Journal of Artificial Intelligence Research 81 (2024) 793-836 Submitted 05/2024; published 12/2024

Declarative Approaches to Outcome Determination in Judgment Aggregation

Ari Conati ARI.CONATI@HELSINKI.FI
Andreas Niskanen ANDREAS.NISKANEN@HELSINKI.FI
Matti Järvisalo MATTI.JARVISALO@HELSINKI.FI
Department of Computer Science,
University of Helsinki,
Finland

Abstract

Judgment aggregation offers a general formal framework for modeling various settings involving information aggregation by social choice mechanisms. For many judgment aggregation rules, computing collective judgments is computationally notoriously hard. The central outcome determination problem, in particular, is often complete for higher levels of the polynomial hierarchy. This complexity barrier makes it challenging to develop practical exact algorithms to outcome determination. Taking on this challenge, in this work we develop practical exact algorithms for outcome determination under a range of the most central judgment aggregation rules—namely Kemeny, Slater, MaxHamming, Young, Dodgson, Reversal scoring, Condorcet, Ranked agenda, and LexiMax—by harnessing the declarative approach, in particular, Boolean satisfiability (SAT) and integer programming techniques. For the Kemeny, Slater, MaxHamming, Young, and Dodgson rules, we detail direct approaches based on maximum satisfiability (MaxSAT) and integer programming. For the Reversal scoring, Condorcet, Ranked agenda, and LexiMax rules, we develop iterative algorithms, including algorithms based on the counterexample-guided abstraction refinement (CEGAR) paradigm, making use of recent advances in incremental MaxSAT solving and preferential SAT-based reasoning. We provide an open-source implementation of the algorithms, and empirically evaluate them using real-world preference data. We compare the performance of our implementation to a recent approach which makes use of declarative solver technology for answer set programming (ASP). The results demonstrate that our approaches scale significantly beyond the reach of the ASP-based algorithms for all of the judgment aggregation rules considered.

1. Introduction

Social choice theory (Arrow et al., 2002, 2011; Arrow, 2012) considers collective decision-making scenarios where individual agents must reach a group-level consensus. Computational social choice (Brandt et al., 2016) is a modern area of artificial intelligence research that considers social choice mechanisms from the computer science perspective. Of particular interest are the application of computational techniques to provide more stringent analysis of social choice mechanisms (in terms of, e.g., computational complexity and representational aspect) and the development of practical algorithms for computationally hard problems in social choice. For instance, the practical viability of a voting rule is dependent not only on normative criteria assessed by scholars from other disciplines but on its computational feasibility.

In this article, we develop practical exact algorithms for the outcome determination problem in judgment aggregation (Endriss, 2016). Judgment aggregation offers a general, formal framework for modeling various settings involving aggregation of information by social choice mechanisms. For example, in an election, voters delineate a preference ranking over a set of candidates, and an aggregation procedure identifies a winning candidate based on voter preferences. Judgment aggregation considers more generally the aggregation of individual judgments regarding the truth or falsehood of logical statements, thereby capturing various aggregation scenarios, including preference aggregation (Chingoma et al., 2022; Dietrich & List, 2007; Endriss, 2018; Endriss et al., 2012; Lang et al., 2017; Lang & Slavkovik, 2013; Miller & Osherson, 2009), graph aggregation (Endriss & Grandi, 2017), and various further collective decision-making scenarios involving multiple agents (Bodanza et al., 2017; Chen & Endriss, 2019; Rey et al., 2020). Various aggregation rules, governing what constitute judgment sets (i.e., aggregation results) of interest, have been proposed and analyzed, for different settings (Endriss, 2018; Lang et al., 2011; Miller & Osherson, 2009; Nehring et al., 2014).

In this work, we focus on outcome determination as a central problem in judgment aggregation (Endriss et al., 2020). Outcome determination constitutes the task of deciding (under a given aggregation rule) whether a given subset of the agenda—consisting of the issues the individual agents have expressed their judgments over—is accepted in the judgment sets defined by the aggregation rule. In terms of its generality, outcome determination generalizes, e.g., the winner determination problem in voting (Conitzer, 2006; Hemaspaandra et al., 1997, 2005; Rothe et al., 2003), where the task is to determine if a given alternative is a winner of an election.

For many judgment aggregation rules, computing collective judgments is computationally notoriously hard (de Haan & Slavkovik, 2017; Endriss & de Haan, 2015; Endriss et al., 2012; Lang & Slavkovik, 2014). The outcome determination problem in particular is even complete for complexity classes on the second level of the polynomial hierarchy under specific aggregation rules (Endriss et al., 2020). This complexity barrier makes it challenging to develop practical, generic algorithmic approaches to outcome determination. Various approaches have been developed for specific settings—in particular for specific voting rules (Conitzer, 2006; Conitzer et al., 2006; Davenport & Kalagnanam, 2004; Meila et al., 2007) and (web-based) tools for preference aggregation (Brandt et al., 2015; Charwat & Pfandler, 2015). However, a first more generic exact approach to judgment aggregation was only recently introduced (de Haan & Slavkovik, 2019) based on the declarative paradigm of answer set programming (ASP; see, e.g., Lifschitz, 2019). Despite its generality and motivations as a practical algorithmic approach to judgment aggregation, this ASP-based method was not empirically evaluated by the original authors. Furthermore, as we demonstrate in Section 7 of this work, the implementation of the approach does not scale beyond very small data. However, the approach does serve as the primary baseline against which to evaluate new algorithmic approaches to judgment aggregation.

In this work, we instead harness both Boolean satisfiability (SAT) based and integer programming declarative techniques for developing new exact algorithms for outcome determination. Complementing the more classical integer programming techniques, our motivations for considering SAT-based techniques are three-fold. Firstly, advances in SAT solving (Marques-Silva et al., 2021) have recently given rise to increasingly-effective solvers

for maximum satisfiability (MaxSAT; see, e.g., Bacchus et al., 2021; Li & Manyà, 2021) as the optimization extension of SAT. Secondly, since judgment sets take the form of logical formulas, propositional logic can be considered a natural choice for an encoding language towards computing optimal collective judgments under various rules—although, as we will detail, specific rules turn out to be more naturally represented using linear inequalities in the language of integer programming, which also motivate the use of integer programming in this context. Thirdly, specific judgment aggregation rules by complexity-theoretic argument require iterative calls to an NP-oracle (Endriss et al., 2020), and as we will detail, such rules are naturally captured by specific types of iterative SAT-based procedures, including instantiations of SAT-based counterexample-guided abstraction refinement (CEGAR; see, e.g., Clarke et al., 2003, 2004), incremental MaxSAT (Niskanen et al., 2022) and SAT-based preferential reasoning (Dodaro & Previti, 2019; Rosa et al., 2010).

The main contribution of this work is the development of practical, generic algorithms for outcome determination under a range of central judgment aggregation rules, namely, the Kemeny (Endriss, 2018; Endriss et al., 2012; Lang et al., 2011; Miller & Osherson, 2009; Nehring et al., 2014; Pigozzi, 2006), Slater (Endriss, 2018; Lang et al., 2011; Miller & Osherson, 2009; Nehring et al., 2014), Young (Lang et al., 2011), Dodgson (Miller & Osherson, 2009), MaxHamming (Lang et al., 2011), Reversal scoring (Dietrich, 2014), Condorcet (Endriss, 2018; Endriss et al., 2020; Lang et al., 2011; Nehring et al., 2014), Ranked agenda (Endriss & de Haan, 2015; Lang et al., 2011; Porello & Endriss, 2014), and LexiMax (Everaere et al., 2014; Nehring & Pivato, 2019) rules. Our algorithmic design is based on the computational complexity of the problem for the individual rules, utilizing various declarative techniques most suited for the individual aggregation rules. In particular, we develop a generic approach covering judgment aggregation under Kemeny, Slater, MaxHamming, Young, and Dodgson, and detail its instantiation through both MaxSAT and integer programming. We also capture further judgment aggregation rules (namely Reversal scoring, Condorcet, Ranked agenda and LexiMax) through incremental MaxSAT, SAT-based CEGAR, and SAT-based preferential reasoning (PrefSAT). We provide an open-source implementation of all of the procedures for outcome determination developed in this work (`https://bitbucket.org/coreo-group/satcha`). We empirically evaluate the performance of our implementation on preference data from the PrefLib (Mattei & Walsh, 2013) reference library on real-world voting data arising from, e.g., elections and surveys. The empirical results show that our approaches scale significantly beyond the reach of the recently proposed alternative ASP-based algorithms (de Haan & Slavkovik, 2019) for all of the judgment aggregation rules considered.

While our focus is on developing new declarative approaches to outcome determination in judgment aggregation, we note that declarative techniques have been earlier employed for solving problems in the realm of computational social choice. For example, SAT solvers and their extensions were harnessed by Tang and Lin (2009) and subsequently in various works (Brandl et al., 2018, 2019, 2021; Brandt & Geist, 2016; Brandt et al., 2017, 2018, 2022; Delemazure et al., 2023; Endriss, 2020; Geist & Endriss, 2011; Kluiving et al., 2020; Peters, 2018) to generate automated proofs for both already-established and new impossibility theorems. Declarative programming approaches have also been proposed for computationally hard problems such as computing justifications for collective decisions (Boixel & Endriss, 2020) and stable matching (Drummond et al., 2015).

Most recently, and perhaps most closely related to our work, an integer programming approach to judgment aggregation with weighted and asymmetric agendas was presented (Boes et al., 2023). For standard judgment aggregation (i.e., for unweighted and symmetric agendas), our integer programming encodings coincide with their approach for the specific cases of the Kemeny and MaxHamming rules (i.e., the median and egalitarian rules, respectively). However, Boes et al. (2023) do not cover the outcome determination problem. Furthermore, our SAT-based approaches are to the best of our knowledge entirely novel.

The rest of this article is structured as follows. We start with preliminaries on the judgment aggregation framework and the aggregation rules covered in this work (Section 2), as well as SAT and MaxSAT (Section 3). In Section 4, we describe a general algorithm for outcome determination that covers the Kemeny, Slater, MaxHamming, Young, and Dodgson rules, and detail its instantiation through MaxSAT. Further, in Section 5 we describe iterative SAT-based algorithms for the Reversal scoring, Condorcet, Ranked agenda, and LexiMax rules. Complementing the MaxSAT approach, in Section 6 we detail integer programming encodings which allow for capturing the Kemeny, Slater, MaxHamming, Young, Dodgson and LexiMax rules. Finally, we provide an overview of the results from empirical evaluation of our approaches on preference data in Section 7 before conclusions.

This article considerably extends and revises a preliminary version of this work presented at the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023; Conati et al., 2023). Algorithms and declarative encodings are presented in full detail with examples. The integer programming encodings presented in this article are entirely new content. The empirical evaluation has also been extended with results for the integer programming approaches included.

2. Judgment Aggregation

We start with background on judgment aggregation and outcome determination, following Endriss et al. (2020) and Endriss et al. (2016).

2.1 Modeling Judgment Aggregation

Judgment aggregation (Endriss et al., 2016; Lang et al., 2017) provides a general logical framework for modeling decisions made by consensus. We consider an *agenda* consisting of a set of *issues* which are each either accepted or rejected by a group of *agents*. The task is to aggregate the opinions of the individual agents to arrive at a single collective opinion over the issues. Optimal collective judgments are determined according to a *judgment aggregation rule*.

Formally, let $X = \{x_1, ..., x_m\}$ be a set of propositional variables representing the *issues*, and $\overline{X} = \{\neg x_1, ..., \neg x_m\}$. The sets X and $\overline{X}$ then denote acceptance and rejection of the issues, and the set $\Phi = X \cup \overline{X}$ denotes the entire *agenda*. We additionally denote by X_Φ the set of non-negated literals in the agenda Φ (this is also sometimes called the *pre-agenda* corresponding to Φ). A *judgment set* J is a subset $J \subseteq \Phi$, representing the opinion of an individual agent on the agenda. We say a judgment set J is *complete* if for all $x \in X$ we have either $x \in J$ or $\neg x \in J$.

Example 2.1. A hiring committee needs to decide whether to accept or reject each of three applicants. Let a_1, a_2, a_3 be propositional variables where a_i denotes the hiring status of applicant i, giving rise to the agenda $\Phi = \{a_1, \neg a_1, a_2, \neg a_2, a_3, \neg a_3\}$. The opinion of a committee member who favors accepting the first two applicants and rejecting the third is represented by the judgment set $J = \{a_1, a_2, \neg a_3\}$. This judgment set is complete because, it includes either a_i or $\neg a_i$ for each variable a_i. ◊

Issues may be subject to constraints based on features of the specific scenario being modeled or the relationships among the issues. For instance, casting voting as judgment aggregation, variables are constrained by requiring that judgment sets must represent valid orderings of candidates. Generally, a constraint is expressed as a propositional formula, and a judgment set is logically consistent if it satisfies the formula. Formally, given a propositional formula Γ, we say J is Γ-*consistent* with respect to Γ if $\Gamma \wedge \bigwedge_{l \in J} l$ is satisfiable. We typically represent complete and Γ-consistent judgment sets as truth assignments satisfying Γ, where each issue $x \in X$ is assigned to 1 if $x \in J$ and 0 otherwise.

Example 2.2. Continuing from Example 2.1, suppose the second and third applicants are applying for the same position, so they cannot both be hired. This constraint can be expressed as the propositional formula $\Gamma = (\neg a_2 \vee \neg a_3)$. The judgment set $J = \{a_1, a_2, \neg a_3\}$ is Γ-consistent since $\neg a_3 \in J$. ◊

Let $\mathcal{J}(\Phi, \Gamma)$ denote the collection of all complete and Γ-consistent judgment sets. The notion of a judgment aggregation frameworks is defined as follows.

Definition 2.1 (Judgment aggregation frameworks)**.** A *judgment aggregation framework* is a tuple $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$, where Φ is the agenda, Γ_{in} and Γ_{out} are propositional formulas referred to as the *input* and *output constraints*, respectively, and $P = (J_1, ..., J_n)$, where the J_i are judgment sets representing the opinions of the individual agents, is the *profile*.

The input and output constraints are problem-specific restrictions on individual and collective judgment sets, respectively (Endriss, 2018). We assume without loss of generality that $J_i \in \mathcal{J}(\Phi, \Gamma_{\text{in}})$ for all $J_i \in P$. A *collective judgment set* J for a given judgment aggregation framework $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$ is similarly required to be consistent with respect to Γ_{out}, that is, we require that $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$. Note that the formulas Γ_{in} and Γ_{out} may contain variables outside Φ (e.g., auxiliary variables employed in expressing logical relationships among the issues).[1]

The *support* for an issue $l \in \Phi$, denoted by $N(P, l)$, is the number of supporters of agenda item l, i.e., $N(P, l) = |\{J_i \in P \mid l \in J_i\}|$. The *majoritarian judgment set* $J_m(P)$ is the set of issues which are supported by a majority of the agents, i.e., $J_m(P) = \{l \in \Phi \,|\, N(P, l) > \frac{|P|}{2}\}$. The majoritarian judgment set is not complete when there there is equal support for l and $\neg l$ (i.e., $N(P, l) = N(P, \neg l)$) for an issue l. Additionally, the majoritarian judgment set is

1. Note that our definition of a judgment aggregation framework is as general as the most general variant in which the agenda Φ may include arbitrary formulas instead of only literals (Endriss et al., 2020; see also Endriss et al., 2016). This can be seen from the following construction. For every formula $\varphi_i \in \Phi$, $i = 1, \ldots, m$, let x_i be a fresh variable. Replace the input and output constraints Γ_{in} and Γ_{out} with $\Gamma_{\text{in}} \wedge \bigwedge_{i=1}^{m} (x_i \leftrightarrow \varphi_i)$ and $\Gamma_{\text{out}} \wedge \bigwedge_{i=1}^{m} (x_i \leftrightarrow \varphi_i)$, respectively. The resulting framework with issues as variables $X = \{x_1, \ldots, x_m\}$ is equivalent to the formula-based framework.

not guaranteed to be Γ_{out}-consistent even if $\Gamma_{\text{in}} = \Gamma_{\text{out}}$, as demonstrated by the well-known *discursive dilemma* (Pettit, 2001).

Example 2.3. An office utilizes a network of sensors which, under certain conditions, will automatically activate an air conditioning system. Specifically, the system follows a policy that air conditioning is turned on if and only if temperature is abnormally high *and either* (i) humidity is abnormally high or (ii) air quality is abnormally low (based on particulate matter levels). Let t, h, and q be propositional variables which take the value 1 if temperature, humidity, or air quality levels are undesirable, and 0 otherwise. Additionally let the propositional variable a denote that the air conditioning system should be turned on. The system policy can be expressed as a constraint as the propositional formula $\Gamma_{\text{in}} = a \leftrightarrow \big((t \wedge h) \vee (t \wedge q)\big)$. The following profile P consisting of $J_1, \ldots, J_{11}$ shows the local readings of the 11 sensors in the network.

	t	h	q	a
J_{1-4}	1	0	0	0
J_{5-7}	0	1	1	0
J_{8-10}	1	1	1	1
J_{11}	1	1	0	1
$J_m(P)$	1	1	1	0

This scenario can be represented as the judgment aggregation framework with agenda $\Phi = \{t, h, q, a, \neg t, \neg h, \neg q, \neg a\}$, input/output constraints $\Gamma_{\text{in}} = \Gamma_{\text{out}} = a \leftrightarrow ((t \wedge h) \vee (t \wedge q))$, and profile P. Note that while each judgment set J_i satisfies Γ_{in}, the majoritarian judgment set $J_m(P)$ does not satisfy Γ_{out}. ◊

2.2 Judgment Aggregation Rules

A *judgment aggregation rule* defines a set of optimal (complete and Γ_{out}-consistent) judgment sets which are claimed to best reflect a given profile. For many of the rules considered in this work, optimality is based on some notion of distance from the majoritarian judgment set; the rules are hence designed to respect the majority opinion as much as possible in cases where the majoritarian judgment set is inconsistent with Γ_{out} (Endriss, 2018; Lang et al., 2011; Nehring et al., 2014).

Definition 2.2 (Judgment aggregation rule)**.** A *judgment aggregation rule* is a function that maps a profile P to a nonempty set of collective judgment sets $R(P)$. A rule is *consistent* if $R(P) \subseteq \mathcal{J}(\Phi, \Gamma_{\text{out}})$ for any profile P.

We now give formal definitions of the well-known judgment aggregation rules considered in this work. For the following, let $J_m(P)$ represent the majoritarian judgment set for a profile P.

Definition 2.3 (Condorcet rule)**.** The *Condorcet rule* (Endriss, 2018; Endriss et al., 2020; Lang et al., 2011; Nehring et al., 2014) defines the set of complete and Γ_{out}-consistent judgment sets which maximally agree with the majoritarian judgment set with respect to set inclusion: $\textsc{condorcet}(P)$ consists of those $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ for which there is no $J' \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ with $J \cap J_m(P) \subset J' \cap J_m(P)$.

Definition 2.4 (Slater rule)**.** The *Slater rule* (Endriss, 2018; Lang et al., 2011; Miller & Osherson, 2009; Nehring et al., 2014) defines the set of complete and Γ_{out}-consistent judgment sets which agree with $J_m(P)$ on a maximal number of issues: $\textsc{slater}(P)$ is the set of $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ which maximize $|J \cap J_m(P)|$.

Judgment sets under the Condorcet rule agree with the majority opinion $J_m(P)$ in a subset-maximal sense, i.e., reverting judgment on any single issue to increase agreement with the majority will violate Γ_{out}. On the other hand, under the Slater rule, judgment sets agree with the majority opinion on the highest number of issues. This means that for any profile P, $\textsc{condorcet}(P) \supseteq \textsc{slater}(P)$.

Example 2.4. We will use the judgment aggregation framework from Example 2.3 as a running example for illustrating the various aggregation rules overviewed in this section.

The Condorcet rule selects 3 judgment sets for the profile P: $J^1_{\text{con}} = \{t, h, q, a\}, J^2_{\text{con}} = \{\neg t, h, q, \neg a\}, J^3_{\text{con}} = \{t, \neg h, \neg q, \neg a\}$. Consider for demonstration purposes the judgment set J^3_{con}, where $J^3_{\text{con}} \cap J_m(P) = \{t, \neg a\}$. It is not possible to construct a Γ_{out}-consistent judgment set J' with $J' \cap J_m(P) \supset \{t, \neg a\}$ since (i) if $J' \supset \{t, \neg a, h\}$, then $a \leftrightarrow (t \wedge h)$ does not hold, and (ii) if $J' \supset \{t, \neg a, q\},$, then $a \leftrightarrow (t \wedge q)$ does not hold.

The Slater rule maximizes the cardinality of the agreement with $J_m(P)$. For the $J^i_{\text{con}} \in \textsc{condorcet}(P)$, we have

$$
\begin{aligned}
|J^1_{\text{con}} \cap J_m(P)| &= |\{t, h, q\}| = 3,\\
|J^2_{\text{con}} \cap J_m(P)| &= |\{h, q, \neg a\}| = 3,\\
|J^3_{\text{con}} \cap J_m(P)| &= |\{t, \neg a\}| = 2,
\end{aligned}
$$

and hence $\textsc{slater}(P) = \{J^1_{\text{con}}, J^2_{\text{con}}\}$. ◊

The *Kemeny* (Endriss, 2018; Endriss et al., 2012; Lang et al., 2011; Miller & Osherson, 2009; Nehring et al., 2014; Pigozzi, 2006) and *MaxHamming* (Lang et al., 2011) rules consider variations of agreement across the judgment sets in the profile P. Let $H(J, J') = |J \backslash J'|$ be the *Hamming distance* between the judgment sets J and J'.

Definition 2.5 (Kemeny rule)**.** The *Kemeny rule* maximizes agreement with the profile P by minimizing the total Hamming distance between the collective judgment and each judgment set $J \in P$: $\textsc{kemeny}(P)$ is the set of $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ which maximize $\sum_{l \in J} N(P, l)$ or equivalently minimize $\sum_{J_i \in P} H(J, J_i)$.

In contrast, the *MaxHamming rule* applies the maximum distance between the collective judgment and the judgment sets $J_i \in P$.

Definition 2.6 (MaxHamming rule)**.** The *MaxHamming rule* minimizes the maximum Hamming distance between the collective judgment and the agents' judgment sets $J_i \in P$: $\textsc{maxhamming}(P)$ is the set of $J \in \mathcal{J}(\Phi, \Gamma_{out})$ which minimize $\max_{J_i \in P} H(J, J_i)$.

Example 2.5. Continuing with the running example, the Kemeny rule selects the singleton $\textsc{Kemeny}(P) = \{J_{\text{kem}}\}$ with $J_{\text{kem}} = \{t, h, q, a\}$. The judgment set J_{kem} has total support

$$\sum_{l \in J_{\text{kem}}} N(P, l) = N(P, t) + N(P, h) + N(P, q) + N(P, a) = 8 + 7 + 6 + 4 = 25$$

which is the maximum among all Γ_{out}-consistent judgment sets. On the other hand, the MaxHamming rule returns the following set of six optimal judgment sets, each of which has maximum Hamming distance 3 to the judgment sets $J_i \in P$, which is the minimum among all Γ_{out}-consistent judgment sets:

$$\textsc{maxhamming}(P) = \{\{\neg t, h, \neg q, \neg a\}, \{\neg t, h, q, \neg a\}, \{t, \neg h, \neg q, \neg a\}, \\ \{t, \neg h, q, a\}, \{t, h, \neg q, a\}, \{t, h, q, a\}\}.$$

$\Diamond$

Note that the Kemeny rule is also sometimes defined in relation to the family of *scoring rules* (Dietrich, 2014). These rules assign a score for every judgment-literal pair (J_i, l) (for $J_i \in P$, $l \in \Phi$), where the score is designed to indicate how strongly J_i supports l. The scoring rule then returns those judgment sets J which maximize the total score of the literals $l \in J$ across the judgment sets $J_i \in P$. The Kemeny rule can then be equivalently defined as a scoring rule which defines the score of (J_i, l) as 1 if $l \in J_i$ and 0 otherwise. The *reversal scoring rule* (Dietrich, 2014), on the other hand, weights agenda items by their *reversal score*.

Definition 2.7 (Reversal scoring rule)**.** Let the *reversal score* $R(J, l)$ be the least number of issues on which judgment must be reverted in J in order to obtain a Γ_{out}-consistent judgment set which rejects issue l, i.e., $R(J, l)$ is the minimum Hamming distance $H(J, J')$ among $J' \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ with $l \notin J'$. The *reversal scoring rule* selects $\textsc{revsco}(P)$, the set of judgment sets which maximize the total reversal score $\sum_{J_i \in P} \sum_{l \in J} R(J_i, l)$.

The *ranked agenda* (Endriss & de Haan, 2015; Lang et al., 2011; Porello & Endriss, 2014) and *Leximax* (Everaere et al., 2014; Nehring & Pivato, 2019) *rules* construct a hierarchy for agenda items based on their respective support. Let $L_k^P = \{l \in \Phi \mid N(P, l) = k\}$ be the set of literals in Φ with k support. Note that $\bigcup_{k=\lfloor \frac{|P|}{2} \rfloor + 1}^{n} L_k^P$ is a partition of $J_m(P)$ which groups the issues into sets where each issue has equal support. The ranked agenda rule greedily accepts literals from the L_k^P in decreasing order of support. For instance, $\textsc{RankedAgenda}(P)$ consists of those judgments sets which can be obtained by adding literals of highest support to the collective judgment so long as doing so does not create an inconsistency (where literals of equal support can be added in any order). An equivalent formal definition is based on an ordering of prospective judgment sets $\succ_{\text{RA}}$.

Definition 2.8 (Ranked agenda rule)**.** Given judgment sets J and J', let $J \succ_{\text{RA}} J'$ if there is a k with $\frac{|P|}{2} \leq k \leq |P|$ such that $J \cap L_k^P \supset J' \cap L_k^P$, and for all $j > k$, $J \cap L_j^P = J' \cap L_j^P$. The *ranked agenda rule* selects $\textsc{RankedAgenda}(P)$ containing $\succ_{\text{RA}}$-maximal $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$, i.e., those $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ for which there is no $J' \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ with $J' \succ_{\text{ra}} J$.

The LexiMax rule is based on a lexicographic ordering of judgment sets with respect to agreement with the L_k^P. Specifically, for a judgment set J, the sequence $|J \cap L_{|P|}^P|, |J \cap L_{|P|-1}^P|, \ldots, |J \cap L_{\lceil \frac{|P|}{2} \rceil}^P|$ is used to define a lexicographical order.

Definition 2.9 (LexiMax rule)**.** Let $J \succ_{\text{lex}} J'$ if there is a k with $\frac{|P|}{2} \leq k \leq |P|$ such that $|J \cap L_k^P| > |J' \cap L_k^P|$, and $|J \cap L_j^P| = |J' \cap L_j^P|$ for all $j > k$. The *LexiMax rule* selects

LEXIMAX(P), the set of $\succ_{\text{lex}}$-maximal $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$, i.e., those $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ for which there is no $J' \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ with $J' \succ_{\text{lex}} J$.

Example 2.6. We continue with the running example. Under the ranked agenda and LexiMax rules, the agenda items are divided into preference classes based on their support: $L_8^P = \{t\}$, $L_7^P = \{h, \neg a\}$, $L_6^P = \{q\}$, $L_5^P = \{\neg q\}$, $L_4^P = \{\neg h, a\}$, $L_3^P = \{\neg t\}$. The ranked agenda rule selects RANKEDAGENDA$(P) = \{J^1_{\text{ra}}, J^2_{\text{ra}}\}$ where $J^1_{\text{ra}} = \{t, h, q, a\}$ and $J^2_{\text{ra}} = \{t, \neg h, \neg q, \neg a\}$. Ranked agenda judgment sets can be obtained by selecting issues from the L_k^P in decreasing order of support. The judgment set J^1_{ra} is constructed by selecting t, then h, then q (skipping $\neg a$ to avoid violating $a \leftrightarrow (t \wedge h)$), and finally a. The judgment set J^2_{ra} is similarly obtained, but we select $\neg a$ from L_7^P instead of h. For the LexiMax rule, note that both ranked agenda solutions select one literal each from L_8^P and L_7^P. However, J^1_{ra} contains the literal q while J^2_{ra} does not. Therefore $|J^1_{\text{ra}} \cap L_6^P| = 1$ and $|J^2_{\text{ra}} \cap L_6^P| = 0$, so that $J^1_{\text{ra}} \succ_{\text{lex}} J^2_{\text{ra}}$, and hence LEXIMAX$(P) = \{J^1_{\text{ra}}\}$. $\Diamond$

The *Young* (Lang et al., 2011) and *Dodgson* (Miller & Osherson, 2009) rules are based on making minimal changes to the profile P such that the majoritarian judgment set of the modified profile P' (i.e., $J_m(P')$) is Γ_{out}-consistent. The solution set under these rules consists of complete and Γ_{out}-consistent judgment sets which are each a superset of $J_m(P')$ for some P' which minimizes the number of changes from P.

Definition 2.10 (Young rule)**.** The *Young rule* considers the majoritarian judgment sets of profiles $P' \subseteq P$, i.e., profiles obtained by removing judgment sets from P. First select the set of all $P' \subseteq P$ where $J_m(P')$ is Γ_{out}-consistent and $|P'|$ is maximized. Then YOUNG(P) is the set of $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ for which $J \supseteq J_m(P')$ for some P'.

Definition 2.11 (Dodgson rule)**.** The *Dodgson rule* considers the majoritarian judgment sets of Γ_{in}-consistent profiles obtained by reverting the opinions of agents on individual issues (the subset of issues over which an agent's opinion is reverted is not necessarily the same for all agents). Importantly, each judgment set J' in the modified profile P' must be Γ_{in}-consistent. Let $\mathcal{P}(\Phi, \Gamma)$ be the set of all Γ-consistent profiles over Φ. Now first select all profiles $P' \in \mathcal{P}(\Phi, \Gamma_{\text{in}})$ with $|P'| = |P|$ for which $J_m(P')$ is Γ_{out}-consistent and $\sum_{i=1}^{n} H(J_i, J_i')$ is minimized. Then DODGSON(P) is the set of $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ for which $J \supseteq J_m(P')$ for some P'.

Example 2.7. Continuing with the running example, the Young rule returns the set YOUNG$(P) = \{J^1_{\text{yng}}, J^2_{\text{yng}}, J^3_{\text{yng}}\}$ where $J^1_{\text{yng}} = \{t, h, q, a\}$, $J^2_{\text{yng}} = \{t, h, \neg q, a\}$, and $J^3_{\text{yng}} = \{t, \neg h, \neg q, \neg a\}$. Each solution is obtained as the superset of the majoritarian judgment set of a modified profile which removes 3 judgment sets from P. To obtain J^1_{yng}, for instance, we can remove J_{1-3} from P to obtain a modified profile with majoritarian judgment set $\{t, h, q\}$ (note that a and $\neg a$ have equal support in this modified profile). The remaining Young solutions are obtained similarly. There is no profile obtained by removing 2 or fewer judgment sets from P which has a Γ_{out}-consistent majoritarian judgment set. The Dodgson rule returns the set DODGSON$(P) = \{J^1_{\text{dod}}, J^2_{\text{dod}}\}$ where $J^1_{\text{dod}} = \{\neg t, h, q, \neg a\}$ and $J^2_{\text{dod}} = \{t, \neg h, \neg q, \neg a\}$, The collective judgment set J^1_{dod}, for instance, is obtained as the majoritarian judgment set of, e.g., the profile P'

	t	h	q	a
J_{1-3}	0	0	0	0
J_4	1	0	0	0
J_{5-7}	0	1	1	0
J_{8-10}	1	1	1	1
J_{11}	1	1	0	1
$J_m(P)$	0	1	1	0

in which J_{1-3} have been modified, each reverting judgment from t to $\neg t$. Note that the modified J_{1-3} satisfy Γ_{in}. This constitutes 3 total modifications to the profile, the minimum number of modifications needed in order to make $J_m(P')$ satisfy Γ_{out}. ◊

2.3 Outcome Determination

Our main focus is on *outcome determination* (Endriss et al., 2020) in judgment aggregation.

Definition 2.12 (Outcome determination)**.** For a given judgment aggregation framework $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$, a judgment aggregation rule R, and a set of literals $L \subset \Phi$, the *outcome determination* problem consists of finding a judgment set $J \in R(P)$ with $L \subseteq J$, or determining that such a judgment set does not exist.

We note that several variants of outcome determination have been proposed and studied (Endriss et al., 2020). The definition for outcome determination we use corresponds to the general search variant outlined by Endriss et al. (2020)[2].

Example 2.8. Recall the judgment aggregation framework from Example 2.3, for which we have $\textsc{slater}(P) = \{\{t, h, q, a\}, \{\neg t, h, q, \neg a\}\}$ (see Example 2.4). Consider outcome determination for this framework under the Slater rule with outcome $\{a\}$. The outcome is accepted since the judgment set $\{t, h, q, a\} \in \textsc{slater}(P)$ contains the literal a. In contrast, the outcome $\{\neg t, a\}$ is rejected since no Slater-optimal judgment set contains both $\neg t$ and a. ◊

Outcome determination has been shown to be computationally hard for various aggregation rules, including the rules we consider in this work. Towards recalling the known complexity results (see, e.g., Arora & Barak, 2009), recall that the complexity class $\Delta_2^{\text{P}} = \text{P}^{\text{NP}}$ consists of decision problems decidable in polynomial time given access to an NP-oracle for some NP-hard problem; the class $\Sigma_2^{\text{P}} = \text{NP}^{\text{NP}}$ consists of problems decidable in nondeterministic polynomial time with access to an NP-oracle; and the class Θ_2^{P} consists of problems decidable in polynomial time with a logarithmic number of calls to an NP-oracle.

Proposition 2.1. (Endriss et al., 2020). Outcome determination is Θ_2^{P}-complete for the Kemeny, Slater, MaxHamming, Young, Dodgson, and reversal scoring rules, Δ_2^{P}-complete for the Leximax rule, and Σ_2^{P}-complete for the Condorcet and ranked agenda rules.

2. Our definition of outcome determination captures the general variant defined by Endriss et al. (2020), where in addition to $L \subseteq \Phi$ sets $L_1, \ldots, L_u \subseteq \Phi$ are given as input, and the task is to decide if there is $J \in R(P)$ with $L \subseteq J$ and $L_i \not\subseteq J$ for each $i = 1, \ldots, u$. This can be seen from the following construction. For each $i = 1, \ldots, u$, let o_i be a fresh variable, and consider $X \cup \{o_i \mid i = 1, \ldots, u\}$ as the set of issues. Finally, we define $\Gamma_{\text{out}} \wedge \bigwedge_{i=1}^{u} (o_i \leftrightarrow \bigwedge_{l \in L_i} l)$ as the output constraint, and the set $L \cup \{\neg o_1, \ldots, \neg o_u\}$ as the outcome.

We note that preference aggregation (Dietrich & List, 2007; Endriss, 2016; List, 2012) is a widely-studied variant of judgment aggregation in which judgments are orderings of the issues. A preference aggregation framework (Endriss, 2016) is a tuple (C, V) where $C = \{1, \ldots, m\}$ is the set of *candidates* (alternatives) and $V = (\succ_1, \ldots, \succ_n)$ is the *voter* profile, where the preference rankings $\succ_i$ are linear orders on C. Under the common assumption (Endriss, 2016; Zwicker, 2016) that voters express strict linear orders on the candidates,[3] a preference aggregation framework can represented as a judgment aggregation framework by representing orderings as a set of pairwise comparisons between the candidates. The *winner determination* task asks whether a given candidate is a possible winner under a given voting rule (Conitzer, 2006; Hemaspaandra et al., 1997, 2005; Rothe et al., 2003). As in general in judgment aggregation, votes are aggregated via a *voting rule* (Zwicker, 2016), and aggregating according to the majority opinion in preference aggregation does not necessarily result in a logically consistent judgment set due to the possibility of cyclic preferences (Zwicker, 2016).[4]

Definition 2.13. Given a preference aggregation framework (C, V), a voting rule R, and a winner $x \in C$, the *winner determination* task is to find an order $\succ \in R(V)$ where $x \succ x'$ for all $x' \in C \setminus \{x\}$, or determine that such an order does not exist.

3. SAT, PrefSAT and MaxSAT

In this section, we provide an overview of the propositional satisfiability problem (SAT), SAT with preferences (PrefSAT), and maximum satisfiability (MaxSAT). These serve as the basis for our SAT-based algorithms detailed in Sections 4 and 5.

3.1 Propositional Satisfiability

For a propositional variable x there are two *literals*, x and $\neg x$. A *clause* C is a disjunction ($\vee$) of literals, and a formula in *conjunctive normal form* (CNF) F is a conjunction ($\wedge$) of clauses (Prestwich, 2021). Any propositional formula can be converted in linear time to an equivalent CNF formula whose size grows linearly using the standard *Tseitin transformation* (Tseitin, 1983). When convenient, we present encodings using propositional formulas which are not in CNF as proxies for sets of clauses. We denote by $\textsc{var}(F)$ and $\textsc{lit}(F)$ the sets of variables and literals appearing in F, respectively. A *truth assignment* is a function $\tau : \textsc{var}(F) \to \{0, 1\}$ which assigns each variable in a CNF formula to 0 (false) or 1 (true). Truth assignments are additionally extended logically for literals, clauses, and formulas via $\tau(\neg x) = 1 - x$, $\tau(C) = \max\{\tau(l) \mid l \in C\}$, and $\tau(F) = \min\{\tau(C) \mid C \in F\}$. When convenient, we interchangeably represent a truth assignment τ as the set of non-contradictory literals $\{l \in \textsc{lit}(F) \mid \tau(l) = 1\}$. If there is a truth assignment that satisfies a propositional formula F, the formula is *satisfiable* and otherwise *unsatisfiable*. The *propositional satisfiability problem* (SAT) asks to decide whether a given CNF formula F (a SAT instance) is satisfiable.

3. That is, for all $x, y \in C$, $x \succ_i y$ or $y \succ_i x$ for each $\succ_i \in V$.
4. While the term "voting rule" is a general term that has been used in a variety of contexts with various types of "ballots" (Zwicker, 2016), we use the term to refer specifically to a function which maps a voter profile V to a nonempty set of partial orders $R(V)$.

A SAT solver is an implementation of an algorithm which determines the satisfiability of a given CNF formula. Despite the fact that SAT is a well-known NP-complete problem (Cook, 1971), modern SAT solvers are capable of determining the satisfiability of CNF formulas arising from real-world settings containing up to millions of variables and clauses (Marques-Silva et al., 2021). Most SAT solvers implement the conflict-driven clause learning (CDCL) algorithm (Marques-Silva et al., 2021), constituting a complete search procedure for SAT which readily also outputs a satisfying truth assignment at termination when invoked on a satisfiable formula.

The SAT-based judgment aggregation algorithms we develop in Section 5 involve solving a series of related SAT instances. In order to solve such sequences of instances more efficiently, modern SAT solvers also provide an assumptions interface (Eén & Sörensson, 2003; Marques-Silva et al., 2021). A set of assumptions $A \subset \text{LIT}(F)$, where F is a given CNF formula, is an additional parameter for a SAT solver call, where the assumptions are required to be satisfied in any satisfying assignment τ. Specifically, a SAT solver call under assumptions A will either return a satisfying assignment $\tau \supseteq A$, or report unsatisfiability under the given assumptions. In the the latter case the solver also reports an *unsatisfiable core* $U \subseteq A$ as a reason for unsatisfiability (Marques-Silva et al., 2021), expressing that there is no satisfying assignment for F that would satisfy each literal in U (i.e., $F \cup \bigwedge_{l \in U}(l)$ is unsatisfiable).

3.2 SAT with Preferences

We will also employ SAT with preferences (PrefSAT; Dodaro & Previti, 2019; Rosa et al., 2010), an extension of SAT that allows enforcing a weak preference ordering over the literals of input CNF formulas. In particular, it turns out that PrefSAT is a suitable formalism as a basis for developing outcome determination algorithms under the Condorcet and Ranked agenda judgment aggregation rules. In the PrefSAT problem, in addition to a CNF formula F, we are given as input a weight function which expresses *the preference level* of each literal in F. A higher weight indicates that a literal is more preferred than a literal with lower weight; that is, the literals are partitioned into preference classes based on their weights. PrefSAT solutions are truth assignments of F which additionally satisfy maximal subsets of the preference classes in decreasing order. More formally, for a given input CNF formula F, a weight function $w : \text{LIT}(F) \to \mathbb{Z}^+$ and a set $L \subseteq \text{LIT}(F)$, let Λ^L_k denote the preference class of literals in L with weight k, i.e., $\Lambda^L_k = \{l \in L \mid w(l) = k\}$.

Definition 3.1 (SAT with preferences)**.** Let F be a CNF formula and $w : \text{LIT}(F) \to \mathbb{Z}^+$ a weight function. A truth assignment τ' is *preferred* to another truth assignment τ, denoted $\tau' > \tau$, if there exists an integer $k > 0$ with $\Lambda^{\tau'}_k \supset \Lambda^{\tau}_k$ and for all $i > k$ it holds that $\Lambda^{\tau'}_i = \Lambda^{\tau}_i$. Given F and w, the *SAT with preferences* task is to find a truth assignment τ that satisfies F such that there is no truth assignment τ' with $\tau' > \tau$ that satisfies F.

Example 3.1. Consider the PrefSAT instance consisting of the CNF formula $F = (x_1 \vee \neg x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3)$ and the weight function w that maps the positive literals x_1, x_2, x_3 to 1 and the negative literals $\neg x_1, \neg x_2, \neg x_3$ to 0. The truth assignments $\tau_1 = \{x_1, \neg x_2, x_3\}$, $\tau_2 = \{\neg x_1, \neg x_2, x_3\}$, and $\tau_3 = \{\neg x_1, \neg x_2, \neg x_3\}$ satisfy F. Under w, $\tau_1 > \tau_2 > \tau_3$ since for the highest preference class, $\Lambda^{\tau_1}_1 = \{x_1, x_3\} \supset \Lambda^{\tau_2}_1 = \{x_3\} \supset \Lambda^{\tau_3}_1 = \{\}$. Hence τ_1 is a unique solution to the PrefSAT instance. $\Diamond$

A PrefSAT solver (Dodaro & Previti, 2019) extends the CDCL algorithm to find a solution which respects the preferences defined by the weight function. In practice, this is relatively straightforwardly achieved by enforcing the decision heuristics of a CDCL SAT solver to follow the order imposed by the given preferences, assigning variables to their preferred values.

3.3 Maximum Satisfiability

We also make use of *Maximum satisfiability* (MaxSAT; Bacchus et al., 2021; Li & Manyà, 2021) solvers. MaxSAT is the the optimization extension of SAT, allowing for finding optimal solutions to CNF encodings under linear objective functions over propositional (binary) variables. A (weighted partial) MaxSAT instance is a tuple $(F_{\text{hard}}, F_{\text{soft}}, w)$, where F_{hard} is a set of *hard clauses*, F_{soft} is a set of *soft clauses*, and $w : F_{\text{soft}} \to \mathbb{N}_+$ is a weight function which associates a non-negative weight to each clause in F_{soft}. A *solution* is an assignment τ which satisfies each of the hard clauses, and its cost $c(\tau)$ is the sum of the weights of the soft clauses it does not satisfy, i.e., $\sum_{C \in F_{\text{soft}}} w(C) \cdot (1 - \tau(C))$. In MaxSAT, the goal is to find an *optimal solution*, that is, a solution τ which minimizes $c(\tau)$. Recent advances in MaxSAT solvers (Bacchus et al., 2021) have made MaxSAT a competitive alternative to e.g., modern classical integer programming techniques in solving NP-hard combinatorial optimization problems.

4. MaxSAT Approaches to Judgment Aggregation

In this section, we describe how judgment aggregation under the Kemeny, Slater, MaxHamming, Young, and Dodgson rules can be encoded as MaxSAT. We employ the encodings within a generic algorithm for the outcome determination task covering all of the rules.

4.1 Overview

We use MaxSAT as a suitable declarative paradigm to encode the Kemeny, Slater, MaxHamming, Young, and Dodgson rules. Recall that for a judgment aggregation framework $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$, the issues are a set of propositional variables $X_\Phi = \{x_1, \ldots, x_m\}$. Note also that for a consistent judgment aggregation rule R, any judgment set $J \in R(P)$ can be viewed as a truth assignment over X_Φ. It is therefore natural to directly use the x_i's as variables in a MaxSAT encoding. Specifically, we construct a MaxSAT instance $F_R(P)$ with variables $\textsc{var}(F_R(P)) \supseteq X_\Phi$, where each optimal solution projected to the variables in X_Φ corresponds one-to-one with a judgment set $J \in R(P)$.

Before detailing the encodings for each of the rules, we present a generic algorithm outlined as Algorithm 1 for outcome determination, making use of the encodings to cover each of the individual rules. Let $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$ be a given judgment aggregation framework, $L \subseteq \Phi$ an input subset of the agenda, and $R \in \{\textsc{Kemeny}, \textsc{Slater}, \textsc{MaxHamming}, \textsc{Young}, \textsc{Dodgson}\}$ a judgment aggregation rule. First, the judgment aggregation rule R is encoded as MaxSAT (line 1), i.e., we construct a MaxSAT instance $F_R(P)$ whose optimal solutions τ are in a one-to-one correspondence with judgment sets $J \in R(P)$ via $\tau \cap \Phi = J$.[5] Then,

5. The intersection accounts for the fact that $F_R(P)$ may contain auxiliary variables outside of X_Φ, depending on the encoding of the individual rules.

Algorithm 1 Generic MaxSAT-based algorithm for outcome determination. **Input**: JA framework $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$, $L \subseteq \Phi$, rule $R \in \{\text{Kemeny}, \text{Slater}, \text{MaxHamming}, \text{Young}, \text{Dodgson}\}$.

1: Construct MaxSAT encoding $F_R(P) = (F_{\text{hard}}, F_{\text{soft}}, w)$
2: $(c^*, \tau^*) \leftarrow \text{MaxSAT}(F_R(P))$
3: $F_{R,L}(P) \leftarrow (F_{\text{hard}} \wedge \bigwedge_{l \in L} l, F_{\text{soft}}, w)$
4: $(c^L, \tau^L) \leftarrow \text{MaxSAT}(F_{R,L}(P))$
5: **if** $c^L > c^*$ **then return false else return** $\tau^L \cap \Phi$

we compute the optimal cost c^* of the MaxSAT instance $F_R(P)$ by invoking a MaxSAT solver (line 2). We then construct a MaxSAT instance $F_{R,L}(P)$ which differs from $F_R(P)$ by additional hard clauses $\bigwedge_{l \in L} l$ enforcing the desired outcome (line 3). A second MaxSAT solver call is then made on the MaxSAT instance $F_{R,L}(P)$ (line 4). If there are no solutions or if for the optimal cost c^L of this instance we have $c^L > c^*$, there is no $J \in R(P)$ with $L \subseteq J$, and hence we return **false**. Otherwise, i.e., if $c^L = c^*$, we return $\tau \cap \Phi$, where τ is the optimal MaxSAT solution obtained from the second MaxSAT solver invocation (line 5).

Since the second MaxSAT instance $F_{R,L}(P)$ in Algorithm 1 (line 3) differs from $F_R(P)$ (line 1) only by additional hard clauses, Algorithm 1 can be implemented using a single instantiation of an *incremental* MaxSAT solver (Niskanen et al., 2022) which supports the addition of hard clauses. In particular, this means that the state of the MaxSAT solver is retained, allowing the solver to make use of information learned during the first call.[6] Assuming that $F_R(P)$ is a MaxSAT encoding of a JA rule R of interest, Algorithm 1 is correct for outcome determination, stated as follows.

Proposition 4.1. Let $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$ be a given judgment aggregation framework, $L \subseteq \Phi$, and $R \in \{\text{Kemeny}, \text{Slater}, \text{MaxHamming}, \text{Young}, \text{Dodgson}\}$. Suppose $F_R(P)$ is a MaxSAT instance such that every optimal solution τ corresponds to a collective judgment set $J \in R(P)$ via $\tau \cap \Phi = J$. If there is a judgment set $J \in R(P)$ with $L \subseteq J$, Algorithm 1 returns J and otherwise **false**.

In the rest of this section we describe in detail the MaxSAT encoding F_R for each of the individual judgment aggregation rules R. For each of the MaxSAT encodings, we employ X_Φ directly as MaxSAT variables, and initialize the hard clauses F_{hard} as the output constraint Γ_{out} to ensure that any satisfying assignment τ to F_{hard} corresponds to a complete and Γ_{out}-consistent judgment set $J = \tau \cap \Phi$. To encode the MaxHamming, Young, and Dodgson rules, we additionally make use of cardinality constraints of the form $\sum_{l \in L} l \circ k$ where L is a set of literals, $\circ \in \{\leq, \geq\}$, and k is an integer. In the summation, a negative literal $\neg x \in L$ is used as a shorthand for $(1 - x)$. Any such cardinality constraint can be readily encoded as a CNF formula, which we denote by $\text{CNF}(\sum_{l \in L} l \circ k)$, using one of various CNF encodings of cardinality constraints (Bailleux & Boufkhad, 2003; Eén & Sörensson, 2006; Martins et al., 2014; Sinz, 2005).

6. Alternatively, a SAT solver can be used for the second call by querying for a solution that has the same cost as the solution from the initial MaxSAT call.

4.2 Slater and Kemeny Rules

The Slater rule is encoded by introducing a soft unit clause (l) with unit weight for each $l \in J_m(P)$. The Kemeny rule adds the same clauses, but the weight of a clause (l) is given instead by $N(P, l) - N(P, \neg l)$. Note that this is equivalent to introducing a soft clause (l) with weight $N(P, l)$ for each $l \in \Phi$ as a direct representation of the objective function of the Kemeny rule, but the latter needlessly inflates the number of clauses (and their weights). Summarizing, let $F_{\text{KEMENY}}(P)$ and $F_{\text{SLATER}}(P)$ be MaxSAT instances with hard clauses Γ_{out} and soft clauses (l) for each $l \in J_{\text{m}}(P)$, with the distinction that $w_{\text{KEMENY}}(l) = N(P, l) - N(P, \neg l)$ and $w_{\text{SLATER}}(l) = 1$.

Example 4.1 (Encodings of the Slater and Kemeny rules)**.** We illustrate the MaxSAT encodings using as a running example the judgment aggregation framework $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$ with issues $\{x_1, x_2, x_3\}$, constraints $\Gamma_{\text{out}} = \Gamma_{\text{in}} = x_3 \leftrightarrow (x_1 \wedge x_2)$, and the profile

	x_1	x_2	x_3
J_1	1	1	1
J_2	1	0	0
J_3	0	1	0
$J_m(P)$	1	1	0

with majoritarian judgment set is $J_m(P) = \{x_1, x_2, \neg x_3\}$. Hence, the soft clauses for both $F_{\text{SLATER}}(P)$ and $F_{\text{KEMENY}}(P)$ are (x_1), (x_2), and $(\neg x_3)$. The weights of the soft clauses under the Slater rule are all 1. Under the Kemeny rule the weights are given by

$$\begin{aligned} w_{\text{KEMENY}}[(x_1)] &= N(P, x_1) - N(P, \neg x_1) = 2 - 1 = 1, \\ w_{\text{KEMENY}}[(x_2)] &= N(P, x_2) - N(P, \neg x_2) = 2 - 1 = 1, \text{and} \\ w_{\text{KEMENY}}[(\neg x_3)] &= N(P, \neg x_3) - N(P, x_3) = 2 - 1 = 1. \end{aligned}$$

$\Diamond$

The correctness of the MaxSAT encodings can be stated as follows.

Proposition 4.2. Let $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$ be a given judgment aggregation framework, and $R \in \{\text{SLATER}, \text{KEMENY}\}$. If $J \in R(P)$, then there exists an optimal solution τ to the MaxSAT instance $F_R(P)$ with $\tau \cap \Phi = J$. Vice versa, if τ is an optimal solution to $F_R(P)$, then there is a judgment set $J \in R(P)$ with $J = \tau \cap \Phi$.

4.3 MaxHamming Rule

The MaxSAT encodings for the MaxHamming, Young, and Dodgson rules are more intricate.

Towards an encoding for the MaxHamming rule, note that the Hamming distance between a judgment set J_i and the output is given by $\sum_{l \in J_i} \neg l$. It follows that the cardinality constraint $\sum_{l \in J_i} \neg l \geq k$ models the condition that the Hamming distance between J_i and the output is at least k. The CNF formula $\text{CNF}(\sum_{l \in J_i} \neg l \geq k)$ enforces this constraint. Further, the condition "the maximum Hamming distance between the input judgment sets and the output J is at least k" can be modeled as the following disjunction over these cardinality constraints for all $J_i \in P$:

$$\bigvee_{i=1}^{|P|} \text{CNF}\left(\sum_{l \in J_i} \neg l \geq k\right).$$

We declare variables p_k for each $k = 1, \ldots, |X_\Phi|$ and constrain them to true if this condition holds. We enforce the following hard clauses for all $k = 1, \ldots, |X_\Phi|$:

$$\textsc{disagreement}_k(P) = \left(\bigvee_{i=1}^{|P|} \text{CNF}\left(\sum_{l \in J_i} \neg l \geq k\right)\right) \rightarrow p_k.$$

These clauses enforce for each $k = 1, \ldots, |X_\Phi|$ that $p_k = 1$ if there is an agent who disagrees with the output on at least k issues. Note that $p_{k+1} \rightarrow p_k$ for any $1 \leq k \leq |X_\Phi| - 1$. In other words: if some agent disagrees with the output on $k+1$ issues, the agent necessarily disagree on k issues.[7]

Our task under MaxHamming is to minimize the maximum disagreement among the agents. We encode this objective directly with the soft clauses $(\neg p_k)$ for all $k = 1, \ldots, |X_\Phi|$ with unit weights. Summarizing, $F_{\textsc{MaxHamming}}(P)$ has hard clauses

$$\Gamma_{\text{out}} \wedge \bigwedge_{k=1}^{|X_\Phi|} \textsc{disagreement}_k(P) \wedge \bigwedge_{k=1}^{|X_\Phi|-1} (p_{k+1} \rightarrow p_k)$$

and soft clauses $(\neg p_k)$ with $w(\neg p_k) = 1$ for each $k = 1, \ldots, |X_\Phi|$.

Example 4.2 (Encoding of the MaxHamming rule)**.** Continuing with the running example 4.1, for the MaxHamming encoding we declare the variables p_1, p_2, p_3 and enforce the following hard clauses for $k = 1, 2, 3$:

$$\begin{aligned}\textsc{disagreement}_k(P) =& (\neg x_1 + \neg x_2 + \neg x_3 \geq k) \vee (\neg x_1 + x_2 + x_3 \geq k) \vee \\ & (x_1 + \neg x_2 + x_3 \geq k) \rightarrow p_k.\end{aligned}$$

We additionally enforce the hard clauses $p_3 \rightarrow p_2$ and $p_2 \rightarrow p_1$, and encode the objective with the soft clauses $(\neg p_1)$, $(\neg p_2)$, and $(\neg p_3)$, each with weight 1. ◊

A solution τ to $F_{\textsc{MaxHamming}}(P)$ yields an output-consistent judgment set $J = \tau \cap \Phi$. Furthermore, the cardinality constraints encoded as hard clauses in $F_{\textsc{MaxHamming}}(P)$ ensure that if $p_k = 0$ for some $k = 1, \ldots, |X_\Phi|$, then the Hamming distance of J to any $J_i \in P$ is at most k. The soft clauses ensure that the bound k is minimized. The correctness of the MaxSAT encoding follows from these observations.

Proposition 4.3. Let $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$ be a given judgment aggregation framework. If $J \in \textsc{MaxHamming}(P)$, then there exists an optimal solution τ to the MaxSAT instance $F_{\textsc{MaxHamming}}(P)$ with $\tau \cap \Phi = J$. Vice versa, if τ is an optimal solution to $F_{\textsc{MaxHamming}}(P)$, then there is a judgment set $J \in \textsc{MaxHamming}(P)$ with $J = \tau \cap \Phi$.

7. We note that the binary clauses $p_{k+1} \rightarrow p_k$ are redundant, i.e., they are not required for the correctness of the encoding. On the other hand, binary clauses are efficiently handled in SAT solvers for propagation, and do not typically result in runtime degradation.

4.4 Young Rule

The Young rule considers a modified profile defined by a set of judgment sets which are removed from the profile P. Towards a MaxSAT encoding, we introduce variables y_i for each $i = 1, \ldots, n$ with the interpretation that $\tau(y_i) = 1$ if and only if J_i is *included* in the modified profile. With these variables, a solution τ to the encoding specifies a profile $P' = \{J_i \mid \tau(y_i) = 1\}$. What remains is to specify constraints for each variable $x_j \in X_\Phi$ so that they are assigned according to the majoritarian judgment set associated with the modified profile defined by the y_i. Specifically, an issue x_j is necessarily included in the collective judgment if the number of judgment sets included in the modified profile which support x_j constitute a strict majority of the included judgment sets.

The number of judgment sets which constitutes a majority of the modified profile depends on the number of agents included. Suppose the number of judgment sets in the modified profile is k. This is modeled by the cardinality constraint $\sum_{i=1}^{|P|} y_i = k$. An issue $x_j \in X_\Phi$ must be included in the collective judgment if it is included in a strict majority ($\lfloor k/2 \rfloor + 1$) of the judgment sets which are included in the modified profile. This is modeled by the cardinality constraint

$$\sum_{\substack{i=1\ldots|P| \\ x_j \in J_i}} y_i \geq \lfloor k/2 \rfloor + 1.$$

Altogether we have

$$\left(\sum_{i=1}^{|P|} y_i = k \wedge \sum_{\substack{i=1,\ldots,|P| \\ x_j \in J_i}} y_i \geq \lfloor k/2 \rfloor + 1 \right) \rightarrow x_j.$$

We enforce these constraints for all values of $k \in 1, \ldots, |P|$. Note that if the number of judgment sets in the modified profile is less than k, i.e., $\sum_{i=1}^{n} y_i < k$, then $\lfloor k/2 \rfloor + 1$ judgment sets still constitute more than half of the modified profile, so x_j must be included in the collective judgment if it is included in $\lfloor k/2 \rfloor + 1$ of the judgment sets of the modified profile. Hence we can replace the equality constraint by an *at-most* constraint, which has the benefit of being more compact when represented as a CNF formula. Specifically, we enforce the hard clauses $\textsc{support}_j^+(P)$ for all $j = 1, \ldots, |X_\Phi|$, where $\textsc{support}_j^+(P) =$

$$\bigwedge_{k=1}^{|P|} \left(\left(\text{CNF}\left(\sum_{i=1}^{|P|} y_i \leq k \right) \wedge \text{CNF}\left(\sum_{\substack{i=1,\ldots,|P| \\ x_j \in J_i}} y_i \geq \lfloor k/2 \rfloor + 1 \right) \right) \rightarrow x_j \right).$$

Further, we enforce a similar set of constraints for all $j = 1, \ldots, |X_\Phi|$ to require that if a strict majority of the modified profile supports $\neg x_j$, we must have $x_j = 0$ in the collective judgment set. In CNF, these constraints are enforced with the hard clauses $\textsc{support}_j^-(P) =$

$$\bigwedge_{k=1}^{|P|} \left(\left(\text{CNF}\left(\sum_{i=1}^{|P|} y_i \leq k \right) \wedge \text{CNF}\left(\sum_{\substack{i=1,\ldots,|P| \\ \neg x_j \in J_i}} y_i \geq \lfloor k/2 \rfloor + 1 \right) \right) \rightarrow \neg x_j \right).$$

Note that x_j (or $\neg x_j$) is constrained only when the judgment sets that accept it constitute a strict majority of the modified profile P'. If $N(P', x_j) = N(P', \neg x_j)$, we have $x_j \notin J_m(P')$ and $\neg x_j \notin J_m(P')$, and hence either x_j or $\neg x_j$ can be included in a candidate solution, which is consistent with the fact that the Young rule considers *supersets* of the majority judgment set of the modified profile.

In terms of the objective, the Young rule identifies solutions which minimize the number of removed agents. We encode this objective with the soft clauses (y_i) with weight 1 for each $i = 1, \ldots, |P|$.

To summarize, $F_{\text{YOUNG}}(P)$ contains hard clauses

$$\Gamma_{\text{out}} \wedge \bigwedge_{j=1}^{|X_\Phi|} (\text{SUPPORT}_j^+(P) \wedge \text{SUPPORT}_j^-(P))$$

and soft clauses (y_i) with $w(y_i) = 1$ for each $i = 1, \ldots, |P|$.

Example 4.3 (Encoding of the Young rule)**.** For the running example, we declare the variables y_1, y_2, y_3 which correspond to the inclusion of the judgment sets J_1, J_2, J_3 in the modified profile, respectively. We enforce the hard clauses $\text{SUPPORT}_j^+(P)$ for $j = 1, 2, 3$ as

$$\begin{aligned}
\text{SUPPORT}_1^+(P) &= \bigwedge_{k=1}^{3} [((y_1 + y_2 + y_3 \leq k) \wedge (y_1 + y_2 \geq \lfloor k/2 \rfloor + 1)) \rightarrow x_1] , \\
\text{SUPPORT}_2^+(P) &= \bigwedge_{k=1}^{3} [((y_1 + y_2 + y_3 \leq k) \wedge (y_1 + y_3 \geq \lfloor k/2 \rfloor + 1)) \rightarrow x_2] , \\
\text{SUPPORT}_3^+(P) &= \bigwedge_{k=1}^{3} [((y_1 + y_2 + y_3 \leq k) \wedge (y_1 \geq \lfloor k/2 \rfloor + 1)) \rightarrow x_3] .
\end{aligned}$$

We similarly enforce $\text{SUPPORT}_j^-(P)$ for $j = 1, 2, 3$ as

$$\begin{aligned}
\text{SUPPORT}_1^-(P) &= \bigwedge_{k=1}^{3} [((y_1 + y_2 + y_3 \leq k) \wedge (y_3 \geq \lfloor k/2 \rfloor + 1)) \rightarrow \neg x_1] , \\
\text{SUPPORT}_2^-(P) &= \bigwedge_{k=1}^{3} [((y_1 + y_2 + y_3 \leq k) \wedge (y_2 \geq \lfloor k/2 \rfloor + 1)) \rightarrow \neg x_2] , \\
\text{SUPPORT}_3^-(P) &= \bigwedge_{k=1}^{3} [((y_1 + y_2 + y_3 \leq k) \wedge (y_2 + y_3 \geq \lfloor k/2 \rfloor + 1)) \rightarrow \neg x_3] .
\end{aligned}$$

Finally, we include (y_1), (y_2), and (y_3) as soft clauses, each with weight 1. ◊

The correctness of the MaxSAT encoding is established by noting that any solution τ to $F_{\textsc{Young}}(P)$ results in a Γ_{out}-consistent judgment set $J = \tau \cap \Phi$, and J agrees with the strict majority of a modified profile from which a minimum number of agents have been removed (those for which $\tau(y_i) = 0$).

Proposition 4.4. Let $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$ be a given judgment aggregation framework. If $J \in \textsc{Young}(P)$, then there exists an optimal solution τ to the MaxSAT instance $F_{\textsc{Young}}(P)$ with $\tau \cap \Phi = J$. Vice versa, if τ is an optimal solution to $F_{\textsc{Young}}(P)$, then there is a judgment set $J \in \textsc{Young}(P)$ with $J = \tau \cap \Phi$.

4.5 Dodgson Rule

The Dodgson rule is related to the Young rule in that it also considers a modified profile. This is reflected in similarities between the MaxSAT encodings of these two rules. However, instead of considering profiles with judgment sets removed (as in Young), for Dodgson we consider profiles which revert the opinions of the agents on individual issues. We declare variables x_{ij} for each $i = 1, \ldots, |P|$ and $j = 1, \ldots, |X_\Phi|$, with the interpretation that $\tau(x_{ij}) = 1$ if and only if the ith judgment set supports issue $x_j \in X_\Phi$ in the modified profile. A solution τ to the encoding therefore specifies a modified profile P' which contains judgment sets $J_i = \{x_{i1}, \ldots, x_{i|X_\Phi|}\}$.

As for the Young rule, we include hard clauses which enforce that any valid assignment τ assigns the $x_i \in X_\Phi$ according to the majoritarian judgment set of the modified profile specified by the x_{ij}. The number of supporters of issue x_j in the modified profile is given by $\sum_{i=1}^{|P|} x_{ij}$. If the supporters constitute a strict majority, we constrain $x_j = 1$ by

$$\left(\sum_{i=1}^{|P|} x_{ij} \geq \left\lfloor \frac{|P|}{2} \right\rfloor + 1 \right) \rightarrow x_j.$$

A similar constraint on $\neg x_j$ enforces the condition that if $\neg x_j$ is supported by a strict majority of the modified profile, it is then included in the collective judgment set. As hard clauses, this constraint enforced for every $j = 1, \ldots, |X_\Phi|$ as

$$\begin{aligned} \textsc{support}_j(P) = &\left(\text{CNF}\left(\sum_{i=1}^{|P|} x_{ij} \geq \left\lfloor \frac{|P|}{2} \right\rfloor + 1 \right) \rightarrow x_j \right) \wedge \\ &\left(\text{CNF}\left(\sum_{i=1}^{|P|} \neg x_{ij} \geq \left\lfloor \frac{|P|}{2} \right\rfloor + 1 \right) \rightarrow \neg x_j \right). \end{aligned}$$

Note that, as in the case of the Young rule, x_j is not constrained if the x_{ij} designate a modified profile P' where $N(P', x_j) = N(P', \neg x_j)$, since neither x_j nor $\neg x_j$ are part of the majoritarian judgment set.

Additionally, the Dodgson rule specifies that each judgment set in the modified profile must be consistent with respect to the input constraint Γ_{in}. To enforce this, for each judgment set $J_i \in P$ we include the hard clauses $\Gamma_{\text{in}}^i = \Gamma_{\text{in}}[x_j \mapsto x_{ij} \mid j = 1, \ldots, m]$; that is, clauses are equivalent to the clauses in Γ_{in} but with each occurrence of x_j replaced by x_{ij}.

Altogether, this ensures that the input constraint is enforced for the judgment sets in the modified profile.

The objective in the case of Dodgson is encoded by including for every pair i, j for $i = 1, \ldots, |P|$ and $j = 1, \ldots, |X_\Phi|$ (i) the soft clause (x_{ij}) if $x_j \in J_i$, and (ii) the soft clause $(\neg x_{ij})$ if $\neg x_j \in J_i$, with weight 1. These soft clauses exactly encode minimization of the number of changes to the input profile.

To summarize, $F_{\textsc{Dodgson}}(P)$ consists of the hard clauses $\Gamma_{\text{out}} \wedge \bigwedge_{i=1}^{|P|} \Gamma_{\text{in}}^i \wedge \bigwedge_{j=1}^{|X_\Phi|} \textsc{support}_j$ and soft clauses (x_{ij}) if $x_j \in J_i$ and $(\neg x_{ij})$ if $\neg x_j \in J_i$, with unit weights. A solution τ to $F_{\textsc{Dodgson}}(P)$ corresponds to a Γ_{out}-consistent judgment set $J = \tau \cap \Phi$. Further, J agrees with the strict majority of the modified profile defined by the x_{ij} variables consisting of Γ_{in}-consistent judgment sets, and this modified profile has minimum distance to the input profile P. The correctness of the MaxSAT encoding follows.

Proposition 4.5. Let $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$ be a given judgment aggregation framework. If $J \in \textsc{Dodgson}(P)$, then there exists an optimal solution τ to the MaxSAT instance $F_{\textsc{Dodgson}}(P)$ with $\tau \cap \Phi = J$. Vice versa, if τ is an optimal solution to $F_{\textsc{Dodgson}}(P)$, then there is a judgment set $J \in \textsc{Dodgson}(P)$ with $J = \tau \cap \Phi$.

5. Iterative SAT-Based Approaches

While outcome determination under the Kemeny, Slater, MaxHamming, Young, and Dodgson judgment aggregation rules is each captured through the generic algorithm detailed in the previous section, a similar approach, based on two MaxSAT solver calls does not presumably exist for the Condorcet, ranked agenda, and LexiMax rules due to intrinsic complexity-theoretic barriers. Furthermore, adapting the generic algorithm to cover reversal scoring would require a monolithic and complex MaxSAT encoding which "computes" $R(J_i, l)$ within the encoding. With these considerations, in this section we develop iterative algorithms conforming to the known computational complexity of outcome determination under the remaining rules. Specifically, we detail a MaxSAT-based approach to outcome determination under reversal scoring; PrefSAT-based counterexample-guided abstraction refinement (CEGAR) under Condorcet and Ranked agenda; and a MaxSAT-based multi-level optimization approach to outcome determination under the LexiMax rule.

5.1 Reversal Scoring Rule

Outcome determination under the reversal scoring rule is known to be Θ_2^p-complete (Endriss et al., 2020). Instead of a monolithic approach, here we propose an iterative MaxSAT approach to reversal scoring based on solving a series of simpler MaxSAT queries. Recall that for judgment set $J_i \in P$ and issue $l \in \Phi$, the reversal score $R(J_i, l)$ is defined as the minimum Hamming distance $H(J, J_i)$ over all $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$, and the task is to find a judgment set J consisting of issues l which minimize the total reversal score $\sum_{J_i \in P} \sum_{l \in J} R(J_i, l)$. Our strategy is to first compute for each J_i and $l \in \Phi$ the reversal score $R(J_i, l)$ using a MaxSAT solver call. After computing the reversal scores, we encode the reversal scoring rule directly via another MaxSAT instance, and finally apply the same generic algorithm for outcome determination from Section 4.

Algorithm 2 MaxSAT-based algorithm for outcome determination under the reversal scoring rule. **Input**: JA framework $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$, $L \subseteq \Phi$.

1: $S \leftarrow \{(l) \mid l \in \Phi\}$
2: **for** $J_i \in P$ **do**
3: $\quad w \leftarrow \{(l) \mapsto 1 \mid l \in J_i\} \cup \{(l) \mapsto 0 \mid l \notin J_i\}$
4: $\quad$ **for** $l \in J_i$ **do**
5: $\quad\quad (c, \_) \leftarrow \textsc{MaxSAT}(\Gamma_{\text{out}} \wedge (\neg l), S, w)$
6: $\quad\quad R(J_i, l) \leftarrow c$
7: $\quad\quad (c, \_) \leftarrow \textsc{MaxSAT}(\Gamma_{\text{out}} \wedge (l), S, w)$
8: $\quad\quad R(J_i, \neg l) \leftarrow c$
9: $w \leftarrow \{l \mapsto \sum_{J_i \in P} R(J_i, l) \mid l \in \Phi\}$
10: $(c^*, \tau) \leftarrow \textsc{MaxSAT}(\Gamma_{\text{out}}, S, w)$
11: $(c^L, \tau) \leftarrow \textsc{MaxSAT}(\Gamma_{\text{out}} \wedge \bigwedge_{l \in L} l, S, w)$
12: **if** $c^* = c^L$ **return** $\tau \cap \Phi$ **else return false**

Our approach for reversal scoring is outlined as Algorithm 2. We begin by initializing a set S of (unweighted) unit soft clauses (l) for each $l \in \Phi$ (line 1). We then proceed by computing $R(J_i, l)$ for each $J_i \in P$ and $l \in \Phi$ via a series of related MaxSAT queries (lines 2–8). In order to query the solver for a judgment set which minimizes the Hamming distance to J_i, we assign weight 1 to each $(l) \in S$ where $l \in J_i$, resulting in incurring unit cost for each literal in the output judgment set not included in J_i (line 3). Then, we iterate through $l \in J_i$ (lines 4–8), and solve a MaxSAT instance with hard clauses $\Gamma_{\text{out}} \wedge (\neg l)$ (line 5) and the set of soft clauses S. An optimal solution obtained from this solver call corresponds to a judgment set which satisfies Γ_{out}, contains the literal $(\neg l)$, and minimizes the Hamming distance to J_i. We set $R(J_i, l)$ as the cost of this optimal MaxSAT solution (line 6). We then similarly query for a judgment set containing literal (l) to compute $R(J_i, \neg l)$ with another MaxSAT solver call. Note that if J_i is Γ_{out}-consistent—which is the case for any judgment set $J_i \in P$ if Γ_{in} logically entails Γ_{out}, i.e., any satisfying truth assignment to Γ_{in} also satisfies Γ_{out}—then $R(J_i, \neg l)$ is trivially zero.[8]

After exiting the for-loop (lines 1–8), we have computed the reversal score for each pair (J_i, l). Finally, similarly as in Algorithm 1, we make two more MaxSAT solver calls. First we solve the MaxSAT instance with Γ_{out} as hard clauses, and set the weight of each soft clause (l) according to the just-computed value $R(J_i, l)$ (line 9). This solver call provides the optimal reversal score c^* (line 10). The final MaxSAT solver call is made on the hard clauses $\Gamma_{\text{out}} \wedge \bigwedge_{l \in L} l$ and the same soft clauses as the previous call (line 11). As in Algorithm 1, the two optimal costs obtained are compared. If the costs are the same, we return the truth assignment obtained from the final solver call as a collective judgment set which includes L. Otherwise, there is no such collective judgment set (line 12).

We note that Algorithm 2 can be implemented using a single instantiation of an incremental MaxSAT solver (Niskanen et al., 2022). In particular, after initializing the MaxSAT solver with the output constraint Γ_{out} as hard clauses, every MaxSAT solver call made within

8. We note that this algorithm fixes a version of the algorithm presented in the preliminary conference version of this article (Conati et al., 2023) that set $R(J_i, \neg l) := 0$ for all J_i, l such that $l \in J_i$—this is, as stated, only true under the assumption that all $J_i \in P$ are Γ_{out}-consistent.

Algorithm 3 PrefSAT-based CEGAR for outcome determination under the Condorcet rule. **Input**: JA framework $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$, $L \subseteq \Phi$.

1: $F \leftarrow \Gamma_{\text{out}}$
2: $w \leftarrow \{l \mapsto 1 \mid l \in J_{\text{m}}(P)\}$
3: **while true do**
4: $\quad (result, \tau) \leftarrow \textsc{PrefSAT}(F \wedge \bigwedge_{l \in L} l,\, w)$
5: $\quad$ **if** $result = unsat$ **then return false**
6: $\quad (result, \tau) \leftarrow \textsc{PrefSAT}(F \wedge \bigwedge_{l \in J_{\text{m}}(P) \cap \tau} l \wedge \bigvee_{l \in J_{\text{m}}(P) \setminus \tau} l, w)$
7: $\quad$ **if** $result = unsat$ **then return** $\tau \cap \Phi$
8: $\quad F \leftarrow F \wedge \bigvee_{l \in J_{\text{m}}(P) \setminus \tau} l$

the for-loop (lines 1–8) involves changing weights of soft clauses (line 3) or is made under different assumptions, i.e., partial assignments (lines 5 and 7). After exiting the loop, we change the weights of soft clauses once more (line 9), and similarly to Algorithm 1 for generic MaxSAT-based outcome determination, issue two final MaxSAT solver calls (lines 10–11) where the second call is performed on an instance with additional hard clauses. All of these operations are supported by modern incremental MaxSAT solvers.

5.2 Condorcet Rule

We next consider outcome determination under the Condorcet rule.

Outcome determination under both the Condorcet and ranked agenda rules is known to be Σ_2^p-complete (Endriss et al., 2020). This problem complexity motivates the design of approaches based on counterexample-guided abstraction refinement (CEGAR; Clarke et al., 2003).

In short, CEGAR refers to a generic algorithmic approach, typically based on the use of multiple SAT solvers. A CEGAR algorithm starts from an abstraction which is an implicit representation of an overapproximation of the set of solutions to the original problem at hand. Then the following steps are iterated over: (i) solve the current abstraction to obtain a candidate solution; (ii) check if there is a counterexample witnessing that the candidate solution is not an actual solution to the original problem; and (iii) if there is no counterexample, the candidate solution is necessarily an actual solution, and the algorithm terminates; otherwise, the current abstraction is refined so that the latest candidate solution is ruled out from the current overapproximation of the the set of solutions to the original problem.

Here we detail CEGAR algorithms to outcome determination under the Condorcet and ranked agenda rules based on using a PrefSAT solver (instead of the more typically applied SAT solver) for steps (i) and (ii). In particular, the preferences applicable in PrefSAT turn out to be well-suited for these tasks.

We start with the Condorcet rule. Under the Condorcet rule, solutions are judgment sets which agree with the majoritarian judgment set subset-maximally. Recall that in PrefSAT (see Section 3.2), the preference level of each literal is indicated by a weight function. A key observation here is that by setting the weight of a given set of literals to 1 and the rest to 0, a PrefSAT solver computes as a solution a truth assignment where these literals are

set to true in a subset-maximal sense. This means that by setting the weight of the issues supported by a majority of the agents to 1, PrefSAT solutions will correspond to optimal judgment sets under the Condorcet rule.

Proposition 5.1. Let $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$ be a given judgment aggregation framework. Define the preferences $w_{\text{C}}(l) = 1$ for each $l \in J_{\text{m}}(P)$. If $J \in \textsc{Condorcet}(P)$, then there exists a solution τ to the PrefSAT instance $(\Gamma_{\text{out}}, w_{\text{C}})$ with $\tau \cap \Phi = J$. Vice versa, if τ is a PrefSAT solution to $(\Gamma_{\text{out}}, w_{\text{C}})$, then there is a judgment set $J \in \textsc{Condorcet}(P)$ with $J = \tau \cap \Phi$.

This proposition allows us to compute optimal judgment sets under the Condorcet rule using a PrefSAT solver and, in particular, forms the basis for the PrefSAT-based CEGAR approach outlined as Algorithm 3 to outcome determination under the Condorcet rule. First, we initialize the abstraction F as the output constraint Γ_{out} (line 1) and the preferences as $w(l) = 1$ for each $l \in J_{\text{m}}(P)$ (line 2). Then we enter the main CEGAR loop (lines 3–8). Within the CEGAR loop, we first query a PrefSAT solver for a candidate solution including L (line 4). If the PrefSAT solver reports unsatisfiability, no such judgment set exists (line 5) and the algorithm terminates. Otherwise, the obtained candidate solution corresponds to a judgment set J for which $J \cap J_{\text{m}}(P)$ is subset-maximal under the constraint $L \subseteq J$. We proceed by checking for a counterexample, i.e., checking whether the candidate solution J is Condorcet-optimal—the candidate solution may not be Condorcet-optimal due to the the outcome constraint $L \subseteq J$ that was enforced. Specifically, we query a PrefSAT solver for a counterexample J' (line 6). In this solver call, we (i) drop the outcome constraint $L \subseteq J$ and (ii) enforce the additional constraints

$$\bigwedge_{l \in J_m(P) \cap \tau} l \text{ and} \tag{1}$$

$$\bigvee_{l \in J_m(P) \setminus \tau} l \ . \tag{2}$$

Constraint (1) enforces that every literal $l \in J$ in the majoritarian judgment set is assigned to 1. Constraint (2) enforces that J' includes at least one majority-supported agenda item not included in J. Together, these constraints ensure that $J' \cap J_{\text{m}}(P) \supset J \cap J_{\text{m}}(P)$. Hence such a J', if one exists, is a counterexample witnessing the fact that $J \notin \textsc{condorcet}(P)$. If there is no such counterexample, then $J \in \textsc{condorcet}(P)$ and we return the collective judgment set J which includes L as a solution (line 7). On the other hand, if a counterexample is obtained, this iteration did not identify a Condorcet solution containing L. We then refine the current abstraction F by adding clauses to it which block all assignments that are at least as close to the majority as the counterexample (line 8), and proceed with the next iteration.

5.3 Ranked Agenda Rule

We also propose a PrefSAT-based CEGAR approach for the Σ_2^p-complete problem of outcome determination under the ranked agenda rule. Recall that the ranked agenda rule defines a set of optimal judgment sets which are subset-maximal with respect to the preference classes $L_k^P = \{l \in \Phi \mid N(P, l) = k\}$ in decreasing order of support. This allows

Algorithm 4 PrefSAT-based CEGAR for outcome determination under the ranked agenda rule. **Input**: JA framework $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$, $L \subseteq \Phi$.

1: $F \leftarrow \Gamma_{\text{out}}$
2: $w \leftarrow \{l \mapsto N(P,l) \mid l \in \Phi, N(P,l) > |P|/2\}$
3: **while true do**
4: $\quad (result, \tau) \leftarrow \textsc{PrefSAT}(F \wedge \bigwedge_{l \in L} l, w)$
5: $\quad$ **if** $result = unsat$ **then return false**
6: $\quad (result, \tau) \leftarrow \textsc{PrefSAT}(F \wedge \textsc{MorePref}(\tau), w)$
7: $\quad$ **if** $result = unsat$ **then return** $\tau \cap \Phi$
8: $\quad F \leftarrow F \wedge \neg\tau \wedge \neg\textsc{LessPref}(\tau)$

ranked agenda solutions to be obtained as truth assignments to PrefSAT instances similarly to Condorcet solutions. However, instead of having two preference levels defined by the majoritarian judgment set (as in the case of Condorcet), to capture ranked agenda, the preference level of each literal with majority support is given by the number of supporters for the literal.

Proposition 5.2. Let $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$ be a given judgment aggregation framework. Define the preferences $w_{\text{RA}}(l) = N(P,l)$ for each $l \in \Phi$ with $N(P,l) > n/2$. If it holds that $J \in \textsc{RankedAgenda}(P)$, then there exists a solution τ to the PrefSAT instance $(\Gamma_{\text{out}}, w_{\text{RA}})$ with $\tau \cap \Phi = J$. Vice versa, if τ is a PrefSAT solution to $(\Gamma_{\text{out}}, w_{\text{RA}})$, then there is a judgment set $J \in \textsc{RankedAgenda}(P)$ with $J = \tau \cap \Phi$.

Proposition 5.2 forms the basis for a CEGAR approach to outcome determination under the ranked agenda rule, presented as Algorithm 4. In addition to the assignment of preference levels, the key differences here to Algorithm 3 (for the case of Condorcet) are in the counterexample-check (line 6) and refinement (line 8) steps, reflecting Proposition 5.2.

In detail, we initialize the abstraction as the output constraint Γ_{out} (line 1), and set the preference level of each $l \in \Phi$ as $N(P,l)$ if $N(P,l) > n/2$ (line 2). We then enter the main CEGAR loop (lines 3–8). First, a PrefSAT solver is queried for an assignment that includes L (line 4). If the solver reports unsatisfiability, we know that no such judgment set exists (line 5), and the algorithm terminates by reporting "false". Otherwise, we obtain as a candidate solution a truth assignment τ corresponding to a judgment set J that is $\succ_{\text{RA}}$-maximal under the constraint that $L \subseteq J$. The candidate solution is not an actual solution if there is a judgment set that does not include the outcome L and that is preferred to J. We use a PrefSAT solver to check for the existence of such a counterexample judgment set as follows. Let Λ_k denote the set of literals with weight k, i.e., $\Lambda_k = \{l \in \textsc{lit}(F) \mid w(l) = k\}$, and define:

$$\textsc{Level}_k^\tau = \bigwedge_{l \in \tau \cap \Lambda_k} l \ ,$$
$$\textsc{More}_k^\tau = \bigvee_{l \in \neg\tau \cap \Lambda_k} l \ ,$$
$$\textsc{EqUpTo}_k^\tau = \textsc{EqUpTo}_{k+1}^\tau \wedge \textsc{Level}_k^\tau \wedge \neg\textsc{More}_k^\tau \qquad \text{for } k < |P| \ ,$$
$$\textsc{EqUpTo}_{|P|}^\tau = \textsc{Level}_{|P|}^\tau \wedge \neg\textsc{More}_{|P|}^\tau \ .$$

Here $\textsc{Level}_k^\tau$ encodes that all literals with weight k satisfied by τ must be satisfied, $\textsc{More}_k^\tau$ encodes that some literal with weight k that is falsified by τ must be satisfied, and $\textsc{EqUpTo}_k^\tau$ encodes that all literals with weight at least k be assigned according to τ. Using these auxiliary definitions, we define

$$\textsc{MorePref}(\tau) = \bigvee_{k=\lfloor n/2 \rfloor+1}^{n} \Big(\textsc{Level}_k^\tau \wedge \textsc{More}_k^\tau \wedge \textsc{EqUpTo}_{k+1}^\tau \Big).$$

The constraint $\textsc{MorePref}(\tau)$ enforces that (i) there is a preference level k in terms of which all literals in $\bigcup_{i=k}^{|P|} \Lambda_i$, i.e., all literals with weight at least k, which are satisfied by τ are satisfied, and that (ii) some literal $l \in \Lambda_k$ with $l \notin \tau$, i.e., a literal with weight k that is not satisfied by τ, is now satisfied. This constraint enforces exactly the condition for a counterexample, i.e., a judgment set $J' \succ_{\text{RA}} J$ which does not include L. Hence for the counterexample check, we call a PrefSAT solver on $F \wedge \textsc{morepref}(\tau)$ (line 6). If the solver reports unsatisfiability, there are no counterexamples, and hence the algorithm terminates by returning the $\succ_{\text{RA}}$-maximal truth assignment τ representing the collective judgment set which includes L (line 7). Otherwise, this iteration did not identify a ranked agenda solution containing L. We hence refine the current abstraction F by adding a clausal representation of a constraint that rules out assignments corresponding to judgment sets J' to which the counterexample judgment set J is preferred (or equivalent), i.e., $J \succ_{\text{RA}} J'$ or $J = J'$ (line 8). This refinement is defined as

$$\textsc{LessPref}(\tau) = \bigvee_{k=\lfloor n/2 \rfloor+1}^{n} \Big(\neg\textsc{Level}_k^\tau \wedge \neg\textsc{More}_k^\tau \wedge \textsc{EqUpTo}_{k+1}^\tau \Big),$$

enforcing that (i) there is a preference level k for which a strict subset of literals satisfied by τ are satisfied, and that (ii) all literals with weight greater than k are satisfied exactly according to τ.

Example 5.1. Recall the example judgment aggregation framework with agenda $\Phi = \{t, h, q, a, \neg t, \neg h, \neg q, \neg a\}$, input/output constraints $\Gamma_{\text{in}} = \Gamma_{\text{out}} = a \leftrightarrow ((t \wedge h) \vee (t \wedge q))$, and the profile P outlined in Example 2.3. Given the outcome $L = \{\neg q\}$ as input, Algorithm 4 proceeds as follows. The abstraction F is initialized as the output constraint Γ_{out} (line 1) and the weights of the literals as $w \leftarrow \{t \mapsto 8, h \mapsto 7, \neg a \mapsto 7, q \mapsto 6\}$, i.e., the weight to each literal is the majority support according to their support in P (line 2). We then enter the main CEGAR loop. There are 4 solutions satisfying F under the assumption $\neg q$, namely $\{t, h, \neg q, a\}, \{t, \neg a, \neg q, \neg h\}, \{h, \neg a, \neg q, \neg t\}$, and $\{\neg a, \neg q, \neg h, \neg t\}$. Of these, the first two are both preferred to the latter two since the first two contain the most supported literal t while the latter two do not. Hence $\{t, h, \neg q, a\}$ and $\{t, \neg a, \neg q, \neg h\}$ are the two possible solutions to the PrefSAT call on line 4. Suppose that the solver arbitrarily returns $\tau = \{t, h, \neg q, a\}$. Since we obtained a solution (*result = sat*), we proceed to the counterexample check (line 5), relaxing the constraint that the solution must contain $\neg q$ and instead enforcing $\textsc{morepref}(\tau)$, i.e., that a counterexample must be preferred to τ under w (line 6). One such counterexample is $\{t, h, q, a\}$ (note also that the counterexample does not contain L and was therefore not a solution to the abstraction). We then proceed by

refining the current abstraction (line 8) by the constraint LESSPREF(τ), ruling out solutions to the abstraction over which the counterexample is preferred. For the second iteration of the main loop, note that out of the 4 solutions to the initial abstraction, the just-obtained counterexample is preferred to all of them except $\{t, \neg a, \neg q, \neg h\}$ and hence the three other previously-available candidate solutions are now ruled out after the refinement step. Hence the next PrefSAT solver call for a candidate solution will return $\{t, \neg a, \neg q, \neg h\}$ as the only solution to the refined abstraction (line 4). Now the counterexample check will return *unsat* (line 6), since there is no solution which is preferred to this candidate solution. Hence the algorithm terminates by returning $\{t, \neg a, \neg q, \neg h\}$ as an RA-optimal solution containing L (line 7). ◊

5.4 LexiMax Rule

Finally, we consider outcome determination under the LexiMax rule which has been shown to be Δ_2^p-complete. Specifically, by identifying that the LexiMax rule is tightly connected to the so-called boolean multilevel optimization (BMO) property (Argelich et al., 2009), we detail an iterative MaxSAT-based algorithm to outcome determination under the LexiMax rule.

The BMO property allows for solving MaxSAT instances which potentially have very high cumulative weight over their soft clauses by viewing them as a series of hierarchically dependent subproblems. More formally, the soft clauses of a MaxSAT instance can be partitioned into a series of weight classes $(C_1, \ldots, C_n)$ with associated weights $(w_1, \ldots, w_n)$, where each clause $C \in C_i$ has weight w_i. A MaxSAT instance has the BMO property if, within each weight class, a single clause has weight exceeding the weight of all clauses in lower weight classes combined, i.e., if

$$w_i > \sum_{j=1}^{i-1} w_j \cdot |C_j| \qquad i = 2, \ldots, n.$$

MaxSAT instances with the BMO property can be solved as a series of simpler MaxSAT instances, by optimizing with respect to each of the weight classes in order, beginning with the highest weight class C_1. After determining the maximum number k of clauses that can be simultaneously satisfied in C_j, we enforce that in subsequently MaxSAT solver calls k clauses from C_j must be satisfied.[9]

Towards a connection between LexiMax and BMO, recall that under the LexiMax rule, the agenda items are partitioned based on their support into sets $L_k^P = \{l \in \Phi \mid N(P, l) = k\}$ for different levels of support k. The LexiMax rule maximizes the *cardinality* of the agreement between the output J and each of the L_k^P ($|J \cap L_k^P|$) in decreasing order of support. This naturally maps to MaxSAT with the BMO property. Specifically, we assign weights to literals so that a literal is prioritized over the set of *all* less preferred literals. More formally, we define $F_{\text{LEXIMAX}}(P)$ as the MaxSAT instance with hard clauses Γ_{out}, soft clauses $S_{|P|} \cup \cdots \cup S_{\lfloor |P|/2 \rfloor}$ where $S_k = \{(l) \mid l \in L_k^P\}$, and associated weights $w_{|P|}, \ldots, w_{\lfloor |P|/2 \rfloor}$ where the weights satisfy the BMO property i.e., $w_i > \sum_{j=\lfloor |P/2 \rfloor}^{i-1} w_j \cdot |S_j|$ for $i = \lfloor |P|/2 \rfloor, \ldots, |P|$.

9. This restriction is implemented by appending a relaxation variable to each of the soft clauses in C_1 and enforcing a lower bound on the number of them which are allowed to be relaxed. This lower bound is implemented via a cardinality constraint.

Algorithm 5 MaxSAT-based algorithm for outcome determination under the LexiMax rule. **Input**: JA framework $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$, $L \subseteq \Phi$.

1: $F \leftarrow \Gamma_{\text{out}}$
2: **for** $k = |P|, \ldots, \lfloor |P|/2 \rfloor + 1$ **do**
3: $\quad S \leftarrow \{(l) \mid l \in L_k^P\}$, $w \leftarrow \{(l) \mapsto 1 \mid l \in L_k^P\}$
4: $\quad$ **if** $k = \lfloor |P|/2 \rfloor + 1$ **then break**
5: $\quad (c_k, \tau_k) \leftarrow \textsc{MaxSAT}(F, S, w)$
6: $\quad F \leftarrow F \wedge \left(\sum_{l \in L_k^P} \neg l = c_k\right)$
7: $(c^*, \tau) \leftarrow \textsc{MaxSAT}(F, S, w)$
8: $(c^L, \tau) \leftarrow \textsc{MaxSAT}(F \wedge \bigwedge_{l \in L} l, S, w)$
9: **if** $c^* = c^L$ **return** $\tau \cap \Phi$ **else return false**

Proposition 5.3. Let $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$ be a given judgment aggregation framework. If $J \in \textsc{LexiMax}(P)$, then there exists an optimal solution τ to the MaxSAT instance $F_{\textsc{LexiMax}}(P)$ with $\tau \cap \Phi = J$. Vice versa, if τ is an optimal solution to $F_{\textsc{LexiMax}}(P)$, then there is a judgment set $J \in \textsc{LexiMax}(P)$ with $J = \tau \cap \Phi$.

Our algorithm for outcome determination under the LexiMax rule is presented as Algorithm 5. The algorithm follows a generic so-called level-wise approach to solving MaxSAT instance with the BMO property (Argelich et al., 2009). Specifically, Algorithm 5 iterates over the L_k^P in decreasing order. At each iteration, the maximum number c_k of literals $l \in L_k^P$ that can be satisfied simultaneously is determined with a MaxSAT solver call, and then it is enforced that c_k literals with support k must be satisfied in subsequent iterations via cardinality constraint. Once each level $|P| \ldots \lfloor |P|/2 \rfloor$ has been iterated over, a judgment set J obtained from a satisfying assignment τ (via $\tau \cap \Phi$) will be a Leximax solution ($J \in \textsc{leximax}(P)$). In more detail, we start by initializing a formula with the output constraint Γ_{out} (line 1). Now, starting from $k = |P|$, we declare (l) for each $l \in L_k^P$ as a soft clause with unit weight (line 3), and solve the resulting MaxSAT instance (line 5). Specifically, we obtain the optimal cost c_k as the number of soft clauses falsified by the optimal solution on level k, where each clause corresponds to a literal with support k. We then refine F with the the cardinality constraint $\sum_{l \in L_k^P} \neg l = c_k$ (line 6), enforcing optimal cost over level-k literals to be maintained in subsequent iterations. Decrementing k after each iteration, we continue until k is the smallest possible level $\lfloor |P|/2 \rfloor + 1$, after which the main loop is exited (line 4). At this point, the optimal solutions to the current MaxSAT instance correspond exactly to LexiMax judgment sets; we obtain such a a solution of cost c^* with a call to a MaxSAT solver (line 7). Finally, we call the MaxSAT solver under the additional constraint $L \subseteq J$ to obtain the optimal cost c^L for this instance (line 8). If the costs c^* and c^L are the same, the truth assignment obtained from the last MaxSAT solver call is returned as a guaranteed LexiMax judgment set which includes L. Otherwise, there is no such judgment set (line 9).

6. Integer Programming Encodings for Outcome Determination

So far, we have detailed algorithms for outcome determination under the various aggregation rules by making use of SAT-based solvers, including MaxSAT and PrefSAT as suitable formalisms. The Condorcet and ranked agenda rules are, in particular, tightly connected to PrefSAT, requiring forms of subset-maximization that are generally not directly supported by standard constraint optimization solvers. The other rules, on the other hand, may be captured by instantiating our approaches alternatively through different constraint optimization paradigms. This is most evident for the generic approach (recall Algorithm 1 in Section 4) that captures outcome determination for the Kemeny, Slater, MaxHamming, Young, and Dodgson rules in a uniform way. Instantiating the algorithm specifically requires encoding the rules using the declarative language of choice, assuming that the language supports optimization statements and that efficient exact solvers are available for the language. As a second alternative to MaxSAT, in this section we detail integer programming (IP) encodings for the Kemeny, Slater, MaxHamming, Young, and Dodgson rules. We also outline how the LexiMax rule can be covered by an iterative procedure similar to Algorithm 5 from Section 5.

The most natural choice of declarative language for encoding different judgment aggregation rules depends heavily on the rule in question. The use of linear inequalities (the language of IP) is tempting especially for rules which require enforcing cardinality constraints with potentially high bounds. Whereas encoding of such constraints is achieved in MaxSAT by the use of optimized CNF encodings of cardinality constraints, for rules such as Dodgson the IP alternative can be considered more natural, as we will see. However, it should be noted that regardless of the rule considered, in the JA framework the input/output constraints are expressed propositionally, making SAT-based solvers a natural choice. For instantiating our algorithms through integer programming we will employ a standard encoding of a given propositional constraint Γ to linear inequalities over binary variables. For example, a constraint in conjunctive normal form (CNF) can be directly expressed as at-least constraints: a clause C is equivalent to the constraint $\sum_{x \in C} x + \sum_{\neg x \in C} (1 - x) \geq 1$.

Similarly to our MaxSAT encodings, we include all issues X_Φ into the IP encodings as binary variables. An IP-based instantiation of Algorithm 1 is obtained as follows. The constraint $\bigwedge_{l \in L} l$ on the outcome is expressed as the set of equality constraints $x_j = 1$ for $x_j \in L$ and $x_j = 0$ for $\neg x_j \in L$.

Recall that for the MaxSAT encoding of the Kemeny and Slater rules (Section 4.2), we set Γ_{out} as hard clauses and soft clauses as $\{(l) \mid l \in J_{\text{m}}(P)\}$, with weight 1 for the Slater rule and weight $N(P, l) - N(P, \neg l)$ for the Kemeny rule. For an IP encoding, we use each issue x_j, $j = 1, \ldots, m$, as a binary variable, and include Γ_{out} as integer linear constraints. The IP objective for the Slater rule is

$$\text{maximize} \quad \sum_{x \in J_{\text{m}}(P)} x + \sum_{\neg x \in J_{\text{m}}(P)} (1 - x),$$

by directly mapping the soft clauses to linear terms. Likewise, for the Kemeny rule, the IP objective is

$$\text{maximize} \quad \sum_{x \in J_{\text{m}}(P)} (N(P, x) - N(P, \neg x)) \cdot x + \sum_{\neg x \in J_{\text{m}}(P)} (N(P, \neg x) - N(P, x)) \cdot (1 - x),$$

which in contrast to the Slater objective accounts for the weights of the soft clauses.

The MaxHamming rule can be encoded in IP as

$$\begin{aligned}
\text{minimize} \quad & d && && (3)\\
\text{subject to} \quad & \Gamma_{\text{out}} && && (4)\\
& \sum_{x_j \in J_i} (1 - x_j) + \sum_{\neg x_j \in J_i} x_j \leq d && \forall i = 1, \ldots, n && (5)\\
& d \in \mathbb{Z}_+ && && (6)\\
& x_j \in \{0, 1\} && \forall j = 1, \ldots, m && (7)
\end{aligned}$$

The integer variable d (6) represents the maximum Hamming distance over all judgment sets in the profile, which is minimized (3). The constraint (5) counts the number of issues in the collective judgment set which are set differently from the individual judgment set J_i, and sets this number to be at most d, effectively minimizing the maximum Hamming distance.

For the Young rule, similarly to the MaxSAT encoding (recall Section 4.4), we make use of additional binary variables y_i for $i = 1, \ldots, n$, indicating that the judgment set J_i is included in the Young subprofile. Now, the Young rule can be be encoded as

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{n} y_i && && (8)\\
\text{subject to} \quad & \Gamma_{\text{out}} && && (9)\\
& x_j = 1 \Rightarrow \sum_{J_i : x_j \in J_i} y_i \geq \frac{1}{2} \sum_{i=1}^{n} y_i && \forall j = 1, \ldots, m && (10)\\
& x_j = 0 \Rightarrow \sum_{J_i : x_j \in J_i} y_i \leq \frac{1}{2} \sum_{i=1}^{n} y_i && \forall j = 1, \ldots, m && (11)\\
& y_i \in \{0, 1\} && \forall i = 1, \ldots, n && (12)\\
& x_j \in \{0, 1\} && \forall j = 1, \ldots, m && (13)
\end{aligned}$$

As the Young objective (8), we maximize the number of agents included. For the constraints, $\Rightarrow$ denotes a so-called indicator constraint[10] which enforces that the right-hand side must hold whenever the variable assignment on the left-hand side holds. In particular, if $x_j = 1$ (respectively, $x_j = 0$), the majority of agents in the Young subprofile must support the issue x_j, as enforced by constraint (10) (respectively, $\neg x_j$ and constraint (11)).

Our IP encoding for the Dodgson rule, similar to the corresponding MaxSAT encoding (recall Section 4.5), makes use of additional binary variables x_{ij} for $i = 1, \ldots, n$ and $j = 1, \ldots, m$ indicating that the Dodgson modified profile the ith judgment set includes issue

10. Indicator constraints are widely supported in IP solvers.

x_j. Using these auxiliary variables, the Dodgson rule is encoded in IP as

$$\text{maximize} \quad \sum_{i=1}^{n} \left(\sum_{x_j \in J_i} x_{ij} + \sum_{x_j \notin J_i} (1 - x_{ij}) \right) \tag{14}$$

$$\text{subject to} \quad \Gamma_{\text{out}} \tag{15}$$

$$\Gamma_{\text{in}}[x_j \mapsto x_{ij} \mid j = 1, \ldots, m] \qquad \forall i = 1, \ldots, n \tag{16}$$

$$x_j = 1 \Rightarrow \sum_{J_i : x_j \in J_i} x_{ij} \geq n/2 \qquad \forall j = 1, \ldots, m \tag{17}$$

$$x_j = 0 \Rightarrow \sum_{J_i : x_j \in J_i} x_{ij} \leq n/2 \qquad \forall j = 1, \ldots, m \tag{18}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i = 1, \ldots, n, j = 1, \ldots, m \tag{19}$$

$$x_j \in \{0, 1\} \qquad \forall j = 1, \ldots, m \tag{20}$$

For the Dodgson objective (14), we minimize the number of changes to the input profile, i.e., maximize the agreement with the input profile. The constraint (16) ensures the Γ_{in}-consistency of the Dodgson modified profile. The two indicator constraints (17) and (18) ensure that if $x_j = 1$ (respectively, $x_j = 0$), the majority of agents in the Dodgson modified profile must support the issue x_j (respectively, $\neg x_j$).

Finally, an IP-based version of Algorithm 5 for outcome determination under the Lexi-Max rule is obtained with the following changes. Again, the output constraint Γ_{out} is represented with linear constraints. The additional cardinality constraints are a specific form of linear inequalities with 0-1 coefficients. On each level k, the unit-weighted soft clauses (l) for $l \in L_k^P$ map exactly to maximizing the linear function $\sum_{x \in L_k^P} x + \sum_{\neg x \in L_k^P} (1 - x)$. Finally, the cardinality constraint used for enforcing optimal cost for each level k is directly expressed as a linear equality.

7. Empirical Evaluation

We report on an empirical evaluation of the runtime performance of our implementations of all of the approaches detailed in Sections 4-6. All of the experiments reported on in this section were run on Intel Xeon E5-2670 CPUs and 57-GB memory under RHEL 8.5 using a per-instance 30-minute time and 16-GB memory limit.

7.1 Implementation and Competing Approaches

Our implementations of the encodings and algorithms described in Sections 4–6 are available as open source software at `https://bitbucket.org/coreo-group/satcha/`[11]. As the MaxSAT solver, we employ the freely-available UWrMaxSAT (version 1.5.3; Piotrów, 2020) as a representative state-of-the-art solver implementing the so-called core-guided approach

11. As a side-remark, while the main focus of this article is on outcome determination in judgment aggregation, our implementation allows also for enumerating all collective judgment sets. In short, all of the algorithms proposed in this article straightforwardly extend to enumeration; for some intuition, this achieved by iteratively adding a constraint that rules out the latest found judgment set and calling the solver again until no further (optimal) judgment sets exist.

to MaxSAT solving (Bacchus et al., 2021). We make use of the IPAMIR API for incremental MaxSAT (Niskanen et al., 2022) supported by UWrMaxSAT, since incrementality can be beneficial in terms of runtime performance when making several related MaxSAT solver calls as is done in our iterative MaxSAT-based algorithms. For encoding cardinality constraints into CNF when necessary in the algorithms, we adjusted the iterative totalizer encoding (Bailleux & Boufkhad, 2003; Martins et al., 2014) from PySAT (Ignatiev et al., 2018), a toolkit for prototyping using SAT solvers and SAT-related technologies in Python, for the purposes of our approaches. For the PrefSAT solver within our algorithms for outcome determination under the Condorcet and ranked agenda rules, we use MiniPref (Dodaro & Previti, 2019). In the integer programming instantiations, we use the state-of-the-art commercial IP solver Gurobi (`https://www.gurobi.com/`) version 9.5.2 under an academic license using the automatic (default) solving method. In the implementation for the outcome determination procedure via Gurobi, we terminate the second solver call (with the additional constraint on the outcome) early using a callback function.

In addition to comparing our SAT-based and IP-based implementations against each other, we compare the runtime performance of our approaches to the JA-ASP (de Haan & Slavkovik, 2019) system (`https://github/rdehaan/ja-asp`) based on ASP.[12] In short, JA-ASP utilizes direct ASP optimization statements for Θ_2^p-complete and Δ_2^p-complete rules, and disjunctive ASP-based techniques (saturation) for Σ_2^p-complete rules. In the experiments, we use the state-of-the-art ASP solver Clingo (v5.6.1; Gebser et al., 2016). JA-ASP supports outcome determination via enumeration of answer sets. To enable support for the Young and Dodgson voting rules in JA-ASP, we adapted JA-ASP by separating the input and output constraints within JA-ASP (ASP encoding file `ja.lp`); the adapted ASP encoding is available together with our other implementations.

7.2 Benchmarks

For benchmarks, we employ preference profile data available in the standard PrefLib library (Mattei & Walsh, 2013), consisting of heterogenous real-world preference data from e.g., elections, rankings, and surveys. Specifically, we selected all strict total orderings available in `https://www.preflib.org/static/data/types/soc.zip` (as of March 13, 2024), filtering out those which have a (weak) *Condorcet winner* in order to guarantee removal of instances which can be considered trivial. This resulted in our final set consisting of 405 preference profile datasets as the basis of our benchmarks. Using this data, we benchmark the algorithm implementations on the winner determination problem, i.e., the problem of whether a given candidate is a winner in terms of a given input profile and aggregation rule, for each of the judgment aggregation rules considered in this work. Specifically, we determine whether the (arbitrarily chosen) first candidate is a winner for the input profile and rule. With the exception of MaxHamming and LexiMax, all of the rules considered in this work generalize voting rules from the literature (Endriss, 2018; Endriss et al., 2012; Lang & Slavkovik, 2013; Lang et al., 2017; Miller & Osherson, 2009) in the following sense. For a preference aggregation framework with candidates $C = \{1, \ldots, m\}$ and voters

12. It should be noted that an ASP-based approach specific to voting rules called Democratix has been proposed earlier (Charwat & Pfandler, 2015). However, it appears not to be currently available and we were unable to make contact with the authors despite our efforts to do so.

Table 1: Runtime comparison of our SAT-based and IP-based approaches (on outcome determination) against JA-ASP (on the subproblem of computing a collective judgment set): Number of solved instances (#slv), percentage of solved instances (%slv), and average runtime over solved instances in seconds (avg. (s)

	SAT-based			**IP-based**			**JA-ASP**		
Rule	#slv	%slv	avg. (s)	#slv	%slv	avg. (s)	#slv	%slv	avg. (s)
Kemeny	**380**	93.8	26.05	345	85.2	9.74	38	9.4	86.63
Slater	**379**	93.6	29.81	333	82.2	8.17	177	43.7	43.68
MaxHamming	63	15.6	113.74	**245**	60.5	62.05	36	8.9	177.84
Young	404	99.8	5.17	**405**	100	4.54	270	66.7	150.20
Dodgson	**334**	82.5	82.47	255	63.0	63.47	68	16.8	473.87
LexiMax	**373**	92.1	28.77	341	84.2	19.18	223	55.1	72.87
Reversal scoring	**262**	64.7	97.79	—	—	—	0	0.0	—
Condorcet	**240**	59.3	59.26	—	—	—	114	28.1	576.47
Ranked agenda	**390**	96.3	43.65	—	—	—	0	0.0	—

$V = (\succ_1, \ldots, \succ_n)$, construct the preference agenda with issues $p_{x \succ y}$ for $x, y \in C$, $x < y$ and let $p_{y \succ x} = \neg p_{x \succ y}$ for $x > y$. Then, for the profile $P = (J_1, \ldots, J_n)$ via $J_i = \{p_{x \succ y} \mid x \succ_i y\}$, and

$$\textsc{Transitivity} = \bigwedge_{\substack{x,y,z \in C \\ x \neq y \neq z}} ((p_{x \succ y} \wedge p_{y \succ z}) \rightarrow p_{x \succ z}),$$

$$\textsc{Winner} = \bigvee_{x \in C} \bigwedge_{\substack{y \in C \\ y \neq x}} p_{x \succ y},$$

the Kemeny and Slater judgment aggregation rules correspond to their voting counterparts with $\Gamma_{\text{out}} = \Gamma_{\text{in}} = \textsc{transitivity}$. With $\Gamma_{\text{out}} = \textsc{winner}$ and $\Gamma_{\text{in}} = \textsc{transitivity}$, the Young and Dodgson JA rules similarly correspond to their voting counterparts. And finally, with $\Gamma_{\text{out}} = \Gamma_{\text{in}} = \textsc{transitivity}$, the Condorcet, ranked agenda, and reversal scoring rules correspond to the top cycle, ranked pairs, and Borda voting rules, respectively. While this results in outcome determination problem instances with specific types of structural properties, recall that, as mentioned in Section 2.3, preference aggregation is a widely-studied special case of judgment aggregation, and that winner determination is an instantiation of outcome determination.

7.3 Results

An overview of the results of our empirical evaluation is provided as Table 1. Firstly, it should be noted that while JA-ASP supports outcome determination via enumeration of answer sets, this turned out not to be feasible for almost any of the benchmark instances. In order to provide a form of comparison to JA-ASP, the results we report on for JA-ASP are only for the first part of outcome determination, namely, computation of a single collective judgment set. In contrast, the results for our MaxSAT and IP approaches are reported on the "full" outcome determination task. Hence the number for JA-ASP and our approaches are not directly comparable, but rather, too optimistic for the competing JA-ASP approach.

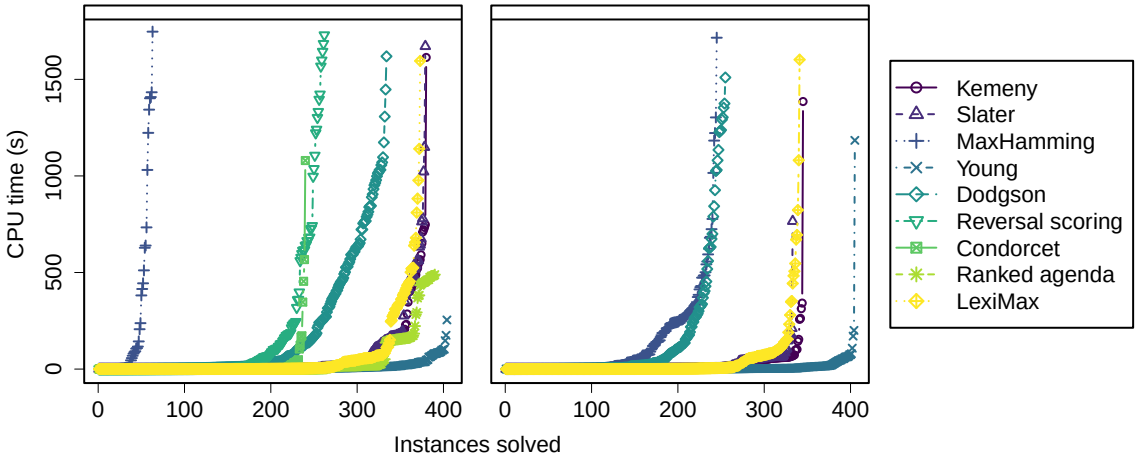

Figure 1: Runtime distribution for SAT-based approaches (left) and IP-based approaches (right).

In spite of this, both the SAT-based and IP-based approaches outperform JA-ASP on each rule, solving up to ten times (Kemeny) as many instances. Furthermore, average runtimes over solved instances are also noticeably lower for both SAT-based and IP-based approaches. Notably, JA-ASP did not terminate on any of the benchmark instances under the reversal scoring and ranked agenda rules. Under Condorcet, the PrefSAT-based approach solved more than two times as many instances as JA-ASP, with notably faster runtimes.

When comparing the performance of the SAT and IP instantiations of our algorithms, we observe that the best choice of a declarative language depends on the aggregation rule. A noticeable difference for the benefit of IP is observed under MaxHamming. This is in line with the fact that MaxHamming requires enforcing a significant number of cardinality constraints with a high bound; this is more suitable for IP than MaxSAT, as the encoding size blows up faster in such cases for MaxSAT. For MaxHamming, we indeed observed that the complex and large MaxSAT instances arising from the MaxSAT encoding resulted in the MaxSAT solver to run out of memory on 31% of the benchmarks (for further details on time vs memory outs, see Appendix A). On the other hand, the MaxSAT-based instantiation of the generic algorithm results in better performance with significantly more instances solved under the Kemeny, Slater and Dodgson rules; more than 92% of the instances are solved under the Kemeny, Slater, and LexiMax rules, and more than 82% are solved under Dodgson. While the MaxHamming rule results in complex and large MaxSAT instances, we observed that under many rules the IP instantiation had higher memory requirements, more often resulting in memory outs under Kemeny, Slater, Dodgson and LexiMax (see again Appendix A).

Comparing the performance differences across the different voting rules, Figure 1 shows the runtimes of each solved instance (sorted by runtime) for our SAT-based and IP-based

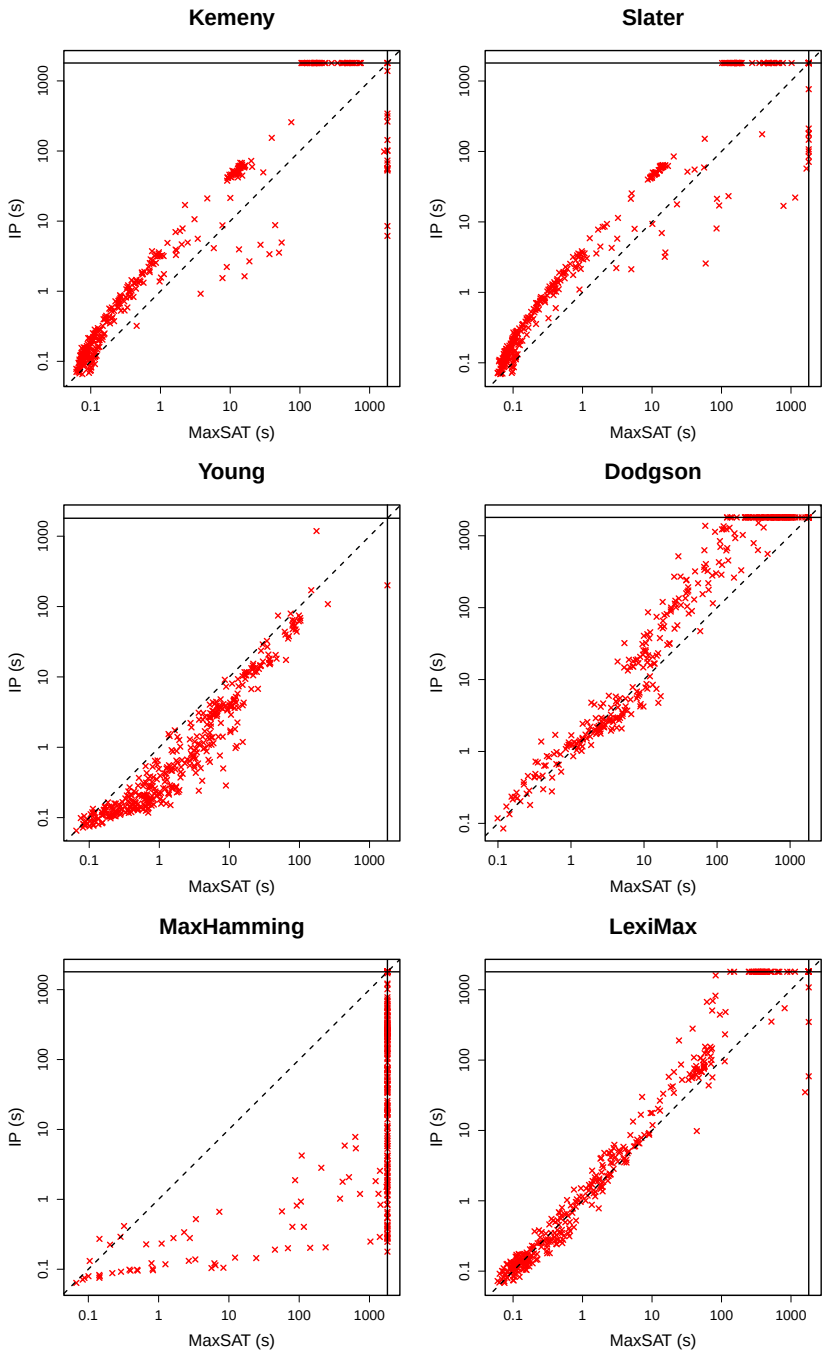

Figure 2: Per-instance comparison of MaxSAT and IP runtimes.

instantiations. The least (although still significant) number of instances are solved under the Condorcet rule, which is also in terms of computational complexity among the hardest of the problem variants. On the other hand, outcome determination under the Young rule appears to be the easiest among the problem variants, with the IP instantiation solving all and the MaxSAT instantiation solving all but one of the benchmark instances. Finally, Figure 2 shows a per-instance runtime comparison between the MaxSAT-based and and IP-based instantiations individually for the applicable aggregation rules. Under the Kemeny, Slater, Dodgson, and LexiMax rules, MaxSAT results in more instances solved overall. Interestingly, there are also some instances which are solved faster (or solely) with the IP instantiation, particularly under the Kemeny, Slater and LexiMax rules. An opposite performance difference is observed under Young and MaxHamming, with IP exhibiting noticeably better runtimes.

8. Conclusions

Judgment aggregation is a general formal framework that allows for modeling agreements of multiple agents through social choice mechanisms. Determining the outcome of judgment aggregation is known to be computationally notoriously hard. This imposes non-trivial challenges for developing practical algorithms for reasoning about the aggregation of judgments under a wide range of central aggregation rules.

Harnessing the power of Boolean satisfiability (SAT) based solvers and integer programming solvers as the underlying NP solving engines, in this work we developed declarative algorithms for outcome determination in judgment aggregation under a range of central judgment aggregation rules, namely, Kemeny, Slater, MaxHamming, Young, Dodgson, reversal scoring, Condorcet, ranked agenda, and LexiMax. Our algorithms adhere to the known complexity bounds for the problem under a range of aggregation rules via identifying well-suited SAT-based techniques for each setting. Specifically, we detailed a generic approach which, using two calls to a declarative solver, captures the Kemeny, Slater, MaxHamming, Young, Dodgson, and Reversal scoring rules, under which outcome determination is Θ_2^{P}, and detailed its instantiation through both MaxSAT and integer programming encodings. For the Δ_2^{P}-complete problem of outcome determination under the LexiMax rule, we developed an iterative approach based on Boolean multilevel optimization and instantiated it with both MaxSAT and integer programming solvers. Furthermore, for the Σ_2^{P}-complete Condorcet and Ranked agenda rules, we proposed a PrefSAT-based CEGAR approach. Our implementations are all available in open source. We also presented results from an empirical evaluation of the implementation on data from PrefLib (Mattei & Walsh, 2013), arising from real-world voting scenarios. The empirical results suggest that that our approaches scale noticeably better than JA-ASP, an earlier-proposed exact declarative approach to judgment aggregation that is based on answer set programming (de Haan & Slavkovik, 2019). In terms of the relative empirical performance of the MaxSAT-based and integer programming instantiations, the results suggest that the better choice of a declarative approach depends on the specific aggregation rule at hand. Overall, the empirical results obtained suggest that the approach is viable and could potentially even be employed as a backend e.g. for web-applications such as Whale (`https://whale5.noiraudes.net/`)

as well as in other application scenarios such as multiwinner voting (Chingoma et al., 2022) and abstract argumentation (Baumeister et al., 2021; Bodanza et al., 2017).

For directions for further work, studying ways of extending the approaches presented in this article to capturing further NP-hard judgment aggregation rules, such as the binomial rule (Costantini et al., 2016), the MaxEq rule (Botan et al., 2021), and rules based on geodesic distance (Duddy & Piggins, 2012; Endriss et al., 2020) would be of interest. Furthermore, it would be interesting to study so-called asymmetric variants of the Kemeny, Slater, and LexiMax rules (Rey et al., 2020) which could lead to extending our approach to cover approval-based participatory budgeting (Aziz & Shah, 2021) via judgment aggregation. Beyond outcome determination, there is promise for additionally developing declarative approaches for further problem settings in judgment aggregation and in computational social choice scenarios more generally. Finally, ways of further improving on the runtime performance of the approaches presented in this work could be studied, e.g., by developing alternative MaxSAT encodings, more tailored SAT-based solvers, or by employing column/constraint generation based IP solving techniques.

Acknowledgments

This work was financially supported by Research Council of Finland under grants 347588 and and 356046. The authors thank the Finnish Computing Competence Infrastructure (FCCI) for computational and data storage resources.

Appendix A. Details on Resource Limits and Sizes of MaxSAT Instances

Table 2 shows a distribution of time and memory outs for the MaxSAT-based and IP-based instantiations. Table 3 provides an overview of the sizes of the solved MaxSAT and IP instances produced according to the encodings detailed in Sections 4 and 6, respectively.

Table 2: MaxSAT-based vs IP-based instantiations: percentage of solved, timeouts (to), and memouts (mo).

	MaxSAT			**IP**		
Rule	solved (%)	to (%)	mo (%)	solved (%)	to (%)	mo (%)
Kemeny	93.8	5.9	0.2	85.2	0	14.8
Slater	93.6	6.2	0.2	82.2	3.0	14.8
MaxHamming	15.6	52.1	32.3	60.5	32.8	6.7
Young	99.8	0	0.2	100	0	0
Dodgson	82.5	2.0	15.6	63.0	8.1	28.9
LexiMax	92.1	3.2	4.7	84.2	1.5	14.3
Reversal scoring	64.7	32.8	2.5	—	—	—

Table 3: Sizes of solved MaxSAT (top) and IP (bottom) instances by JA rule.

MaxSAT	**Variables**			**Clauses**		
Rule	Min	Max	Median	Min	Max	Median
Kemeny	10	61,776	561	70	43,304,256	36,368.5
Slater	10	61,776	528	70	43,304,256	33,194
MaxHam.	224	63,082	7493	613	727,713	59,432
Young	81	3,403,798	28,626.5	355	18,996,868	199,996.5
Dodgson	285	747,270	52,085.5	1356	38,339,905	818,577
IP	**Variables**			**Constraints**		
Kemeny	10	13,203	351	60	4,251,366	17,550
Slater	10	13,041	325	60	4,173,120	15,600
MaxHam.	11	8386	172	67	2,146,570	5839
Young	32	335,253	1321	36	502,573	1927
Dodgson	95	47,619	5551	456	7,824,376	191,422

References

Argelich, J., Lynce, I., & Marques Silva, J. P. (2009). On solving boolean multilevel optimization problems. In Boutilier, C. (Ed.), *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pp. 393–398.

Arora, S., & Barak, B. (2009). *Computational Complexity: A Modern Approach*. Cambridge University Press.

Arrow, K. J. (2012). *Social Choice and Individual Values*. Yale University Press.

Arrow, K. J., Sen, A. K., & Suzumura, K. (Eds.). (2011). *Handbook of Social Choice and Welfare* (1 edition)., Vol. 2. Elsevier.

Arrow, K. J., Sen, A., & Suzumura, K. (2002). *Handbook of Social Choice and Welfare*. Elsevier.

Aziz, H., & Shah, N. (2021). Participatory budgeting: Models and approaches. In Rudas, T., & Péli, G. (Eds.), *Pathways Between Social Science and Computational Social Science: Theories, Methods, and Interpretations*, pp. 215–236. Springer International Publishing.

Bacchus, F., Järvisalo, M., & Martins, R. (2021). Maximum satisfiabiliy. In Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.), *Handbook of Satisfiability - Second Edition*, Vol. 336 of *Frontiers in Artificial Intelligence and Applications*, pp. 929–991. IOS Press.

Bailleux, O., & Boufkhad, Y. (2003). Efficient CNF encoding of boolean cardinality constraints. In Rossi, F. (Ed.), *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings*, Vol. 2833 of *Lecture Notes in Computer Science*, pp. 108–122. Springer.

Baumeister, D., Neugebauer, D., & Rothe, J. (2021). Collective acceptability in abstract argumentation. *fCoLog Journal of Logics and their Applications*, *8*(6), 1503–1542.

Bodanza, G. A., Tohmé, F., & Auday, M. (2017). Collective argumentation: A survey of aggregation issues around argumentation frameworks. *Argument & Computation*, *8*(1), 1–34.

Boes, L., Colley, R., Grandi, U., Lang, J., & Novaro, A. (2023). Collective combinatorial optimisation as judgment aggregation. *Annals of Mathematics and Artificial Intelligence*, *91*, 1–29.

Boixel, A., & Endriss, U. (2020). Automated justification of collective decisions via constraint solving. In Seghrouchni, A. E. F., Sukthankar, G., An, B., & Yorke-Smith, N. (Eds.), *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, pp. 168–176. International Foundation for Autonomous Agents and Multiagent Systems.

Botan, S., de Haan, R., Slavkovik, M., & Terzopoulou, Z. (2021). Egalitarian judgment aggregation. In Dignum, F., Lomuscio, A., Endriss, U., & Nowé, A. (Eds.), *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, pp. 214–222. ACM.

Brandl, F., Brandt, F., Eberl, M., & Geist, C. (2018). Proving the incompatibility of efficiency and strategyproofness via SMT solving. *Journal of the ACM*, *65*(2), 6:1–6:28.

Brandl, F., Brandt, F., Geist, C., & Hofbauer, J. (2019). Strategic abstention based on preference extensions: Positive results and computer-generated impossibilities. *Journal of Artificial Intelligence Research*, *66*, 1031–1056.

Brandl, F., Brandt, F., Peters, D., & Stricker, C. (2021). Distribution rules under dichotomous preferences: Two out of three ain't bad. In Biró, P., Chawla, S., & Echenique, F. (Eds.), *EC '21: The 22nd ACM Conference on Economics and Computation, Budapest, Hungary, July 18-23, 2021*, pp. 158–179. ACM.

Brandt, F., Chabin, G., & Geist, C. (2015). Pnyx: : A powerful and user-friendly tool for preference aggregation. In Weiss, G., Yolum, P., Bordini, R. H., & Elkind, E. (Eds.), *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pp. 1915–1916. ACM.

Brandt, F., Conitzer, V., Endriss, U., Lang, J., & Procaccia, A. D. (2016). Introduction to computational social choice. In Brandt, F., Conitzer, V., Endriss, U., Lang, J., & Procaccia, A. D. (Eds.), *Handbook of Computational Social Choice*, pp. 1–20. Cambridge University Press.

Brandt, F., & Geist, C. (2016). Finding strategyproof social choice functions via SAT solving. *Journal of Artificial Intelligence Research*, *55*, 565–602.

Brandt, F., Geist, C., & Peters, D. (2017). Optimal bounds for the no-show paradox via SAT solving. *Mathematical Social Sciences*, *90*, 18–27.

Brandt, F., Saile, C., & Stricker, C. (2018). Voting with ties: Strong impossibilities via SAT solving. In André, E., Koenig, S., Dastani, M., & Sukthankar, G. (Eds.), *Proceedings of*

the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018, pp. 1285–1293. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM.

Brandt, F., Saile, C., & Stricker, C. (2022). Strategyproof social choice when preferences and outcomes may contain ties. *Journal of Economic Theory*, *202*, 105447.

Charwat, G., & Pfandler, A. (2015). Democratix: A declarative approach to winner determination. In Walsh, T. (Ed.), *Algorithmic Decision Theory - 4th International Conference, ADT 2015, Lexington, KY, USA, September 27-30, 2015, Proceedings*, Vol. 9346 of *Lecture Notes in Computer Science*, pp. 253–269. Springer.

Chen, W., & Endriss, U. (2019). Preservation of semantic properties in collective argumentation: The case of aggregating abstract argumentation frameworks. *Artificial Intelligence*, *269*, 27–48.

Chingoma, J., Endriss, U., & de Haan, R. (2022). Simulating multiwinner voting rules in judgment aggregation. In Faliszewski, P., Mascardi, V., Pelachaud, C., & Taylor, M. E. (Eds.), *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022, Auckland, New Zealand, May 9-13, 2022*, pp. 263–271. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS).

Clarke, E. M., Grumberg, O., Jha, S., Lu, Y., & Veith, H. (2003). Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM*, *50*(5), 752–794.

Clarke, E. M., Gupta, A., & Strichman, O. (2004). SAT-based counterexample-guided abstraction refinement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *23*(7), 1113–1123.

Conati, A., Niskanen, A., & Järvisalo, M. (2023). SAT-based judgment aggregation. In Agmon, N., An, B., Ricci, A., & Yeoh, W. (Eds.), *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023*, pp. 1412–1420. IFAAMAS.

Conitzer, V. (2006). Computing slater rankings using similarities among candidates. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pp. 613–619. AAAI Press.

Conitzer, V., Davenport, A. J., & Kalagnanam, J. (2006). Improved bounds for computing Kemeny rankings. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pp. 620–626. AAAI Press.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In Harrison, M. A., Banerji, R. B., & Ullman, J. D. (Eds.), *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pp. 151–158. ACM.

Costantini, M., Groenland, C., & Endriss, U. (2016). Judgment aggregation under issue dependencies. In Schuurmans, D., & Wellman, M. P. (Eds.), *Proceedings of the Thirtieth*

AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA, pp. 468–474. AAAI Press.

Davenport, A. J., & Kalagnanam, J. (2004). A computational study of the Kemeny rule for preference aggregation. In McGuinness, D. L., & Ferguson, G. (Eds.), *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pp. 697–702. AAAI Press / The MIT Press.

de Haan, R., & Slavkovik, M. (2017). Complexity results for aggregating judgments using scoring or distance-based procedures. In Larson, K., Winikoff, M., Das, S., & Durfee, E. H. (Eds.), *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pp. 952–961. ACM.

de Haan, R., & Slavkovik, M. (2019). Answer set programming for judgment aggregation. In Kraus, S. (Ed.), *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 1668–1674. ijcai.org.

Delemazure, T., Demeulemeester, T., Eberl, M., Israel, J., & Lederer, P. (2023). Strategyproofness and proportionality in party-approval multiwinner elections. In Williams, B., Chen, Y., & Neville, J. (Eds.), *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pp. 5591–5599. AAAI Press.

Dietrich, F. (2014). Scoring rules for judgment aggregation. *Social Choice and Welfare*, *42*(4), 873–911.

Dietrich, F., & List, C. (2007). Arrow's theorem in judgment aggregation. *Social Choice and Welfare*, *29*(1), 19–33.

Dodaro, C., & Previti, A. (2019). Minipref: A tool for preferences in SAT (short paper). In Maratea, M., & Vallati, M. (Eds.), *Joint Proceedings of the RCRA International Workshop and of the RCRA Incontri e Confronti Workshop co-located with the 18th International Conference of the Italian Association for Artificial Intelligence (AIIA 2019), Rende, Italy, November 19-20, 2019*, Vol. 2538 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Drummond, J., Perrault, A., & Bacchus, F. (2015). SAT is an effective and complete method for solving stable matching problems with couples. In Yang, Q., & Wooldridge, M. J. (Eds.), *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 518–525. AAAI Press.

Duddy, C., & Piggins, A. (2012). A measure of distance between judgment sets. *Social Choice and Welfare*, *39*(4), 855–867.

Eén, N., & Sörensson, N. (2003). Temporal induction by incremental SAT solving. *Electronic Notes in Theoretical Computer Science*, *89*(4), 543–560.

Eén, N., & Sörensson, N. (2006). Translating pseudo-boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, *2*(1-4), 1–26.

Endriss, U. (2016). Judgment aggregation. In Brandt, F., Conitzer, V., Endriss, U., Lang, J., & Procaccia, A. D. (Eds.), *Handbook of Computational Social Choice*, pp. 399–426. Cambridge University Press.

Endriss, U. (2018). Judgment aggregation with rationality and feasibility constraints. In André, E., Koenig, S., Dastani, M., & Sukthankar, G. (Eds.), *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pp. 946–954. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM.

Endriss, U. (2020). Analysis of one-to-one matching mechanisms via SAT solving: Impossibilities for universal axioms. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 1918–1925. AAAI Press.

Endriss, U., & de Haan, R. (2015). Complexity of the winner determination problem in judgment aggregation: Kemeny, Slater, Tideman, Young. In Weiss, G., Yolum, P., Bordini, R. H., & Elkind, E. (Eds.), *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pp. 117–125. ACM.

Endriss, U., de Haan, R., Lang, J., & Slavkovik, M. (2020). The complexity landscape of outcome determination in judgment aggregation. *Journal of Artificial Intelligence Research*, *69*, 687–731.

Endriss, U., & Grandi, U. (2017). Graph aggregation. *Artificial Intelligence*, *245*, 86–114.

Endriss, U., Grandi, U., de Haan, R., & Lang, J. (2016). Succinctness of languages for judgment aggregation. In Baral, C., Delgrande, J. P., & Wolter, F. (Eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016*, pp. 176–186. AAAI Press.

Endriss, U., Grandi, U., & Porello, D. (2012). Complexity of judgment aggregation. *Journal of Artificial Intelligence Research*, *45*, 481–514.

Everaere, P., Konieczny, S., & Marquis, P. (2014). Counting votes for aggregating judgments. In Bazzan, A. L. C., Huhns, M. N., Lomuscio, A., & Scerri, P. (Eds.), *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, pp. 1177–1184. IFAAMAS/ACM.

Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., & Wanko, P. (2016). Theory solving made easy with clingo 5. In Carro, M., King, A., Saeedloei, N., & Vos, M. D. (Eds.), *Technical Communications of the 32nd International Conference on Logic Programming, ICLP 2016 TCs, October 16-21, 2016, New York City, USA*, Vol. 52 of *OASIcs*, pp. 2:1–2:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Geist, C., & Endriss, U. (2011). Automated search for impossibility theorems in social choice theory: Ranking sets of objects. *Journal of Artificial Intelligence Research*, *40*, 143–174.

Hemaspaandra, E., Hemaspaandra, L. A., & Rothe, J. (1997). Exact analysis of Dodgson elections: Lewis carroll's 1876 voting system is complete for parallel access to NP. In Degano, P., Gorrieri, R., & Marchetti-Spaccamela, A. (Eds.), *Automata, Languages and Programming, 24th International Colloquium, ICALP'97, Bologna, Italy, 7-11 July 1997, Proceedings*, Vol. 1256 of *Lecture Notes in Computer Science*, pp. 214–224. Springer.

Hemaspaandra, E., Spakowski, H., & Vogel, J. (2005). The complexity of Kemeny elections. *Theoretical Computer Science*, *349*(3), 382–391.

Ignatiev, A., Morgado, A., & Marques-Silva, J. (2018). PySAT: A Python toolkit for prototyping with SAT oracles. In Beyersdorff, O., & Wintersteiger, C. M. (Eds.), *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, Vol. 10929 of *Lecture Notes in Computer Science*, pp. 428–437. Springer.

Kluiving, B., de Vries, A., Vrijbergen, P., Boixel, A., & Endriss, U. (2020). Analysing irresolute multiwinner voting rules with approval ballots via SAT solving. In Giacomo, G. D., Catalá, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., & Lang, J. (Eds.), *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, Vol. 325 of *Frontiers in Artificial Intelligence and Applications*, pp. 131–138. IOS Press.

Lang, J., Pigozzi, G., Slavkovik, M., & van der Torre, L. W. N. (2011). Judgment aggregation rules based on minimization. In Apt, K. R. (Ed.), *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2011), Groningen, The Netherlands, July 12-14, 2011*, pp. 238–246. ACM.

Lang, J., Pigozzi, G., Slavkovik, M., van der Torre, L., & Vesic, S. (2017). A partial taxonomy of judgment aggregation rules and their properties. *Social Choice and Welfare*, *48*(2), 327–356.

Lang, J., & Slavkovik, M. (2013). Judgment aggregation rules and voting rules. In Perny, P., Pirlot, M., & Tsoukiàs, A. (Eds.), *Algorithmic Decision Theory - Third International Conference, ADT 2013, Bruxelles, Belgium, November 12-14, 2013, Proceedings*, Vol. 8176 of *Lecture Notes in Computer Science*, pp. 230–243. Springer.

Lang, J., & Slavkovik, M. (2014). How hard is it to compute majority-preserving judgment aggregation rules?. In Schaub, T., Friedrich, G., & O'Sullivan, B. (Eds.), *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, Vol. 263 of *Frontiers in Artificial Intelligence and Applications*, pp. 501–506. IOS Press.

Li, C. M., & Manyà, F. (2021). MaxSAT, hard and soft constraints. In Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.), *Handbook of Satisfiability - Second Edition*, Vol. 336 of *Frontiers in Artificial Intelligence and Applications*, pp. 903–927. IOS Press.

Lifschitz, V. (2019). *Answer Set Programming*. Springer.

List, C. (2012). The theory of judgment aggregation: an introductory review. *Synthesis*, *187*(1), 179–207.

Marques-Silva, J., Lynce, I., & Malik, S. (2021). Conflict-driven clause learning SAT solvers. In Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.), *Handbook of Satisfiability - Second Edition*, Vol. 336 of *Frontiers in Artificial Intelligence and Applications*, pp. 133–182. IOS Press.

Martins, R., Joshi, S., Manquinho, V. M., & Lynce, I. (2014). Incremental cardinality constraints for MaxSAT. In O'Sullivan, B. (Ed.), *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, Vol. 8656 of *Lecture Notes in Computer Science*, pp. 531–548. Springer.

Mattei, N., & Walsh, T. (2013). PrefLib: A library for preferences http://www.preflib.org. In Perny, P., Pirlot, M., & Tsoukiàs, A. (Eds.), *Algorithmic Decision Theory - Third International Conference, ADT 2013, Bruxelles, Belgium, November 12-14, 2013, Proceedings*, Vol. 8176 of *Lecture Notes in Computer Science*, pp. 259–270. Springer.

Meila, M., Phadnis, K., Patterson, A., & Bilmes, J. A. (2007). Consensus ranking under the exponential model. In Parr, R., & van der Gaag, L. C. (Eds.), *UAI 2007, Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, Vancouver, BC, Canada, July 19-22, 2007*, pp. 285–294. AUAI Press.

Miller, M. K., & Osherson, D. N. (2009). Methods for distance-based judgment aggregation. *Social Choice and Welfare*, *32*(4), 575–601.

Nehring, K., & Pivato, M. (2019). Majority rule in the absence of a majority. *Journal of Economic Theory*, *183*, 213–257.

Nehring, K., Pivato, M., & Puppe, C. (2014). The condorcet set: Majority voting over interconnected propositions. *Journal of Economic Theory*, *151*, 268–303.

Niskanen, A., Berg, J., & Järvisalo, M. (2022). Incremental maximum satisfiability. In Meel, K. S., & Strichman, O. (Eds.), *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, Vol. 236 of *LIPIcs*, pp. 14:1–14:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Peters, D. (2018). Proportionality and strategyproofness in multiwinner elections. In André, E., Koenig, S., Dastani, M., & Sukthankar, G. (Eds.), *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pp. 1549–1557. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM.

Pettit, P. (2001). Deliberative democracy and the discursive dilemma. *Philosophical Issues*, *11*, 268–299.

Pigozzi, G. (2006). Belief merging and the discursive dilemma: an argument-based account to paradoxes of judgment aggregation. *Synthesis*, *152*(2), 285–298.

Piotrów, M. (2020). UWrMaxSat: Efficient solver for MaxSAT and pseudo-boolean problems. In *32nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2020, Baltimore, MD, USA, November 9-11, 2020*, pp. 132–136. IEEE.

Porello, D., & Endriss, U. (2014). Ontology merging as social choice: judgment aggregation under the open world assumption. *Journal of Logic and Computation*, *24*(6), 1229–1249.

Prestwich, S. D. (2021). CNF encodings. In Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.), *Handbook of Satisfiability - Second Edition*, Vol. 336 of *Frontiers in Artificial Intelligence and Applications*, pp. 75–100. IOS Press.

Rey, S., Endriss, U., & de Haan, R. (2020). Designing participatory budgeting mechanisms grounded in judgment aggregation. In Calvanese, D., Erdem, E., & Thielscher, M. (Eds.), *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020*, pp. 692–702.

Rosa, E. D., Giunchiglia, E., & Maratea, M. (2010). Solving satisfiability problems with preferences. *Constraints*, *15*(4), 485–515.

Rothe, J., Spakowski, H., & Vogel, J. (2003). Exact complexity of the winner problem for Young elections. *Theory of Computing Systems*, *36*(4), 375–386.

Sinz, C. (2005). Towards an optimal CNF encoding of boolean cardinality constraints. In van Beek, P. (Ed.), *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, Vol. 3709 of *Lecture Notes in Computer Science*, pp. 827–831. Springer.

Tang, P., & Lin, F. (2009). Computer-aided proofs of Arrow's and other impossibility theorems. *Artificial Intelligence*, *173*(11), 1041–1053.

Tseitin, G. S. (1983). On the complexity of derivation in propositional calculus. In Siekmann, J. H., & Wrightson, G. (Eds.), *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970*, pp. 466–483. Springer Berlin Heidelberg, Berlin, Heidelberg.

Zwicker, W. S. (2016). Introduction to the theory of voting. In Brandt, F., Conitzer, V., Endriss, U., Lang, J., & Procaccia, A. D. (Eds.), *Handbook of Computational Social Choice*, pp. 23–56. Cambridge University Press.