SAT-based Judgment Aggregation

Ari Conati
University of Helsinki
Helsinki, Finland
ari.conati@helsinki.fi

Andreas Niskanen
University of Helsinki
Helsinki, Finland
andreas.niskanen@helsinki.fi

Matti Järvisalo
University of Helsinki
Helsinki, Finland
matti.jarvisalo@helsinki.fi

ABSTRACT

Judgment aggregation (JA) offers a generic formal logical framework for modeling various settings where agents must reach joint agreements through aggregating the preferences, judgments, or beliefs of individual agents by social choice mechanisms. In this work, we develop practical JA algorithms for outcome determination by harnessing Boolean satisfiability (SAT) based solvers as the underlying reasoning engines, leveraging on their ability to efficiently reason over logical representations incrementally. Concretely, we provide algorithms for outcome determination under a range of aggregation rules, using natural choices of SAT-based techniques adhering to the computational complexity of the problem for the individual rules. We also implement and empirically evaluate the approach using both synthetic and PrefLib data, showing that the approach can scale significantly beyond recently proposed alternative algorithms for JA.

KEYWORDS

judgment aggregation, outcome determination, Boolean satisfiability, maximum satisfiability, SAT with preferences

ACM Reference Format:
Ari Conati, Andreas Niskanen, and Matti Järvisalo. 2023. SAT-based Judgment Aggregation. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

Judgment aggregation offers a generic formal logical framework for modeling various settings where agents must reach joint agreements through aggregating the preferences, judgments, or beliefs of individual agents by social choice mechanisms [6, 29, 37, 48, 49]. As such, judgment aggregation captures various aggregation scenarios such as preference aggregation (including various voting scenarios) [14, 24, 30, 35, 43, 45, 54], graph aggregation [33], as well as further collective decision-making scenarios involving multiple agents [9, 13, 62].

Computing collective judgments, and in particular outcome determination in judgment aggregation, is computationally notoriously hard for various social choice mechanisms, being often NP-hard or even complete for higher levels of the polynomial hierarchy [21, 31, 32, 35, 46]. This complexity barrier makes it challenging to develop practical generic algorithmic approaches to outcome determination. In fact, beyond earlier-developed approaches for more specific settings—in particular for specific voting rules [17, 19, 53] and (web-based) tools for preference aggregation [11, 12]—a first more generic approach to judgment aggregation was only recently introduced [22]. The approach makes use of declarative solver technology for answer set programming (ASP) [47] based on encoding second-level complete judgment aggregation scenarios directly with disjunctive answer set programs.

Despite its generality and motivations as a practical algorithmic approach to judgment aggregation, the ASP-based approach was not empirically evaluated by the original authors [22]. As we will show in this paper, the implementation of the approach, although it makes use of state-of-the-art ASP solvers, does not unfortunately scale beyond very small data. However, the approach does serve as the main current baseline against which to evaluate new algorithmic approaches to judgment aggregation.

Taking on the challenge of developing increasingly-efficient practical algorithms for judgment aggregation, we develop judgment aggregation algorithms by harnessing Boolean satisfiability (SAT) [8] based solvers as the underlying reasoning engines. The main motivations for employing SAT-based techniques for JA are two-fold. (i) Judgment sets take the form of logical formulas for which SAT solvers are the reasoning engine of choice. In fact, SAT solving is arguably today a key technology for solving computationally hard real-world problems efficiently. (ii) By enabling high levels of incremental computations [28], SAT solvers have been employed in developing highly-efficient decision and optimization procedures for various real-world problems, including ones complete for the second level of the polynomial hierarchy, often surpassing in efficiency alternative direct approaches such as disjunctive ASP and quantified satisfiability (QBF solving); see e.g. [27, 41].

As outcome determination under various aggregation rules is complete for complexity classes on the second level of the polynomial hierarchy (in particular, complete for Θ_2^p, Σ_2^p, and Δ_2^p [32]), incremental SAT-based procedures—although non-trivial to develop—hold significant promise in scaling up further than the current state of the art in practical approaches to judgment aggregation.

Concretely, we provide algorithms for outcome determination under a range of aggregation rules, using the most natural choice of SAT-based techniques adhering to the computational complexity of the problem for the individual rules. In terms of aggregation rules, we cover a wide range of the most central ones, including Kemeny, Slater, Young, Dodgson, MaxHamming, Reversal scoring, Condorcet, Ranked agenda, and LexiMax. In terms of SAT-based techniques, the procedures we develop for these settings make use of recent advances in incremental maximum satisfiability (MaxSAT) solving [57, 60] and preferential SAT-based reasoning [25]. In particular, we develop a generic MaxSAT-based approach covering judgment aggregation under Kemeny, Slater, MaxHamming, Young, and Dogdson; and show how to capture the other judgment aggregation rules through incremental MaxSAT and PrefSAT-based

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

counterexample-guided abstraction refinement [15, 16]. For practical applicability, we provide an implementation of the procedures developed here for outcome determination and empirically evaluate the approach using both synthetic and PrefLib data. Our approach scales significantly beyond the reach of the recently proposed alternative ASP-based algorithms for judgment aggregation. Emphasizing the general applicability of SAT-based approaches, we also outline incremental SAT-based algorithms for the related tasks of manipulation, bribery and control, using the Kemeny rule as a specific instantiation.

2 PRELIMINARIES

We start by overviewing background on judgment aggregation. We follow definitions from a recent taxonomy on judgment aggregation rules [43] and recall the computational complexity of outcome determination [32] as the main problem focused on in this work.

Judgment Aggregation. Consider a set $X = \{x_1, \ldots, x_m\}$ of propositional variables called *issues*, and let $\overline{X} = \{\neg x_1, \ldots, \neg x_m\}$. The set of literals $\Phi = X \cup \overline{X}$ is called the *agenda*. A *judgment set* $J \subseteq \Phi$ represents an individual opinion on the agenda. The judgment set J is *complete* if for all $x \in X$ either $x \in J$ or $\neg x \in J$ (but not both), and Γ-*consistent* with respect to a propositional formula Γ if $\Gamma \wedge \bigwedge_{l \in J} l$ is satisfiable. We interchangeably represent complete and consistent judgment sets as satisfying truth assignments to Γ, mapping each issue $x \in X$ to 1 if $x \in J$ and to 0 otherwise. Let $\mathcal{J}(\Phi, \Gamma)$ be the collection of all complete and Γ-consistent judgment sets.

Definition 2.1. 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 called the *input* and *output constraints*, which may contain variables outside Φ, and $P = (J_1, \ldots, J_n)$ is a *profile* consisting of *judgment sets* $J_i \in \mathcal{J}(\Phi, \Gamma_{\text{in}})$ representing the opinions of *individual* agents.[1]

For any $l \in \Phi$, we denote by $N(P, l) = |\{J_i \in P \mid l \in J_i\}|$ the number of supporters of agenda item l. The *majoritarian judgment set* is then defined as $J_{\text{m}}(P) = \{l \in \Phi \mid N(P, l) > n/2\}$. Note that this judgment set is not guaranteed to be Γ_{out}-consistent even if $\Gamma_{\text{out}} = \Gamma_{\text{in}}$; this is generally known as the discursive dilemma [58].

Judgment Aggregation Rules. A *judgment aggregation rule* R maps each profile P to a collection of *collective* judgment sets $R(P)$. For the following, the Hamming distance between complete judgment sets J and J' is defined as $H(J, J') = |J \setminus J'| = |J' \setminus J|$.

We briefly recall the central judgment aggregation rules considered in this work. We start with rules based on the majoritarian set [43]: the Condorcet and Slater rules.

Condorcet [30, 32, 44, 56]. The Condorcet rule maximizes the agreement with the majoritarian judgment set in a subset-wise sense. That is, $\textsc{Condorcet}(P)$ selects those $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ which $J \cap J_{\text{m}}(P)$ is subset-maximal, i.e., there is no $J' \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ with $J' \cap J_{\text{m}}(P) \supset J \cap J_{\text{m}}(P)$.

[1]Note that our framework for judgment aggregation is as general as the general definition (see e.g. [32]) where the agenda $\Phi = \{\varphi_1, \neg\varphi_1, \ldots, \varphi_m, \neg\varphi_m\}$ contains arbitrary formulas instead of literals. This can be seen by reasoning similarly to [34]: for every non-negated formula $\varphi_j \in \Phi$, construct the formula $x_j \leftrightarrow \varphi_j$, where x_j is a fresh variable, and add it to the input and output constraints. The resulting framework with issues as variables $X = \{x_1, \ldots, x_m\}$ is equivalent to the formula-based framework.

Slater [30, 44, 54, 56] also maximizes the agreement with the majoritarian judgment set, but in terms of the cardinality. Formally, $\textsc{Slater}(P)$ selects those $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ which maximize $|J \cap J_{\text{m}}(P)|$.

Next, we cover rules based on the number of supporters $N(P, \cdot)$, also called the weighted majoritarian set [43]: the Kemeny, Ranked agenda, and LexiMax rules. As a distance-based rule related to Kemeny, we also recall the MaxHamming rule [32].

Kemeny [30, 35, 44, 54, 56, 59] maximizes the agreement with the profile P, hence minimizing the sum of the Hamming distances to the judgment sets in P. Formally, $\textsc{Kemeny}(P)$ selects those $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ which maximize $\sum_{l \in J} N(P, l)$ (or equivalently, minimize $\sum_{J_i \in P} H(J, J_i)$).

MaxHamming [44]. In contrast to minimizing the sum of Hamming distances, the MaxHamming rule minimizes the maximum Hamming distance to judgment sets in P, selecting those $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ which minimize $\max_{J_i \in P} H(J, J_i)$.

The Ranked agenda and LexiMax rules are based on different preference orders over judgment sets. Towards the formal definitions, let $L_k^P = \{l \in \Phi \mid N(P, l) = k\}$.

Ranked agenda [31, 44, 61] is based on the ordering of agenda items by their support. As long as the resulting judgment set remains consistent, we iteratively include items based on this ordering, breaking ties nondeterministically. An equivalent formal definition is based on an ordering $\succ_{\text{RA}}$. We have that $J \succ_{\text{RA}} J'$ if there is a k with $n/2 \leq k \leq n$ such that $J \cap L_k^P \supset J' \cap L_k^P$, and for all $j > k$, $J \cap L_j^P = J' \cap L_j^P$. Now $\text{RA}(P)$ consists of $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ which are $\succ_{\text{RA}}$-maximal, i.e., there is no $J' \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ with $J' \succ_{\text{RA}} J$.

LexiMax [36, 55] is based on a lexicographic ordering $\succ_{\text{lex}}$ of judgment sets. We say that $J \succ_{\text{lex}} J'$ if there is a k with $n/2 \leq k \leq n$ such that $|J \cap L_k^P| > |J' \cap L_k^P|$, and for all $j > k$, $|J \cap L_j^P| = |J' \cap L_j^P|$. Now $\textsc{LexiMax}(P)$ consists of $\succ_{\text{lex}}$-maximal $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$, i.e., for which there is no $J' \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ with $J' \succ_{\text{lex}} J$.

Furthermore, we consider the Young and Dodgson rules which are based on modifications to the input profile [43]. Computationally, the goal here is to minimize the number of modifications in such a way that the majoritarian judgment set becomes Γ_{out}-consistent. Let $\mathcal{P}(\Phi, \Gamma)$ be the collection of all complete and Γ-consistent *profiles* over Φ.

Young [44] selects those complete and Γ_{out}-consistent judgment sets which are obtained as supersets of majoritarian judgment sets of profiles from which the least possible number of agents are removed. That is, $\textsc{Young}(P)$ first selects all profiles $P' \in \mathcal{P}(\Phi, \Gamma_{\text{in}})$ with $P' \subseteq P$ for which $J_{\text{m}}(P')$ is Γ_{out}-consistent, maximizing $|P'|$, and then all $J \in \mathcal{J}(P, \Gamma_{\text{out}})$ for which $J \supseteq J_{\text{m}}(P')$.

Dodgson [54]. The Dodgson rule selects those complete and Γ_{out}-consistent judgment sets which are obtained as majoritarian judgment sets of profiles in which the least possible number of opinions are reverted. Formally, $\textsc{Dodgson}(P)$ first selects all profiles $P' \in \mathcal{P}(\Phi, \Gamma_{\text{in}})$ with $|P'| = |P|$ for which $J_{\text{m}}(P')$ is Γ_{out}-consistent, minimizing $\sum_{i=1}^{n} H(J_i, J_i')$, and then all $J \in \mathcal{J}(P, \Gamma_{\text{out}})$ for which $J \supseteq J_{\text{m}}(P')$.

Finally, the **reversal scoring rule [23]** selects $J \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$ that maximize the reversal score $\sum_{J_i \in P} \sum_{l \in J} R(J_i, l)$ where $R(J_i, l)$ is the least number of issues on which judgment has to be reverted in J_i in order to reject issue l, that is, the minimum Hamming distance $H(J_i, J')$ for $J' \in \mathcal{J}(\Phi, \Gamma_{\text{out}})$, $l \notin J'$.

Outcome Determination. Our main focus is on the outcome determination problem. Outcome determination asks to decide whether a subset of the agenda is included in some collective judgment set returned by the given judgment aggregation rule. Formally, the input consists of a judgment aggregation framework $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$, a set $L \subseteq \Phi$, and a judgment aggregation rule R. The task is to decide whether there is a judgment set $J \in R(P)$ for which $L \subseteq J$ holds.[2] The algorithms developed in this work solve the search variant of the outcome determination problem, i.e., the algorithm will provide a witnessing judgment set if one exists.

For all of the rules we consider, it is known that the decision problem version of outcome determination lies on the second level of the polynomial hierarchy. In particular, outcome determination is Θ_2^p-complete for Kemeny, Slater, Young, Dodgson, MaxHamming, and Reversal scoring; Σ_2^p-complete for Condorcet and Ranked agenda, and Δ_2^p-complete for LexiMax [32].

SAT and Boolean Optimization. The algorithms developed in this work for outcome determination make use of Boolean satisfiability (SAT) and maximum satisfiability (MaxSAT) solvers, which are iteratively called (in different ways, depending on the judgment aggregation rule at hand) under rule-dependent declarative encodings. To this end, we introduce necessary preliminaries on SAT and MaxSAT.

SAT. For a Boolean variable x there are two literals, x and $\neg x$. A clause C is a disjunction ($\vee$) of literals. A conjunctive normal form (CNF) formula F is a conjunction ($\wedge$) of clauses. For convenience we view clauses as sets of literals and formulas as sets of clauses. We denote by $V(F)$ and $L(F)$ the set of variables and literals of F, respectively. A truth assignment $\tau\colon V(F) \to \{0,1\}$ maps each variable to 0 (false) or 1 (true), and is extended to literals via $\tau(\neg x) = 1 - \tau(x)$, to clauses via $\tau(C) = \max\{\tau(l) \mid l \in C\}$, and to formulas via $\tau(F) = \min\{\tau(C) \mid C \in F\}$. We interchangeably represent truth assignments τ as sets of non-contradictory literals: $\{l \in L(F) \mid \tau(l) = 1\}$. The *Boolean satisfiability* problem (SAT) asks for an input formula F whether there is an assignment τ with $\tau(F) = 1$. In this case we say F is satisfiable, and otherwise F is unsatisfiable.

MaxSAT. In the *maximum satisfiability* problem (MaxSAT) [2], the input consists of "hard" clauses F_{hard}, "soft" clauses F_{soft}, and a weight function $w\colon F_{\text{soft}} \to \mathbb{Z}_+$. The task is to find a truth assignment τ which satisfies F_{hard} and minimizes the cost $c(\tau) = \sum_{C \in F_{\text{soft}}} w(C)(1 - \tau(C))$ incurred by not satisfying soft clauses.

3 DIRECT MAXSAT APPROACHES

We use MaxSAT as a suitable declarative paradigm to cover rules for which outcome determination is Θ_2^p-complete, that is, Kemeny, Slater, MaxHamming, Young, and Dodgson (we will consider reversal scoring later on). For each of these rules, outcome determination can be solved by the following generic algorithm, using two MaxSAT solver calls. Let $(\Phi, \Gamma_{\text{in}}, \Gamma_{\text{out}}, P)$ be a given judgment aggregation framework, $L \subseteq \Phi$ an input subset of the agenda for outcome determination, and $R \in \{\textsc{Kemeny}, \textsc{Slater}, \textsc{MaxHamming}, \textsc{Young}, \textsc{Dodgson}\}$ a judgment aggregation rule.

(1) Encode the judgment aggregation rule R in MaxSAT, that is, 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 $\tau \cap \Phi = J$.
(2) Compute the optimal cost c^* of the instance $F_R(P)$ by invoking a MaxSAT solver.
(3) Modify the MaxSAT instance $F_R(P)$ by adding $\bigwedge_{l \in L} l$ to the set of hard clauses. If there are no solutions or if the optimal cost $c^L > c^*$, there is no $J \in R(P)$ with $L \subseteq J$; return **false**. Otherwise (i.e., if $c^L = c^*$), return $\tau \cap \Phi$, where τ is the optimal MaxSAT solution obtained from the second MaxSAT solver invocation.[3]

What remains is to describe the MaxSAT encodings F_R for the individual judgment aggregation rules R. For each of the MaxSAT encodings, we initialize the hard clauses F_{hard} as the output constraint Γ_{out} to ensure that all satisfying assignments τ to F_{hard} represent complete and Γ_{out}-consistent judgment sets $J = \tau \cap \Phi$. 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; these cardinality constraints are encoded as clauses using readily-available CNF encodings [40].

3.1 Kemeny and Slater

We encode the Kemeny rule in MaxSAT by introducing for each $l \in J_{\text{m}}(P)$ a soft clause (l) with weight $N(P,l) - N(P,\neg l)$. 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. Similarly, the Slater rule is encoded in MaxSAT by introducing for each $l \in J_{\text{m}}(P)$ a soft clause (l) with unit weight. Summarizing, let $F_{\textsc{Kemeny}}(P)$ and $F_{\textsc{Slater}}(P)$ be MaxSAT instances with the hard clauses Γ_{out} and a soft clause (l) for each $l \in J_{\text{m}}(P)$, and with $w_{\textsc{Kemeny}}(l) = N(P,l) - N(P,\neg l)$ and $w_{\textsc{Slater}}(l) = 1$.

3.2 MaxHamming

For encoding the MaxHamming rule, we declare fresh variables p_k for each $k = 1, \ldots, m$ with the interpretation "$\tau(p_k) = 1$ if the maximum Hamming distance between the input judgment sets and the output $\max_{J_i \in P} H(J, J_i)$ is at least k". For $k = 1, \ldots, m$, the formula

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

enforces that p_k is set to true if any agent disagrees with the output at least by k issues. The additional implications $p_{k+1} \rightarrow p_k$ for each $k = 1, \ldots, m-1$ then ensure that $\tau(p_k) = 1$ implies $\tau(p_{k'}) = 1$ for all $k' < k$, i.e., that if there is a disagreement by at least k issues, then there is a disagreement by at least $k' < k$ issues. To minimize the maximum Hamming distance, we introduce the soft clauses $(\neg p_k)$ with unit weights. Summarizing, $F_{\textsc{MaxHamming}}(P)$ has hard

[2] Our definition of outcome determination captures the general variant defined by [32], 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 by constructing for each $i = 1, \ldots, u$ a formula $o_i \leftrightarrow \bigwedge_{l \in L_i} l$, where o_i is a fresh variable, adding it to the output constraint Γ_{out}, and considering the set $L \cup \{\neg o_1, \ldots, \neg o_u\}$ as input.

[3] As an optimization, we note that the second MaxSAT solver call can be replaced by a SAT solver call on the hard clauses of the corresponding MaxSAT instance, $\bigwedge_{l \in L} l$, and a cardinality constraint which enforces that the MaxSAT cost of any satisfying truth assignment to this SAT instance is exactly c^*.

clauses $\Gamma_{\text{out}} \wedge \bigwedge_{k=1}^{m} \textsc{disagreement}_k(P) \wedge \bigwedge_{k=1}^{m-1}(p_{k+1} \rightarrow p_k)$ and soft clauses $(\neg p_k)$ with $w(\neg p_k) = 1$ for each $k = 1, \ldots, m$.

3.3 Young

For encoding the Young rule, we introduce variables y_i for each $i = 1, \ldots, n$ with the interpretation that $\tau(y_i) = 1$ iff J_i is included in the modified profile. For $j = 1, \ldots, m$, the formula

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

enforces that issue x_j is included in the collective judgment set if a strict majority of the modified profile supports it. This means that for each $k = 1, \ldots, n$, at most k judgment sets are included in the modified profile and there are at least $\lfloor k/2 \rfloor + 1$ judgment sets which support the issue. Similarly, $\textsc{support}_j^-(P)$ defined as

$$\bigwedge_{k=1}^{n} \left(\left(\sum_{i=1}^{n} y_i \geq k \wedge \sum_{\substack{i=1,\ldots,n \\ x_j \in J_i}} y_i \leq \lceil k/2 \rceil - 1 \right) \rightarrow \neg x_j \right)$$

enforces that if a strict majority of the modified profile supports $\neg x_j$, we must have $x_j = 0$ in the collective judgment set. To minimize the number of removed agents, we use (y_i) for each $i = 1, \ldots, n$ as a soft clause with unit weight. To summarize, $F_{\textsc{Young}}(P)$ contains hard clauses $\Gamma_{\text{out}} \wedge \bigwedge_{j=1}^{m}(\textsc{support}_j^+(P) \wedge \textsc{support}_j^-(P))$ and soft clauses (y_i) with $w(y_i) = 1$ for each $i = 1, \ldots, n$.

3.4 Dodgson

The MaxSAT encoding of the Dodgson rule is somewhat similar to that of Young. We declare variables x_{ij} for each $i = 1, \ldots, n$ and $j = 1, \ldots, m$ with the interpretation that $\tau(x_{ij}) = 1$ iff in the modified profile, the ith judgment set supports issue x_j. Hard clauses are used to enforce that for each $i = 1, \ldots, n$, the set $\{x_{ij} \mid j = 1, \ldots, m\}$ represents a Γ_{in}-consistent judgment set via $\Gamma_{\text{in}}^i = \Gamma_{\text{in}}[x_j \mapsto x_{ij} \mid j = 1, \ldots, m]$. To express that issues must be set according to the majority of the modified profile, $\textsc{support}_j(P)$ defined as

$$\left(\left(\sum_{i=1}^{n} x_{ij} \geq \lfloor n/2 \rfloor + 1 \rightarrow x_j \right) \wedge \left(\sum_{i=1}^{n} x_{ij} \leq \lceil n/2 \rceil - 1 \rightarrow \neg x_j \right) \right),$$

enforces that x_j (resp. $\neg x_j$) is included in the collective judgment set if it is supported by the strict majority of the modified profile. To summarize, $F_{\textsc{Dodgson}}(P)$ contains hard clauses $\Gamma_{\text{out}} \wedge \bigwedge_{i=1}^{n} \Gamma_{\text{in}}^i \wedge \bigwedge_{j=1}^{m} \textsc{support}_j$, 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, to ensure the least number of changes to the input profile.

The correctness of the encodings is established as follows, directly implying the correctness of the MaxSAT-based algorithm outlined above for outcome determination.

Proposition 3.1. *Let $(\Phi, \Gamma_{in}, \Gamma_{out}, P)$ be a given judgment aggregation framework. For each $R \in \{$Kemeny, Slater, MaxHamming, Young, Dodgson$\}$, we have that $J \in R(P)$ if and only if there is an optimal solution τ to the MaxSAT instance $F_R(P)$ with $\tau \cap \Phi = J$.*

Algorithm 1 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: $\quad\quad R(J_i, l) \leftarrow c$; $R(J_i, \neg l) \leftarrow 0$;
7: $w \leftarrow \{l \mapsto \sum_{J_i \in P} R(J_i, l) \mid l \in \Phi\}$
8: $(c^*, \tau) \leftarrow \textsc{MaxSAT}(\Gamma_{\text{out}}, S, w)$
9: $(c^L, \tau) \leftarrow \textsc{MaxSAT}(\Gamma_{\text{out}} \wedge \bigwedge_{l \in L} l, S, w)$
10: **if** $c^* = c^L$ **return** $\tau \cap \Phi$ **else return false**

4 ITERATIVE SAT-BASED APPROACHES

While the so-far considered judgment aggregation rules are captured through a generic algorithm outlined in the previous section, the remaining judgment aggregation rules considered in this work, namely, Reversal scoring, Condorcet, Ranked agenda, and LexiMax, call for more intricate SAT-based techniques. As described in this section, we develop an incremental MaxSAT approach to Reversal scoring; PrefSAT-based counterexample-guided abstraction refinement (CEGAR) for Condorcet and Ranked agenda; and a MaxSAT-based lexicographic optimization approach to LexiMax.

4.1 Reversal Scoring

First we consider outcome determination under the reversal scoring rule, which is a Θ_2^p-complete problem. An adaptation of the direct MaxSAT-based algorithm covered in Section 3 would require a monolithic MaxSAT encoding consisting of i) an encoding of a judgment set $J_i^l \in \mathcal{J}(P, \Gamma_{\text{out}})$ for each $i = 1, \ldots, n$ and $l \in \Phi$, ii) minimizing $H(J_i, J_i^l)$ to compute $R(J_i, l)$ within the encoding, iii) finally minimizing the reversal score as the sum of $R(J_i, l)$. In contrast to such a complex direct encoding, we propose an iterative MaxSAT approach where the reversal score is computed via a series of simpler MaxSAT solver calls.

The algorithm for reversal scoring is presented in pseudocode as Algorithm 1. We begin by initializing a soft clause (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 slightly different MaxSAT queries (lines 2–6). First, for each $J_i \in P$, we consider soft clauses (l) with $l \in J_i$ with unit weights, and the rest with weight zero. This minimizes the Hamming distance to J_i. Then, we iterate through $l \in J_i$ (lines 4–6), solve a MaxSAT instance with hard clauses $\Gamma_{\text{out}} \wedge \neg l$ (line 5), and set $R(J_i, l)$ as the cost of the optimal MaxSAT solution (line 6). Note that $R(J_i, \neg l)$ is trivially zero. Finally, we set the weight of each soft clause (l) according to just-computed values of $R(J_i, l)$ (line 7), and solve a MaxSAT instance with Γ_{out} as hard clauses, obtaining the optimal reversal score c^* (line 8). Now, similarly to rules covered in Section 3, we solve one more MaxSAT instance with hard clauses $\Gamma_{\text{out}} \wedge \bigwedge_{l \in L} l$ (line 9). If the cost remains the same, we return the obtained truth assignment as the collective judgment set which includes L, and otherwise we know that no such set exists (line 10).

The algorithm can be implemented using a single incremental MaxSAT solver [57]. After initializing the MaxSAT solver with

Algorithm 2 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: $(result, \tau) \leftarrow \textsc{PrefSAT}(F \wedge \bigwedge_{l \in L} l, w)$
5: **if** $result = unsat$ **then return false**
6: $(result, \tau) \leftarrow \textsc{PrefSAT}(F \wedge \bigwedge_{l \in J_{\text{m}}(P) \cap \tau} l \wedge \bigvee_{l \in J_{\text{m}}(P) \setminus \tau} l, w)$
7: **if** $result = unsat$ **then return** $\tau \cap \Phi$
8: $F \leftarrow F \wedge \bigvee_{l \in J_{\text{m}}(P) \setminus \tau} l$

the output constraint Γ_{out} as hard clauses, every MaxSAT query involves changing weights of soft clauses or different assumptions (i.e. partial assignments), that is, operations which are supported by state-of-the-art incremental MaxSAT solvers. The correctness of this algorithm is based on the fact that optimal solutions of the constructed MaxSAT instance correspond to judgment sets under the reversal scoring rule.

Proposition 4.1. *Let $(\Phi, \Gamma_{in}, \Gamma_{out}, P)$ be a given judgment aggregation framework. Let F_{REV} be the MaxSAT instance constructed by line 8 of Algorithm 1. We have that $J \in \textsc{ReversalScore}(P)$ if and only if there is an optimal solution τ to the MaxSAT instance F_{REV} with $\tau \cap \Phi = J$.*

4.2 Condorcet

Outcome determination under the Condorcet rule is Σ_2^p-complete, which suggests developing a SAT-based CEGAR algorithm for this problem. For intuition on the generic CEGAR approach [15], an abstraction, i.e., an overapproximation of the set of solutions, is first constructed. Then, we iteratively obtain candidate solutions by solving the abstraction, and check their validity by checking for a counterexample. If there are no counterexamples, we have found an actual solution to the problem. Otherwise, we refine the search space by analyzing the counterexample.

We make use of a modified SAT solver, i.e., a PrefSAT solver to directly compute judgment sets which agree with the majority subset-maximally. In the *SAT with preferences* problem (PrefSAT) [25, 63], the input is a CNF formula F with a weight function $w\colon L(F) \to \mathbb{Z}_+$. For a set of literals L and an integer k, denote $\Lambda_k^L = \{l \in L \mid w(l) = k\}$. Now, a truth assignment τ is preferred to another truth assignment τ', denoted by $\tau \succ \tau'$, if there is an integer $k > 0$ with $\Lambda_k^\tau \supset \Lambda_k^{\tau'}$ and for all $i > k$ it holds that $\Lambda_i^\tau = \Lambda_i^{\tau'}$. The task is to find a satisfying assignment τ to F so that there is no satisfying assignment τ' to F with $\tau' \succ \tau$. Our algorithm for outcome determination under the Condorcet rule is presented as Algorithm 2, and is based on the following observation.

Proposition 4.2. *Let $(\Phi, \Gamma_{in}, \Gamma_{out}, P)$ be a given judgment aggregation framework. Define the preferences $w(l) = 1$ for each $l \in J_m(P)$. Now $J \in \textsc{Condorcet}(P)$ if and only if there is a solution τ to the PrefSAT instance Γ_{out} and preferences w with $\tau \cap \Phi = J$.*

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).

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

Then, we enter the main CEGAR loop (lines 3–8). Within this loop, we first query a PrefSAT solver for a candidate solution including L (line 4); if the PrefSAT solver reports unsatisfiability, we know that no such judgment set exists (line 5). 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 query a PrefSAT solver for a counterexample, checking whether removing L from the query results in a judgment set J' with $J' \cap J_{\text{m}}(P) \supset J \cap J_{\text{m}}(P)$ (line 6). If there are no counterexamples, we return the obtained truth assignment as the collective judgment set which includes L (line 7). Otherwise this iteration was not successful, so we block all assignments which are at least as close to the majority as the counterexample (line 8), and continue.

4.3 Ranked Agenda

Similarly as for the case of the Condorcet rule, we propose a PrefSAT-based CEGAR approach for the Σ_2^p-complete problem of outcome determination under the ranked agenda rule. However, in contrast to the Condorcet rule, instead of having a single preference level encoding subset-maximality, the preference level of each literal with majority support is set as the number of supporters. The PrefSAT-based algorithm for the ranked agenda rule is presented as Algorithm 3, and is based on the following observation.

Proposition 4.3. *Let $(\Phi, \Gamma_{in}, \Gamma_{out}, P)$ be a given judgment aggregation framework. Define the preferences $w(l) = N(P, l)$ for each $l \in \Phi$ with $N(P, l) > n/2$. Now $J \in RA(P)$ if and only if there is a solution τ to the PrefSAT instance Γ_{out} and preferences w with $\tau \cap \Phi = J$.*

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). Then, we enter a CEGAR loop (lines 3–8), and query a PrefSAT solver for an assignment which includes L (line 4). If the result is unsatisfiable, we know that no such judgment set exists (line 5). Otherwise, we obtain a truth assingment τ corresponding to a judgment set J which is $\succ_{\text{RA}}$-maximal under the constraint that $L \subseteq J$. Thus, we need to check whether a counterexample exists, that is, whether removing L results in a judgment set $J' \succ_{\text{RA}} J$. We encode this relation as follows. We let

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

where for each k, $\textsc{Level}_k^\tau = \bigwedge_{l \in \tau \cap \Lambda_k} l$, $\textsc{More}_k^\tau = \bigvee_{l \in \neg\tau \cap \Lambda_k} l$, $\textsc{EqUpTo}_k^\tau = \textsc{EqUpTo}_{k+1}^\tau \wedge \textsc{Level}_k^\tau \wedge \neg\textsc{More}_k^\tau$ for $k < n$, and

Algorithm 4 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** $k = n, \ldots, \lfloor n/2 \rfloor + 1$ **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** $k = \lfloor n/2 \rfloor + 1$ **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**

$\textsc{EqUpTo}_n^\tau = \textsc{Level}_n^\tau \wedge \neg\textsc{More}_n^\tau$. Here $\textsc{Level}_k^\tau$ enforces that all preferences with weight k satisfied by τ are satisfied, $\textsc{More}_k^\tau$ enforces that there is a preference with weight k not satisfied by τ but which is satisfied, and $\textsc{EqUpTo}_{k+1}^\tau$ enforces that all preferences with weight greater than k are satisfied exactly as in τ. Finally, $\textsc{MorePref}(\tau)$ states that there is such a weight k via a disjunction. A PrefSAT call with this additional formula then asks whether there is a judgment set $J' >_{\text{RA}} J$ (line 6). If the result is unsatisfiable, we return the candidate truth assignment τ as the collective judgment set which includes L. Otherwise, this iteration was unsuccessful, so we block all assignments corresponding to judgment sets J' which are at least as preferred as J, i.e., $J >_{\text{RA}} J'$ or $J = J'$ (line 8). Now, $J >_{\text{RA}} J'$ is encoded by the constraint

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

stating that there is a weight k for which a strict subset of preferences is satisfied (via $\neg\textsc{Level}_k^\tau \wedge \neg\textsc{More}_k^\tau$), while all preferences with weight greater than k are satisfied exactly according to the current assignment (via $\textsc{EqUpTo}_{k+1}^\tau$).

4.4 LexiMax

Finally, we consider the Δ_2^p-complete outcome determination problem under the LexiMax rule, and develop a MaxSAT-based lexicographic optimization approach for the problem. An instance of the Boolean lexicographic optimization (BLO) problem [1, 50] consists of a formula F and a sequence of objective functions $(c_1, \ldots, c_m)$. The goal is to find a truth assignment τ satisfying F and minimizing $(c_1(\tau), \ldots, c_m(\tau))$ in the lexicographic sense. This means that there is no satisfiable truth assignment τ' for F for which $c_q(\tau') < c_q(\tau)$, where $q = \min\{k = 1, \ldots, m \mid c_k(\tau) \neq c_k(\tau')\}$. According to the following observation, given a judgment aggregation framework, judgment sets under the LexiMax are in a one-to-one correspondence with a specific instance of BLO.

Proposition 4.4. *Let $(\Phi, \Gamma_{in}, \Gamma_{out}, P)$ be a given judgment aggregation framework. Now $J \in \textsc{LexiMax}(P)$ if and only if there is a solution τ to the BLO instance Γ_{out} with the objective function $(c_{\lfloor n/2 \rfloor + 1}, \ldots, c_n)$, where $c_k(\tau) = \sum_{l \in L_k^P}(1 - \tau(l))$ for each $k = \lfloor n/2 \rfloor + 1, \ldots, n$, with $\tau \cap \Phi = J$.*

Our algorithm for outcome determination under the LexiMax rule is an iterative MaxSAT BLO approach [1, 50], presented as Algorithm 4. We initialize a formula with the output constraint Γ_{out} (line 1). Now, starting from $k = n$, we declare (l) for each $l \in L_k^P$ as a soft clause with unit weight (line 3), and solve the corresponding MaxSAT problem (line 5), obtaining an optimal cost c_k. This gives us the number of soft clauses falsified by the optimal solution on level k. We add the cardinality constraint $\sum_{l \in L_k^P} \neg l = c_k$ (line 6), and iteratively continue by decrementing k. We proceed until k is the smallest possible level $\lfloor n/2 \rfloor + 1$, when we exit the loop (line 4). At this point, the optimal solutions to the current MaxSAT instance correspond exactly to LexiMax judgment sets. Calling the MaxSAT solver gives such a set (line 7). We call the MaxSAT solver once more to obtain the optimal cost c^L under the constraint that $L \subseteq J$ (line 8). If the cost remains the same, we return the obtained truth assignment as the LexiMax judgment set which includes L, and otherwise we know that no such set exists (line 9).

5 EMPIRICAL EVALUATION

Our implementation of the algorithms in Sections 3–4, named SATcha, and empirical data are available at https://bitbucket.org/coreo-group/satcha. We use the incremental MaxSAT solvers UWrMaxSAT [60] and iMaxHS [57], and MiniPref [25] as the PrefSAT solver for the Condorcet and ranked agenda rules. For encoding cardinality constraints into CNF, we adjusted the iterative totalizer encoding [3, 51] from PySAT [40].

We compare the runtimes of our approach to those of the JA-ASP [22] (https://github.com/rdehaan/ja-asp) disjunctive ASP based approach using the ASP solver Clingo (v5.6.1).[4] However, JA-ASP supports outcome determination via enumeration of answer sets, which turned out not to be feasible for almost any of the considered benchmarks. Hence for JA-ASP we only consider the first part of outcome determination, namely, computation of a single collective judgment set, and compare its runtimes to the runtimes of our approach on the full outcome determination task. To support Young and Dodgson voting rules, we adapted JA-ASP by separating the input and output constraints in the ASP encoding of JA-ASP (`ja.lp`).

For the experiments, we consider preference aggregation as a widely-studied instantiation of JA [24, 29, 48]. Many of the JA rules considered in this work generalize specific voting rules [30, 35, 43, 45, 54] in the following sense. Let $C = \{1, \ldots, m\}$ be a set of candidates and $V = (>_1, \ldots, >_n)$ be a profile of n voters. Construct the preference agenda with issues $p_{x>y}$ for $x, y \in C$, $x < y$ and let $p_{y>x} = \neg p_{x>y}$ for $x > y$. Consider the profile $P = (J_1, \ldots, J_n)$ via $J_i = \{p_{x>y} \mid x >_i y\}$, and let $\textsc{Transitivity} = \bigwedge_{\substack{x,y,z \in C \\ x \neq y \neq z}} ((p_{x>y} \wedge p_{y>z}) \rightarrow p_{x>z})$, $\textsc{Winner} = \bigvee_{x \in C} \bigwedge_{y \neq x} p_{x>y}$. For $\Gamma_{\text{out}} = \Gamma_{\text{in}} = \textsc{Transitivity}$, the Kemeny and Slater JA rules correspond to their voting counterparts. For $\Gamma_{\text{out}} = \textsc{Winner}$ and $\Gamma_{\text{in}} = \textsc{Transitivity}$, the Young and Dodgson JA rules correspond to their voting counterparts. The winner determination problem for the aforementioned voting rules is also Θ_2^p-complete [38, 39, 42, 64],

[4] While an ASP-based approach specific to voting rules called Democratix has been earlier proposed [12], it is not currently available and we were unable to contact the authors despite our efforts to do so.

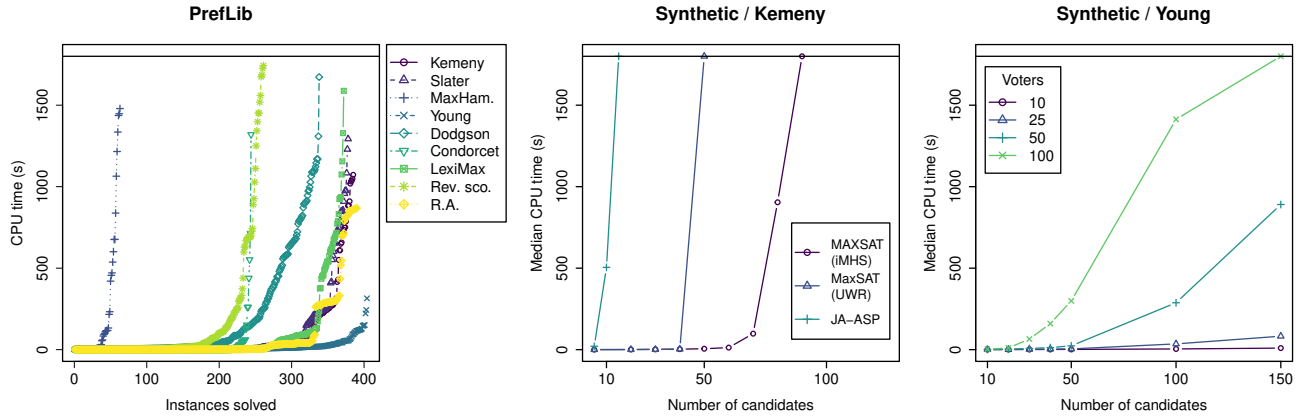

Figure 1: Left: Runtime distribution for SAT-based approaches on PrefLib instances under different rules; Middle: median JA-ASP and MaxSAT runtimes on synthetic instances for Kemeny; Right: scalability of UWrMaxSAT w.r.t. number of voters under Young.

matching the complexity of outcome determination [32], implying that focusing the experiments on the Kemeny, Slater, Young, and Dodgson voting rules gives complexity-wise relevant benchmarks for our procedures. While the Condorcet, ranked agenda, and reversal scoring judgment aggregation rules (which are in general NP-hard) correspond (with $\Gamma_{\text{in}} = \Gamma_{\text{out}} = \textsc{Transitivity}$) to polynomial-time computable voting rules (specifically, top cycle for Condorcet, ranked pairs for ranked agenda, and Borda for reversal scoring), we also consider these in our experiments for a further comparison with the earlier-proposed ASP-based approach.

As benchmarks, we use both synthetically generated and PrefLib [52] data. For PrefLib instances, we selected all strict total orderings available in https://www.preflib.org/static/data/types/soc.zip (as of October 7, 2022) which do not have a (strong) Condorcet winner, resulting in a total of 405 instances. The synthetic data was generated using the *preflibtools* package (https://github.com/PrefLib/preflibtools) according to the impartial culture, i.e., n votes over m candidates are randomly selected from a uniform distribution over all ($m!$) possible ballots. For the Kemeny and Slater rules, we fix $n = 1000$ voters, and used $m = 5, 10, 15, 20, 30, \ldots, 100$ candidates, generating 100 instances for each m. (Note that the size of the MaxSAT encoding depends only on m.) For the Young and Dodgson rules, we used $n = 10, 25, 50, 100$ and $m = 10, 20, 30, 40, 50, 100, 150$, generating 20 instances for each (n, m). We set $L = \{p_{x \succ y} \mid x \neq y\}$ as input to outcome determination, where x is the first candidate in the voting instance. The experiments 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.

Results for our approach on PrefLib instances are summarized in Figure 1 (left) using UWrMaxSat as the MaxSAT solver and MiniPref as the PrefSAT solver. Note that instances which were not solved reached resource limits (time or memory). The Young rule appears the easiest for our approach. More than 93% of instances are solved also under Kemeny and Slater (most within 2 seconds). MaxHamming appears to be the hardest empirically with only 15.5% of instances solved. In terms of encoding size over solved instances, for example Kemeny and Young resulted in up to 62k and 3.4M variables, and 43M and 19M clauses, respectively. Our approach outperforms JA-ASP on each rule, solving 1.5x (Young) to 10x (Kemeny) the instances (see Table 1) despite the fact that JA-ASP is actually only solving a subset of the problem (computing any optimal collective judgment set). For the iterative procedures, the difference to JA-ASP is even more pronounced.

Table 1: SAT vs ASP approach on PrefLib instances: average runtime (avg. t) in seconds (s), number of solved instances (#slv), and percentage of solved instances (%slv).

	SATcha			JA-ASP		
Rule	avg. t (s)	#slv	%slv	avg. t (s)	#slv	%slv
Kemeny	157.11	385	95.1	1641.81	37	9.1
Slater	192.36	378	93.3	1038.83	177	43.7
MaxHam.	1555.13	63	15.5	1657.54	35	8.6
Young	19.59	404	99.8	716.93	269	66.4
Dodgson	451.82	338	83.5	1585.87	64	15.8
Rev. sco.	728.73	261	64.4	1800.00	0	0.0
Condorcet	726.81	244	60.2	1482.19	106	26.2
R.A.	144.31	390	96.3	1800.00	0	0.0
LexiMax	220.44	372	91.9	859.37	221	54.6

Figure 1 (middle) and Figure 1 (right) demonstrate the scalability of our MaxSAT-based approach under Kemeny and Young, respectively, on synthetic instances (see supplement for detailed results under the other rules). Figure 1 (middle) shows that the SAT-based approach (iMaxHS scaling to 80 candidates) scales noticeably further than JA-ASP (scaling up to 15 candidates); similar scalability differences were observed under Slater. Figure 1 (right) shows scalability under Young w.r.t. number of candidates. Most of the instances are solved in under 200 seconds for 25 voters and up to 150 candidates. In contrast, JA-ASP times out on most of the instances already for 10 voters and 100 candidates. For 50 voters, we reach 50 candidates, showing that the empirical hardness of the Young rule depends heavily on the number of voters.

6 MANIPULATION, BRIBERY, AND CONTROL

We briefly outline how the algorithms described so-far can be adapted to cover manipulation [4, 35], bribery [4], and control [5] in judgment aggregation [6]. We focus specifically on the Kemeny rule under which these problems are Σ_2^p-complete [20], and describe MaxSAT-based CEGAR algorithms for them. Note that while in the following definitions we consider subset-based preferences where a better outcome is specified as a subset of the agenda, the algorithms can be adapted to the more general Hamming distance preferences [20]. Furthermore, the main ideas behind the algorithms can also be applied to other judgment aggregation rules which allow for direct MaxSAT encodings, such as those considered in Section 3; for achieving this, the MaxSAT encoding in question should be adapted to accommodate modifications to the profile and the resulting change of the cost function.[5]

Manipulation. In the manipulation problem, we are given an agenda Φ, an integrity constraint Γ, a profile $P = (J_1, \ldots, J_n) \in \mathcal{P}(\Phi, \Gamma)$, and a subset $L \subseteq J_1$. The question is whether there is a judgment set $J' \in \mathcal{J}(\Phi, \Gamma)$ and a corresponding profile $P' = (J', J_2, \ldots, J_n) \in \mathcal{P}(\Phi, \Gamma)$ such that for all $J^*_{\text{new}} \in \textsc{Kemeny}(P')$ it holds that $L \subseteq J^*_{\text{new}}$, i.e., whether a given (w.l.o.g. the first) agent can manipulate the outcome by providing an insincere individual judgment set to guarantee that a particular outcome is achieved.

We shortly outline a MaxSAT-based CEGAR algorithm for manipulation under the Kemeny rule. As the abstraction, use a MaxSAT encoding the solutions of which correspond to an individual judgment set J' and a Kemeny judgment set $J^*_{\text{new}} \in \textsc{Kemeny}(P')$. To construct the abstraction, we first form the MaxSAT encoding of $\textsc{Kemeny}(J_2, \ldots, J_n)$. We represent the judgment set J' as Boolean variables $s_1, \ldots, s_m$. Now, the constraint $\Gamma[x_j \mapsto s_j \mid j = 1, \ldots, m]$ enforces that this judgment set is Γ-consistent. For each $j = 1, \ldots, m$, the soft clauses $(s_j \rightarrow x_j)$ and $(\neg s_j \rightarrow \neg x_j)$ with unit weights are introduced to model the additional Kemeny cost incurred by the judgment set J'.

The CEGAR loop begins by querying a MaxSAT solver for a judgment set J' and a corresponding Kemeny judgment set $J^*_{\text{new}} \supseteq L$ by including $\bigwedge_{l \in L} l$ to the hard clauses. If no such J' and J^*_{new} exist, we return false and terminate. If there is such a J', we check whether there is another judgment set $J^*_{\text{cex}} \in \textsc{Kemeny}(P)$ which does not include L. To this end, we remove $\bigwedge_{l \in L} l$ from the hard clauses, fix the judgment set J' as hard clauses s_j for $x_j \in J'$ and $\neg s_j$ for $\neg x_j \in J'$, and query the MaxSAT solver again. If the optimal cost for this call increases compared to the first one, we return J' as a successful manipulation. Otherwise we obtain the counterexample judgment set J^*_{cex} and refine our abstraction by adding the clause $\bigvee_{l \in J'} \neg l$ which excludes the judgment set J'.

Bribery. In bribery the task is to decide whether there is a way to change up to k judgment sets in P so that a predescribed outcome L is always achieved in the modified profile P'. We represent the selection of up to k judgment sets by Boolean variables $y_1, \ldots, y_n$ (similarly as for Young) and a cardinality constraint $\sum_{i=1}^{n} y_i \leq k$. Further variables x_{ij} for each $i = 1, \ldots, n$ and $j = 1, \ldots, m$ are used to represent how the at most k judgment sets are changed (similarly as for Dodgson). Constraints $\Gamma[x_j \mapsto x_{ij} \mid j = 1, \ldots, m]$ for each $i = 1, \ldots, n$ ensure that the judgment sets remain consistent. We enforce that each judgment set not changed (i.e., for each y_i assigned to 0) is to remain unchanged via $\neg y_i \rightarrow \bigwedge_{x_j \in J_i} x_{ij} \wedge \bigwedge_{\neg x_j \in J_i} \neg x_{ij}$ for each $i = 1, \ldots, n$. To ensure the correct Kemeny cost for solutions, the soft clauses of the MaxSAT encoding for Kemeny are modified to $(x_{ij} \rightarrow x_j)$ and $(\neg x_{ij} \rightarrow \neg x_j)$ for each $i = 1, \ldots, n$ and $j = 1, \ldots, m$, with unit weights.

Using this abstraction, the outline of the CEGAR algorithm for bribery is the same as the one just outlined for manipulation, with the following exceptions. In the counterexample call, we fix the modified profile via unit hard clauses over x_{ij} variables. Furthermore, we refine the abstraction by a clause stating that we either need a different selection of judgment sets or the currently selected judgment sets need to be changed in a different way.

Control. In control the task is to decide whether there is a way to remove issues to arrive at a subset $\Phi' \subseteq \Phi$ so that a predescribed outcome L is always achieved in the modified profile P'. We represent the issues removed by variables z_j for each $x_j \in \Phi \setminus (L \cup \neg L)$ (no issues in L can be removed).[6] The soft clauses (l) for $l \in J_{\text{m}}(P) \cap (\Phi \setminus (L \cup \neg L))$ of the MaxSAT encoding for Kemeny are changed to $\neg z_j \rightarrow l$ where z_j corresponds to the variable $V(l) = x_j$, keeping the original weights. Using this abstraction, the general outline of the CEGAR algorithm for control is the same as for manipulation and bribery, apart for the following differences: when checking for a counterexample we fix via z_j variables which issues to keep in the agenda, and we refine the abstraction via a clause stating that we need to select a different subset of issues.

7 CONCLUSION

We developed novel outcome determination algorithms adhering to the known complexity bounds for the problem under a range of aggregation rules via identifying well-suited SAT-based techniques for each setting. We also showed that an implementation of the approach scales noticeably beyond an earlier-proposed exact declarative approach to judgment aggregation. Overall, the empirical results obtained suggest that the approach is viable and could potentially 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 [14] and abstract argumentation [7, 9]. We further outlined how the approach can be adapted to obtain algorithms for manipulation, bribery, and control. The approach could also be extended to cover further NP-hard judgment aggregation rules, such as the binomial rule [18], the MaxEq rule [10], and rules based on geodesic distance [26, 32].

ACKNOWLEDGMENTS

Work financially supported by Academy of Finland under grants 322869, 328718, and 347588. The authors thank the Finnish Computing Competence Infrastructure (FCCI) for computational and data storage resources.

[5] For example, manipulation under Slater and the cost function on the modified profile are encoded using variables for issues whose reversal changes the majoritarian judgment set. For bribery, the cost function can be similarly encoded using cardinality constraints. The other direct MaxSAT encodings in this work rely on cardinality constraints whose left-hand sides are changed to consider the modified profile.

[6] To also cover the related task of control by *introducing additional issues*, where we are additionally given a subset $\Phi'' \subseteq \Phi$ which cannot be excluded, we would simply not declare the z_j variables for issues in Φ''.

REFERENCES

[1] Josep Argelich, Inês Lynce, and João P. Marques Silva. 2009. On Solving Boolean Multilevel Optimization Problems. In *IJCAI*. ijcai.org, 393–398.

[2] Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. 2021. Maximum Satisfiabiliy. In *Handbook of Satisfiability - Second Edition*. FAIA, Vol. 336. IOS Press, 929–991.

[3] Olivier Bailleux and Yacine Boufkhad. 2003. Efficient CNF Encoding of Boolean Cardinality Constraints. In *CP (LNCS, Vol. 2833)*. Springer, 108–122.

[4] Dorothea Baumeister, Gábor Erdélyi, Olivia Johanna Erdélyi, and Jörg Rothe. 2015. Complexity of manipulation and bribery in judgment aggregation for uniform premise-based quota rules. *Math. Soc. Sci.* 76 (2015), 19–30.

[5] Dorothea Baumeister, Gábor Erdélyi, Olivia Johanna Erdélyi, Jörg Rothe, and Ann-Kathrin Selker. 2020. Complexity of control in judgment aggregation for uniform premise-based quota rules. *J. Comput. Syst. Sci.* 112 (2020), 13–33.

[6] Dorothea Baumeister, Gábor Erdélyi, and Jörg Rothe. 2016. Judgment Aggregation. In *Economics and Computation, An Introduction to Algorithmic Game Theory, Computational Social Choice, and Fair Division*. Springer, 361–391.

[7] Dorothea Baumeister, Daniel Neugebauer, and Jörg Rothe. 2021. Collective Acceptability in Abstract Argumentation. *FLAP* 8, 6 (2021), 1503–1542.

[8] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh (Eds.). 2021. *Handbook of Satisfiability - Second Edition*. FAIA, Vol. 336. IOS Press.

[9] Gustavo Adrian Bodanza, Fernando Tohmé, and Marcelo Auday. 2017. Collective argumentation: A survey of aggregation issues around argumentation frameworks. *Argument Comput.* 8, 1 (2017), 1–34.

[10] Sirin Botan, Ronald de Haan, Marija Slavkovik, and Zoi Terzopoulou. 2021. Egalitarian Judgment Aggregation. In *AAMAS*. ACM, 214–222.

[11] Felix Brandt, Guillaume Chabin, and Christian Geist. 2015. Pnyx: A Powerful and User-friendly Tool for Preference Aggregation. In *AAMAS*. ACM, 1915–1916.

[12] Günther Charwat and Andreas Pfandler. 2015. Democratix: A Declarative Approach to Winner Determination. In *ADT (LNCS, Vol. 9346)*. Springer, 253–269.

[13] Weiwei Chen and Ulle Endriss. 2019. Preservation of semantic properties in collective argumentation: The case of aggregating abstract argumentation frameworks. *Artif. Intell.* 269 (2019), 27–48.

[14] Julian Chingoma, Ulle Endriss, and Ronald de Haan. 2022. Simulating Multiwinner Voting Rules in Judgment Aggregation. In *AAMAS*. IFAAMAS, 263–271.

[15] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. 2003. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM* 50, 5 (2003), 752–794.

[16] Edmund M. Clarke, Anubhav Gupta, and Ofer Strichman. 2004. SAT-based counterexample-guided abstraction refinement. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 23, 7 (2004), 1113–1123.

[17] Vincent Conitzer, Andrew J. Davenport, and Jayant Kalagnanam. 2006. Improved Bounds for Computing Kemeny Rankings. In *AAAI*. AAAI Press, 620–626.

[18] Marco Costantini, Carla Groenland, and Ulle Endriss. 2016. Judgment Aggregation under Issue Dependencies. In *AAAI*. AAAI Press, 468–474.

[19] Andrew J. Davenport and Jayant Kalagnanam. 2004. A Computational Study of the Kemeny Rule for Preference Aggregation. In *AAAI*. AAAI Press / The MIT Press, 697–702.

[20] Ronald de Haan. 2017. Complexity Results for Manipulation, Bribery and Control of the Kemeny Judgment Aggregation Procedure. In *AAMAS*. ACM, 1151–1159.

[21] Ronald de Haan and Marija Slavkovik. 2017. Complexity Results for Aggregating Judgments using Scoring or Distance-Based Procedures. In *AAMAS*. ACM, 952–961.

[22] Ronald de Haan and Marija Slavkovik. 2019. Answer Set Programming for Judgment Aggregation. In *IJCAI*. ijcai.org, 1668–1674.

[23] Franz Dietrich. 2014. Scoring rules for judgment aggregation. *Soc. Choice Welf.* 42, 4 (2014), 873–911.

[24] Franz Dietrich and Christian List. 2007. Arrow's theorem in judgment aggregation. *Soc. Choice Welf.* 29, 1 (2007), 19–33.

[25] Carmine Dodaro and Alessandro Previti. 2019. Minipref: A Tool for Preferences in SAT. In *RCRA (CEUR Workshop Proc., Vol. 2538)*. CEUR-WS.org.

[26] Conal Duddy and Ashley Piggins. 2012. A measure of distance between judgment sets. *Soc. Choice Welf.* 39, 4 (2012), 855–867.

[27] Wolfgang Dvorák, Matti Järvisalo, Johannes Peter Wallner, and Stefan Woltran. 2014. Complexity-sensitive decision procedures for abstract argumentation. *Artif. Intell.* 206 (2014), 53–78.

[28] Niklas Eén and Niklas Sörensson. 2003. Temporal induction by incremental SAT solving. *Electron. Notes Theor. Comput. Sci* 89, 4 (2003), 543–560.

[29] Ulle Endriss. 2016. Judgment Aggregation. In *Handbook of Computational Social Choice*. Cambridge University Press, 399–426.

[30] Ulle Endriss. 2018. Judgment Aggregation with Rationality and Feasibility Constraints. In *AAMAS*. IFAAMAS / ACM, 946–954.

[31] Ulle Endriss and Ronald de Haan. 2015. Complexity of the Winner Determination Problem in Judgment Aggregation: Kemeny, Slater, Tideman, Young. In *AAMAS*. ACM, 117–125.

[32] Ulle Endriss, Ronald de Haan, Jérôme Lang, and Marija Slavkovik. 2020. The Complexity Landscape of Outcome Determination in Judgment Aggregation. *J. Artif. Intell. Res.* 69 (2020), 687–731.

[33] Ulle Endriss and Umberto Grandi. 2017. Graph aggregation. *Artif. Intell.* 245 (2017), 86–114.

[34] Ulle Endriss, Umberto Grandi, Ronald de Haan, and Jérôme Lang. 2016. Succinctness of Languages for Judgment Aggregation. In *KR*. AAAI Press, 176–186.

[35] Ulle Endriss, Umberto Grandi, and Daniele Porello. 2012. Complexity of Judgment Aggregation. *J. Artif. Intell. Res.* 45 (2012), 481–514.

[36] Patricia Everaere, Sébastien Konieczny, and Pierre Marquis. 2014. Counting votes for aggregating judgments. In *AAMAS*. IFAAMAS/ACM, 1177–1184.

[37] Davide Grossi and Gabriella Pigozzi. 2014. *Judgment Aggregation: A Primer*. Morgan & Claypool Publishers.

[38] Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. 1997. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *J. ACM* 44, 6 (1997), 806–825.

[39] Edith Hemaspaandra, Holger Spakowski, and Jörg Vogel. 2005. The complexity of Kemeny elections. *Theor. Comput. Sci.* 349, 3 (2005), 382–391.

[40] Alexey Ignatiev, António Morgado, and João Marques-Silva. 2018. PySAT: A Python Toolkit for Prototyping with SAT Oracles. In *SAT (LNCS, Vol. 10929)*. Springer, 428–437.

[41] Mikolás Janota, Radu Grigore, and João Marques-Silva. 2010. Counterexample Guided Abstraction Refinement Algorithm for Propositional Circumscription. In *JELIA (LNCS, Vol. 6341)*. Springer, 195–207.

[42] Michael Lampis. 2022. Determining a Slater Winner Is Complete for Parallel Access to NP. In *STACS (LIPIcs, Vol. 219)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 45:1–45:14.

[43] Jérôme Lang, Gabriella Pigozzi, Marija Slavkovik, Leon van der Torre, and Srdjan Vesic. 2017. A partial taxonomy of judgment aggregation rules and their properties. *Soc. Choice Welf.* 48, 2 (2017), 327–356.

[44] Jérôme Lang, Gabriella Pigozzi, Marija Slavkovik, and Leendert W. N. van der Torre. 2011. Judgment aggregation rules based on minimization. In *TARK*. ACM, 238–246.

[45] Jérôme Lang and Marija Slavkovik. 2013. Judgment Aggregation Rules and Voting Rules. In *ADT (LNCS, Vol. 8176)*. Springer, 230–243.

[46] Jérôme Lang and Marija Slavkovik. 2014. How Hard is it to Compute Majority-Preserving Judgment Aggregation Rules?. In *ECAI (FAIA, Vol. 263)*. IOS Press, 501–506.

[47] Vladimir Lifschitz. 2019. *Answer Set Programming*. Springer.

[48] Christian List. 2012. The theory of judgment aggregation: an introductory review. *Synth.* 187, 1 (2012), 179–207.

[49] Christian List and Philip Pettit. 2002. Aggregating Sets of Judgments: An Impossibility Result. *Economics and Philosophy* 18, 1 (2002), 89–110.

[50] João Marques-Silva, Josep Argelich, Ana Graça, and Inês Lynce. 2011. Boolean lexicographic optimization: algorithms & applications. *Ann. Math. Artif. Intell.* 62, 3-4 (2011), 317–343.

[51] Ruben Martins, Saurabh Joshi, Vasco M. Manquinho, and Inês Lynce. 2014. Incremental Cardinality Constraints for MaxSAT. In *CP (LNCS, Vol. 8656)*. Springer, 531–548.

[52] Nicholas Mattei and Toby Walsh. 2013. PrefLib: A Library for Preferences http://www.preflib.org. In *ADT (LNCS, Vol. 8176)*. Springer, 259–270.

[53] Marina Meila, Kapil Phadnis, Arthur Patterson, and Jeff A. Bilmes. 2007. Consensus ranking under the exponential model. In *UAI*. AUAI Press, 285–294.

[54] Michael K. Miller and Daniel N. Osherson. 2009. Methods for distance-based judgment aggregation. *Soc. Choice Welf.* 32, 4 (2009), 575–601.

[55] Klaus Nehring and Marcus Pivato. 2019. Majority rule in the absence of a majority. *J. Econ. Theory* 183 (2019), 213–257.

[56] Klaus Nehring, Marcus Pivato, and Clemens Puppe. 2014. The Condorcet set: Majority voting over interconnected propositions. *J. Econ. Theory* 151 (2014), 268–303.

[57] Andreas Niskanen, Jeremias Berg, and Matti Järvisalo. 2022. Incremental Maximum Satisfiability. In *SAT (LIPIcs, Vol. 236)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 14:1–14:19.

[58] Philip Pettit. 2001. Deliberative Democracy and the Discursive Dilemma. *Philosophical Issues* 11 (2001), 268–299.

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

[60] Marek Piotrów. 2020. UWrMaxSat: Efficient Solver for MaxSAT and Pseudo-Boolean Problems. In *ICTAI*. IEEE, 132–136.

[61] Daniele Porello and Ulle Endriss. 2014. Ontology merging as social choice: judgment aggregation under the open world assumption. *J. Log. Comput.* 24, 6 (2014), 1229–1249.

[62] Simon Rey, Ulle Endriss, and Ronald de Haan. 2020. Designing Participatory Budgeting Mechanisms Grounded in Judgment Aggregation. In *KR*. ijcai.org, 692–702.

[63] Emanuele Di Rosa, Enrico Giunchiglia, and Marco Maratea. 2010. Solving satisfiability problems with preferences. *Constraints An Int. J.* 15, 4 (2010), 485–515.

[64] Jörg Rothe, Holger Spakowski, and Jörg Vogel. 2003. Exact Complexity of the Winner Problem for Young Elections. *Theory Comput. Syst.* 36, 4 (2003), 375–386.