in $\boldsymbol{X}^{\boldsymbol{B}}$.

Note that there is no logical connection between these sets. Intuitively, agents want lower priority issues in these sets, as they reflect the delegations that have been ignored in finding consistent votes.

For a profile $\boldsymbol{B}$, rules $\mathcal{F}$ and $\mathcal{F}'$, and measure $c \in \{del, dir\}$, we say that an agent i *top-prefers* $c^i_{\mathcal{F}}(\boldsymbol{B})$ *to* $c^i_{\mathcal{F}'}(\boldsymbol{B})$ (denoted by $c^i_{\mathcal{F}}(\boldsymbol{B}) \succeq^{\texttt{top}}_i c^i_{\mathcal{F}'}(\boldsymbol{B})$) if $\texttt{top}^i(c^i_{\mathcal{F}}(\boldsymbol{B})) \preceq_i \texttt{top}^i(c^i_{\mathcal{F}'}(\boldsymbol{B}))$, where $\texttt{top}^i(c)$ gives the highest element of c with respect to $\prec_i$. Thus, top-preferring $c^i_{\mathcal{F}}(\boldsymbol{B})$ to $c^i_{\mathcal{F}'}(\boldsymbol{B})$ entails that the top issue whose delegation is ignored using $\mathcal{F}$ has a lower priority than the top issue using $\mathcal{F}'$. Let $\boldsymbol{c}^{\boldsymbol{B}}_{\mathcal{F}} = (c^1_{\mathcal{F}}(\boldsymbol{B}), \cdots, c^n_{\mathcal{F}}(\boldsymbol{B}))$, and $\boldsymbol{c}^{\boldsymbol{B}}_{\mathcal{F}'} \preceq^{\texttt{top}}_{\mathcal{N}} \boldsymbol{c'}^{\boldsymbol{B}}_{\mathcal{F}}$ if and only if $c^i_{\mathcal{F}'}(\boldsymbol{B}) \preceq^{\texttt{top}}_i c^i_{\mathcal{F}}(\boldsymbol{B})$ for every $i \in \mathcal{N}$.

The following proposition shows that `PVC` respects issues with higher priorities than its minimisation counterpart `MVC`.

Proposition 3.8. *$\boldsymbol{dir}^{\boldsymbol{B}}_{\texttt{MVC}} \preceq^{top}_{\mathcal{N}} \boldsymbol{dir}^{\boldsymbol{B}}_{\texttt{PVC}}$ for any profile $\boldsymbol{B}$.*

Proof. We assume for a contradiction that there exists a profile $\boldsymbol{B}$ such that $\boldsymbol{dir}^{\boldsymbol{B}}_{\texttt{MVC}} \not\preceq^{\texttt{top}}_{\mathcal{N}} \boldsymbol{dir}^{\boldsymbol{B}}_{\texttt{PVC}}$. Therefore, there is an agent $i \in \mathcal{N}$ such that $\texttt{top}^i(dir^i_{\texttt{MVC}}(\boldsymbol{B})) \prec_i$

$\texttt{top}^i(dir^i_{\texttt{PVC}}(\boldsymbol{B}))$. Suppose that $\texttt{top}^i(dir^i_{\texttt{MVC}}(\boldsymbol{B}))$ is i's k^th priority and that $\texttt{top}^i(dir^i_{\texttt{PVC}}(\boldsymbol{B}))$ is i's m^{th} priority (thus, $m < k$). MVC outputs a consistent vote including i's first $(k-1)$ top-priority issues. Thus, the partial truth assignment $\overline{X}_i$ including i's direct votes and accepting the delegations on the top $(k-1)$ priority issues is such that $\overline{X}_i \not\approx \Gamma$. PVC on the other hand only accepts the delegations on i's top $(m-1)$ priorities and rejects the delegation on the m^{th} issue. Yet if $\overline{X}_i \not\approx \Gamma$, then any $\overline{X'} \subseteq \overline{X}_i$ is such that $\overline{X'} \not\approx \Gamma$, including the partial assignment accepting i's direct votes and their delegations on the first m priority issues. We have reached a contradiction as PVC rejected a partial assignment that could be completed. □

We now show that the analogous result is not true for PDC.

Proposition 3.9. *There exists profiles $\boldsymbol{B}$ and $\boldsymbol{B'}$ such that $\boldsymbol{del}^{\boldsymbol{B}}_{MDC} \preceq^{top}_{\mathcal{N}} \boldsymbol{del}^{\boldsymbol{B}}_{PDC}$ and $\boldsymbol{del}^{\boldsymbol{B'}}_{PDC} \preceq^{top}_{\mathcal{N}} \boldsymbol{del}^{\boldsymbol{B'}}_{MDC}$.*

Proof. First, we give $\boldsymbol{B}$ such that $\boldsymbol{del}^{\boldsymbol{B}}_{\texttt{MDC}} \preceq^{\texttt{top}}_{\mathcal{N}} \boldsymbol{del}^{\boldsymbol{B}}_{\texttt{PDC}}$. We have $\mathcal{N}=\{C, D, E, F\}$ and issues $\mathcal{I}=\{i, j, k\}$ with the set of consistent votes being $\mathcal{X}_\Gamma = \{(1, 0, 0), (0, 1, 1)\}$. We let $B_C = (1, 0, 0)$, $B_D = (0, 1, 1)$, $B_E = (0, 1, 1)$ and agent F delegates as such: $B_F = (C, D, E)$ with the following priorities over the issues $i \succ_F j \succ_F k$. PDC first adds F's delegation on i, giving $X_F = (1, \Delta, \Delta)$. It then attempts to add the delegations on j and k but rejects both, resulting in $X_F = (1, 0, 0)$. Thus, $del^F_{\texttt{PDC}}(\boldsymbol{B}) = \{j, k\}$. MDC returns $X'_F = (0, 1, 1)$, where only the delegation on issue i is changed, thus $del^F_{\texttt{MDC}}(\boldsymbol{B}) = \{i\}$. All agents $T \in \mathcal{N}\backslash\{F\}$ are direct voters, hence $del^T_{\texttt{MDC}}(\boldsymbol{B})=del^T_{\texttt{PDC}}(\boldsymbol{B})=\emptyset$. As $\texttt{top}^F(del^F_{\texttt{MDC}}) \prec_F \texttt{top}^F(del^F_{\texttt{PDC}})$, we can conclude that $\boldsymbol{del}^{\boldsymbol{B}}_{\texttt{MDC}} \preceq^{\texttt{top}}_{\mathcal{N}} \boldsymbol{del}^{\boldsymbol{B}}_{\texttt{PDC}}$.

Now let $\mathcal{N}$, $\mathcal{I}$, and $\mathcal{X}_\Gamma$ be as in the first part of the proof, yet consider profile $\boldsymbol{B'}$ with the following ballots: B_C=$(0, 1, 1)$, B_D=$(1, 0, 0)$, B_E=$(C, C, 1)$, and B_F=(E, E, D), with E and F having the priority $i \succ j \succ k$. The first iteration of Algorithm 6 gives $X_E = (0, 1, 1)$ and $X_F = (\Delta, \Delta, 0)$. In the second iteration, we have $X_F = (1, 0, 0)$, as the delegations to E are inconsistent. Therefore, $del^C_{\texttt{PDC}}(\boldsymbol{B'}) = del^D_{\texttt{PDC}}(\boldsymbol{B'}) = del^E_{\texttt{PDC}}(\boldsymbol{B'}) = \emptyset$, whereas, $del^F_{\texttt{PDC}}(\boldsymbol{B'})$=$\{i, j\}$. MDC however lets X_F=$(0, 1, 1)$ where $del^F_{\texttt{MDC}}(\boldsymbol{B'})$=$\{k\}$. As $\texttt{top}^F(del^F_{\texttt{MDC}}(\boldsymbol{B'})) \prec_F \texttt{top}^F(del^F_{\texttt{PDC}}(\boldsymbol{B'}))$, we have that $\boldsymbol{del}^{\boldsymbol{B'}}_{\texttt{PDC}} \preceq^{\texttt{top}}_{\mathcal{N}} \boldsymbol{del}^{\boldsymbol{B'}}_{\texttt{MDC}}$. □

3.5 Knapsack Constraints

This section focuses on knapsack or budget constraints, i.e., sets of feasible votes respecting a budget limit $L \in \mathbb{N}$ of the form Γ_L=$\{X \mid \sum_{j\in\mathcal{I}} x_j \leq L\}$. They have the property that turning a vote from acceptance to rejection in a consistent ballot preserves consistency. In the algorithm proposed by Jain et al. [2021], this property is exploited to remove cycles and solve inconsistent delegations. However, this can lead to rejecting many issues and potentially leaving funds unassigned.

This section evaluates this factor for the four rules we proposed. Let $\texttt{count}(\boldsymbol{X}) = \sum_{i\in\mathcal{N}, j\in\mathcal{I}} x_{ij}$ be the number of acceptances in the profile of votes $\boldsymbol{X}$.

Remark 3.3. Given a profile $\boldsymbol{B}$ and budget constraint Γ_L, we have that $\texttt{count}(\boldsymbol{X}^{\boldsymbol{B}}) \geq \texttt{count}(\mathcal{F}(\boldsymbol{B}))$ for $\mathcal{F} \in \{\texttt{MDC}, \texttt{MVC}, \texttt{PDC}, \texttt{PVC}\}$.

Proposition 3.10. *For any profile $\boldsymbol{B}$, $\texttt{count}(\texttt{MVC}(\boldsymbol{B})) \geq \texttt{count}(\mathcal{F}(\boldsymbol{B}))$ for $\mathcal{F} \in \{\texttt{MDC}, \texttt{PVC}, \texttt{PDC}\}$.*

Proof. Let $\boldsymbol{B}$ be an arbitrary profile and Γ_L be a budget constraint. $\texttt{MVC}(\boldsymbol{B})$ outputs all consistent profiles of votes obtained from $\boldsymbol{X}^{\boldsymbol{B}}$ with a minimal number of 1s changed to 0s. Our rules do not change 0s to 1s, as rejections do not use any of the budget. Note that if $\boldsymbol{X}, \boldsymbol{X}' \in \texttt{MVC}(\boldsymbol{B})$ then $\texttt{count}(\boldsymbol{X}) = \texttt{count}(\boldsymbol{X}')$. Thus, any other consistent delegation rule must approve at most the same number of issues as $\texttt{MVC}(\boldsymbol{B})$. □

An analogous result showing that PVC accepts more issues than MDC or PDC does not hold, as can be shown by counterexample. To conclude, the approach of minimal delegation changes proposed in the literature might not be appropriate for budgeting applications, where it is outperformed by minimal vote changes in terms of the number of projects accepted.

3.6 Removing Assumptions on the Model

So far in this chapter, we have assumed that there are no issue-wise delegation cycles. As such, an ultimate delegate can be found for every agent on every issue. In this section, we remove this assumption as well as extend the model to allow each agent to have their own rationality constraint. We show that the results in the slightly extended models do not change from the initial results.[5]

In Section 3.6.1, we will allow cyclic delegations on single issues. Thus, we are no longer guaranteed that all agents will have a vote in $\{0, 1\}$ in the canonical profile $\boldsymbol{X}^{\boldsymbol{B}}$. Then, in Section 3.6.2, we allow the notion of rationality to vary between agents. Thus, each agent $i \in \mathcal{N}$ can give their own notion of rationality. Hence, each $i \in \mathcal{N}$ gives their own set of constraints Γ_i.

3.6.1 Allowing Cycles Among Issues

We first inspect how our model changes when allowing delegation cycles on specific issues. More formally, we have different conditions on the delegation graph $G_{\boldsymbol{B}} = (V, E)$. Recall that $V = \mathcal{N}$ and for each $B_{ij} = k$, we create an edge $(i, k, j) \in E$ from agent i to agent k on issues j, where j is a label on the edge. Allowing issue-wise delegation cycles means that for an issue j, there can be a j-path starting and returning to a single agent. For example, we will allow for $B_{ij} = k$ and $B_{kj} = i$, giving a cycle between i and k on issue j.

[5]Although we will show that many of the equivalent results do not change, we leave the presentation of this chapter as focusing on the more basic model. This is due to the slight extensions making the presentation of the procedures and the proofs more complicated.

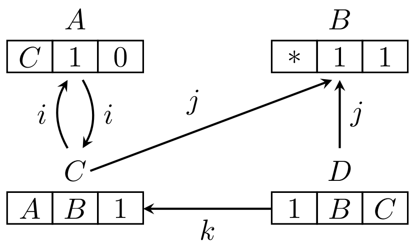


Figure 3.4: The profile $\boldsymbol{B}$ from Example 3.6 for the agents $\mathcal{N} = \{A, B, C, D\}$ voting on the issues $\mathcal{I} = \{i, j, k\}$. The box below their names displays their ballot showing their ballots on issues i, j, then k (reading from left to right). The edges show delegations in the figure and the edges labels represent the delegation's issue.

We now update the model described in Section 3.2 to account for cycles. We do not resolve cycles by assigning a direct vote for or against the cyclic issue. Instead, we extend the domain of alternatives for the issues to allow for abstentions. Hence, we now consider that every $j \in \mathcal{I}$ has a domain $\mathcal{D}(j) = \{0, 1, *\}$. Although the use of abstentions could be exclusively for cycles, we will consider that those voting directly on an issue may also choose to abstain.[6] Hence, we now allow ballots to be $B_i \in \Pi_{j\in\mathcal{I}}(\mathcal{D}(j) \cup \mathcal{N}\backslash\{i\})$ for each $i \in \mathcal{N}$ on our new domain of alternatives. With this addition to the domain of the issues in our model, we update the definition of $\hat{\boldsymbol{B}}$ as such:

$$\hat{\boldsymbol{B}}^* = \begin{cases} B_{ij}, & \text{if } B_{ij} \in \{0, 1, *\} \\ *, & \text{if } i \text{ has no ultimate delegate on issue } j \\ \Delta, & \text{otherwise.} \end{cases}$$

Hence, now $\hat{\boldsymbol{B}}^*$ gives the votes that we will not consider changing to preserve consistency among the votes on the issues. This is analogous to the previous model, where we did not change the direct votes of the agents to gain consistency.

In this new model, we may now have complete vectors of votes $X \in \{0, 1, *\}$ and partial vectors $\overline{X} \in \{0, 1, *, \Delta\}$, and with this, we need to update the notion of consistency. Note that we keep the restrictions imposed on rationality constraints Γ, that a partial vote can be completed consistently in polynomial time. We say that a vector X is consistent with Γ (with $X \in \{0, 1, *\}$ or $X \in \{0, 1, *, \Delta\}$) if and only if there is some $X' \supseteq X_{\restriction\{0,1\}}$ such that $X' \in \{0, 1\}$ and $X' \models \Gamma$ where $X_{\restriction\{0,1\}}$ changes all $*$s in X to Δs. Observe that this can also be checked in polynomial time. The canonical profile of votes $\boldsymbol{X}^{\boldsymbol{B}}$ is found in the same way as described in Section 3.2, however due to our new definition on $\hat{\boldsymbol{B}}^*$ we have that $\boldsymbol{X}^{\boldsymbol{B}} \in \{0, 1, *\}^{n\times m}$.

Example 3.6. Consider the agents $\mathcal{N} = \{A, B, C, D\}$ who are voting on the issues $\mathcal{I} = \{i, j, k\}$. The rationality constraint Γ reflects that at most two of the three issues may be accepted. Their ballots can be seen in bee seen in Figure 3.4. Observe

[6]The model where abstentions are reserved for only the final vote on cyclic issues is a restricted version of this new model.

that agent B is directly abstaining on issue i and that there is a delegation cycle on issue i between A and C. The latter entails that A and C have $*$ recorded for their vote on issue i in $\hat{\boldsymbol{B}}^*$ (as seen in Table 3.1).

	i	j	k
$\hat{\boldsymbol{B}}^*{}_A$	$*$	1	0
$\hat{\boldsymbol{B}}^*{}_B$	$*$	1	1
$\hat{\boldsymbol{B}}^*{}_C$	$*$	Δ	0
$\hat{\boldsymbol{B}}^*{}_D$	1	Δ	Δ

Table 3.1: The direct votes given by $\hat{\boldsymbol{B}}^*$ from the profile $\boldsymbol{B}^*$ given in Example 3.6.

Note that every $*$ in $\hat{\boldsymbol{B}}^*$ will never be changed to a vote of 0 or 1.

To check consistency, consider the complete ballot of B. $X_B = (*, 1, 1)$ is consistent with the constraint Γ as there exists $X' = (0, 1, 1)$ such that $X_B \subseteq X'$ and $X' \models \Gamma$. Although X' is the only consistent, complete extension of X_B, we do not alter the B's abstention $*$ in X_B.

The canonical profile of votes $\boldsymbol{X}^{\boldsymbol{B}}$ would differ from $\hat{\boldsymbol{B}}^*$ on the vectors of votes for agents C and D. We find that $\boldsymbol{X}^{\boldsymbol{B}}_C = (*, 1, 1)$ which is consistent with Γ. However, $\boldsymbol{X}^{\boldsymbol{B}}_D = (1, 1, 1)$ which is not consistent with Γ. △

From seeing the extension of our model to account for cycles, the next step we address is how these additions affect the results from the previous sections of this chapter. We start by assessing the extension of Proposition 3.1 to the problem MinimalDelChange*. The only difference between MinimalDelChange and MinimalDelChange* is that we allow for the profile in the input to have the domain $\mathcal{D}(j) = \{0, 1, *\}$ for all $j \in \mathcal{I}$ and we allow profiles with issue-wise delegation cycles.

Proposition 3.11. *MinimalDelChange* is NP-complete.*

Proof. First, notice that every instance of MinimalDelChange is also an instance of MinimalDelChange*. Therefore, as MinimalDelChange is NP-complete, we have that MinimalDelChange* is NP-hard. For membership of MinimalDelChange* in NP, we take the same certificate as in Proposition 3.1. Consider a certificate that lists agents and issues, indicating delegations to be changed to direct votes $\{0, 1, *\}$ to obtain $\boldsymbol{B}'$ from $\boldsymbol{B}$. First, we check that the list has at most k entries. Then, we check whether $\boldsymbol{B}'$ is consistent, which can be done in polynomial time by Remark 3.1. □

By similar reasoning as to why Proposition 3.11 holds given Proposition 3.1, analogous results can be shown for Propositions 3.2, 3.4, and 3.5.

Observe that as in their new model, as the delegations on cycles are never changed to votes that could affect the rationality constraint, we can see that the bound on the approximations given in Propositions 3.6 and 3.7 cannot be worse than the bounds given. Similarly, for Propositions 3.8 and 3.9, as the delegations on cycles are never being changed, these results also remain in our new setting.

3.6.2 Personal Rationality Constraints

We now inspect what happens to our model when we allow each agent to set their own rationality constraint. Each agent may submit their own constraints to represent how a rational collection of votes looks for them after their delegations have been resolved, i.e., for all $i \in \mathcal{N}$, we have Γ_i such that $X_i \models \Gamma_i$. We impose the same restrictions on the personal constraints as in the previous model, i.e., that a partial vote can be completed to a feasible one in polynomial time. We will denote the collection of the constraints as such $\mathbf{\Gamma} = (\Gamma_1, \cdots, \Gamma_n)$.

By allowing agents to submit their own rationality constraints, we let them determine how their delegations should be resolved. For example, when the issues represent the projects to be funded, one agent may want their constraint to be a knapsack constraint (that their final votes reflect the budget constraint). Another agent may want their final votes to accept at most four projects to ensure their ballot does not contain too many acceptances.

We first comment the effect on the complexity of MinimalVoteChange and MinimalDelChange (Propositions 3.2 and 3.1). We first note that, as in Section 3.6.1, we see that the original model is a sub-case of our model where for all $i, j \in \mathcal{N}$ we have that $\Gamma_i = \Gamma_j$. Thus, both problems remain NP-hard. Membership for both problems remains close to the original proofs. The only difference is that consistency is checked with respect to the agent's constraint rather than the general constraint. Thus, the results of Propositions 3.2 and 3.1 carry over to this new domain.

Furthermore, the procedures `PDC` and `PVC` remain tractable (Propositions 3.4 and 3.5). The only difference between the propositions would be that ℓ now represents the longest time needed to check if a partial vote has a Γ_i-consistent completion for any $i \in \mathcal{N}$.

Finally, we then consider the remaining results from Section 3.4.2 and 3.4.3 are not affected by the addition of personal rationality constraints. We do not consider the results from Section 3.5 as this setting insists on voters having the same constraint.

3.7 Conclusion and Future Work

This chapter's starting point was the two existing approaches in the literature of multi-issue liquid democracy with constraints connecting the issues. However, the existing research has focused on specific settings such as knapsack voting and preference aggregation. Both studied the changes to the ballots to maintain consistency in the final resolved votes. We generalise this by studying liquid democracy on multiple interconnected issues, putting forward two novel ideas: first, we propose two rules that start by resolving delegations and then make changes to the final votes; second, we design polynomial algorithms to maintain vote consistency by eliciting agents' priorities over the issues.

Models of multi-issue liquid democracy have clear applications for digital democracy platforms. Given that platforms such as LiquidFeedback are already running liquid democracy elections on multiple issues, I believe that the extension studied in Section 3.6.2, where every agent can set their own rationality constraint, can make an impact on the running of the platform. It will allow voters who cannot keep track of many elections on the different issues some constraints over their overall final votes. Thus, they can impose their own global rationality over the issues, and it is the mechanism's job to ensure the voter's rationality is upheld.

The main open problem is whether `PDC` can be improved since Proposition 3.9 shows that it does not use priorities in the best way. We conjecture that this cannot be done in polynomial time, but a result showing this impossibility is yet to be shown. Our result in Section 3.5 shows that specific constraints require specific treatment, and thus, the choice of a consistent delegation rule is not trivial. Hence, a prominent direction for future research is to specify our general setting on specific classes of constraints.

Chapter 4

Boolean Opinion Diffusion

4.1 Introduction

This chapter considers a model of opinion diffusion, which is structurally related to delegative democracy. Instead of modelling delegations in the mechanism, this chapter models influence. Opinion diffusion models opinions spreading throughout a social network. In the network, every node represents an agent with an opinion on a given issue, which we will assume is binary. The network's edges determine each agent's influencers. Many digital platforms, such as social media platforms, have this same structure; the platform users are the network's nodes, and each edge between two nodes represents a connection in their (para)social sphere.

A classical modelling assumption in opinion diffusion is that an agent's opinion changes with respect to a threshold function. Thus, an agent's opinion changes when a given proportion of their influencers have a different opinion (see, e.g., the seminal work of Granovetter [1978]). Some typical problems studied in the opinion diffusion literature are stable diffusions and opinion control. In the former, we recognise whether the diffusion process will stabilise so no agent wants to update their opinion. In the latter, we study how to control certain opinion characteristics, such as gaining a consensus on the issue, for instance, by changing some initial opinions or the network structure. In this chapter, we show that Boolean networks, a well-studied mathematical model from biology, generalise models of binary opinion diffusion (with opinions being either 0 or 1) to define influence updates among the agents in a more fine-grained way. Thus, it allows for a more realistic framework to understand the spread of opinions.

Boolean networks are graphs where each node has a state, typically on or off, 1 or 0, yes or no. A discrete-time dynamical process starts from an initial state. A set of Boolean update functions determines the state in the following iterations of the network. Each node has a function that takes as input the binary states of their influencers. Akutsu et al. [2008], Cheng et al. [2010a] and Kauffman [1969] all provide good introductions to this model. There have been many mathematical advancements in the study of Boolean networks due to their ability to model gene regulatory networks in biology (see, e.g., Kauffman [1993] and Shmulevich and Zhang [2002]).

Boolean networks can model opinion diffusion on binary issues where update functions are arbitrary Boolean functions. We assume that such functions are represented as logical formulas built from the standard connectives $(\wedge, \vee, \neg, \cdots)$, and the atomic propositions are the influencers of the agent. These functions allow us to

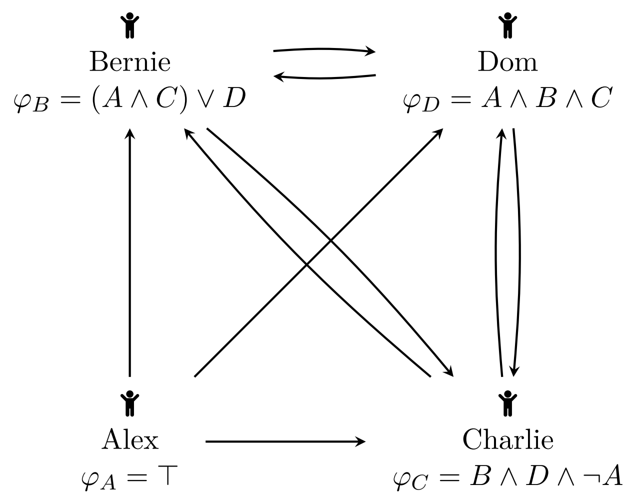

Bernie
$\varphi_B = (A \wedge C) \vee D$
Dom
$\varphi_D = A \wedge B \wedge C$
Alex
$\varphi_A = \top$
Charlie
$\varphi_C = B \wedge D \wedge \neg A$

Figure 4.1: The Boolean network associated with agents described in Example 4.1. Under each agent, we display their update function as a Boolean function. The arrows in the network describe the influence within the social network. For example, Alex only has outgoing edges as they do not have any influencers.

study fine-grained relationships between the agents, as showcased in the following example.

Example 4.1. A group is deciding whether they should go on holiday together. Alex has organised a holiday for themself and their friends Bernie, Charlie and Dom. Alex believes it is the perfect holiday for the group and wants to go. However, the opinions of the rest of the group correspond to the following formulas (their update functions are also depicted in Figure 4.1):

- Bernie's closest friend is Dom, and Bernie will go if Dom decides to go. However, Bernie would also go on holiday if both Alex and Charlie decided to go as well. Therefore, Bernie's opinion would be updated with respect to the following propositional formula: $(\textit{Alex} \wedge \textit{Charlie}) \vee \textit{Dom}$.

- Charlie has currently fallen out with Alex. Therefore, Charlie will only go if Bernie and Dom are both going and Alex is not. This could be expressed as $\textit{Bernie} \wedge \textit{Dom} \wedge \neg \textit{Alex}$.

- Dom is reluctant to go on the holiday, and thus, will only go on the holiday if all of their friends go as well: $\textit{Alex} \wedge \textit{Bernie} \wedge \textit{Charlie}$.

Consequently, as Alex will attend, Charlie will not, given they will only go if Alex does not. As Alex will attend and Charlie will not, Bernie will only go if Dom does. However, as Charlie will not go, neither will Dom. In turn, neither will Bernie. △

4.1.1 Contribution

This chapter introduces Boolean networks as opinion diffusion models and is based on the work of Colley and Grandi [2022b]. As such, we focus on some well-studied topics in binary opinion diffusion and see if they extend to our setting with Boolean update functions. We show that:

- The computational complexity of determining if a given Boolean network and set of initial opinions will lead to a stable state under synchronous updates is PSPACE-complete. We build upon a similar result from Chistikov et al. [2020] where opinion updates are majority functions by showing a non-trivial lemma (Section 4.3).
- There does not necessarily exist a sequence of asynchronous updates that leads to a stable profile and maximises agreement. Thus, the result from Bredereck and Elkind [2017] for majority update functions does not hold in our Boolean network setting. However, when update functions are restricted to contain only positive (or only negated) literals, we give a procedure that finds a stable state of opinions with a maximal number of agreements (Section 4.4).
- Synchronous opinion diffusion always terminates if the Boolean network imitates a multi-agent delegative voting problem, such as liquid democracy. Moreover, it does so in polynomial time and gives the same outcomes as the polynomial unravelling procedures defined in Chapter 2. We also show that manipulating collective opinions in this setting is a NP-complete problem (Section 4.5).
- Known results from the vast literature on Boolean networks can be applied to classical problems of opinion diffusion (Section 4.6).

4.1.2 Related Work

This section overviews some existing literature on Boolean networks and opinion diffusion, focusing on the most pertinent areas: threshold models, multi-issue opinion diffusion, and delegative voting.

Algorithmic Approaches to Boolean Networks. A recent stream of papers provided algorithmic results closer to our purposes, each having a biological application. Most notably, Akutsu et al. [2006] studied the complexity of choosing which nodes to control to gain a specific outcome and showed that this problem is NP-hard. Inoue [2011] gives a logical language by which Boolean networks can be entirely expressed. Kosub [2008] studies fixed points in social networks relating to our notion of stability. Section 4.6 expands on this related work by importing and rephrasing some known results in the opinion diffusion terminology.

Threshold Models. Much opinion diffusion research has focused on threshold models where agents update their opinions when a certain proportion of their influencers have differing opinions. The seminal work on binary opinion diffusion was explored by Granovetter [1978]. The problem of convergence for binary opinions has been widely studied. For example, Goles and Olivos [1980] showed that threshold models either terminate or cycle between two different collections of opinions. Christoff and Grossi [2017b] gave the conditions by which a social network stabilises on majority updates. Another popular problem is trying to maximise the influence in the network (see, e.g., Domingos and Richardson [2001], Richardson and Domingos [2002]). In particular, Zhuang et al. [2020] define graph-theoretic notions when using a threshold model which determines how to reach a consensus. Auletta et al. [2018] identifies conditions on the graph structure that allow a minority to influence the majority's opinions. Instead, Ferraioli and Ventre [2017] study a version of opinion games where pressure is put on agents to reach a consensus. Furthermore, they show that deciding whether the two opinions can coexist in some stable configuration is an NP-hard problem. Bredereck and Elkind [2017] study an asynchronous model with majority updates to maximise the number of agreements in the model with polynomial algorithms. One application of opinion diffusion with threshold updates is to model product adoption [Auletta et al., 2020b, Shen et al., 2019]. A large part of the study of product adoption is the introduction of seed nodes to influence the group of agents to adopt a new product, Auletta et al. [2020a] gives an algorithm to identify the agents as good seeds, yet this is not tractable. Moreover, Auletta et al. [2021] looks at an extended version of this model that moves away from binary opinions.

Multi-Issue Opinion Diffusion. General models of opinion diffusion have been proposed, assuming that agents have opinions on multiple interconnected issues. The problem studied here is how to ensure that influence on the different issues results in consistent opinions for each agent: Brill et al. [2016] need to ensure that individual preference orders remain transitive and acyclic; Botan et al. [2019] use arbitrary integrity constraints. Both focus on majoritarian update functions. In this context, the closest work to ours is that of Grandi et al. [2015], which considers opinion diffusion with arbitrary update functions on multiple interconnected binary issues.

Opinion Diffusion with Complex Opinion Updates. Rosenkrantz et al. [2020] focuses on opinion diffusion where update functions are given as Boolean functions. However, they only consider social networks that are directed and acyclic graphs. A different direction taken by Morrison and Naumov [2020] extends the update functions to have multiple thresholds and labels for their influencers. One example could allow different thresholds for different groups. Thus, an agent may update their opinion if 80% of their work colleagues have a different opinion or if 30% of their close friends do. Salehi-Abari et al. [2019] study a model of collective decision-making where a voter's utility is reflected not only in their own opinion being represented by the outcome but also if their neighbour's opinions are also reflected, modelling a form of empathy in a social network.

Multi-Agent Delegations. Christoff and Grossi [2017a] first observed the connection between models of delegative democracy and opinion diffusion. The authors express liquid democracy as a model of opinion diffusion where every agent is influenced by at most one agent. Models of delegative democracy have recently been extended to account for multi-agent delegations: Degrave [2014] allowed delegations to be fractionally spread among their delegates, and in Chapter 2, we let voters express a ranking of Boolean functions to model delegations. Section 4.5 expands upon the work of Christoff and Grossi [2017a] to consider the connection between Boolean opinion diffusion and voting models that allow for multi-agent delegations.

4.2 The Model

A set of agents $\mathcal{N} = \{1, \cdots, n\}$ can influence each other's opinion via a social network $G = (V, E)$. The agents are the nodes $V = \mathcal{N}$. The directed edges represent influence; thus $(i, j) \in E$ if agent i can influence j's opinion. Furthermore, we let the *influence neighbourhood* of agent $i \in \mathcal{N}$ be $N_i = \{j \mid (j, i)\}$. Therefore, $j \in N_i$ means that j can influence i's opinion on the issue (sometimes this set is referred to as i's in-neighbours). We allow the edge $(i, i) \in E$, as an agent i's current opinion may influence their future opinion. We study a setting with a single binary issue, denoting agent i's opinion as $o_i \in \{0, 1\}$. The agents' opinions are not static, so we let o_i^t be i's opinion at time $t \in \mathbb{N}$. Thus, o_i^0 is agent i's *initial opinion.* We will refer to the collection of the agents' opinions as a *profile of opinions.* At time t, we denote the profile of opinions by $O^t = (o_1^t, \cdots, o_n^t)$.

In this model, each agent has a Boolean function γ_i that represents when agent i's opinion changes, known as their *update function.* The update function γ_i for agent i is represented as a compact propositional formula in DNF built from the connectives $\neg, \wedge, \vee$ where the atomic variables are given by N_i (i.e., assuming that $Var(\gamma_i) = N_i$). If for some $i \in \mathcal{N}$, $N_i = \emptyset$, then their update function is constant. Hence, either $\gamma_i = \top$ or $\bot$. The collection of the agents' update functions is denoted by $\overline{\gamma} = (\gamma_1, \cdots, \gamma_n)$.

Most of this chapter concerns synchronous updates $\circ$ on the agents' opinions. For all agents, $\circ$ at time $t + 1$ checks if each agent's neighbours' opinions at t induce that agent's opinion to change. Thus, $\circ$ iteratively lets $o_i^{t+1} = 1$ if and only if $\bigwedge_{j \in N_i} o_j^t \vDash \gamma_i$ and lets $o_i^{t+1} = 0$, otherwise.[1] We denote an instance of opinion diffusion with $\langle G, \overline{\gamma}, O^0 \rangle$. We denote t iterations of the diffusion process as $\circ(G, \overline{\gamma}, O^0, t) = O^t$. We say that the synchronous update is *stable* if there is some

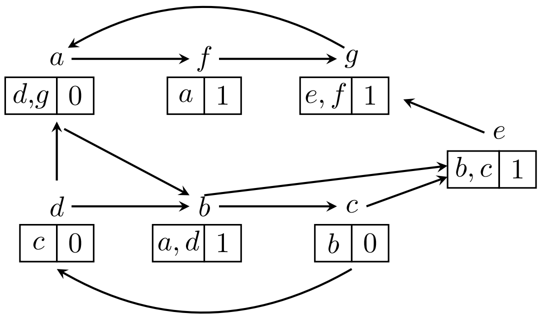
[1]Note that $\vDash$ is the logical symbol for logical entailment in propositional logic. For two propositional formulas φ and ψ, if $\varphi \vDash \psi$ holds, every truth assignment making φ true will also make ψ true. Another way of phrasing this is that $\varphi \to \psi$ is a tautology.

$t \in \mathbb{N}$ such that for all $t' > t$, we have that $\circ(G, \overline{\gamma}, O^0, t) = \circ(G, \overline{\gamma}, O^0, t')$, i.e., no more changes to the opinion profile can happen. At times, we refer to the stable profile of opinions as O^T. In the literature, the diffusion process stabilising is also referred to as converging [Chistikov et al., 2020]. As there are a finite number of opinion profiles of the agents, the process is cyclic if the synchronous update does not lead to a stable opinion profile.

Example 4.2. Let the set of agents be $\mathcal{N} = \{a, b, c, d, e, f, g\}$ have the following initial opinions $O^0 = (0, 1, 0, 0, 1, 1, 1)$. On the right-hand side of Figure 4.2, we see the social network G where $V = \mathcal{N}$ are the nodes and E represents the directed edges. The neighbourhoods of influence for each agent can be seen from G. For example, agent a has two incoming edges from d and g, thus $N_a = \{d, g\}$. On the left-hand side of Figure 4.2, Table (a) lists each agent's neighbourhood of influence, update function and initial opinion. Consider the update function $\gamma_b = a \wedge \neg d$. The intuition behind γ_b is that b will only update their opinion at time $t + 1$ to be $o_b^{t+1} = 1$ if and only if at time t, a is for the issue $o_a^t = 1$ and d is against the issue $o_d^t = 0$. Note that in all other scenarios for b (dictated by the different combinations of opinions of a and d, excluding b's initial opinion), that b's opinion is against the issue.

$i \in \mathcal{N}$	N_i	γ_i	o_i^0
a	d, g	$d \vee g$	0
b	a, d	$a \wedge \neg d$	1
c	b	b	0
d	c	c	0
e	b, c	$b \wedge c$	1
f	a	a	1
g	e, f	$e \vee f$	1

(a) The instance $\langle G, \overline{\gamma}, O^0 \rangle$



(b) Social network G

Figure 4.2: This figure describes the social network and the agents' initial opinions in Example 4.2. Table (a) on the left-hand side gives every agent's neighbourhood, update function and initial opinion. On the right-hand side, Figure (b) depicts the social network G where the box under an agent's name gives their neighbourhood of influencers and their initial opinion.

We now follow the synchronous diffusion $\circ$ on $\langle G, \overline{\gamma}, O^0 \rangle$. Agent a updates their opinion from 0 to 1 when either d or g are for the issue in the previous iteration. As $o_g^0 = 1$ at time $t = 1$, we have that $o_a^1 = 1$. Next, we address the opinion update of b at time $t = 1$, as $o_a^0 = 0$, we see that b's update function evaluates to false; thus, $o_b^1 = 0$. Following this, we arrive at $\circ(G, \overline{\gamma}, O^0, 1) = O^1 = (1, 0, 1, 0, 0, 0, 1)$. The opinions at $t = 1$ are used at time $t = 2$ giving $O^2 = (1, 1, 0, 1, 0, 1, 0)$. Following this process iteratively, we have:

$$O^3 = (1, 0, 1, 0, 0, 1, 1); \qquad O^4 = (1, 1, 0, 1, 0, 1, 1); \qquad O^5 = (1, 0, 1, 0, 0, 1, 1).$$

The instance is not stable using $\circ$ as $O^3 = O^5 \neq O^4$. Thus, the process would alternate between O^3 and O^4. Note that the opinions of a, e, f and g are stable. $\triangle$

4.2.1 Restricted Languages for Update Functions

We let $\mathcal{L}$ denote a language for update functions such that $\gamma \in \mathcal{L}$ if and only if γ abides by the criteria of $\mathcal{L}$. Generally, we assume that all Boolean formulas are in DNF. In threshold models, update functions are compactly represented as quota, i.e., $i \in \mathcal{N}$ has opinion $o_i^{t+1} = 1$ if and only if $\sum_{j \in N_i} o_j^t \geq q$ for some quota $q \in \mathbb{N}$. Expressing such a quota as a propositional formula leads to an exponential blow-up in constraint size when the quota depends on the input. As such, quotas like the majority rule would give an exponential blow-up when $q = n/2$. Yet this is not the case when the quota is a constant, e.g., needing only two approvals $q = 2$. We denote the formulas corresponding to quota rules with $\mathcal{L}^{\text{quota}}$. Another restriction on the update functions we study is $\mathcal{L}^+$, where update functions are Boolean functions that do not contain negated literals.

4.3 The Complexity of Convergence

This section examines the computational complexity of stability, i.e., the problem of detecting if a given initial configuration leads to a stable state. To follow the literature, we refer to the problem as *convergence.*

Convergence-$\mathcal{L}$	
Given:	An instance of Boolean opinion diffusion $\langle G, \overline{\gamma}, O^0 \rangle$ with every $\gamma \in \overline{\gamma}$ such that $\gamma \in \mathcal{L}$
Question:	Does the diffusion process stabilise on $\langle G, \overline{\gamma}, O^0 \rangle$?

To prove PSpace-completeness for Convergence-$\mathcal{L}$, we use the reduction given by Chistikov et al. [2020] for Convergence-**Maj** where **Maj** is the family of majority updates. In their proof, however, **Maj** is represented compactly as a quota rule for each agent. In contrast, **Maj** can only be represented as an exponential Boolean formula (for example, listing all the possible majorities). We refer to an instance of their model as $\langle G, \mathbf{Maj}, O^0 \rangle$. We first prove a lemma that allows us to translate an instance of $\langle G, \mathbf{Maj}, O^0 \rangle$ into a Boolean network in polynomial time by adding some dummy agents.

Lemma 4.1. *For every majoritarian opinion diffusion instance $\langle G, \mathbf{Maj}, O^0 \rangle$ we can create a Boolean opinion diffusion instance $\langle G', \overline{\gamma}, O'^0 \rangle$ in $\mathcal{O}(n^3)$ time that converges exactly when $\langle G, \mathbf{Maj}, O^0 \rangle$ does.*

Proof. This proof uses the connection between quota rules and budget constraints. We consider budget constraints on issues with unary weights. For some budget b, a truth assignment on the issues will be consistent with the constraint when less than or equal to b issues are accepted. Note that quota rules are made true if there are

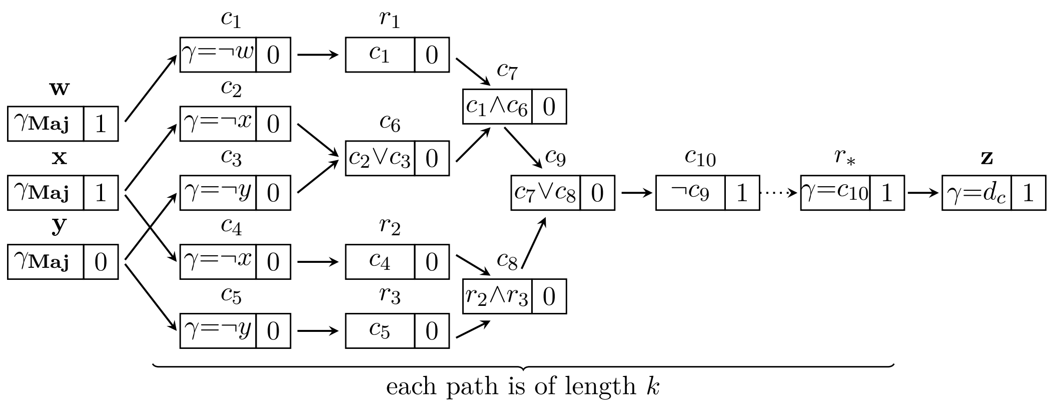


Figure 4.3: This diagram shows how to create a DNNF circuit to mimic the majority update function of agent **z** who is influenced by **w**, **x** and **y**. The circuit nodes $C_{\mathbf{z}}$ represent a circuit reflecting if *at most one* of their influencers is for the issue with c_9 giving this outcome. Thus, c_{10}'s opinion reflects the majority opinion of **z**'s influencers. The regulatory nodes ensure that each path from the influencers to **z** is of length k, either within the circuit (as seen by r_1, r_2 or r_3) or the nodes which can appear between c_{10} and r_* (represented by the dotted line).

at least the quota number of acceptances, while budget constraints are true when there are less than a given number of acceptances. Hence, the negation of a quota rule is a budget constraint on issues with unary weights.

Our proof relies on the fact that every budget constraint can be translated into Boolean circuits in decomposable negation normal form (DNNF) in polynomial time [De Haan, 2018]. A circuit has leaf nodes labelled with either $\top, \bot, x, \neg x$ for any variable x. Furthermore, each internal node is labelled with $\wedge$ or $\vee$ dictating the operation performed on its children's values to determine its own value. A DNNF circuit must be directed, acyclic, and have a single root. Moreover, for every conjunction in the circuit, each of its conjuncts cannot share variables. Finally, note that any budget constraint over a set of costed issues $\mathcal{I}$ and budget limit B can be represented by a DNNF circuit in $B + |\mathcal{I}|$-time [De Haan, 2018, Theorem 16]. A clear connection to Boolean opinion diffusion can be made where parent nodes are influenced by their children with respect to their labels. Thus, translating a node from a DNNF circuit into a Boolean network will let its children become its influencers and its label become its update function. Furthermore, the circuit's leaf nodes either represent an accepted or rejected issue (negation). Thus, the input of a circuit is the opinions of an agent's influencers.

We use these DNNF circuits to encode the majoritarian opinion diffusion in our Boolean setting (moving from $\langle G, \mathbf{Maj}, O^0\rangle$ to $\langle G', \overline{\gamma}, O'^0\rangle$). We create dummy agents between each agent $i \in \mathcal{N}$ from G and their in-neighbours $N_i \neq \emptyset$ (recall that if $N_i = \emptyset$, their opinions never update). These dummy agents are either part of a DNNF circuit C_i or are *regulatory* agents in R_i. The dummy agents in C_i allow the final agent in the circuit's opinion to reflect the majority opinion of i's in-neighbours. The dummy agents in R_i ensure that every path from the agents in N_i to i is of length k (the maximum path length of any required circuit). This ensures that the opinions of the original agents in $\mathcal{N}$ will be updated simultaneously.

We give an example of this translation in Figure 4.3. It translates the majority update function of $\mathbf{z}$ who is influenced by $\mathbf{w}$, $\mathbf{x}$ and $\mathbf{y}$ via a DNNF circuit. The dummy agents who are part of the budget limit circuit $C_{\mathbf{z}}$ ensure that the opinion of c_9 is 1 if at most one of the agents $\mathbf{w}$, $\mathbf{x}$ or $\mathbf{y}$ has the opinion 1. Thus, as c_{10}'s update function represents the negation of c_9's opinion, c_{10}'s opinion will represent the majority opinion of $\mathbf{w}$, $\mathbf{x}$ and $\mathbf{y}$. The regulatory nodes $R_{\mathbf{z}}$ before c_{10} ensure that the opinions of $\mathbf{w}$, $\mathbf{x}$ and $\mathbf{y}$ reach c_{10} after the same number of steps. The regulatory nodes from c_{10} to r_* ensure this majority opinion reaches the original agents of $\mathcal{N}$ at the same step.

When an agent has an even number of influencers (i.e., $|N_i|$ is even), the strict majority rule on N_i requires a different budget limit depending on the agent's current opinion. This would require two circuits for the two different budget limits. The use of the circuit would depend on the current opinion of the agent.

The translation requires at most $2n$ circuits (at most 2 circuits per agent in $\mathcal{N}$). Each circuit being found in $B + |\mathcal{I}| = \lfloor \frac{|N_i|}{2} \rfloor \pm 1 + |N_i|$ time. Hence, at most $\mathcal{O}(2n^2)$ time is required to build every circuit, including adding the regulatory nodes. The initial opinions of these agents are the final information needed to create $\langle G', \overline{\gamma}, O'^0 \rangle$. All original agents retain their initial opinions. If the dummy agents of agent i appear before the final circuit agent (c_{10} in Figure 4.3), then they have the opposite initial opinion to agent i, $1 - o_i^0$. All subsequent dummy agents (c_{10} and the dummy agents coming after c_{10} in Figure 4.3) have the same initial opinion o_i^0. This translation can be found in polynomial time.

Claim: For any step t and any $\ell \in [1, k]$ (where k is the largest depth of any C_i), it is the case that $O_i^t = O_i'^{((t-1)k+\ell) \bmod k}$ always holds for any non-dummy agent $i \in \mathcal{N}$.

We prove this claim by induction.

Base Case: Starting when $t = 0$, $\langle G, \mathbf{Maj}, O^0 \rangle$ has the initial opinions O^0, we need to check that in the first k steps ($\ell \in [1, k]$) of $\langle G', \overline{\gamma}, O'^0 \rangle$ that for all $i \in \mathcal{N}$ that $o_i'^{\ell} = o_i^0$. In the first k steps of $\langle G', \overline{\gamma}, O'^0 \rangle$, the initial opinions of an agent's influencers in G have not reached them yet. This is due to their being k agents between i and the agents in N_i, namely the agents in C_i and R_i. Secondly, the opinion of i will not change before then due to the initial opinions of the dummy agents being chosen so that they will not change i's opinion.

Inductive Hypothesis: The inductive hypothesis assumes that for all previous steps t and for any $\ell \in [1, k]$ that $o_i^t = o_i'^{((t-1)k+\ell) \bmod k}$. We now show this for $t + 1$.

Inductive Step: At the step kt of $\langle G', \overline{\gamma}, O'^0 \rangle$, we see that for an arbitrary agent $i \in \mathcal{N}$ that their influencers in N_i will influence the leaf nodes of C_i. By the inductive hypothesis, we know that the opinions of these agents from N_i have been static for the last k iterations. The opinions of the agents in N_i update at time kt.

Thus, the update of their opinions will only affect the opinion of i after k steps, as this is the number of agents between i and N_i. Thus, $o_i'^{kt+\ell}$ is static for the next k steps.

By the previous claim, when $\langle G', \overline{\gamma}, O'^0 \rangle$ is built from $\langle G, \mathbf{Maj}, O^0 \rangle$, then either both or neither will converge. The isolated nature of the dummy agents (as $(C_i \cup R_i) \cap (C_j \cup R_j) = \emptyset$ for any $i \neq j \in \mathcal{N}$) means that their opinions only update to update the non-dummy agents. Due to the periodic nature of opinion updates in $\langle G', \overline{\gamma}, O'^0 \rangle$ when there is no new input to the circuit, the opinions of the dummy agents $C_i \cup R_i$ also do not change. Thus, neither does the opinion of i.

Finally, this process can be completed in $\mathcal{O}(n^3)$ time: $\mathcal{O}(2n^2)$ time is needed to build the circuits; $\mathcal{O}(n^3)$ time is needed to alter all of the circuits to be of the same length; then at most $\mathcal{O}(nk)$ time is needed to add the remaining agents in R_i. □

Proposition 4.1. *Convergence-$\mathcal{L}$ is PSpace-complete.*

Proof. To show membership of Convergence-$\mathcal{L}$ in Pspace, we need to ensure that the problem requires no more than polynomial space. We require two vectors with n entries, the first storing the agents' current opinions and the second storing the updated opinions which use the current opinions. Furthermore, we have a counter that counts the number of iterations completed in the diffusion process. This counts maximally to $2^n + 1$, which can be represented with a polynomial amount of space, $n + 1$ bits when writing the number in binary. In addition, we need a polynomial amount of space to compute each agent's update function. Since model checking of Boolean functions can be done in polynomial time, it can also be done in polynomial space. The answer is a "yes" when the two vectors are equal. However, the answer is "no" when the counter has reached $2^n + 1$. At this point, we are sure there is a cycle among the opinions. Thus, Convergence-$\mathcal{L}$ is in Pspace.

To prove Pspace-hardness for Convergence-$\mathcal{L}$, we reduce from the problem Convergence-**Maj** which was shown to be Pspace-complete by Chistikov et al. [2020, Theorem 1]. The reduction is provided by Lemma 4.1, which shows that we can translate every majoritarian opinion diffusion to Boolean opinion diffusion in polynomial time.

As Convergence-$\mathcal{L}$ is in PSpace and Pspace-hard, we can conclude that Convergence-$\mathcal{L}$ is Pspace-complete. □

Remark 4.1. A consequence of Proposition 4.1 is that checking the necessary and sufficient stability conditions given by Christoff and Grossi [2017b, Lemma 3] is a Pspace-complete problem.

We study another decision problem from Chistikov et al. [2020], asking if there is an initial set of opinions for a social network such that the process does not stabilise.

Guarantee-Convergence-$\mathcal{L}$	
Given:	A network G and $\overline{\gamma}$ such that for all $\gamma \in \overline{\gamma}$, $\gamma \in \mathcal{L}$
Question:	Is there an O^0 such that $\langle G, \overline{\gamma}, O^0 \rangle$ does not stabilise?

Proposition 4.2. *Guarantee-Convergence-$\mathcal{L}$ is PSpace-complete.*

We provide a proof sketch as it is similar to the proof of Proposition 4.1.

Proof Sketch. Membership of Guarantee-Convergence-$\mathcal{L}$ in PSpace uses the basic idea used for proving membership in Proposition 4.1. The only difference is that we must repeat the process for all initial opinions until one stabilises. Hence, we store an extra vector with the current initial set of opinions in a vector with n entries. A "yes" answer is found when the diffusion process is shown not to stabilise on some initial set of opinions. A "no" answer is found when all possible initial opinions have been shown to stabilise. Note that both can be done with polynomial space. Thus, Guarantee-Convergence-$\mathcal{L}$ in PSpace.

A similar reduction can obtain hardness to the one in Proposition 4.1 using Theorem 2 from Chistikov et al. [2020]. Hence, Guarantee-Convergence-$\mathcal{L}$ is PSpace-complete. □

Remark 4.2. Convergence-$\mathcal{L}$ and Guarantee-Convergence-$\mathcal{L}$ are in P when the network is restricted such that for every $i \in \mathcal{N}$ we have that $|N_i| \leq 1$. This relates to the correspondence between opinion diffusion and liquid democracy observed by Christoff and Grossi [2017a].

Remark 4.3. To our knowledge, this result has yet to be shown in the context of Boolean networks. The closest result previously studied was checking if an agent's opinion is stable when updates happen block-sequentially, which is a PSpace-complete problem [Goles et al., 2016]. Hence, our results are potentially useful for applications of Boolean networks. For example, checking if there is a fixed point in a gene regulatory network starting from a particular initial state.

4.4 Asynchronous Updates

This section extends the asynchronous majoritarian opinion diffusion model results from Bredereck and Elkind [2017] to see if they still hold in our model. In particular, they give an asynchronous procedure to find a sequence of agents such that the diffusion process stabilises on a profile of opinions and maximises the agreement on the issue. We now extend our model to be able to account for asynchronous updates.

Following the notation of Bredereck and Elkind [2017], we let $\sigma \in 2^{\mathcal{N}} \times \cdots \times 2^{\mathcal{N}}$ be the sequence in which the updates happen. Note that when $\sigma = (\mathcal{N}, \cdots, \mathcal{N})$, we regain the synchronous model (all agents updating simultaneously). Generally, asynchronous updates ensure that every entry in σ is a singleton (one agent updating at a time). Most cases between these two extremes relate to *block sequencing*, studying when subsets of agents can update their opinions synchronously [Goles et al., 2016]. We slightly abuse notation by letting $\circ$ denote the asynchronous update function, which takes an instance $\langle G, \overline{\gamma}, O^0 \rangle$ and a sequence σ and returns a profile of opinions $O^{|\sigma|}$ found by following σ. Thus, $\circ[\langle G, \overline{\gamma}, O^0 \rangle, \sigma] = O^{|\sigma|}$. If we want to look at step t of this sequence, we let $\circ[\langle G, \overline{\gamma}, O^0\rangle, \sigma, t] = O^t$. One can distinguish an asynchronous update from a synchronous update by the presence of σ in the input.

Bredereck and Elkind [2017] show that when all agents use the majority update function, an asynchronous update sequence always exists such that the process stabilises. However, this does not hold in our general model due to the possibility of negations in the update functions.

Proposition 4.3. *For some $\langle G, \overline{\gamma}, O^0\rangle$, there may exist no σ such that $\circ[\langle G, \overline{\gamma}, O^0\rangle, \sigma] = O^{|\sigma|}$ where $O^{|\sigma|}$ is stable.*

Proof. Consider the following counter-example with no stable profiles. Let $\mathcal{N} = \{a, b\}$ with update functions $\gamma_a = b$ and $\gamma_b = \neg a$. We let the initial opinions be $O^0_a = 1$ and $O^0_b = 0$. Consider the following profiles of opinions for $x \in \{0, 1\}$:

(x, x): this is not stable as b wants to update their opinion to $1 - x$;

$(x, 1 - x)$: this is not stable as a wants to update their opinion to $1 - x$.

As there is no stable profile of opinions for this social network, this entails that there is no sequence that leads to a stable profile of opinions. $\square$

Proposition 4.3 is a negative result showing that when the update functions contain negated literals, there is no longer a guarantee of a stable outcome. However, we show that this is no longer true when restricting update functions to $\mathcal{L}^+$. Although we focus on $\mathcal{L}^+$, an analogous result can be shown when update functions only contain negated literals.

Bredereck and Elkind [2017] look at sequences that not only stabilise the diffusion process but also maximise or minimise the number of 1s in the stable profile, namely, the *optimistic* or *pessimistic* updates. An update sequence is optimistic (respectively, pessimistic) if it leads to a stable profile of opinions and maximises (respectively, minimises) the number of 1s in the final state. We now show that Proposition 1 from Bredereck and Elkind [2017] extends to our model when update functions contain only positive Boolean functions.

Proposition 4.4. *For every instance of Boolean opinion diffusion $\langle G, \overline{\gamma}, O^0\rangle$ with every $\gamma \in \overline{\gamma}$ such that $\gamma \in \mathcal{L}^+$, there exists an optimist (resp. pessimistic) update sequence σ such that (i) σ is asynchronous, (ii) $|\sigma| \leq 2n$, (iii) every agent $i \in \mathcal{N}$ changes their opinion at most twice, (iv) σ can be computed in $\mathcal{O}(\ell n^2)$ time (where ℓ is the maximum time for model checking for any $\gamma_i \in \overline{\gamma}$), (v) if σ leads to the stable collective opinion $O^{|\sigma|}$, then every other optimistic (resp. pessimistic) update sequence σ^* also gives $O^{|\sigma^*|}$.*

Proof. We emphasise that the following proof follows the same steps and is very similar to the proof of Proposition 1 from Bredereck and Elkind [2017], which was in turn inspired by a similar proof from Chierichetti et al. [2013]. We first give the procedure for the optimistic update, which has two phases. Note that the pessimistic update takes two steps in the opposite order. Then, without loss of generality, we only give the proof sketch for the optimistic update, as the proof for the pessimistic update is very similar.

We initially start with an empty sequence σ and proceed with the following two phases, only moving to phase two when there are no more changes available in phase one:

1. If $o_i^t = 0$ and $O^t_{\restriction N_i} \vDash \gamma_i$ then append σ with $\{i\}$ and let $o_i^{t+1} = 1$;

2. If $o_i^t = 1$ and $O^t_{\restriction N_i} \vDash \neg\gamma_i$ then append σ with $\{i\}$ and let $o_i^{t+1} = 0$.

This procedure is *(i)* asynchronous, as only a single agent's opinion is changed at any time. *(ii)* and *(iii)* are also true as each agent can only have their opinion changed once in each phase. Thus, each agent can maximally have their opinion changed twice in the sequence; as a result, the sequence's length is such that $|\sigma| \leq 2n$.

We now show *(iv)* that the procedure terminates in $\mathcal{O}(\ell n^2)$ time. We let ℓ be the maximum time required to check whether an opinion should be updated. This is model-checking and can be done in a polynomial time. In each phase, there are at most n iterations where at each step t we have to check all agents with the opposite opinion. This is at most $n - t$ checks, each taking ℓ steps. Thus, one phase can take at most $\mathcal{O}(\ell\frac{n(n+1)}{2})$ time. Thus, both phases can be computed in $\mathcal{O}(\ell n^2)$ time.

For *(v)*, we first show that the procedure leads to a stable profile of opinions. We assume that there is an agent $i \in \mathcal{N}$ who wants to update their vote after the sequence given by the procedure to gain a contradiction. We now study two cases. The first case is if $O_i^{|\sigma|} = 1$ and the second is if $O_i^{|\sigma|} = 0$. If $O_i^{|\sigma|} = 1$ yet $O^t_{\restriction N_i} \vDash \neg\gamma_i$, then the procedure is not over as i's opinion would need to be updated at this point in phase 2. On the other hand, if $O_i^{|\sigma|} = 0$ yet $O^t_{\restriction N_i} \vDash \gamma_i$, then it would have also been the case at the end of phase 1 (as $\gamma_i \in \mathcal{L}^+$, there is some cube of γ_i such that all of these neighbours have the opinion 1, which would have also been the case at the end of phase 1). Thus, at the end of phase 1, i's opinion should have been updated to 1. In both cases, we have reached a contradiction and the procedure always leads to a stable profile of opinions.

Finally, we need to show that any other optimistic sequence σ^* gives the same profile of opinions, $\circ[\langle G, \overline{\gamma}, O^0\rangle, \sigma] = \circ[\langle G, \overline{\gamma}, O^0\rangle, \sigma^*]$. We show this via the following two cases: first, that every opinion changed from 0 to 1 in the sequence σ^* was also flipped under σ; second, that every vertex flipped from 1 to 0 under σ is also flipped under σ^*. As the two cases are similar, we only give proof of the first case. We prove the first case via a contradiction. Assume that σ and σ^* do not give the same profile of opinions. We let $i^* \in \mathcal{N}$ be the first agent such that their opinion was changed from 0 to 1 under σ^* (at step k) yet not under σ. By assumption, for all steps $k' < k$, the agents' opinions changed from 0 to 1 under σ^* were also changed in σ as i^* was the first agent with a differing opinion. Thus, $\circ[\langle G, \overline{\gamma}, O^0\rangle, \sigma*, k-1]$ is such that $O^{k-1}_{\restriction N_{i^*}} \vDash \gamma_{i^*}$. Therefore, if enough neighbours of i^* have the opinion 1 such that i^* can change their opinion to 1, then in phase

1 of σ, i^*'s opinion would be flipped from 0 to 1. Therefore, we have reached a contradiction, concluding that *(v)* is true. □

4.5 Multi-Agent Delegations as Opinion Diffusion

Following the work of Christoff and Grossi [2017a], we study the connection between opinion diffusion and delegative democracy. In liquid democracy, an agent can either vote directly or delegate their vote to another agent, who can transitively delegate their vote and any votes they have received to another agent. Thus, a delegation can be seen as an agent influenced by their delegate. There are, however, some differences between the two models. For instance, delegating agents in liquid democracy typically have no initial opinion and agents who have an initial opinion (a direct voter) are not influenced by other agents.

Christoff and Grossi [2017a] were the first to make the connection explicit between liquid democracy and opinion diffusion for the case of delegations to a single agent. We extend this to delegative democracy models with multi-agent delegations in line with Boolean networks.

In Chapter 2, we proposed a model where ballots allow for multi-agent ranked delegations. This model extends liquid democracy in two regards: first, an agent's delegation can use the votes of many other agents; second, ballots can contain ranked delegations to avoid delegation cycles. We will study a restricted version of this model that does not include ranked delegations. Consider a set of $\mathcal{N} = \{1, \cdots, n\}$ agents (or voters) who vote on a single binary issue. Each agent $i \in \mathcal{N}$ gives a ballot:

$$B_i \in \{(S_i, F_i) \mid S_i \subseteq \mathcal{N} \backslash \{i\}, F_i : S_i \to \{0,1\}\} \cup \{0,1\}.$$

Thus, every agent either delegates or votes directly. Note that S_i is a subset of agents acting as i's delegates whose votes determine i's according to the Boolean function F_i.

In Chapter 2, we defined six *unravelling procedures* that take the agents' ballots and return a profile of votes by resolving delegations. All of these procedures are equivalent when considering ballots without ranked delegations. Thus, we refer to them as Unravel. Unravel iteratively adds votes from delegations synchronously, stopping when no more votes can be added from one iteration to the next.

We now translate multi-agent delegative democracy into our synchronous Boolean opinion diffusion model as described in Section 4.2. The set of agents remains the same $\mathcal{N}$. The edges of G are determined by the agents' delegates; thus $E = \{(j, i) \mid j \in S_i\}$. We introduce a third opinion $*$ representing abstention to allow the models to align. We now introduce a language for the update functions in this setting to account for the new ternary domain of opinions. We say an update function $\gamma \in \mathcal{L}_{\texttt{update}*}$ if $\gamma : \{0,1,*\} \to \{0,1,*\}$ and for $O \in \{0,1,*\}^{|Var(\gamma)|}$ computing $\gamma(O)$ can be done in polynomial time. This is equivalent to finding a necessary winner of a Boolean function on a partial assignment [Konczak and Lang,

if $B_i = 1$	$O_i^0 = 1;$	$\gamma_i = \top$
if $B_i = 0$	$O_i^0 = 0;$	$\gamma_i = \bot$
if $B_i = (S_i, F_i)$	$O_i^0 = *;$	$\gamma_i = F_i$

$$O_i^{t+1} = \begin{cases} 1 & \text{if } O^t_{\restriction N_i} \models \gamma_i; \\ 0 & \text{if } O^t_{\restriction N_i} \models \neg\gamma_i; \\ *, & \text{otherwise.} \end{cases}$$

Table 4.1: For $i \in \mathcal{N}$, the left-hand-side table gives i's initial opinion and update function from their ballot; on the right-hand-side, it shows how to compute a delegating agent's opinion update.

2005]. For example, the necessary winner of a Boolean function in complete DNF[2] can be computed in polynomial time and would fall into this category (as shown in Proposition 2.1). On the left-hand side of Table 4.1, we see how we translate an agent's multi-agent delegative democracy ballot into an initial opinion and update function. Recall that if an agent's update function is $\top$ or $\bot$, they do not have any influencers.

For a delegating agent $i \in \mathcal{N}$, their update function $\gamma_i = F_i$ can take as input $\{0, 1, *\}$ even though γ_i is represented as a propositional formula.[3] The right-hand-side of Table 4.1 shows how to compute their opinion when the input of γ_i can contain abstentions, $O^t_{\restriction N_i} \in \{0, 1, *\}^{|N_i|}$.

Following the diffusion process described in Section 4.2, we prove a lemma that shows that the diffusion process always terminates in a stable state.

Lemma 4.2. *Let $\langle G, \overline{\gamma}, O^0\rangle$ be such that $\overline{\gamma} \in \mathcal{L}_{\texttt{update}*}$. If at some time t we have that $O_i^t = v \in \{0, 1\}$, then $O_i^{t'} = v$ for all steps $t' > t$.*

Proof. We prove this lemma by induction on the step t, showing that if $O_i^t \in \{0, 1\}$, then it changes for no $t' > t$.

Base Case: As $O_i^0 \in \{0, 1\}$, this means that $\gamma_i = \bot$ or $\top$, respectively. As their update functions are constant, their opinion remains static at all steps $t \geq 1$, $O_i^t = O_i^0$.

Inductive hypothesis: For some step t, every $i \in \mathcal{N}$ such that $O_i^t \in \{0, 1\}$, their opinion does not change in any subsequent step, $O_i^{t'} = O_i^t$ for all $t' > t$.

Inductive step: We want to show that given the inductive hypothesis is true at t, it remains true at step $t+1$. By assumption, we know that all agents with an opinion of 0 or 1 in O^t are such that their opinions will not change in future steps. For the agents in $S \subseteq \mathcal{N}$ such that $S = \{i \mid O_i^t = * \text{ and } O_i^{t+1} \in \{0, 1\}\}$, we want to show that for each $i \in S$ that their opinion will not change after $t + 1$. For an arbitrary $j \in S$, we assume that $O_j^{t+1} = 1$ without loss of generality. Therefore, $O^t_{\restriction N_j} \models \gamma_j$. We let $V_j \subseteq N_j$ be the agents of N_j such that they have a non-abstaining vote in

[2]The necessary winner of an update function γ in complete DNF is 1 if and only if there exists at least one cube of the formula where every literal is true, and the necessary winner is 0 if and only if every cube of the formula is made false by at least one literal.

[3]Note the similarities between this way of resolving delegations and in Section 3.6.1 where we allowed for abstentions in the domain of alternatives and issue-wise delegation cycles. In both settings, necessary winners are used to avoid the problem of partial binary truth assignments. The difference between them is that here we are interested in a single issue, whereas in Section 3.6.1, we are interested in multiple issues.

O^t, hence, that $O^t_{\restriction V_j} \vDash \gamma_j$. By our inductive hypothesis, the votes of the agents in V_j will not change after time t. As j's vote changes at this step, a necessary winner of γ_j was found from the votes in V_j. As no agent's vote in V_j will change, neither will the necessary winner of γ_j, no matter the votes of the agents in $N_j \backslash V_j$. As j was chosen arbitrarily, the votes of all agents in S do not change after $t+1$. Our inductive hypothesis has been shown. □

From Lemma 4.2, we see that the process terminates as only a finite number of opinion updates can be made.

Corollary 4.1. *The diffusion processes terminate on $\langle G, \overline{\gamma}, O^0 \rangle$ when $\overline{\gamma} \in \mathcal{L}_{\mathit{update*}}$.*

Although the process terminates, we remark that this does not necessarily mean that all agents have an opinion in $\{0, 1\}$. Furthermore, the outcome found by the diffusion process is the same as Unravel. As Unravel terminates in polynomial time (see Proposition 2.4), we now show that the diffusion process also does. We let Terminate-$\mathcal{L}_*$ be the functional problem that given an instance $\langle G, \overline{\gamma}, O^0 \rangle$ with $\overline{\gamma} \in \mathcal{L}_{\mathtt{update*}}$ finds the stable profile of opinions found by the diffusion process.

Proposition 4.5. *Terminate-$\mathcal{L}_*$ is in P.*

Proof. Recall that finding a necessary winner of a $\gamma \in \mathcal{L}_{\mathtt{update*}}$ can be done in polynomial time; let ℓ be the maximum amount of time required for any $\gamma \in \overline{\gamma}$. Lemma 4.2 tells us that when an opinion is in $\{0, 1\}$, it does not change. Thus, at most n diffusion iterations, all agents will have their update function checked for a necessary winner. Therefore, the process terminates in $\mathcal{O}(n^2 \ell)$ time. □

Proposition 4.5 shows that our diffusion process can unravel a multi-agent delegation profile in polynomial time, giving the same computation complexity bound as Unravel (Proposition 2.4).

4.5.1 Control in Multi-Agent Delegation

One common area of research in opinion diffusion is opinion control [Akutsu et al., 2006, Langmead and Jha, 2009]. We extend this to our model of opinion diffusion when focusing on multi-agent delegative democracy. This section focuses on the computational complexity of being able to change the outcome of the collective decision by bribing a given number of agents to change their votes. Here, we look at ensuring that the collective outcome is for the issue. Note that the problem when ensuring that the collective outcome is against the issue is equivalent. We focus on the collective outcome being determined by a quota rule while the update function remains expressed as propositional formulas in $\mathcal{L}_{\mathtt{update*}}$, i.e., given an assignment $O \in \{0, 1, *\}^{|Var(\gamma)|}$, then $\gamma(O)$ can be computed in polynomial time. Moreover, by an instance $\langle G, \overline{\gamma}, O^0 \rangle$ reflecting multi-agent delegative democracy, we mean that from a set of agents $\mathcal{N}$ who give a profile of multi-agent delegations as $\boldsymbol{B}$, the translation (as given at the beginning of the section) has been undertaken to obtain

G, $\overline{\gamma}$, and O^0. Hence, $G = (\mathcal{N}, E)$ with $E = \{(j,i) \mid j \in S_i\}$ and the pair $\overline{\gamma}$ and O^0 are found by the description given in Table 4.1.

Quota-Control	
Given:	An instance $\langle G, \overline{\gamma}, O^0 \rangle$ reflecting multi-agent delegative democracy such that for all $\gamma \in \overline{\gamma}$ we have that $\gamma \in \mathcal{L}_{\texttt{update}*}$ and constants $k \in \mathbb{N}$ and $q \in [0, n]$
Question:	Is there a $D \subseteq \mathcal{N}$ such that $\lvert D \rvert \leq k$ and for all $i \in D$ changing $\gamma_i = \top$ and $O_i^0 = 1$ gives a stable profile of opinions O^T such that $\sum\limits_{o_i \in O^T} o_i \geq q$?

Informally, Quota-Control asks if there is a subset of agents D, who by bribing them to change their ballot to be for the issue means that there are at least q agents in the stable profile of opinions in favour of the issue.

Proposition 4.6. *Quota-Control is an NP-complete problem.*

Proof. Quota-Control can be shown to be in NP by checking a certificate in polynomial time. The certificate lists the agents in D whose update function and initial opinion will be changed to represent a direct vote for the issue. We make these changes to the instance, giving $\langle G', \overline{\gamma}', O'^0 \rangle$. As shown in Proposition 4.5, the diffusion process on such an instance terminates in polynomial time. Then, it is required to check if the stable profile of opinions on termination exceeds the quota. Therefore, a certificate can be checked in polynomial time, and Quota-Control is in NP.

We show NP-hardness of Quota-Control by giving a reduction from the NP-complete problem *Feedback Vertex Set*, FVS [Karp, 1972]. FVS takes as input a directed graph $G = (V, E)$ and $k \in \mathbb{N}$ and then asks if there is a subset $X \subseteq V$ such that $\lvert X \rvert \leq k$ and the remaining graph when only considering the vertices $V \backslash X$ is cycle-free.

The translation of FVS to our problem lets the nodes remain the same $\mathcal{N} = V$ and for each $i \in \mathcal{N}$, N_i is determined by E. Each agent's update function depends on their neighbourhood: for each $i \in \mathcal{N}$ if $N_i = \emptyset$ then $O_i^0 = 1$ and $\gamma_i = \top$; else $O_i^0 = *$ and $\gamma_i = \bigwedge_{j \in N_i} j$. Note that the update functions of each delegating agent will update to 1 only when every one of their neighbours has the opinion 1. Otherwise, it will be $*$. Note that no agent's vote can be against the issue as no agents have the initial opinion of 0 and there are no negations in the delegation functions. The quota represents unanimity $q = n$, i.e., the collective decision is 1 only when there is a consensus.

First, assume that we have a solution to FVS, and we want to show that there is also a solution to Quota-Control. Given X, we change the update functions and the initial opinions of the agents in X. As X is a solution to FVS, we see that the remaining network is cycle-free and all votes can be assigned. As all direct voters vote for the issue, every opinion on termination will be 1, and our quota

$q = n$ has been met. Next, we assume that there is no solution to FVS. Therefore, more than k nodes need to be removed to make the network cycle-free. Thus, no matter which agents' initial opinions and update functions are changed, there will still be at least one cycle. For each i in this cycle, no necessary winner will be found for $\gamma_i = \bigwedge_{j \in N_i} j$. Their opinion remains as $*$, as no opinions can be 0 at any diffusion stage. Thus, a necessary winner can only be found when each of the agents in i's neighbourhood is in favour of the issue. Therefore, for every agent still in a cycle, their opinion at termination is $*$. Thus, the quota cannot be met and the final collective opinion is not 1. Therefore, there is no solution either for Quota-Control. We have shown NP-hardness and membership for Quota-Control. Thus, it is NP-complete. □

This result is unsurprising given that an equivalent in majoritarian opinion diffusion is known to be an NP-hard problem [Kempe et al., 2003] and manipulation via bribery remains an intractable problem in this voting scenario.

4.6 Results from the Boolean Network Literature

This chapter connects the well-established research area of Boolean networks (BN) and opinion diffusion. BN have impacted many different disciplines, such as cryptography [Cardell and Fúster-Sabater, 2019], ecological complexity [Gaucherel et al., 2017], and biomedical sciences, such as estrogen transcriptional regulation [Anda-Jáuregui et al., 2019], and most notably, regulatory gene networks. The model used in this paper aligns with standard BN; thus, many results can be translated into our model with only a few details to be checked. The following remarks take known results from the BN literature and rephrase them in terms of Boolean opinion diffusion.

Remark 4.4. Akutsu et al. [1999] showed that a unique Boolean network can be found in polynomial time from a sequence of opinion profiles when the number of agents in any in-neighbourhood is bounded by some constant, $|N_i| \leq k$ for all $i \in \mathcal{N}$ and $k \in \mathbb{N}$.

Remark 4.5. Farrow et al. [2004] showed that finding a stable profile of opinions for a Boolean opinion diffusion instance where the network is monotonic[4] is NP-complete.

One BN topic that is not widely studied in opinion diffusion is the effect of negative influence. However, negative influence can be a key reason a network does not stabilise, as seen in Proposition 4.3. We guide the reader to the survey from Richard [2019] for an overview of positive and negative cycles in Boolean networks.

[4]Take any Boolean function F and any $X \in \{0,1\}^{|F|}$ such that $F(X) = 1$, F is monotonic if and only if $F(X') = 1$ still holds for any X' found by changing a single 0 entry in X to a 1. A BN is monotonic if every Boolean function within it is also monotonic.

Remark 4.6. Goles and Salinas [2010] showed that finding a stable profile of opinions can be done in polynomial time when every cycle in the network G has an even number of decreasingly monotonic arcs with respect to the update functions.

It may be sufficient for the opinion diffusion process not to stabilise in some cases if it only cycles through a small number of profiles.

Remark 4.7. Akutsu et al. [2012] showed that in polynomial time, a profile of opinions could be found that leads to a cycle among two profiles of opinions when all $\gamma \in \overline{\gamma}$ are such that $\gamma \in \mathcal{L}^{\vee} \cap \mathcal{L}^{+}$ (functions only containing positive literals and disjunctions).

Fixed points are well studied in the BN literature; in our terminology, this equates to whether a stable collection of opinions exists for a network.

Remark 4.8. Kobayashi [2019] showed that it is an NP-complete problem to check if a stable collection of opinions exists for a Boolean opinion diffusion instance.

This remark follows from the fact that this problem is equivalent to checking if there is a solution to an ILP where the set of constraints contains a constraint $\gamma_i(O_{\restriction N_i}) = o_i$ for each $i \in \mathcal{N}$. In Section 4.3, we studied a similar problem, the difference being that in Proposition 4.1, we ask if a stable state is coming from an initial profile of opinions. The increase in complexity from NP to PSpace comes from the fact that it is hard to verify if a stable profile of opinions can come from a particular initial profile of opinions.

Boolean network control, as defined by Akutsu et al. [2006], asks if, from a given set of agents whose opinions can be controlled, is it possible to gain a particular profile of opinions O^M in M steps by controlling only the given subset of agents.

Remark 4.9. Akutsu et al. [2006] showed that Boolean network control is an NP-complete problem yet is polynomial when the underlying graph is a tree.

4.7 Conclusion and Future Work

This chapter has studied algorithmic problems from opinion diffusion on the model given by Boolean networks. We have shown that it is PSpace-complete to recognise whether a given initial state leads to stability in synchronous diffusion, generalising a known result on majoritarian opinion diffusion. In contrast, when moving to asynchronous diffusion, we showed that the existence of a diffusion sequence leading to stability and maximising consensus could not be guaranteed for arbitrary Boolean networks. However, we demonstrated its existence when negative influence is not allowed. Finally, we showed that stability is guaranteed when a delegative voting problem induces the influence structure. Moreover, we showed that influence maximisation is NP-hard in this restriction of the model. We also rephrased known results from the Boolean network literature in terms of diffusion to showcase the synergy of the two research subjects.

This chapter has opened several directions for future work. Perhaps the most interesting is to explore the use of semi-tensor products in opinion diffusion, as they currently constitute one of the primary techniques used by recent research on Boolean networks (see, e.g., Cheng [2007] and Cheng et al. [2010b]). We conjecture that this will draw a parallel between our setting and using DeGroot processes given by Christoff and Grossi [2017a]. Another area for future work is looking at probabilistic BNs for opinion diffusion, particularly with respect to existing work on control Akutsu et al. [2007].

Following a similar line of research as Akutsu et al. [1999] (see Remark 4.4), one possible direction of future research is extracting BNs from a social network and the opinions over time. For example, extracting the users' update functions from a real-world social network. This would aid in the understanding of how users are influenced on digital democracy platforms and also how we should be modelling influence, i.e., are quotas sufficient to model how humans update their opinions or do we require more complex functions?

Chapter 5

Analysing Classical Liquid Democracy

5.1 Introduction

This chapter returns to the classical setting of liquid democracy, where voters can either vote directly on the issue or delegate their vote to another agent. Specifically, we consider a set of agents $\mathcal{N}$ who can either vote directly on a binary issue $\{0, 1\}$ or delegate to another voter in $\mathcal{N}$ (or some subset of $\mathcal{N}$). Within this model, we assume that delegations are transitive and that cycles can occur. Therefore, delegations are resolved by every voter receiving their ultimate delegate's vote. If a voter is part of a delegation cycle and has no ultimate delegate, then they receive an abstention.

This chapter focuses on new ways to analyse the classic model of liquid democracy and understand its properties. Thus, we are interested in performing *responsive research*, as described in Chapter 1. Section 5.2 does this by extending a priori voting power measures — which are well-studied in many different voting models — to liquid democracy. These measures allow us to analyse the impact of introducing liquid democracy to weighted voting games, asking questions such as *can all voters affect the outcome of the vote?* In Section 5.3, we study a different method of analysing the classical model of liquid democracy. This analysis starts from the observed behaviour of voters in instances of liquid democracy platforms, where delegations have a slightly different purpose. Instead of delegations being given for a one-off election, they are chosen to be used for multiple votes on different issues when voters cannot vote on all issues directly. For instance, Behrens et al. [2022] illustrates the observed voter behaviour where they intentionally create delegation cycles when delegations are given for more long-term purposes. Hence, this model aims to characterise the observed voter strategy in a game-theoretic model.

5.2 A Priori Voting Power in Liquid Democracy

Voting games have been used extensively to study the a priori voting power of voters participating in an election [Felsenthal and Machover, 1998]. A priori voting power refers to the power granted solely by the rules governing the election process to a voter. Notably, these measures do not consider the content of the bill being voted on nor the affinities between voters. It is instead used to quantify the distribution of power in different voting models, allowing the impact of their structure on the collective decision to be compared. The class of I-power measures (where I denotes influence) calculates how likely a voter will influence the outcome. Several I-power measures have been defined, the most well-known being the Penrose-Banzhaf measure in simple voting games [Banzhaf III, 1964, Penrose, 1946].

In simple voting games, an assembly of voters must make a collective decision on a proposal, and each voter may either support or oppose the proposal. Each voter in the assembly can have a different voting weight, which affects their impact on changing the collective decision. For example, it was shown, by using I-power, that the European Union's commission in 1958 had assigned each country's voting weights relative to their population size in such a way that Luxembourg's voting power was zero. Thus, Luxembourg was a dummy agent in this election as their vote never affected the outcome [Felsenthal and Machover, 2004].

As a measure of a priori voting power, the Penrose-Banzhaf assumes that voters vote independently of one another and are equally likely to vote for or against the proposal. Given this probabilistic model of the voters' behaviour, it then measures the probability that a voter can alter the election's outcome. As the study of simple voting games has been extended to account for complex and realistic models, so has the notion of I-power. Such extensions include taking into account abstention [Freixas, 2012], using multiple levels of approval [Freixas and Zwicker, 2003], or coalition structures [Owen, 1981]. These new power indices are crucial in the development of voting models, as they allow the structure of the models to be understood in terms of the voters' *criticality* in deciding the outcome. However, election frameworks with delegations have been largely unexplored so far in terms of a priori voting power and thus make for an interesting research direction.

5.2.1 Contribution

This section highlights the collaborative work undertaken with Théo Delemazure and Hugo Gilbert who are both at LAMSADE, Université Paris-Dauphine (see [Colley et al., 2023a,b] for complete details of proofs and experimental details) on extending the simple voting games to model elections where voters can delegate to one of their neighbours in a social network, modelled as any digraph G. We designed a new I-voting power measure to quantify voters' criticality in each setting. We argue that our power measure is a natural extension of the Penrose-Banzhaf measure, and we illustrate the intuitions behind it through various examples. When G is an arbitrary digraph, we show in this section that the computation of our measure is #P-hard, even when voters' weights are polynomially bounded in the number of voters. Outside of this section, in Colley et al. [2023c], we also looked at restrictions on the underlying social network that allow the measure to be computed in pseudo-polynomial time. These specific underlying networks represented proxy voting (G is a complete bipartite graph) and in an unrestricted version of liquid democracy (G is complete).

5.2.2 Related Work

The inspiration for our line of research comes from the overview given by Felsenthal and Machover [1998].[1] They introduce each of the well-known power indices and measures, such as those presented by Shapley and Shubik [1954], Penrose [1946], Banzhaf III [1964] and Coleman [1971], as well as giving the motivation behind each of them. In this section, we will give some background on the different measures of voting power and the extensions that have already been explored. Finally, we look at the only other paper that inspects measures of voting power in delegative voting.

Voting Power. Measuring a voter's voting power in a specific setting quantifies how *critical* a voter is in casting the deciding vote in the election. For example, a voter i's voting power can be considered as the difference in probability of i voting for the issue when the outcome is also for the issue and i voting for the issue when the outcome does not accept the issue [Gelman et al., 2002]. We recommend the following technical study from Lucas [1974] for an overview of voting power and the work of Felsenthal and Machover [2005] for a historical overview; however, we give an overview of some standard measures.

The most well-known voting power measure is the Shapley-Shubik measure [Shapley and Shubik, 1954]; it quantifies a voter's expected payoff, known as P-power, unlike the other measures we will discuss. Measures of P-power and I-power differ in their motivation. P-power measures quantify the frequency of a voter being in the winning coalition and sharing the coalition's utility between the members, with those not receiving utility 0. In contrast, I-power is more policy-seeking in its motivation and is therefore concerned with the voter's stance on the issue. Hence, they differ in measuring what they regard as an outcome: I-power concerns the passing of a bill, whereas P-power is concerned with the shared power of the winning coalition (thus the longevity of being able to determine the outcomes of the votes). Hence, the Shapley-Shubik power index is defined as a restriction of the Shapley value [Shapley, 1953] to simple voting games. This index can be seen as the probability that a voter will be the last member to join a losing coalition to make it a winning one.

I-power was independently given a mathematical explanation by Penrose [1946], Banzhaf III [1964], and Coleman [1971]. By convention, it is usually called the Banzhaf index, and we thus describe it through this lens. It considers the possible combinations rather than permutations, as in the Shapley-Shubik index. Thus, the *Banzhaf measure* (or absolute Banzhaf index) quantifies the power of an agent i by counting how many profiles of the possible profiles changing i's vote from 0 to 1 cause the outcome to change. The Banzhaf measure is denoted by β', whereas the *Banzhaf index* is the relative quantity denoted by β (found by normalising β'). Note that the concept defined by Penrose [1946] is instead given by a ratio between the voters' power. Therefore, it compares the number of coalitions in which different

[1]The title of our publication [Colley et al., 2023c] is a nod to Chapter 8 of a book by Machover and Felsenthal entitled "Taking abstention seriously" [Felsenthal and Machover, 1998].

voters are critical. Whereas the measure introduced by Coleman [1971] rescales the Banzhaf measure of voting power (see Remark 3.2.21 in [Felsenthal and Machover, 1998]).

Extending the Notion of Voting Power. Voting power measures were initially defined on binary issues and would occasionally take into consideration abstentions. However, the study of voting power measures has advanced with the study of voting models, as the power measures need to be reassessed in each new model. One way this is done is by generalising the domain of the available votes, such as including abstentions or considering an issue with many possible alternatives instead of a binary domain. Influenced by Felsenthal and Machover [2001], several authors have studied probabilistic models of voting power with abstention and a binary outcome [Freixas, 2012, Felsenthal and Machover, 1997]. Freixas and Lucchetti [2016] extended the Banzhaf index in this setting, introducing two measures of being positively critical. The first of these measures models changing a vote from being in favour of the issue to abstention, and the second from abstaining on the issue to being against it. Voting games with approvals form a subclass of voting games with several levels of approvals in the input and output of the election [Freixas and Zwicker, 2003].

Kurz [2014] extended simple voting games to allow agents to cast a ballot for any real number between 0 and 1. They considered how the power indices could be naturally extended in this setting. When considering this continuous and convex voting space, they study the index for an a priori uniform distribution of votes and distributions where the alternatives are not equally probable.

Owen [1988] extended the framework to allow the representation of the Shapley value with multilinear equations and then went on to give the Banzhaf index in this setting [Owen, 1975]. In this new framework, Owen [1981] studied these measures of voting power, extending them further to account for some agents being more likely to act together than others, moving away from the uniformity assumption.

Liquid Democracy. The closest work to ours is that of Zhang and Grossi [2021], who study a version of the Banzhaf index in liquid democracy. For a given delegation graph, their measure determines how critical an agent is in changing the outcome. However, our work differs as we focus on a priori voting power, where no prior knowledge is known about the agents, such as the delegation graph.[2]

5.2.3 The Model

Let V be a set of n voters participating in an election to decide whether some binary proposal should be accepted. Each voter has different possible actions: they may delegate their vote to another voter or vote directly, either for (1) or against (-1) the proposal. A voter who decides to vote (respectively, delegate) on the

[2]In Appendix A.2 of [Colley et al., 2023a], we present a probabilistic model which can be used to interpret the Banzhaf measure from Zhang and Grossi [2021] as a measure of I-power.

issue will be termed a delegatee (respectively, delegator). An underlying social network $G = (V, E)$ restricts the possible delegations between the agents, hence voter $i \in V$ can only delegate to a voter in their out-neighbourhood $\texttt{NB}_{out}(i) = \{j \in V \mid (i, j) \in E\}$. In our previous work, we studied two specific social networks: when G is complete and when G is bipartite, where the former corresponds to the liquid democracy setting when voters can choose any voter as their delegate and the latter corresponds to proxy voting.

Definition 5.1. Given a digraph $G = (V, E)$, a *G-delegation partition* D is a map defined on V such that $D(i) \in \texttt{NB}_{out}(i) \cup \{-1, 1\}$ for all $i \in V$. We let $\mathcal{D}$ be the set of all such partitions and D^-, D^+, and D^v be the inverse images of $\{-1\}$, $\{1\}$ and $\{v\}$ for each $v \in V$ under D.

A direct-vote partition divides the voters so that each partition cell corresponds to a possible voting option. We allow for abstentions to model situations where a delegator does not have a delegatee voting on their behalf (e.g., due to delegation cycles).

Definition 5.2. A *direct-vote partition* of a set V is a map T from V to the votes $\{-1, 0, 1\}$. We let T^-, T^0, and T^+ denote the inverse images of $\{-1\}$, $\{0\}$ and $\{1\}$ under T.

A G-delegation partition D naturally induces a direct-vote partition T_D by resolving the delegations. First, we let voters in D^- and D^+ be in T^-, and T^+, respectively. From this point, for some $\circ \in \{-, +\}$, if $v' \in D^v$ and $v \in T^\circ$, then $v' \in T^\circ$. This continues until no more voters can be added to T^+ or T^-. The remaining unassigned agents abstain and thus are in T^0. This procedure assigns agents their delegate's vote unless it leads to a cycle; in this case, their vote is recorded as an abstention.[3]

Next we define a partial ordering $\leq$ among direct-vote partitions: if T_1 and T_2 are two direct-vote partitions of V, we let: $T_1 \leq T_2 \Leftrightarrow T_1(x) \leq T_2(x)$ for all $x \in V$.

Definition 5.3. A ternary (resp. binary) voting rule is a map W from the set $\{-1, 0, 1\}^n$ (resp. $\{-1, 1\}^n$) of all direct-vote partitions (resp. all direct-vote partitions without abstention) of V to $\{-1, 1\}$ satisfying the following conditions:

1. $W(\mathbb{1}) = 1$ and $W(-\mathbb{1}) = -1$ where $\mathbb{1} = (\underbrace{1, \ldots, 1}_{\times n})$;

2. Monotonicity: $T_1 \leq T_2 \Rightarrow W(T_1) \leq W(T_2)$.[4]

[3]Note that this returns the outcomes as Unravel as described in Section 4.5 and the same as Unravel(#) from Chapter 2 for any # when ballots are restricted to classic liquid democracy ballots.

[4]No ternary voting rule satisfies monotonicity, e.g., a weighted voting rule with an additional quorum condition. However, we enforce this condition such that we must only look at the election's result to define a voter's criticality when only they change their vote.

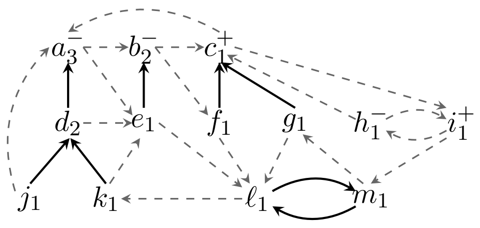


Figure 5.1: The underlying network G used in Example 5.1. While all edges give us E, the solid edges give us a valid G-delegation partition where the superscripts of $+$ or $-$ represent delegatees' direct votes for or against the issue, respectively. Each node's subscript refers to its voting weight.

Note that ternary (and binary) voting rules only use the direct-vote partition to find an outcome, i.e., only the information of which agents voted directly or indirectly for or against the proposal or abstained. Thus, these rules do not need the delegations to find an outcome.

A ternary (resp. binary) voting rule is *symmetric* if $W(T) = -W(-T)$, where $-T$ is the direct-vote partition defined by $-T(x) = -(T(x))$, for all $x \in V$. Moreover, for ease of notation, we may also use $W(T^+, T^-)$ to denote $W(T)$.

Weighted voting games. Weighted Voting Games (WVGs) express ternary voting rules compactly, with a quota $q \in (0.5, 1]$ and a map $w : V \to \mathbb{N}_{>0}$ assigning each voter a positive weight. Given a set $S \subseteq V$, we let $w(S) = \sum_{i \in S} w(i)$. In a WVG with weight function w, we let $W(T(V)) = 1$ iff $w(T^+) > q \times w(T^+ \cup T^-)$, i.e., the proposal is accepted if the sum of the voters' weights for the proposal is greater than a proportion q of the total weight of non-abstaining voters; otherwise, the proposal is rejected.

Example 5.1. Consider agents $V = \{a, b, \cdots, m\}$ connected by the underlying network G as depicted in Figure 5.1. The solid lines give a valid G-delegation partition D with $D^+ = \{c, i\}$, $D^- = \{a, b, h\}$, $D^a = \{d\}$, $D^b = \{e\}$, $D^c = \{f, g\}$, $D^d = \{j, k\}$, $D^\ell = \{m\}$, and $D^m = \{\ell\}$. This G-delegation partition induces the following direct-vote partition: $T^+ = \{c, f, g, i\}$, $T^- = \{a, b, d, e, h, j, k\}$, and $T^0 = \{\ell, m\}$. Consider the voting rule W where $q = 0.5$ and that the voters have the following voting weights: $w(a) = 3$, $w(b) = w(d) = 2$ and the remaining voters $x \in V \backslash \{a, b, d\}$ have weight $w(x) = 1$. The proposal is rejected under this G-delegation partition as $w(T^+) = 4 \leq {}^{15}/_{2} = q \cdot w(T^+ \cup T^-)$. △

We conclude this subsection with some notation. Given a set S, let $\mathcal{P}_k(S)$ denote the set of k ordered partitions of S. By ordered partitions, we mean that $(\{1\}, \{2, 3\})$ should be considered different to $(\{2, 3\}, \{1\})$. Next, given a voting rule W, a voter $i \in V$, and (S, T, U) three non-intersecting subsets of $V \setminus \{i\}$, we define:

$$\delta^W_{i,-\to+}(S, T, U) = \frac{W(S \cup U \cup \{i\}, T) - W(S, T \cup U \cup \{i\})}{2}.$$

We say a voter $i \in V$ is critical when they can affect the outcome of the vote.

Thus, for three non-intersecting subsets of $V \setminus \{i\}$. We let S (resp. T) denote the set of voters opposing (resp. supporting) the proposal through their vote of delegation. We let U be the set of voters delegating both directly and indirectly to i. Then we say that i is critical if and only if $\delta^W_{i,-\to+}(S,T,U) > 0$. We say a voter $i \in V$ is *positively* (resp, *negatively*) critical if by changing a positive (resp. negative) vote to a negative (resp. positive) one, the outcome will also change from being for to against (resp. against to for) the issue. In Example 5.1, we see that agent a is critical in this G-delegation partition, as $V\setminus\{a\}$ is partitioned as such $S = \{c, f, g, i\}$, $T = \{b, e, h\}$ and $U = \{d\}$ and thus $\delta^W_{a,-\to+}(S,T,U) = \frac{1-(-1)}{2} = 1$.

5.2.4 Modelling a priori voting power

We aim to measure a priori voting power in this setting. An agent's voting power is their probability of being able to affect the election's outcome. With a similar motivation to that behind the Penrose-Banzhaf measure, we invoke the principle of insufficient reason. There are two ways of seeing this principle.

The global uniformity assumption. If there is no information about the proposal or voters, we assume all G-delegation partitions are equally likely with probability $\Pi_{i\in V}\frac{1}{|\mathtt{NB}_{out}(i)|+2}$. In Example 5.1, as $|\mathtt{NB}_{out}(i)| = 2$ for every $i \in V$ and $|V| = 13$, this means that every G-delegation partition occurs with probability $(\frac{1}{4})^{13}$.

The individual uniformity assumption. The global uniformity assumption is similar to a model in which each voter delegates with probability $p^i_d = {}^{|\mathtt{NB}_{out}(i)|}/_{|\mathtt{NB}_{out}(i)|+2}$ and votes with probability $p^i_v = 1 - p^i_d = {}^2/_{|\mathtt{NB}_{out}(i)|+2}$. Under this assumption, delegation (resp. voting) options are chosen uniformly at random. This is consistent with the idea that we have no information about voters' personalities, interests, or the nature of the proposal. Hence, voters should be equally likely to support (probability p_y) or oppose (probability p_n) the proposal, i.e., $p_y = p_n = 1/2$. Moreover, in ignorance of any concurrence or opposition of interests between voters, we should assume that a voter's choice of delegate is also equally likely, i.e., the probability that a delegator i delegates to a voter $j \in \mathtt{NB}_{out}(i)$ is $1/\mathtt{NB}_{out}(i)$. The *individual uniformity assumption* is an extension of the global uniformity assumption in which p^i_d can be any value in $[0,1]$ dependent on $|\mathtt{NB}_{out}(i)|$, such that $p^i_d = 0$ when $\mathtt{NB}_{out}(i) = \emptyset$.

We study our model under this latter assumption of generality. We now define the LD Penrose-Banzhaf measure of a voter i for a given underlying graph G when considering that the probability of each G-delegation partition is determined by the individual uniformity assumption.

Definition 5.4 (*LD* Penrose-Banzhaf measure)**.** Given a digraph $G = (V,E)$ and a ternary voting rule W, the *LD Penrose-Banzhaf measure* of voter $i \in V$ is defined as:

$$\mathcal{M}_i^{ld}(W, G) = \sum_{D \in \mathcal{D}} \mathbb{P}(D) \frac{W(T_{D_i^+}) - W(T_{D_i^-})}{2},$$

where $\mathbb{P}(D)$ is the probability of the G-delegation partition D occurring, and D_i^+ (resp. D_i^-) is the G-delegation partition identical to D with the only possible difference being that i supports (resp. opposes) the proposal.

$\mathcal{M}_i^{ld}$ quantifies the probability to sample a delegation partition where i is able to alter the election's outcome (formally stated in the following Theorem).[5]

Theorem 5.1. *Given a digraph $G = (V, E)$, a ternary voting rule W, and a voter $i \in V$, we have that:*

$$\mathbb{P}(i \textit{ is critical}) = \mathcal{M}_i^{ld}(W, G).$$

Moreover, if $\mathtt{NB}_{out}(i) = \emptyset$ or W is symmetric, we have that:

$$\begin{aligned}\mathbb{P}(i \textit{ is positively critical}) &= \mathcal{M}_i^{ld}(W, G)/2 \\ &= \mathbb{P}(i \textit{ is negatively critical}).\end{aligned}$$

This proof relies on the fact that we are summing over the probability of each D with respect to $W(T_{D_i^+}) - W(T_{D_i^-})$, which measures when the voter i is critical. Recall that being positively critical means that changing a vote from for to against will also change the outcome in the same direction (negatively critical is defined similarly). Furthermore, this happens equally when $\mathtt{NB}_{out}(i) = \emptyset$ (the only option is to vote either for or against the issue) or when W is symmetric.

For the second part of Theorem 5.1, the condition is necessary as if W reflects unanimity, i.e., $W(T) = 1$ if and only if $T = \mathbb{1}$, then voters will be more likely to be positively critical than negatively critical.[6] Additionally, observe that the *LD* Penrose-Banzhaf measure of voting power extends the standard Penrose-Banzhaf measure (formalised in Proposition 5.1) and that its values are not normalised (i.e., summing over the agents does not yield 1). The corresponding voting power index can be defined by normalising over voters.

Proposition 5.1. *If $p_d^i = 0$ for all $i \in V$, e.g., if $E = \emptyset$, then the* LD *Penrose-Banzhaf measure of voting power is equivalent to the standard Penrose-Banzhaf voting power measure.*

Example 5.2. We return to the agents $V = \{a, \cdots, m\}$ from the previous example with the same weights; however, we are in the LD setting where the underlying

[5] Full proofs can be found in Colley et al. [2023a].

[6] If the voting rule requires total agreement to accept the proposal, then voter i will be critical if and only if all other voters agree on the proposal. Thus, the probability that i is critical while voting directly or indirectly for the proposal is higher than i being critical while voting directly or indirectly against the proposal.

Table 5.1: $\mathcal{M}_x^{ld}$ (rounded to 3 decimal places) for $p_d \in \{0, 0.5, 0.9\}$ for $v = \{a, \cdots, m\}$ from Example 5.1 when considering a *complete network*.

Agent $x \in V$	$p_d = 0$	$p_d = 0.5$	$p_d = 0.9$
a: $w = 3$	0.511	0.424	0.696
b, d: $w = 2$	0.306	0.308	0.638
$V \backslash \{a, b, d\}$: $w = 1$	0.148	0.212	0.568

network is a complete digraph, i.e., for each $i \in V$ we have that $\texttt{NB}_{out}(i) = V \backslash \{i\}$. In Table 5.1, we see the power measures of each agent where the probability of delegating varies.

When $p_d = 0$, we are in the standard weighted voting game case where all agents vote directly. Thus, the probability of voting for the issue and the probability of voting against the issue are equally likely, both happening with a probability of 0.5.

If $p_d = 0.5$, a voter i delegates to voter $j \in V \backslash \{i\}$ with probability $p_d/|V|-1 = 1/24$. Moreover, these voters vote directly with probability $(1 - p_d) = 0.5$, and thus, for the issue with probability 0.25 and against with the same probability. When this is the case, those with less voting weight have their voting power measure increase. This is due to the possibility of them gaining voting weight through receiving delegating.

Observe that when $p_d = 1$, all agents are caught in delegation cycles and thus $T^0 = V$. Hence, we study the case when $p_d = 0.9$. Notice here that the voters' voting power becomes closer together than when $p_d = 0$ or $p_d = 0.5$. This is explained by the voting weight becoming less important in determining the outcome when delegations are very likely. $\triangle$

5.2.5 Hardness of Computation

Computing the standard Penrose-Banzhaf measure in WVGs without delegations is $\#P$-complete [Prasad and Kelly, 1990]. However, it can be computed by a pseudo-polynomial algorithm that runs in polynomial time with respect to the number of voters and the maximum weight of a voter [Matsui and Matsui, 2000]. We show that the problem of computing the LD Penrose-Banzhaf measure is $\#P$-hard even when voter's weights are linearly bounded by the number of voters. Hence, a similar pseudo-polynomial algorithm is unlikely to exist in the LD setting with any underlying graph. The proof uses an enumeration trick inspired by that of Chen et al. (2010, Theorem 1).

Theorem 5.2. *Given a digraph $G = (V, E)$ and a WVG defined on V, computing the* LD*-Penrose-Banzhaf power measure of a voter is $\#P$-hard even when voter's weights are linearly bounded by the number of voters.*

Proof. To prove this, we will give a Turing reduction[7] from the #P-complete problem of counting simple paths in a graph [Valiant, 1979]. The problem takes a

[7]There are many types of reductions for counting problems. There are also *parsimonious* reductions and *polynomial-time counting* reductions. Using a Turing reduction allows for a polynomial number of calls to a subroutine of our problem to solve the #P-complete problem.

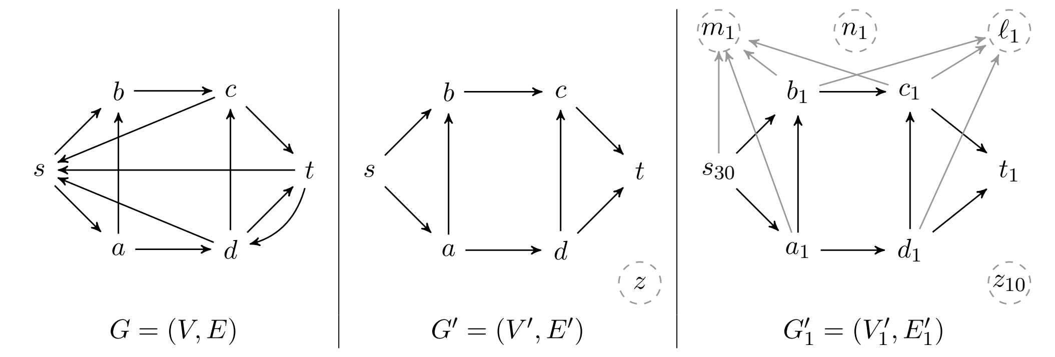


Figure 5.2: This figure gives an example of the reduction given in the proof of Theorem 5.2. From left to right, we see the different stages of the reduction. The left image shows an initial graph G for which we want to use the problem from Valiant [1979] to count the number of paths from s to t. The middle graph $G' = (V', E')$ shows the pre-processing step where the incoming edges of s have been removed, the outgoing edges of t have been removed, and the first dummy agent z has been added. In these figures, dummy agents are indicated by a grey dashed circle around them. Note that $d_{max} = 2$ in G' and $|V| = 6$. The image on the right of the figure shows one of the 6 elections created for this instance when $k = 1$ (the 6 instances come from $k = 1, \cdots, 6$). We see that three additional dummy agents are required $D_1 = \{\ell, m, n\}$ such that all agents in $V \backslash \{t\}$ can have $d_{max} + k = 3$ outgoing edges. The subscripts for the nodes in G'_1 denote the agents' voting that $|V'_1| = 10$.

directed graph $G = (V, E)$ and nodes $s, t \in V$ as input and returns the number of simple paths from s to t in G.

Let $G = (V, E)$ be a directed graph and $s, t \in V$ two connected vertices of this graph. For our problems to align, we will make some alterations to G as a pre-processing step, yet retaining the correct outcome of the original problem. Thus, we construct a second graph $G' = (V', E')$ in which we remove every incoming edge of s in E and every outgoing edge of t to get E'. We add a dummy node z with no outgoing or incoming edges, giving $V' = V \cup \{z\}$. See Figure 5.2 for an example of how the reduction works on a specific instance of G.

Let d_{max} be the highest out-degree of any vertex in G'. For $k \geq 0$, we let $G'_k = (V'_k, E'_k)$ be the graph G' with additional dummy agents D_k and additional outgoing edges to D_k in order that every $x \in V' \backslash \{z, t\}$ has out-degree $d_{max} + k$. However, z and t have out-degree 0. Note that we only add a polynomial number of vertices and edges by doing this.

Now, we will run $|V| + 1$ elections, one for each graph G'_k with $k = 1, \ldots, |V|$. For election k, we set $w(s) = 3|V'_k|$, $w(z) = |V'_k|$ and $w(x) = 1$ for $x \in V'_k \backslash \{s, z\})$. Observe that when node s is not in a cycle, they are always critical as $w(s) > \sum_{i \in V'_k \backslash \{s\}} w(i) = 2|V'_k| - 2$ and the election's outcome is determined by the vote of s (the total voting weight in the network is $5|V'_k| - 2$). However, if s is in a cycle, then the outcome of the election depends only on z's vote as $|V'_k| = w(z) > \sum_{i \in V'_k \backslash \{s,z\}} w(i) = |V'_k| - 2$. Moreover, observe that z cannot be in a cycle as they have no outgoing edges. Thus, when s is not in a cycle, a vertex $x \notin \{z, s\}$ is critical for their given delegation partition if and only if x is part of s's delegation path. In particular, this is true for t, which, with the additional point that t cannot be in a cycle (t has no outgoing edges), means that t is critical if and only if there is a path from s to t via the delegations.

Next, we let P be a delegation path from s to t in G'_k of length l. Each vertex on this path has $d_{max} + k$ outgoing edges. Thus, the probability of this path occurring under the uniformity assumption is equal to $\pi_k^l = \left(p_d^{d_{max}+k} \frac{1}{d_{max}+k}\right)^l$. Let $\mathcal{P}_l$ be the set of simple paths of length l between s and t in G' (also in G). In Figure 5.2 we see that $\mathcal{P}_2 = \emptyset$ and $|\mathcal{P}_3| = 2$. Note that the maximum length of a simple path in the graph is n, thus $l \in [1, n]$. Since t is critical if and only if there is a path from s to t on the delegation graph, then:

$$\mathbb{P}(t \text{ is critical in } G'_k) = \sum_{l=1}^{n} |\mathcal{P}_l| \pi_k^l.$$

We obtain a set of $|V|$ linear equations (one for each value of $k = 1, \cdots, |V|$), each equation is built for the variables $|\mathcal{P}_l|$ with coefficient π_k^l (one variable for each $l \in [1, n]$). Hence, we have $|V|$ variables as $l = 1, \cdots, |V|$ and note that all coefficients differ.

With these equations, we can make a Vandermonde matrix M from the coefficients of the equations. Hence, M is an $|V| \times |V|$ matrix where $M_{ij} = \pi_i^j$ for $i, j = 1, \cdots, n$. We let the vector $X^{\mathcal{P}} = (|\mathcal{P}_1|, \cdots, |\mathcal{P}_{|V|}|)$ be the variable vector and Y^c be a vector with n entries such that $Y_k^c = \mathbb{P}(t \text{ is critical in } G'_k)$. Hence, we get the following equation:

$$Y^c = MX^{\mathcal{P}}.$$

Note that M is a square Vandermonde matrix with different entries, so it can be easily inverted [Macon and Spitzbart, 1958]. Hence, given M and Y^c, $X^{\mathcal{P}}$ can be computed in polynomial time. As a result, we obtain the values of $|\mathcal{P}_1|, \cdots, |\mathcal{P}_{|V|}|$ and can thus calculate the number of paths from s to t by summing these values. Finally, given that solving the simple path counting problem is #P-hard, so is ours. □

Thus, we have shown that computing our liquid democracy Penrose-Banzhaf measure is hard in general. In Colley et al. [2023c], we showed that this can be reduced to being computable in pseudo-polynomial time when restricted to a complete underlying graph or a complete bipartite graph (representing proxy voting when all delegators can pick any delegatee).

5.3 Modelling Long-Term Delegation in Liquid Democracy

This section analyses a different usage of the classical model of liquid democracy. Models of liquid democracy often consider a one-off election, whether it be on a single issue (e.g., the model of Smart Voting described in Chapter 2) or multiple issues simultaneously (e.g., the multi-issue model described in Chapter 3). In this work, we now consider a more practical use, inspired by the voting platform LiquidFeedback [Behrens et al., 2014], where each user sets up a delegation to be enacted only in the case when they cannot vote on an issue directly. Hence, this section is interested in *long-term delegations* in a classical liquid democracy setting.

Long-term delegations are suggested in the LiquidFeedback platform to ensure every voter has a ballot on every issue. LiquidFeedback allows for a delegation to be set up to last over many issues, where each issue has its own single-issue election happening asynchronously. Therefore, a delegation is used, for example, when a voter is relying on their delegate as an expert to vote on their behalf or as a means of dividing the labour of becoming sufficiently well-informed on every issue when there are too many among many voters.

In the LiquidFeedback software [Behrens et al., 2014], multiple delegations are given, each with instructions on when it should be used. These instructions refer to the most accurate delegation given for an issue. For example, a delegation given on a specific issue should be used if the user did not vote directly. However, if no such specific delegation was used, then a delegation given relating to the broader genre of the issue should be used in this case (i.e., if no issue-specific delegation was given on the issue of the new bus routes, then a delegation given on the broader genre of public transport should be used). A system specifying when different delegations should be used is a valuable functionality in practice. It allows for a satisfactory delegate to be used in many scenarios, especially when voters cannot be perfectly informed to vote directly on all issues nor to remember to delegate perfectly for every issue. Thus, these levels of delegations are safety nets to ensure adequate delegates are used each time. This section only considers a *single* long-term delegation to be set up, thus simplifying the model used in LiquidFeedback.

One behaviour observed on the platform was that the users were intentionally creating cycles [Behrens et al., 2022], which was unexpected, given that the theoretical study of liquid democracy has viewed cycles as something to be avoided or minimised (e.g., Chapter 2 and Section 3.6.1). Thus, this work aims to expand the knowledge of liquid democracy to try and account for this phenomenon by remodelling liquid democracy in terms of *long-term delegations*.

We provide a probabilistic model in which every voter may or may not vote directly with some probability, and this probability can vary among voters. This probability reflects the intuition that there are too many issues for a voter to learn about them all adequately or even be aware of them all. We then model the utility gained by a voter from a single election as the distance or the dissimilarity between them and the voter who voted on their behalf, known as their *ultimate delegate*. Thus, a voter gains a maximal payoff when they vote themself and a lower payoff when a dissimilar voter votes on their behalf. This payoff can even be negative when the dissimilarity between a voter and their ultimate delegate is too large. One way in which to model this dissimilarity between voters could be quantifying the distance between them on the *left-right* political spectrum. Thus, we are interested in whether a stable collection of delegations exists so that no agent's expected utility can be improved by changing their delegation. Moreover, we want to see if these stable states contain delegation cycles.

5.3.1 Contribution

This section details preliminary ongoing work with Markus Brill (Warwick University), Anne-Marie George (University of Olso) and Ulrike Schmidt-Kraepelin (Universidad de Chile), which stems from discussions with Andreas Nitsche from the Interaktive Demokratie, a co-creator of LiquidFeedback. Our preliminary work consists of modelling such a framework and providing a notion of a Nash equilibrium in this model where no voters would change their delegation to increase their expected utility. Furthermore, we provide a best-response dynamic that finds Nash equilibria from random profiles of delegations. Finally, we give some preliminary results found through simulations, highlighting promising future research directions.

5.3.2 Related Work

We highlight some work closely related to the model we will introduce in the next section, the closest of which is that of Escoffier et al. [2019, 2020]. They propose a model where voters have an ordinal preference over the direct voters from a model relating to our notion of distance. They show that equilibria are not always guaranteed to exist in their model in general, yet they do for many different preference structures over the delegators' possible ultimate delegates. There are two main differences between our models. First, we are interested in long-term delegations, whereas they are interested in a one-off election. Thus, we look for stability in the expected utility over all possible realisations of a delegation graph rather than the utility gained from a single election. The second is that we have different notions about which delegates are deemed suitable by the agents. We have a distance-based approach, whereas they use ordinal preferences.

Another similar model to ours is that of Anshelevich et al. [2021], who study a game-theoretical model of proxy voting where the agents are also positioned on a line. However, they are trying to find a subset of proxies that can best represent the population as a whole.

We are also interested in the model from Bloembergen et al. [2019], they also studied liquid democracy from a game-theoretic point of view to determine when it is beneficial for a voter to delegate. They also consider a probabilistic model where the utility given is the voter's ability to communicate their *type* (either being of type 0 or 1). The probabilities in this model measure the accuracy of the voter communicating their type correctly when voting directly. They study equilibria in delegation games where the accuracy of choosing a delegate is being weighed against voting directly, with their accuracy minus the effort they expend in voting directly. Many similar game-theoretic models define utility in terms of the group's accuracy in finding a ground truth. These models differ from ours in that they are not interested in the individual's utility; see Section 1.1 for more details of these models.

While the preceding models all share properties with the model we are about to introduce, there are many clear distinctions — crucially, our treatment uses probabilities to model the long-term effects of single delegation.

5.3.3 The Model

Consider a set of agents (or voters) $\mathcal{N} = \{1, \cdots, n\}$ who are participating in a liquid democracy platform where they can choose to vote on any issue or use their long-term delegation. In our probabilistic model, we consider the long-term expected utility of delegations. Thus, each voter has a probability of voting directly $p_i \in [0,1]$ for every $i \in \mathcal{N}$. These probabilities are collected into a vector $\boldsymbol{p} = (p_1, \cdots, p_n)$. The utility that each agent gains from a single issue depends on how *far away* their *ultimate delegate* is, given that all voters have decided if they will vote directly or delegate on this issue. Distance between voters can be defined in many ways, but in this section, we focus on the distance between points on a line. Thus, every agent $i \in \mathcal{N}$ has a position $x_i \in [0,1]$. We collect these positions into the vector $\boldsymbol{x} = (x_1, \cdots, x_n)$. We let $dist(i,j) = |x_i - x_j|$ denote the distance between two voters i and j's positions on the line.[8]

We now introduce the long-term delegations to the model. Every agent $i \in \mathcal{N}$ chooses a delegate $d_i \in \mathcal{N}$, who they will delegate to with a probability of $1 - p_i$. Note that when $d_i = i$, i has chosen not to select a long-term delegate and will only have a vote recorded on issues when they vote directly. The profile of delegations is denoted by $\boldsymbol{d} = (d_1, \cdots, d_n)$. From the profile of delegations, a delegation graph $G(\boldsymbol{d}) = (\mathcal{N}, E)$ can be given where for every $i \in \mathcal{N}$, there exists a single edge $(i, d_i) \in E$. From a delegation graph $G(\boldsymbol{d})$, we can obtain a realised version of the delegation graph $G(\boldsymbol{d})_r$, which represents a profile of votes on a specific issue where some agents have voted directly, thus $G(\boldsymbol{d})_r$ removes the outgoing edges from any agent who is voting directly on this specific issue.

Example 5.3. Consider six agents $\mathcal{N} = \{A, B, C, D, E, F\}$ whose opinions can be placed on a line such that $\boldsymbol{x} = (0.2, 0.25, 0.4, 0.4, 0.6, 0.8)$ as depicted in Figure 5.3. Each agent will vote directly on issues with different probabilities, given by vector $\boldsymbol{p} = (0.5, 0.5, 0.9, 0.3, 0.5, 0.3)$, thus agent C votes directly with probability 0.9. Consider the profile of delegations $\boldsymbol{d} = (A, D, E, B, F, E)$ which induces a delegation

[8]One extension of the work we are interested in is modelling the agent's positions in a unit square. Hence, $x_i \in [0,1]^2$. Here, the notion of distance would be updated. For example, if $x_i = (a,b)$ and $x_j = (c,d)$, then we define $dist(i,j) = \sqrt{(a-c)^2 + (b-d)^2}$.

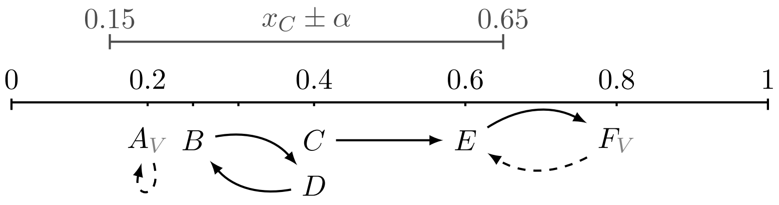


Figure 5.3: A diagram showing the positions $\boldsymbol{x}$ and delegations in $\boldsymbol{d}$ from the Example 5.3. The profile of delegations $\boldsymbol{d}$ is seen in the figure by considering both the solid and dashed lines. The realisation of the delegation graph $G(\boldsymbol{d})_r$ is seen by only considering the solid edges, and the direct voters in this model are A and F (denoted by the subscript V). Finally, we see that for agent C, their limit of what they consider an acceptable ultimate delegate given by $x_C \pm \alpha$.

graph $G(\boldsymbol{d})$, depicted in Figure 5.3 by considering both the dashed and solid lines. However, let us consider a realisation of this delegation graph where the only voters to vote directly are A and F (in Figure 5.3 this is indicated by them having the subscript V). This gives the realisation of the graph $G(\boldsymbol{d})_r$, seen in the figure when only considering the solid edges. This realisation of the graph happens with probability $p_A \times p_F \times \Pi_{i \in \mathcal{N} \setminus \{A,F\}}(1-p_i) = 0.002625$. As A and F are the only direct voters in this realisation of the graph, they are the only possible ultimate delegates. Hence, C, E and F have F vote on their behalf, A just votes for themselves. The votes of voters B and D are not counted in this realisation as they are caught in a delegation cycle. △

We see from Example 5.3 that we next need to be able to define the payoff a voter receives from a given realised delegation graph. We let $\alpha \in \mathbb{R}$ denote a voter's maximum payoff. Intuitively, this maximum payoff can only be achieved when their ultimate delegate has the same position as them. Hence, if a delegator $i \in \mathcal{N}$ has the ultimate delegate of $j \in \mathcal{N}$, they receive the utility of α only when they share the same position $x_i = x_j$. In particular, this holds if $i \in \mathcal{N}$ votes directly, as they are their own ultimate delegate. In general, we define the utility gained by an agent in a realised delegation graph to be the maximum payoff minus the distance between the voter and their ultimate delegate, given by the following formula:

$$u_i(G_r) = \begin{cases} \alpha - dist(i,j) & \text{if there exists an ultimate delegate } j \in \mathcal{N} \text{ for } i \text{ in } G_r, \\ 0 & \text{otherwise (if } i \text{ is in a cycle).} \end{cases}$$

where $dist(i,j)$ is the distance between two voters' positions described above.

In Example 5.3, we see that the utility from the realised graph for agent A is $\alpha = 0.25$ as they vote on their behalf. Agent E has F as their ultimate delegate, as $dist(E,F) = 0.2$, this means that $u_E(G(\boldsymbol{d})_r) = \alpha - dist(E,F) = 0.05$. Agent C has a negative payoff from this realisation of the graph where F is their ultimate delegate as $u_C(G(\boldsymbol{d})_r) = 0.25 - 0.4 = -0.15$. Therefore, C would have rather not delegated in this case and received utility 0, as B and D did.

We are interested in the expected utility of this model rather than the utility of a single realised graph. Therefore, we use this utility to define the voters' expected utility for a given delegation graph $G(\boldsymbol{d})$. To do so, we find the longest outgoing simple path from an agent $i \in \mathcal{N}$ in $G(\boldsymbol{d})$, which we will denote the length of as k. Note that every agent on this path could be i's ultimate delegate in its different realisations. We can relabel the agents in this path such that $v_1 = i$ and v_k is the final agent in this path. The expected utility of i in G given by:

$$\mathbb{E}(i, G) = \sum_{\ell=1}^{k} (\alpha - dist(v_1, v_\ell)) p_{v_\ell} \prod_{m=1}^{\ell-1} (1 - p_m)$$

This sums for each outgoing simple path from $i = v_1$ to v_ℓ (thus $\ell \leq k$), the utility i would obtain from v_ℓ being their ultimate delegate, given by $\alpha - dist(v_1, v_\ell)$, multiplied by the likelihood of that happening, given by $p_{v_\ell} \prod_{m=1}^{\ell-1} (1 - p_m)$. This product corresponds to the probability that v_ℓ is i's ultimate delegate. This can only happen if v_ℓ is the first delegate in this path to vote directly. Thus, all voters prior to v_ℓ in the path do not vote, that is $(1 - p_m)$ for each $m \in [1, \ell - 1]$, and v_ℓ does vote with probability p_ℓ.

Example 5.4. Returning to Example 5.3, we see that the longest outgoing path from A in $G(\boldsymbol{d})$ is just A. Therefore, A's expected utility from $\boldsymbol{d}$ is $p_a \times \alpha = 0.5 \times 0.25 = 0.125$. We now consider the expected payoff of C. We see that the longest outgoing path from C goes to F via E. Thus, we consider the three sub-paths: C, C to E and C to F. We will now calculate the expected utility contributed by each of the possible paths from C in $\boldsymbol{d}$.

Path$= C$	This path represents C voting directly with probability 0.9 receiving a payoff of $\alpha = 0.25$, contributing 0.225 to the expected utility.
Path$= CE$	This path represents that C did not vote and E did, which happens with probability $(1-0.9)\times 0.5 = 0.05$. As $dist(C, E) = 0.2$, if this happened C would receive a utility of 0.05. Hence, this contributes to the expected utility of 0.0025.
Path$= CEF$	This represents C and E not voting and F voting. This realisation occurs with probability $(1-0.9)\times(1-0.5)\times 0.3 = 0.015$. The payoff C would receive from F being their ultimate delegate is -0.15 as $dist(C, F) = 0.4$. Hence, the expected utility contributed by this realisation is -0.00225.

Thus, in total, the expected utility of this profile of delegation for C is $\mathbb{E}(C, G(\boldsymbol{d})) = 0.22525$. △

5.3.4 Best-Responses and Equilibrium

This section defines the solution concepts for our model. We first define a best response. Informally, this means that for an agent $i \in \mathcal{N}$ and profile of delegations $\boldsymbol{d}$, an alternative delegation d'_i is a best response to $\boldsymbol{d}$ when replacing their delegation in $\boldsymbol{d}$ to d'_i results in the highest expected utility for agent i.

Definition 5.5. For agent $i \in \mathcal{N}$, d'_i is a *best response* to $\boldsymbol{d}$ if and only if $\mathbb{E}(i, G(\boldsymbol{d}_{-i}, d'_i))$ is maximal and $\mathbb{E}(i, G(\boldsymbol{d}_{-i}, d'_i)) > \mathbb{E}(i, G(\boldsymbol{d}))$, for some $\boldsymbol{x}, \boldsymbol{p}$ and $\boldsymbol{d}$.

Example 5.5. We return to the scenario described in Example 5.3 and the profile of delegations $\boldsymbol{d} = [A, D, E, B, F, E]$. Agent C has a best response to $\boldsymbol{d}$, as their expected utility increases when they change their delegation from E to D (as seen in Table 5.2). △

d'_C	A	B	C	D	E	F
$\mathbb{E}(C, G(\boldsymbol{d}_{-C}, d'_C))$	0.2275	0.23375	0.2250	**0.2360**	0.22525	0.22225

Table 5.2: Agent C's expected utility for each of their possible delegations ($d'_C \in \mathcal{N}$) in response to the remaining agents' delegations in $\boldsymbol{d}_{-C}$. Their expected utility is maximal when delegating to D and is therefore, their best response to $\boldsymbol{d}$.

Using the notion of a best response, we can now define a Nash equilibrium in this model. This refers to no agent having a higher expected utility when considering another delegation when all others are fixed (i.e., no unilateral deviation is beneficial for that agent).

Definition 5.6. A profile of delegations $\boldsymbol{d}$ is a *Nash equilibrium* (NE) given $\boldsymbol{x}$ and $\boldsymbol{p}$ if and only if no $i \in \mathcal{N}$ has a best response d'_i to $\boldsymbol{d}$.

Note that for a given pair $\boldsymbol{x}$ and $\boldsymbol{p}$, there can be multiple equilibria, yet we do not know whether an equilibrium is guaranteed for any $\boldsymbol{x}$ and $\boldsymbol{p}$.

Example 5.6. Returning to Example 5.3, it is clear that $\boldsymbol{d}$ is not a Nash Equilibrium as in Example 5.5 we saw that C has a best response to $\boldsymbol{d}$, which is changing their delegation to D instead of E.

agent $i \setminus d'_i$	A	B	C	D	E	F
A	0.12	**0.175**	0.14825	0.14825	0.099125	0.0725
B	**0.175**	0.125	0.1715	0.1715	0.12325	0.08100
C	0.23	0.23125	0.225	**0.2325**	0.23125	0.2205
D	0.1100	0.1187	**0.2325**	0.075	0.0925	0.0435
E	0.0750	0.08125	**0.14825**	**0.14825**	0.125	0.1325
F	-0.1	-0.09125	-0.02265	-0.02265	0.043675	**0.075**

Table 5.3: The expected utilities of the agents from Example 5.3 with the same positions and probabilities yet considering the profile of delegations $\boldsymbol{d}' = [B, A, D, C, D, F]$. Each row represents that agent's expected utility when considering their possible delegations. The highest expected utility for each agent is in bold. Notice that the delegations of $\boldsymbol{d}'$ are highlighted; thus, there is no best response.

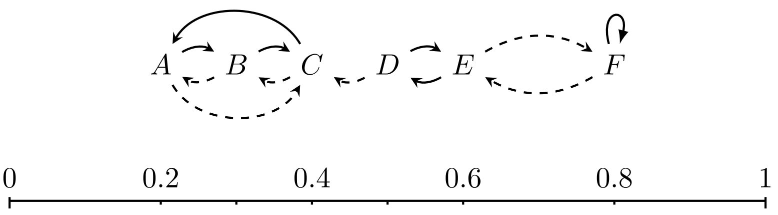


Figure 5.4: The delegation graphs $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ from Example 5.7 represented by solid and dashed lines, respectively.

Instead, consider the profile of delegations $\boldsymbol{d}' = [B, A, D, C, D, F]$. This is a NE as for each of the agents, choosing a different delegation will not increase their utility, as seen in Table 5.3. Note that although E has two equally good delegations, delegating to either C or to D gives the same expected utility. However, $\boldsymbol{d}'$ is still a NE as E cannot increase their expected utility by deviating. $\triangle$

We currently do not have proof guaranteeing the existence of NEs. Thus, we support this conjecture via simulations in Section 5.3.5.

Observation. For a given $\boldsymbol{x}$, $\boldsymbol{p}$, and α, there can be many profiles of delegations $\boldsymbol{d}$ that are NEs. Moreover, for two profiles of delegations $\boldsymbol{d}$ and $\boldsymbol{d}'$ that are NEs, their respective delegation graphs $G(\boldsymbol{d})$ and $G(\boldsymbol{d}')$ can have node-wise different connected components, as shown in the following example.

Example 5.7. Consider $\alpha = 0.25$, $\mathcal{N} = \{A, B, C, D, E, F\}$ with positions $\boldsymbol{x} = [0.2, 0.3, 0.4, 0.5, 0.6, 0.8]$ and probabilities $\boldsymbol{p} = [0.8, 0.3, 0.2, 0.3, 0.1, 0.3]$. Here we have two NEs, whose delegation graphs have different connected components, $\boldsymbol{d}_1 = [B, C, A, E, D, F]$ and $\boldsymbol{d}_2 = [C, A, B, C, F, E]$ (as shown in Figure 5.4). It is clear that in $\boldsymbol{d}_1$ that there are three connected components (A, B, C), (D, E) and (F), whereas $\boldsymbol{d}_2$ has only two connected components (A, B, C, D) and (E, F). Hence, these equilibria are substantially different, i.e., not just the same nodes in a cycle with a different ordering of agents. We note that here: $\sum_{i \in \mathcal{N}} \mathbb{E}(i, G(\boldsymbol{d}_1)) = 0.68652$ and $\sum_{i \in \mathcal{N}} \mathbb{E}(i, G(\boldsymbol{d}_2)) = 0.6888$, again supporting that these two NEs are substantially different. $\triangle$

5.3.5 Best Response Protocols Returning Nash Equilibria

With a definition for Nash equilibria in our model, we now want a procedure which finds them. We consider a best response dynamic (using Definition 5.5) that iteratively asks agents if they have a best response to the current profile of delegations. We detail the best response dynamic in Algorithm 8.

The protocol initially starts with a random profile of delegations, $\boldsymbol{d} \in \mathcal{N}^n$. It then updates the agents' delegations one at a time (in the same order as in $\boldsymbol{x}$) with their new best response when one exists (ties are broken arbitrarily). We will refer to each time the protocol checks if an agent has a best response as a *round.* The algorithm will only terminate when there have been n rounds without a BR (when

Algorithm 8 BR Protocol

```
1: Input: $\boldsymbol{p}$, $\boldsymbol{x}$, $n$, and $\alpha$
2: $\boldsymbol{d} \in [1, n]^n$    ▷ randomly initialise a dummy vector of delegations
3: $it = 0$    ▷ initialise counter of the number of rounds without a BR
4: while $it \leq n$ do    ▷ while there has been a BR in one of the last $n$ rounds
5:     $\boldsymbol{d}^1 = \boldsymbol{d}$
6:     for each $i \in \mathcal{N}$ do
7:         $it = it + 1$
8:         $exp = (0)^n$    ▷ initialise the vector for expected utilities
9:         for each $j \in \mathcal{N}$ do    ▷ for each possible delegation
10:            $exp(j) = \mathbb{E}(i, (G_{-i}(\boldsymbol{d}^1), j))$    ▷ add the delegation's expected utility
11:        if $\max_{k \in [1,n]}(exp) > \mathbb{E}(i, G(\boldsymbol{d}^1))$ then    ▷ if there is a BR
12:            $\boldsymbol{d}(i) = \arg\max_{j \in \mathcal{N}} exp(j)$    ▷ add their BR (random tie-breaking)
13:            $it = 0$    ▷ reset the counter to 0
14:        else if $it \geq n$ then    ▷ else if there have been $n$ rounds without a BR
15:            return $\boldsymbol{d}$    ▷ return the NE
```

$it \geq n$). Therefore, the resulting delegations returned will be a NE as no agent has a BR to the profile of delegations. Each time a BR has been updated (in line 11), the counter it is set to 0 (at line 13).

Example 5.8. We will perform the best response protocol on our running example, first given in Example 5.3. We assume that the randomly initiated set of delegations is $\boldsymbol{d} = [A, D, E, B, F, E]$. Following the BR protocol, we start with agent A and check if there is a BR. A's BR is delegating to agent B. We then check for agent B, who has a BR to delegate to A. We then inspect C and see that their BR is to delegate to D. This continues until we arrive at a NE $\boldsymbol{d}'' = [B, A, D, C, C, F]$ after 6 iterations (we can confirm this is a NE with Table 5.3). In these first 6 steps, the counter it is set to 0 each time an agent updates their delegations with a BR. The protocol will then continue for another 6 iterations with no more updates (as we have reached a NE). The protocol terminates when $it = 6$. Moreover, we note that for our example, there are only two NEs, $\boldsymbol{d}'$ and $\boldsymbol{d}''$. △

Finding NEs manually via this protocol is difficult by hand due to the number of calculations required to check which of the delegations is a BR, if one exists. Thus, we implemented the protocol given in Algorithm 8 in Python 3 [Van Rossum and Drake, 2009] and ran simulations to guide our research. The primary goal of these simulations was to give us some intuition of whether there is a guarantee of a NE existing for any $\boldsymbol{x}$, $\boldsymbol{p}$ and α.

Experiment 1 We ran the BR protocol on 20,000 instances of our problem where the parameters n, $\boldsymbol{x}$, $\boldsymbol{p}$, α were chosen at random as such:

- n chosen randomly from $[1, 100]$;

- $\alpha \in [0, \frac{2}{3}]$ rounded to 2 decimal places;[9]
- $x \in [0, 1]$ rounded to 2 decimal places;
- $p \in [0, 1]$ rounded to 2 decimal places.

For each of these instances, we found a NE. We also wanted to test how well the protocol found a NE. We did this by measuring how many rounds were needed. However, as there are different numbers of agents in each of the instances, we instead compared the total number of rounds carried out by the protocol divided by n, $^r/_n$. This can be thought of as the number of times the protocol checked if an agent had a BR. Over the $20,000$ instances, the protocol found a NE after checking if each agent had a BR $^r/_n = 2.5$ on average. Furthermore, for different ranges of n, the average remains similar (each $^r/_n \in [2.5, 3.3]$ when restricting to different ranges of n). The maximum value of $^r/_n$ from our $20,000$ instances was 9.

5.3.6 Using Potential Functions to Find Nash Equilibria

In the study of this model, the last thing we explore in this chapter is the use of potential functions to find Nash Equilibria. Our delegation game is finite as we have a finite set of actions (the possible delegations) and clear utilities from every profile of delegations (the expected utilities). Therefore, we are guaranteed that a mixed strategy Nash equilibria will exist [Nash, 1951]. A mixed strategy involves multiple actions, each played with some probability. For example, a mixed strategy could be an agent choosing to delegate to their first delegate with probability p and to a second with probability $(1 - p)$.

However, the question remains whether a pure equilibrium is guaranteed to exist in our model. Experiment 1 supports this claim, as the BR protocol found an equilibrium for each random instance. However, we still need proof of this being the case.

One proof method of interest is turning our delegation game into a potential game [Monderer and Shapley, 1996]. Potential games are characterised by a global *potential function*, a global measure that aligns with the agents' incentives. Moreover, Monderer and Shapley [1996] showed that a game with an ordinal potential function is guaranteed to have a Nash equilibrium. Hence, if we can find a function that increases and decreases in the same direction as the agents' utility in every unilateral deviation, our model is guaranteed a Nash equilibrium.

A natural place to start looking for a potential function for our delegation game is by looking at the total expected utility of all agents from a profile of delegations. Hence, we start with the potential function:

$$\boldsymbol{pf}(\boldsymbol{d}) = \sum_{i \in \mathcal{N}} \mathbb{E}(i, G(\boldsymbol{d})).$$

[9]This range of values for α was chosen due to it being easier to find NEs when α is high. Intuitively, when α is high, there will be more positive neighbours and therefore the expected payoff will be higher.

However, the next example shows that this is not a potential function for our delegation game.

Example 5.9. Consider agents $\mathcal{N} = \{A, B, C\}$ with positions $\boldsymbol{x} = [0.2, 0.5, 0.7]$, probabilities $\boldsymbol{p} = [0.4, 0.8, 0.1]$ and $\alpha = 0.35$. Consider the current profile of delegations $\boldsymbol{d} = (B, C, B)$ for which the potential function gives $\boldsymbol{pf}(\boldsymbol{d}) = 0.5882$. We now consider the unilateral deviation of B. B's best response is to delegate to A, strictly increasing their utility from 0.283 to 0.284. However, in this new profile of delegations $\boldsymbol{d}' = (B, A, B)$, we see that the potential function lowers to $\boldsymbol{pf}(\boldsymbol{d}') = 0.5802$. Therefore, the potential function has decreased by the agent B increasing their utility. △

The consequence of Example 5.9 is that the function $\boldsymbol{pf}$ is not a potential function for our model. However, an interesting area for future work is trying to see if there are potential functions for our model to show the guaranteed existence of pure Nash equilibria in our delegation game.

5.4 Conclusion and Future Work

This chapter has outlined two models for which classical liquid democracy has been analysed. The first is the introduction of the Penrose-Banzhaf index in liquid democracy as an a priori voting power measure. This allows us to measure the impact of the election's structure on how the voters can impact the vote's outcome. The second model introduced a more realistic use of delegations on liquid democracy platforms, such as LiquidFeedback. In particular, its purpose was to make sense of the observed behaviour on the platforms, such as users intentionally creating delegation cycles.

5.4.1 Discussion on A Priori Voting Power in Liquid Democracy

Section 5.2 introduced the well-studied notion of a priori voting power and extended it to liquid democracy. A priori voting power refers to a voter's ability to change the outcome without knowing the issue or the connections between the voters. This aims to measure each agent's power obtained from the structural aspects of the model. In liquid democracy, this could be the voting weight of the voter or their position in the underlying social network, which determines their possible delegations.

We gave the generalised version of the Penrose-Banzhaf index in liquid democracy, where an agent's delegations are restricted to their neighbours in an underlying social network connecting the voters. The main result of this section was giving the #P-hardness result showing computing the index for an agent on an arbitrary underlying social network.

We have also studied subcases of our model where the computation of the measures becomes pseudo-polynomial rather than #P-hard. These cases arise where there is some structure on the delegations, for example, when the underlying network is complete (and there are no restrictions on the delegations) or when we have a complete bipartite graph which represents proxy voting (see [Colley et al., 2023c,a] for full details). We also conducted various simulations on these subcases. For example, we showed the relationship in the liquid democracy setting with a complete underlying graph between the probability of an agent being critical and the probability of delegating. The association observed by the simulations was that the probability of being critical became more similar as the probability of delegating increased.

There are many potential areas for future work stemming from this line of research. One area of particular interest would be in conducting more simulations. We have started inspecting the effect of the underlying graph on the power measures, showing a strong correlation between the voters' criticality and their in-degree in the network, similarly between the network's structure and the voters' criticality. We hope that further research in that direction will work towards the understanding of precisely when agents gain more voting power solely due to network structure. Another natural extension is to redefine these measures in other models of delegative democracy, such as smart voting (Chapter 2) or a model with a non-binary domain that uses delegations.

5.4.2 Discussion on Long-Term Delegations

Section 5.3 highlighted a new way of modelling liquid democracy, rephrasing the use of delegations in terms of their use on digital democracy platforms. This has been done by moving away from the idea that delegations are given for a single issue; in our model they are instead set up to be used for many issues. Hence, we created a probabilistic model where voters have a probability of how likely they are to vote directly on an issue, with their delegation being used when they do not vote. From this, we created a delegation game where the strategy of each agent is to find a delegation which maximises their expected utility based on how close they are to their ultimate delegate. In this model, we gave the notion of a best response and Nash equilibria. Our main open question is whether a Nash equilibrium in guaranteed to exist for every instance. However, we do know that they are not necessarily unique (see Example 5.7). We believe that a promising direction for proving the existence of Nash equilibria is via potential functions and potential games.

There are many possible extensions of the current model and we hope to explore them in our future work. The first one is considering more complex notions of the agents' positions, such as being placed in a m-dimensional space. Within these extensions, there may be cases where Nash equilibria do not exist. The second extension is letting α be more general – in particular, for every agent $i \in \mathcal{N}$ to have their own α_i, which dictates how far away an acceptable ultimate delegate is for them.

Chapter 6

Conclusion

This thesis has examined how delegations can be made expressive and rational. We emphasise that this line of theoretical research has direct application to the realm of digital democracy platforms. Moreover, adapting these platforms to incorporate delegations improves the users' experience, making the process more accessible and allowing users to convey their opinions more accurately. In Section 6.1, we will address how each chapter of this thesis has worked towards this aim. Section 6.2 will address some of my other research directions that pertain to making the voting process on digital democracy platforms more expressive and rational.

6.1 Summary of Contributions and Discussion

We first looked at making delegations more expressive by allowing for ranked multi-agent delegations in Chapter 2. Our smart voting model allowed agents to give more complex delegations to express how their votes should be determined. Valid ballots in this model permitted delegations to use the votes of multiple trusted agents. Hence, a delegation could let the majority opinion of a group determine their vote instead of being forced to choose a single agent. The second aspect allowed ballots to contain ranked delegations, where agents can specify how their vote should be determined in case of a delegation cycle. From proposing this new form of delegations, we then determined how these complex delegations should be resolved. We introduced six unravelling procedures to do so. The first two optimised the level of ranked delegations used when resolving delegations and were both intractable problems. However, we showed that we regain tractability with these minimisation procedures when the ballots reflect a ranked liquid democracy election. The remaining four greedy procedures were all shown to resolve the complex delegations tractably. We showed that the outcomes found by MinSum are Pareto optimal with respect to the outcomes found from any consistent certificate. Moreover, the greedy procedures do not Pareto dominate one another.

In Chapter 3, we investigated whether models of delegative democracy could be extended to ensure that delegations can remain rational when there are multiple interconnected issues. Ensuring rationality is upheld is important in fine-grained liquid democracy, where voters can delegate to different agents on different decisions and may leave the agents' resulting votes to be irrational. Our contribution was to extend the model to a more general case where rationality is not restricted to a single setting with specific constraints. From this model, we gave procedures to obtain consistent votes for the voters. We proposed two procedures that minimise the changes required to regain consistency in the agents' votes. The first procedure, known from the literature, finds consistent votes by making minimal changes to the agents' ballots. The second novel procedure resolves the delegations and then makes minimal changes to these votes to find consistent votes. We showed that these minimisation procedures are intractable and then looked for other directions to gain tractable procedures. We proposed to elicit the agents' priorities over the issues to guide the procedures to find consistent votes. We showed that our two priority procedures, changing either votes or delegations, can find consistent votes in polynomial time. We showed that the priority procedures do not approximate their minimisation counterparts well — this was to be expected, as they have different objectives than the minimisation procedures. Hence, we saw that `PVC` is guaranteed to respect the agents' priorities more than its minimisation counterpart `MVC`. However, the same cannot be said about `PDC` and `MDC`. This chapter also explored how the original model can be extended to become more realistic. We did this in two ways: first, we allowed deletion cycles on specific issues; second, we allowed agents to give their own rationality constraints. We saw that these new models did not change the corresponding results from earlier in the chapter. The extension allowing voters to give their own personalised rationality constraint over their delegations is the most promising area for implementation as it perfectly reflects the initial purpose of the model.

Chapter 4 studied a related model of opinion diffusion, where we extended the notion of influence to be more expressive, similar to Chapter 2. Instead of agents updating opinions when some proportion of their influencers had a different opinion, our model looked at opinions updating with respect to a Boolean function. We showed that our underlying model of Boolean opinion diffusion is a Boolean network. The next purpose of this chapter was to take well-known results from binary opinion diffusion (where opinions are updated with quota rules) and investigate whether there was an equivalent result in our model. We showed that it is a PSpace-complete problem to recognise whether a given initial state leads to stability in synchronous diffusion, generalising a known result on majoritarian opinion diffusion. A result for majoritarian updates showed an asynchronous update procedure that maximises agreement among the agents' opinions. We showed that this result does not generally hold in our model. Yet, an equivalent result holds when we restrict the update functions to contain only positive or only negative literals. We connected Boolean opinion diffusion and multi-agent delegative democracy (as given in Chapter 2), and we showed that the models align and give the same final opinions or votes. This reflects the idea that delegations can happen outside the mechanism. Moreover, we inspected opinion control in this model and showed that influence maximisation is an NP-hard problem. We also rephrased known results from the Boolean network literature in terms of diffusion to showcase the synergy of the two research subjects.

Lastly, in Chapter 5, we analysed delegations in the classical model of liquid democracy. This comprised two different streams of research. The first, studied in Section 5.2, extended the Penrose-Banzhaf index in classical liquid democracy.

We assumed that the voters had no predetermined opinions on the issue or affinity to have certain voters as delegates. Hence, in our extension, agents were indifferent with respect to their possible delegates, modelled as their neighbours in the underlying social network. Given this assumption over the agent's voting options, our measure quantifies the power a voter has within the structure of the election to change its outcome. We showed that computing our index is #P-hard when we imposed no restrictions on the underlying network. The measure allows us to see which properties a voter has, such as if they are well-connected in the social network or if their voting weight affects their power to make the final decision. The second method we inspected to analyse classical liquid democracy was via long-term delegations in Section 5.3. We did not model the use of delegations classically. Instead of delegations being given each election separately, we developed the function of delegations as a means of voting on an issue when voting directly is not possible (due to time constraints or lack of expertise). This model reflects delegative democracy platforms. However, it has not been studied theoretically. Moreover, our purpose was to explore the intentional creation of delegation cycles seen on the LiquidFeedback platform. To mimic this real-world application, each agent votes with some probability and relies on their delegation otherwise. From this, we built a delegation game where every agent chooses their delegation as their action, and their utility is the expected payoff of their delegation. Given this game, we gave the standard notions of best responses and Nash equilibria. Although we did not prove that Nash equilibria are not guaranteed, we have made strides towards showing this. Through simulations, we showed that best response dynamics found a Nash equilibrium from our varied 20,000 instances. We also looked at using potential functions to prove the existence of Nash equilibria in this model; moreover, we note that many equilibria contain cycles.

6.2 Perspectives and Future Work

This thesis has focused on extending the notion of delegations to be more expressive and rational in order to improve digital democracy platforms. There are still many areas of future work in extending delegations, both as outlined in the conclusions of each chapter and related topics involving delegations untouched by this thesis. For instance, one area for future work could be creating voting models which mix different frameworks, such as addressing the effects of combining sortition[1] and liquid democracy. There are many ways in which digital democracy can be improved that do not involve delegations. In this section, we will outline two of these areas explored, which pertain to a better understanding of digital democracy platforms. The first avenue we explored was addressing that digital democracy platforms can return more information to the voters than just their collective agreements. This

[1]A model where a set of representatives is randomly chosen from the population. In some models, the random selection is done consistently with proportionality constraints based on description features of the population, e.g., 50% of the representatives should be women.

is unlike standard voting scenarios where brevity in returning outcomes is key. Hence, we study a family of aggregation rules that measure the divisiveness of an issue. The second avenue that we explored uses weighted judgment aggregation as a unifying framework for models of *collective combinatorial optimisation problems* (CCO). These are voting scenarios with similar structures, such as multi-winner elections, participatory budgeting and collective scheduling. CCO problems are a natural candidate to be implemented by digital democracy platforms due to digital ballots making the combinatorial aspect of the model (many issues to be voted on, each with their own details) more accessible. Moreover, many of these voting scenarios have already been implemented online, such as participatory budgeting, or have some subcase being voted on, for example, Doodle[2] is used to schedule a meeting instead of ordering a string of meetings.

6.2.1 Introducing Divisiveness Measures into Digital Platforms

Digital democracy platforms benefit from the ability to return more information about the vote to the voters with a very low communication cost. Therefore, there is a need for theoretical research into other purposes of aggregating opinions. Navarrete et al. [2022] created a preference elicitation platform called MonProgramme[3] during the French presidential election in 2022 to determine a collective governmental programme among the users. They returned not only a collective ranking of the proposals but also a ranking of how divisive the proposals were. Therefore, one of my research directions has contributed to the theoretical study of divisiveness first introduced by Navarrete et al. [2022] and is joint work with Carlos Navarrete, Mariana Macedo and César Hidalgo who are based at CCL, ANITI, IRIT, Université de Toulouse and Umberto Grandi (IRIT, Université Toulouse Capitole) [Colley et al., 2023d].

Rank aggregation is the problem of ordering issues in such a way that summarises a collection of individual rankings. This problem has been studied extensively in computational social choice (see, for example, Brandt et al. [2016]) when the rankings are assumed to represent human preferences. For example, one may be interested in each individual's ranking of preference over candidates in a political election, a set of projects to be funded, or any set of alternative proposals. The most common approach in this literature is to find normative desiderata for the aggregation process, including computational requirements such as the existence of algorithms for tractable computation and characterising the aggregators that satisfy them. Previous work in rank aggregation focused on how to best elicit which issues are the most agreed upon by the underlying population, with little interest in identifying those issues that divide them. We illustrate this in the following example:

Example 6.1. Consider three groups of agents and each group has a homogeneous preference over the issues a, b, c, d and e. The preferences of the three groups can

[2] `https://doodle.com/`
[3] `https://monprogramme2022.org/`

be described as such:

Group 1	$a \succ b \succ c \succ d \succ e$
Group 2	$a \succ d \succ c \succ e \succ b$
Group 3	$a \succ d \succ b \succ e \succ c$

We understand the preferences as such: each member of Group 1 prefers issue a to any other issue, then they prefer b to issues c, d and e, and so on. Given the groups' preferences, it is reasonable to assume that most aggregation rules will ensure that issue a is ranked the highest in the collective agreement ranking. For example, we see that two well-known rank aggregation rules, namely the Borda rule and Copeland rule, would rank a first, no matter the size of the groups. However, it is not immediately apparent from these rankings which issue should be considered the most divisive. One approach could say that issue b is the most divisive as it is sometimes ranked second and sometimes last. Thus, it has the most significant variance of where it is ranked out of the issues. However, if Group 2 only consisted of a few agents, then b may be less divisive than issues c and d, depending on the function used to measure divisiveness. $\triangle$

We put forward a family of functions that, starting from a collection of individual rankings, can order issues based on their divisiveness, extending the definition from Navarrete et al. [2022]. An issue's divisiveness is found by aggregating the disagreement among all possible subpopulations defined by the relative preference among the other issues. This work connects the definition of divisiveness to the field of computational social choice by parameterising it in terms of collective rank aggregation functions. We also connect our divisiveness measure to other measures of disagreement, such as a classical measure of polarisation from Esteban and Ray [1994] and the rank-variance of an issue. Although using a divisiveness measure is beneficial because it can keep voters informed about the collective's opinions, it can also have other uses. One such application of a divisiveness measure could be to help the community work towards consensus. There is hope that unity in a community can be achieved by addressing and discussing the most divisive issue. Hence, the population would be more cohesive by decreasing the total divisiveness. For example, recent work has suggested that the creation of recommender systems could help depolarise a population [Stray, 2022].

6.2.2 Connecting Collective Combinatorial Optimisation Problems

In this second avenue of research into digital democracy, we examine many different settings of collective decision-making over which items should be accepted from an agenda with a combinatorial nature and the accepted issue must respect some collective constraint, such as ordering a complete transitive schedule without gaps or respecting a budget limit. We characterise each of these settings as collective combinatorial optimisation (CCO) problems. Many of these CCO problems have been studied separately, while they all share many common structural features. Our contribution establishes novel and fundamental connections between several lines of research and between several problems that have been studied independently thus far. Identifying those connections may lead to meaningful insights and implications across different streams of research. In doing so, we give an engineering flavour to judgment aggregation, a field that up until now has focused on impossibility results, axiomatisation and computational complexity of winner determination. This work connecting weighted judgment aggregation and CCO problems is joint work with Linus Boes (Heinrich-Heine-Universität Düsseldorf), Umberto Grandi (IRIT, Université Toulouse Capitole), Jérôme Lang (CNRS, PSL), and Arianna Novaro (CES, Université Paris 1 Panthéon-Sorbonne). Full details are given in Boes et al. [2021].

We use weighted judgment aggregation as a general model in which many specific CCO settings can be framed. Thus, each of these specific CCO settings can be aligned structurally. Moreover, this ensures that the independent study in each setting can be shared with either judgment aggregation or even with the sister settings directly. One judgment aggregation rule we consider is the median rule from Nehring and Pivato [2022]. We showed that the following sister CCO rules are instances of the median rule:

- the standard *multi-winner approval voting rule* that outputs the most approved k candidates (modulo tie-breaking) in multi-winner elections;
- the *max rule with cardinality satisfaction* from Talmon and Faliszewski [2019] in participatory budgeting;
- the maximum collective spanning tree from Darmann et al. [2008] is the collective networking setting;
- the utilitarian aggregation rule with swap distance from Pascual et al. [2018] in the collective scheduling problem.

This study showed that judgment aggregation is a good candidate for declarative language to express various CCO problems, although other models may be equally good candidates. Yet, we needed a slight generalisation of judgment aggregation where issues are weighted, and these weights may be asymmetric. However, this generalisation allows for specific CCO problems to be seen through the lens of judgment aggregation. We showed that many rules studied for their specific settings are instances of the more general judgment aggregation rules. This shows strong connections between two specific ‘sister’ rules that are instances of the same general rules and share common normative properties.

6.3 Final Remarks

This thesis has made strides towards modelling delegations to be more expressive and more rational in delegative democracy. We did this with *exploratory research* into new models that extend classical liquid democracy by making delegations in our models more expressive (i.e., multi-agent ranked delegations) and more rational (i.e., by preserving consistency in multi-issue liquid democracy). In these settings, we saw that this allowed voters to decide exactly how their vote should be determined. We also studied classical liquid democracy through the lens of *responsive research*, as described in the introduction, by analysing its structure through a priori voting power and in terms of rational delegation choices on long-term delegations. There are still many interesting open research directions in the study of delegative democracy, mainly in line with turning exploratory research into responsive research. For instance, models exploring the possibilities of delegative democracy (Chapter 2 and 3) have not yet been fully assessed, e.g., a priori voting power has not been studied in these settings. The most promising research currently being undertaken in studying delegative democracy is through the study of how delegations are used in practice, and it is from here that we should be deciding the direction of our theoretical research.

Bibliography

Ben Abramowitz and Nicholas Mattei. Flexible representative democracy: An introduction with binary issues. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

Leyla Ade, Matteo Michelini, and Pietro Vigiani. Proportionality in Liquid Democracy and Representative Democracy. *Proceedings of the ESSLLI 2022 Student Session*, 2022.

Tatsuya Akutsu, Satoru Miyano, and Satoru Kuhara. Identification of genetic networks from a small number of gene expression patterns under the boolean network model. In *Biocomputing'99*, pages 17–28. World Scientific, 1999.

Tatsuya Akutsu, Morihiro Hayashida, Wai-Ki Ching, and Michael K Ng. On the complexity of finding control strategies for boolean networks. In *Proceedings of the Fourth Asia-Pacific Bioinformatics Conference*, 2006.

Tatsuya Akutsu, Morihiro Hayashida, Wai-Ki Ching, and Michael K Ng. Control of boolean networks: Hardness results and algorithms for tree structured networks. *Journal of theoretical biology*, 244(4):670–679, 2007.

Tatsuya Akutsu, Morihiro Hayashida, and Takeyuki Tamura. Algorithms for inference, analysis and control of boolean networks. In *International Conference on Algebraic Biology*, 2008.

Tatsuya Akutsu, Sven Kosub, Avraham A Melkman, and Takeyuki Tamura. Finding a periodic attractor of a boolean network. *IEEE/ACM transactions on computational biology and bioinformatics*, 9(5):1410–1421, 2012.

Dan Alger. Voting by proxy. *Public Choice*, 126(1-2):1–26, 2006.

R Michael Alvarez and Thad E Hall. *Electronic elections: The perils and promises of digital democracy.* Princeton University Press, 2010.

Guillermo de Anda-Jáuregui, Jesús Espinal-Enríquez, Santiago Sandoval-Motta, and Enrique Hernández-Lemus. A boolean network approach to estrogen transcriptional regulation. *Complexity*, 2019, 2019.

Elliot Anshelevich, Zack Fitzsimmons, Rohit Vaish, and Lirong Xia. Representative proxy voting. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

Ch Anwar ul Hassan, Muhammad Hammad, Jawaid Iqbal, Saddam Hussain, Syed Sajid Ullah, Hussain AlSalman, Mogeeb AA Mosleh, and Muhammad Arif. A liquid democracy enabled blockchain-based electronic voting system. *Scientific Programming*, 2022:1–10, 2022.

Ben Armstrong and Kate Larson. On the limited applicability of liquid democracy. In *Proceedings of the Third Games, Agents, and Incentives Workshop (GAIW)*, 2021.

Vincenzo Auletta, Diodato Ferraioli, and Gianluigi Greco. Reasoning about consensus when opinions diffuse through majority dynamics. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

Vincenzo Auletta, Diodato Ferraioli, and Gianluigi Greco. On the complexity of reasoning about opinion diffusion under majority dynamics. *Artificial Intelligence*, 284:103–288, 2020a.

Vincenzo Auletta, Diodato Ferraioli, and Gianluigi Greco. On the effectiveness of social proof recommendations in markets with multiple products. In *Proceedings of the Twenty-Fourth European Conference on Artificial Intelligence (ECAI)*, 2020b.

Vincenzo Auletta, Diodato Ferraioli, and Gianluigi Greco. Optimal majority dynamics for the diffusion of an opinion when multiple alternatives are available. *Theoretical Computer Science*, 869:156–180, 2021.

John F Banzhaf III. Weighted voting doesn't work: A mathematical analysis. *Rutgers Law Review*, 19:317, 1964.

Ruben Becker, Gianlorenzo D'angelo, Esmaeil Delfaraz, and Hugo Gilbert. Unveiling the truth in liquid democracy with misinformed voters. In *Algorithmic Decision Theory: Seventh International Conference (ADT)*, pages 132–146. Springer, 2021.

Benjamin B Bederson, Bongshin Lee, Robert M Sherman, Paul S Herrnson, and Richard G Niemi. Electronic voting system usability issues. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2003.

J. Behrens, A. Kistner, A. Nitsche, and B. Swierczek. *The principles of LiquidFeedback*. Interacktive Demokratie, Berlin, 2014.

Jan Behrens and Björn Swierczek. Preferential delegation and the problem of negative voting weight. *The Liquid Democracy Journal*, 3, 2015.

Jan Behrens, Axel Kistner, Andreas Nitsche, and Björn Swierczek. The Temporal Dimension in the Analysis of Liquid Democracy Delegation Graphs. *The Liquid Democracy Journal*, 2022.

Sebastian Bervoets and Vincent Merlin. Gerrymander-proof representative democracies. *International Journal of Game Theory*, 41:473–488, 2012.

Gili Bielous and Reshef Meir. Proxy manipulation for better outcomes. In *Proceedings of the Multi-Agent Systems: Nineteenth European Conference (EUMAS)*, 2022.

Daan Bloembergen, Davide Grossi, and Martin Lackner. On rational delegations in liquid democracy. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

Christian Blum and Christina Isabel Zuber. Liquid democracy: Potentials, problems, and perspectives. *Journal of Political Philosophy*, 24(2):162–182, 2016.

F. C. Bock. An algorithm to construct a minimum directed spanning tree in a directed network. *Developments in operations research*, pages 29–44, 1971.

Linus Boes, Rachael Colley, Umberto Grandi, Jérôme Lang, and Arianna Novaro. Collective discrete optimisation as judgment aggregation. *arXiv preprint arXiv:2112.00574*, 2021.

Vernon Bogdanor. First-past-the-post: An electoral system which is difficult to defend. *Representation*, 34(2):80–83, 1997.

Paolo Boldi, Francesco Bonchi, Carlos Castillo, and Sebastiano Vigna. Viscous democracy for social networks. *Communications of the ACM*, 54(6):129–137, 2011.

Sirin Botan, Umberto Grandi, and Laurent Perrussel. Multi-issue opinion diffusion under constraints. In *Proceedings of the Eighteenth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2019.

Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. *Handbook of computational social choice*. Cambridge University Press, 2016.

Robert Bredereck and Edith Elkind. Manipulating opinion diffusion in social networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

Markus Brill. Interactive democracy. In *Proceedings of the Seventeenth International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018.

Markus Brill and Nimrod Talmon. Pairwise liquid democracy. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

Markus Brill, Edith Elkind, Ulle Endriss, and Umberto Grandi. Pairwise diffusion of preference rankings in social networks. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

Markus Brill, Théo Delemazure, Anne-Marie George, Martin Lackner, and Ulrike Schmidt-Kraepelin. Liquid democracy with ranked delegations. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

Ahto Buldas and Triinu Mägi. Practical security analysis of e-voting systems. In *Proceedings of the Advances in Information and Computer Security: Second International Workshop on Security*, 2007.

Joseph Campbell, Alessandra Casella, Lucas de Lara, Victoria R Mooers, and Dilip Ravindran. Liquid democracy. two experiments on delegation in voting. Technical report, National Bureau of Economic Research, 2022.

Ioannis Caragiannis and Evi Micha. A contribution to the critique of liquid democracy. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

Sara D Cardell and Amparo Fúster-Sabater. Binomial representation of cryptographic binary sequences and its relation to cellular automata. *Complexity*, 2019: 1–13, 2019.

John R Chamberlin and Paul N Courant. Representative deliberations and representative decisions: Proportional representation and the Borda rule. *The American Political Science Review*, pages 718–733, 1983.

Wei Chen, Yifei Yuan, and Li Zhang. Scalable influence maximization in social networks under the linear threshold model. In *2010 IEEE International Conference on Data Mining*, pages 88–97, 2010.

Daizhan Cheng. Semi-tensor product of matrices and its applications-a survey. *Procceeings of the Fourth International Congress of Chinese Mathematicians (ICCM)*, 2007.

Daizhan Cheng, Zhiqiang Li, and Hongsheng Qi. A survey on boolean control networks: A state space approach. *Three Decades of Progress in Control Sciences*, pages 121–139, 2010a.

Daizhan Cheng, Hongsheng Qi, and Zhiqiang Li. *Analysis and control of Boolean networks: a semi-tensor product approach.* Springer Science & Business Media, 2010b.

Flavio Chierichetti, Jon Kleinberg, and Sigal Oren. On discrete preferences and coordination. In *Proceedings of the fourteenth ACM conference on Electronic commerce (EC)*, 2013.

Dmitry Chistikov, Grzegorz Lisowski, Mike Paterson, and Paolo Turrini. Convergence of opinion diffusion is pspace-complete. In *Proceedings of the Association for the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Zoé Christoff and Davide Grossi. Binary voting with delegable proxy: An analysis of liquid democracy. In *Proceedings Sixteenth Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, 2017a.

Zoé Christoff and Davide Grossi. Stability in binary opinion diffusion. In *Logic, Rationality, and Interaction: Sixth International Workshop (LORI)*, 2017b.

Yoeng-Jin Chu. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400, 1965.

Gal Cohensius, Shie Mannor, Reshef Meir, Eli A. Meirom, and Ariel Orda. Proxy voting for better outcomes. In *Proceedings of the Sixteenth Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2017.

James S Coleman. Control of collectivities and the power of a collectivity to act. *Social choice (Routledge Revivals)*, pages 269–300, 1971.

Rachael Colley and Umberto Grandi. Preserving consistency in multi-issue liquid democracy. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI)*, 2022a.

Rachael Colley and Umberto Grandi. The spread of opinions via boolean networks. In *Proceedings of the Multi-Agent Systems: Nineteenth European Conference (EUMAS)*, 2022b.

Rachael Colley, Umberto Grandi, and Arianna Novaro. Smart voting. In *Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

Rachael Colley, Umberto Grandi, and Arianna Novaro. Unravelling multi-agent ranked delegations. *Autonomous Agents and Multi-Agent Systems*, 36(1):9, 2022.

Rachael Colley, Théo Delemazure, and Hugo Gilbert. Measuring a priori voting power–taking delegations seriously. *arXiv preprint arXiv:2301.02462*, 2023a.

Rachael Colley, Théo Delemazure, and Hugo Gilbert. Measuring a priori voting power in liquid democracy. *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)*, 2023b.

Rachael Colley, Théo Delemazure, and Hugo Gilbert. Taking delegations seriously: Measuring a priori voting power (Extended abstract). In *Proceedings of the Twenty-Second International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2023c.

Rachael Colley, Umberto Grandi, César Hidalgo, Mariana Macedo, and Carlos Navarrete. Measuring and controlling divisiveness in rank aggregation. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, 2023d.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

Yves Crama and Peter L Hammer. *Boolean functions: Theory, algorithms, and applications*. Cambridge University Press, 2011.

Andreas Darmann, Christian Klamler, and Ulrich Pferschy. Computing spanning trees in a social choice context. In *Proceedings of the Second International Workshop on Computational Social Choice (COMSOC)*, 2008.

Ronald De Haan. Hunting for tractable languages for judgment aggregation. In *Proceedings of the Sixteenth International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2018.

Jonas Degrave. Resolving multi-proxy transitive vote delegation. *arXiv preprint arXiv:1412.4039*, 2014.

Palash Dey, Arnab Maiti, and Amatya Sharma. On parameterized complexity of liquid democracy. In *Proceedings of the Algorithms and Discrete Applied Mathematics: Seventh International Conference (CALDAM)*, 2021.

Amrita Dhillon, Grammateia Kotsialou, Peter McBurney, Luke Riley, et al. Introduction to voting and the blockchain: some open questions for economists. Technical report, Competitive Advantage in the Global Economy (CAGE), 2019.

Charles Lutwidge Dodgson. *The Principles of Parliamentary Representation*. Harrison and Sons, 1884.

Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.

Kevin Doyle. EU recount delay now shows need for e-voting - Minister — independent.ie. `https://www.independent.ie/irish-news/elections-2019/european/eu-recount-delay-now-shows-need-for-e-voting-minister-38173482.html`, 2019. [Accessed 25-Mar-2023].

Jack Edmonds. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240, 1967.

Piret Ehin, Mihkel Solvak, Jan Willemson, and Priit Vinkel. Internet voting in estonia 2005–2019: Evidence from eleven elections. *Government Information Quarterly*, 39(4):101718, 2022.

Bruno Escoffier, Hugo Gilbert, and Adèle Pass-Lanneau. The convergence of iterative delegations in liquid democracy in a social network. In *Proceedings of the Twelfth International Symposium on Algorithmic Game Theory (SAGT)*, 2019.

Bruno Escoffier, Hugo Gilbert, and Adèle Pass-Lanneau. Iterative delegations in liquid democracy with restricted preferences. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Joan-Maria Esteban and Debraj Ray. On the measurement of polarization. *Econometrica: Journal of the Econometric Society*, pages 819–851, 1994.

Piotr Faliszewski and Jörg Rothe. *Handbook of Computational Social Choice*, chapter Control and Bribery in Voting. Cambridge University Press, 2016.

Piotr Faliszewski, Rica Gonen, Martin Koutecý, and Nimrod Talmon. Opinion diffusion and campaigning on society graphs. *Journal of Logic and Computation*, 32(6):1162–1194, 2022.

Christopher Farrow, Jack Heidel, John Maloney, and Jim Rogers. Scalar equations for synchronous boolean networks with biological applications. *IEEE Transactions on Neural Networks*, 15(2):348–354, 2004.

Dan S Felsenthal and Moshé Machover. Ternary voting games. *International journal of game theory*, 26(3):335–351, 1997.

Dan S Felsenthal and Moshé Machover. The measurement of voting power: Theory and practice, problems and paradoxes. In *The Measurement of Voting Power*. Edward Elgar Publishing, 1998.

Dan S Felsenthal and Moshé Machover. Models and reality: the curious case of the absent abstention. In *Power indices and coalition formation*, pages 87–103. Springer, 2001.

Dan S Felsenthal and Moshé Machover. A priori voting power: what is it all about? *Political Studies Review*, 2(1):1–23, 2004.

Dan S Felsenthal and Moshé Machover. Voting power measurement: a story of misreinvention. *Social choice and welfare*, 25(2):485–506, 2005.

Diodato Ferraioli and Carmine Ventre. Social pressure in opinion games. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

Bryan Alexander Ford. Delegative democracy. Technical report, 2002. Unpublished Manuscript. Available at: `https://bford.info/deleg/deleg.pdf`.

Josep Freixas. Probabilistic power indices for voting rules with abstention. *Mathematical Social Sciences*, 64(1):89–99, 2012.

Josep Freixas and Roberto Lucchetti. Power in voting rules with abstention: an axiomatization of a two components power index. *Annals of operations research*, 244(2):455–474, 2016.

Josep Freixas and William S Zwicker. Weighted voting, abstention, and multiple levels of approval. *Social choice and welfare*, 21(3):399–431, 2003.

Noah E. Friedkin, Anton V. Proskurnikov, Roberto Tempo, and Sergey E. Parsegov. Network science on belief system dynamics under logic constraints. *Science*, 354 (6310):321–326, 2016.

Cédric Gaucherel, H Théro, A Puiseux, and Vincent Bonhomme. Understand ecosystem regime shifts by modelling ecosystem development using boolean networks. *Ecological Complexity*, 31:104–114, 2017.

Andrew Gelman, Jonathan N Katz, and Francis Tuerlinckx. The mathematics and statistics of voting power. *Statistical Science*, pages 420–435, 2002.

E. Goles and J. Olivos. Periodic behaviour of generalized threshold functions. *Discrete Mathematics*, 30(2):187–189, 1980.

Eric Goles and Lilian Salinas. Sequential operator for filtering cycles in Boolean networks. *Advances in Applied Mathematics*, 45(3):346–358, 2010.

Eric Goles, Pedro Montealegre, Ville Salo, and Ilkka Törmä. Pspace-completeness of majority automata networks. *Theoretical Computer Science*, 609:118–128, 2016.

Paul Gölz, Anson Kahng, Simon Mackenzie, and Ariel D Procaccia. The fluid mechanics of liquid democracy. In *Proceedings of the International Conference on Web and Internet Economics (ICWIE)*, 2018.

Umberto Grandi. Social choice and social networks. *Trends in Computational Social Choice*, pages 169–184, 2017.

Umberto Grandi, Emiliano Lorini, and Laurent Perrussel. Propositional opinion diffusion. In *Fourteenth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2015.

M. Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 83(6):1420–1443, 1978.

Moyra Grant. *UK Parliament.* Edinburgh University Press, 2009.

James Green-Armytage. Direct voting and proxy voting. *Constitutional Political Economy*, 26(2):190–220, 2015.

Jacqueline Harding. Proxy selection in transitive proxy voting. *Social Choice and Welfare*, 58(1):69–99, 2022.

S. Hardt and L. CR Lopes. Google votes: A liquid democracy experiment on a corporate social network. 2015.

Katsumi Inoue. Logic programming for boolean networks. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

Pallavi Jain, Krzysztof Sornat, and Nimrod Talmon. Preserving consistency for liquid knapsack voting. In *Proceedings of the Twentieth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2021.

Pallavi Jain, Krzysztof Sornat, and Nimrod Talmon. Preserving consistency for liquid knapsack voting. In *Proceedings of Multi-Agent Systems: Nineteenth European Conference (EUMAS)*, 2022.

Anson Kahng, Simon Mackenzie, and Ariel D Procaccia. Liquid democracy: An algorithmic perspective. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Anson Kahng, Simon Mackenzie, and Ariel Procaccia. Liquid democracy: An algorithmic perspective. *Journal of Artificial Intelligence Research*, 70:1223–1252, 2021.

Richard M Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.

Stuart Kauffman. Homeostasis and differentiation in random genetic control networks. *Nature*, 224(5215):177–178, 1969.

Stuart A Kauffman. *The origins of order: Self-organization and selection in evolution.* Oxford University Press, USA, 1993.

David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*, 2003.

Christoph Kling, Jérôme Kunegis, Heinrich Hartmann, Markus Strohmaier, and Steffen Staab. Voting behaviour and power in online democracy: A study of liquidfeedback in germany's pirate party. In *Proceedings of the International AAAI Conference on Web and Social Media*, 2015.

Koichi Kobayashi. Design of fixed points in boolean networks using feedback vertex sets and model reduction. *Complexity*, 2019.

Kathrin Konczak and Jérôme Lang. Voting procedures with incomplete preferences. In *Proceedings of the IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling (MPREF)*, 2005.

Sven Kosub. Dichotomy results for fixed-point existence problems for boolean dynamical systems. *Mathematics in Computer Science*, 1(3):487–505, 2008.

Grammateia Kotsialou and Luke Riley. Incentivising participation in liquid democracy with breadth-first delegation. In *Proceedings of the Nineteenth International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2020.

Dexter C Kozen. *The design and analysis of algorithms.* Springer Science & Business Media, 2012.

Romane Kulesza. Development of a web platform to study voting procedures with ranked delegations. Technical report, Université PSL (Paris Sciences & Lettres), 2022. Internship report.

Sascha Kurz. Measuring voting power in convex policy spaces. *Economies*, 2(1): 45–77, 2014.

Jérôme Lang, Gabriella Pigozzi, Marija Slavkovik, and Leendert van der Torre. Judgment aggregation rules based on minimization. In *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, 2011.

Christopher James Langmead and Sumit Kumar Jha. Symbolic approaches for finding control strategies in boolean networks. *Journal of Bioinformatics and Computational Biology*, 7(02):323–338, 2009.

W Lucas. Measuring power in weighted voting systems. Technical report, Cornell University Operations Research and Industrial Engineering, 1974.

Georg Lutz. Low turnout in direct democracy. *Electoral Studies*, 26(3):624–632, 2007.

Nathaniel Macon and Abraham Spitzbart. Inverses of vandermonde matrices. *The American Mathematical Monthly*, 65(2):95–100, 1958.

Pia Mancini. How to upgrade democracy for the Internet era. `https://www.ted.com/talks/pia_mancini_how_to_upgrade_democracy_for_the_internet_era`, 2014. [Accessed 25-Mar-2023].

Evangelos Markakis and Georgios Papasotiropoulos. An approval-based model for single-step liquid democracy. In *Algorithmic Game Theory: Fourteenth International Symposium, (SAGT)*, 2021.

Tomomi Matsui and Yasuko Matsui. A survey of algorithms for calculating power indices of weighted majority games. *Journal of the Operations Research Society of Japan*, 43(1):71–86, 2000.

Walter R Mebane. The wrong man is president! overvotes in the 2000 presidential election in florida. *Perspectives on Politics*, 2(3):525–535, 2004.

James C Miller. A program for direct and proxy voting in the legislative process. *Public choice*, 7(1):107–113, 1969.

Dov Monderer and Lloyd S Shapley. Potential games. *Games and economic behavior*, 14(1):124–143, 1996.

Colby Morrison and Pavel Naumov. Group conformity in social networks. *Journal of Logic, Language and Information*, 29(1):3–19, 2020.

Hervé Moulin. *Axioms of Cooperative Decision Making.* Cambridge University Press, 1988.

J Nash. Non-cooperative games. *Annals of Mathematics*, 54, 1951.

Carlos Navarrete, Nicole Ferrada, Mariana Macedo, Rachael Colley, Jingling Zhang, Umberto Grandi, Jérôme Lang, and César A Hidalgo. Understanding political agreements and disagreements: Evidence from the 2022 french presidential election. *arXiv preprint arXiv:2211.04577*, 2022.

Klaus Nehring and Marcus Pivato. The median rule in judgement aggregation. *Economic Theory*, 73(4):1051–1100, 2022.

Mahdi Nejadgholi, Nan Yang, and Jeremy Clark. Short paper: ballot secrecy for liquid democracy. In *Financial Cryptography and Data Security (FC)*, 2021.

Jonathan A. Noel, Mashbat Suzuki, and Adrian Vetta. Pirates in wonderland: Liquid democracy has bicriteria guarantees. In *Algorithmic Game Theory - Fourteenth International Symposium (SAGT)*, 2021.

Guillermo Owen. Multilinear extensions and the banzhaf value. *Naval research logistics quarterly*, 22(4):741–750, 1975.

Guillermo Owen. Modification of the Banzhaf-Coleman index for games with a priori unions. In *Power, voting, and voting power*, pages 232–238. Springer, 1981.

Guillermo Owen. Multilinear extensions of games. *The Shapley Value. Essays in Honor of Lloyd S. Shapley*, pages 139–151, 1988.

Fanny Pascual, Krzysztof Rzadca, and Piotr Skowron. Collective schedules: Scheduling meets computational social choice. In *Proceedings of the Seventeenth International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018.

Alois Paulin. An overview of ten years of liquid democracy research. In *Proceedings of the Twenty-First Annual International Conference on Digital Government Research*, pages 116–121, 2020.

Lionel S Penrose. The elementary statistics of majority voting. *Journal of the Royal Statistical Society*, 109(1):53–57, 1946.

Kislaya Prasad and Jerry S Kelly. NP-completeness of some problems concerning voting games. *International Journal of Game Theory*, 19(1):1–9, 1990.

Ahmed Rana, Ibrahim Zincir, and Samsun Basarici. The security and the credibility challenges in e-voting systems. In *European Conference on Cyber Warfare and Security*, 2015.

Manon Revel, Daniel Halpern, Adam Berinsky, and Ali Jadbabaie. Liquid democracy in practice: An empirical analysis of its epistemic performance. *ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization*, 2022a.

Manon Revel, Tao Lin, and Daniel Halpern. How many representatives do we need? the optimal size of a congress voting on binary issues. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2022b.

Manon Revel, Niclas Boehmer, Rachael Colley, Markus Brill, Piotr Faliszewski, and Edith Elkind. Selecting representative bodies: An axiomatic view. *arXiv preprint arXiv:2304.02774*, 2023.

Adrien Richard. Positive and negative cycles in boolean networks. *Journal of theoretical biology*, 463:67–76, 2019.

Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the Eighth ACM SIGKDD International Conference on knowledge discovery and data mining*, 2002.

Daniel J Rosenkrantz, Madhav V Marathe, SS Ravi, and Richard E Stearns. Synchronous dynamical systems on directed acyclic graphs (DAGs): Complexity and algorithms. Technical report, Biocomplexity Institute and Initiative, University of Virginia, 2020.

Amirali Salehi-Abari, Craig Boutilier, and Kate Larson. Empathetic decision making in social networks. *Artificial intelligence*, 275:174–203, 2019.

Ehud Shapiro. Grassroots vs. constitutional liquid democracy, 2022. Presentation at the Liquid Democracy Workshop at the University of Zurich, Switzerland, website at `https://democracynet.eu/activities/ldws2022/`.

Ehud Shapiro and Nimrod Talmon. Foundations for grassroots democratic metaverse: Blue sky ideas track. In *Proceedings of the Twenty-First International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1814–1818, 2022.

Lloyd S. Shapley and Martin Shubik. A method for evaluating the distribution of power in a committee system. *American political science review*, 48(3):787–792, 1954.

Llyod S. Shapley. A value for n-person games. *Contributions to the Theory of Games II, Annals of Mathematical Studies*, 28, 1953.

Wen Shen, Yang Feng, and Cristina V Lopes. Multi-winner contests for strategic diffusion in social networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

Ilya Shmulevich and Wei Zhang. Binary analysis and optimization-based normalization of gene expression data. *Bioinformatics*, 18(4):555–565, 2002.

Jonathan Stray. Designing recommender systems to depolarize. *First Monday*, 27 (5), 2022. doi: 10.5210/fm.v27i5.12604. URL `https://doi.org/10.5210/fm.v27i5.12604`.

Nimrod Talmon and Piotr Faliszewski. A framework for approval-based budgeting methods. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

Nicolaus Tideman. The single transferable vote. *Journal of Economic Perspectives*, 9(1):27–38, 1995.

Gordon Tullock. *Toward a mathematics of politics.* Ann Arbor: University of Michigan Press, 1967.

Giannis Tyrovolas. The limits of smart voting in liquid democracy. Master's thesis, University of Oxford, 2022.

Leslie G Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

Chiara Valsangiacomo. Clarifying and defining the concept of liquid democracy. *Swiss Political Science Review*, 28(1):61–80, 2022.

Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual.* CreateSpace, Scotts Valley, CA, 2009.

Yuzhe Zhang and Davide Grossi. Power in liquid democracy. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

Yuzhe Zhang and Davide Grossi. Tracking truth by weighting proxies in liquid democracy. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2022.

Zhiqiang Zhuang, Kewen Wang, Junhu Wang, Heng Zhang, Zhe Wang, and Zhiguo Gong. Lifting majority to unanimity in opinion diffusion. In *Proceedings of the Twenty-Fourth European Conference on Artificial Intelligence (ECAI)*, 2020.