Solving Soft Constraints for Multi-Criteria Ethical Reasoning

Hiroshi Hosobe[a,*] **and Ken Satoh**[b]

[a]Faculty of Computer and Information Sciences, Hosei University, Tokyo, Japan
[b]Center for Juris-Informatics, Research Organization of Information and Systems, Tokyo, Japan
ORCID (Hiroshi Hosobe): https://orcid.org/0000-0002-7975-052X

Abstract. It is becoming increasingly important to develop trustworthy computer systems. Especially, the ethics of artificial intelligence (AI) is attracting much attention, which requires various ethical issues to be considered in the development of AI-based systems. To tackle this problem, Taheri et al. proposed a framework for ethical decision making, which Hosobe and Satoh further integrated with Borning et al.'s constraint hierarchy framework to enable the powerful constraint-based modeling of problems for making decisions by using ethical norms. In this paper, we propose a method for solving constraint hierarchies with ethical norm constraints by adopting Hosobe and Satoh's framework. To our knowledge, this is the first proposed and implemented algorithm for actually solving constraint hierarchies based on this framework. Our method is hybrid in the sense that it combines a dedicated optimization algorithm with an external constraint solver. While the external solver satisfies ordinary hard constraints, the optimization algorithm treats soft ethical norm constraints. We also present the implementation of our method and the results of the experiment that we conducted to evaluate it.

1 Introduction

It is becoming increasingly important to develop trustworthy computer systems. Especially, the ethics of artificial intelligence (AI) is attracting much attention. This requires various ethical issues to be considered in the development of AI-based systems including, for example, privacy, manipulation of behavior, opacity of systems, and bias in decision making [16].

To tackle this problem, Taheri et al. [19] proposed a framework for ethical decision making. To declaratively treat ethical norms such as privacy and fairness, it formalized several important aspects of ethical decision making. The aspects include how to organize ethical norms, how to evaluate choices according to such norms, how to aggregate such evaluations for classes of norms, and how to decide the best choices for the entire norms.

Hosobe and Satoh [13] integrated Taheri et al.'s framework with Borning et al.'s constraint hierarchy framework [4] to enable the powerful constraint-based modeling of problems for making decisions by using ethical norms. Its main advantage is that ethical norms can be represented as soft constraints and can be combined with other kinds of hard constraints to model problems. Since ethical norms can be regarded as criteria that may conflict, this framework can be regarded also as a framework for multi-criteria ethical reasoning. However, their work still remained the proposal of the framework that lacked any mechanism of automated constraint solving.

In this paper, we propose a method for solving constraint hierarchies with ethical norm constraints by adopting Hosobe and Satoh's framework. To our knowledge, this is the first proposed and implemented algorithm for actually solving constraint hierarchies based on this framework. Although researchers have proposed various algorithms for solving constraint hierarchies, the previous ones were based on Borning et al.'s standard constraint hierarchy framework. Since Hosobe and Satoh's framework introduced a new kind of soft constraints for treating ethical norms, the previous algorithms are not applicable to this framework. To treat such soft constraints, we design our method in a hybrid manner in the sense that it combines a dedicated optimization algorithm with an external constraint solver. While the external solver satisfies hard constraints, the optimization algorithm treats soft ethical norm constraints. We also present the implementation of our method and the results of the experiment that we conducted to evaluate it.

The rest of this paper is organized as follows. Section 2 describes related work, Section 3 presents preliminaries, and Section 4 gives an example problem. Then Section 5 proposes our method, Section 6 describes its implementation, and Section 7 presents the results of the experiment. After Section 8 discusses the proposed method, Section 9 describes conclusions and future work.

2 Related Work

Borning et al. [4] proposed constraint hierarchies as a framework for modeling and solving over-constrained problems. In a constraint hierarchy, constraints are associated with preferences called strengths, and solutions are determined by maximally satisfying stronger constraints. Constraint hierarchies have been used especially for interactive graphical applications. For this purpose, researchers have developed many solvers of several kinds of constraints such as dataflow constraints [6, 20], linear arithmetic constraints [1, 10, 14], and nonlinear constraints [11].

There have been only a few solvers of constraint hierarchies over finite domains. Bistarelli et al. [3] proposed consistency techniques for the reduction of constraint hierarchies over finite domains as well as a branch-and-bound search algorithm for solving the reduced constraint hierarchies. Hosobe and Satoh [12] proposed binary search-based methods for solving constraint hierarchies over finite domains by encoding them into ordinary constraint satisfaction problems and

* Corresponding Author. Email: hosobe@acm.org.

solving them with an external constraint solver.

Fungwacharakorn et al. [7, 8] recently proposed the use of constraint hierarchies for reasoning with legal and ethical norms. In particular, they studied the effects of fundamental revisions on constraint hierarchies [7] and the connections of constraint hierarchies with case-based representation of norms [8]. However, their studies were focused on constraints represented as propositional logical formulas.

3 Preliminaries

In this section, we explain three frameworks as preliminaries to our proposed method.

3.1 Ethical Decision Making

First, we briefly explain Taheri et al.'s ethical decision making framework [9, 19]. It treats ethical norms such as data minimality, data sensitivity, gender fairness, racial fairness, and system performance. It organizes such norms into multiple norm classes that can be hierarchically structured. For example, data minimality and data sensitivity can be classified in a norm class called privacy. There may be other norm classes such as fairness and performance. Each norm is used to rank alternatives (or choices to be compared in decision making). A norm class aggregates the results of the rankings of the norms in the class by using a voting mechanism. The framework especially chooses Copeland's rule [18] for this purpose. The framework selects the best alternatives by using a relation called superiority for comparing alternatives according to norm classes with a given partial order.

3.2 Constraint Hierarchies

Next, we explain Borning et al.'s constraint hierarchy framework [4]. Let X be a set of variables. How to assign values to variables is expressed as a *valuation* that is a function from variables to their values. Let Θ be the set of all the valuations. Then a valuation $\theta \in \Theta$ obtains the value of a variable $x \in X$ as $\theta(x)$. Let C be a set of constraints. How much a constraint is satisfied by a valuation is given by an *error function* e. Specifically, $e(c, \theta)$ returns a non-negative real number: returning 0 means that c is exactly satisfied by θ; returning a larger number means that c is less satisfied.

A constraint hierarchy is typically represented as $H = \langle H_0, H_1, \ldots, H_l \rangle$, where l is some positive integer, and each H_i, called a *level*, is a set of constraints with strength i (that indicates their preference). Level H_0 consists of *required* (or hard) constraints that must be exactly satisfied while each H_i with $i \geq 1$ consists of *preferential* (or soft) constraints that can be relaxed if necessary. A typical constraint hierarchy is totally ordered, which means that a preferential level H_i with smaller i consists of more important constraints.

A *partially ordered hierarchy*, another type of constraint hierarchies, is represented as $\langle H, <_H \rangle$, where $H = \langle H_0, H_1, \ldots, H_l \rangle$, and $<_H$ is a partial order on $\{0, 1, \ldots, l\}$ with 0 as the only smallest element. The intuitive meaning of the required level H_0 is the same as in the case of totally ordered hierarchies. By contrast, the importance of the preferential levels is specified by partial order $<_H$; that is, if $i <_H j$, H_i has more important constraints than H_j. To define solutions to a partially ordered hierarchy, the notion of *consistent totally ordered hierarchies* is used. Given a partially ordered hierarchy $\langle H, <_H \rangle$, $\langle H, <'_H \rangle$ is a consistent totally ordered hierarchy if $<'_H$ is a total order on $\{0, 1, \ldots, l\}$ with 0 as the smallest element and there is a bijective mapping m of type $\{0, 1, \ldots, l\} \to \{0, 1, \ldots, l\}$ such that $i <_H j \to m(i) <'_H m(j)$. Intuitively, a consistent totally ordered hierarchy is a modification of a partially ordered hierarchy that rearranges preferential levels in a total order in such a way that any pair of ordered levels in the original partially ordered hierarchy will keep the same order in the resulting totally ordered hierarchy.

The importance of constraints in a constraint hierarchy is treated by a *comparator* better. Intuitively, $\mathsf{better}(\theta, \theta', S_0, \langle H, <'_H \rangle)$ judges whether a valuation θ is better than another θ' with respect to alternative valuations S_0 according to a totally ordered hierarchy $\langle H, <'_H \rangle$ that is consistent with a partially ordered hierarchy $\langle H, <_H \rangle$. It should be noted that the better comparator internally uses the error function e to evaluate individual constraints.

The solution set of a partially ordered hierarchy is defined as the union of the solution sets of all the totally ordered constraint hierarchies consistent with the original partially ordered hierarchy.

Definition 1 (solution)**.** Given a partially ordered hierarchy $\langle H, <_H \rangle$, the set $S_{\mathsf{po}}(\langle H, <_H \rangle)$ of all the solutions to $\langle H, <_H \rangle$ is defined as

$$S_{\mathsf{po}}(\langle H, <_H \rangle) = \bigcup_{\langle H, <'_H \rangle \in \mathcal{H}} S_{\mathsf{to}}(\langle H, <'_H \rangle)$$

where $\mathcal{H}$ is the set of all the totally ordered hierarchies consistent with $\langle H, <_H \rangle$ and

$$S_{\mathsf{to}}(\langle H, <'_H \rangle) = \{\theta \in S_0 \mid \forall \theta' \in S_0 \; (\neg\, \mathsf{better}(\theta', \theta, S_0, \langle H, <'_H \rangle))\}$$

where

$$S_0 = \{\theta \in \Theta \mid \forall c \in H_0 (e(c, \theta) = 0)\}.$$

3.3 Constraint Hierarchies with Ethical Norm Constraints

Finally, we explain Hosobe and Satoh's extended constraint hierarchy framework [13], which was constructed by integrating Taheri et al.'s framework [9, 19] with Borning et al.'s framework [4]. A *norm constraint* evaluates a given valuation according to the associated ethical norm and assigns a rank to the valuation. Let C_{norm} be a set of norm constraints. Then the meaning of norm constraints is defined with a *ranking function* r. Intuitively, $r(c, \theta)$ obtains the rank of a valuation θ according to a norm constraint c. Ranks are expressed with positive integers, and a smaller positive integer indicates a better rank, typically with 1 as the best. This is formally defined as follows.

Definition 2 (ranking function)**.** Let $\mathbb{Z}^+$ be the set of all the positive integers. A ranking function r is a function of type $C_{\mathrm{norm}} \times \Theta \to \mathbb{Z}^+$.

The framework defines a constraint hierarchy as a partially ordered hierarchy $\langle H, <_H \rangle$ with $H = \langle H_0, H_1, \ldots, H_l \rangle$ and a partial order $<_H$ with 0 as the only smallest element. It assumes that the required level H_0 consists of only ordinary constraints and that the other levels $H_1, \ldots, H_l$, called the *norm levels*, consist of only norm constraints.

To evaluate how much a valuation satisfies a norm level, it uses a *combining function* g that computes the Copeland score of the valuation according to the norm level. As already mentioned in Subsection 3.1, Copeland's rule [18] was used by Taheri et al.'s ethical decision making framework. Since Copeland's rule gives a voting mechanism that allows multiple voters to cooperatively determine the best choices by computing Copeland scores, it is well suited also

to this framework in evaluating multiple ethical norm constraints at a norm level. Intuitively, $g(\theta, S_0, H_i)$ computes the Copeland score of a valuation θ with respective to alternative valuations S_0 according to a norm level H_i. This is defined as follows.

Definition 3 (combining function)**.** Given a valuation θ, a set S_0 of valuations with θ (i.e., $\theta \in S_0$), and a set H_i of norm constraints, the combining function g is defined as

$$g(\theta, S_0, H_i) = \sum_{\theta' \in S_0 \setminus \{\theta\}} p(\theta, \theta', H_i)$$

where

$$p(\theta, \theta', H_i) = \begin{cases} 1 & \text{if } w(\theta, \theta', H_i) > w(\theta', \theta, H_i) \\ 1/2 & \text{if } w(\theta, \theta', H_i) = w(\theta', \theta, H_i) \\ 0 & \text{otherwise} \end{cases}$$

where

$$w(\theta, \theta', H_i) = \left|\{c \in H_i \mid r(c, \theta) < r(c, \theta')\}\right|.$$

This definition is based on an adaptation of Copeland scores to this framework. Function $w(\theta, \theta', H_i)$ uses the ranking function r to compute the number of the wins of θ against θ' according to the norm constraints in H_i. Function $p(\theta, \theta', H_i)$ computes the elemental score of θ against θ' according to H_i. Function $g(\theta, S_0, H_i)$ accumulates the elemental scores to compute the Copeland score of θ according to H_i

The framework defines a comparator better as follows.

Definition 4 (comparator)**.** Given a constraint hierarchy $\langle H, <_H \rangle$, valuations θ and θ', a set S_0 of valuations with θ and θ' (i.e., $\theta, \theta' \in S_0$), and a total order $<'_H$ consistent with $<_H$, the comparator better is defined as

$$\begin{aligned} &\mathsf{better}(\theta, \theta', S_0, \langle H, <'_H \rangle) \equiv \\ &\quad \exists k \in \{1, \ldots, l\} (\forall i \in \{1, \ldots, l\} \\ &\qquad (i <'_H k \rightarrow g(\theta, S_0, H_i) = g(\theta', S_0, H_i)) \wedge \\ &\qquad g(\theta, S_0, H_k) > g(\theta', S_0, H_k)). \end{aligned}$$

The framework defines solutions to a constraint hierarchy $\langle H, <_H \rangle$ as $S_{\mathsf{po}}(\langle H, <_H \rangle)$, which is formulated by Definition 1.

4 Example

In this section, we illustrate an example problem based on Hosobe and Satoh's constraint hierarchy framework [13], which was explained in Subsection 3.3. This example also was presented in [13], and we use and extend it for our experiment in Section 7. This example was adapted from Hayashi et al.'s use case of their multi-agent real-time compliance mechanism [9]. The original use case considered a job recommendation service operated by an employment platform company and performed legal and ethical reasoning. The company implements this service as a distributed system consisting of multiple nodes, some of which are inside the European Union (EU) and the others of which are outside. It wants to process its customers' data by using a remote processing node. Although both its user node and the remote processing node are inside the EU, some of the intermediate nodes between these nodes are outside the EU. Since the data include the customers' privacy information, it needs to treat the data by following legal rules such as the EU's General Data Protection Regulation (GDPR). In addition, it wants to treat the data by respecting ethical norms such as privacy and fairness.

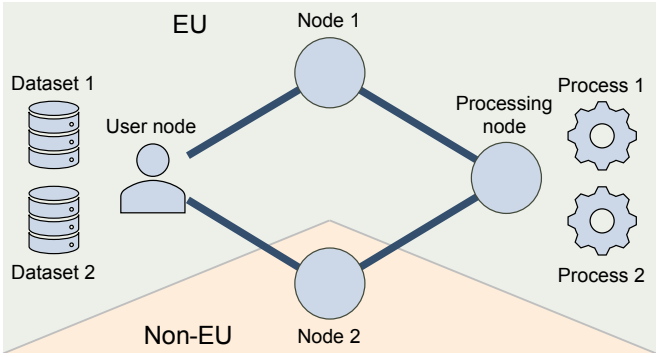

Figure 1. Problem of legal and ethical reasoning for a job recommendation service adapted from Hayashi et al.'s use case of their compliance mechanism [9].

This example problem is modeled as a constraint hierarchy by simplifying this use case as illustrated in Figure 1. It consists of five variables d, u, n, m, and p, each of which ranges over a domain with two values, and two required and six norm constraints associated with ethical norms data_minimality, data_sensitivity, transfer_safety, node_safety, algo_unbiasedness, and transfer_efficiency as follows:

$$\mathsf{required}\ d = \mathsf{data2} \rightarrow u = \mathsf{analysis} \tag{1}$$

$$\mathsf{required}\ u = \mathsf{recommendation} \tag{2}$$

$$\mathsf{data\_minimality}\ (d = \mathsf{data1}) \triangleright (d = \mathsf{data2}) \tag{3}$$

$$\mathsf{data\_sensitivity}\ (d = \mathsf{data1}) \triangleright (d = \mathsf{data2}) \tag{4}$$

$$\begin{aligned} \mathsf{transfer\_safety}\ &(n = \mathsf{node1} \wedge m = \mathsf{node1}) \triangleright \\ &(n = \mathsf{node1} \wedge m = \mathsf{node2}) \circ \\ &(n = \mathsf{node2} \wedge m = \mathsf{node1}) \triangleright \\ &(n = \mathsf{node2} \wedge m = \mathsf{node2}) \end{aligned} \tag{5}$$

$$\begin{aligned} \mathsf{node\_safety}\ &(n = \mathsf{node1} \wedge m = \mathsf{node1}) \triangleright \\ &(n = \mathsf{node1} \wedge m = \mathsf{node2}) \circ \\ &(n = \mathsf{node2} \wedge m = \mathsf{node1}) \triangleright \\ &(n = \mathsf{node2} \wedge m = \mathsf{node2}) \end{aligned} \tag{6}$$

$$\mathsf{algo\_unbiasedness}\ (p = \mathsf{process1}) \triangleright (p = \mathsf{process2}) \tag{7}$$

$$\begin{aligned} \mathsf{transfer\_efficiency}\ &(n = \mathsf{node2} \wedge m = \mathsf{node2}) \triangleright \\ &(n = \mathsf{node1} \wedge m = \mathsf{node2}) \circ \\ &(n = \mathsf{node2} \wedge m = \mathsf{node1}) \triangleright \\ &(n = \mathsf{node1} \wedge m = \mathsf{node1}). \end{aligned} \tag{8}$$

The five variables have the following roles. First, variable d indicates which of two datasets data1 and data2 should be used. Next, variable u indicates for which of two purposes recommendation and analysis they can use the datasets. In this use case, the dataset is processed at a remote processing node. Variable n and m indicate which of intermediate nodes node1 and node2 should be used when the selected dataset is transferred to and returned from the remote processing node respectively. Finally, variable p indicates which of two processes process1 and process2 should be used at the remote processing node.

The eight constraints have the following meanings. First, required constraint (1) means that dataset data2 can be used only for purpose analysis. Next, required constraint (2) means that the current situation needs to consider purpose recommendation. The other six are

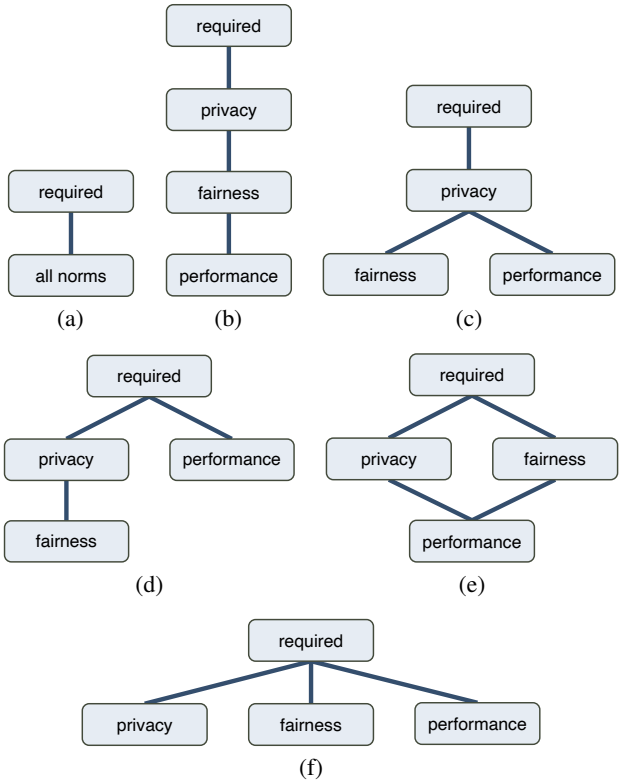

Figure 2. Structures of constraint hierarchies.

norm constraints. Each norm constraint is represented as a sequence of more primitive constraints with either operator $\triangleright$ (meaning that the next one is worse ranked) or $\circ$ (meaning that the next one is equally ranked). Norm constraints (3) and (4) mean that, in the senses of data_minimality and data_sensitivity respectively, using data1 is better than using data2. Constraints (5) and (6) mean that, in the senses of transfer_safety and node_safety respectively, using node1 twice is the best, using node2 only once is the second best, and using node2 twice is the worst. Constraint (7) means that, in the sense of algo_unbiasedness, using process1 is better than using process2. Constraint (8) means that, in the sense of transfer_efficiency, using node2 twice is the best, using node1 only once is the second best, and using node1 twice is the worst.

This example considers different structures of norm levels shown in Figure 2. In the case of Figure 2(a), all the norm constraints belong to one norm level. In the other cases, the four norm constraints (3), (4), (5), and (6) associated with norms data_minimality, data_sensitivity, transfer_safety, and node_safety belong to the same norm level privacy, and norm constraints (7) and (8) associated with algo_unbiasedness and transfer_efficiency belong to other norm levels fairness and performance respectively. In addition, in the case of Figure 2(b), all the levels are totally ordered, but in the other cases of Figures 2(c), 2(d), 2(e), and 2(f), the levels are partially ordered.[1] It also considers further different subcases where the norm levels are differently placed for the same structure of the partial order.

This constraint hierarchy obtains solutions $\langle d, u, n, m, p\rangle$ as $\langle$data1, recommendation, node1, node1, process1$\rangle$, $\langle$data1, recommendation, node2, node2, process1$\rangle$, or both, depending on the given partial order of the norm levels. In Table 1 of Section 7, we give the solutions computed by our implementation of the proposed algorithm. For more details, see [13].

[1] In this paper, we add the case corresponding to Figure 2(d), which was missing in [13].

5 Proposed Method

In this paper, we propose a method for solving constraint hierarchies with ethical norm constraints. Theoretically, our method is based on Hosobe and Satoh's constraint hierarchy framework [13]. Given a constraint hierarchy on finite domains that consists of required (or hard) constraints and preferential (or soft) ethical norm constraints, it finds the set of all the solutions to the constraint hierarchy in the sense of the underlying framework. The method is hybrid in the sense that it combines a dedicated optimization algorithm with an external constraint solver. While the external solver satisfies the required constraints, the optimization algorithm treats the ethical norm constraints.

Our method internally performs two kinds of multi-objective optimization. First, inside each norm level, it treats multiple norm constraints that may conflict. For this type of conflicts, it performs multi-objective optimization by using a voting mechanism called the Copeland's rule [18], which it inherits from Taheri et al.'s framework [19] and Hosobe and Satoh's framework [13]. Second, it treats multiple norm levels that also may conflict. For this type of conflicts, it performs multi-objective optimization by using a partially ordered hierarchy, which it inherits from Borning et al.'s framework [4] and Hosobe and Satoh's framework.

Algorithm 1 shows the entire process of the proposed method. Initially, it takes a partially ordered hierarchy $\langle H, <_H\rangle$ as an input. At line 2, it solves the required constraints H_0 by calling the external constraint solver and obtains the result Θ_0, which corresponds to S_0 in the underlying framework. Next, at lines 3 to 7, it processes each potential solution $\theta \in \Theta_0$ by computing the tuple $\boldsymbol{g} = \langle g(\theta, \Theta_0, H_1), \ldots, g(\theta, \Theta_0, H_l)\rangle$ of all the combining functions based on Definition 3, and obtains the set G of all the pairs $\langle \theta, \boldsymbol{g}\rangle$. Next, at line 8, it obtains the set T of all the total orders $<'_H$ consistent with $<_H$ in the same way as Boring et al.'s original framework. Next, at lines 9 to 22, it performs further optimization by treating the ethical norm constraints to find the best solutions from Θ_0. More specifically, for each total order $<'_H \in T$, it obtains the best solutions Θ^* in the sense of $<'_H$ and merges them with Θ. To find the best solutions Θ^*, it enumerates all the potential solutions θ to gradually update the tentatively best combining function results $\boldsymbol{g}^*$. For this purpose, it uses a special function greater$(\boldsymbol{g}, \boldsymbol{g}^*, <'_H)$, which compares the currently treated combining function results $\boldsymbol{g}$ with the tentatively best results $\boldsymbol{g}^*$ in the sense of $<'_H$, which corresponds to the comparator better in Definition 4. Finally, at line 23, it returns Θ, which corresponds to $S_{\mathsf{po}}(\langle H, <_H\rangle)$ in the underlying framework.

Computing the set of all the total orders consistent with a given partially ordered constraint hierarchy $\langle H, <_H\rangle$ in getConsistentTotalOrders at line 8 of Algorithm 1 is done as follows. First, all the permutations $m(1), m(2), \ldots, m(l)$ of $1, 2, \ldots, l$ for the number l of the norm levels are generated. Next, for each permutation, it is checked whether the permutation is consistent with $<_H$ (based on the definition presented in Subsection 3.2). Finally, the set of all the consistent permutations is returned. It should be noted that a simple implementation of generating permutations is sufficient because l is usually small (e.g., $l = 1$ or $l = 3$ in the constraint hierarchies shown in Figure 2). However, in the case of many norm levels, a more sophisticated algorithm would be needed.

Algorithm 1: Method for solving a partially ordered constraint hierarchy with ethical norm constraints.

Data: A partially ordered constraint hierarchy $\langle H, <_H \rangle$
Result: The set Θ of all the solutions to $\langle H, <_H \rangle$

1 **begin**
2 $\quad \Theta_0 \leftarrow \mathsf{solveRequiredConstraints}(H_0)$;
3 $\quad G \leftarrow \emptyset$;
4 $\quad$ **foreach** $\theta \in \Theta_0$ **do**
5 $\quad\quad \boldsymbol{g} \leftarrow \langle g(\theta, \Theta_0, H_1), \ldots, g(\theta, \Theta_0, H_l) \rangle$;
6 $\quad\quad G \leftarrow G \cup \{\langle \theta, \boldsymbol{g} \rangle\}$;
7 $\quad$ **end**
8 $\quad T \leftarrow \mathsf{getConsistentTotalOrders}(<_H)$;
9 $\quad \Theta \leftarrow \emptyset$;
10 $\quad$ **foreach** $<'_H \in T$ **do**
11 $\quad\quad \boldsymbol{g}^* \leftarrow \langle 0, \ldots, 0 \rangle$;
12 $\quad\quad \Theta^* \leftarrow \emptyset$;
13 $\quad\quad$ **foreach** $\langle \theta, \boldsymbol{g} \rangle \in G$ **do**
14 $\quad\quad\quad$ **if** $\mathsf{greater}(\boldsymbol{g}, \boldsymbol{g}^*, <'_H)$ **then**
15 $\quad\quad\quad\quad \boldsymbol{g}^* \leftarrow \boldsymbol{g}$;
16 $\quad\quad\quad\quad \Theta^* \leftarrow \{\theta\}$;
17 $\quad\quad\quad$ **else if** $\boldsymbol{g} = \boldsymbol{g}^*$ **then**
18 $\quad\quad\quad\quad \Theta^* \leftarrow \Theta^* \cup \{\theta\}$;
19 $\quad\quad\quad$ **end**
20 $\quad\quad$ **end**
21 $\quad\quad \Theta \leftarrow \Theta \cup \Theta^*$;
22 $\quad$ **end**
23 $\quad$ **return** Θ;
24 **end**

6 Implementation

We developed a constraint solver in Scala Native by implementing the algorithm proposed in the previous section. It adopts a solver of satisfiability modulo theory (SMT) [15] called Z3 [5] as the external constraint solver for solving required constraints (to implement solveRequiredConstraints in Algorithm 1). However, it should be noted that it could replace the external solver with another standard one from the field of constraint programming [17] or satisfiability (SAT) [2] since it performs only finite-domain constraint solving for this purpose. The solver represents ethical norm constraints as simple functions written in Scala Native that correspond to ranking functions in Definition 2.

7 Experiment

This section reports the experiment that we conducted to evaluate the method proposed in Section 5.

7.1 Problem Setting

In our experiment, we used the constraint hierarchy shown in Section 4. In addition, we extended this basic form of the constraint hierarchy by introducing more variables and larger-arity constraints. Specifically, we increased variables n and m by k times to $n_1, n_2, \ldots, n_k$ and $m_1, m_2, \ldots, m_k$ respectively. Also, we extended the arities of norm constraints (5), (6), and (8) by making them to refer to $n_1, n_2, \ldots, n_k$ and $m_1, m_2, \ldots, m_k$. Specifically, the ranking function of norm constraints (5) and (6) was defined to return one plus the number of the assignments of node2 to $n_1, n_2, \ldots, n_k$ and $m_1, m_2, \ldots, m_k$, and the ranking function of norm constraint (8) was defined similarly but using the assignments of node1 instead. For example, in the case of $k = 2$, norm constraint (5) is changed to the following:

$$\begin{aligned}\mathsf{transfer\_safety}\ (&n_1 = \mathsf{node1} \wedge n_2 = \mathsf{node1} \wedge \\ &m_1 = \mathsf{node1} \wedge m_2 = \mathsf{node1}) \triangleright \\ &(n_1 = \mathsf{node1} \wedge n_2 = \mathsf{node1} \wedge \\ &m_1 = \mathsf{node1} \wedge m_2 = \mathsf{node2}) \circ \\ &(n_1 = \mathsf{node1} \wedge n_2 = \mathsf{node1} \wedge \\ &m_1 = \mathsf{node2} \wedge m_2 = \mathsf{node1}) \circ \\ &\cdots \triangleright \\ &(n_1 = \mathsf{node2} \wedge n_2 = \mathsf{node2} \wedge \\ &m_1 = \mathsf{node2} \wedge m_2 = \mathsf{node2}).\end{aligned}$$

It should also be noted that the basic form of the constraint hierarchy corresponds to the case of $k = 1$. Hereafter, we refer to k as the size of the constraint hierarchy. A constraint hierarchy of size k consists of $2k + 3$ variables and two required constraints (1) and (2), three norm constraints (3), (4), and (7), and three versions of norm constraints (5), (6), and (8) extended to treat $n_1, n_2, \ldots, n_k$ and $m_1, m_2, \ldots, m_k$.

7.2 Results

We solved the constraint hierarchies shown in the previous subsection by executing our constraint solver on an Apple M3 processor with 8 GB of memory using macOS 14.6.1, Scala Native 0.5.5, Clang 15.0.0, and Z3 4.13.0. In the following and Table 1, we denote privacy, fairness, and performance as pv, fr, and pf respectively, and data1, data2, recommendation, node1, node2, process1, and process2 as d1, d2, rec, n1, n2, p1, and p2 respectively for brevity.

Table 1 shows all the results. Column "Structure" corresponds to structures (a) to (f) shown in Figure 2. Column "Hierarchy" indicates how the norm levels were placed based on the structure. Especially, the first row of each structure corresponds to the placement of the norm levels shown in Figure 2. The notation $\mathsf{pv} <_H \{\mathsf{fr}, \mathsf{pf}\}$ for instance means that both $\mathsf{pv} <_H \mathsf{fr}$ and $\mathsf{pv} <_H \mathsf{pf}$ hold but the order does not hold between fr and pf. Column "Computed solutions for $k = 1$" indicates what solutions were computed by the solver for the constraint hierarchy of size $k = 1$. We confirmed that the solutions were the same as the ones shown in [13] (except for structure (d), which was missing). Finally, columns "Mean times (ms)" indicate the means of the execution times in milliseconds needed to solve the same hierarchy ten times in each case. Overall, the solver required almost the same length of time to solve a single constraint hierarchy of each size regardless of its structure.

8 Discussion

The proposed method uses only a simple algorithm for finding the best solutions from the potential ones obtained by enumerating all solutions to required constraints. Because of this limitation, our solver is currently not efficient. In fact, we observed that, in the case of large problems, most of the execution time was spent on the computation of the Copeland scores of the potential solutions according to ethical norm constraints (at lines 3 to 7 in Algorithm 1). Especially, this inefficiency was caused by the brute-force nature of comparison among potential solutions. To solve this limitation, we need to incorporate a more sophisticated algorithm such as dynamic programming and branch-and-bound search.

Table 1. Results of solving different structures and sizes of constraint hierarchies with ethical norm constraints.

Structure	Hierarchy	Computed solutions for $k = 1$	Mean times (ms)				
			$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
(a)	$\mathsf{pv} \cup \mathsf{fr} \cup \mathsf{pf}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n1}, \mathsf{n1}, \mathsf{p1} \rangle$	7.0	13.1	59.7	968.1	24160.9
(b)	$\mathsf{pv} <_H \mathsf{fr} <_H \mathsf{pf}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n1}, \mathsf{n1}, \mathsf{p1} \rangle$	7.0	13.2	62.8	983.5	26228.3
	$\mathsf{pv} <_H \mathsf{pf} <_H \mathsf{fr}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n1}, \mathsf{n1}, \mathsf{p1} \rangle$	7.0	12.3	61.8	983.1	26746.5
	$\mathsf{fr} <_H \mathsf{pv} <_H \mathsf{pf}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n1}, \mathsf{n1}, \mathsf{p1} \rangle$	7.1	11.9	62.4	985.9	26839.5
	$\mathsf{fr} <_H \mathsf{pf} <_H \mathsf{pv}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n2}, \mathsf{n2}, \mathsf{p1} \rangle$	7.3	12.4	61.5	989.9	26694.6
	$\mathsf{pf} <_H \mathsf{pv} <_H \mathsf{fr}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n2}, \mathsf{n2}, \mathsf{p1} \rangle$	7.1	12.9	61.3	983.0	26664.8
	$\mathsf{pf} <_H \mathsf{fr} <_H \mathsf{pv}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n2}, \mathsf{n2}, \mathsf{p1} \rangle$	7.1	12.3	61.9	983.9	26352.6
(c)	$\mathsf{pv} <_H \{\mathsf{fr}, \mathsf{pf}\}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n1}, \mathsf{n1}, \mathsf{p1} \rangle$	7.1	13.2	60.7	982.1	26632.8
	$\mathsf{fr} <_H \{\mathsf{pv}, \mathsf{pf}\}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n1}, \mathsf{n1}, \mathsf{p1} \rangle$, $\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n2}, \mathsf{n2}, \mathsf{p1} \rangle$	7.1	13.5	60.8	979.6	26712.2
	$\mathsf{pf} <_H \{\mathsf{pv}, \mathsf{fr}\}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n2}, \mathsf{n2}, \mathsf{p1} \rangle$	7.1	12.9	61.3	982.4	26572.3
(d)	$\{\mathsf{pv} <_H \mathsf{fr}, \mathsf{pf}\}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n1}, \mathsf{n1}, \mathsf{p1} \rangle$, $\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n2}, \mathsf{n2}, \mathsf{p1} \rangle$	7.4	13.1	61.2	986.0	26464.3
	$\{\mathsf{pv} <_H \mathsf{pf}, \mathsf{fr}\}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n1}, \mathsf{n1}, \mathsf{p1} \rangle$	8.6	12.9	61.7	985.4	26501.3
	$\{\mathsf{fr} <_H \mathsf{pv}, \mathsf{pf}\}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n1}, \mathsf{n1}, \mathsf{p1} \rangle$, $\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n2}, \mathsf{n2}, \mathsf{p1} \rangle$	8.0	12.1	60.9	983.5	26470.7
	$\{\mathsf{fr} <_H \mathsf{pf}, \mathsf{pv}\}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n1}, \mathsf{n1}, \mathsf{p1} \rangle$, $\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n2}, \mathsf{n2}, \mathsf{p1} \rangle$	7.8	12.9	61.8	988.4	26558.1
	$\{\mathsf{pf} <_H \mathsf{pv}, \mathsf{fr}\}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n2}, \mathsf{n2}, \mathsf{p1} \rangle$	9.1	12.4	61.8	979.5	26347.0
	$\{\mathsf{pf} <_H \mathsf{fr}, \mathsf{pv}\}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n1}, \mathsf{n1}, \mathsf{p1} \rangle$, $\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n2}, \mathsf{n2}, \mathsf{p1} \rangle$	8.6	13.0	61.1	981.2	26533.2
(e)	$\{\mathsf{pv}, \mathsf{fr}\} <_H \mathsf{pf}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n1}, \mathsf{n1}, \mathsf{p1} \rangle$	9.0	13.0	61.5	980.8	26676.1
	$\{\mathsf{pv}, \mathsf{pf}\} <_H \mathsf{fr}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n1}, \mathsf{n1}, \mathsf{p1} \rangle$, $\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n2}, \mathsf{n2}, \mathsf{p1} \rangle$	8.9	13.2	61.6	983.4	26541.2
	$\{\mathsf{fr}, \mathsf{pf}\} <_H \mathsf{pv}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n2}, \mathsf{n2}, \mathsf{p1} \rangle$	8.1	12.9	62.1	981.9	26435.8
(f)	$\{\mathsf{pv}, \mathsf{fr}, \mathsf{pf}\}$	$\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n1}, \mathsf{n1}, \mathsf{p1} \rangle$, $\langle \mathsf{d1}, \mathsf{rec}, \mathsf{n2}, \mathsf{n2}, \mathsf{p1} \rangle$	8.4	12.2	61.5	983.1	26391.5

Although the proposed method uses an external solver to solve required constraints, the method does not use it to treat ethical norm constraints. In [12], Hosobe and Satoh solved constraint hierarchies by encoding soft constraints and satisfying them with an external solver. However, we think that it is not straightforward to similarly encode ethical norm constraints because of the complex formulation of their combining functions in Definition 3. Therefore, further exploring this approach will need to construct an efficient way of encoding the combining functions of ethical norm constraints.

9 Conclusions and Future Work

In this paper, we proposed a method for solving constraint hierarchies with ethical norm constraints. To our knowledge, the method is the first proposed and implemented algorithm for actually solving constraint hierarchies based on Hosobe and Satoh's framework [13]. We also presented its implementation and the results of the experiment that we had conducted to evaluate our method.

Our future work includes improving of the performance of the method by incorporating a more sophisticated search algorithm. Another direction is to construct a method for encoding ethical norm constraints into ordinary constraint satisfaction problems or SAT problems. We also need to perform a further experiment by using larger and more complex problems. Furthermore, we will explore practical applications of our constraint solver to show its effectiveness in real-world problems.

Acknowledgements

This work was supported by JST AIP Trilateral AI Research Grant Number JPMJCR20G4.

References

[1] G. J. Badros, A. Borning, and P. J. Stuckey. The Cassowary linear arithmetic constraint solving algorithm. *ACM Trans. Comput.-Human Interact.*, 8(4):267–306, 2001.

[2] A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*. IOS Press, 2nd edition, 2021.

[3] S. Bistarelli, P. Codognet, K. C. Hui, and J. H.-M. Lee. Solving finite domain constraint hierarchies by local consistency and tree search. In *Proc. CP*, volume 2833 of *LNCS*, pages 138–152, 2003.

[4] A. Borning, B. Freeman-Benson, and M. Wilson. Constraint hierarchies. *Lisp Symbolic Comput.*, 5(3):223–270, 1992.

[5] L. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *Proc. TACAS*, volume 4963 of *LNCS*, pages 337–340, 2008.

[6] B. N. Freeman-Benson, J. Maloney, and A. Borning. An incremental constraint solver. *Comm. ACM*, 33(1):54–63, 1990.

[7] W. Fungwacharakorn, K. Tsushima, and K. Satoh. Fundamental revisions on constraint hierarchies for ethical norms. In *Proc. JURIX*, volume 362 of *FAIA*, pages 182–187, 2022.

[8] W. Fungwacharakorn, K. Tsushima, H. Hosobe, H. Takeda, and K. Satoh. Connecting rule-based and case-based representations of soft-constraint norms. In *Proc. JURIX*, volume 379 of *FAIA*, pages 149–154, 2023.

[9] H. Hayashi, T. Mitsikas, Y. Taheri, K. Tsushima, R. Schäfermeier, G. Bourgne, J.-G. Ganascia, A. Paschke, and K. Satoh. Multi-agent online planning architecture for real-time compliance. In *RuleML+RR Companion*, volume 3485 of *CEUR WS*, 2023.

[10] H. Hosobe. A scalable linear constraint solver for user interface construction. In *Proc. CP*, volume 1894 of *LNCS*, pages 218–232, 2000.

[11] H. Hosobe. A modular geometric constraint solver for user interface applications. In *Proc. ACM UIST*, pages 91–100, 2001.

[12] H. Hosobe and K. Satoh. Binary search-based methods for solving constraint hierarchies over finite domains. In *Proc. IEEE ICTAI*, pages 186–193, 2023.

[13] H. Hosobe and K. Satoh. A soft constraint-based framework for ethical reasoning. In *Proc. ICAART*, volume 3, pages 1355–1362. SCITEPRESS, 2024.

[14] K. Marriott and S. S. Chok. QOCA: A constraint solving toolkit for interactive graphical applications. *Constraints*, 7(3–4):229–254, 2002.

[15] D. Monniaux. A survey of satisfiability modulo theory. In *Proc. CASC*, volume 9890 of *LNCS*, pages 401–425, 2016.

[16] V. C. Müller. Ethics of artificial intelligence and robotics. In *The Stanford Encyclopedia of Philosophy*. 2023. https://plato.stanford.edu/entries/ethics-ai/.

[17] F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.

[18] D. G. Saari and V. R. Merlin. The Copeland method—I.: Relationships and the dictionary. *Econ. Theory*, 8:51–76, 1996.

[19] Y. Taheri, G. Bourgne, and J.-G. Ganascia. Modelling integration of responsible AI values for ethical decision making. In *Proc. Workshop on Computational Machine Ethics (CME)*, 2023. https://github.com/yousef-taheri/responsibleAi.

[20] B. Vander Zanden. An incremental algorithm for satisfying hierarchies of multi-way dataflow constraints. *ACM Trans. Prog. Lang. Syst.*, 18 (1):30–72, 1996.