Fragment of the necessary decisions explained by an argument scheme in a non compensatory sorting model

Khaled Belahcene[1] and **Jérome Gaigne**[2] and **Sylvain Lagrue**[3]

Abstract. We focus on an argument scheme explaining a necessary decision with respect to a jurisprudence considering the non compensatory sorting model. In this model, a set of candidates is sorted in two ordered categories by a set of points of view expressing an individual preference over the set of candidates. Candidates are ranked in the upper category if they are accepted by a sufficient subset of points of view. We show that the argument scheme is an approximation of this model and share some counterintuitive results concerning the relative performance of some well-known solvers.

1 Introduction

The accountable approval sorting problem has been introduced in [3]. It corresponds to the situation where a jury, composed of multiple members, has to assign a set of candidates into two ordered categories. The authors used GOOD and BAD, it could also be ACCEPTED and REFUSED if the context is a review of job applicants. Each member of the jury has an opinion over all the candidates as a preference relation and approves a subset of them. Those approval sets are aggregated with respect to sufficient coalitions in order to assign each candidate to the right category. Therefore a candidate is assigned to the upper category (*i.e.* ACCEPTED) if this candidate is approved by a sufficient subset of the jury.

The authors named two situations of interest for accountability in an approval sorting setting :

- being able to justify that all the decisions taken by the jury follow the approval sorting procedure;
- being able to justify a candidate's category.

Here, we focus on the second case where a candidate wishes a justification to his/her ranking. Again there are two situations of interest :

- the candidate could have been ranked in the other category;
- the candidate cannot be ranked in the other category.

In order to be accountable in this last situation, it has to be shown that the decision taken for this candidate was necessary, namely if the candidate is ranked in the other category, the decision would not respect the approval sorting procedure. The authors proposed an argument scheme, whose principle is presented by the authors of [8], to get a justification of the necessity of the decision which is humanly readable. This scheme has been proven to be a sufficient condition to determine the necessity of a decision, but, to our knowledge it has not been proven to be a necessary condition. Furthermore, to diminish the amount of private information about the juries' decision, a *jurisprudence* can be used as a reference case to build the argument scheme.

In Section 2, we introduce or recall important definitions about the non compensatory sorting model underlying the approval sorting, and about its inverse problem. Then, Section 3 introduces the argument scheme justifying the necessity of a decision that has been proposed by the authors of [3]. In this section, we study the necessity of the scheme and we prove it only covers a fragment of all the necessary decisions, *i.e.* the scheme is not necessary when there is a necessary decision. For obtaining these results, we used multiple solvers and models to compute the scheme, so we provide a study of their relative performances in Section 4. Lastly, Section 5 recalls our results and provides new questions.

2 Non compensatory sorting model and inverse NCS problem

In this section, we introduce the non compensatory sorting model with two categories together with some of its properties. We recall its inverse problem along with an important representation theorem brought by the authors of [3].

2.1 Non compensatory sorting model

The approval sorting model is also called non compensatory sorting model, *NCS* for short.

In order to ease the definition of the non compensatory sorting model, we recall the definition of an upset.

Definition 1 (upset)**.** *Let A be a set and R be a binary relation over A. The subset $B \subseteq A$ is an upset for (A, R) iff $\forall a \in A, \forall b \in B$, if bRa, then $a \in B$.*

Definition 2 (Non compensatory sorting model, [4])**.** *Let $\mathbb{X}$ be a set of alternatives and $\mathcal{N}$ be a set of points of view. Let ACCEPTED and REFUSED be two ordered categories. Each point of view $i \in \mathcal{N}$ has a preference over the whole set of alternatives as a complete preorder written $\succsim_i$. To stay syntactically and semantically coherent, we may use the preorder written $\precsim_i$ defined as $\forall a, b \in \mathbb{X}, a \precsim_i b \quad iff \quad b \succsim_i a$. Each point of view $i \in \mathcal{N}$ approves a subset of alternatives written $\mathcal{A}_i$ such as $\mathcal{A}_i$ is an upset for $(\mathbb{X}, \precsim_i)$. Lastly, let S be the set of all the sufficient coalitions of points of view such as S is an upset for $\left(2^{\mathcal{N}}, \subseteq\right)$. A sorting process respects the NCS model if and only if each ACCEPTED alternative has been approved by a sufficient coalition*

[1] Heudiasyc, Université de Technologie de Compiègne et CNRS, France, email: khaled.belahcene@hds.utc.fr
[2] Heudiasyc, Université de Technologie de Compiègne et CNRS, France, email: jerome.gaigne@hds.utc.fr
[3] Heudiasyc, Université de Technologie de Compiègne et CNRS, France, email: sylvain.lagrue@hds.utc.fr

of points of view, i.e. *let* $\alpha : \mathbb{X} \mapsto \{ACCEPTED, REFUSED\}$ *be the function characterizing the decision process,* α *corresponds to a non compensatory sorting if and only if*

$$\forall x \in \mathbb{X}, \alpha(x) = \begin{cases} ACCEPTED, \text{ if } \{i \in \mathcal{N} : x \in \mathcal{A}_i\} \in S \\ REFUSED, \text{ otherwise} \end{cases}$$

In short, a sorting process respecting the NCS model takes as inputs a set of alternatives $\mathbb{X}$, a set of points of view $\mathcal{N}$, the preferences of each point of view over the alternatives $\succsim_i, \forall i \in \mathcal{N}$, the approval set of each point of view $\mathcal{A}_i, \forall i \in \mathcal{N}$ and the set of all sufficient coalitions of points of view S. Its output is the sorting of the alternatives into the two categories.

From this definition, we can pull two rationality properties that this process satisfies. Firstly, the individual rationality which states that there is no alternative disapproved by $i \in \mathcal{N}$ that is preferred to an alternative accepted by i, *i.e.* $\mathcal{A}_i$ is an upset for $(\mathbb{X}, \succsim_i)$. Secondly, the collective rationality states that if a subset of points of view is a sufficient coalition, then any super set of it is also sufficient, *i.e.* S is an upset for $\left(2^{\mathcal{N}}, \subseteq\right)$.

Example 1. *Let* $\mathbb{X} = \{a, b, c, d, e\}$ *be a set of alternatives,* $\mathcal{N} = \{1, 2, 3\}$ *be a set of points of view. Individual preferences are as follows:*

$$e \succ_1 a \succ_1 c \succ_1 b \succ_1 d$$
$$b \succ_2 a \succ_2 c \succ_2 d \succ_2 e$$
$$c \succ_3 b \succ_3 a \succ_3 d \succ_3 e$$

Here are the approval sets of each jury. They approve their two most preferred ones:

$$\mathcal{A}_1 = \{a, e\}$$
$$\mathcal{A}_2 = \{a, b\}$$
$$\mathcal{A}_3 = \{b, c\}$$

Lastly, the sufficient coalitions are as follows:

$$S = \{\{1, 2\}, \{2, 3\}, \{1, 3\}, \{1, 2, 3\}\}$$

This setting results in the assignment: $\alpha = \{(a, ACCEPTED), (b, ACCEPTED), (c, REFUSED), (d, REFUSED), (e, REFUSED)\}$.

2.2 Inverse NCS problem

If the process is to be inquired, it is necessary to be able from the output to find input parameters that could have led to obtaining α through a non compensatory sorting process. To do so, we need the output of the sorting process, α, and some other data, the individual preferences of the points of view. With those, we verify that it is possible to find a set of approval sets and a set of sufficient coalitions that allow obtaining α through a non compensatory sorting process. In the inverse NCS problem (also written *inv-NCS*), we only focus on finding the existence or the nonexistence of those parameters (*i.e.* it is unnecessary to reveal their actual values).

The representation theorem is presented in [3]. This theorem is important because it allows one to answer inv-NCS without having to deal with the notion of sufficient coalition. By this theorem, the sufficient coalitions do not have to be revealed.

Theorem 1 (NCS pairwise formulation, [3])**.** *A function* α *mapping the alternatives to the categories can be represented in NCS if and only if there exists* $\langle \mathcal{A}_i \rangle_{i \in \mathcal{N}} \subseteq (2^{\mathbb{X}})^{\mathcal{N}}$ *such as:*

1. $\forall i \in \mathcal{N}, \mathcal{A}_i$ *is an* upset *for* $(\mathbb{X}, \succsim_i)$.
2. $\forall \langle g, b \rangle \in \alpha^{-1}(ACCEPTED) \times \alpha^{-1}(REFUSED), \exists i \in \mathcal{N}$ *such as* $g \in \mathcal{A}_i$ *and* $b \notin \mathcal{A}_i$.

Theorem 1 allows one to reduce inv-NCS to SAT ([3], Section 3.4). In [7], this formulation has been extended in case where one has more than 2 categories, and relaxed in a MaxSAT formulation to be able to take into account noisy data.

3 Argument scheme's study

The authors of [3] proposed an argument scheme allowing one to build a humanly readable justification of the necessity of a decision about the sorting of an alternative. In this section, we recall this scheme. Then we answer the question about the necessity of this scheme to assess the necessity of a decision. We found out various counter examples showing that the scheme isn't a necessary condition.

3.1 Argument scheme justifying a necessary decision with respect to a jurisprudence

First of all, we recall the definition of a necessary decision. Let us say that we have a set of reference cases (a jurisprudence) $\alpha^\star : \mathbb{X}^\star \mapsto \{\text{ACCEPTED}, \text{REFUSED}\}$ with $\mathbb{X}^\star \subseteq \mathbb{X}$. To show that a decision about an alternative sorting is necessary, we can use the fact the inv-NCS problem tells us if a decision process is a possible outcome of a NCS process. If we prove that the other decision concerning the alternative would not be possible (*i.e.* this new instance is then negative of the inv-NCS) it results that the original decision was necessary.

Definition 3 (Necessary decision with respect to a jurisprudence, [3])**.** *Let* $\alpha^\star$ *be a positive instance of the inv-NCS problem, we say that* $x \in \mathbb{X}$ *is necessarily assigned to* $C \in \{ACCEPTED, REFUSED\}$ *with respect to* $\alpha^\star$ *if* $\alpha^\star \cup \left\{\left(x, \bar{C}\right)\right\}$ *is a negative instance of the inv-NCS problem, with* $\bar{C}$ *the category opposed to* C.

The argument scheme below is a sufficient condition to show the necessity of a decision. An open question is to know if it is also a necessary condition to it.

Definition 4 (Necessary decision argument scheme with respect to a jurisprudence, [3])**.** *We say that a set* $\{\langle g_1, b_1 \rangle, ..., \langle g_k, b_k \rangle\}$ *instantiates the argument scheme rejecting the function* α *with respect to* $\alpha^\star$*, with* $\forall j = 1, ..., \left|\alpha^{-1}(ACCEPTED)\right|, \alpha(g_j) = ACCEPTED$ *and* $\forall j = 1, ..., \left|\alpha^{-1}(REFUSED)\right|, \alpha(b_j) = REFUSED$ *and* $\forall x \in \mathbb{X}^\star, \alpha(x) = \alpha^\star(x)$*, if :*

$$\exists B \subseteq \mathcal{N}, |B| \leq \min(|\mathcal{N}|, k - 1),$$
$$\forall i \notin B, \forall u \in \{1, ..., k\}, g_u \succsim_i b_u \text{ does not hold} \quad (1)$$

and

$$\forall j \in B, \forall p, q \in \{1, ..., k\}, \text{if } p \neq q \text{ then}$$
$$\exists \langle g, b \rangle \in \{g_p, g_q\} \times \{b_p, b_q\}, g \succsim_j b \text{ does not hold} \quad (2)$$

This definition relies on a subset B of points of view of cardinality lesser than the number of pairs of alternatives that we want to split (*cf.* theorem 1 about the pairwise model). Condition 1 states that none of the points of view outside of B is able to split any pair. Condition 2 says that only one pair at most at a time can be separated by each point of view inside B.

Proof. (Sufficient condition). The fact that this scheme is a sufficient condition to assess that an instance is negative to the inv-NCS problem is straightforward since k, the number of pairs to split, is strictly greater that the cardinality of B that is at most $k-1$. Knowing that, it says that at most $k-1$ pairs can be separated over the k ones thanks to condition 2 and 0 pairs can be separated by condition 1, so one remains non-split. Thus, by theorem 1, proving that the instance is indeed negative for inv-NCS. □

This scheme allows one to keep the individual approval sets private while justifying the decision in a humanly readable way.

Example 2. *Let* $\mathbb{X} = \{a, b, c, d, e\}$, $\mathcal{N} = \{1, 2\}$ *and* $\alpha = \{(a, \textit{ACCEPTED}), (b, \textit{REFUSED}), (c, \textit{REFUSED}), (d, \textit{REFUSED}), (e, \textit{REFUSED})\}$.

$$a \succ_1 c \succ_1 d \succ_1 b \succ_1 e$$
$$e \succ_2 b \succ_2 d \succ_2 a \succ_2 c$$

This configuration is positive for inv-NCS. If we choose $\mathcal{A}_1 = \{a\}$ *and* $S = \{\{1\}, \{1, 2\}\}$*, we produce* α.

Now, let us say that we want a justification of b's sorting. If we switch b to the other category, i.e. $\alpha' = \alpha \setminus \{(b, \textit{REFUSED})\} \cup \{(b, \textit{ACCEPTED})\}$*, the new function* α' *is a negative instance of inv-NCS. Searching for an argument scheme in this configuration, we can for instance find the set* $B = \{1\}$ *with the pairs* $\{\langle a, d\rangle, \langle b, e\rangle\}$*. In other words, the point of view* 1 *can only split one pair at a time, while the point of view* 2 *cannot split any of them since for each interesting pair, the REFUSED alternative is preferred to the ACCEPTED one according to* $\succ_2$.

This scheme is not unique, we can also take $B = \{1, 2\}$ *with* $\{\langle a, d\rangle, \langle b, d\rangle, \langle b, e\rangle\}$ *for instance.*

3.2 Argument scheme as a necessary condition for necessary decision

To verify that the scheme is a necessary condition for identifying inv-NCS negative instances, we decided to try to compute a counter-example. To do this, we have to look for an instance satisfying the two following conditions:

1. the instance is negative for the inv-NCS problem; and
2. there's no scheme to be found in it.

We check the first condition using the SAT formulation of the inv-NCS problem proposed in [3]. We then check the second condition using a novel SAT formulation Φ_{scheme} described in the next section.

3.2.1 Argument scheme's SAT formulation

To ease the search, we restrict the search space to preference profiles in the form of strict total orders. In this case, the argument scheme can be written as follows:

Definition 5 (Necessary decision argument scheme with respect to a jurisprudence composed of linear orders)**.** *We say that the set* $\{\langle g_1, b_1\rangle, ..., \langle g_k, b_k\rangle\}$ *instantiates the argument scheme to reject the function* α *with respect to* $\alpha^\star$*, with* $\forall j = 1, ..., \left|\alpha^{-1}(\textit{ACCEPTED})\right|, \alpha(g_j) = \textit{ACCEPTED}$ *and* $\forall j = 1, ..., \left|\alpha^{-1}(\textit{REFUSED})\right|, \alpha(b_j) = \textit{REFUSED}$ *and* $\forall x \in \mathbb{X}^\star, \alpha(x) = \alpha^\star(x)$*, if:*

$$\exists B \subseteq \mathcal{N}, |B| \leq \min(|\mathcal{N}|, k-1),$$

$$\forall i \notin B, \forall u \in \{1, ..., k\},\ b_u \succ_i g_u \tag{3}$$

and

$$\forall j \in B, \forall p, q \in \{1, ..., k\}, \textit{if } p \neq q \textit{ then}$$
$$\exists \langle g, b\rangle \in \{g_p, g_q\} \times \{b_p, b_q\}, b \succ_j g \tag{4}$$

In this section, we denote: m $:= \left|\alpha^{-1}(\text{ACCEPTED})\right|$, $n := \left|\alpha^{-1}(\text{REFUSED})\right|$ and $\mathbb{A} := \alpha^{-1}(\text{ACCEPTED}) \times \alpha^{-1}(\text{REFUSED})$.

We use the following propositional variables:

$$\begin{array}{rcl} \beta_i & := & i \in B \\ \lambda_{gb} & := & \langle g, b\rangle \in \{\langle g_1, b_1\rangle, ..., \langle g_k, b_k\rangle\} \end{array}$$

We seek to express the following cardinality constraint:

$$|\{i \in \mathcal{N} : \beta_i \text{ is true }\}| < |\{\langle g, b\rangle \in \mathbb{A} : \lambda_{gb} \text{ is true }\}| \tag{5}$$

To write cardinality constraints in SAT, we use Bailleux and Boufkhad's encoding introduced in [2] modified so as to allow us to directly compare the cardinalities of two sets. We keep the *totalizer* as is (which computes the cardinality of each set) but we modify the *comparator* (which, in its original version, sets a lower and an upper bound to the set cardinality). The idea is to create a *comparator* between the two cardinalities. To do so, we add the following rule between the variables of both *totalizers* (which are a unary representation of each set cardinality):

$$\exists i \in \{1, ..., n\}, S_i^\lambda \wedge \neg S_i^\beta$$

This rule simply says that, there is at least one less true variable in the unary representation of the cardinality set $\{(g, b) \in \mathbb{A} : \lambda_{gb} \text{ is true }\}$ than in the one of the set $\{i \in \mathcal{N} : \beta_i \text{ is true }\}$. In other words, the cardinality of the set $\{\langle g, b\rangle \in \mathbb{A} : \lambda_{gb} \text{ is true }\}$ is at least equal to $|\{i \in \mathcal{N} : \beta_i \text{ is true }\}| + 1$.

We denote by $n_\lambda := \left|\alpha^{-1}(\text{ACCEPTED})\right| \times \left|\alpha^{-1}(\text{REFUSED})\right|$ the upper bound of the cardinality of the set $\{\langle g, b\rangle \in \mathbb{A} : \lambda_{gb} \text{ is true }\}$, and by $n_\beta := |\mathcal{N}|$ the upper bound of the cardinality of the set $\{i \in \mathcal{N} : \beta_i \text{ is true }\}$.

In order to represent this rule, we can create $l := \max(n_\beta, n_\lambda)$ new variables noted $W_i, i \in \{1, ..., l\}$ which represent the previous rule as it follows :

$$S_i^\lambda \wedge \neg S_i^\beta \leftrightarrow W_i$$
$$\bigvee_{i=1...l} W_i$$

The unary representation with the least number of variables has to be completed to get to the same amount as the largest one, *i.e.* l. All these new variables have to be set to false. In other words, $S^\star$ with $\star = \text{argmin}_{x \in \{\lambda, \beta\}}(n_x)$:

$$\text{Ext} : \bigwedge_{i=n_\star+1...l} \neg S_i^\star$$

Thus the cardinality representation is as follows :

- *Totalizers*: Totalizer $(\lambda_{gb}) \wedge$ Totalizer (β_i)
- Cardinalities' comparator :

$$\begin{array}{rl} \text{Comp}_1 : & \bigwedge_{i=1...l} \neg S_i^\lambda \vee S_i^\beta \vee W_i \\ \text{Comp}_2 : & \bigwedge_{i=1...l} S_i^\lambda \vee \neg W_i \\ \text{Comp}_3 : & \bigwedge_{i=1...l} \neg S_i^\beta \vee \neg W_i \\ \text{Comp}_4 : & \bigvee_{i=1...l} W_i \end{array}$$

The other constraints of the scheme are almost straightforwardly translated to SAT as follow:

$$\mathrm{C}_1: \quad \bigwedge_{\substack{\forall i \in \mathcal{N}, \\ \forall \langle g,b \rangle \in \mathbb{A}, \\ g \succ_i b}} \beta_i \vee \neg \lambda_{gb}$$

$$\mathrm{C}_2: \quad \bigwedge_{\substack{\forall i \in \mathcal{N}, \\ \forall p \neq q = 1,\dots,mn, \\ \forall \langle g,b \rangle \in \{b_p, b_q\} \times \{g_p, g_q\}, \\ g \succ_i b}} \neg \beta_i \vee \neg \lambda_{g_p b_p} \vee \neg \lambda_{g_q b_q}$$

This cardinality comparison encoding is useful so that we can call the solver only once to solve the problem and not once per each cardinality value. The total formula is then:

$$\phi_{scheme}: \quad \text{Ext} \wedge \text{Totalizers} \wedge \bigwedge_{i=1}^{4} \text{Comp}_i \wedge \mathrm{C}_1 \wedge \mathrm{C}_2$$

Theorem 2. *Let a preference profile be composed of linear orders and α a negative instance of inv-NCS. There is an argument scheme explaining why α is rejected iff ϕ_{scheme} is satisfiable.*

3.2.2 Counterexample search

Using this SAT representation and the inv-NCS SAT representation, we found out that there are negative inv-NCS instances which do not have argument schemes. To do so, we exhaustively search all instances for various parameters' settings: number of alternatives, number of points of view and number of ACCEPTED (thus REFUSED) alternatives.

To reduce the search space and avoid a rather good number of symmetries, we used the following procedure to generate all instances:

- Set the following parameters : number of alternatives ($n_{\text{alternatives}}$), number of points of view (n_{POVs}), number of ACCEPTED alternatives (n_{ACCEPTED} and thus $n_{\text{REFUSED}} = n_{\text{alternatives}} - n_{\text{ACCEPTED}}$).
- Generate a preference profile:
  - For the first point of view, compute a permutation of the multiset containing two types of items, g, the ACCEPTED alternatives, and b, the REFUSED alternatives. g has a multiplicity of n_{ACCEPTED} and b has a multiplicity of n_{REFUSED}. This point of view sets the name of each alternative, *i.e.* for the ACCEPTED alternatives: $g_1, ..., g_{n_{\text{ACCEPTED}}}$ by adding an index to the g items in the obtained permutation starting from the most preferred to the least preferred and for the REFUSED alternatives: $b_1, ..., b_{n_{\text{REFUSED}}}$ in the same manner.
  - For every other points of view, compute a permutation over the alternatives.
- The function α is easily generated by taking α^{-1} (ACCEPTED) $= \{g_1, ..., g_{n_{\text{ACCEPTED}}}\}$ and α^{-1} (REFUSED) $= \{b_1, ..., b_{n_{\text{REFUSED}}}\}$.
- Compute a hash for each point of view.
- Sort this hash in the increasing order and store this sorted tuple in a table.
- If this tuple is already in the table then generate a new profile.
- Otherwise, solve the inv-NCS problem using the SAT formulation from Section 3.4 of [3].
- If it is unsatisfiable, then use ϕ_{scheme} presented in Section 3.2.1 to search for any scheme.
- If there is none, then store this instance as a counterexample.

In both inv-NCS and the argument scheme, we only care about the (ACCEPTED, REFUSED) pairs of alternatives and the positions of the alternatives in each preference. Thus we can avoid the symmetries consisting of swapping g_i and g_j by generating the first point of view's preference using the permutation of the multiset containing n_{ACCEPTED} times a g item and n_{REFUSED} times a b item. Using this point of view to set the name of the alternatives allows us to remove this type of symmetry.

Furthermore, to avoid the symmetries coming from the permutations of the points of view, we store a sorted tuple of hash (one hash per point of view computed with the point of view's preference) in a table that is checked every time a new profile is created.

There is at least one other type of symmetry that we ignore. For instance, the one consisting in swapping the categories of all alternatives. Despite this, we were able to exhaustively search the space for several parameters values. The results can be found in Table 1 and Table 2.

# alternatives	# points of view	# counterexamples
3	2	0
4	2	0
4	3	0
5	2	0
5	3	0
5	4	0
6	2	0
6	3	898

Table 1. Number of counterexamples with respect to the number of alternatives and the number of points of view

The number of counterexamples for 6 alternatives and 3 points of view is very small considering the number of tested configurations. Without the hash table, we have $\sum_{k=1}^{5} \left(C_6^k\right) \times 6! \times 6! = 32\,140\,800$ configurations, which can be shrunk to $15\,785\,210$ by using the hash table. In this, $8\,387\,380$ are negative for the inv-NCS problem. Thus leading us to a frequency of $\frac{898}{15\,785\,210} \simeq 5.69 \times 10^{-5}$ over all configurations (positive and negative ones) and a ratio of negative instances not covered by the scheme of $\frac{898}{8\,387\,380} \simeq 1.07 \times 10^{-4}$.

# ACCEPTED alternatives	# counterexamples
1	0
2	12
3	874
4	12
5	0

Table 2. Number of counterexamples with respect to the number of ACCEPTED alternatives when there are 6 alternatives and 3 points of view

The symmetrical aspect of Table 2 can be explained by the remaining symmetry between the role of ACCEPTED and REFUSED alternatives.

We found out that there are counterexamples, however, it is important to note that the ratio of counterexamples over the total number of instances or even the number of negative instances is really small knowing there are 6 alternatives and 3 points of view and all preferences are strict total orders.

Here is one of the counterexamples found by this process with 6 alternatives and 3 points of view:

Example 3. *This example is a continuation of Example 2 adding one point of view and one alternative. Let us take* $\mathbb{X} = \{a, b, c, d, e, x\}$, $\mathcal{N} = \{1, 2, 3\}$ *and* $\alpha = \{(a, \textit{ACCEPTED}), (b, \textit{ACCEPTED}), (x, \textit{ACCEPTED}),$

$(c, \textit{REFUSED})\,,\ (d, \textit{REFUSED})\,,\ (e, \textit{REFUSED})\}$

$$x \succ_1 c \succ_1 a \succ_1 b \succ_1 d \succ_1 e$$
$$a \succ_2 c \succ_2 x \succ_2 d \succ_2 b \succ_2 e$$
$$e \succ_3 b \succ_3 d \succ_3 a \succ_3 c \succ_3 x$$

*This configuration does not satisfy the inv-NCS problem neither does it contain an argument scheme (*i.e. *there is no way of selecting pairs of alternatives with one ACCEPTED and one REFUSED, so that we could create a subset B of points of view respecting the conditions of the argument scheme). However, if we consider $\alpha' = \{(a, \textit{ACCEPTED})\,,\ (b, \textit{ACCEPTED})\,,\ (x, \textit{REFUSED})\,,\ (c, \textit{REFUSED})\,,\ (d, \textit{REFUSED})\,,\ (e, \textit{REFUSED})\}$ where x is now in the REFUSED category, then it is a positive instance for the inv-NCS problem because:*

$$\mathcal{A}_1 = \{a, b, c, x\}$$
$$\mathcal{A}_2 = \{a\}$$
$$\mathcal{A}_3 = \{b, e\}$$

and

$$S = \{\{2\}, \{1, 3\}, \{1, 2\}, \{2, 3\}, \{1, 2, 3\}\}$$

leads to α'. Thus there is a necessary condition not covered by the argument scheme.

3.2.3 Other formulations

The definition of an argument scheme naturally translates into a pseudo boolean representation, because it mixes logical constraints and cardinality constraints.

$$\begin{aligned} \text{Card1PB}: &\quad \textstyle\sum_{(g,b)\in\mathbb{A}} \lambda_{gb} = k \\ \text{Card2PB}: &\quad \textstyle\sum_{i\in\mathcal{N}} \beta_i = k - 1 \end{aligned}$$

$$\text{C1PB}: \quad \forall i \in \mathcal{N}, \forall \langle g, b\rangle \in \mathbb{A}, g \succ_i b, \quad \beta_i + \neg\lambda_{gb} \geq 1$$

$$\begin{aligned} \text{C2PB}: \quad & \forall i \in \mathcal{N}, \forall p \neq q = 1, ..., mn, \\ & \forall \langle g, b\rangle \in \{g_p, g_q\} \times \{b_p, b_q\}, g \succ_i b, \\ & \qquad \neg\beta_i + \neg\lambda_{g_p b_p} + \neg\lambda_{g_q b_q} \geq 1 \end{aligned}$$

The reader can note that this formulation is not a perfect translation of the scheme's definition since, here, the number of pairs and the number of points of view in B are constants. However, we are able to use this formulation because it does not impact the decision problem result, *i.e.* "is there an argument scheme?". Indeed, if we consider a scheme with $k+n$ pairs to be split, with $n > 0$, and only $k-1$ points of view able to split at most one pair at a time, removing n pairs will not change the result since we still have more pairs to split than points of view to split them. This is possible because as it is defined, all pairs are two-by-two disjoints on all B's points of view meaning that for each pair of pairs of alternatives there is at least one of the REFUSED alternatives that is preferred to an ACCEPTED one, implying that for each pair of pairs, only one can be split at a time by a point of view in B. By removing n pairs, we still have k pairs two-by-two disjoint on each of the $k-1$ points of view, resulting in having k pairs to be split and only $k-1$ points of view capable to split at most one pair at a time. On all other points of view, the pairs are still non-splittable. Thus, we still have our deadlock leading to have one pair that is not splittable. The decision problem is then solved by this formulation but it cannot be used to enumerate all the schemes a configuration can contain. In other words, every scheme with a lot more pairs than points of view in B is subsumed by a scheme with only k pairs and $k-1$ points of view.

There exists a SAT counterpart of this formulation:

$$\begin{aligned} \text{Card1SAT}: &\quad |\{\langle g, b\rangle \in \mathbb{A} : \lambda_{gb} \text{ is true }\}| = k \\ \text{Card2SAT}: &\quad |\{i \in \mathcal{N} : \beta_i \text{ is true }\}| = k - 1 \\ \text{C1SAT}: &\quad \bigwedge_{\substack{\forall i\in\mathcal{N} \\ \forall\langle g,b\rangle\in\mathbb{A}, \\ g\succ_i b}} \beta_i \vee \neg\lambda_{gb} \end{aligned}$$

C2SAT :

$$\bigwedge_{\substack{\forall i\in\mathcal{N} \\ \forall p\neq q=1,\ldots,mn \\ \forall\langle g,b\rangle\in\{g_p,g_q\}\times\{b_p,b_q\}, \\ g\succ_i b}} \neg\beta_i \vee \neg\lambda_{g_p b_p} \vee \neg\lambda_{g_q b_q}$$

Resulting to the formula:

$$\Phi_2: \quad \text{Card1SAT} \wedge \text{Card2SAT} \wedge \text{C1SAT} \wedge \text{C2SAT}$$

3.2.4 Explainable fragments

The fact that we found instances that both are negative to the inv-NCS problem and do not contain any argument scheme means that the scheme is not a complete representation of inv-NCS. However, it is too early to say that it lost all its value because its goal is to build a humanly readable explanation of a complex combinatorial problem, and so it does. Thus the question of how well it approximates the inv-NCS problem opens up.

To further investigate this question, we can see what happens if we fix the scheme's maximum length (*i.e.* the number of points of view in the set B). Its goal, as said before, is to build a humanly readable justification for a necessary decision and thus fixing the maximum size of the scheme can be interesting since the smaller, the clearer it is. This new model allows us to create nested explainable fragments that grow with the upper bound we place on the length. For instance, if we only consider schemes of size 1, we obtain a Pareto dominance characterization. If we increase the maximum size, we keep the fragment but add new deadlock characterizations with it. This is interesting because:

- The search for schemes of a given length can be implemented in polynomial time w.r.t the number of pairs in the assignment. This opens the question of the practical efficiency of this algorithm, compared to the SAT-based modeling and solving approach we used.
- Increasing the size of the scheme may push the border of the uncovered necessary decision meanwhile it also reduces the readability of the explanation. Thus the purpose of simplifying a complicated combinatorial model is less and less respected, the longer the scheme is. The question of setting the value of the upper bound so as to achieve a good trade-off between the expressiveness of the scheme and its intelligibility is an interesting question, lying at the edge between decision theory and cognitive psychology.

This nesting opens up several research questions like quantifying the proportion that each maximum size model can explain.

Knowing that the original scheme is not complete, it can be interesting to see if there are remarkable structures that can be added into it that do not increase the complexity of the explanation. See for example the structure of example 3 where we added one point of view and one alternative to example 2 resulting into a counterexample.

4 Performance comparison

Searching for counterexamples, we compare various argument scheme's formulations' and multiple solvers' efficiency. To implement these experimentations we used an ASUS G551J computer with an Intel® Core™ i7-4750HQ CPU @ 2.00GHz × 8 and 8 GiB of DDR3 clocked at 1600 MHz RAM. All execution times have been recorded using the user time of `/usr/bin/time` command on an Ubuntu 20.04.4 LTS OS.

Considering that the argument scheme contains cardinality constraints, we tried to use a pseudo boolean formulation. To solve both the SAT and pseudo boolean formulations we used Gophersat version 1.2 [5] which can solve both types of problems and has a useful API that helps us perform quick experiments. We compared the SAT resolution time by using Glucose version 4.0 [1] which is based on Minisat [6].

We first introduce the process we followed to compare the performances between formulations and solvers. We compared a SAT and an equivalent pseudo boolean formulation using Gophersat solver. Finally, we compared the efficiency of Gophersat and Glucose on finding an argument scheme.

On this last part, we expected to have at least better performances with Glucose. However this assumption has been proven wrong on most of the instances. We have to note that we used the default parameters of each solver, without trying to fit them to the task at hand, which might seem more penalizing for Glucose, which offers many options.

4.1 Process

In this subsection we introduce the process used to compare the influence of various parameters on the solving time. We still focus on the case where all preferences are strict total orders.

The parameters we studied are the following:

- The number of alternatives: n
- The number of points of view: m
- The rate of ACCEPTED alternatives: p
- The minimal scheme's size (the number of pairs concerned): s

We generated the test samples using the algorithm of Listing 1

To test each parameter's influence, we generated the following set of configurations.

The default value for each configuration is:

- number of alternatives: 20
- number of points of view: 5
- rate of ACCEPTED alternatives: 35%
- minimal scheme's size: 2

Here are the various values tested for each parameter (only one parameter is tested at a time, the others take there default value):

- number of alternatives : 10, 20, 30, 40, 50
- number of points of view : 3, 4, 5, 7, 10
- rate of ACCEPTED alternatives : 10%, 35%, 50%, 75%, 90%
- minimal scheme's size : -1 (no scheme), 1, 2, 3, 4, 5, 6

100 instances were generated for each setting.

4.2 Formulation comparison (pseudo boolean versus SAT)

In order to perform a comparison between a pseudo-boolean formulation and a SAT formulation, we cannot use the SAT formulation presented in Section 3.2.1. The reason is that this formulation allows us to call only once on the solver whereas the pseudo-boolean necessitate multiple calls (one for each possible value of k). Thus, we will compare the pseudo-boolean formulation and its SAT counterpart proposed in Section 3.2.3

```
# INPUT :
#   n: number of alternatives
#   m: number of points of view
#   p: rate of ACCEPTED alternatives
#   s: minimal scheme's size
# OUTPUT :
#   - The preferences
#   - The sorting
GENERATE(n,m,p,s):
    alternatives := [1..n]
    accepted := [1..floor(n*p)]
    refused := [floor(n*p)+1..n]
    do
      preferences := []
      for i from 0 to m-1
        preferences[i] := copy(alternatives)
        preferences[i] = shuffle(preferences[i])
      done
      formula := formulate(preferences,
                           accepted, refused)
      min_size := resolveInc(formula)
    while min_size != s
    return (preferences, (accepted, refused))
```

Listing 1: Sample generation process. **shuffle()** use the Fisher-Yates shuffle to ensure equiprobability to all permutations. **formulate()** use the SAT formulation presented in Section 3.2.3. **resolveInc(formula)** is a function that incrementally solves the formula using Gophersat by increasing the size of the scheme on each iteration.

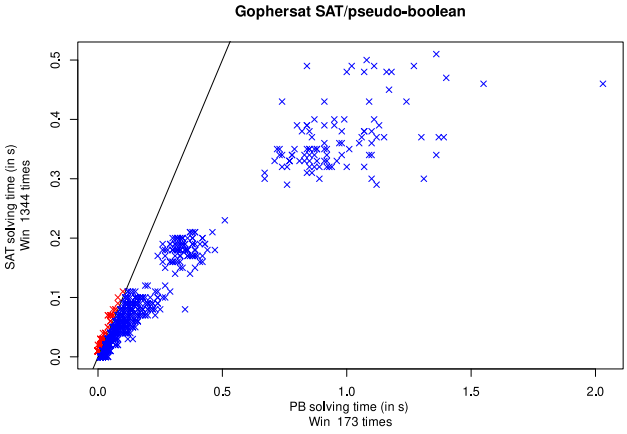

Figure 1. Scatter plot comparing the solving time (in seconds) with Gophersat of each instance in SAT to its equivalent pseudo boolean instance

Using Gophersat to solve equivalent SAT and pseudo-boolean equivalent instances with all generated instances, we obtain the scatter plot of Figure 1. This figure focuses only on the reading and solving

time of either a SAT or pseudo-boolean instance. In this setting, one can see that the SAT solving time is most of the time faster than its equivalent pseudo-boolean instance.

Name	1	2	3	4	5	6	7
nAlt	2/1	14/61	0/95	0/100	0/99	x	x
nPOV	10/61	8/52	8/62	8/53	5/62	x	x
%Acc	7/29	13/50	14/52	6/54	10/18	x	x
min_sol	0/100	16/42	10/50	12/67	9/69	15/74	6/92

Table 3. Comparing pseudo boolean (PB) and SAT relative performances on Gophersat with respect to all parameters. Each entry shows the number of instances where PB is faster/the number of instances where SAT is faster.
nAlt line shows the results with 1: 10, 2: 20, 3: 30, 4: 40, 5: 50 alternatives.
nPOV line shows the results with 1: 3, 2: 4, 3: 5, 4: 7, 5: 10 points of view.
%Acc line shows the results with 1: 10%, 2: 35%, 3: 50%, 4: 75%, 5: 90% of ACCEPTED alternatives.
min_sol line shows the results with scheme of minimal size equal to 1: -1, 2: 1, 3: 2, 4: 3, 5: 4, 6: 5, 7: 6, with -1 meaning that there is no scheme in the instance.

Table 3 shows the relative performances of pseudo-boolean and SAT instances with respect to each parameter. We can easily see that the more alternatives there are, the more SAT solving outperforms pseudo boolean solving. If we focus on the impact of the number of points of view, we can see that there seems to have two effects. Firstly, we can note that the more points of view there are, the less pseudo-boolean instances are faster than the equivalent SAT instances. Secondly, we can see that the parity of the number of points of view impacts the number of SAT instances that are faster that the equivalent pseudo-boolean instances. The rate of ACCEPTED alternatives seems to slow down the SAT solving more than the pseudo-boolean solving when this rate tends to extreme values. Finally, we can see that the size of the smallest scheme present in the instance has a strong influence on the relative performances. In particular, we can note that the greater the smallest size is, the more the SAT solving outperforms the pseudo-boolean solving. Considering the extreme case where there are no schemes, SAT is faster on every instance.

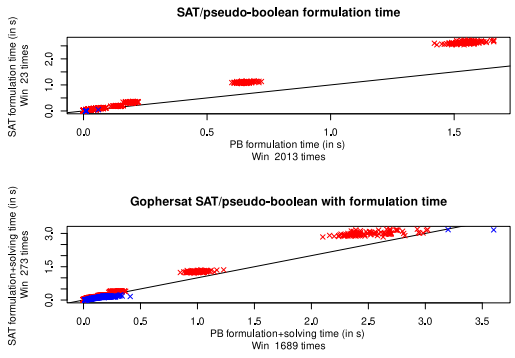

Figure 2. Line 1 : Scatter plot comparing the formulation time (in seconds) of each SAT instance to its equivalent pseudo boolean instance. Line 2 : Scatter plot comparing the formulation+solving time (in seconds) with Gophersat of each SAT instance to its equivalent pseudo boolean instance.

We saw that solving SAT instances is generally faster with Gophersat than its pseudo boolean equivalent when considering our problem. This holds when we focus on solving time, but if we take into account the formulation time, which is mostly faster with pseudo boolean output, the pseudo boolean instances become faster to solve as Figure 2 show us. We can note that this difference is less important than the one when considering only the solving time.

4.3 Solver comparison (Gophersat versus Glucose)

This subsection shares some results concerning the relative performances of Gophersat 1.2 and Glucose 4.0 SAT solvers on finding argument scheme. It is important to note that no tweaking has been done on the solvers' settings.

We compared the efficiency of the two previously presented SAT formulations. We will note SAT the one using a constant k explained in Section 3.2.3 and SATNoK the one without any constant introduced in Section 3.2.1.

We expected to find out that Glucose is more efficient than Gophersat, however Figure 3 shows us that for both formulations (SAT and SATNoK), Gophersat is at best equivalent or at worst outperforms Glucose.

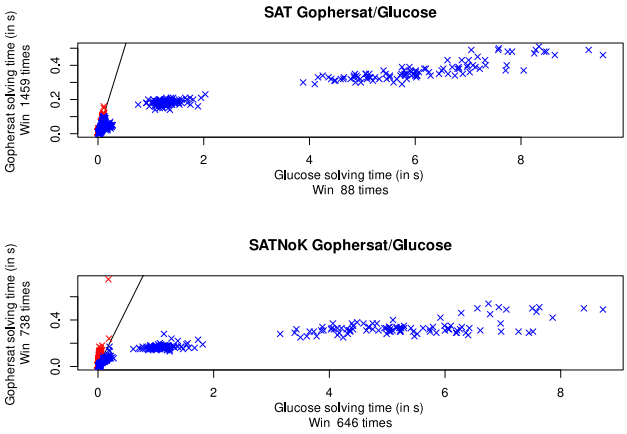

Figure 3. Line 1 : Scatter plot comparing the resolution time of a SAT instance on Gophersat and on Glucose. Line 2 : Scatter plot comparing the resolution time of a SATNoK instance on Gophersat and on Glucose.

To seek more details about those differences, Table 4 and Table 5 report the details about the parameters' influence.

We can see that the higher the number of alternatives is, the more Gophersat outperforms Glucose. However, the high number of points of view seems to be beneficial to Glucose. The ACCEPTED alternative rate is favorable to Glucose when it is set to an extreme value and to Gophersat when it is near 50%. The scheme size has a different influence with each of the formulations. With SAT the greater it is, the more Gophersat outperforms Glucose whereas it is the contrary if we use SATNoK.

We did not dig deeper on those results. We think that even if Gophersat is based on Glucose, the fact that we did not tweak any of the settings on either of them may influence the solving process to this point.

5 Conclusion

We considered the problem of explaining the results of a non compensatory model that are necessary w.r.t. a given jurisprudence, using

Name	1	2	3	4	5	6	7
nAlt	2/0	5/62	0/100	0/100	0/100	x	x
nPOV	0/74	1/73	7/66	4/63	6/57	x	x
%Acc	10/0	5/70	1/60	2/60	12/0	x	x
min_sol	10/82	4/64	4/69	1/93	0/97	3/90	11/79

Table 4. Comparing SAT instances relative performances when solved on Glucose and Gophersat with respect to all parameters. Each entry shows the number of instances where Glucose is faster/the number of instances where Gophersat is faster.
nAlt line shows the results with 1: 10, 2: 20, 3: 30, 4: 40, 5: 50 alternatives.
nPOV line shows the results with 1: 3, 2: 4, 3: 5, 4: 7, 5: 10 points of view.
%Acc line shows the results with 1: 10%, 2: 35%, 3: 50%, 4: 75%, 5: 90% of ACCEPTED alternatives.
min_sol line shows the results with scheme of minimal size equal to 1: -1, 2: 1, 3: 2, 4: 3, 5: 4, 6: 5, 7: 6, with -1 meaning that there is no scheme in the instance.

Name	1	2	3	4	5	6	7
nAlt	2/0	19/31	2/98	0/100	0/100	x	x
nPOV	15/35	23/28	27/29	37/32	64/22	x	x
%Acc	44/0	24/31	11/32	26/16	40/1	x	x
min_sol	96/0	0/61	22/28	37/35	48/23	50/25	59/11

Table 5. Comparing SATNoK instances relative performances when solved on Glucose and Gophersat with respect to all parameters. Each entry shows the number of instances where Glucose is faster/the number of instances where Gophersat is faster.
nAlt line shows the results with 1: 10, 2: 20, 3: 30, 4: 40, 5: 50 alternatives.
nPOV line shows the results with 1: 3, 2: 4, 3: 5, 4: 7, 5: 10 points of view.
%Acc line shows the results with 1: 10%, 2: 35%, 3: 50%, 4: 75%, 5: 90% of ACCEPTED alternatives.
min_sol line shows the results with scheme of minimal size equal to 1: -1, 2: 1, 3: 2, 4: 3, 5: 4, 6: 5, 7: 6, with -1 meaning that there is no scheme in the instance.

an argument scheme found in the literature. Contrarily to what has been conjectured, we show this scheme does not allow to explain all necessary decisions. This opens the question of assessing the size of the explainable fragment. During our experimentation, we found out that with preferences modeled as strict total orders and 6 alternatives and 3 points of view, the unexplainable fragment is extremely small. Furthermore, since the purpose of the scheme is to build a humanly readable explanation, it can be interesting to study a bounded version of the scheme where we fix an upper bound for the number of pairs or the number of points of view that are not trivially irrelevant. We found an interesting structure to extend the scheme to catch a part of the decision not covered by the scheme. The question of characterizing the decisions not covered by the scheme is left open.

REFERENCES

[1] G. Audemard and L. Simon. Glucose.
[2] O. Bailleux and Y. Boufkhad, 'Efficient cnf encoding of boolean cardinality constraints', in *CP2003*.
[3] K. Belahcene, Y. Chevaleyre, C. Labreuche, N. Maudet, V. Mousseau, and W. Ouerdane, 'Accountable approval sorting', in *2018 International Joint Conference on Artificial Intelligence (IJCAI)*.
[4] Denis Bouyssou and Thierry Marchant, 'An axiomatic approach to non-compensatory sorting methods in MCDM, I: the case of two categories', *Eur. J. Oper. Res.*, **178**(1), 217–245, (2007).
[5] Centre de Recherche en informatique de Lens. Gophersat.
[6] N. Eén and N. Sörensson. Minisat.
[7] A. Tlili, K. Belahcène, O. Khaled, V. Mousseau, and W. Ouerdane, 'Learning non-compensatory sorting models using efficient sat/maxsat formulations.', 979–1006.
[8] Douglas Walton, Christopher Reed, and Fabrizio Macagno, *Argumentation schemes*, Cambridge University Press, 2008.