On the Computational Complexity of Non-dictatorial Aggregation

Lefteris Kirousis[1], Phokion G. Kolaitis[2], and John Livieratos[1](✉)

[1] Department of Mathematics,
National and Kapodistrian University of Athens, Athens, Greece
{lkirousis,jlivier89}@math.uoa.gr
[2] Computer Science Department,
UC Santa Cruz and IBM Research - Almaden, Santa Cruz, USA
kolaitis@cs.ucsc.edu

Abstract. We investigate when non-dictatorial aggregation is possible from an algorithmic perspective, where non-dictatorial aggregation means that the votes cast by the members of a society can be aggregated in such a way that the collective outcome is not simply the choices made by a single member of the society. We consider the setting in which the members of a society take a position on a fixed collection of issues, where for each issue several different alternatives are possible, but the combination of choices must belong to a given set X of allowable voting patterns. Such a set X is called a possibility domain if there is an aggregator that is non-dictatorial, operates separately on each issue, and returns values among those cast by the society on each issue. We design a polynomial-time algorithm that decides, given a set X of voting patterns, whether or not X is a possibility domain. Furthermore, if X is a possibility domain, then the algorithm constructs in polynomial time such a non-dictatorial aggregator for X. We also design a polynomial-time algorithm that decides whether X is a uniform possibility domain, that is, whether X admits an aggregator that is non-dictatorial even when restricted to any two positions for each issue. As in the case of possibility domains, the algorithm also constructs in polynomial time a uniform non-dictatorial aggregator, if one exists.

1 Introduction

The study of vote aggregation has occupied a central place in social choice theory. A broad framework for carrying out this study is as follows. There is a fixed collection of issues on each of which every member of a society takes a *position*, that is, for each issue, a member of the society can choose between a number of alternatives. However, not every combination of choices is allowed, which means that the vector of the choices made by a member of the society must belong to a given set X of allowable voting patterns, called *feasible evaluations*. The goal is to investigate properties of *aggregators*, which are functions that take as input the votes cast by the members of the society and return as output a feasible

J. Desharnais et al. (Eds.): RAMiCS 2018, LNCS 11194, pp. 350–365, 2018.
https://doi.org/10.1007/978-3-030-02149-8_21

evaluation that represents the collective position of the society on each of the issues at hand. A concrete key problem studied in this framework is to determine whether or not a *non-dictatorial* aggregator exists, i.e., whether or not it is possible to aggregate votes in such a way that individual members of the society do not impose their voting preferences on the society. A set X of feasible evaluations is called a *possibility domain* if it admits a non-dictatorial aggregator; otherwise, X is called an *impossibility domain.* This framework is broad enough to account for several well-studied cases of vote aggregation, including the case of *preference* aggregation for which Arrow [1] established his celebrated impossibility theorem and the case of *judgment* aggregation [9].

The investigation of the existence of non-dictatorial aggregators is typically carried out under two assumptions: (a) the aggregators are *independent of irrelevant alternatives* (IIA); and (b) the aggregators are *conservative* (also known as *supportive*). The IIA assumption means that the aggregator is an issue-by-issue aggregator, so that an IIA aggregator on m issues can be identified with an m-tuple $(f_1, \dots, f_m)$ of functions aggregating the votes on each issue. The conservativeness (or supportiveness) assumption means that, for every issue, the position returned by the aggregator is one of the positions held by the members of the society on that issue.

By now, there is a body of research on identifying criteria that characterize when a given set X of feasible evaluations is a possibility domain. The first such criterion was established by Dokow and Holzman [7] in the Boolean framework, where, for each issue, there are exactly two alternatives (say, 0 and 1) for the voters to choose from. Specifically, Dokow and Holzman [7] showed that a set $X \subseteq \{0,1\}^m$ is a possibility domain if and only if X is affine or X is not totally blocked. Informally, the notion of *total blockedness*, which was first introduced in [13], asserts that any position on any issue can be inferred from any position on any issue. As regards the non-Boolean framework (where, for some issues, there may be more than two alternatives), Dokow and Holzman [8] extended the notion of total blockedness and used it to give a sufficient condition for a set X of feasible evaluations to be a possibility domain. Szegedy and Xu [16] used tools from universal algebra to characterize when a totally blocked set X of feasible evaluations is a possibility domain. A consequence of these results is that a set X of feasible evaluations is a possibility domain if and only if X admits a binary non-dictatorial aggregator or a ternary non-dictatorial aggregator; in other words, non-dictatorial aggregation is possible for a society of some size if and only if it is possible for a society with just two members or with just three members. This line of work was pursued further by Kirousis et al. [10], who characterized possibility domains in terms of the existence of binary non-dictatorial aggregators or ternary non-dictatorial aggregators of a particular form.

The aforementioned investigations have characterized possibility domains (in both the Boolean and the non-Boolean frameworks) in terms of *structural* conditions. Our goal is to investigate possibility domains using the *algorithmic lens* and, in particular, to study the following algorithmic problem: given a set X of feasible evaluations, determine whether or not X is a possibility domain. Szegedy and Xu [16, Theorem 36] give algorithms for this problem, but these algorithms have very high running time; in fact, they run in exponential time in the number of issues and in the number of positions over each issue, even when confined to the Boolean framework.

We design a polynomial-time algorithm that, given a set X of feasible evaluations (be it in the Boolean or the non-Boolean framework), determines whether or not X is a possibility domain. Furthermore, if X is a possibility domain, then the algorithm produces a binary non-dictatorial or a ternary non-dictatorial aggregator for X. Along the way, we also show that there is a polynomial-time algorithm for determining, given X, whether or not it is totally blocked.

After this, we turn our attention to *uniform possibility domains*, which were introduced in [10] and form a proper subclass of the class of possibility domains. Intuitively, uniform possibility domains are sets of feasible evaluations that admit an aggregator that is non-dictatorial even when restricted to any two positions for each issue. In [10], a tight connection was established between uniform possibility domains and constraint satisfaction by showing that multi-sorted conservative constraint satisfaction problems on uniform possibility domains are tractable, whereas such constraint satisfaction problems defined on all other domains are NP-complete.

Here, using Carbonnel's result in [5], we give a polynomial-time algorithm for the following decision problem: given a set X of feasible evaluations (be it in the Boolean or the non-Boolean framework), determine whether or not X is a uniform possibility domain; moreover, if X is a uniform possibility domain, then the algorithm produces a suitable uniform non-dictatorial aggregator for X.

The results reported here contribute to the developing field of computational social choice and pave the way for further exploration of algorithmic aspects of vote aggregation.

2 Preliminaries and Earlier Work

In this section, we formally describe the framework we will work on and the necessary tools in order to obtain our results. In Subsect. 2.1 we consider possibility domains both in the Boolean and non-Boolean case, whereas in Subsect. 2.2 we turn our attention to uniform possibility domains.

2.1 Possibility Domains

Let $I = \{1, \ldots, m\}$ be a set of issues. Assume that the *possible position values* of an individual (member of a society) for issue j are given by the finite set A_j, where $j = 1, \ldots, m$.

We assume that each set A_j has cardinality at least 2. If $|A_j| = 2$ for all $j \in \{1, \ldots, m\}$, we say that we are in the *binary* or *Boolean* framework; otherwise we say that we are in the *non-binary* or *non-Boolean* framework.

An *evaluation* is an element of $\prod_{j=1}^m A_j$. Let $X \subseteq \prod_{j=1}^m A_j$ be a set of *permissible* or *feasible* evaluations. To avoid degenerate cases, we assume that for each $j = 1, \ldots, m$, the j-th projection X_j of X is equal to A_j. Note that this assumption in no way implies that $X = \prod_{j=1}^m X_j$.

Let $n \geq 2$ represent the number of individuals. We view the elements of $x \in X^n$ as $n \times m$ matrices that represent the choices of all individuals over every issue. The element x_j^i of such a matrix x will be the choice of the i-th individual over the j-th issue, for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. The i-th row x^i will represent the choices of the i-th individual over every issue, $i = 1, \ldots, n$, and the j-th column x_j the choices of every individual over the j-th issue, $j = 1, \ldots, m$.

To *aggregate* a set of n feasible evaluations, we use m-tuples $\bar{f} = (f_1, \ldots, f_m)$ of functions, where $f_j : A_j^n \to A_j$, $j = 1, \ldots, m$. Such a m-tuple $\bar{f}$ of functions will be called an (n-ary) *aggregator* for X if the following two conditions hold for all $x \in X^n$:

1. $(f_1(x_1), \ldots, f_m(x_m)) \in X$ and
2. $\bar{f}$ is *conservative*, that is, $f_j(x_j) \in \{x_j^1, \ldots, x_j^n\}$, for all $j \in \{1, \ldots, m\}$.

An aggregator $\bar{f} = (f_1, \ldots, f_m)$ is called *dictatorial on* X if there is a number $d \in \{1, \ldots, n\}$ such that $(f_1, \ldots, f_m) = (\mathrm{pr}_d^n, \ldots, \mathrm{pr}_d^n)$, where pr_d^n is the n-ary projection on the d-th coordinate; otherwise, $\bar{f}$ is called *non-dictatorial on* X. We say that X *has a non-dictatorial aggregator* if, for some $n \geq 2$, there is a n-ary non-dictatorial aggregator on X.

A set X of feasible evaluations is a *possibility domain* if it has a non-dictatorial aggregator. Otherwise, it is an *impossibility domain*. A possibility domain is, by definition, one where aggregation is possible for societies of some cardinality, namely, the arity of a non-dictatorial aggregator.

The notion of an aggregator is akin to, but different from, the notion of a polymorphism – a fundamental notion in universal algebra (see Szendrei [17]). Intuitively, a polymorphism is a single-sorted aggregator.

Let A be a finite non-empty set. A *constraint language* over A is a finite set Γ of relations of finite arities.

Let R be an m-ary relation on A. A function $f : A^n \to A$ is a *polymorphism* of R if the following condition holds:

$$\text{if } x^1, \ldots, x^n \in R, \text{ then } (f(x_1), \ldots, f(x_m)) \in R,$$

where $x^i = (x_1^i, \ldots, x_m^i) \in R$, $i = 1, \ldots, n$ and $x_j = (x_j^1, \ldots, x_j^n)$, $j = 1, \ldots, m$. In this case, we also say that R *is closed under* f or that f *preserves* R. Finally, we say that f is a *polymorphism of a constraint language* Γ if f preserves every relation $R \in \Gamma$.

A function $f : A^n \to A$ is *conservative* if, for all $a_1, \ldots, a_n \in A$, we have that $f(a_1, \ldots, a_n) \in \{a_1, \ldots, a_n\}$. Clearly, if $f : A^n \to A$ is a conservative polymorphism of an m-ary relation R on A, then the m-tuple $\bar{f} = (f, \ldots, f)$ is an n-ary aggregator for R.

We say that a ternary operation $f : A^3 \to A$ on an arbitrary set A is a *majority* operation if for all x and y in A,

$$f(x,x,y) = f(x,y,x) = f(y,x,x) = x;$$

we say that f is a *minority* operation if for all x and y in A,

$$f(x,x,y) = f(x,y,x) = f(y,x,x) = y.$$

We also say that a set X of feasible evaluations *admits a majority (respectively, minority) aggregator* if it admits a ternary aggregator every component of which is a majority (respectively, minority) operation. Clearly, X admits a majority aggregator if and only if there is a ternary aggregator $\bar{f} = (f_1, \ldots, f_m)$ for X such that, for all $j = 1, \ldots, m$ and for all two-element subsets $B_j \subseteq X_j$, we have that $f_j \restriction B_j = \text{maj}$, where

$$\text{maj}(x,y,z) = \begin{cases} x & \text{if } x = y \text{ or } x = z, \\ y & \text{if } y = z. \end{cases}$$

Similarly, X admits a minority aggregator if and only if there is a ternary aggregator $\bar{f} = (f_1, \ldots, f_m)$ for X such that, for all $j = 1, \ldots, m$ and for all two-element subsets $B_j \subseteq X_j$, we have that $f_j \restriction B_j = \oplus$, where

$$\oplus(x,y,z) = \begin{cases} z & \text{if } x = y, \\ x & \text{if } y = z, \\ y & \text{if } x = z. \end{cases}$$

In the Boolean framework, a set $X \subseteq \{0,1\}^m$ admits a majority aggregator if and only if the majority operation maj on $\{0,1\}^3$ is a polymorphism of X. Moreover, it is known that this happens precisely when X is a bijunctive logical relation, i.e., X is the set of satisfying assignments of a 2CNF-formula. A set $X \subseteq \{0,1\}^m$ admits a minority aggregator if and only if the minority operation $\oplus$ on $\{0,1\}^3$ is a polymorphism of X. Moreover, it is known that this happens precisely when X is an affine logical relation, i.e., X is the set of solutions of a system of linear equations over the two-element field (see Schaefer [14]).

Example 1. Consider the sets X_1 and X_2 below.

(i) The set $X_1 = \{0,1\}^3 \setminus \{(1,0,1), (0,0,1), (0,0,0)\}$ is bijunctive, since it is the set of satisfying assignments of the 2CNF-formula $(x \vee y) \wedge (y \vee \neg z)$.
(ii) The set $X_2 = \{(0,0,1), (0,1,0), (1,0,0), (1,1,1)\}$ is affine, since it is the set of solutions of the equation $x + y + z = 1$ over the two-element field.
(iii) Both sets X_1 and X_2 are possibility domains, since X_1 admits a majority aggregator and X_2 admits a minority aggregator.

The next two theorems characterize possibility domains in the Boolean and the non-Boolean framework. They are the stepping stones towards showing that the following decision problem is solvable in polynomial time: given a set X of feasible evaluations, is X a possibility domain?

Theorem A (Dokow and Holzman [7]). *Let $X \subseteq \{0,1\}^m$ be a set of feasible evaluations. The following two statements are equivalent.*

1. *X is a possibility domain.*
2. *X is affine or X admits a binary non-dictatorial aggregator.*

Theorem B (Kirousis et al. [10]). *Let X be a set of feasible evaluations. The following two statements are equivalent.*

1. *X is a possibility domain.*
2. *X admits a binary non-dictatorial aggregator, or a majority aggregator, or a minority aggregator.*

We illustrate the two preceding theorems with several examples.

Example 2. Let $X_3 = \{(1,0,0),(0,1,0),(0,0,1)\} \subseteq \{0,1\}^3$ be the set of Boolean triples that contain exactly one 1. By Theorem A, the set X_3 is an impossibility domain, since it is not affine ($\oplus((1,0,0),(0,1,0),(0,0,1)) = (0,0,0) \notin X_3$) and it does not have a binary non-dictatorial aggregator. For the latter, one has to check each of the 62 possible 3-tuples of conservative binary functions over $\{0,1\}$, which is a fairly tedious but straightforward task.

Example 3. The following two sets X_4 and X_5 are possibility domains.

(i) Let $X_4 = \{(0,1,2),(1,2,0),(2,0,1),(0,0,0)\}$. This set has been studied in [7, Example 4]. Let $\bar{f} = (f_1, f_2, f_3)$ be such that, for each $j = 1,2,3$:

$$f_j(x,y,z) = \begin{cases} \mathrm{maj}(x,y,z) & \text{if } |\{x,y,z\}| \leq 2, \\ 0 & \text{else.} \end{cases}$$

Clearly, $\bar{f}$ is a majority operation. To see that $\bar{f}$ is an aggregator for X_4, we need to check that $\bar{f}(a,b,c) \in X_4$, only when a,b,c are *pairwise distinct* vectors of X_4. In this case, $\bar{f}(a,b,c) = (0,0,0) \in X_4$, since the input of each f_j contains either two zeros or three pairwise distinct elements.

(ii) Let $X_5 = X_3 \times X_3$, where X_3 is as in Example 2. It is straightforward to check that $(pr_1^2, pr_1^2, pr_1^2, pr_2^2, pr_2^2, pr_2^2)$ is a non-dictatorial aggregator for X_5. In fact, a stronger fact holds: if Y and Z are arbitrary sets, then their Cartesian product $Y \times Z$ is a possibility domain, since it admits non-dictatorial aggregators of any arity $n \geq 2$, defined as the d-th projection pr_d^n on coordinates from Y and as the d'-th projection $\mathrm{pr}_{d'}^n$ on coordinates from Z, where $1 \leq d, d' \leq n$ and $d \neq d'$.

2.2 Uniform Possibility Domains

We consider a subclass of possibility domains, introduced in [10], called uniform possibility domains.

Let $\bar{f} = (f_1, \ldots, f_m)$ be an n-ary aggregator for X. We say that $\bar{f}$ is a *uniform non-dictatorial* aggregator for X (of arity n) if, for all $j \in \{1, \ldots, m\}$ and for every two-element subset $B_j \subseteq X_j$, it holds that

$$f_j\restriction_{B_j} \neq pr_d^n,$$

for all $d \in \{1, \ldots, n\}$. We say that a set X is a *uniform possibility domain* if it has a uniform non-dictatorial aggregator.

The aforementioned sets X_1, X_2 and X_4 are uniform possibility domains, as X_1 and X_4 admit a majority aggregator, while X_2 admits a minority aggregator. Clearly, if X is a uniform possibility domain, then X is also a possibility domain. The converse, however, is not true. Indeed, suppose that X is a Cartesian product $X = Y \times Z$, where $Y \subseteq \prod_{j=1}^{l} A_j$ and $Z \subseteq \prod_{j=l+1}^{m} A_j$, with $1 \leq l < m$. If Y or Z is an impossibility domain, then X is not a uniform possibility domain, although it is a possibility domain, since, as seen earlier, every Cartesian product of two sets is a possibility domain. It is also clear that if Y and Z are uniform possibility domains, then so is their Cartesian product $Y \times Z$. In particular, the Cartesian product $X_1 \times X_2$ is a uniform possibility domain.

The next result characterizes uniform possibility domains. It is the stepping stone towards showing that the following decision problem is solvable in polynomial time: given a set X of feasible evaluations, is X a uniform possibility domain? To state this result, we first need to give a definition.

We say that $f : A^n \rightarrow A$ is a *weak near-unanimity operation* [11] if, for all $x, y \in A$, we have that

$$f(y, x, x, \ldots, x) = f(x, y, x, \ldots, x) = \ldots = f(x, x, x, \ldots, y).$$

In particular, a ternary weak near-unanimity operation is a function $f : A^3 \rightarrow A$ such that for all $x, y \in A$, we have that

$$f(y, x, x) = f(x, y, x) = f(x, x, y).$$

Thus, the notion of a ternary weak near-unanimity operation is a common generalization of the notions of a majority operation and a minority operation.

As with the majority/minority aggregators, we say that X *admits a ternary weak near-unanimity aggregator* $\bar{f} = (f_1, \ldots, f_m)$, if it admits a ternary aggregator every component of which is a weak near-unanimity operation, i.e. for all $j = 1, \ldots, m$ and for all $x, y \in X_j$, we have that $f_j(y, x, x) = f_j(x, y, x) = f_j(x, x, y)$.

Theorem C (Kirousis et al. [10]). *Let X be a set of feasible evaluations. The following two statements are equivalent.*

1. *X is a uniform possibility domain.*
2. *X admits a ternary weak near-unanimity aggregator.*

3 Results

In this section, we show that there are polynomial-time algorithms for telling, given a set X of feasible evaluations, whether or not X is a possibility domain and whether or not X is a uniform possibility domain.

3.1 Tractability of Possibility Domains

Theorems A and B provide necessary and sufficient conditions for a set X to be a possibility domain in the Boolean framework and in the non-Boolean framework, respectively. Admitting a binary non-dictatorial aggregator is a condition that appears in both of these characterizations. Our first result asserts that this condition can be checked in polynomial time.

Theorem 1. *There is a polynomial-time algorithm for solving the following problem: given a set X of feasible evaluations, determine whether or not X admits a binary non-dictatorial aggregator and, if it does, produce one.*

Proof. We will show that the existence of a binary non-dictatorial aggregator on X is tightly related to connectivity properties of a certain directed graph H_X defined next.

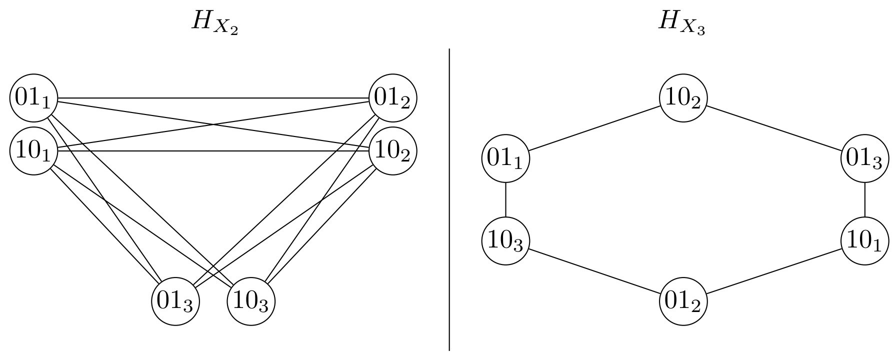
If $X \subseteq \prod_{j=1}^{m} A_j$ is a set of feasible evaluations, then H_X is the following directed graph:

- The vertices of H_X are the pairs of *distinct* elements $u, u' \in X_j$, for $j \in \{1, \dots, m\}$. Each such vertex will usually be denoted by uu'_j. When the coordinate j is understood from the context, we will often be dropping the subscript j, thus denoting such a vertex by uu'.
  Also, if $u \in X_j$, for some $j \in \{1, \dots, m\}$, we will often use the notation u_j to indicate that u is an element of X_j.
- Two vertices uu'_k and vv'_l, where $k \neq l$, are connected by a directed edge from uu'_k to vv'_l, denoted by $uu'_k \rightarrow vv'_l$, if there are a total evaluation $z \in X$ that extends the partial evaluation (u_k, v_l) and a total evaluation $z' \in X$ that extends the partial evaluation (u'_k, v'_l), such that there is no total evaluation $y \in X$ that extends (u_k, v'_l), and has the property that $y_i = z_i$ or $y_i = z'_i$, for every $i \in \{1, \dots, m\}$.

For vertices uu'_k, vv'_l, corresponding to issues k, l (that need not be distinct), we write $uu'_k \rightarrow\rightarrow vv'_l$ to denote the existence of a directed path from uu'_k to vv'_l. In the next example, we describe explicitly the graph H_X for several different sets X of feasible voting patters. Recall that a *directed* graph G is *strongly connected* if for every pair of vertices (u, v) of G, there is a (directed) path from u to v.

Example 4. Recall the sets $X_2 = \{(0,0,1), (0,1,0), (1,0,0), (1,1,1)\}$ and $X_3 = \{(0,0,1), (0,1,0), (1,0,0)\}$ of Examples 1 and 2.

Both H_{X_2} and H_{X_3} have six vertices, namely 01_j and 10_j, for $j = 1, 2, 3$. In the figures below, we use undirected edges between two vertices uu'_k and vv'_l to denote the existence of both $uu'_k \rightarrow vv'_l$ and $vv'_l \rightarrow uu'_k$.

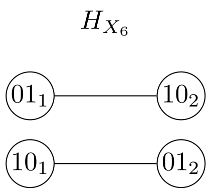


Consider 01_1, 01_2 of H_{X_2}. Since the partial vectors $(0,0)$ and $(1,1)$ extend to $(0,0,1)$ and $(1,1,1)$, respectively, we need to check if there is a vector in X_2 extending $(0,1)$, but whose third coordinate is 1. Since $(0,1,1) \notin X_2$, we have that H_{X_2} contains both edges $01_1 \rightarrow 01_2$ and $01_2 \rightarrow 01_1$. Now, since the partial vectors $(0,1)$ and $(1,0)$ extend to $(0,1,0)$ and $(1,0,0)$, respectively, and since neither $(0,0,0)$ nor $(1,1,0)$ are in X_2, we have that $01_1 \leftrightarrow 10_2$. By the above and because of the symmetric structure of X_2, it is easy to see that every two vertices uu'_i and vv'_j of H_{X_2} are connected if and only if $i \neq j$.

For X_3, observe that, since *no* partial vector containing two "1"'s, in any two positions, extends to an element of X_3, there are no edges between the vertices 01_i, 01_j and 10_i, 10_j, for any $i, j \in \{1,2,3\}$, $i \neq j$. In the same way as with H_{X_2}, we get that H_{X_3} is a cycle.

There are two observations to be made, concerning H_{X_2} and H_{X_3}. First, they are both strongly connected graphs. Also, neither X_2 nor X_3 admit binary non-dictatorial aggregators (X_2 admits only a minority aggregator and X_3 is an impossibility domain, as shown in Example 2).

Finally, consider $X_6 := \{(0,1),(1,0)\}$. The graph H_{X_6} has four vertices, 01_1, 10_1, 01_2 and 10_2, and it is easy to see that H_{X_6} has only the following edges:



Observe that X_6 is not strongly connected (it is not even connected) and that in contrast to the sets X_2 and X_3, the set X_6 admits two binary non-dictatorial aggregators, namely, $(\wedge, \vee)$ and $(\vee, \wedge)$. In Lemma 2, we establish a tight connection between strong connectedness and the existence of binary non-dictatorial aggregators.

We now state and prove two lemmas about the graph H_X.

Lemma 1. *Assume that $\bar{f} = (f_1, \ldots, f_m)$ is a binary aggregator on X.*

1. *If $uu'_k \to vv'_l$ and $f_k(u, u') = u$, then $f_l(v, v') = v$.*
2. *If $uu'_k \to\to vv'_l$ and $f_k(u, u') = u$, then $f_l(v, v') = v$.*

Proof. The first part of the lemma follows from the definitions and the fact that $\bar{f}$ is conservative. Indeed, if $uu'_k \to vv'_l$, then there are a total evaluation $z = (z_1, \ldots, z_m) \in X$ that extends (u_k, v_l) (i.e., $z_k = u$ and $z_l = v$) and a total evaluation $z' = (z'_1, \ldots, z'_m) \in X$ that extends (u'_k, v'_l) (i.e., $z'_k = u'$ and $z'_l = v'$), such that there is no total evaluation in X that extends (u_k, v'_l) and agrees with z or with z' on every coordinate. Consider the total evaluation $(f_1(z_1, z'_1), \ldots, f_m(z_m, z'_m))$, which is in X because $\bar{f}$ is an aggregator on X. Since each f_j is conservative, we must have that $f_j(z_j, z'_j) \in \{z_j, z'_j\}$, for every j, hence $f_l(z_l, z'_l) = f_l(v, v') \in \{v, v'\}$. Consequently, if $f_k(u, u') = u$, then we must have $f_l(v, v') = v$, else $(f_1(z_1, z'_1), \ldots, f_m(z_m, z'_m))$ extends (u_k, v'_l) and agrees with z or with z' on every coordinate.

The second part of the lemma follows from the first part by induction.

Lemma 2. *X admits a binary non-dictatorial aggregator if and only if the directed graph H_X is not strongly connected.*

Before delving into the proof, consider the graphs of Example 4. Using the fact that the graphs H_{X_2} and H_{X_3} are strongly connected and also using the second item of Lemma 1, it is easy to see that X_2 and X_3 admit no binary non-dictatorial aggregator; indeed, let $\bar{f} = (f_1, f_2, f_3)$ be a binary aggregator of either of these two sets and suppose that $f_1(0, 1) = 0$. Since in both graphs H_{X_2} and H_{X_3}, there are paths from 01_1 to *every* other vertex, it follows that $f_j = pr^2_1$, $j = 1, 2, 3$. If $f_1(0, 1) = 1$, we get that $f_j = pr^2_2$, $j = 1, 2, 3$, in the same way.

In contrast, consider H_{X_6} and let $\bar{g} = (g_1, g_2)$ be a pair of binary functions with $g_1(0, 1) = 0$. For $\bar{g}$ to be an aggregator, Lemma 1 forces us to set $g_2(1, 0) = 1$. But now, by setting $g_1(1, 0) = 0$, and thus $g_2(0, 1) = 1$, we get that $(g_1, g_2) = (\wedge, \vee)$ is a non-dictatorial binary aggregator for X_6.

Proof. We first show that if X admits a binary non-dictatorial aggregator, then H_X is not strongly connected. In the contrapositive form, we show that if H_X is strongly connected, then X admits no binary non-dictatorial aggregator. This is an easy consequence of the preceding Lemma 1. Indeed, assume that H_X is strongly connected and let $\bar{f} = (f_1, \ldots, f_m)$ be a binary aggregator on X. Take two distinct elements x and x' of X_1. Since $\bar{f}$ is conservative, we have that $f_1(x, x') \in \{x, x'\}$. Assume first that $f_1(x, x') = x$. We claim that $f_j = \mathrm{pr}^2_1$, for every $j \in \{1, \ldots, m\}$. To see this, let y and y' be two distinct elements of X_j, for some $j \in \{1, \ldots, m\}$. Since H_X is strongly connected, we have that $xx'_1 \to\to yy'_j$. Since also $f_1(x, x') = x$, Lemma 1 implies that $f_j(y, y') = y = \mathrm{pr}^2_1(y, y')$ and so $f_j = \mathrm{pr}^2_1$. Next, assume that $f_1(x, x') = x'$. We claim that $f_j = \mathrm{pr}^2_2$, for every $j \in \{1, \ldots, m\}$. To see this, let y and y' be two distinct elements of X_j, for some $j \in \{1, \ldots, m\}$. Since H_X is strongly connected, we have that $yy'_j \to\to xx'_1$, hence, if $f_j(y, y') = y$, then, Lemma 1, implies that $f_1(x, x') = x$, which is a contradiction because $x \neq x'$. Thus, $f_j(y, y') = y'$ and so $f_j = \mathrm{pr}_2^2$.

For the converse, assume that H_X is not strongly connected and let uu'_k, vv'_l be two vertices of H_X such that there is no path from uu'_k to vv'_l in H_X, i.e., it is not true that $uu'_k \to\to vv'_l$. Let V_1, V_2 be a partition of the vertex set such that $uu'_k \in V_1, vv'_l \in V_2$, and there is no edge from a vertex in V_1 to a vertex in V_2. We will now define a binary aggregator $\bar{f} = (f_1, \ldots, f_m)$ and prove that it is non-dictatorial.

Given $z, z' \in X$, we set $f_j(z_j, z'_j) = z_j$ if $zz'_j \in V_1$, and we set $f_j(z_j, z'_j) = z'_j$ if $zz'_j \in V_2$, for $j \in \{1, \ldots, m\}$. Since $uu'_k \in V_1$, we have that $f_k \neq \mathrm{pr}_2^2$; similarly, since $vv'_l \in V_2$, we have that $f_l \neq \mathrm{pr}_1^2$. Consequently, $\bar{f}$ is not a dictatorial function on X. Thus, what remains to be proved is that if $z, z,' \in X$, then $\bar{f}(z, z') \in X$. For this, we will show that if $\bar{f}(z, z') \notin X$, then there is an edge from an element of V_1 to an element of V_2, which is a contradiction.

Assume that $q = \bar{f}(z, z') \notin X$. Let K be a minimal subset of $\{1, \ldots, m\}$ such that $q\restriction K$ cannot be extended to a total evaluation w in X that agrees with z or with z' on $\{1, \ldots, m\} \setminus K$ (i.e., if $j \in \{1, \ldots, m\} \setminus K$, then $w_j = z_j$ or $w_j = z'_j$). Since z' is in X, it does not extend $q\restriction K$, hence there is a number $s \in K$ such that $q_s = f_s(z_s, z'_s) = z_s \neq z'_s$. It follows that the vertex zz'_s is in V_1. Similarly, since z is in X, it does not extend $q\restriction K$, hence there is a number $t \in K$ such that $q_t = f_t(z_t, z'_t) = z'_t \neq z_t$. It follows that the vertex zz'_t is in V_2. Consequently, there is no edge from zz'_s to zz'_t in H_X. We will arrive at a contradiction by showing that $zz'_s \to zz'_t$, i.e., there is an edge zz'_s to zz'_t in H_X. Consider the set $K \setminus \{t\}$. By the minimality of K, there is a total evaluation w in X that extends $q\restriction K \setminus \{t\}$ and agrees with z or with z' outside $K \setminus \{t\}$. In particular, we have that $w_s = q_s = z_s$ and $w_t = z_t$. Similarly, by considering the set $K \setminus \{s\}$, we find that there is a total evaluation w' in X that extends $q\restriction K \setminus \{s\}$ and agrees with z or with z' outside $K \setminus \{s\}$. In particular, we have that $w'_s = z'_s$ and $w_t = q_t = z'_t$. Note that w and w' agree on $K \setminus \{s, t\}$. Since $q\restriction K$ does not extend to a total evaluation that agrees with z or with z' outside K, we conclude that there is no total evaluation y in X that extends (z_s, z'_t) and agrees with w or with w' on every coordinate. Consequently, $zz'_s \to zz'_t$, thus we have arrived at a contradiction.

We are now ready to complete the proof of Theorem 1. Given a set X of feasible evaluations, the graph H_X can be constructed in time bounded by a polynomial in the size $|X|$ of X (in fact, in time $O(|X|^5)$. There are well-known polynomial-time algorithms for testing if a graph is strongly connected and, in case it is not, producing the strongly connected components of the graph (e.g., Kosaraju's algorithm presented in Sharir [15] and Tarjan's algorithm [18]). Consequently, by Lemma 2, there is a polynomial-time algorithm for determining whether or not a given set X admits a binary non-dictatorial aggregator. Moreover, if X admits such an aggregator, then one can be constructed in polynomial-time from the strongly connected components of H_X via the construction in the proof of Lemma 2.

As mentioned in the introduction, the existence of a binary non-dictatorial aggregator on X is closely related to the total blockedness of X.

Theorem D (Kirousis et al. [10]). *Let X be a set of feasible evaluations. The following two statements are equivalent.*

1. *X is totally blocked.*
2. *X admits no binary non-dictatorial aggregator.*

The next corollary follows from Theorems 1 and D.

Corollary 1. *There is a polynomial-time algorithm for the following decision problem: given a set $X \subseteq \{0,1\}^m$ of feasible evaluations, is X totally blocked?*

Furthermore, by combining Theorem 1 with Theorem A, we obtain the following result.

Theorem 2. *There is a polynomial-time algorithm for the following decision problem: given a set $X \subseteq \{0,1\}^m$ of feasible evaluations in the Boolean framework, determine whether or not it is a possibility domain.*

We now turn to the problem of detecting possibility domains in the non-Boolean framework.

Theorem 3. *There is a polynomial-time algorithm for solving the following problem: given a set X of feasible evaluations, determine whether or not X is a possibility domain and, if it is, produce a binary non-dictatorial aggregator for X or a ternary non-dictatorial aggregator for X.*

Proof. It is straightforward to check that, by Theorems B and 1, it suffices to show that there is a polynomial-time algorithm that, given X, detects whether or not X admits a majority aggregator or a minority aggregator, and, if it does, produces such an aggregator.

Let X be a set of feasible evaluations, where $I = \{1, \ldots, m\}$ is the set of issues and A_j, $j = 1, \ldots, m$, are the sets of the position values. We define the *disjoint union* A of the sets of position values as

$$A = \bigsqcup_{j=1}^m A_j = \bigcup_{j=1}^m \{(x, j) \mid x \in A_j\}.$$

We also set

$$\tilde{X} = \{((x_1, 1), \ldots, (x_m, m)) \mid (x_1, \ldots, x_m) \in X\} \subseteq A^m.$$

We will show that we can go back-and-forth between conservative majority or minority polymorphisms for $\tilde{X}$ and majority or minority aggregators for X.

Let $f : A^n \to A$ be a conservative polymorphism for $\tilde{X}$. We define the m-tuple $\bar{f} = (f_1, \ldots, f_m)$ of n-ary functions $f_1, \ldots, f_m$ as follows: if $x_j^1, \ldots, x_j^n \in A_j$, for $j \in \{1, \ldots, m\}$, then we set $f_j(x_j^1, \ldots, x_j^n) = y_j$, where y_j is such that $f((x_j^1, j), \ldots, (x_j^n, j)) = (y_j, j)$. Such a y_j exists and is one of the x_j^i's because

f is conservative, and hence $f((x_j^1, j), \ldots, (x_j^n, j)) \in \{(x_j^1, j), \ldots, (x_j^n, j)\}$. It is easy to see that $\bar{f}$ is an aggregator for X. Moreover, if f is a majority or a minority operation on $\tilde{X}$, then $\bar{f}$ is a majority or a minority aggregator on X.

Next, let $\bar{f} = (f_1, \ldots, f_m)$ be a majority or a minority aggregator for X. We define a ternary function $f : A^3 \to A$ as follows. Let $(x, j), (y, k), (z, l)$ be three elements of A.

– If $j = k = l$, then we set $f((x, j), (y, k), (z, l)) = (f_j(x, y, z), j)$.
– If j, k, l are *not* all equal, then if at least two of $(x, j), (y, k), (z, l)$ are equal to each other, we set

$$f((x, j), (y, k), (z, l)) = \mathrm{maj}((x, j), (y, k), (z, l)),$$

if $\bar{f}$ is a majority aggregator on X, and we set

$$f((x, j), (y, k), (z, l)) = \oplus((x, j), (y, k), (z, l)),$$

if $\bar{f}$ is a minority aggregator on X;
– otherwise, we set $f((x, j), (y, k), (z, l)) = (x, j)$.

It is easy to see that if $\bar{f}$ is a majority or a minority aggregator for X, then f is a conservative majority or a conservative minority polymorphism on $\tilde{X}$. It follows that X admits a majority or a minority aggregator if and only if $\tilde{X}$ is closed under a conservative majority or minority polymorphism. Bessiere et al. [2] and Carbonnel [6] design polynomial-time algorithms that detect if a given constraint language Γ has a conservative majority or a conservative minority polymorphism, respectively, and, when it has, compute such a polymorphism. Here, we apply these results to $\Gamma = \{\tilde{X}\}$.

3.2 Tractability of Uniform Possibility Domains

Recall that a constraint language is a finite set Γ of relations of finite arities over a finite non-empty set A. The *conservative constraint satisfaction problem for Γ*, denoted by c-CSP(Γ) is the constraint satisfaction problem for the constraint language $\overline{\Gamma}$ that consists of the relations in Γ and, in addition, all unary relations on A. Bulatov [3,4] established a dichotomy theorem for the computational complexity of c-CSP(Γ): if for every two-element subset B of A, there is a conservative polymorphism f of Γ such that f is binary and $f \restriction B \in \{\wedge, \vee\}$ or f is ternary and $f \in \{\mathrm{maj}, \oplus\}$, then c-CSP($\Gamma$) is solvable in polynomial time; otherwise, c-CSP(Γ) is NP-complete. Carbonnel [5] showed that the boundary of the dichotomy for c-CSP(Γ) can be checked in polynomial time.

Theorem E (Carbonnel [5]). *There is a polynomial-time algorithm for solving the following problem: given a constraint language Γ on a set A, determine whether or not for every two-element subset $B \subseteq A$, there is a conservative polymorphism f of Γ such that f is binary and $f \restriction B \in \{\wedge, \vee\}$ or f is ternary and $f \restriction B \in \{\mathrm{maj}, \oplus\}$. Moreover, if such a polymorphism exists, then the algorithm produces one in polynomial time.*

The final result of this section is about the complexity of detecting uniform possibility domains.

Theorem 4. *There is a polynomial-time algorithm for solving the following problem: given a set X of feasible evaluations, determine whether or not X is a uniform possibility domain and, if it is, produce a ternary weak near-unanimity aggregator for X.*

Proof. In what follows, given a two-element set B, we will arbitrarily identify its elements with 0 and 1. Consider the functions $\wedge^3$ and $\vee^3$ on $\{0,1\}^3$, where $\wedge^3(x,y,z) := (\wedge(\wedge(x,y),z))$ and $\vee^3(x,y,z) := (\vee(\vee(x,y),z))$. It is easy to see that the only ternary, conservative, weak near-unanimity functions on $\{0,1\}$ are $\wedge^3$, $\vee^3$, maj, and $\oplus$. We will also make use of the following lemma, which also gives an alternative formulation of the boundary of the dichotomy for conservative constraint satisfaction.

Lemma 3. *Let Γ be a constraint language on set A. The following two statements are equivalent.*

1. *For every two-element subset $B \subseteq A$, there exists a conservative polymorphism f of Γ (which, in general, depends on B), such that f is binary and $f\restriction_B \in \{\wedge, \vee\}$ or f is ternary and $f\restriction_B \in \{\mathrm{maj}, \oplus\}$.*
2. *Γ has a ternary, conservative, weak near-unanimity polymorphism.*

Proof. (*Sketch*) ($1 \Rightarrow 2$) Given a two-element subset $B \subseteq A$ and a binary conservative polymorphism f of Γ such that $f\restriction_B \in \{\wedge, \vee\}$, define f' to be the ternary operation such that $f'(x,y,z) = f(f(x,y),z)$, for all $x,y,z \in A$. It is easy to see that f' is a conservative polymorphism of Γ as well and also that $f'\restriction_B \in \{\wedge^{(3)}, \vee^{(3)}\}$.

The hypothesis and the preceding argument imply that, for each two-element subset $B \subseteq A$, there exists a ternary conservative polymorphism f of Γ (which, in general, depends on B) such that $f\restriction_B \in \{\wedge^{(3)}, \vee^{(3)}, \mathrm{maj}, \oplus\}$. For each two-element subset $B \subseteq A$, select such a polymorphism and let $f^1, \ldots, f^N$, $N \geq 1$, be an enumeration of all these polymorphisms. Clearly, the restriction of each f^i to its respective two element subset is a weak near-unanimity operation.

Consider the '$\diamond$' operator that takes as input two ternary operations $f, g : A^3 \to A$ and returns as output a ternary operation $f \diamond g$ defined by

$$(f \diamond g)(x,y,z) := f(g(x,y,z), g(y,z,x), g(z,x,y)).$$

If f, g are conservative polymorphisms of Γ, then so is $(f \diamond g)$. Also, if B is a two-element subset of A such that $f\restriction_B$ or $g\restriction_B$ is a weak near-unanimity operation, then so is $(f \diamond g)\restriction_B$. Consider now the iterated diamond operation h with

$$h := f^1 \diamond (f^2 \diamond (\ldots \diamond (f^{N-1} \diamond f^N) \ldots)).$$

By the preceding discussion, h is a conservative polymorphism such that $h\restriction_B$ is a weak near-unanimous operation for *every* two-element subset B of A, hence h itself is a weak near-unanimity, conservative, ternary operation of Γ.

($2 \Rightarrow 1$) Let h be a ternary, conservative, weak near-unanimity polymorphism of Γ. Thus, for every two-element subset $B \subseteq A$, we have that $h{\restriction}_B \in \{\wedge^{(3)}, \vee^{(3)}, \mathrm{maj}, \oplus\}$.

If there is a two-element subset $B \subseteq A$ such that $h{\restriction}_B \in \{\wedge^{(3)}, \vee^{(3)}\}$, then consider the binary function g defined by

$$g(x, y) := h(x, x, y) = h(pr_1^2(x, y), pr_1^2(x, y), pr_2^2(x, y)).$$

Obviously, g is a binary conservative polymorphism of Γ; moreover, for every two-element subset $B \subseteq A$, if $h{\restriction}_B \in \{\wedge^{(3)}, \vee^{(3)}\}$, then $g{\restriction}_B \in \{\wedge, \vee\}$.

By Theorem C, a set X of feasible evaluations is a uniform possibility domain if and only if there is a ternary aggregator $\bar{f} = (f_1, \ldots, f_m)$ such that each f_j is a weak near-unanimity operation, i.e., for all $j \in \{1, \ldots, m\}$ and for all $x, y \in X_j$, we have that $f_j(x, y, y) = f_j(y, x, y) = f_j(y, y, x)$. As in the proof of Theorem 3, we can go back-and-forth between X and the set $\tilde{X}$, and verify that X is a uniform possibility domain if and only if $\tilde{X}$ has a ternary, conservative, weak near-unanimity polymorphism. Theorem E and Lemma 3 then imply that the existence of such a polymorphism can be tested in polynomial time, and that such a polymorphism can be produced in polynomial time, if one exists.

4 Concluding Remarks

In this paper, we established the first results concerning the tractability of non-dictatorial aggregation. Specifically, we gave polynomial-time algorithms that take as input a set X of feasible evaluations and determine whether or not X is a possibility domain and a uniform possibility domain, respectively. In these algorithms, the set X of feasible evaluations is given to us explicitly, i.e., X is given by listing all its elements. It is natural to ask how the complexity of these problems may change if X is given implicitly via a succinct representation. Notice that Theorems 1, 3 and 4, directly imply that in this case, the problems of whether X is a possibility domain and of whether it is a uniform possibility domain respectively, are EXPTIME. Furthermore, by Theorem 2, we obtain that in the Boolean framework, the problem of whether X is a possibility domain is in PSPACE. It is an open problem weather there are respective lower bounds for the complexity of the above problems in the succinct case. This scenario merits further investigation, because it occurs frequently in other areas of social choice, including the area of judgment aggregation, where X is identified with the set of satisfying assignments of a Boolean formula (for surveys on judgment aggregation, see [9,12]).

The work reported here assumes that the aggregators are conservative, an assumption that has been used heavily throughout the paper. There is a related, but weaker, notion of an *idempotent* (or *Paretian*) aggregator $\bar{f} = (f_1, \ldots, f_m)$ where each f_j is assumed to be an idempotent function, i.e., for all $x \in X_j$, we have that $f(x, \ldots, x) = x$. Clearly, every conservative aggregator is idempotent. In the Boolean framework, idempotent aggregators are conservative, but, in the non-Boolean framework, this need not hold. It remains an open problem to investigate the computational complexity of the existence of non-dictatorial idempotent aggregators in the non-Boolean framework.

Acknowledgements. The research of Lefteris Kirousis was partially supported by the Special Account for Research Grants of the National and Kapodistrian University of Athens. The work of Phokion G. Kolaitis is partially supported by NSF Grant IIS-1814152.

References

1. Arrow, K.J.: Social Choice and Individual Values. Wiley, New York (1951)
2. Bessiere, C., Carbonnel, C., Hebrard, E., Katsirelos, G., Walsh, T.: Detecting and exploiting subproblem tractability. In: IJCAI, pp. 468–474 (2013)
3. Bulatov, A.A.: A dichotomy theorem for constraint satisfaction problems on a 3-element set. J. ACM (JACM) **53**(1), 66–120 (2006)
4. Bulatov, A.A.: Complexity of conservative constraint satisfaction problems. ACM Trans. Comput. Logic (TOCL) **12**(4), 24 (2011)
5. Carbonnel, C.: The dichotomy for conservative constraint satisfaction is polynomially decidable. In: Rueher, M. (ed.) CP 2016. LNCS, vol. 9892, pp. 130–146. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44953-1_9
6. Carbonnel, C.: The meta-problem for conservative Mal'tsev constraints. In: Thirtieth AAAI Conference on Artificial Intelligence (AAAI-2016) (2016)
7. Dokow, E., Holzman, R.: Aggregation of binary evaluations. J. Econ. Theory **145**(2), 495–511 (2010)
8. Dokow, E., Holzman, R.: Aggregation of non-binary evaluations. Adv. Appl. Math. **45**(4), 487–504 (2010)
9. Endriss, U.: Judgment aggregation. In: Brandt, F., Conitzer, V., Endriss, U., Lang, J., Procaccia, A.D. (eds.), Handbook of Computational Social Choice, pp. 399–426. Cambridge University Press (2016)
10. Kirousis, L., Kolaitis, P.G., Livieratos, J.: Aggregation of votes with multiple positions on each issue. In: Höfner, P., Pous, D., Struth, G. (eds.) RAMICS 2017. LNCS, vol. 10226, pp. 209–225. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57418-9_13
11. Larose, B.: Algebra and the complexity of digraph CSPs: a survey. In: Dagstuhl Follow-Ups, vol. 7. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017)
12. List, C., Puppe, C.: Judgment Aggregation: A Survey (2009)
13. Nehring, K., Puppe, C.: Strategy-proof social choice on single-peaked domains: possibility, impossibility and the space between. University of California at Davis (2002). http://vwl1.ets.kit.edu/puppe.php
14. Schaefer, T.J.: The complexity of satisfiability problems. In: Proceedings of the 10th Annual ACM Symposium on Theory of Computing, pp. 216–226 (1978)
15. Sharir, M.: A strong-connectivity algorithm and its applications in data flow analysis. Comput. Math. Appl. **7**(1), 67–72 (1981)
16. Szegedy, M., Xu, Y.: Impossibility theorems and the universal algebraic toolkit. CoRR, abs/1506.01315 (2015)
17. Szendrei, Á.: Clones in Universal Algebra, vol. 99. Presses de l'Université de Montréal, Montreal (1986)
18. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972)