Run-Time Norms Synthesis
in Multi-objective Multi-agent Systems

Maha Riad[(✉)] and Fatemeh Golpayegani

School of Computer Science, University College Dublin, Dublin, Ireland
maha.riad@ucdconnect.ie, fatemeh.golpayegani@ucd.ie

Abstract. Norms represent behavioural aspects that are encouraged by a social group of agents or the majority of agents in a system. Normative systems enable coordinating synthesised norms of heterogeneous agents in complex multi-agent systems autonomously. In real applications, agents have multiple objectives that may contradict each other or contradict the synthesised norms. Therefore, agents need a mechanism to understand the impact of a suggested norm on their objectives and decide whether or not to adopt it. To address these challenges, a utility-based norm synthesis (UNS) model is proposed which allows the agents to coordinate their behaviour while achieving their conflicting objectives. UNS proposes a utility-based case-based reasoning technique, using case-based reasoning for run-time norm synthesising in a centralised approach, and a utility function derived from the objectives of the system and its operating agents to decide whether or not to adopt a norm. The model is evaluated using a traffic junction scenario and the results show its efficacy to optimise multiple objectives while adopting synthesised norms.

Keywords: Norms synthesis · Multi-objective · Heterogeneous multi-agent systems

1 Introduction

Multi-agent systems (MAS) are complex systems consisting of agents which are autonomous entities with their own objectives, and can act dynamically. Agents' objectives can be represented by tasks they want to achieve, these tasks can be unintentionally supportive to other agents' objectives or incompatible with them [8]. Aside from the ability of the agents to have multiple objectives, agents may have heterogeneous types, in which each type has its own characteristics, preferences or category [7]. Moreover, agents can operate in open system settings where they can move freely inside and outside of the system. MAS is applied in many real world applications such as traffic systems [1,7], computer networks [4], smart energy grids [9] and the internet of things systems [13]. However, in such systems, it is not only crucial to model the heterogeneity, openness and autonomy of the agents, but also it is essential to consider the agents' behaviour coordination.

Norms are behaviour guidelines imposed by a society or social group to regulate agents' actions. For example, in a traffic system, one norm is to slow down when seeing a senior driver because he might be more cautious than other drivers and drive slowly. Another example is represented in the norm of leaving the right (fast) lane empty when there is an ambulance. Accordingly, norms representation helps agents to achieve their objectives in an acceptable manner within their social groups without compromising their autonomy. This would facilitate group decision making, cooperation, and coordination between agents [12].

Multi-agent systems that encapsulate norms concepts such as prohibitions, obligations and permissions are called Normative Multi-Agent Systems (NorMAS) [2]. NorMAS rely on norms for regulating the behaviour of agents while reserving their autonomy property [2]. Norms have dynamic nature, and so each norm's life cycle begins with norm synthesis (which relies on creating and composing a set of norms [11]) and ends with norm disappearance [6].

Various efforts of researchers were directed to proposing a reliable norm synthesising mechanisms that can be used to synthesise norms at run-time and/or in an open system. The challenge of an open environment is that agents can enter and leave the system freely, and so a special technique is needed for aligning all of the agents with the system norms, particularly for the new agents entering the system. Moreover, synthesising norms at run-time would demand an online strategy for triggering new norms creation and update according to the changing environment. IRON [11], a state-of-the-artwork, was one of the most prominent mechanisms that showed its efficacy in synthesising norms at run-time in an open NorMAS. However, it has two main limitations. First, the synthesising strategy used may produce biased norms. For example, in a traffic junction scenario, IRON can synthesise a norm that obligates the driver to stop when he is at an intersection and there is another vehicle to his right trying to cross at the same time. Although this norm will ensure avoiding the collision of the vehicles, this will cause the left lane to have higher congestion and traffic density than the right lane because vehicles in the right lane would have higher priority to pass. Second, IRON does not consider whether the synthesised norms contradict the objectives of the system or other norms or not. If the norm synthesised in the previous traffic junction example is applied while having an emergency vehicle (ambulance) in the left lane, it will be against the system' objectives; if it aims to minimise the total waiting time of emergency vehicles. An example of contradicting norms is seen when it is a norm that a driver drives at an average or slow speed when having a child on board, and also the same car might drive too fast when the child has an emergency. So, in this case, two different unmatchable norms appear: (i) a car drives slowly if a child is on board, and (ii) a car drives too fast in case of having an emergency case.

In this paper, we overcome the limitations of IRON and the other related work by proposing UNS, a utility-based norm synthesise model. UNS coordinates norms and objectives, handle unmatchable norms, and support fairer technique of norm synthesising. In UNS, a utility-based case-based reasoning technique is proposed to facilitate the coordination of norms and objectives of

agents and the system. UNS uses the case-based reasoning algorithm to synthesise norms. The utility function determines the necessity of norms adoption and elicits the suitable norm when there are unmatchable norms. Two norms are called unmatchable norms, when only one of them should be applied at the same time and context. For example, consider norm n_a, which suggests to stop if there is a car on the left side of a junction, and norm n_b, recommending to stop if there is a car on the right side of a junction. Although by applying these two norms a collision would be avoided, a deadlock situation will be created as well. The utility function is constructed based on the objectives of the system to ensure that they are considered in the process of norms reasoning. UNS is evaluated using a simulated traffic scenario in SUMO and results show the system's capability of synthesising and reasoning norms at run-time while reaching the system's objectives.

The remainder of this paper is as follows: Sect. 2 covers the related work and the essential state-of-the-art work (IRON) needed to understand our model. The problem statement is defined and formulated in Sect. 3. In Sect. 4, the proposed model (UNS) is illustrated, and then it is empirically evaluated in Sect. 5. Finally, in Sect. 6, the conclusion and future work are elaborated.

2   Related Work

Synthesising norms is more challenging in open and run-time NorMAS. In open systems, the challenge is to transfer norms to new agents entering the system and make use of the norms adopted by other agents before leaving the system. Mahmoud *et al.* [10] address this challenge by proposing a potential norms detection technique (PNDT) for norms detection by visitor agents in open MAS. They implemented an imitating mechanism which is triggered if the visitor agents, who are monitoring the norms of the host agents, discovered that their norms are in-compliant with the norms of the other host agents. However, PNDT technique used a fixed set of norms, which are commonly practised by the domain, ignoring the dynamic nature of norms.

In run-time NorMAS, it is challenging to define a dynamic set of norms and initiate it. Moreover, real run-time applications would not only demand synthesising new norms but would also require handling the whole norms lifecycle including norms refinement and disappearance. One of the efforts directed towards run-time norms revision was carried out in [3], in which a supervision mechanism for run-time norms revision was proposed, addressing the challenge of norms modification when weather changes or when accidents happen. However, the norms revision mechanism is developed using a primary defined pool of norms and situations. In the revision process, the model just substitutes the norms depending on the situation; limiting the norms to a static set of norms. Accordingly, the dynamism is in altering the chosen norms set depending on an optimisation mechanism constructed based on the system's objectives and does not handle the changes and evolution of the norms. In [5], Edenhofer *et al.*. present a mechanism for dynamic online norm adaption in a heterogeneous distributed multi-agent system for handling colluding attacks from agents with bad

behaviour. The agents interact together and build a trust metric to represent the reputation of the other agents. The main focus of this paper is identifying the bad agents and showing that using norms improves the system's robustness. Although this work is based on an open, heterogeneous and distributed environment, it does not identify how norms can be revised and updated in this context.

IRON machine was developed by Morales *et al.* and presented in [11]. It addresses the limitations of the other previously mentioned works, as the main aim of IRON is to synthesise norms online using an effective mechanism that not only synthesises norms in run-time but also revises these synthesised norms according to their effectiveness and necessity and further dismisses the inefficient norms. IRON simulates multi-agent systems, in which norms are synthesised for coordinating the behaviour of agents, and handles conflicting situations that can occur, such as collisions of vehicles in a traffic scenario. As presented in [11], IRON is capable of run-time norm synthesising and addressing the issues of using static norms, however, the idea of coordinating norms and objectives is not addressed.

Accordingly in this paper, we will propose UNS which is not only responsible for online-norm synthesis in open multi-agent systems but also guarantees objective consideration in the process of norm reasoning by the aid of utility-based technique.

As this work represents a series of the closest and comprehensive efforts exerted towards online norm synthesis for MAS, in the following sub-section an extended elaboration for IRON strategy and algorithm will take place and will be further used as the baseline of our model.

2.1 Intelligent Robust On-line Norm Synthesis Machine (IRON)

IRON machine is composed of a central unit that is responsible for detection of conflicts, synthesises of new norms to avoid conflicts, evaluation of the synthesised norms, refinement of norms, and announcement of the norm set to the agents. To simplify the illustration of the responsibilities of IRON, we will use a traffic junction example with two orthogonal roads scenario. The vehicles represent the agents, each occupying a single cell and moving in a specific direction per time-step.

– In **conflicts detection**, conflicts are detected when a collision occurs between two or more vehicles. The occurrence of a collision will trigger IRON to synthesise new norm to avoid future collisions of similar cases. As for **norms synthesising**, norms are created based on a case-based reasoning algorithm. In the algorithm, the conflicting situation at time t is compared to the conflicting vehicles' context at time $t - 1$. Then a norm is created using the conflicting views as a precondition for applying the norm and prohibiting the 'Go' action in this context. The synthesised norm is then added to a norms set and communicated to the agents (vehicles) of the system. For example, in Fig. 2, if vehicle A and B collided at the intersection (grey cell) then the context and action of

A or B is chosen randomly by the system to create a new norm. If A is chosen the new norm will be $n = if(left(<), front(-), right(<)) \longrightarrow proh(\text{`Go'})$. The $left()$ attribute in the precondition of the norm stores the direction of the left neighbour vehicle of vehicle A. While the $right()$ attribute stores the direction of the right vehicle to vehicle A, which is in this case vehicle B. Similarly, the $front()$ attribute would store the direction of the front vehicle, however, because there is nothing in front of the vehicle the symbol $(-)$ is used.

– **Norms Evaluation** is carried out by measuring necessity and effectiveness of a norm and comparing it to a threshold. Necessity is measured according to the ratio of harmful violated norms, which are norms that resulted in conflicts when violated, compared to the total number of violated norms. The methodology used in the calculations is akin of reinforcement learning, in which the norm's necessity reward NNR is calculated by:

$$NNR = \frac{m_{V_C}(n) \times w_{V_C}}{m_{V_C}(n) \times w_{V_C} + m_{V_{\bar{C}}}(n) \times w_{V_{\bar{C}}}} \tag{1}$$

$m_{V_C}(n)$: Number of violations which led to conflicts
w_{V_C}: Weight that measure the importance of harmful applications
$m_{V_{\bar{C}}}(n)$: Number of violations which did not led to conflicts
$w_{V_{\bar{C}}}$: Weight that measure the importance of harmless applications
The effectiveness of norms is measured based on the extent to which the norm is successful (i.e. which resulted in the minimum number of conflicts). The norm's effectiveness reward NER is calculated by:

$$NER = \frac{m_{A_C}(n) \times w_{A_C}}{m_{A_C}(n) \times w_{A_C} + m_{A_{\bar{C}}}(n) \times w_{A_{\bar{C}}}} \tag{2}$$

$m_{A_C}(n)$: Number of applied norms which led to conflicts
w_{A_C}: Weight that measure the importance of unsuccessful applications
$m_{A_{\bar{C}}}(n)$: Number of applied norms which did not led to conflicts
$w_{A_{\bar{C}}}$: Weight that measure the importance of successful applications

– **Norms refinement** is carried out by generalisation or specialisation of norms. Norms are mapped in a connected graph that expresses the relationships between them. In other words, the graph shows the child and parent norms and their links. Norms generalisation is applied when two or more norms have acceptable necessity and effectiveness results compared to a threshold, which is primarily specified before the system run, for time-interval T. Specialisation or deactivation of norms is conducted when the effectiveness and necessity of the norm or its children have been below the threshold for time-interval T.
– **Norms communication** is the final step, in which the norms are communicated to the agents.

The main flow of activities that are carried out in the scenario of the traffic junction (similar to Fig. 2), is as follows. Vehicles (agents) movements take place

per time-step, however, prior to these movements, the vehicles check the norms set for applicable norms. Applicable norms are norms with preconditions that matches the context (local view) of the agents. When a new collision is detected, a random agent/vehicle is chosen and then its context is added as a precondition of a new norm that prohibits the 'GO' action. Afterwards, this norm is added to the norms set (initially empty). In addition, norms evaluation and refinement are carried out per time-step, in which all the views at time-step t are revised to determine the set of applicable norms for each of the views. The retrieved set of applicable norms is divided into four subsequent sets: (i) applied norms that led to conflicts (ii) applied norms that did not lead to conflicts (iii) violated norms that led to conflicts (iv) violated norms that did not lead to conflicts. Then set (i) and (ii) are used to calculate the effectiveness of each of the norms, while set (iii) and set (iv) represent the main inputs for the necessity calculation. Finally, norms refinement is conducted.

3   Problem Statement

Let us consider a norm-aware multi-objective multi-agent system that is composed of a finite set of mobile agents as $Ag = \{ag_1, ag_2, ..., ag_n\}$. Each agent ag_i has a type t_{ag_i}, set of properties P_{ag_i}, set of objectives O_{ag_i} and set of adopted norms N_{ag_i}. In addition, the system itself has its own set of objectives O_s and set of norms N_s, where $O_{ag_i} \subseteq O_s$ and $N_{ag_i} \subseteq N_s$.

The norms are created by a centralised unit in the system in the form of a pair $(\alpha, \theta(ac))$ and then messaged to the agents. α represents a precondition for triggering the norm applicability. This precondition reflects a specific context of the agent co_{ag_i}, which is the local view of the agent ag_i that defines its direct neighbours $Ng_{ag_i} = \{ag_1, ag_2, ..., ag_k\}$ and their properties such as their moving direction in the traffic scenario example. So, $co_{ag_i} = \{P_{ag_k} : ag_k \subseteq Ng_{ag_i}\}$. θ symbolises a deontic operator (obligation, prohibition or permission) with a specific action ac_{ag_i} of agent ag_i which will apply the norm. For example, if an action is beneficial for an agent then it is obligated and if an action is harmful it is prohibited.

The central unit synthesises new norms after a conflicting state c arises between agents and uses the synthesised norm in future similar cases to avoid conflicts. Conflicting state c belongs to set of conflicts C, a conflict is considered detected when two agents or more carry out actions that result in a problem. The norm is synthesised by comparing the view at conflicting situation at time-step t, V_t to the view before the conflict occurrence V_{t-1}. The series of views that represent different situations at each time-step are added in a ViewTransition V set (i.e. $V_t \in V$ and $V_{t-1} \in V$).

In such a system, there are three main problems to be tackled. First, the process of synthesising norms should ensure fairness (i.e. created norms cannot be biased towards specific agents' situation). For example, if there is a norm created to coordinate the behaviour of two vehicles ag_1 and ag_2 in an intersection, this norm cannot always give priority to the vehicles on the right, because this

will make the vehicles in the left lane always delayed. Second, when there is more than one applicable norm in the same context, often unmatchable ones, only one should be applied to avoid a deadlock situation. For example, in a scenario of vehicles crossing a junction, if there were two norms created: n_1 for stopping if there was a vehicle on the right, and $ń_1$ for stopping if there is a vehicle on the left, a decision should be made to apply one of these unmatchable norms only. Third, the agent's norms N_{ag_i} and objectives O_{ag_i} should be coordinated to ensure that the norms' compliance does not contradict reaching the objectives.

4   UNS: Utility-Based Norm Synthesis Model

UNS is a utility-based norm synthesis mechanism implemented in a normative, open, run-time, multi-objective, multi-agent system. UNS aims at reaching three main goals. First, to synthesise norms while supporting fairness during norm creation. Second, to handle unmatchable synthesised norms. Finally, to coordinate the objectives of agents with the synthesised norms. Figure 1 shows the architecture of UNS. It shows the five main responsibilities of UNS that take place per time-step at run-time: conflicts detection, norms synthesising, norms reasoning, norms evaluation and refinement.
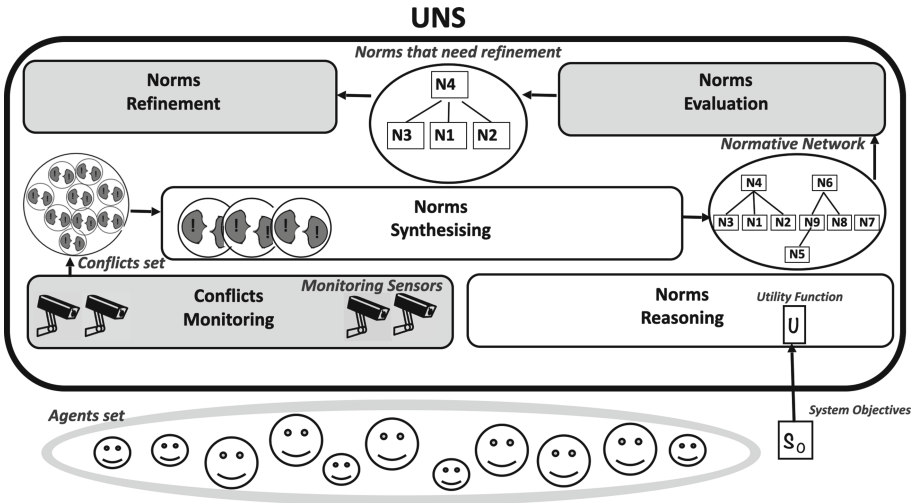

Fig. 1. Utility-based norm synthesis model architecture (components coloured in grey are inherited from IRON) (Color figure online)

Conflicts detection, norms evaluation and refinement are inherited from IRON and integrated in UNS. The details of the steps carried out by UNS are as follows:

4.1 Conflicts Detection

At each time-step t as agents take actions, a set of monitors (e.g. traffic cameras) $M = \{m_1, m_2, ..., m_n\}$ monitor these actions to detect any conflicts. A conflict c is detected when more than one agent actions contradict at the same view v_i, where $v_i \in V$. For example, in a traffic system, if vehicles standing before a junction in opposite directions decided to move (do a 'Go action') towards the same position, a collision will occur and so a conflict will arise. To detect conflicts, views V are sent as a parameter to the $ConflictDection$ function (see Algorithm 1, line 6). A conflict object definition is composed of responsible agents $Ag^r \subseteq Ag$, context of these agents (which is the local views of each of these agents), and the views transition of a state s between time-step t and $t-1$, $(v_{s_{t-1}}^i, v_{s_t}^i)$.

4.2 Norms Synthesising

Case-based reasoning technique is used for norm synthesising. When a new conflict arises, a new case is created and then compared to similar cases and the best solution is chosen accordingly. In case that no similar case is found a new random solution is created for this case and added to the set of cases. In this manner, after conflicts are detected, UNS carries out the norms synthesising steps for each of these conflicts (see Algorithm 1, line 7 to line 17). All the agents responsible for the conflict are retrieved in Ag^r (e.g. all the vehicles that collided in the same intersection are considered as responsible agents). For each of these agents' context at $t-1$ if an applicable norm was not found (applicable norms are norms that have the same context as a pre-condition of the norm and the same agent action prohibited in the norm), a new norm creation process takes place (line 13). A new norm is composed of agents' context co_{ag_i} and prohibited action θac_{ag_i}. Getting the context of the agent at the previous time-step as a precondition of a norm and prohibiting the action that resulted in a conflict avoids future conflicts that might rise in similar situations. After the norm is created it is added to the system's norms set Ω (line 14).

UNS Supporting Fairness: In IRON norms synthesising was carried out by creating norms as a solution for only one randomly chosen agent from the agents involved in a conflict. However, in UNS we have proposed a norm synthesising process, which considers all the contexts of the agents involved in a conflict. For example, in IRON if two vehicles had a conflict in an intersection, the norm will be created based on prohibiting a Go action of only one of the two vehicles. Although this will decrease the probability of creating unmatchable norms, it will not ensure fairness as one side will always have priority of moving over the other side.

4.3 Norms Reasoning

The norm reasoning process must meet systems' objectives and handle unmatchable norms simultaneously. This is reached through defining a utility function U

Algorithm 1. UNS Strategy

1: **for each** t **do**
2:     **Input:** Ω, V
3:     **Output:** Ω
4: **/*Conflicts Detection*/**
5: //Inherited from IRON
6:     $conflicts \leftarrow ConflictDetection(V)$
7: **/*Norms Synthesis*/**
8:     **for each** $c \in conflicts$ **do**
9:         $Ag^r \leftarrow AgentsInConflictContexts(c)$
10:        **for each** $ag_i \in Ag^r$ **do**
11:            $co_{ag_i} \leftarrow GetAgentContext(ag_i)$
12:            **if** $hasApplicableNorm(co_{ag_i}) == false$ **then**
13:                $n \leftarrow CreateNorm(co_{ag_i}, \theta(ac_{ag_i}))$
14:                $\Omega \leftarrow \Omega \cup n$
15:            **end if**
16:        **end for**
17:    **end for**
18: **/*Norms Reasoning*/**
19:    **for each** $V_{t+1} \in V$ **do**
20:        $N_a \leftarrow GetApplicableNorms(V)$
21:        **if** $N_a.size > 1$ **then**
22:            $Utilities[] \leftarrow null$ //comment:$Utilities[k] = (ag_i, n, U_i)$
23:            **for each** $n \in N_a$ **do**
24:                $Utilities.add(calculateUtility(n), n)$
25:            **end for**
26:            $ag_x \leftarrow getAgent(max(Utilities))$
27:            $ag_x.applyApplicableNorm()$
28:        **end if**
29:    **end for**
30: **/*Norms Evaluation*/**
31: //Inherited from IRON
32: **/*Norms Refinement*/**
33: //Inherited from IRON
34: **end for**

---

that is constructed based on the system's objectives O_s and is used during the norm selection.

Utility Function Construction: In this paper, the utility function is constructed by adding the objectives with a maximisation function and subtracting the objectives with a minimisation function. For example, if O_s include two objectives $O_s = \{o_1, o_2\}$ and o_1 is to minimise all vehicles' average waiting time and o_2 is to minimise the average waiting time of emergency vehicles specifically, then the system utility function U will be defined as:

$$U = -o_1 - o_2 = -1 * (o_1 + o_2) \tag{3}$$

The utility function introduced can be considered as a type of unweighted additive utility function. In which using an additive approach is supported by the indifference property assumed between the objectives as all objectives need to be reached. Moreover, due to the equal preference to satisfying all of the objectives and eliminating any prioritisation no weights are needed. The general format of the defined utility function is:

$$U = \sum_{i=1}^{|X|} u(x_i) - \sum_{j=1}^{|M|} u(m_j) \tag{4}$$

The $|X|$ is the number of the system objectives that needs to be maximised, and $|M|$ is the number of the system objectives that needs to be minimised. $u(x_i)$ reflects the sub-utility gained from the maximisation of objective x_i, while $u(m_j)$ presents the sub-utility gained from the minimisation of objective m_j.

Accumulated Utility Calculation: At each time-step before the agents start moving (taking actions) UNS determines the set of applicable norms N_a in each view V_t (see Algorithm 1, line 20). If more than one norm is applicable for the same view V_t, then UNS carries out the steps in Algorithm 1 (from line 22 to line 27) to choose the norm with the highest utility and dismisses the rest of the norms. For example, if we have a traffic scenario as seen in Fig. 2, where vehicle A, ag_1, and vehicle B, ag_2, are willing to move to the same junction (coloured in grey) at time t, the stored view at time t will be represent by (V_t). UNS will retrieve the set of applicable norms $N_a = \{n1, \acute{n}1\}$, where $n1$: is to stop if there is a vehicle on the right and $\acute{n}1$ is to stop if there is a vehicle on the left. $n1$ is suggested for vehicle A and $\acute{n}1$ is suggested for vehicle B. If both vehicles apply the norms then none of them will move, which result in a deadlock state. So, a decision must be made to choose only one of the two unmatchable norms. Accordingly, an empty array of struct is initialised (in line 21). The struct is composed of ag_i (which is the responsible agent), n (which is the applicable norm for this agent ag_i situation), and U_i (which is the calculated utility gained by the system if this norm n is applied). For each of the applicable norms in N_a, the utility function is calculated (line 23). However, in our utility calculation strategy, we calculate an accumulated utility function, which does not only consider the utility gained by the agent applying the norm, but also considers all the agents that are indirectly affected by the norm adoption decision. For example, in Fig. 2, if vehicle A will be the agent that will apply the norm and will stop if there is a vehicle on the right, it will force vehicles C, D, E and F to stop as well. While if vehicle B decides to apply the norm $\acute{n}1$ and to stop, vehicle G will be forced to stop as well. Based on this justification, to ensure gaining the actual maximum utility, UNS aggregates the utility of all the agents that are affected directly and indirectly with the norms adoption or dismissal. Then, the norm that gives the maximum utility is applied (lines 26 and 27), and the rest of the norms are dismissed.
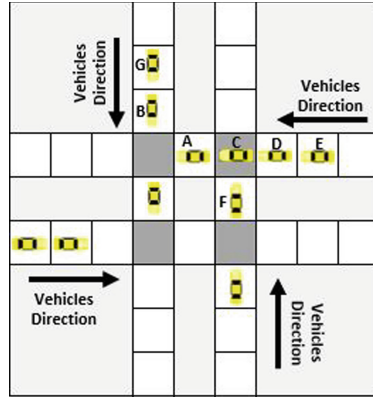
Fig. 2. A traffic junction composed of two orthogonal roads

4.4 Norms Evaluation and Refinement

The norm evaluation and refinement processes are inherited from IRON (illustrated in Sect. 2.1). These processes are used to evaluate norms at run-time using efficiency and necessity equations (Eqs. 1 and 2). If a norm's efficiency and necessity does not reach a certain threshold its refinement takes place and it is specialised or deactivated. Also, if a norm's efficiency and necessity exceeds a specified threshold it can be generalised.

5 Empirical Evaluation

In this section we show UNS capability to synthesise norms that support fairness, to handle unmatchable norms and to coordinate norms and objectives.

5.1 Empirical Settings

We simulate a traffic-based scenario, with a 19×19 grid as a road network with a junction of two orthogonal roads (see Fig. 2). Each road has two lanes; one for each direction. In Fig. 2, the cells coloured in grey show the four cells that represent the intersections. Vehicles are the agents and they have two main types: ordinary vehicles and high priority vehicles to represent heterogeneity. The ratio of generating priority vehicles to ordinary vehicles is 12:100 respectively. Also, as it is an open MAS, vehicles can enter and leave the road network freely. Vehicles move per time-step aiming to reach their final destination which was randomly generated by the simulator at the beginning of the trip of the vehicle. In each time-step, the system randomly chooses the number of new vehicles (between 2 to 8 vehicles) to be emitted to start their trip. The system aims at avoiding conflicts (i.e., the collisions between vehicles) through the synthesised norms. Norms are defined as a pair that includes the agent context and the prohibited

action. The agent context is the local view of the vehicle describing the direction of vehicles on its left, front, and right, which we call neighbouring vehicles. For example, in Fig. 2 the vehicles in the local context of vehicle F are vehicles A, C and D. The action prohibited is a 'Go action' to avoid vehicle movement in future similar contexts. UNS synthesises norms and adds them to a norm set that is initially empty at the beginning of the simulation. When a stable normative system is reached the system converges. The system has two main objectives, minimising the average waiting time for all vehicles and minimising the total waiting time of priority vehicles. The utility function used in the norm reasoning is constructed based on the previous two objectives as follows:

$$-1 * (\frac{X_{wt} + Y_{wt}}{X + Y} + Y_{wt}) \tag{5}$$

X_{wt}: Total waiting time of ordinary vehicles
Y_{wt}: Total waiting time of priority vehicles
X: Number of ordinary vehicles
Y: Number of priority vehicles

5.2   Experiment Results

To evaluate UNS's performance, three main scenarios are tested with the settings illustrated in the previous sub-section with varying violation rate of norms, which represents the ratio of agents obeying the adoption of the norms. UNS will be compared to IRON machine (explained in Sect. 2.1). The average waiting time for all vehicles and the total waiting time of priority vehicles are reported to show the performance of UNS and IRON. Moreover, the number of collisions is used to reflect the efficiency of the synthesised norms in avoiding conflicts. We present the moving average of the results at every 50 time-steps obtained from 10 runs of simulation as plotted in Fig. 3, 4, and 5.

5.3   Scenario A (Violation Rate 10%)

Figure 3(a) shows the average waiting time of all vehicles in UNS compared to the average waiting time of all vehicles in IRON. The average waiting time is decreased in UNS, particularly from time-step 322. Moreover, it can be noted that from time-step 322 almost the average waiting time in UNS is constant with an average value of 1.5 time-steps. As results show, UNS has minimised the average waiting time of the vehicles and so fulfilling the first objective of the system.

Figure 3(b) shows the total time taken by priority vehicles per time-step in UNS compared to IRON. The average total waiting time of priority vehicles using UNS is 8.09 time-steps, while the average total waiting time of priority vehicles reached in IRON is 12 time-steps. Moreover, Fig. 3(a) and (b) do not only emphasise how UNS can coordinate objectives and norms, but the noticed stability and uniformity of the results show the reliability of UNS which is necessary in real-applications.
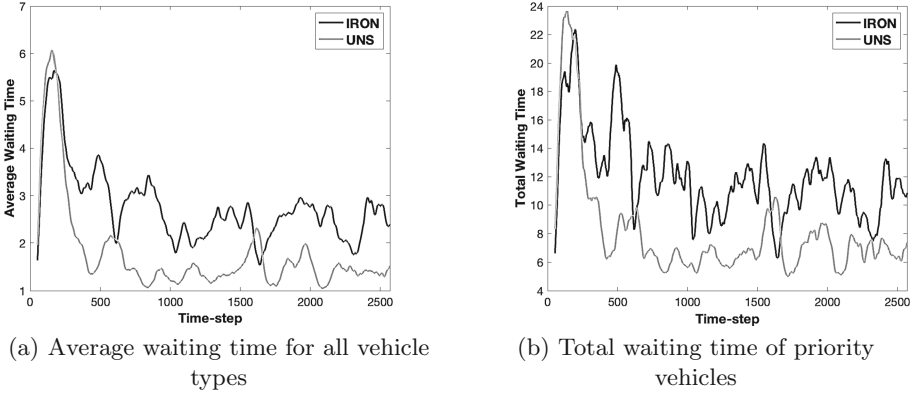
(a) Average waiting time for all vehicle types

(b) Total waiting time of priority vehicles

Fig. 3. Scenario A

(a) Average waiting time for all vehicle types

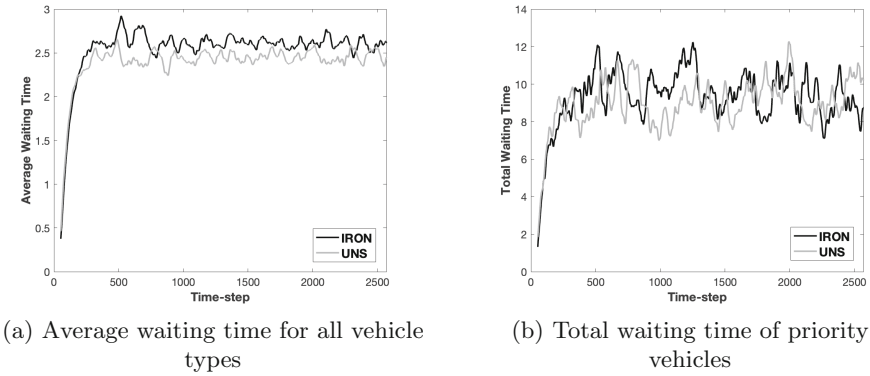(b) Total waiting time of priority vehicles

Fig. 4. Scenario B

Figure 5(a) presents the total number of collisions per time-step, which shows UNS is able to successfully synthesise norms at run-time to handle collisions. The results show UNS outperforms the synthesised norms set in IRON. Furthermore, observations showed that in a lot of time-steps UNS reached zero collisions, unlike IRON. The average number of collisions in UNS is 0.08 while the average number of collisions in IRON is 0.17. Also, comparing the total number of collisions, the total number of collisions in UNS is 51% lower than IRON, which shows the efficacy of the norm synthesis process.

5.4   Scenario B (Violation Rate 70%)

Figure 4(a) shows the average waiting time of all vehicles in UNS compared to IRON. The average waiting time in this scenario is increased to 2.45 time-steps compared to scenario A. However, UNS still outperforms IRON, in which its

average waiting time per time-step decreased from 2.75 to 2.53 time-steps. This unexplained decrease in IRON shows the essence of the primary definition of the system objectives and its incorporation in the model. Moreover, the results show that even with a high violation rate the system objectives can be achieved using UNS.

Figure 4(b) shows that UNS and IRON have quite similar range of total waiting time for priority vehicles. However, UNS outperforms IRON as the average of the total waiting time of priority vehicles is 8.92 time-steps using UNS and 9.24 time-step using IRON.

The results also show that although the violation rate has increased by 60% compared to scenario A, the average total waiting time of priority vehicles in UNS has only increased by 9.30%. Furthermore, the number of collisions occurred in this scenario using UNS is 4.66% fewer compared to IRON as seen in Fig. 5(b).
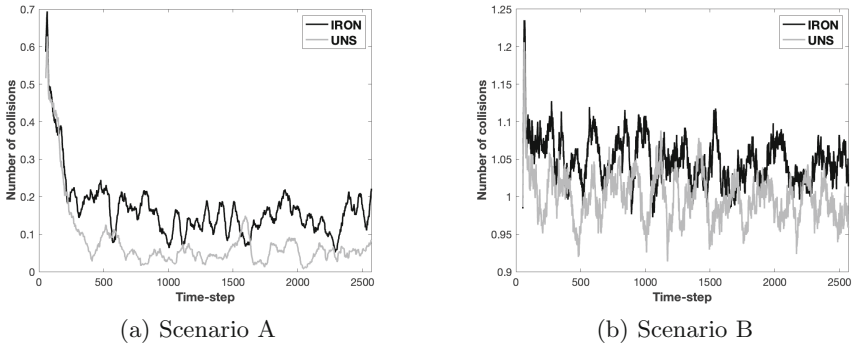

(a) Scenario A (b) Scenario B

Fig. 5. Total number of collisions

5.5 Scenario C (Violation Rate 0%)

When using 0% violation rate with IRON simulation, IRON is not able to converge and continue the simulation. The reason behind this is that the system reaches a deadlock when all vehicles obey to the norms. Although, IRON strategy in synthesising norms relies on creating only one norm at a time, it might synthesises two unmatchable norms at different instances that when applied in the same conflict causes a deadlock. For example, if one norm is to stop if there is a vehicle on the right hand side and the second norm is to stop when there is a vehicle on the left, two lanes of the vehicles standing at the beginning of a junction will stop endlessly, when there is no violation. However, this situation does not arise in UNS because it handles unmatchable norms and if more than one norm is applicable, the utility for both norms is calculated and only one norm is applied (i.e. in the previous example, one vehicle will 'Stop' and the other will 'Go').

In all scenarios, UNS synthesises more norms than IRON. This is due to synthesising all the norms that would contribute in avoiding collision in a specific

situation, supporting the idea of fairness. For example, in one of the runs IRON synthesised 15 norms, while UNS synthesised 17 norms. For example, UNS synthesises norm n_a, $n_a = (left(-), front(-), right(<), Proh(Go))$ and norm n_b, $n_b = (left(>), front(-), right(-), Proh(Go))$, both contributing in avoiding a collision. However, IRON only synthesises n_b which will always give priority to vehicles on the right side of the intersection, and consequently cannot support fairness.

6   Conclusion and Future Work

In this paper, we proposed a centralised utility-based norm synthesis (UNS) model which aims at coordinating objectives of the system with the synthesised norms in real-time. Norms in UNS are created to resolve conflicts that occur between agents and they are synthesised using case-based reasoning technique. UNS uses a utility function constructed based on the system objectives for norm reasoning. This ensures that when agents come to applying the synthesised norms, unmatchable norms and coordinated objectives of the system are handled. In addition, to ensure the effectiveness of the synthesised normative system the norms evaluation and refinement technique is inherited from IRON strategy [11]. The model was evaluated using a traffic scenario of two intersecting roads and results were compared with IRON. Results showed the efficiency of the model to meet the objectives of the system while synthesising norms in real-time. As future work, in addition to applying the model on another application domains two main directions will be followed. First, to use a decentralised architecture that involves the coordination of the agents in the process of norm synthesis. This would facilitate building several sets of norms according to each agent group's learning and objectives. Second, to transfer the norm reasoning process to be carried out in the level of agents rather than the system to ensure the agent's autonomy in the decision-making process.

References

1. Bailey, J.M., Golpayegani, F., Clarke, S.: CoMASig: a collaborative multi-agent signal control to support senior drivers. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 1239–1244. IEEE (2019)
2. Dastani, M., Dix, J., Verhagen, H., Villata, S.: Normative multi-agent systems (Dagstuhl seminar 18171). Dagstuhl Rep. **8**(4), 72–103 (2018)
3. Dell'Anna, D., Dastani, M., Dalpiaz, F.: Runtime norm revision using Bayesian networks. In: Miller, T., Oren, N., Sakurai, Y., Noda, I., Savarimuthu, B.T.R., Cao Son, T. (eds.) PRIMA 2018. LNCS (LNAI), vol. 11224, pp. 279–295. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03098-8_17
4. Dorri, A., Kanhere, S.S., Jurdak, R.: Multi-agent systems: a survey. IEEE Access **6**, 28573–28593 (2018)
5. Edenhofer, S., et al.: Bottom-up norm adjustment in open, heterogeneous agent societies. In: 2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W), pp. 36–41. IEEE (2016)

6. Frantz, C., Pigozzi, G.: Modeling norm dynamics in multiagent systems. FLAP **5**(2), 491–564 (2018)
7. Ghanadbashi, S., Golpayegani, F.: Using ontology to guide reinforcement learning agents in unseen situations. Appl. Intell. **52**(2), 1808–1824 (2021). https://doi.org/10.1007/s10489-021-02449-5
8. Golpayegani, F., Dusparic, I., Clarke, S.: Using social dependence to enable neighbourly behaviour in open multi-agent systems. ACM Trans. Intell. Syst. Technol. (TIST) **10**(3), 1–31 (2019)
9. Golpayegani, F., Dusparic, I., Taylor, A., Clarke, S.: Multi-agent collaboration for conflict management in residential demand response. Comput. Commun. **96**, 63–72 (2016)
10. Mahmoud, M., Ahmad, M.S., Mohd Yusoff, M.Z.: Development and implementation of a technique for norms-adaptable agents in open multi-agent communities. J. Syst. Sci. Complex. **29**(6), 1519–1537 (2016). https://doi.org/10.1007/s11424-016-5036-1
11. Morales, J., Lopez-Sanchez, M., Rodriguez-Aguilar, J.A., Wooldridge, M.J., Vasconcelos, W.W.: Automated synthesis of normative systems. In: AAMAS, vol. 13, pp. 483–490 (2013)
12. Oliveira, A., Girardi, R.: An analysis of norm processes in normative multi-agent systems. In: 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW), pp. 68–71. IEEE (2016)
13. Singh, M.P., Chopra, A.K.: The internet of things and multiagent systems: decentralized intelligence in distributed computing. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 1738–1747. IEEE (2017)