Runtime Norm Revision Using Bayesian Networks

Davide Dell'Anna(✉), Mehdi Dastani, and Fabiano Dalpiaz

Utrecht University, Utrecht, The Netherlands
{d.dellanna,m.m.dastani,f.dalpiaz}@uu.nl

Abstract. To guarantee the overall intended objectives of a multiagent systems, the behavior of individual agents should be controlled and coordinated. Such coordination can be achieved, without limiting the agents' autonomy, via runtime norm enforcement. However, due to the dynamicity and uncertainty of the environment, the enforced norms can be ineffective. In this paper, we propose a runtime supervision mechanism that automatically revises norms when their enforcement appears to be ineffective. The decision to revise norms is taken based on a Bayesian Network that gives information about the likelihood of achieving the overall intended system objectives by enforcing the norms. Norms can be revised in three ways: relaxation, strengthening, and alteration. We evaluate the supervision mechanism on an urban smart traffic simulation.

Keywords: Norm revision · Multiagent systems · Bayesian networks

1 Introduction

A multiagent system consists of (heterogeneous) autonomous agents that coexist and interact in a shared open environment [1]. In order to guarantee the overall intended objectives of a multiagent system, the behavior of the autonomous agents should be coordinated [2]. In the multiagent systems literature, runtime norm enforcement is a widely studied mechanism for controlling and coordinating the runtime behavior of the agents without limiting their autonomy [3,4]. For example, a smart road populated by autonomous cars can control the cars' behavior by enforcing traffic rules, such as speed limitations, in order to improve throughput and safety of the road. In this paper, we do not focus on norm enforcement and assume that norms can be enforced on autonomous agents by means of, e.g., regimentation and sanctioning mechanisms.

However, due to the dynamicity and uncertainty involved in the agents' operating environments, such as sudden changes of weather conditions or accidents due to heavy traffic, enforcing the existing norms may not be sufficient to ensure the overall intended objectives [5]. For example, the enforcement of some speed norms may not improve traffic throughput or safety when the weather conditions change from extreme to normal (or vice versa). Given a set of norms and

T. Miller et al. (Eds.): PRIMA 2018, LNAI 11224, pp. 279–295, 2018.
https://doi.org/10.1007/978-3-030-03098-8_17

a set of environmental conditions, it is often hard—or practically impossible—to predict the effectiveness of enforcing specific norms in various environmental conditions [6]. This suggests that continuous evaluation of norm enforcement and dynamic revision of the norms at runtime are key factors to build an effective normative multiagent system capable of ensuring the overall system objectives within a dynamic and uncertain environment [7].

We introduce a runtime norm supervision mechanism that monitors the behavior of a multiagent system, evaluates the enforcement of the norms in terms of the overall system objectives, and, when needed, intervenes by revising the norms. For example, if the enforcement of adaptive traffic lights on a smart road significantly decreases the safety of cars in extreme weather conditions, our runtime norm supervision mechanism will suggest a revision of such norm proposing the enforcement of static traffic lights.

This paper focuses on the evaluation and revision of the norms enforced in the system. We describe three main types of norm revision (relaxation, strengthening and alteration) and provide two heuristic algorithms for suggesting norm revisions based on data that is collected and encoded into a Bayesian Network (called *Norm Bayesian Network*) at runtime. Such network is used to learn and reason about the correlation between norm satisfaction/violation and the overall system objectives achievement. The runtime norm supervision mechanism is implemented as an optimization process that uses a variation of the hill climbing optimization technique. Our revision algorithms are used in the optimization process to determine the next sets of norms to enforce in the system.

We report on an experimental evaluation of the supervision mechanism by applying it to an urban traffic simulation. Our implementation guarantees the identification of norm sets that ensure the overall system objectives. We compare the results obtained using hill climbing combined with and without our revision engine (i.e., Bayesian Network and revision strategies) and show that the proposed mechanism, using runtime information about norms effectiveness, allows to find optimal solutions with less revisions.

The paper is structured as follows. Section 2 describes a case study concerning urban traffic management. Section 3 presents the runtime supervision mechanism and the *Norm Bayesian Network*. Section 4 describes two algorithms for the suggestion of norms revision. Section 5 evaluates the effectiveness of the approach. Finally, Sects. 6 and 7 review related studies and conclude the paper.

2 Case Study: Norms for Urban Traffic Management

Consider a city where 10% of the cars are autonomous self-driving cars operating in a road network that is enriched with autonomous traffic controllers such as smart traffic lights and panels. The city council aims at improving the urban traffic by achieving two overall objectives: *minimize the average travel time* and *minimize the number of accidents*. To achieve such objectives, the city council plans to control and coordinate the traffic by enforcing traffic norms. To this end, the council is able to prescribe the self-driving cars to use a centralized

navigation service (CNS) instead of their personal navigation system. The CNS can be either adaptive (able to autonomously change its own parameters at runtime) or static. The council may also prescribe the use of specific traffic controllers at road junctions. In particular, the council can enforce five possible obligation norms in the city:

- *every self-driving car is obliged to employ the routes suggested by an adaptive or static CNS* (denoted by $O(ans)$ and $O(sns)$).
- *every road junction is obliged to employ adaptive traffic lights* ($O(atl)$), *static traffic lights* ($O(stl)$), or *priority lines panels* ($O(pl)$).

Due to the highly dynamic nature of the city, drivers and cars can behave differently in different contexts (in this paper, we consider four contexts: *day* or *night*, *normal* or *extreme* weather). This makes it hard to determine in advance which set of norms will be the most effective in every context. For this reason, the city council aims at developing a traffic management system that starts with a set of norms, enforces the norms at run-time and monitors whether the enforcement of the norms is effective in the sense that they will guarantee the achievement of the overall objectives of minimizing travel time and accidents. In case norms are learned not to be effective, the traffic management system is expected to revise the norms accordingly.

Consider the norms $O(ans)$ and $O(sns)$ to be mutually exclusive, as well as the norms $O(atl), O(stl)$ and $O(pl)$. Based on these relationships, the city council disposes of a set $\mathcal{N}$ of 12 possible norm sets that could be enforced in the city: $\mathcal{N} = \{\{O(ans)\}, \{O(sns)\}, \{O(atl)\}, \{O(stl)\}, \{O(pl)\}, \{O(ans), O(atl)\}, \{O(ans), O(stl)\}, \{O(ans), O(pl)\}, \{O(sns), O(atl)\}, \{O(sns), O(stl)\}, \{O(sns), O(pl)\}, \emptyset\}$. When a navigation service is prescribed (e.g., in case of $\{O(ans)\}$) the routes suggested are followed by the self-driving cars in the 80% of the cases; when no navigation service is prescribed (e.g., in case of $\{O(pl)\}$) the self-driving cars use their own navigation system; finally when no junction management is prescribed (e.g., in case of $\{O(ans)\}$) the cars approaching the smart junctions follow the default priority-to-the-right rule.

In this paper norms are obligations expressed as propositional state formulae in conjunctive normal form and O denotes the obligation operator. Norms are considered here to be regimented, rather than enforced by means of sanctions. We leave sanction-based enforcement mechanism for future work; here, we study in detail the effect of imposing different norms on a multiagent system, their relationship with overall system objectives in different contexts and the possible strategies to revise norms when they are not effective.

3 Runtime Norm Supervision

We present the control loop performed by the runtime supervision mechanism (sketched in Fig. 1) to automatically evaluate and revise norms in order to achieve the overall objectives of a multiagent system.

Given a set $\mathcal{O}$ of boolean overall system objectives, a set $\mathcal{N}$ of all possible norm sets enforceable in the system and a set $\mathcal{C}$ of all possible contexts of the multiagent system, we call system *configuration* an assignment of a norm set $N \in \mathcal{N}$ to each possible context in $\mathcal{C}$. For example, given the four possible contexts $(day, normal)$, $(day, extreme)$, $(night, normal)$, $(night, extreme)$, and given the set $\mathcal{N}$ of possible norm sets defined in Sect. 2, an example of system configuration is $\{\langle(day, normal), \{O(p)\}\rangle, \langle(day, extreme), \{O(ans), O(p)\}\rangle, \langle(night, normal), \{O(sns)\}\rangle, \langle(night, extreme), \{O(atl), O(ans)\}\rangle\}$. A norm n is said to be *active* in a context c if $\langle c, N\rangle$ is in the system configuration and $n \in N$. Otherwise n is said *disabled*.

The control loop of the supervision mechanism starts with a *Norm Base* containing an initial system configuration. We assume a runtime *Monitoring* component that perfectly collects information about the satisfaction or violation of the norms in the contexts in which they are evaluated. Such component provides a boolean evaluation of the overall system objectives. This knowledge is used to learn, by means of a *Norm Bayesian Network* (described in Sect. 3.1), the dependencies between the satisfaction of the norms and the achievement of the objectives in the different contexts. A *Revision Trigger* component (described in Sect. 3.2) uses the learned knowledge to determine whether some norms should be revised. The norm revision process is executed by the *Revision Engine* component (described in Sect. 3.3) and generates as output a (possibly) new system configuration, replacing the current one in the *Norm base*.

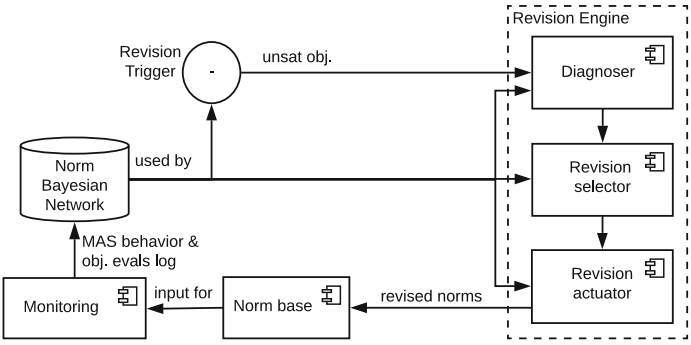

Fig. 1. The control loop of the runtime supervision mechanism.

We propose an implementation of the control loop above described as a variation of the hill climbing optimization technique. In particular, we consider the system configurations as possible solutions to explore in order to find an optimal one. The quality of a solution is determined, by means of runtime data, as the average probability of achievement of the overall objectives in all the contexts. Instead of terminating the exploration of the space when a local optimum is found, as in traditional hill climbing, we use as stopping criterion a constraint

defined by the system designer that determines whether or not the current solution is acceptable (see Sect. 3.2). We use the *Revision Engine* to determine the next solution to try, when the current one is not acceptable.

Sections 3.1 and 4 describe the main components involved in the control loop. Section 5 provides an experimental evaluation of our implementation of the control loop. We compare the results that hill climbing can obtain by using our Revision Engine for the neighborhood definition with results that can be obtained by using heuristics that do not evaluate the effectiveness of the norms.

3.1 Norm Bayesian Network

Bayesian Networks have been widely used in many fields as knowledge representation structures for learning and reasoning about the inter-dependencies between their nodes [8]. We define a type of Bayesian Network called *Norm Bayesian Network* to represent and reason about norms and their relationship with overall objectives in different contexts. We call *contextual variables* monitorable environmental properties such as *Time* and *Weather*. Each of these variables is associated to a domain of values (e.g., *Time* can be either *day* or *night*, *Weather* can be either *normal* or *extreme*). Given a set of contextual variables we call *context* an assignment of a value to each contextual variable (e.g., given *Time* and *Weather* as above described, we have four possible contexts: (*day*, *normal*), (*day*, *extreme*), (*night*, *normal*), (*night*, *extreme*)).

A *Norm Bayesian Network* $\mathcal{NBN} = (\mathcal{X}, \mathcal{A}, \mathcal{P})$ is a Bayesian Network where:

- $\mathcal{X} = \mathbf{N} \cup \mathbf{O} \cup \mathbf{C}$ is a set of nodes, representing random variables in probability theory. The sets $\mathbf{N}$, $\mathbf{O}$ and $\mathbf{C}$ are disjoint. The set $\mathbf{N}$ consists of *norm nodes*; each node $N \in \mathbf{N}$ corresponds to one norm and has a discrete domain of 3 possible values: *obeyed*, *violated* and *disabled*. The set $\mathbf{O}$ consists of *objective nodes*; each node $O \in \mathbf{O}$ corresponds to a boolean objective and has a discrete domain of 2 values: *true* and *false*. Finally, the set $\mathbf{C}$ consists of *context nodes*; each node $C \in \mathbf{C}$ corresponds to a contextual variable and can have a discrete or continuous domain of values.
- $\mathcal{A} \subseteq (\mathbf{C} \times \mathbf{N}) \cup (\mathbf{C} \times \mathbf{O}) \cup (\mathbf{N} \times \mathbf{O})$ is the set of arrows connecting pairs of nodes. If there is an arrow from node X to node Y, X is called parent of Y.
- $\mathcal{P}$ is a set of conditional probability distributions, each one associated with a node in $\mathcal{X}$ and quantifying the effect of the parents on the node.

In the context of Bayesian Networks we use the following notation. X, Y, ... (italic uppercase) denotes random variables; $\mathbf{X}$, $\mathbf{Y}$, ... (bold uppercase) denotes sets of random variables; v_1, v_2, ... (italic lowercase) denotes values in the domain of a random variable; X_v denotes an assignment ($X = v$) of value v to a random variable X; $\mathbf{x}$ (bold lowercase) denotes an assignment of values to a set of nodes $\mathbf{X}$; $\mathbf{X}_v$ denotes an assignment of value v to all nodes in $\mathbf{X}$; X_{act}

denotes the fact $\neg X_{dis} = \neg(X = disabled)$); $\neg \mathbf{x}_v^{\mathbf{X}}$ is equivalent to $\bigwedge_{X \in \mathbf{X}}(\neg X_v)$; $\mathbf{P}$ denotes a probability distribution; P denotes a single probability[1].

Figure 2 reports the *structure* $(\mathcal{X}, \mathcal{A})$ of an $\mathcal{NBN}$ for our case study. Notice the three types of nodes representing contextual variables, norms and overall objectives. Arrows connect each contextual variable to all norms and all objectives, and each norm to all the objectives. Nodes *Travel_Time* and *Accidents* correspond to the two overall objectives of minimizing travel time and accidents.

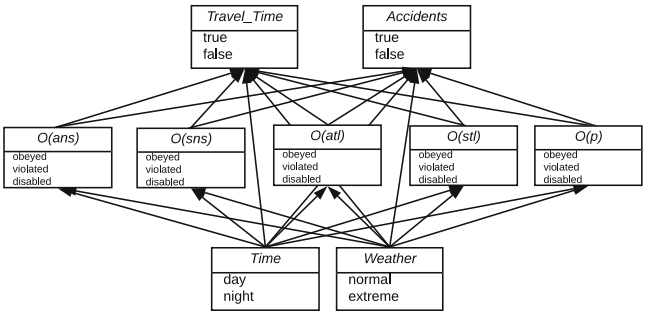

Fig. 2. The structure of the $\mathcal{NBN}$ for the case study of Sect. 2.

In Bayesian Networks, an evidence $\mathbf{e}$ is an observed assignment of values for some or all of the random variables in the network. An evidence $\mathbf{c}$ for all the context nodes $\mathbf{C}$ is an observation for a certain context (e.g., *Time* has value *day* and *Weather* has value *extreme*). For simplicity we use the term *context* also to refer to the associated evidence in the Bayesian Network. An evidence of value *obeyed* or *violated* for a norm node can be obtained when the corresponding norm is enforced in the multiagent system. When the norm is not enforced the only possible evidence for its corresponding node is *disabled*. Note that the evidence values are determined by the *Monitoring* component. Norms nodes, therefore, only collect statistical information about norm obedience in different contexts.

Note also that, for the sake of brevity, we omit a discussion about the learning technique (e.g., classical Bayesian learning) to train the network and we refer the reader to the existing literature (e.g., [8,9]). In the following, we assume that we dispose of a network trained with data produced by the *Monitoring* component.

3.2 Revision Trigger

Let e be an event denoting that changes in the probability distributions in the *Norm Bayesian Network* are not significant anymore (i.e., the variations in the distribution when a new sample is given are below a specified δ). Assuming

[1] When we refer to nodes of a specific type we use the corresponding notation convention, e.g., N refers to a node in $\mathbf{N}$, $\mathbf{c}$ refers to an assignment of values to nodes in $\mathbf{C}$, $\mathbf{N}_{viol}$ refers to an assignment of value *violated* to a set of norm nodes $\mathbf{N}$, etc.

a population of agents that behaves consistently, such event will occur after some time. Let t_{oa} be a threshold defining the minimum average probability of achievement of the objectives desired by the system designer, and let c be the related constraint (e.g., $c = P(Travel\_Time_{true} \wedge Accidents_{true}) \geq 0.95$, with $t_{oa} = 0.95$). A revision (i.e., a new iteration of the hill climbing procedure) is triggered every time e occurs and c is not satisfied.

3.3 Revision Engine

Assume that the *Norm Bayesian Network* $\mathcal{NBN}$ of Fig. 2 is trained with data and that a norm revision is triggered by the *Revision Trigger*. We describe the three components of the *Revision Engine* shown in Fig. 1. Such components are used to determine the new norm set to enforce in the multiagent system.

Diagnoser. This component uses the $\mathcal{NBN}$ to generate an explanation for the objectives not being achieved. To do so, it first determines a context **mpc** that corresponds to the most problematic context in which the objectives are not achieved. $\mathbf{mpc} = argmax_{\mathbf{c} \in all(\mathbf{c})} P(\mathbf{O}_{false} \mid \mathbf{c})$, where $all(\mathbf{c})$ is the set of all possible contexts (assignments of a value to each of the context nodes in $\mathcal{NBN}$).

Let $\mathcal{N}_{\mathbf{mpc}}$ be the set of norms currently active in the context corresponding to **mpc**, and $\mathbf{N}_a$ and $\mathbf{N}_d$ be the two disjoint sets of nodes in $\mathcal{NBN}$ that corresponds to the norms that are respectively active and disabled in the most problematic context. The Diagnoser determines the *most likely explanation* [10] $\mathbf{n}_e$ for $\mathbf{O}_{false}$ given **mpc**, in terms of satisfaction of the active norms in $\mathcal{N}_{\mathbf{mpc}}$.

$$\mathbf{n}_e = argmax_{\mathbf{n} \in \mathbf{n}^{\mathbf{N}_a}_{\{ob,viol\}}} P(\mathbf{n} \mid \mathbf{O}_{false} \wedge \mathbf{mpc} \wedge \mathbf{n}^{\mathbf{N}_d}_{dis} \wedge \neg \mathbf{n}^{\mathbf{N}_a}_{dis})$$

where $\mathbf{n}^{\mathbf{N}_a}_{\{ob,viol\}} = \{N_v | N \in \mathbf{N}_a, v \in \{ob, viol\}\}$ is the set of all the possible assignments of values (either *obeyed* or *violated*) to nodes $\mathbf{N}_a$.

Revision Selector. Given the most likely explanation $\mathbf{n}_e$, we aim to revise $\mathcal{N}_{\mathbf{mpc}}$ so to increase $P(\mathbf{O}_{true} \mid \mathbf{mpc})$ above the threshold t_{oa}. The Revision Selector determines the most adequate *type of revision* to perform.

We define three types of norm revision: *relaxation*, *strengthening*, *alteration*. Relaxing (strengthening) a propositional obligation norm $O(n)$ means replacing it with a new norm $O(n')$ such that n' is a less strict (stricter) formula than n (e.g., $O(ans \vee \bot)$ is a less strict obligation than $O(ans)$). A less strict (stricter) obligation makes the norm violated in fewer (more) situations, which means that more (less) behaviors are allowed. Any other revision of a norm $O(n)$ is an alteration. Relaxing (strengthening) a *set* of norms N means replacing it with a new norm set N' such that one or more norms in N' are a relaxation (strengthening) of norms in N and all the other norms are unchanged (e.g., $\{O(ans), O(atl)\}$ is a strenghtening of $\{O(atl)\}$). Any other revision of a set of norms N is an alteration of N (e.g., $\{O(ans)\}$ is an alteration of $\{O(sns)\}$).

The Revision Selector applies the following idea. To achieve the overall objectives in a certain context, some of the active norms are more useful if obeyed, others are more useful if violated, and the rest are always harmful (regardless of the fact that they are obeyed or violated). This information can be derived from the conditional probability distributions of nodes $\mathbf{N}$ as follows.

Consider the case of a single active norm N in $\mathcal{N}_{\mathbf{mpc}}$ ($\mathbf{N}_a = \{N\}$), a single boolean objective O, and a single context[2]. Figure 3a illustrates revision strategies based on the three types of relationships between N and O above described.

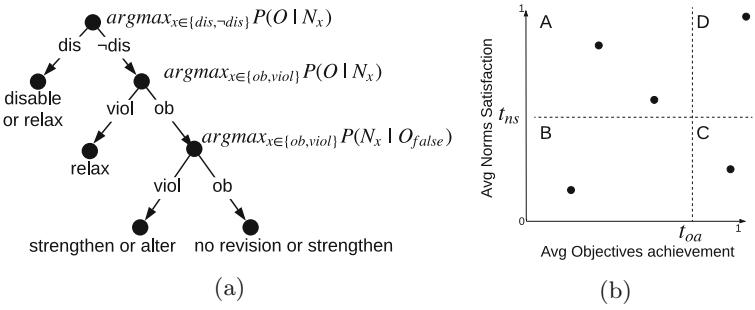

Fig. 3. (a) Decision tree for determining a suitable type of revision. (b) system configurations (points) in four states (A–D) w.r.t. the average satisfaction of the enforced norms and the average probability of achievement of the objectives.

The decision tree first determines whether the norm N is more useful for O when disabled or active. In the former case the norm is considered harmful, i.e., $N_{harm} \equiv (P(O_{true} \mid N_{dis}) > P(O_{true} \mid \neg N_{dis}))$. The most suitable strategy in this case is either to disable N or to relax N (relaxation is a "soft" kind of disabling, for it allow more behaviors). If N is not harmful, the decision tree compares the probabilities of N supporting O when obeyed and when violated.

If N is more useful for O when violated ($P(O_{true} \mid N_{viol}) > P(O_{true} \mid N_{ob}) \wedge \neg N_{harm}$), the suggested revision is to relax it, thereby turning some non-compliant (but useful to achieve O) behaviors into compliant ones. If N is more useful when obeyed (i.e., $P(O_{true} \mid N_{ob}) > P(O_{true} \mid N_{viol}) \wedge \neg N_{harm}$), the tree computes the most likely explanation for O not being achieved, between N being violated or obeyed. In the former case, the suggested revision is to strengthen N (restricting the allowed behaviors) or alter it (N is not effective). In the latter case, the suggestion is either to not revise (N may not be the cause of the problem) or to strengthen (N may not be strong enough to achieve O).

Section 4 presents two revision selection algorithms that apply the principle here described to the entire set of norms enforced in the system. The output of the Revision Selector is a set of norms annotated with suggested revisions (if any).

[2] In the following we omit the context from the conditional probabilities since implicit.

Revision Actuator. The task of this component is to determine the new norm set to enforce in the system. Given the set of norms annotated with suggested revisions (from the Revision Selector), and given a set of possible norm sets $\mathcal{N}$, the Revision Actuator selects a norm set $\mathcal{N}' \in \mathcal{N}$ that is as aligned as possible with the direction provided by the suggestions. For example, given the norm set $\{O(ans)\}$ and a suggestion of disabling norm $O(ans)$, the norm set in $\mathcal{N}$ that is the most aligned with the suggestion is the empty set $\{\}$. If multiple norm sets are available, different distance metrics can be defined, e.g., the similarity with the current norm set, or the sensitivity of the objectives to the change of the selected norms. In this paper, we use as distance metric the number of revision of norms needed to obtain $\mathcal{N}'$ from $\mathcal{N}$. For instance, two revisions are necessary to to obtain $\{O(ans), O(p)\}$ from $\{O(sns), O(atl)\}$: $O(sns)$ must be altered to $O(ans)$ and $O(atl)$ must be altered to $O(p)$. If there is no new norm set that is aligned with the provided suggestion (i.e., either the defined neighborhood of the current solution is empty or it contains only configurations that have already been tried) the Revision Actuator randomly selects a configuration never tried before, if any. Notice that this makes our implementation of hill climbing different from a traditional one and it guarantees to always converge to an optimal solution.

4 Revision Selection Algorithms

We present two algorithms for the Revision Selector component. The pureBN algorithm enacts the decision tree of Fig. 3a for all the enforced norms. The stateBased algorithm takes also into account the overall status of the system in the current configuration. Both algorithms are invoked on the most problematic context defined in Sect. 3.3 and they both return a set of norms annotated with suggested revisions. Such set of norms is used as heuristic to determine the neighborhood of the current norm set in the hill climbing process.

4.1 PureBN

Algorithm 1 reports the pureBN heuristic for the selection of norms revision. Line 2 determines, based on the top decision node of Fig. 3a, the norm set $\mathcal{N}'$ that has the highest probability to satisfy the objectives. If the current norm set has active norms, and there is some norm set that has not been attempted in the most problematic context yet (line 3), the most likely explanation $\mathbf{n}_e$ for not fulfilling the objectives is determined (line 4). Then, for each active norm in the new norm set $\mathcal{N}'$ (line 5), the algorithm determines the "desired" state, i.e., whether that norm helps better the satisfaction of the objectives when obeyed or violated (line 6). Finally, in line 7, the suggestion for the examined norm is determined based on the decision tree of Fig. 3a. If no norms are active or all possible norm sets have already been tried, pureBN returns $\mathcal{N}'$: the best possible norm set for the most problematic context (skipping lines 4–7).

Algorithm 1. The pureBN algorithm for revision selection

```
1: function pureBN(c)
2:     N' ← getBestNormSet(c)                          ▷ obtain N'_a and N'_d
3:     if (|N_a| > 0) && ¬allNormSetTried(c) then
4:         n_e ← getMLE(O_false, c)
5:         for all norms N ∈ N'_a do
6:             n_des ← getDesiredVal(N, c)
7:             setSugg({N}, getSugg(N, n_e, n_des))
       return N'
```

4.2 StateBased

While pureBN provides a suggestion for each active norm, stateBased considers the average satisfaction of active norms and the average achievement of the overall objectives to suggest a specific type of revision per time.

Figure 3b plots system configurations in four states with respect to norms satisfaction and objectives achievement. The configurations in state A sufficiently satisfy the norms, but the objectives are not achieved to a sufficient extent. State B denotes insufficient norms satisfaction and objectives achievement. State C indicates that the objectives are achieved even though the norms are not satisfied. State D is the ideal area: the norms are satisfied and the objectives are achieved.

Assume we have, together with the threshold t_{oa}, an additional threshold t_{ns} that defines the desired probability of satisfaction of norms. stateBased first determines the average norms satisfaction and objectives achievement based on the evidence from the active norms in the most problematic context (lines 2). Like in pureBN, the best possible norm set in the most problematic context is generated (line 3). If there are currently no active norms or $\mathcal{N}'$ contains at least one suggestion of type `disable`, the function returns immediately (line 4).

Three empty sets of norms are defined in line 5: obeyed norms that are better when violated *obbv*, violated norms that are better when violated *vabv*, and violated norms that are better when obeyed *vbbo*. After determining the most likely explanation $\mathbf{n}_e$ (line 6), the algorithm determines (line 7) the desired state (obeyed, violated) of all active norms of the new norm set. Using $\mathbf{n}_e$ (obeyed/violated) and the desired state (obeyed/violated), the norms are added to the corresponding sets *obbv*, *vavb*, and *vbbo* (line 8).

Lines 9–16 implement the idea visualized in Fig. 3b by comparing the state of the current norm set (*ns* and *oa*) against the thresholds t_{ns} and t_{oa}. If the configuration is in state A (lines 9–11), if *obbv* contains norms, the suggestion is to relax them; if *obbv* is empty, a suggestion to alter or strengthen is given for the active norms (the active norms behave as expected but the objectives are not achieved). In state B (lines 12–13), a relaxation of the norms that are better if violated is suggested, and an alteration or strengthening is suggested for the norms that are better if obeyed. In state C (lines 14–16), if there are violated norms that are better if violated, they are relaxed; if, instead, there are violated norms that are better if obeyed, the suggestion is to either alter or strengthen.

Algorithm 2. The STATEBASED algorithm for revision selection

```
1: function STATEBASED(c)
2:     ns ← AVGNORMSAT(N_a, c); oa ← AVGOBJACH(N_a, c)
3:     N' ← GETBESTNORMSET(c)                          ▷ obtain N'_a and N'_d
4:     if (|N_a| = 0) || HASDIS(N') then return N'
5:     obbv ← vabv ← vbbo ← {}
6:     n_e ← GETMLE(O_false, c)
7:     n_des ← GETDESIREDVAL(N'_a, c)
8:     DETERMINETYPE(N'_a, n_e, n_des, obbv, vabv, vbbo)
9:     if ns ≥ t_ns && oa < t_oa then
10:        if |obbv| > 0 then SETSUGG(obbv, relax)
11:        else SETSUGG(N_a, alter ∨ strengthen)
12:    else if rs < t_ns && ga < t_oa then
13:        SETSUGG(oobv ∪ vabv, relax); SETSUGG(vbbo, alter ∨ strengthen)
14:    else if rs < t_ns && ga ≥ t_oa then
15:        if |vabv| > 0 then SETSUGG(vabv, relax)
16:        else if |vbbo| > 0 then SETSUGG(vbbo, alter ∨ strengthen)
       return N'
```

5 Evaluation

We conducted an experimental evaluation of our implementation of the runtime norm supervision mechanism in terms of *convergence speed* and *quality of the final solution*. We compare the results that can be obtained by using our *Revision Engine* with results that can be obtained with heuristics that do not evaluate the effectiveness of the norms.

We make use of a simulation of the scenario described in Sect. 2. We adopt the CrowdNav simulator from the self-adaptive systems literature [11]. CrowdNav consists of a number of cars traveling in a medium-size city (Eichstädt, Germany) with 450 streets and 1,200 intersections. Each car relies on a navigation service to receive a route. 90% of the cars use a default routing algorithm implemented in SUMO (the underlying traffic simulation engine), while the remaining 10% are smart cars that use a centralized navigation service. In CrowdNav such service is adaptive, i.e., it is able to autonomously adapt its parameters at runtime.

We extended CrowdNav to support, besides the adaptive service, also a static service, as well as different ways of managing junctions, in line with the norms described in Sect. 2. We use the two contextual properties *Time*, which can assume the values *day* (600 cars in the simulation) or *night* (300 cars), and *Weather*, which can be either *normal* or *extreme* (the maximum allowed speed is reduced by 25% in all the streets). We instrumented the extended CrowdNav to collect data about norm satisfaction and objectives achievement. The boolean value of the objective *Travel_Time* is obtained, every simulation-day, by determining whether, on average, the cars in the city took less than 2.5 times the optimal trip time[3] to reach their destination. The boolean value of the objective

[3] Actual trip time over the theoretical time w.r.t. to length and speed limits.

Accidents is obtained by determining whether there are less than 4 accidents per day.

5.1 Experiments

We implemented the runtime norm supervision mechanism as a modified version of hill climbing that accounts for t_{oa} as a stopping criterion: it stops only when either all the configurations have been tried or it finds a local optimum with an average objectives achievement probability higher than t_{oa}.

We used the algorithms pureBN and stateBased of Sect. 4 as two possible heuristics for defining the neighborhood of a configuration: the neighborhood defined by pureBN and stateBased is the set of all the configurations that satisfy their suggestions. We defined three additional configuration neighborhood metrics that do not take into account the acquired knowledge about the effectiveness of the enforced norms. (1) maximum distance 4 (MD4) includes in the neighbourhood all the configurations that are obtained by revising at most 4 norms[4]; (2) Maximum size 10 (NMS10) and (3) maximum size 20 (NMS20) define a neighborhood that includes the 10 and 20 closest configurations to the current one, respectively. We tested the hill climbing implementation with the 3 uninformed neighborhoods and we compared them with our proposals employing the pureBN (HCPB) and the stateBased (HCSB) suggestion algorithms.

In order to determine the average quality in terms of objectives achievement, each algorithm has been executed starting from all of the possible configurations. The system has $12^4 = 20,736$ possible configurations (12 possible norm sets enforceable in any of the 4 contexts). Since in the worst case every algorithm may need to try all the configurations before stopping, to limit the experimentation time, we reduced the data set to 81 configurations via test case generation techniques[5]. We introduced then three additional configurations more distant from the others. Two of them are the best-scoring configurations. We defined experiments for three different thresholds t_{oa}, based on the distribution of values in the 84 tested configurations (see Fig. 4). $T1 = 0.5$, accepts as final configurations only the two best-scoring configurations in group A (2.4% of the total), which are significantly different (and therefore more distant) from the others; $T2 = 0.4$ accepts the configurations in groups A and B (9.5% of the total); and $T3 = 0.37$ accepts configurations in groups A, B and C (17.8% of the total).

We compare the results in terms of (i) *convergence speed*: the number of steps attempted before stopping; and (ii) *solution quality*: the average objectives achievement probability of the final solution.

[4] Revising one norm leads to a distance of 2–3 from the original configuration, and each configuration has 10%–20% of all configurations in its neighborhood.

[5] We obtained 12 variants from *pairwise testing* with variables: time (day, night), weather (normal, extreme), CNS (none, adaptive, static), and junctions (none, adaptive lights, static lights, priority lanes). We grouped those variants in 4 groups (one per context) and we generated all their combinations to obtain 81 configurations.

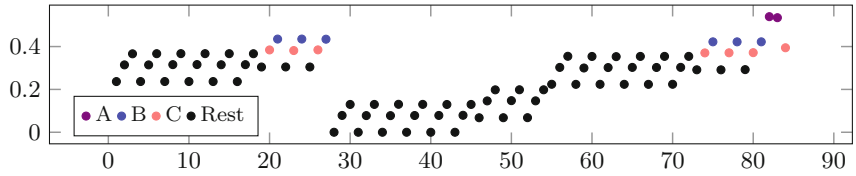

Fig. 4. Avg. probability of objectives achievement for the 84 tried configurations.

5.2 Analysis of the Results

Table 1 compares the 5 different tested algorithms. With threshold T1 ($t_{oa} = 0.5$), we tested a scenario where the (few) optimal configurations are slightly more distant from the others (while the median distance between the 82 suboptimal configurations is 6.3, the median distance with the remaining two is 7).

Table 1. Comparison of the algorithms with thresholds T1, T2 and T3. Values of Steps and Final columns are average values over the 84 different simulations.

Algo	T1 ($t_{ga} = 0.5$, 2 optimal conf.)			T2 ($t_{ga} = 0.4$, 8 optimal conf.)			T3 ($t_{ga} = 0.37$, 15 optimal conf.)		
	Steps (σ)	Final (σ)	Opt.	Steps (σ)	Final (σ)	Opt.	Steps (σ)	Final (σ)	Opt.
MD4	68.94 (21.47)	0.54 (0.00)	100%	24.48 (12.56)	0.45 (0.04)	100%	17.54 (9.56)	0.43 (0.05)	100%
ML10	88.54 (15.15)	0.54 (0.00)	100%	18.80 (10.62)	0.43 (0.02)	100%	11.15 (6.15)	0.40 (0.03)	100%
ML20	73.81 (17.93)	0.54 (0.00)	100%	23.80 (11.44)	0.44 (0.03)	100%	17.94 (9.08)	0.42 (0.04)	100%
HCPB	**64.86 (27.48)**	**0.54 (0.00)**	**100%**	**11.90 (8.04)**	**0.43 (0.02)**	**100%**	**2.99 (3.03)**	**0.40 (0.03)**	**100%**
HCSB	**79.70 (22.05)**	**0.54 (0.00)**	**100%**	**5.10 (3.50)**	**0.43 (0.02)**	**100%**	**0.82 (0.39)**	**0.40 (0.03)**	**100%**

In terms of convergence speed, with T1 our heuristics behave similarly to the others. On average, uninformed heuristics take around 77.08 steps, while HCPB and HCSB require 72.28 steps. Since all the algorithms give priority to the closest configurations in the neighborhood, they need to explore big part of the solution space before finding the 2 optimal configurations, which are more distant. HCPB slightly outperforms all the other heuristics. The reason is that its strategy is to suggest a revision for all the active norms in the current configuration. This strategy, compared to the others, accomodates the exploration of more diverse configurations (i.e., HCPB defines a more heterogeneous neighborhood), thereby favouring the discovery of the optimal ones.

With less demanding thresholds (T2 and T3), our algorithms significantly outperform the others. For T2, our heuristics HCPB and HCSB offer, on average, an improvement of $61.9\% = 1-(8.5/22.36)$ over the tested uninformed heuristics in terms of convergence speed, while for T3 the efficiency gain is $87.7\% = 1-(1.9/15.54)$. In particular, with T3, HCSB requires on average only 0.82 steps, i.e., it finds an optimal configuration after only one revision. Concerning solution quality, the results are comparable for all the algorithms in all the experiments.

These preliminary results, which shall be confirmed on other cases, support our hypothesis that heuristics that leverage knowledge about norm satisfaction

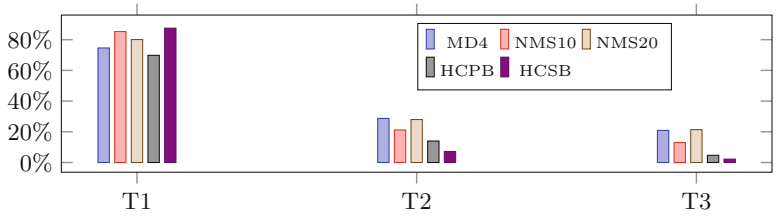

Fig. 5. Average percentage of explored configurations before finding an optimal one. 100% means trying, on average, all the 84 configurations.

allow efficient runtime norm revision, by altering a suboptimal norm set into an optimal one with only few revisions. Results concerning T1 suggest improvements of our algorithms in order to consider more exploration when a configuration is not found quickly. Figure 5 summarizes the convergence speed results.

It is finally worth to note that, in this paper, the selection of a norm set that satisfies the given suggestions depends on the number of available norm sets (only 12 in our experiments). However, thanks to the use of a Bayesian Network and to the concept of norm revision, the Revision Engine also provides information about the effectiveness of the norms w.r.t. the achievement of the overall objectives. Differently from classical search algorithms, therefore, our suggestion algorithms also provide direct information about how to revise the norms, supporting targeted human or automated intervention on the norms.

6 Related Work

Several papers have focused on deciding and proving the correctness of normative systems by model checking formulas describing desired properties such as liveness or safety properties [12–14]. Despite their elegance, these approaches do not fully cope with the dynamicity of today's complex systems. Recently, frameworks emerged to cope with norms dynamics and their impact on system specification.

Knobbout *et al.* [7] propose a dynamic logic to formally characterize the dynamics of state-based and action-based norms. Both in Knobbout's works [7,15] and in Alechina *et al.*'s [14], norm change is intended as norm addition. In this paper we investigated further types of norms revision that could be used to extend such framework for dynamic normative systems.

Aucher *et al.* [16] introduce a dynamic context logic to describe the operations of contraction and expansion of theories by introducing or removing new rules. Governatori *et al.* [17] investigate from a legal point of view the application of theory revision to reason about legal abrogations and annulments. The types of revision presented in this paper can be related to theory revision. However we have taken a multiagent systems standpoint, in which norm revision should be studied in terms of its impact on the overall system behavior. We leave for future work the study of the impact of a revision on the existing normative system.

Norms update has also been studied from the perspective of approximation [18], where an approximated version of a norm is obtained to cope with imperfect monitors for the original norm. The concept of approximation is similar to our notion of relaxation, however it is defined with respect to a specific monitor: an approximated norm is synthesized from the original one to maximize the violations detectable by the available imperfect monitor. In this paper we assumed perfect monitors and we focused instead on the effectiveness of norm enforcement to develop algorithms that suggest a norm revision when needed.

Cranefield *et al.* [19] present a Bayesian approach to norm identification. They show that agents can internalize norms that are already present in an environment, by learning from both norm compliant and norm violating behaviours. In this paper, instead of focusing on the agents, we used a Bayesian Network to collect information about norms effectiveness, regardless of their internalization in the agents. We used then the acquired knowledge to develop strategies for the suggestion of revisions of the norms that are enforced in the system.

7 Conclusion and Future Work

The complexity and unpredictability of modern multiagent systems allows only partial and incomplete domain knowledge at design-time. We proposed a runtime supervision mechanism to automatically revise the norms enforced in a multiagent system. The mechanism employs a Bayesian Network to collect data about norms satisfaction, and to learn their relationship with objectives achievement in different contexts. Informed by such data, the supervisor performs norm revision based on the revision suggested by the pureBN and stateBased algorithms.

We implemented the supervision mechanism as a variant of the hill climbing optimization technique. Such variant always guarantees to find, if it exists, a norm set that ensures the overall system objectives. We evaluated our implementation in terms of convergence speed and quality of the final enforced norm set. We used an urban traffic simulation to compare the results that can be obtained by taking into account the knowledge learned at runtime about norms effectiveness, with results that can be obtained without such knowledge. The results show that the our heuristics outperform the tested uninformed ones by identifying the optimal solutions in significantly less number of revisions.

Future work will focus on four main directions. *Evaluation*: the scalability and computational complexity of the approach must be properly evaluated: the conditional probability tables of the objective nodes in the Bayesian Network grow exponentially with the number of norms. The structure of the *Norm Bayesian Network* does not leverage the conditional independence properties that may exist between different norms. In case of large sets of norms, the use of such network may become intractable. A solution is to use, when building the network, a model representing the hierarchy between norms (e.g., [20]). *Refined revision algorithms* shall be developed with a memory that provides information about the effectiveness of previous revisions; possible techniques include Q-Learning [21] and Dynamic Decision Networks [8]. Additional types of analysis of the Bayesian Network, such as sensitivity analysis [22], should be studied

to help select a new norm set. It is possible to extend the current structure of the *Norm Bayesian Network* to support imperfect monitoring and uncertainty, through the introduction of an additional layer of nodes and a sensor model [8]. *Norm synthesis*, i.e., the automated generation of norms that can regulate the multiagent system by preventing harmful behaviors and by promoting useful behaviors, based on the revisions suggested and on agents preferences and their relationship with the system objectives. *Sanctions and their revision* can be studied as a way to influence the behavior of agents that goes beyond norm relaxation or strengthening.

References

1. Wooldridge, M.J.: An Introduction to MultiAgent Systems, 2nd edn. Wiley, Hoboken (2009)
2. Bulling, N., Dastani, M.: Norm-based mechanism design. Artif. Intell. **239**, 97–142 (2016)
3. Testerink, B., Dastani, M., Bulling, N.: Distributed controllers for norm enforcement. In: Proceedings of ECAI, pp. 751–759 (2016)
4. Alechina, N., Bulling, N., Dastani, M., Logan, B.: Practical run-time norm enforcement with bounded lookahead. In: Proceedings of AAMAS, pp. 443–451 (2015)
5. Letier, E., Van Lamsweerde, A.: Reasoning about partial goal satisfaction for requirements and design engineering. In: ACM SIGSOFT Software Engineering Notes, vol. 29, pp. 53–62. ACM (2004)
6. Ali, R., Dalpiaz, F., Giorgini, P., Souza, V.E.S.: Requirements evolution: from assumptions to reality. In: Halpin, T., et al. (eds.) BPMDS/EMMSAD. LNBIP, vol. 81, pp. 372–382. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21759-3_27
7. Knobbout, M., Dastani, M., Meyer, J.C.: A dynamic logic of norm change. In: Proceedings of ECAI, pp. 886–894 (2016)
8. Russell, S.J., Norvig, P.: Artificial Intelligence - A Modern Approach, 3rd internat. edn. Pearson Education (2010)
9. Spiegelhalter, D.J., Dawid, A.P., Lauritzen, S.L., Cowell, R.G.: Bayesian analysis in expert systems. Stat. Sci. **8**(3), 219–247 (1993)
10. Kwisthout, J.: Most probable explanations in bayesian networks: complexity and tractability. Int. J. Approx. Reason. **52**(9), 1452–1469 (2011)
11. Schmid, S., Gerostathopoulos, I., Prehofer, C., Bures, T.: Self-adaptation based on big data analytics: a model problem and tool. In: Proceedings of SEAMS, pp. 102–108 (2017)
12. Dastani, M., Grossi, D., Meyer, J.-J.C., Tinnemeier, N.: Normative multi-agent programs and their logics. In: Meyer, J.-J.C., Broersen, J. (eds.) KRAMAS 2008. LNCS (LNAI), vol. 5605, pp. 16–31. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-05301-6_2
13. Knobbout, M., Dastani, M.: Reasoning under compliance assumptions in normative multiagent systems. In: Proceedings of AAMAS, pp. 331–340 (2012)
14. Alechina, N., Dastani, M., Logan, B.: Reasoning about normative update. In: Proceedings of IJCAI, pp. 20–26 (2013)
15. Knobbout, M., Dastani, M., Meyer, J.-J.C.: Reasoning about dynamic normative systems. In: Fermé, E., Leite, J. (eds.) JELIA 2014. LNCS (LNAI), vol. 8761, pp. 628–636. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11558-0_46

16. Aucher, G., Grossi, D., Herzig, A., Lorini, E.: Dynamic context logic. In: He, X., Horty, J., Pacuit, E. (eds.) LORI 2009. LNCS (LNAI), vol. 5834, pp. 15–26. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04893-7_2
17. Governatori, G., Rotolo, A.: Changing legal systems: legal abrogations and annulments in defeasible logic. Log. J. IGPL **18**(1), 157–194 (2010)
18. Alechina, N., Dastani, M., Logan, B.: Norm approximation for imperfect monitors. In: Proceedings of AAMAS, pp. 117–124 (2014)
19. Cranefield, S., Meneguzzi, F., Oren, N., Savarimuthu, B.T.R.: A Bayesian approach to norm identification. In: Proceedings of ECAI, pp. 622–629 (2016)
20. Dell'Anna, D., Dalpiaz, F., Dastani, M.: Validating goal models via Bayesian networks. In: Proceedings of AIRE@RE (2018)
21. Rummery, G.A., Niranjan, M.: On-line Q-learning using connectionist systems. Technical report CUED/F-INFENG/TR-166, vol. 37. University of Cambridge, Cambridge (1994)
22. van der Gaag, L.C., Renooij, S., Coupé, V.M.: Sensitivity analysis of probabilistic networks. In: Lucas, P., Gámez, J.A., Salmerón, A. (eds.) Advances in Probabilistic Graphical Models. STUDFUZZ, pp. 103–124. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-68996-6_5