A Graph-Based Algorithm for the Automated Justification of Collective Decisions

Oliviero Nardi
ILLC, University of Amsterdam
The Netherlands

Arthur Boixel
ILLC, University of Amsterdam
The Netherlands

Ulle Endriss
ILLC, University of Amsterdam
The Netherlands

ABSTRACT

We develop an algorithm for the axiomatic justification problem in social choice that is sufficiently efficient to be applicable in decision making scenarios of real practical interest. Given a profile of individual preferences, a suggested election outcome, and a corpus of axioms encoding fundamental normative principles of electoral fairness, solving this justification problem involves computing a minimal set of instances of some of the axioms in the corpus that together rule out any outcome that is different from the one we want to justify. Our approach combines the use of state-of-the-art tools for computing minimally unsatisfiable sets of constraints with a graph-search algorithm. The latter searches the graph induced by the set of all axiom instances in an incremental manner and relies on a number of heuristics to further improve performance.

KEYWORDS

Computational Social Choice; Explainable AI; Algorithms

ACM Reference Format:
Oliviero Nardi, Arthur Boixel, and Ulle Endriss. 2022. A Graph-Based Algorithm for the Automated Justification of Collective Decisions. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

Imagine a scenario in which a small committee has to take a decision that involves choosing from a handful of alternatives. The committee members have different preferences and there is no fixed voting rule (prescribed by a constitution or similar) they can rely on to decide. Instead they want to justify the choice made directly from fundamental normative principles of social choice (so-called *axioms*) and they want to explain to outsiders why accepting those principles entails the choice made. Such notions of *justification* and *explanation* have recently been advocated by several authors working in the field of computational social choice [9, 11, 22, 24]. Their interest in these concepts aligns with the surge of attention that explainability is receiving from researchers in AI [2].

Boixel and Endriss [9] put forward a general definition of what it means to *justify* a target outcome $X^\star$ for a given preference profile $R^\star$ in axiomatic terms. This definition can be used together with any corpus of axioms one might wish to refer to in a justification—both classical axioms from the literature on social choice theory [32] and new user-defined axioms for specific application scenarios. The downside of this level of generality is that it makes generating such justifications a computationally very demanding task. Indeed, the baseline algorithm of Boixel and Endriss hardly scales beyond scenarios with three voters and three alternatives. In this paper, we set out to develop a new algorithm to solve the justification problem that improves significantly over the performance of that baseline, with the ambition of making justification generation available for use in practice for scenarios such as the one invoked earlier.

Let us briefly recall the definition of Boixel and Endriss in informal terms (see Section 2.2 for a formal definition). Given a profile $R^\star$, a target outcome $X^\star$, and a (possibly large) corpus $\mathbb{A}$ of axioms, we first look for a *normative basis*, a subset of $\mathbb{A}$, such that *every* voting rule that satisfies the axioms in that normative basis would elect $X^\star$ in $R^\star$. This provides a normative argument for choosing $X^\star$. But our audience might not yet understand that argument. So, next, we look for an explanation. Observe that we can think of an axiom, constraining the behaviour of acceptable voting rules in a large number of situations, as consisting of many different *axiom instances*, each talking about one specific situation only (say, one specific pair of profiles and one specific alternative).[1] An *explanation* is a set of instances of the axioms in the normative basis that is small (so we can present it to an audience) yet logically strong enough to still force $X^\star$ as the only possible outcome for $R^\star$.

In any concrete implementation, axioms and their instances need to be encoded in a suitable formal language, such as propositional logic or a constraint modelling language [9].[2] Finding the instances making up the explanation from the set of all instances of the axioms in the corpus $\mathbb{A}$ is the main algorithmic problem we face. The baseline algorithm of Boixel and Endriss consists of a *generation phase* (of the instances of the axioms in $\mathbb{A}$) and a subsequent *solving phase* (where we try to identify instances making up an explanation). The latter, while computationally demanding in its own right [10], naturally maps into the well-studied problem of computing a *minimally unsatisfiable subset* (MUS) of a set of formulas (or constraints) and thus can be tackled using state-of-the-art tools [7, 17, 18, 26, 31]. The real bottleneck is the generation phase.

At the core of our approach is the idea of generating the set of relevant axiom instances in an incremental fashion, and to interleave generation and solving rather than only starting solving once generation is complete. We start by considering only axiom instances that exclusively refer to the given profile $R^\star$. We then proceed to add instances mentioning other profiles as well. We keep on iteratively enlarging the search space by adding—in each round—all axiom instances that mention at least one profile discovered in the previous round. This algorithm is then refined using

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

[1] For example, the very simple FAITHFULNESS axiom [30] says that whenever there is just a single voter, her top alternative should win; one of its many instances says that a should win when there is only one voter and she expresses the preference $a \succ b \succ c$.

[2] Encoding axioms into propositional logic to make it possible to reason about them, e.g., to prove impossibility theorems, with the help of SAT solvers is by now a standard technique in computational social choice [14, 28], which we can use also here.

two types of heuristics. First, using *implied-instance heuristics*, we avoid generating axiom instances that are logically entailed by two or more of the instances generated already. Second, using *derived-axiom heuristics*, we define new "meta-axioms" for combinations of axioms that frequently feature together in justifications observed in experiments, so as to directly stir the algorithm towards common patterns found in solutions to real-world justification problems.

Related work. The term "justification", in the sense in which we use it here, has first been invoked in the literature on computational social choice in the work of Cailloux and Endriss [11], who developed an algorithm for the justification problem for a specific set of axioms, namely those featuring in the axiomatisation of the Borda rule due to Young [30]. This algorithm thus can be used to justify the election of Borda winners (and only those). Peters et al. [22] further refined and generalised this approach and proved it to be optimal in view of the lengths of the arguments it produces. Boixel and Endriss [9] instead provided a general definition of the notion of justification, one that can be applied to any corpus of axioms and one that relies on the use of general-purpose algorithms developed in AI and Operations Research (such as algorithms for constraint programming and SAT solving) rather than on tailor-made algorithms for the problem at hand. Our work directly builds on those ideas. Finally, Boixel and de Haan [10] analysed the computational complexity of (the solving phase of) the justification problem.

Contribution. We develop a heuristic graph-search algorithm for solving the axiomatic justification problem in social choice; we prove that algorithm to be correct; and we demonstrate that it performs well in scenarios big enough to be of practical interest.

Paper outline. The remainder of this paper is organised as follows. We review relevant prior work in Section 2, covering both basic notions from voting theory and the definition of the axiomatic justification problem. We then present our graph-based algorithm and prove it to be correct in Section 3, before analysing its performance on both synthetic and real-world preference data in Section 4. For more details, refer to the MSc thesis of the first author [20]. Our code (including a full record of our experiments) is available online [21].

2 JUSTIFICATION OF ELECTION OUTCOMES

In this section, we first present relevant notions from voting theory. We then recall the definition of a justification for an election outcome as proposed by Boixel and Endriss [9] and briefly discuss the challenging task of retrieving such justifications in practice.

2.1 Voting Theory

Before we begin, let us note that we will model *anonymous* elections, i.e., elections where all voters have the same power and thus the identities of individual voters do not matter. This not only is a fairness requirement imposed on many real-world elections, but allows for a compact representation of voter preferences.

Let $n^\star \in \mathbb{N}$ be the number of *voters* and let X be the finite and nonempty set of *alternatives*. Each voter's preference is modelled as a (strict) linear order over X; the set of all such orders is denoted by $\mathcal{L}(X)$. A *profile* is a function $R : \mathcal{L}(X) \rightarrow \mathbb{N}_0$ (where $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$), mapping any given preference to the number of voters expressing it. We denote by $|R|$ the total number of voters expressing some preference in profile R. We write $\mathcal{R}^{[n]}$ for the set of all profiles with exactly n voters, and $\mathcal{R}^+ = \bigcup_{n=1}^{n^\star} \mathcal{R}^{[n]}$ for the set of all profiles with up to $n^\star$ voters. Furthermore, we define $R + R'$ as the sum of the two profiles (that is, $(R + R')(\succ) = R(\succ) + R'(\succ)$).

A *voting rule* F for $n^\star$ voters and alternatives X is a function $F : \mathcal{R}^+ \rightarrow 2^X \setminus \{\varnothing\}$, mapping any given profile to a (nonempty) set of winning alternatives. This definition reflects the fact that a voting rule might declare a tie between two or more alternatives. Examples of well-known voting rules include the plurality rule, the Borda rule, and the Copeland rule [32].

Different voting rules satisfy different normative principles, known as *axioms* in social choice theory. A large number of such axioms can be found in the literature [32]. We recall here a small selection of such axioms (the ones we will use in our experiments). They apply to all profiles in $R, R' \in \mathcal{R}^+$ and alternatives $x, y \in X$.

- Faithfulness. If there is only a single voter in profile R, then her top-ranked choice should be the unique winner.
- Pareto Principle. If all voters prefer alternative x to alternative y in profile R, then y should not win.
- Cancellation. If all alternatives tie in pairwise majority contests in profile R, then all alternatives should win.
- Neutrality. If profile R' can be obtained from profile R by permuting the occurrences of the alternatives within R, then the winners under R' should be obtained by applying the same kind of permutation to the winners under R.
- Positive Responsiveness. If profile R' can be obtained from profile R by raising the support for alternative x (one of the winners under R), then x must be the sole winner under R'.
- Reinforcement. If the intersection of the winning sets under profiles R and R' is nonempty, then that intersection should win under $R + R'$ (as long as $|R + R'| \leq n^\star$).[3]

Formally, the *interpretation* of an axiom A, which we denote by $\mathbb{I}(A)$, is simply the set containing all the voting rules that satisfy A. The interpretation of a set of axioms is defined analogously.

An axiom is (typically) a complex requirement, constraining the behaviour of rules w.r.t. many profiles and alternatives. So we can think of an axiom as being made up of multiple *axiom instances*. Intuitively, an instance encodes the general restriction of an axiom for a concrete situation (e.g., a specific profile). For example, the Pareto Principle talks about *all* profiles in which *some* alternative dominates *some* other alternative. One of its many instances stipulates that for the specific profile in which two voters report $a \succ b \succ c$ and three voters report $b \succ c \succ a$, alternative c should not win. What exactly constitutes an instance depends on the language in which axioms are expressed, and several definitions of instances are possible w.r.t. the same language. Still, Boixel and Endriss [9] stipulated a list of requirements that such a definition must satisfy (for example, any instance should be an axiom in its own right). We write $A' \triangleleft A$ if A' is an instance of A. Similarly, given two sets of axioms $\mathcal{A}$ and $\mathcal{A}'$, we write $\mathcal{A}' \triangleleft \mathcal{A}$ if every $A' \in \mathcal{A}'$ is an instance of some $A \in \mathcal{A}$. Moreover, given an instance A', let $\mathbb{P}(A')$ denote the set of profiles *mentioned* in (or *constrained*) by A'. For a set of instances $\mathcal{A}'$, the set $\mathbb{P}(\mathcal{A}')$ is defined as $\bigcup_{A' \in \mathcal{A}'} \mathbb{P}(A')$.

[3] Note that this variant of Reinforcement is weaker than the original formulation proposed by Young [30], because it only applies to profiles containing the preferences of at most $n^\star$ voters overall. We refer to Boixel and Endriss [9] for a discussion.

2.2 Axiomatic Justifications

Given a profile $R^\star$, what would be a good argument for why a specific outcome $X^\star$ should be selected? If $X^\star$ happens to be the outcome selected by some voting rule F, then one could try to argue in favour of F, possibly by reference to the axioms characterising F, and thereby—*indirectly*—justify the selection of $X^\star$. An alternative approach would be to argue for the selection of $X^\star$ *directly* from some set of attractive axioms, by demonstrating that accepting those axioms forces us to accept that outcome [9].

Definition 1 (Boixel and Endriss, 2020). *Let $\mathbb{A}$ be a corpus of axioms for voting rules for up to $n^\star$ voters and alternatives X, let $R^\star$ be a profile for $n^\star$ voters, and let $X^\star \subseteq X$ a nonempty set of alternatives. Then a **justification** for $X^\star$ in $R^\star$ given $\mathbb{A}$ is a pair $\langle \mathcal{A}^N, \mathcal{A}^E \rangle$ of sets of axioms, with **normative basis** $\mathcal{A}^N$ and **explanation** $\mathcal{A}^E$, in case the following conditions are satisfied:*

- ***Explanatoriness**. $\mathcal{A}^E$ minimally explains the desired outcome: for every $F \in \mathbb{I}(\mathcal{A}^E)$ it holds that $F(R^\star) = X^\star$ and for every set $\mathcal{A} \subsetneq \mathcal{A}^E$ there is an $F \in \mathbb{I}(\mathcal{A})$ such that $F(R^\star) \neq X^\star$.*
- ***Relevance**. The explanation is composed of instances of the axioms in the normative basis: $\mathcal{A}^E \triangleleft \mathcal{A}^N$.*
- ***Adequacy**. The axioms in the normative basis belong to the provided corpus of axioms: $\mathcal{A}^N \subseteq \mathbb{A}$.*
- ***Nontriviality**. There exists at least one voting rule that satisfies all axioms in the normative basis: $\mathbb{I}(\mathcal{A}^N) \neq \emptyset$.*

This is a complex definition, so let us briefly go through the underlying intuitions. Suppose you want to justify why the outcome should be $X^\star$ when the profile is $R^\star$, and suppose you believe that your audience will find axioms belonging to $\mathbb{A}$ compelling. You will succeed if you can find a set $\mathcal{A}^N$ of axioms such that accepting $\mathcal{A}^N$ logically entails selecting outcome $X^\star$. But not just any $\mathcal{A}^N$ will do! First, the axioms in $\mathcal{A}^N$ need to be acceptable to your audience (*adequacy*). Second, they should not be too strong and lead to an impossibility result, thereby technically implying every conceivable outcome (*nontriviality*). Then, you do not just want to tell your audience *that* accepting $\mathcal{A}^N$ implies choosing $X^\star$, but you also want to provide an explanation for *why* that is the case. But understanding the logical consequences of several axioms, each talking about many different profiles, is difficult. Instead, you want to pinpoint for your audience which instances of the axioms actually play a role for the specific problem at hand. This is where the explanation $\mathcal{A}^E$ comes in. It should be a set of instances of the axioms in $\mathcal{A}^N$ (*relevance*) that retains just enough of the logical strength of $\mathcal{A}^N$ to still imply the selection of $X^\star$ (*explanatoriness*).

Example 1. Consider a two-voter profile $R^\star$ where one voter reports $a \succ b \succ c$ and the other $b \succ a \succ c$. Suppose we want to justify the selection of outcome $X^\star = \{a, b\}$ using the corpus of axioms of Section 2.1. A possible justification would be $\langle \mathcal{A}^N, \mathcal{A}^E \rangle$, where $\mathcal{A}^N = \{$Neutrality, Pareto Principle$\}$ and $\mathcal{A}^E$ is composed of the following two axiom instances:

- (Pareto Principle) Because every voter prefers a over c, alternative c cannot be amongst the winners.
- (Neutrality) Because we cannot distinguish between a and b by looking only at the two ballots (i.e., they are used symmetrically), neither should be treated more favourably than the other: either both a and b are in the outcome, or neither is.

The only outcomes satisfying both instances are those that exclude c and that include a if and only if they include b as well. So the only nonempty set that satisfies these conditions is $\{a, b\}$. △

To encode such an example in propositional logic, we could use one variable $p_{R,x}$ for each $R \in \mathcal{R}^+$ and $x \in X$, with the intended meaning that $p_{R,x}$ is true whenever x should be part of the outcome under profile R. Then, the instance of the Pareto Principle featuring in Example 1 could simply be encoded as $\neg p_{R^\star,c}$.

2.3 Algorithmic Challenges

Given a profile $R^\star$, a target outcome $X^\star$, and a corpus $\mathbb{A}$, we refer to the triple $\langle R^\star, X^\star, \mathbb{A} \rangle$ as a *justification problem*. Solving such a justification problem means computing a justification $\langle \mathcal{A}^N, \mathcal{A}^E \rangle$ that meets the requirements of Definition 1.

Computing justifications is a computationally demanding task. Boixel and de Haan [10] showed that it is at least Σ_2^p-hard [1]; the exact complexity depends on the details of the encoding of the problem. The good news is that the problem of computing $\mathcal{A}^E$ from a given set of instances of $\mathbb{A}$ (together with one additional constraint expressing that $X^\star$ is *not* to be selected) reduces to a well-studied problem, namely computing a *minimally unsatisfiable subset* (MUS). Boixel and Endriss [9] used this insight to design an algorithm for computing justifications. It can be divided into two phases: the *generation phase* and the *solving phase*. In the generation phase, all instances of every axiom in $\mathbb{A}$ are encoded in a suitable language (for example, constraint programming or SAT). Then, in the solving phase, an external reasoning tool (specifically, an MUS extractor) is used to search through the encoded instances and retrieve the justifications. Using this approach, Boixel and Endriss were able to compute justifications for profiles with three voters and three alternatives in 5–30 minutes per profile, but this approach does hardly scale to even moderately larger scenarios.

Since the solving phase can be delegated to a high-performance state-of-the-art tool [17, 18, 31], the bottleneck is the generation phase. So in the sequel we shall focus on this generation phase.

3 ALGORITHM DESIGN AND CORRECTNESS

In this section, we present our main contribution: a graph-based algorithm for solving the justification problem. We also introduce two families of heuristics designed to further improve its performance.

3.1 Core Algorithm: Instance Graph Generation

The core idea underlying our approach is to perform the generation of axiom instances incrementally, in order to interleave generation and solving. To do so, we introduce a notion of *distance* between profiles: this enables us to generate the instances mentioning profiles "close" to $R^\star$ first. If no justification is found, we can then gradually increase the distance and explore the search space further.

To define this distance, we introduce the notion of an *instance graph*. Such a graph can be constructed from any given set of axiom instances (such as those of the axioms in the corpus $\mathbb{A}$). Technically, an instance graph is an *undirected multi-hypergraph*, i.e., a graph where two distinct edges can connect the same profiles, and where an edge may connect any number of profiles. Intuitively, the nodes correspond to profiles and the (hyper-)edges to instances.

Definition 2. *Given a set of instances $\mathcal{A}'$, the* ***instance graph*** *induced by $\mathcal{A}'$ is a pair $\mathcal{G}_{\mathcal{A}'} = \langle P, E \rangle$, where:*

- *P is a set of profiles (also called* ***nodes****) such that $P = \mathbb{P}(\mathcal{A}')$;*
- *E is a multiset of* ***edges****, containing for every instance $A' \in \mathcal{A}'$ an edge $e = \{R \in P \mid R \in \mathbb{P}(A')\}$ that connects all the nodes representing profiles mentioned in A' and that is labelled by A'.*

With a slight abuse of notation, for any given corpus $\mathbb{A}$ of axioms, let $\mathcal{G}_{\mathbb{A}}$ denote the instance graph induced by $\{A' \triangleleft A \mid A \in \mathbb{A}\}$.

A *path* (of length k) is a sequence of edges $e_1 \cdots e_k$ such that two consecutive edges always have at least one profile in common. Two profiles are connected if there is a path between them, and the distance between two connected profiles is the length of the shortest path between them.

The core task we need to accomplish when solving a justification problem $\langle R^\star, X^\star, \mathbb{A} \rangle$ is to find a subset $\mathcal{A}^E$ of the set of instances of $\mathbb{A}$ that meets all our requirements—chief amongst which is the requirement that the set $\mathcal{A}^E \cup \{$"$X^\star$ does *not* win in $R^\star$"$\}$, when expressed in a suitable formal language, is an MUS of the set of instances of $\mathbb{A}$.[4] The problem is that explicitly generating *all* instances of the axioms in $\mathbb{A}$ is not feasible. As outlined earlier, having a notion of distance in place allows us to instead generate instances in an incremental fashion. To do so, we can use familiar graph-search algorithms. In particular, we will employ breadth-first search (BFS) to explore the graph $\mathcal{G}_{\mathbb{A}}$ (starting from $R^\star$) in a systematic way. Our search will be parameterised by a maximum depth $d \in \mathbb{N}_0$ (measured from $R^\star$). We call this algorithm Gen.

Briefly, Gen($\mathcal{G}_{\mathbb{A}}, R^\star, d$) starts by *generating* every instance exclusively mentioning the given profile $R^\star$.[5] Then, if $d > 0$, we *expand* $R^\star$ by generating all instances that mention $R^\star$ together with some other profile(s). For every instance A' generated during this expansion, we add the (newly discovered) profiles in $\mathbb{P}(A')$ to a search (FIFO) queue and generate the instances which exclusively mention them. Then, again, if $d > 1$, we expand the next profile in the queue, and repeat the process. The search is interrupted as soon as a profile at depth d is selected for expansion (as that would mean reaching the profiles at distance $d + 1$ from $R^\star$), and the algorithm returns the sets of instances generated up to that point. Note that every time a profile is *reached* (that is, added to the FIFO queue), the instances that exclusively mention it are generated. This is because, if we explore the graph up to depth d, we want to generate all instances that only mention profiles within distance d from $R^\star$. Throughout the search, we keep track of which axioms in $\mathbb{A}$ have given rise to each of the axiom instances we generate.

So Gen($\mathcal{G}_{\mathbb{A}}, R^\star, d$) returns a set of axiom instances $\mathcal{A}'$ (each of them is linked to an axiom in $\mathbb{A}$). We now need to search through $\mathcal{A}'$ to find a justification—and an explanation $\mathcal{A}^E$ in particular. This is what we call the solving phase. Suppose we have an algorithm available for this task: given a justification problem $\mathcal{J} = \langle R^\star, X^\star, \mathbb{A} \rangle$, the Solve algorithm accepts $\mathcal{J}$ and a set of instances $\mathcal{A}'$ of $\mathbb{A}$ as input and returns any justification for $\mathcal{J}$ for which the explanation is a subset of $\mathcal{A}'$ (if such a justification exists; if not, nothing is returned). As sketched earlier, such an algorithm can be realised with the help of an MUS extractor. Boixel and Endriss [9] designed such an algorithm and we refer to their work for full details.[6]

Algorithm 1: Justify

```
Data: Problem $\mathcal{J} = \langle R^\star, X^\star, \mathbb{A} \rangle$ and depth $d \in \mathbb{N}_0$.
Set $d' \leftarrow 0$ and $\mathcal{A}^{-1} \leftarrow \varnothing$;
while $d' \leq d$ do
    Set $\mathcal{A}^{d'} \leftarrow$ Gen($\mathcal{G}_{\mathbb{A}}, R^\star, d'$);
    If $\mathcal{A}^{d'} = \mathcal{A}^{d'-1}$, then stop;
    If Solve($\mathcal{J}, \mathcal{A}^{d'}$) finds a justification, return it and stop;
    Otherwise, set $d' \leftarrow d' + 1$;
```

We are now ready to define our core algorithm, called Justify (see Algorithm 1). It starts by generating only the instances mentioning $R^\star$, and solving only w.r.t. these instances. If no justification is found, we generate all instances within a distance of 1 from $R^\star$, and solve again; this is repeated until either a justification is found, the maximum depth permitted is reached, or the search reaches a fixed point (meaning that the portion of the graph connected to $R^\star$ has been fully explored).

We are now going to prove Justify to be correct, under the assumption that Solve is. For either one of these algorithms, we say that it is *correct* if it is both *sound* and *complete.* In the case of Justify, soundness means that Justify($\mathcal{J}, d$) only returns actual justifications for $\mathcal{J}$, while completeness means that, for every justification problem $\mathcal{J}$ that has at least one solution there exists a $d \in \mathbb{N}_0$ such that Justify($\mathcal{J}, d'$) returns *something* for every $d' \geq d$.[7] In the case of Solve, soundness means that Solve($\mathcal{J}, \mathcal{A}'$) only returns actual justifications for $\mathcal{J}$, while completeness means that Solve($\mathcal{J}, \mathcal{A}'$) returns *something* whenever there exists at least one justification $\langle \mathcal{A}^N, \mathcal{A}^E \rangle$ with $\mathcal{A}^E \subseteq \mathcal{A}'$. We note that it is possible to devise an algorithm Solve that has both these properties; the algorithm of Boixel and Endriss [9] is an example (the brute-force algorithm simply checking all subsets of $\mathcal{A}'$ is another one). Before we prove correctness, observe that, if we were to replace Gen by an alternative generation routine that simply returns the full set of instances of the axioms in $\mathbb{A}$, then the resulting variant of Justify would be trivially correct (assuming Solve is)—but of course this alternative algorithm would not be efficient.

Theorem 1 (Correctness). *Algorithm* Justify *is correct whenever its subroutine* Solve *is correct.*

Proof. First, observe that soundness of Justify follows immediately from soundness of Solve. So we focus on completeness.

To prove completeness, it is sufficient to show that, if a justification exists, then some set of instances from which an explanation can be extracted will eventually be given as input to Solve. Now, given a sufficiently large depth, all instances mentioning profiles connected to our given profile $R^\star$ will be generated by Gen. It thus remains to be shown that any possible explanation $\mathcal{A}^E$ can only

[4] In addition, we also need to check that $\mathcal{A}^N$, defined as a set the axioms in $\mathbb{A}$ that has the property that its instances fully cover $\mathcal{A}^E$, is satisfiable.

[5] Here, "generating an instance" means generating a suitable expression in our formal language of choice. One option is to use a constraint modelling language [9]. In our experiments we used propositional logic, to be able to work with SAT solvers [14].

[6] While the solving algorithm defined by Boixel and Endriss [9] takes the set of *all* instances of the axioms in $\mathbb{A}$ as input, it can also be applied to any subset of that set.

[7] An alternative, more demanding, definition of completeness would require that *every* justification is retrieved eventually. We shall not explore this idea further in this paper.

ever contain such instances. So, for the sake of contradiction, suppose $\mathcal{A}^E$ includes an instance A' that only mentions profiles that are not connected to $R^\star$. Observe that establishing explanatoriness amounts to showing that $\mathcal{A}^E$ together with the constraint saying that the target outcome $X^\star$ does *not* win in $R^\star$ is inconsistent [9]. But if a given constraint network is inconsistent, then so is at least one of its connected sub-networks [5]. We distinguish two possibilities. First, if the connected component in $\mathcal{A}^E$ that A' belongs to is inconsistent in its own right, then that would violate the requirement of nontriviality of the normative basis for $\mathcal{A}^E$. Second, if that connected component is consistent, then some other connected component and thus also $\mathcal{A}^E \setminus \{A'\}$ would be inconsistent, which would violate the requirement of minimality. So, either way, we obtain the required contradiction and are done. □

Next, we are going to define two families of heuristics to further speed up Justify in practice. Every concrete heuristic belonging to one of these two families will be tied to specific axioms. We are going to present some examples for such concrete heuristics, for some of the best-known axioms from the literature [32]. While, as we just saw, Justify is defined and works correctly for *any* corpus of axioms, it makes sense to try and fine-tune it for use with such widely used and widely accepted axioms. But we stress that the choice of heuristics does not in any way limit the range of axioms we can use; the algorithm will continue to work correctly for any combination of corpus axioms and any combination of heuristics.

3.2 Heuristics: Omitting Implied Instances

The goal of the first family of heuristics is to avoid the generation of redundant instances. Here, a redundant instance is an instance whose constraint (its "effect" over some profiles) is already enforced (or *implied*) by other instances of the same axiom. Observe that we can omit the generation of such an implied instance, as long as we do generate the instances that imply it: everything that can be explained by the former can be explained by the latter instead.

Example 2. Consider some profile R_1. Suppose that by raising the support of alternative x we obtain another profile R_2. Thus, there is an instance of Positive Responsiveness (let's call it $A_{1,2}$) that constrains R_1 and R_2 (by prescribing that, if x wins in R_1, then it must be the unique winner in R_2). Now, suppose that by further increasing the support of x in R_2 we obtain another profile R_3. Hence, there also is an instance of Positive Responsiveness constraining R_2 and R_3 (let's call it $A_{2,3}$). But, clearly, R_3 can also be obtained directly from R_1 by raising the support of x. So there is an instance connecting R_1 to R_3 directly (let's call it $A_{1,3}$). Note that the constraint enforced by $A_{1,3}$ is implied by $A_{1,2}$ and $A_{2,3}$; hence, as long as we generate the latter two, we can avoid generating $A_{1,3}$. △

Let us make this idea formal.

Definition 3. *Given an axiom A, one of its instances $A' \triangleleft A$, and a set $\mathcal{A}'$ of further instances of A such that $\mathbb{I}(\mathcal{A}') \subseteq \mathbb{I}(A')$, we say that A' is an **implied instance** of A with **implicant set** $\mathcal{A}'$.*

Note that $\mathbb{I}(\mathcal{A}') \subseteq \mathbb{I}(A')$ means that every rule that satisfies all instances in $\mathcal{A}'$ will also satisfy the implied instance A'. Hence, explicitly imposing A' on top of $\mathcal{A}'$ has no impact on the range of possible rules we consider when searching for a justification.

We call an *implied-instance heuristic* any refinement of Justify that, in Step 3, instead of exploring $\mathcal{G}_{\mathbb{A}}$, explores $\mathcal{G}_{\mathcal{A}^-}$, where $\mathcal{A}^- \subseteq \{A' \triangleleft A \mid A \in \mathbb{A}\}$ can be thought of as having been obtained as follows. Initialise $\mathcal{A}^-$ with the full set $\{A' \triangleleft A \mid A \in \mathbb{A}\}$ of corpus axiom instances and then repeat any number of times: remove an instance A' from $\mathcal{A}^-$ that has an implicant set that is fully included in $\mathcal{A}^-$. We now show that any such heuristic preserves the correctness of the main algorithm established in Theorem 1.

Proposition 2. *Algorithm* Justify *remains correct when we refine it with any number of implied-instance heuristics.*

Proof (sketch). Soundness clearly is not affected by omitting some of the instances from the instance graph.

To prove completeness we need to show that, if Justify returns *some* justification, then so does the refined algorithm. Let us consider the case where Justify only returns justifications featuring implied instances. We need to make sure that Solve will eventually be given a set of instances, free from implied instances, from which *some* justification can be retrieved. The existence of such a set follows from the fact that the explanation of any justification found by Justify that features some of the implied instances can be rewritten by replacing those implied instances by their implicants.[8] Because implied instances and their implicants refer to profiles that are connected, without any limit on depth, these implicants will eventually be generated. Thus there exists at least one justification that can be retrieved by the refined algorithm. □

3.3 Heuristics: Adding Derived Axioms

During initial experiments, we observed that certain axioms were frequently involved in the same "patterns of explanation". That is, the underlying arguments of many justifications involving such axioms displayed a recurring structure. Our second family of heuristics leverages this *a-posteriori* knowledge regarding the structure of typical justifications in order to speed up the search.

To formulate such a heuristic, we take a commonly observed pattern of interaction between axioms (involving multiple profiles) and amalgamate it into a single new axiom (whose instances mention only one profile at a time), which we refer to as a *derived axiom*. The goal here is to "reduce" an explanation that would require a depth of d into a single instance that only requires a depth of d' (with $d' < d$). Doing so allows us to, sometimes, stop the generation at depth d' instead of exploring the graph up to depth d.

Example 3. Consider the axioms of Positive Responsiveness and Cancellation. We call a *Cancellation profile* any profile where all alternatives tie in pairwise majority contests. We can define the following derived axiom $A^\star$: *"For every profile R and every alternative $x^\star$, if R can be obtained from some Cancellation profile by raising the support for $x^\star$, then $\{x^\star\}$ should win in R."* It is easy to see that any voting rule F that satisfies Positive Responsiveness and Cancellation must satisfy $A^\star$ as well. Furthermore, $A^\star$ refers to just one profile R.[9] Hence, instead of searching through the graph

[8] Such an explanation may originally fail to satisfy the explanatoriness requirement. In this case we can remove instances one-by-one until minimality is restored. At the same time, note that the nontriviality requirement is not affected.

[9] Note that another profile (the "Cancellation profile") is implicitly mentioned. However, it is possible to generate the instances of this derived axiom quickly during the expansion of R without having to reach and expand said Cancellation profile.

for a suitable Cancellation profile R' connected to R, we can directly generate the relevant instance of this axiom while expanding R. △

We now formally define the concept of a derived axiom.[10]

Definition 4. *A **derived axiom** of a set of axioms $\mathcal{A}$ with $\mathbb{I}(\mathcal{A}) \neq \emptyset$ is an axiom $A_d \notin \mathcal{A}$ such that $\mathbb{I}(\mathcal{A}) \subseteq \mathbb{I}(A_d)$ and, for all axiom instances $A' \triangleleft A_d$, it is the case that $|\mathbb{P}(A')| = 1$.*

The restriction of $|\mathbb{P}(A')| = 1$ is enforced for performance reasons only: when we impose a maximum depth of d on the generation, for any profile R at distance d from the given profile $R^\star$, we only generate the instances that exclusively mention R. Hence, if we reduce part of a complex explanation constraining the outcome for R into a single instance only mentioning R itself, we ensure that this part of the explanation is discovered as soon as R is reached (and thus, with the smallest depth possible).

Given a justification problem $\mathcal{J} = \langle R^\star, X^\star, \mathbb{A} \rangle$ and a set of derived axioms $\mathcal{A}_d$ for axioms in $\mathbb{A}$, let $\mathcal{J}_{\mathcal{A}_d}$ be a variant of $\mathcal{J}$ where $\mathbb{A}$ has been extended with $\mathcal{A}_d$. Then, we say that a *derived-axiom heuristic* (w.r.t. $\mathcal{A}_d$) is a refinement of Justify defined as Justify$_{\mathcal{A}_d}(\mathcal{J}, d)$ = Justify$(\mathcal{J}_{\mathcal{A}_d}, d)$. Note that, since the two notions are completely independent, an algorithm implementing both implied-instance and derived-axiom heuristics is naturally defined: we call any such algorithm a *heuristic variant* of Justify.

Now consider one such heuristic variant Justify$_H$ of Justify, and let $\mathcal{A}_d$ be the set of derived axioms extending the original corpus. Assuming again the correctness of Solve, it follows immediately from Theorem 1 and Proposition 2 that, for any input problem $\mathcal{J}$, Justify$_H$ will eventually find a justification for $\mathcal{J}_{\mathcal{A}_d}$, if one exists. However, as that justification might feature instances of some of the derived axioms and thus violate the relevance requirement, it is not necessarily a justification for $\mathcal{J}$. We argue that this is unlikely to be a problem in practice; in fact, it might even be desirable. Indeed, if a derived axiom encapsulates a complex pattern of explanation arising from multiple instances, it might be fruitful to present the concise derived axiom instance to the users instead of the full explanation. Furthermore, as we shall see next, even if we do want to insist on a justification that only involves axioms from the original corpus $\mathbb{A}$, this need not be a problem. Indeed, if $\mathbb{A}$ is nontrivial, then we can always "convert" a justification for $\mathcal{J}_{\mathcal{A}_d}$ back to a justification for $\mathcal{J}$. Let Justify^+_H be the algorithm that first executes Justify$_H$, searching for a justification using the extended corpus of axioms, and then replaces every derived axiom in the normative basis found by the original axioms deriving it and any instance of a derived axiom with the original instances deriving it. In case the resulting explanation is not minimal, we remove instances one-by-one until it is. We now show that this algorithm is correct when used with nontrivial corpora.

Proposition 3. *For nontrivial corpora, algorithm* Justify *refined with any number of implied-instance heuristics remains correct when further refined with any number of derived-axiom heuristics and followed by the aforementioned postprocessing routine.*

Proof (sketch). Consider a justification problem $\mathcal{J} = \langle R^\star, X^\star, \mathbb{A} \rangle$ and a set $\mathcal{A}_d$ of derived axioms for a nontrivial corpus $\mathbb{A}$. By Proposition 2 we know that, internally, Justify^+_H will retrieve a valid justification $\langle \mathcal{A}^N, \mathcal{A}^E \rangle$ for $\mathcal{J}_{\mathcal{A}_d}$ if one exists. Then, in the postprocessing phase, any instance of a derived axiom in $\mathcal{A}^E$ is replaced by the instances of the axioms in $\mathbb{A}$ that imply it.[11] Finally, redundant instances (that would violate minimality) are removed. As this new set $\mathcal{A}^E_\star$ of instances entails $\mathcal{A}^E$, there exists no rule picking an outcome different from $X^\star$ for $R^\star$ while also satisfying $\mathcal{A}^E_\star$ (by explanatoriness of $\mathcal{A}^E$). Call $\mathcal{A}^N_\star$ the normative basis obtained from $\mathcal{A}^N$ where every derived axiom has been replaced by the axioms it is derived from. The nontriviality of $\mathcal{A}^N_\star \subseteq \mathbb{A}$ follows from the nontriviality of $\mathbb{A}$. Hence the pair $\langle \mathcal{A}^N_\star, \mathcal{A}^E_\star \rangle$ returned by Justify^+_H is a correct justification for $\mathcal{J}$. □

If $\mathbb{I}(\mathbb{A}) = \emptyset$, i.e., if the corpus is not nontrivial, then we are not guaranteed to be able to perform such a conversion, since the resulting normative basis might violate nontriviality. However, we can slightly modify the above algorithm to perform a check of the resulting, converted justifications: if they are proper justifications for the input problem $\mathcal{J}$, then they are returned.[12] Otherwise, the search is resumed. Since a derived-axiom heuristic only adds instances to the instance graph, all the original justifications for $\mathcal{J}$ will still be retrievable, so this algorithm would be trivially correct.

4 PERFORMANCE ANALYSIS

In this section, we report on the experiments we conducted to analyse the performance of our approach [21]. We ran our algorithm on a set of justification problems with moderately-sized profiles, using both randomly generated profiles and profiles extracted from real-world data [19]. Overall, we found that, on a powerful computer, our method can handle such queries in a matter of minutes in the worst case and in a matter of seconds on average, thereby improving over the baseline algorithm [9] by several orders of magnitude.

4.1 Experimental Setup and Results

Our experiments were designed as follows. We searched for justifications grounded in the corpus of axioms presented in Section 2.1. This corpus is *nontrivial*, meaning that there exist rules that satisfy all of its axioms.[13] Hence, for any given profile there can be a justification for *at most one* target outcome. Given a set of test profiles, we thus tried to compute a justification for *some* target outcome for every profile in the set (by iterating over all possible outcomes until one is found), and we measured the time it took to do so.

We focused on profiles ranging from 2 to 12 voters, and 3 to 4 alternatives. Although of limited size, we argue that such scenarios are already large enough to capture some real-world, high-stakes scenarios: for example, a research group looking for a PhD candidate to hire, a small committee voting to enact a new policy, or the strategic unit of a company choosing where to open the next branch. To generate the test profiles, we used three different approaches.

- **Exhaustive.** In the first approach, also followed by Boixel and Endriss [9], we exhaustively test our algorithm on every

[10] Our notion of derived axiom is reminiscent of the use of *lemmas* in automated reasoning [3]. We do not use this terminology here to stress the fact that a derived axiom typically is not just a technical tool but also has some normative appeal.

[11] Note that, at least in principle, this is always possible. If A_d is implied by (that is, derived from) some axioms in $\mathcal{A}$, then any instance of A_d is implied by (at least) the set of all instances of $\mathcal{A}$. In practice, any sensible derived axiom will allow for its instances to be replaced by just a handful of the instances of the original axioms.

[12] To do this, we need to check whether the resulting normative basis is nontrivial.

[13] For example, the Borda rule is one of several such rules [30].

possible profile (involving a given number of voters and alternatives), up to symmetry breaking.[14] Note that this is feasible only for the smaller scenarios considered.

- **Random.** In the second approach, we sample uniformly from the space of all profiles (of a given size). This is known as the *impartial culture* (IC) assumption [13].[15]
- **Preflib.** In the third approach, we generate profiles by sampling from Preflib, an online library of real-world preference datasets [19]. Since most profiles in Preflib involve a large number of voters, we use the following bootstrapping approach to generate a profile: given a Preflib profile R with n voters, we sample n' (with $n' < n$) preference orders, and the probability of extracting each preference is proportional to the number of voters reporting that ballot in R. Since this kind of bootstrap sampling, in expectation, preserves the distribution of the preference orders, we consider it a reasonable way of generating profiles. It has also been used in a number of other works in social choice theory [15, 23, 25].

Specifically, for the case of 3 alternatives, we performed exhaustive generation on all profiles with between 2 and 8 voters, and we used both random and Preflib generation for profiles of 10 and 12 voters. For the case of 4 alternatives, we performed exhaustive generation on all profiles with between 2 and 4 voters, and we used random and Preflib generation for profiles of 5 to 8 voters. For both sampling approaches, for each of the scenarios considered, we generated 180 profiles. Specifically, for the Preflib generation, we selected at random 30 profiles, and for each of them generated 6 random subprofiles (by sampling as described above).

A further clarification is in order. In the case of 4 alternatives, we imposed a maximum search depth of 3, for several reasons. First, the *depth of a justification* (i.e., the minimum depth at which the instance graph must be explored to retrieve it) generally correlates with its intricacy. Overly deep justifications can be hard to understand. Second, in our preliminary experiments, we found that in fact most justifications are found within a depth of at most 3, so in case no justification has been found at that depth it is unlikely that one would be found later on, meaning that restricting the depth can be a good trade-off between performance and full coverage.

We used a heuristic variant of JUSTIFY featuring several implied-instance heuristics (for NEUTRALITY, REINFORCEMENT, and POSITIVE RESPONSIVENESS) and derived-axiom heuristics (for NEUTRALITY, POSITIVE RESPONSIVENESS, and CANCELLATION). As our encoding language we used SAT (i.e., propositional logic), and we employed MARCO [17] for the MUS enumeration and LINGELING [6] for checking satisfiablity. All experiments have been run on a machine of the Dutch national computing cluster LISA, equipped with an Intel Xeon Silver 4110 CPU (2.10GHz).[16]

Generation	Explained Profiles	Avg time (s)	Max time (s)
Three Alternatives			
Exhaustive			
2 voters	5/5 (100%)	0.11 (± 0.01)	0.13
3 voters	9/10 (88.89%)	0.12 (± 0.01)	0.14
4 voters	22/23 (95.65%)	0.14 (± 0.13)	0.77
5 voters	42/42 (100%)	0.23 (± 0.31)	1.50
6 voters	83/83 (100%)	0.23 (± 0.37)	2.33
7 voters	132/132 (100%)	0.41 (± 0.69)	4.02
8 voters	222/222 (100%)	0.45 (± 1.06)	9.99
Random			
10 voters	180/180 (100%)	0.40 (± 1.12)	10.74
12 voters	180/180 (100%)	0.93 (± 1.99)	9.26
Preflib			
10 voters	180/180 (100%)	0.48 (± 1.28)	9.36
12 voters	180/180 (100%)	1.19 (± 3.95)	25.69
Four Alternatives			
Exhaustive			
2 voters	15/17 (88.24%)	0.21 (± 0.09)	0.35
3 voters	68/111 (61.26%)	1.11 (± 1.14)	4.95
4 voters	509/762 (66.80%)	3.15 (± 3.82)	16.76
Random			
5 voters	113/180 (62.78%)	12.40 (± 12.67)	42.12
6 voters	132/180 (73.33%)	16.58 (± 21.92)	77.32 [1′17″]
7 voters	140/180 (77.78%)	63.23 (± 65.19)	183.15 [3′03″]
8 voters	167/180 (92.78%)	155.85 (± 189.46)	545.78 [9′05″]
Preflib			
5 voters	133/180 (73.89%)	7.18 (± 10.09)	40.82
6 voters	145/180 (80.56%)	14.40 (± 22.84)	91.57 [1′32″]
7 voters	149/180 (82.78%)	25.83 (± 43.37)	176.67 [2′57″]
8 voters	161/180 (89.44%)	57.49 (± 103.45)	493.46 [8′13″]

Table 1: Experimental results. Average are given with standard deviation in brackets (also expressed in minutes for the slowest cases).

Table 1 shows the results of our experiments. Note that we also ran the same experiments without any heuristics, and we were able to observe a clear positive impact on performance when using heuristics. We do not present these additional results here in any detail, but, for example, the experiment concerning all 222 profiles with 8 voters and 3 alternatives took approximately half an hour in total (compared to less than 2 minutes with the heuristics). A more extreme example is the experiment concerning profiles with 4 voters and 4 alternatives: running all 762 of them took around 22 hours (compared to less than 40 minutes with heuristics).

We stress that our findings are specific to the corpus of axioms we employed and the profiles we considered. This is true, in particular, with regards to the results about the proportion of profiles for which we were able to justify an outcome. Interestingly, all the justifications found turned out to be justifications of Borda winners. This is not entirely surprising, given that the axioms of Section 2.1 closely resemble (but are not completely equivalent to) the axioms used in Young's characterisation of the Borda rule [30].

[14] Indeed, if several profiles are identical up to a renaming of the alternatives, then it is sufficient to test our algorithm on only one representative of this equivalence class.

[15] Note that, due to our symmetry breaking approach, in expectation, the results of an experiment for the same number of voters and alternatives might differ for the exhaustive and the random approach. To obtain the same results, one would need to use what is known as the *impartial anonymous and neutral culture* assumption [12].

[16] Besides the experiments we report on here, we also conducted experiments with two further sampling methods, namely the *Polya-Eggenberger urn model* [4, 27] and uniform sampling restricted to *single-peaked profiles* [8, 29]. These results were consistent with the findings we report here: all queries were answered within a matter of minutes, with the best performance obtained for profiles generated from real-life data. For more details, we refer the reader to the MSc thesis of the first author [20].

4.2 Discussion: Three Alternatives

As is clear from the upper part of Table 1, for most profiles with 3 alternatives a justification can be found within a reasonable amount of time: the absolute maximum time encountered is less than half a minute. Moreover, on average, computing a justification takes less than one second on almost every scenario considered, and takes slightly more (1.19 seconds) only for the largest case considered.

The vast majority of retrieved justifications were found within a maximum depth of 3 (98.95%). Specifically, for the sampling approaches, only one single profile required a depth of 4 (generated with Preflib, for the case of 10 voters). The average depth needed to justify the profiles (across all generation methods and scenarios) was 0.94 (±0.73) and the median depth was 1. Hence, most justifications can be found close to the profile of interest.

Note that the only profiles for which we were not able to justify any outcome involved either 3 or 4 voters. For 3 voters, no justification was retrieved for the profile where one voter reports $a \succ b \succ c$ and two voters report $c \succ a \succ b$. The Borda outcome for this profile is $\{a, c\}$, which can be justified with the classical (stronger) formulation of Reinforcement (together with Neutrality, the Pareto Principle, and Cancellation) by referring to a profile of 5 voters. However, our weaker formulation of Reinforcement only considers profiles with up to $n^\star = 3$ voters (when searching for a justification for a 3-voter profile), so this would not be a valid justification for our corpus. The same issue also arises for one of the 4-voter profiles, where one voter reports $a \succ b \succ c$ and three voters report $c \succ a \succ b$. Similarly as before, the Borda outcome for this profile (where c is the unique winner) can only be explained by referring to a profile that is larger than the profile of interest.

4.3 Discussion: Four Alternatives

The results of the experiments with 4 alternatives case are shown on the lower part of Table 1. As one would expect and as is clearly visible from the results, compared to the case of 3 alternatives, we now are facing a much harder computational problem.

First of all, observe that, across all the experiments considered, for the majority of profiles (74%) a justification can still be found within a reasonable amount of time. Indeed, all queries are answered within a matter of minutes; the absolute worst running time is slightly more than 9 minutes. Although this would be too much for a real-time application, we can envision a tool that works in an asynchronous fashion, e.g., by returning the results via email.

We now also observe greater average and median depths, when compared to the case of 3 alternatives. Across all retrieved justifications, the average depth was 1.61 (±0.90), and the median depth was 2. Indeed, profiles involving more alternatives are usually more complex, so one can expect justifications to be more intricate.

Furthermore, note that outcomes for profiles sampled from Preflib were justified more frequently than for those sampled from the impartial culture (save for the case of 8 voters), and they also required less computing time. Although the number of profiles analysed is limited and our results are specific to the corpus considered, this is very encouraging, given that our goal is to justify outcomes for real-world decision making scenarios.

Moreover, we can observe a general tendency regarding the number of voters: as $n^\star$ increases, the percentage of profiles with a justification generally increases as well (with the exception of the case of 2 voters). This mirrors the pattern observed for 3 alternatives, where the only profiles for which we retrieved no justifications were to be found in the smallest scenarios. A possible explanation for this trend might be related to the role played by Reinforcement. A justification involving this axiom usually is obtained by "breaking down" the given profile $R^\star$ into two subprofiles.

Example 4. Consider the profile $R^\star$ with two voters reporting $a \succ b \succ c$ and one reporting $c \succ b \succ a$. We can partition this profile into two subprofiles: one where there is a single voter reporting $a \succ b \succ c$ (where a wins by Faithfulness), and a 2-voter profile where one votes $a \succ b \succ c$ and the other $c \succ b \succ a$ (so all alternatives tie by Cancellation). Now we can construct a justification for $\{a\}$ for $R^\star$ by appealing to Reinforcement. △

This might explain why, as $n^\star$ grows larger, more profiles have justifiable outcomes: the larger the number of voters, the more choices there are to decompose the given profile $R^\star$ into subprofiles, and so the more likely it is that at least one such partition will lead to a justification involving Reinforcement. We consider this another positive finding: in most real-world decision making scenarios we should expect to see more than just a couple of individuals involved, so it is encouraging that our approach works well for such scenarios.

5 CONCLUSION

We have presented an algorithm to iteratively generate axiom instances that might turn out to be relevant to the justification of an outcome for a given preference profile. This generation proceeds by exploring the graph spanned by the set of all axiom instances over the set of all profiles. We have further refined this algorithm by using heuristics that exploit the structure of specific axioms. The resulting algorithm, when combined with state-of-the-art tools for SAT solving and MUS enumeration, has allowed us to make significant progress on the challenge of solving the axiomatic justification problem for scenarios that are sufficiently complex to be of practical interest for applications. Our experimental results for real-world preference data are particularly encouraging.

While these results, and the concrete heuristics used to refine our algorithm, are specific to our corpus of axioms, the techniques presented are generic and adapting the heuristics to other axioms can be expected to be straightforward in most cases.

We can identify several important directions for future work. First, our experimental study could be extended to a broader range of axioms. Second, our algorithm could be used as a tool in a qualitative study of justifications and explanations, e.g., to better understand typical patterns observed in explanations. Third, our algorithm could be refined with a search strategy that favours justifications with certain characteristics, such as being particularly short or favouring the involvement of some axioms over others [9, 16]. The design of such a strategy should be informed by a user study to establish what kinds of justifications users are most likely to understand and accept. Finally, more research is needed on how best to present an explanation to a nonexpert user in practice.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their feedback.

REFERENCES

[1] Sanjeev Arora and Boaz Barak. 2009. *Computational Complexity: A Modern Approach* (1st ed.). Cambridge University Press, New York.

[2] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. 2020. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. *Information Fusion* 58 (2020), 82–115.

[3] Owen L. Astrachan and Mark E. Stickel. 1992. Caching and Lemmaizing in Model Elimination Theorem Provers. In *Proceedings of the 11th International Conference on Automated Deduction (CADE-1992)*. Springer, 224–238.

[4] Sven Berg. 1985. Paradox of Voting under an Urn Model: The Effect of Homogeneity. *Public Choice* 47, 2 (1985), 377–387.

[5] Christain Bessiere. 2006. Constraint Propagation. In *Handbook of Constraint Programming*, Francesca Rossi, Peter van Beek, and Toby Walsh (Eds.). Elsevier.

[6] Armin Biere. 2017. CaDiCaL, Lingeling, Plingeling, Treengeling, YalSAT Entering the SAT Competition 2017. In *Proceedings of the SAT Competition 2017: Solver and Benchmark Descriptions*, Tomáš Balyo, Marijn Heule, and Matti Järvisalo (Eds.). University of Helsinki, 14–15.

[7] Armin Biere, Marijn Heule, and Hans van Maaren (Eds.). 2009. *Handbook of Satisfiability*. IOS Press.

[8] Duncan Black. 1948. On the Rationale of Group Decision-Making. *Journal of Political Economy* 56, 1 (1948), 23–34.

[9] Arthur Boixel and Ulle Endriss. 2020. Automated Justification of Collective Decisions via Constraint Solving. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS-2020)*. IFAAMAS, Richland, SC, 168–176.

[10] Arthur Boixel and Ronald de Haan. 2021. On the Complexity of Finding Justifications for Collective Decisions. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI-2021)*. AAAI Press, Palo Alto, CA, 5194–5201.

[11] Olivier Cailloux and Ulle Endriss. 2016. Arguing about Voting Rules. In *Proceedings of the 15th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS-2016)*. IFAAMAS, Richland, SC, 287–295.

[12] Ömer Eğecioğlu. 2009. Uniform Generation of Anonymous and Neutral Preference Profiles for Social Choice Rules. *Monte Carlo Methods and Applications* 15, 3 (2009), 241–255.

[13] Mark B. Garman and Morton I. Kamien. 1968. The Paradox of Voting: Probability Calculations. *Behavioral Science* 13, 4 (1968), 306–316.

[14] Christian Geist and Dominik Peters. 2017. Computer-Aided Methods for Social Choice Theory. In *Trends in Computational Social Choice*, Ulle Endriss (Ed.). AI Access.

[15] Vahid Hashemi and Ulle Endriss. 2014. Measuring Diversity of Preferences in a Group. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI-2014)*. IOS Press, Amsterdam, 423–428.

[16] Ulrich Junker. 2004. QuickXplain: Preferred Explanations and Relaxations for Over-Constrained Problems. In *Proceedings of the 19th AAAI Conference on Artificial Intelligence (AAAI-2004)*. AAAI Press, Palo Alto, CA, 167–172.

[17] Mark H. Liffiton, Alessandro Previti, Ammar Malik, and João Marques-Silva. 2016. Fast, Flexible MUS Enumeration. *Constraints* 21, 2 (2016), 223–250.

[18] Mark H. Liffiton and Karem A. Sakallah. 2008. Algorithms for Computing Minimal Unsatisfiable Subsets of Constraints. *Journal of Automated Reasoning* 40, 1 (2008), 1–33.

[19] Nicholas Mattei and Toby Walsh. 2013. PrefLib: A Library of Preference Data. In *Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT-2013)*. Springer, Berlin, 259–270. http://preflib.org

[20] Oliviero Nardi. 2021. *A Graph-Based Algorithm for the Automated Justification of Collective Decisions*. Master's thesis. ILLC, University of Amsterdam.

[21] Oliviero Nardi, Arthur Boixel, and Ulle Endriss. 2022. Supplementary Material for "A Graph-Based Algorithm for the Automated Justification of Collective Decisions". Zenodo. Available at https://doi.org/10.5281/zenodo.5901351.

[22] Dominik Peters, Ariel D. Procaccia, Alexandros Psomas, and Zixin Zhou. 2020. Explainable Voting. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS-2020)*. Curran Associates, Inc., Red Hook, NY, 1525–1534.

[23] Anna Popova, Michel Regenwetter, and Nicholas Mattei. 2013. A Behavioral Perspective on Social Choice. *Annals of Mathematics and Artificial Intelligence* 68, 1 (2013), 5–30.

[24] Ariel D. Procaccia. 2019. Axioms Should Explain Solutions. In *The Future of Economic Design*, Jean-FranÃğois Laslier, Hervé Moulin, M. Remzi Sanver, and William S. Zwicker (Eds.). Springer, Berlin.

[25] Michel Regenwetter, Aeri Kim, Arthur Kantor, and Moon-Ho R Ho. 2007. The Unexpected Empirical Consensus among Consensus Methods. *Psychological Science* 18, 7 (2007), 629–635.

[26] Francesca Rossi, Peter van Beek, and Toby Walsh (Eds.). 2006. *Handbook of Constraint Programming*. Elsevier.

[27] Stanisław Szufa, Piotr Faliszewski, Piotr Skowron, Arkadii Slinko, and Nimrod Talmon. 2020. Drawing a Map of Elections in the Space of Statistical Cultures. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS-2020)*. IFAAMAS, Richland, SC, 1341–1349.

[28] Pingzhong Tang and Fangzhen Lin. 2009. Computer-Aided Proofs of Arrow's and Other Impossibility Theorems. *Artificial Intelligence* 173, 11 (2009), 1041–1053.

[29] Toby Walsh. 2015. Generating Single Peaked Votes. `arXiv:1503.02766 [cs.GT]`.

[30] H. Peyton Young. 1974. An Axiomatization of Borda's Rule. *Journal of Economic Theory* 9, 1 (1974), 43–52.

[31] Wenting Zhao and Mark H. Liffiton. 2016. Parallelizing Partial MUS Enumeration. In *Proceedings of the 28th International Conference on Tools with Artificial Intelligence (ICTAI-2016)*. IEEE Computer Society, San Jose, CA, 464–471.

[32] William S. Zwicker. 2016. Introduction to Voting Theory. In *Handbook of Computational Social Choice*, Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia (Eds.). Cambridge University Press, New York, 23–56.