A SUMO Extension for Norm-Based Traffic Control Systems

Jetze Baumfalk, Mehdi Dastani, Barend Poot and Bas Testerink

Abstract Autonomous vehicles will most likely participate in traffic in the near future. The advent of autonomous vehicles allows us to explore innovative ideas for traffic control such as norm-based traffic control. A norm is a violable rule that describes correct behavior. Norm-based traffic controllers monitor traffic and effectuate sanctions in case vehicles violate norms. In this paper, we present an extension of SUMO that enables the user to apply norm-based traffic controllers to traffic simulations. In our extension, named TrafficMAS, vehicles are capable of making an autonomous decision on whether to comply with norms. We provide a description of the extension, a summary on its implementation and demonstrative experiments.

1 Introduction

Recent developments in the automotive industry steer toward a future where autonomous vehicles are part of everyday traffic (cf. [1, 2, 15]). Vehicles will no longer have a human driver, but will drive themselves and communicate with smart road infrastructures and other vehicles. Clearly, human drivers are different from software programs that operate vehicles. For instance, the response of human drivers to receiving events is considerably slower and less accurate in comparison with a computer program. These differences pose new challenges and opportunities [8]. For example, delegating cruise control to the vehicles' board computers allows vehicles to coordinate and form platoons, which improves the traffic flow [12].

J. Baumfalk (✉) · M. Dastani · B. Poot · B. Testerink
Utrecht University, Utrecht, Netherlands
e-mail: J.T.Baumfalk@uu.nl

M. Dastani
e-mail: M.M.Dastani@uu.nl

B. Poot
e-mail: B.W.A.Poot@uu.nl

B. Testerink
e-mail: B.J.G.Testerink@uu.nl

M. Behrisch and M. Weber (eds.), *Simulating Urban Traffic Scenarios*,
Lecture Notes in Mobility, https://doi.org/10.1007/978-3-319-33616-9_5

In this paper, we are particularly interested in future challenges and opportunities for traffic control. Traffic controllers can exert some level of influence on vehicles in order to improve traffic flow and safety. Currently, traffic is controlled by means of traffic laws and signs which require the education of general traffic regulations and the interpretation of signs. The government as a regulator creates incentive to follow the regulations by imposing sanctions on anybody who is caught violating them. This control mechanism is tailored for humans, as they are currently the road's only occupants. For example, speed limits are given in easy round numbers, as we expect humans to only approximate their limits within an error margin. An autonomously controlled vehicle has a more precise control over its velocity and hence its error margins are different. This allows us to give an autonomous vehicle more precise directives. We shall address the question of how we can design, analyze, and test new traffic controllers where (a portion of) the vehicle is driven by autonomous software systems.

We envision a future where the traffic infrastructure consists of smart roads, enriched with software systems that control traffic. We will call such a software system a traffic controller. The function of a traffic controller is to observe and evaluate traffic, and communicate personalized directive to vehicles. Such a traffic controller is required to respect the autonomy of the vehicles as autonomous vehicles are assumed to be self-interested with possibly personal incentives to violate traffic regulations [5]. Traffic controllers will be allowed to impose sanctions on autonomous vehicles as a way to promote desired behavior. Thus, in our vision autonomous vehicles, which are aware of traffic regulations and their corresponding sanctions, may still violate traffic regulations and accept the imposed sanctions whenever their personal objectives are worth the incurred sanctions. Traffic controllers should also be easily maintainable in terms of the traffic regulations. In our approach, we focus on drivers, the state of traffic, the regulations, and the traffic controllers. We shall argue that a suitable paradigm for these kinds of control systems is that of norm-based control for multi-agent systems. Norms are violable regulations, whose violations result in the imposition of sanctions, much like current traffic regulations.

Aside from design, we also want to see what the effect is of a new traffic control system. As real-world experiments are an expensive affair, a traffic control research project starts often with traffic simulations. We choose SUMO [13] as a simulation platform because it is open-source, has a track record of research behind it, and performs well. We did observe, however, that SUMO does not provide a straightforward platform to implement concepts from the paradigm of norm-based control for multi-agent systems. In this paper, we present an extension to SUMO, named TrafficMAS, that allows the user to specify and execute norm-based traffic controllers for traffic. This software is open-source and can be found at https://github.com/baumfalk/TrafficMAS. A small user guide on the extension can be found in Appendix 2, and more information is available on the Github page.

The running example in this paper is a ramp-merging scenario, where two traffic streams have to merge together. For a schematic overview, see Fig. 1. There is one main input stream and one ramp input stream, resulting into a single output stream. The goal is to make optimal use of the output capacity of the network while not

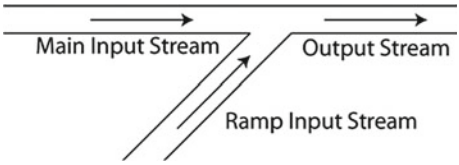

Fig. 1 Example scenario where two traffic streams must merge

causing unnecessary traffic jams for the ramp input stream or compromising safety. A solid analysis of this scenario can be found in [8]. Our approach is to use a traffic controller that assigns individual directives to vehicles. In particular, vehicles are obligated to move or stay on a lane and/or adopt a certain target speed until they are released of this directive.

Our paper is structured as follows. We first give a brief introduction in the field of norm-based control systems. We will then show how the norm-based control system for SUMO is designed and discuss its application in our example scenario. Following that, we describe how vehicles can reason about the norms that are directed to them by the traffic controller. We then explain our implementation approach. In the next section, we evaluate our extension through a series of experiments that highlight different aspects of our contribution. Finally, we look at related work and compare them with our approach.

2 Norm-Based Control Systems

Norm-based control systems are a popular technology for coordination in the multi-agent systems [10] community. A multi-agent system consists of a set of agents that interact within a shared environment. The agents may communicate with each other or perform actions in their shared environment. In addition to reactively responding to environmental changes, autonomous agents are assumed to have their own objectives (goals) for which they proactively initiate actions in order to achieve them. Aside from its objectives, the knowledge/belief of an agent may determine the actions it decides to perform. A simple model of an agent's internal process is the sense–reason–act cycle. In this cycle, the agent first senses what the state of the environment is; then reasons about its goals, preferences, etc., to determine what action it wants to perform; and then executes the action. Although the agents' behavior is not always predictable or controllable, multi-agent systems are often required to satisfy some global properties. For our purposes, we consider smart roads as a multi-agent system where the state of traffic as well as the infrastructure is seen as the environment, and the drivers of vehicles, whether human or not, are seen as the agents in a traffic MAS. The throughput and safety of smart roads are considered as the global properties that are required to be satisfied.

2.1 Background on Norm-Based Control Systems

The behavior of individual agents needs to be controlled and coordinated in order to ensure the desirable properties at the system level [10]. A possible approach would be to design and implement hard constraints at both individual agent and multi-agent system levels. However, this approach may not always be desirable or feasible since limiting the autonomy of individual agents can be costly or even impossible. For instance, in our smart roads scenario it is undesirable, if not impossible, to fully control and determine the behavior of all individual autonomous cars as these vehicles are assumed and designed to be able to make their own decisions.

Norm-based control systems are widely proposed as an effective mechanism to control and coordinate the behavior of individual agents [9]. In norm-based control systems, norms are considered as specifying the standards of behavior that can be used to govern the interaction between autonomous agents (cf. [4, 11, 19]). Across various theories and frameworks, there is no general consensus on what a norm is. In this paper, we use the term norm as a reference to both norm schemes and norm instances. With the concept norm scheme, we refer to the specification of the circumstances after which a specific agent is obliged to achieve a system state, and the sanction that will be imposed should this obligation not be met before a certain deadline (cf. [19]). With the concept norm instance, we refer to a specific directive and sanction that is in effect for a specific agent. In general, norms can take the form of an obligation, prohibition, or permission, but the scope of this paper concerns only obligation norms. The application of norms in multi-agent systems, called norm-based multi-agent systems, requires continuous monitoring of the behavior of individual agents, evaluation of their behavior with respect to the specified norms, and assurance that norm-violating agents are sanctioned. This approach maintains the agents' autonomy and can still promote desirable behavior. We observe that traffic regulations can be formulated following the normative approach. For instance, one can straightforwardly formulate a speed limit measure in terms of a condition (entering the road), obligation (maintaining maximum velocity), deadline (passing a camera), and sanction (a fine).

Some applications are inherently distributed and require distributed control systems. There are various benefits for distributed control such as increased robustness, parallel processing of data, less communication of data, and modular maintenance [18]. We believe this is also strongly the case for future traffic in smart roads where sensors and traffic controllers are geographically distributed, different sections of road networks are governed by different sets of norms and regulations, and the amount of data generated and processed is big. Therefore, we aim at decentralized traffic control consisting of decentralized monitoring of sensor data as well as distributed enforcement of norms. Our extension allows for a straightforward implementation of individual traffic controllers and their communication. Although a decentralized traffic control mechanism is more robust in the sense that it avoids the problem of single point of failure, it also introduces new issues. In decentralized monitoring, one

has to make sure that the sensors are cooperating correctly to detect norm violations in a timely fashion [17].

2.2 Norm-Based Traffic Controllers for Traffic

A norm-based traffic controller monitors vehicle behavior and reacts to it. The monitoring functionality of traffic controller serves two purposes. The first purpose is to detect violations of norms, such as speeding, tailgating, or driving on a priority lane without a permit. If such a violation is detected, then a sanction coupled with this violation will be issued toward the violating vehicle. The second purpose of monitoring is to issue new norms. If the traffic controller continues to observe situations where either the throughput or safety of vehicles declines, then a norm might be issued to improve the situation. This norm might be global, or tailored to a specific vehicle. This allows the traffic controller to be very adaptive to traffic dynamics. Furthermore, the severity of a sanction might be increased if the coupled violation either occurs an excessive amount of times or seems to be the cause of problematic situations. In our framework for norm-based traffic controller, both autonomous vehicles and the traffic controller show adaptive behavior toward the ever-changing traffic flow and enable the system to cope with difficult and dynamic traffic situations.

A norm-based traffic controller on smart roads has a collection of sensors with which it can sense the state of the environment, in this case, the state of traffic and the infrastructure. Examples of these sensors are inductive-loop detectors or cameras that can perform image processing aside from speed measurements. Traffic regulations are formulated as norms. A traffic controller has a set of norm schemes that given sensor data can make norm instances. The norm instances are maintained by the traffic controller for as long as they are applicable. Finally, a traffic controller may have subscribers and subscriptions. In a decentralized setting, traffic controllers can subscribe to each other in order to receive sensor data which they cannot obtain locally. Hence, we obtain a form of decentralized norm-based traffic control.

The norm-based traffic controllers in our extension are modeled as a cyclic process that continuously senses its environment (sense phase), evaluates the norms (reason phase), and imposes sanctions when norms are violated (act phase). In the sense phase, traffic controllers have two ways of observing the environment: (i) Obtain data from the sensors that are placed in the environment, and (ii) obtain data from other traffic controllers (by means of communication) to which the norm-based controller is subscribed. In case of the SUMO simulations, this happens every tick. Immediately after the data is received from the sensors, information is communicated between the traffic controllers. In the reason phase, the traffic controllers apply their information to their norm schemes in order to instantiate norm, if possible at all. Finally, the traffic controllers act on these new norm instances by communicating the directives to the vehicles in order for drivers to adapt their behavior. Furthermore, the traffic controllers act by imposing sanctions on the drivers when they fail to comply to existing obligations when a deadline has been reached.

In our smart road application, norm instances are announced by the norm-based traffic controller to autonomous vehicles, just like drivers are assumed to be aware of traffic regulations. Norm awareness allows an autonomous vehicle to reason and decides whether or not to comply with the norms (cf. [3, 16]). Norm awareness is crucial for future traffic on smart roads as autonomous vehicles need to know the consequences of norm violations before deciding whether or not to comply with the norms.

3 Application of Norm-Based Traffic Control in SUMO

In our use-case scenario, we illustrate a fairly simple norm-based traffic controller. The traffic controller first calculates when individual vehicles would arrive on the merge point of the traffic streams. The algorithm used is described in [21] and returns an ordering of vehicles. Next, the target velocity is calculated for each individual vehicle that ensures that it a) crosses the merge point at least two seconds after the vehicle that will cross the point before it, and b) the maximum safe velocity is still maintained. In case the mainstream road has two lanes, a vehicle can be obliged to move to the left lane too. This happens when target velocity of a vehicle on the main road is below a predefined threshold. The sanctions of violating norms are captured by a low or high fine.

In Fig. 2, a schematic representation of the aforementioned ramp-merging scenario is given. Triangles are vehicles that travel in the direction toward they point. White vehicles are the ones that have not yet received their directive from the traffic controller, while the black vehicles have passed a sensor and have thus received a personalized norm instance. The vehicles without a norm instance in Fig. 2 are vehicles A, B, C, and D. Lane sensors (s_1 to s_5) are placed on the roads. These sensors can detect the status of vehicles that are driving over them. For the scenario to work correctly, it is necessary that either the sensors are sufficiently long or the vehicles are sufficiently slow, so that no vehicle can pass the sensors undetected. The sensors should also be placed at a distance far enough from the merge point m such that vehicles have enough time to comply with the directive before the deadline. There are two important points on the road: point m where the two roads merge, and point e where the vehicles exit the scenario. Distances d_A and d_C are agents A and C's distances to m, d_{exit} is the distance from the m to e, and d_{safe} is the distance between vehicles that are deemed safe (i.e., the minimal gap between cars). Ideally, A and C traverse d_A and d_C such that they arrive at m with a distance d_{safe} and can accelerate to their maximum speed within the distance d_{exit}.

Monitoring happens through interpreting the observations of sensors. In the case of SUMO, we use lane detectors that can sense the vehicles that driving along the area they cover. Specifically, each sensor can detect the identity, velocity, and position of each vehicle on the sensor's area. Further parameters such as the maximum velocity, acceleration, and deceleration capacities can be assumed within reasonable margins. If the traffic controller instantiates a norm, then the subjected vehicle is notified of

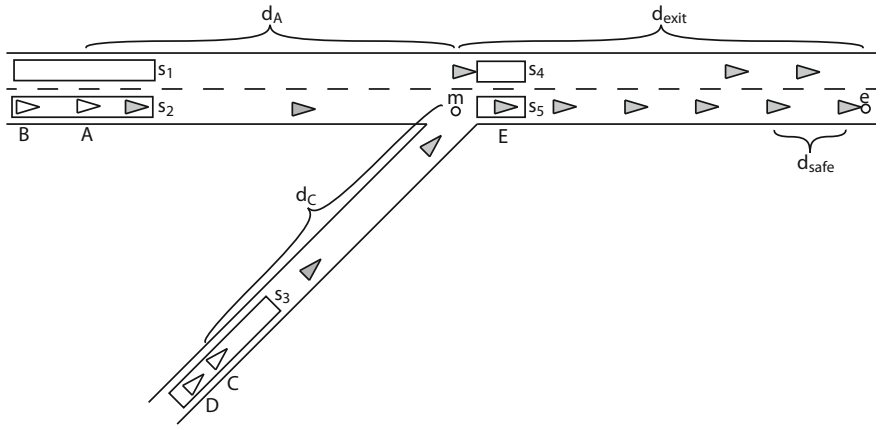

Fig. 2 Scenario with two lanes

the directive that it has to fulfill and the associated sanction that will be imposed if the directive is not followed.

The standard traffic rule in the ramp-merging scenario is that the stream that originates on the main road has priority over the ramp road's stream. However, if the main road is busy, this may lead to large traffic jams on the secondary road. Therefore, the traffic controller uses the traffic data from sensors 1–3 and calculates the optimal velocities of the vehicles in such a way that the traffic streams merge smoothly together at the merge point. More specifically, the traffic controller notifies vehicles passing sensor 1 to stay on the left lane. Sensors 2 and 3 are used by the traffic controller to coordinate the scheduling of vehicles on the merge point. If a vehicle passing sensor 2 has to slow down too much, i.e., to a velocity less than a given threshold, then it receives the directive to move to the left lane.

In order to not overcomplicate the scenario, we decided to simplify some aspects of the traffic controller. The sanction that a vehicle can receive for violating a norm is modeled by either a low or a high fine. A directive that a vehicle can receive is an obligation to be on the left or right lane of the main road at a certain target velocity. For instance, $(right, 10)$ is read as the obligation to be on the right lane at 10 m/s. Given that autonomous vehicles are accurate, we take real numbers for the specification of target velocity rather than multiples of 10 km/h as is common in current traffic controllers. A norm instance consists of a directive that is paired with a sanction. The sets of possible sanctions, directives, and norm instances for this scenario are global throughout the paper and given by:

- $S = \{low, high\}$ are the possible sanctions.
- $O = \{left, right\} \times \mathbb{R}$ are the possible directives.
- $N = O \times S$ are the possible norm instances.

For each simulation step, the new norm instances are created. Recall that norm instances are created based on norm schemes.

3.1 Norm Scheme: Example

For our scenario, the traffic controller instantiates a norm scheme into a vehicle-specific norm in such a way that vehicles cannot arrive on the merge point at the same time if they comply with their norms. This will cause vehicles to slow down considerably on the main road. If they have to slow down too much, then they will be obliged to move to the left lane which is assumed to be more free-flowing. The exact explanation of this norm is provided in Appendix 1.

We also have another norm scheme that obliges vehicles that enter on the left lane of the main road to stay on their lane at a preset maximum velocity v_{MAX}. We shall illustrate the different aspects of a norm scheme according to this scheme's pseudocode that is given in Algorithm 1. We assume that there exists a current set of norm instances N, a function sanction that given a vehicle and fine issues the fine for that vehicle, and a function read that returns the vehicles on a sensor's area that has not been seen before by that sensor. Though not a forced pattern, we do encourage future users of our extension to use the same code structure as in Algorithm 1. Every norm scheme has a specification of when norm instances are created, when they are retracted, and when a sanction should be issued.

Algorithm 1 Pseudocode for the stay-on-lane norm scheme

1: $L \leftarrow read(s_1)$
2: **for all** $agent \in L$ **do**
3: $\quad N \leftarrow N \cup \{(agent, ((left, v_{MAX}), low))\}$
4: $L \leftarrow read(s_4)$
5: **for all** $agent \in L$ **do**
6: $\quad$ **if** $(agent, ((left, v_{MAX}), low)) \in N \& agent.v < v_{MAX}$ **then**
7: $\quad\quad sanction(agent, low)$
8: $\quad N \leftarrow N \backslash \{(agent, ((left, v_{MAX}), low))\}$
9: $L \leftarrow read(s_5)$
10: **for all** $agent \in L$ **do**
11: $\quad$ **if** $(agent, ((left, v_{MAX}), low)) \in N$ **then**
12: $\quad\quad sanction(agent, low)$
13: $\quad N \leftarrow N \backslash \{(agent, ((left, v_{MAX}), low))\}$

The code of Algorithm 1 is executed after every simulation tick. We begin with creating new norm instances (lines 1–3). In this case, sensor 1 is read. In Fig. 2, it can be seen that this sensor detects all vehicles that enter the scenario on the left lane of the road. Hence, we give each vehicle on the left lane the directive to stay on the left lane and also obtain a preset maximum velocity to ensure that the flow stays high (line 3). We then continue by checking sensor 4 (lines 4–8). Vehicles with instances of this norm that pass sensor 4 fulfilled their directive to stay on the left lane. However, if their velocity is not the obliged velocity, they receive a low fine (lines 6–7). After passing this sensor, the vehicles are relieved of their directives (line 8). The same holds for sensor 5 (line 9). However, if a vehicle passes sensor 5 then

it means that it switched to the right lane. Hence, each vehicle that has received a directive to stay left but passes sensor 5 is fined (line 12).

4 Norm-Aware Vehicles

Our goal is to build autonomous norm-aware vehicles that operate on smart roads where vehicles' behaviors are automatically monitored, evaluated with respect to traffic norms, and possibly sanctioned. As mentioned before, we assume that individual vehicles have their own objectives (e.g., destination, arrival time, travel cost) and are able to deliberate and decide on actions that achieve their objectives. This implies that a vehicle can choose to obey/violate a norm, when this contributes to the achievement of its objectives. In this section, we will discuss norm awareness and how we adopted it in our own driver models.

4.1 Background on Norm Awareness

We have explained norm-based controllers in a multi-agent system as being external entities to the agents, which may oblige/forbid certain behavior and impose sanctions. This abstraction comes from the human way of organizing. It is beneficial to make agents in a multi-agent system norm-aware, especially in simulations where we want the agents to change their behavior due to the norms as humans do. Norm awareness falls under the umbrella of organizational awareness [20]. Organizational awareness is about allowing agents to reason about their role within an organization. However, for traffic simulation purposes, all agents have the same role (i.e., being a driver), hence we will focus on norm awareness only.

Being norm-aware means that an agent can reason about the norms that it receives. The directives that an agent receives are unlikely to match its goals and desires. Otherwise, there would be no need for sanctions. Reasoning about norms hence entails weighing for a course of actions the benefits (reaching goals, fulfilling desires, etc.) and the penalties (expected sanctions from the norm-based controller). In general, the more complex the planning mechanism of the agent is, the harder it is to incorporate reasoning about norms. For an example language for programming norm-aware agents that can deliberate about norms w.r.t. their own goals and plans, we refer the reader to N-2APL [3].

4.2 A Norm-Aware Driver Model

In order to model norm awareness, we deviate from the standard SUMO driver models [14]. There are several reasons for this. First, vehicles in SUMO are goal-directed

only in a limited way. For example, the goal of SUMO drivers is to follow a certain route, as opposed to the goal of having a specific location as the destination. Second, SUMO agents are preprogrammed to follow a specific route. They only respond reactively to their environment, instead of deliberating on what action would best suit them. Third, vehicles in SUMO are inherently incapable of deciding to break the rules. For example, they always stop for a red light and they obey to the right of way. This is because of vehicles in SUMO move according to specific car-following and lane-changing rules. The car-following model is not created with norm-aware vehicles in mind. The most commonly used car-following model made by Stefan Krauss is designed purely to create realistic traffic flows in general since in most traffic simulations individual movement on the microscopic level is not interesting [14]. In contrast, we aim at designing futuristic autonomous cars with a more fine-grained sense of control and, most importantly, the ability to violate norms. In our work, we model drivers with different possible actions, a belief state, and personal preferences. The available actions of a vehicle depend on the scenario. The belief state of a vehicle consists of:

- A description of its runtime variables which contain its velocity, position (given by a lane and distance from that lane's start), and current norm instances.
- An expected arrival time (at the goal location) function that reflects, for instance, the GPS planning tools that vehicles have available. This function is equal for all vehicles, but can be parameterized in the future in order to make more optimistic/pessimistic drivers.
- A local effect function that returns the next expected runtime variable configuration, given the runtime variables of a vehicle and an action. For instance, if the current velocity is 20 m/s and a vehicle accelerates by 5 as an action, then it expects for the next simulation tick to be at 25 m/s if this is possible within its acceleration capabilities. This function is also equal for all vehicles.
- A directive distance function that returns a positive expectancy of whether the vehicle can fulfill the directive in time before it is sanctioned, given runtime variables and a directive. The precise definition of this function depends on the possible directives and driver specifics in an application. However, a high distance should mean that it is likely that the sanction will be incurred in the future, whereas a distance of zero should indicate that the current state fulfills the directive. This function is also equal for all vehicles.

The personal preferences of a vehicle are given by its personal profile. This profile consists of:

- A maximum desirable velocity of the vehicle.
- A sanction grading function that returns how bad a sanction is to the vehicle. The higher the number, the worse the sanction. This can be used to model how affluent or greedy a vehicle is.
- A arrival time grading function that returns how good or bad an arrival time is. This encodes the desired arrival time of the vehicle. Anything before the desired time

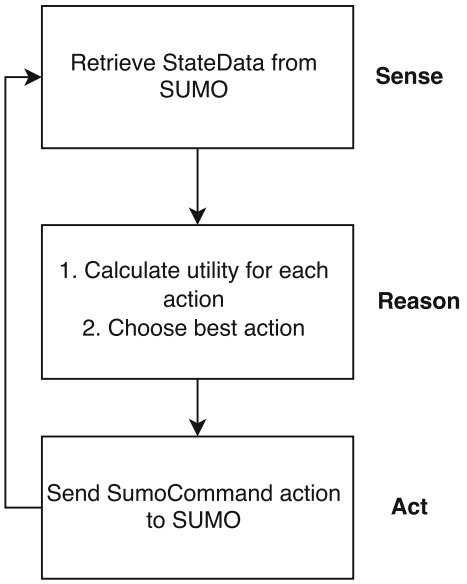

Fig. 3 Sense–reason–act cycle of an agent

should be evaluated to zero or less, and everything after it should monotonically increase in their evaluation.

Our vehicles are assumed to use a sense–reason–act cycle. In our extension, this cycle is performed at each simulation tick of SUMO (Fig. 3). In practice, it is not required that the vehicles run synchronously with the traffic controllers and/or other vehicles. In practice, a vehicle perceives its road environment by its onboard sensors. In our extension, this information is obtained from SUMO, which gives a vehicle its local information. A vehicle also receives newly instantiated or retracted directives that apply to itself. This information updates the vehicle's belief state (its runtime variables). Using its knowledge of the road network, the vehicle estimates the utility of each of its actions by balancing between the value of achieving its objectives and possible sanctions that will occur if it performs the action. The arrival time and expected sanctions are used to calculate the utility for each action.

The action with the highest utility is chosen by the agent's action selection function, using a priority-based tie-break mechanism for ties. The tie-break mechanism is used when two or more actions have the highest utility. If such a situation occurs, then the action with the highest priority is chosen. In our implementation, the less an action changes the agent state, the higher its priority is. For example, in a tie-break situation, doing nothing is preferred to increasing velocity to a small amount, which in turn is preferred to increasing velocity to a larger amount, which in turn is preferred to changing lane. We chose for this 'least impactful action' tie-break ordering since we believe this to be in line with human behavior. However, we stress that this ordering is not essential to our framework and can be replaced by arbitrary tie-break orderings. Finally, the simulator executes this action.

Recall that S is the set of possible sanctions, O the set of possible directives, and N the set of possible norm instances. The relevant components of an agent are modeled as follows:

- A is the set of actions to choose from.
- $\langle v, l, d, n\rangle$ is a specification of the runtime variables of a vehicle, where v is the current velocity, l is the current lane, d is the distance from the starting point of that lane, and $n \subseteq N$ are norm instances.
- B is the set of all possible runtime variables configurations.
- $f : B \mapsto N$ is the expected arrival time function.
- $e : B \times A \mapsto B$ is the local action effect function.
- $\delta : B \times O \mapsto \mathbb{R}$ is the directive distance function.
- $\langle v_{max}, g_s, g_t\rangle$ is a specification of a vehicle's personal profile, where v_{max} is the maximum velocity, $g_s : S \mapsto \mathbb{R}$ is the sanction grading function, and $g_t : N \mapsto \mathbb{R}$ is the arrival time grading function.
- P is the set of all possible personal profiles.
- $u : B \times P \times A \mapsto \mathbb{R}$ is the utility function, given by:

$$u(b, p, a) = g_t(f(b')) + \sum_{(o,s)\in n} (\delta(b', o) \cdot g_s(s)),$$

  where $b = \langle v, l, d, n\rangle$, $p = \langle v_{max}, g_s, g_t\rangle$, and $b' = e(b, a)$.
- $\alpha : B \times P \to A$ is the action selection function given by:

$$\alpha(b, p) = \max_{a\in A} u(b, p, a)$$

5 Application of Norm-Aware Driver Models

In this section, we shall go into detail how the norm-aware driver models are specified for our scenario. We give a specification of the scenario's specific components and discuss utility calculations.

5.1 Vehicle Driver Specification

A vehicle reasons about all its actions when it deliberates for a next action. Among all actions in our scenario are de-/accelerating action. We have simplified this by discretizing the possible de-/acceleration values. The other possible actions available to a vehicle are switching a lane to the left or right. More specifically, the set of actions A that a vehicle can decide to perform are:

$$A = \{a_x \mid x \in \{0, 0.1, -0.1, 1, -1, 5, -5, 10, -10, 20, -20, 50, -50\}\} \cup \{l_{left}, l_{right}\},$$

where a_x is read as adding x to the current velocity (i.e., de-/accelerating) and l_{left}/l_{right} is read as moving a lane to the left or to the right.

We also specify how the directive distance measure is calculated given the possible directives in our scenario. In our scenario, the factors that a vehicle considers are the time it takes to fulfill the directive, the current time, and the expected time that the sanction will be issued if the directive is not followed. We have implemented vehicles in such a way that they expect that the traffic controller check whether a directive is followed somewhere between the current time t and the current expected exit time t_{exit} for the vehicle. The minimal amount of time needed to adhere to the norm is denoted δ_t. For instance, if a vehicle at time step t is being instructed to drive 25 m/s and can accelerate to this speed in minimally three time units, then $\delta_t = 3$. If we need zero steps to adhere to the norm, the distance is zero. Otherwise, the distance proportionally moves to 1 given the current time. If the traffic controller will check directive fulfillment before the driver can achieve compliance (i.e., $t_{exit} - t < \delta_t$), then δ should be 1. Hence, the directive distance measure is given by:

$$\delta(b, (l_o, v_o)) = \frac{\min(\delta_t, t_{exit} - t)}{(t_{exit} - t)},$$

where t is the current time, t_{exit} is the expected exit time, and δ_t is the minimal number of steps needed for compliance of (l_o, v_o) given the current runtime variables b.

Note that this means that the drivers expect the traffic controller to issue a sanction if a directive is not followed between now and $t_{exit} - t$ time units later. This distance measure can be modified easily in our framework.

5.2 Utility Calculations: Example

To illustrate this notion, suppose we have two drivers, a poor one and an affluent one in an identical situation. They currently drive 20 m/s, their maximum speed is 30 m/s, they can accelerate or decelerate with 10 m/s and their travel distance is 1080 m. Both are in a hurry, so their g_t is defined as $g_t(time) = \frac{bestTime}{time}$.

Here, $bestTime$ is defined by the minimal travel time, i.e., the time it would take the drivers to travel the distance if they could go their maximum speed all the time. In this case, $bestTime = 1080/30 = 36$. However, the road the drivers travel on has a speed norm, with the maximum speed being 10 m/s. Not complying with this norm gives a high fine. The poor agent cannot afford this fine, so it has $g_s(high) = -20$. The affluent driver can easily afford this fine, so it has $g_s(high) = -0.2$. Suppose for this example that drivers can only take the actions a_0, a_{10} and a_{-10}.

In Table 1, we see the utilities for each of these actions for both drivers. Here, we see that the highest rewarded action for the poor driver is the one that obeys the norm since it cannot afford the fine, while the affluent driver is in a position to violate the

Table 1 Example of the deliberation of a poor and affluent agent

	a_0	a_{10}	a_{-10}
Speed after action	20 m/s	30 m/s	10 m/s
Norm speed	10 m/s	10 m/s	10 m/s
Travel time remaining	54 s	36 s	108 s
g_t	0.67	1.00	0.33
Steps needed to oblige the norm	1	2	0
δ	0.02	0.06	0.00
Utility poor agent	0.30	−0.11	0.33
Utility rich agent	0.66	0.99	0.33

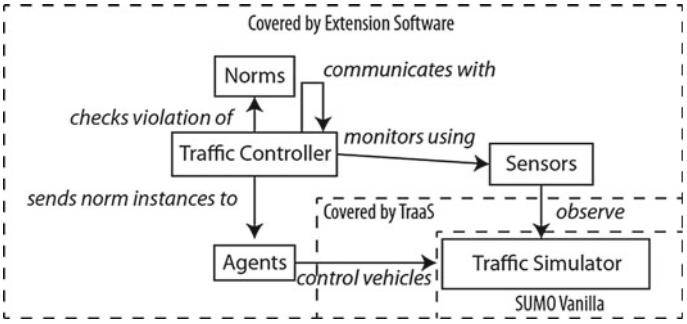

Fig. 4 Overview of the presented extension

speed norm since it can afford the fine. In fact, the affluent driver will increase its speed since it then maximizes its time grade and thereby its utility.

6 Implementation Approach

The structure of the presented norm-based traffic controller for SUMO is depicted in Fig. 4. As stated before, we use the native SUMO application as the environment. However, we do not use SUMO's driver models. Instead, we communicate commands from our own agent model to the vehicles. We have also implemented our own sensor business logic. These sensors are connected to lane area detectors as implemented in SUMO. The communication between the extension software and SUMO is provided by the TraaS library. We use and provide a new version of TraaS. In addition to improved performances, the new version has some extra functionalities that are needed for our extension.

The software in our extension is composed mainly of the agent models and the traffic controller. The software executes in lockstep with SUMO; i.e., each simulation

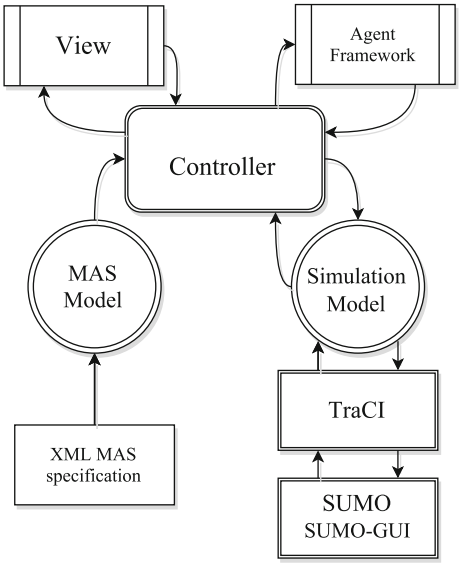

Fig. 5 Model-View-Controller structure

tick in SUMO is also a tick in the extension. We chose to provide our work as an extension to SUMO, rather than modifying the source code of SUMO directly.

There are various pragmatic reasons to propose a SUMO extension rather than altering its code. For example, we can now support multiple versions of SUMO, starting from SUMO 0.20 and upwards. This also makes our extension more accessible to users, since it eliminates the need for users to recompile SUMO before they can use the framework. The framework is developed in Java since most norm-based autonomous agent frameworks are written in Java and Java programs are easy to use on a variety of platforms. We have designed the framework using the Model-View-Controller pattern, as can be seen in Fig. 5. A multi-agent system can be specified in XML and is converted to a MAS model. We furthermore use TraaS to retrieve the simulation state of SUMO. These models can be manipulated by the controller software (note that the controller is a control component in the software engineering sense; it is not the same as the conceptual traffic controller). The agent framework simulates our driver models. The view of the system allows the user to see the state of the MAS side of the simulation. This decomposition allows other researchers to easily create their own front-end by changing the view implementation. It is also possible to create a new data format for scenarios by changing the data model implementation, or to change the simulation package by providing a different simulation model implementation. Each part can be changed without the need to change other parts of the framework.

In Fig. 6 an UML overview of the traffic controller structure is shown. As mentioned previously, traffic controllers observe by using sensors. These sensors are an extension of a physical object, occupying space within the environment on a certain road, lane, and distance and have a certain length. The traffic controller consists of

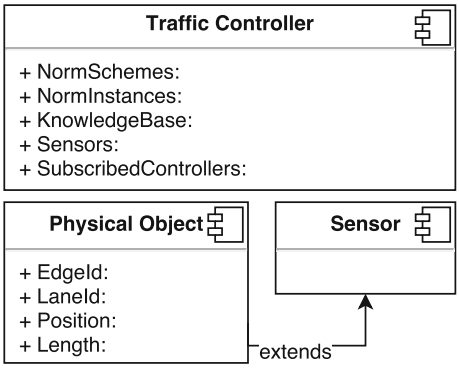

Fig. 6 UML overview of traffic controllers

five attributes. First, NormSchemes, which is a set of traffic regulations to be effectuated by the traffic control system. Second, NormInstances, a set containing all norms instantiated by the controller. In effect, this means a specific regulation is sent to a vehicle. Third is the set Sensors, which is a set of sensors at its disposal to sense the vehicles driving in the environment. Fourth, a traffic controller is subscribed to a set of other traffic control systems denoted by the set SubscribedControllers. Finally, KnowledgeBase is a set containing data about the vehicles in the simulation.

At regular intervals, vehicle data is collected from both the sensors and other traffic controllers and is used to update the traffic controller's knowledge base. It uses its knowledge base in three ways. The first usage is to decide if the traffic regulation should be applied and instantiates a specific regulation for a vehicle. Secondly, a traffic controller employs its knowledge base to determine if the regulation is violated in order to impose sanctions. Finally, the knowledge base is used to determine whether the deadline of an instantiated regulation is met. If so, the regulation is retracted and the relevant vehicle is notified.

The communication of the regulations to vehicles as well as the communication of sensor readings between traffic controllers is assumed to be done wireless. The actual communication protocol is unspecified.

An UML overview of the agent structure is shown in Fig. 7. Agents are an extension of physical objects since they occupy a certain space and have a certain position in the world. A Basic Agent is a prototype AgentProfile. The agent profile is specified by the SanctionGrade and ArrivalTimeGrade functions which are the sanction and arrival time appraisal functions discussed in Sect. 4.2. In the doAction function, the directive distance measure is used to calculate the utility for each individual action. In this manner, the agent profile of the agent together with the current state decides what action the agent will choose toward achieving its goal. These design choices were made to model the agents in such a way that new agent profiles can easily be created to model norm-aware human or driverless autonomous vehicles, while still keeping the agents as simple as possible. With each deliberation cycle, agents can pick the action most suited to their goals and the current state, one of the actions listed by the AgentAction enumeration in Fig. 7.

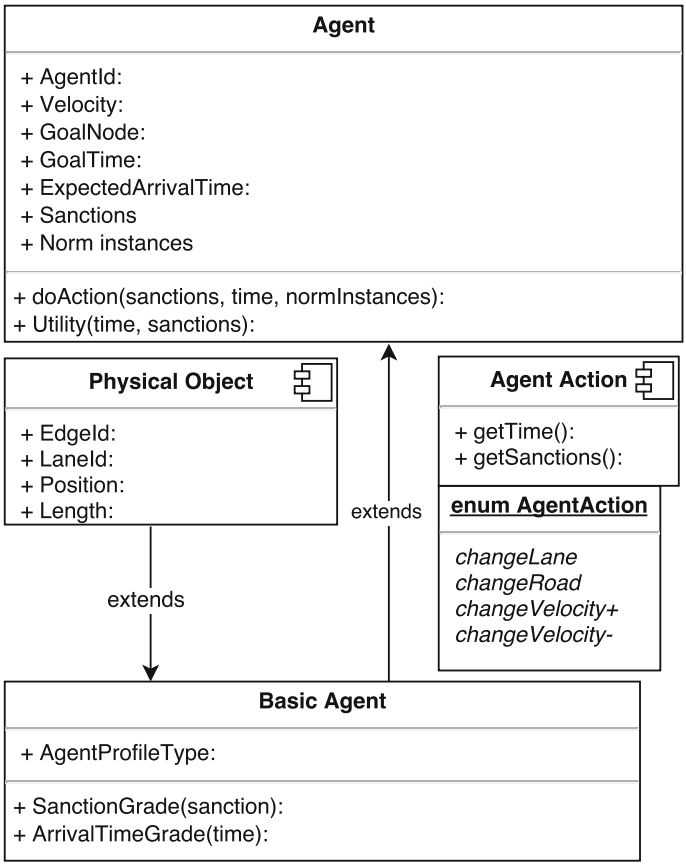

Fig. 7 UML overview of the agent structure

7 Experimental Evaluation

We tested the performance of our normative agent-based traffic approach using four experiments. In these experiments, variations of the ramp-merging scenario, as explained in Sect. 3, are used. The first experiment considers a ramp-merging scenario where the main road consists of a single lane, while on the second, third, and fourth experiment, the main road has two lanes. In the second experiment, the second lane is accessible for all drivers, but in the third experiment the second lane is marked as an emergency-only lane. Finally, in the fourth experiment, the ramp-merging scenario is used twice in succession in order to demonstrate the use of communication and coordination between decentralized traffic controllers.

The experiments were set up as follows. Each experiment is run for a length of one hour (3600 ticks). The spawn rate shown in the tables of the experiments is

defined as the chance of a vehicle spawning every tick. If there is not enough room to spawn a vehicle at a certain time, then SUMO puts the vehicle on hold and spawns it at the earliest possible time when space is available. The maximum speed at the merge point, v_{max}, was set to 80 km/h. Furthermore, the four experiments consist of comparing two scenarios, both ran for one hundred times. The values displayed in the tables are the averages over hundred runs. The throughput is defined as the number of vehicles leaving the simulation every tick. Average speed is the average speed over all runs in m/s, and finally the average gap is the average distance between two cars in meters. Also, we define for each experiment the maximum (expected) throughput. This is the expected throughput if each vehicle could keep driving its maximum speed throughout the scenario and can be calculated by the following formula: $throughput_{max} = 60p$, where p is the probability of a car entering the simulation on that tick.

7.1 Experiment 1: SUMO and TrafficMAS

The first experiment illustrates the behavior difference between the default SUMO vehicles and the norm-aware driver models implemented in the TrafficMAS extension. A single traffic controller observes the vehicles in the simulation and communicates tailored norm instances to each vehicle. In this experiment, a norm instance is simplified to just a target velocity since there is no choice of lanes on the main road. The expected result is that norms result in a higher average velocity and a better throughput of vehicles since traffic jams will be prevented. In this scenario, a classic ramp-merging situation is implemented, where both the main road and ramp consist of a single lane. The spawn rate of the vehicle input stream will be slightly higher on the main road to resemble a realistic traffic situation. In the TrafficMAS scenario, three sensors are placed on the road: one on the main road, one on the ramp, and a control sensor on the output road. The traffic controller instantiates norms, removes norms, or applies sanctions when the vehicles are detected by the sensors. In the SUMO scenario, the main road has priority over the ramp road, comparable to real-life merging situations.

As is clear from the results in Table 2, there is an increase in both throughput, average speed, and the average amount of space between the vehicles. This is the case since in the SUMO scenario a traffic jam instantly forms on the ramp, because of the relatively high density of cars on the main road (Fig. 8). These results confirm our expectation of coordination by a norm-based traffic controller improving on classic ramp-merging scenarios. Note that the throughput % value exceeds a hundred percent, this is possible because the spawn rate is probability based and thus can exceed the maximum expected throughput.

Table 2 Results for the first scenario

	SUMO agents	Norm-aware agents
Main road spawn rate	20%	20%
Ramp spawn rate	15%	15%
Throughput	16.16	21.01
Max throughput	21	21
Throughput %	76.95%	100.05%
Average speed	3	20.97
Average gap	13.81	101.82

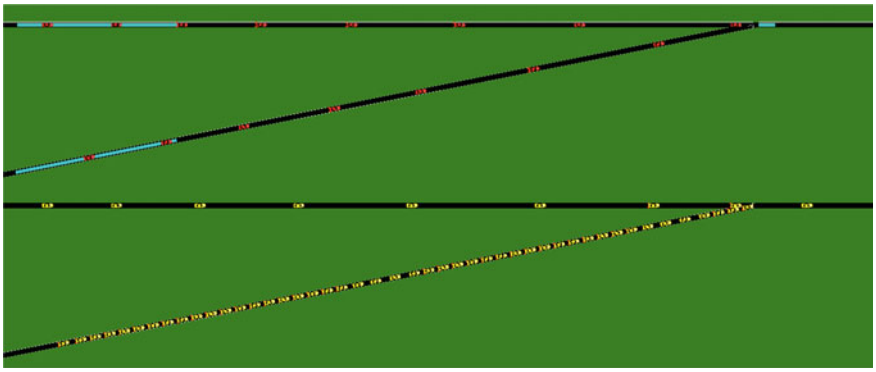

Fig. 8 Screenshot depicting the difference in performance in experiment 1. The top scenario uses our framework and merge norm. The bottom scenario uses the default SUMO driver models

7.2 Experiment 2: Simple Norms and Advanced Norms

The goal of the second experiment is to compare traffic controllers using simple and advanced norms. The traffic controller in the SingleNorm scenario observes and controls the same norm as in experiment 1. In the AdvancedNorm scenario, the traffic controller can also issue directives for the vehicles to change lanes in order to relieve the rightmost lane traffic and prevent congestion. The lane change directive will be given to a vehicle when its calculated velocity on the merge point is below a certain threshold. For this experiment, the threshold was set to $1/2v_{max}$. Our expectation is that in this multi-lane scenario, the traffic controller with the advanced norm can successfully cope with a higher input stream of vehicles.

The setup for the SingleNorm scenario is a copy of the TrafficMAS scenario in experiment 1, except that in this case the main road has two lanes instead of one, and moreover, the input stream of vehicles of both roads are increased. The AdvancedNorm scenario implements extra sensors on the second lane, but is exactly the same in every other aspect.

Table 3 The results for the second scenario

	SimpleNorm	AdvancedNorm
Main road spawn rate	30%	30%
Ramp spawn rate	20%	20%
Throughput	20.38	29.91
Max throughput	30	30
Max throughput %	67.93%	99.70%
Average speed	3.31	14.91
Average gap	14.2	61.49

As can be observed from the results in Table 3, the simple norm cannot cope properly with the increased spawn rate of vehicles in this scenario. The average speed has diminished severely, as well as the average gap between vehicles. This means congestion is abundant in the SimpleNorm scenario. However, the AdvancedNorm seems to cope very well with the increased input stream of vehicles. In this scenario, the throughput approximates the maximum expected throughput by a factor 0.3%, which indicates that the vehicles move throughout the simulation without much congestion.

7.3 Experiment 3: Sanction Severity

The third experiment illustrates that drivers are able to reason about norms. Experiment 1 has shown that drivers are norm-aware. However, TrafficMAS agents also have the capabilities to violate norm if these violations do not have significant impact on them. In this experiment, the leftmost lane is an emergency lane, reserved for certain traffic in order to help with accidents and other emergencies. Therefore regular drivers will get sanctioned if caught driving on this lane. Since this lane remains mostly empty, this is a viable option for drivers who greatly value a faster arrival time and are in a financial position which makes them willing to accept a fine. We expect that the more affluent drivers will choose to accept sanctions in order to improve their arrival time, resulting in distinct behavior between the two groups of drivers.

This experiment is set up in the same way as experiment 2, except that the leftmost lane is reserved for emergencies and the spawn rates are lowered. In the Poor Drivers scenario, the input stream consists of drivers who are impatient, but in a substandard financial position. The Affluent Drivers scenario spawns drivers who care about sanctions, but are willing to accept fines if by doing so they can arrive earlier to their destinations.

The results of this experiment are listed in Table 4. With this experiment, the differences in throughput, average speed, and average gap are much smaller, and

Table 4 The results for the third scenario

	Poor drivers	Affluent drivers
Main road spawn rate	20%	20%
Ramp spawn rate	15%	15%
Throughput	20.48	20.88
Average speed	12.95	14.42
Average gap	48.37	69.29
Sanctions	0	133.12

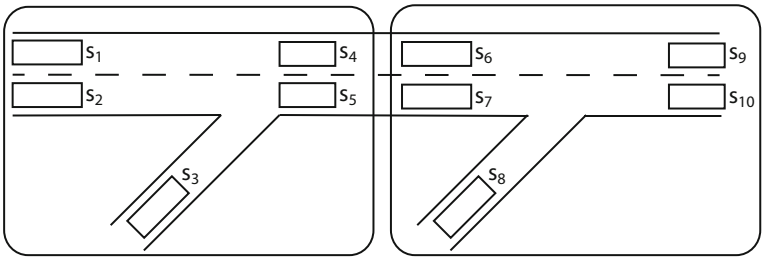

Fig. 9 Distributed traffic control setting. Rounded boxes indicate local traffic controllers. The left controller is connected to sensors 1–5 and the right controller to sensors 6–10

not significant enough to lead to any conclusions about improvement. However, a significant distinction in the number of sanctions can be seen. This indicates a difference in behavior between the groups of drivers. On average about 133 affluent drivers decide to drive on the emergency lane in an hour of simulation. This shows a clear difference in behavior from the poor drivers, who never decide to change lanes.

7.4 Experiment 4: Communication in Distributed Traffic Control Systems

The final variation of the merging scenario that we consider demonstrates the decentralized version of our framework. Specifically we demonstrate the ability of one traffic controller to share data about traffic with another traffic controller so that the receiver can adjust its norms. In this variation there are two merge points in sequence (Fig. 9). Each of the merge points is controlled by a local traffic controller as in the previous scenarios. For this, they have their own local sensors.

When running this scenario without communicating traffic controllers, we observed that the traffic streams tend to flow like the top situation in Fig. 10. Traffic that arrives on the left lane of the main road keeps that lane, as it is faster than switching to the right lane. The ramp traffic streams merge in on the right lane of the main road. After the merge scenario, the vehicles can freely move from left to right and back.

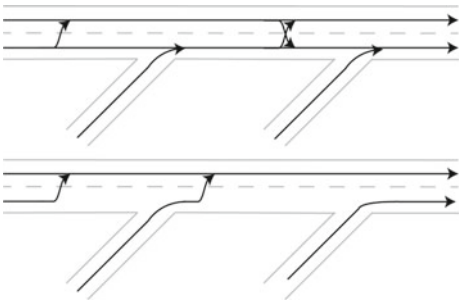

Fig. 10 Top: Traffic streams (arrows) without coordination. Bottom: traffic streams with coordination

However, if the stream of vehicles in the second ramp is too dense, then congestion occurs at the second merge point. The problem is that the second merge point has to process too many vehicles.

A solution might be to redirect all traffic observed by the left traffic controller to the left lane of the main road when high-density streams are observed at the ramp road of the right traffic controller (the second ramp road). This way all traffic on the second ramp road can continue through on the right lane of the main road without being obstructed by oncoming traffic. However, the left traffic controller can only sense the traffic situation using its local sensors. Therefore the right traffic controller needs to inform the left traffic controller about the traffic density on the second ramp.

This coordination is realized as follows. The left traffic controller subscribes to traffic density observed by sensor s_8 of the right traffic controller. If the left traffic controller detects a high traffic density on the second ramp, it will issue new norm instances that obliges vehicles to move to the left lane. As a result the input traffic streams of the right traffic controller should be easily manageable as the vehicles on the main road are obliged to stay left such that the vehicles on the second ramp roads can move on the right lane of the main road. The resulting traffic streams should resemble the streams in the bottom depiction of Fig. 10. We expect that the coordinating traffic controller performs better in terms of throughput, average speed, and average gap, since less congestion should occur at the second merge point.

The results of experiment four are listed in Table 5. A small increase in the throughput and a larger increase the average speed and gap in the coordinated traffic controllers scenario compared with uncoordinated traffic controllers scenario can be observed. Thus, giving vehicles on the main road the obligation to change to the left lane quickly after the first merging point appears to prevent the delays as observed in the original scenario. These preliminary results support our hypothesis that observation sharing and communication between traffic control systems can be effective for traffic regulation.

Table 5 The results for experiment four

	No coordination	Coordination
Main road spawn rate	25%	25%
Ramp #1 spawn rate	15%	15%
Ramp #2 spawn rate	35%	35%
Throughput	43.81	44.74
Max throughput	45	45
Max throughput %	97.36%	99.36%
Average speed	11.32	14.60
Average gap	62.47	66.03

8 Contributions and Comparison to Other Work

The results of our reported experiments were positive in the sense that the advanced version of our framework performed better than the SUMO baseline/simpler versions of our framework. However, as noted previously, some scenarios were not completely realistic. The merge scenario did not have an acceleration lane aligned with the main road. Moreover, in the final experiment, the traffic density of the second merge lane was higher than one would expect in the real life. However, the aim of our experiments was not to show simulate calibrated realistic scenarios. Our goal was rather to show that our extension is an enabling technology for the specification and testing of norm-based traffic control, which can be extended into a more complex and feature-rich framework.

The contributions of our work are as follows. First of all, we have created a lightweight framework for autonomous norm-aware vehicles and norm-based traffic controller on top of SUMO. This framework is easy to extend with different types of driver profiles. It also allows for the easy usage of a different simulation package. Secondly, we have created the possibility to conduct traffic experiments and measuring the impact of a norm-based traffic controller. Finally, we have improved the TraaS performance, yielding a performance increase of up to four times over the original TraaS library in certain scenarios.

Our approach has some similarity with Baines et al. [6] since they employ autonomous agents and use governing institutions to influence agents to have desirable behavior. However, Baines et al. concentrate on agents' internal architecture, situational awareness, and the communication between agents. The project is set up with realistic maps imported from the Open Street Map foundation and used real-world data from a highway in the UK, the M25.

While our framework is related to the work done by Baines et al., the aim of our research is different. Our driver model is deliberately kept simple in order to focus on the interaction between traffic controllers on the one hand, and between agents and traffic controller on the other hand. Furthermore, our framework is not developed in order to simulate the existing real-world scenario. Finally, our framework supports

decentralized traffic controllers while Baines et al. focus on a single, all-knowing, institution.

Another comparable line of research has been done by Balke et al. [7]. In this extended abstract, Balke et al. discuss the difference between off-line and on-line reasoning of institutions (similar to norm-based traffic controllers) governing open multi-agent systems. They state that most research up to that point had been focused on the off-line reasoning of institutions, which can be used to research the static properties of institutions. The on-line reasoning of institutions concerns the monitoring and controlling of agents, observing if norms are being violated and informing agents if this is the case. In this implementation, there is a single institution with the title "The Governor" with which agents can communicate and receive information regarding possible consequences of their actions.

Our approach is most related to the on-line reasoning as described by Balke et al. However, communication between the agents and the institution is handled in a different way. With our framework, the information provided by agents to the traffic controller is acquired via sensors. This is a more realistic representation of traffic situations, since it is often beneficial for the agent to not disclose any information about itself. Furthermore, in TrafficMAS, multiple traffic controllers are present, creating a more robust and better-controlled system through communication and coordination between these institutions.

9 Future Work

Our framework could be extended in a number of ways. One can add support for contrary to duty norms. Contrary to duty norms consist of a hierarchy of norms. An agent should comply with all hierarchies, but if it does not comply with the first norm level (and thus incurring a sanction), it should at least comply to the second norm level, or get an even higher sanction. An example is the norm "You shouldn't break the speed limit, but if you do, you should drive on the leftmost lane."

Secondly, we want to implement a full-fledged decentralized norm-based traffic controller. Currently TrafficMAS supports only decentralization of monitoring in the sense that sensor data can be shared. However, we want to also steer toward a system where norm instances, meta-analyses of sensor data, and sanction commands can be sent between traffic controllers.

Finally, a graphical user interface (GUI) could be added to (i) allow for easy creation of scenarios and (ii) allow for on the fly monitoring and editing of norms. For example, with a GUI one could investigate how the vehicle behavior changes when one changes the sanction severity of a norm. Because of our Model-View-Controller structure of the framework, this can be implemented by only altering the View part of our extension.

10 Conclusion

Our goal was to create a traffic simulation Multi-Agent System where vehicles should generally follow traffic regulations, yet are able to ignore these regulations in certain circumstances without implementing hard constraints on the agents themselves. We used norm-based traffic controllers since they can properly deal with these kinds of situations.

Our work is an extension to SUMO. It features a norm-based traffic controller which monitors and possibly sanctions the vehicles. We assume deliberative pro-active drivers that make autonomous decisions according to their goal and received sanctions. The extension features (i) driver profiles which model different types of behavior, (ii) traffic controllers and norms to control vehicles, and (iii) an easy way to add new driver profiles, traffic controllers and norms. This plug and play extension to SUMO can serve as a testing suite for all experiments concerning norm-based traffic control.

We showed in our experiments that the performance of normative systems is better than the default behavior in a ramp-merge scenario. Furthermore, we presented that complex norms allow for finer grained steering of behavior in complicated scenarios. Moreover, we illustrated the autonomy of drivers, by demonstrating a difference in behavior between poor and affluent drivers. Finally, we demonstrated the ability of traffic controllers to coordinate their activities, yielding better results in certain scenarios.

Appendix 1: Merge Norm Scheme

For the merge norm scheme, we use the same pseudocode structure (Algorithm 2) as for the stay-on-lane norm scheme. As with the other norm scheme we begin with the instance of the norm (lines 0–8).7 Initially we read sensors 1 and 3 and merge the readings using the algorithm of Wang et al. [21] (line 1). The result is an ordered list of agents, which, if they continue as they are, will arrive at the merge point in the same order. We maintain a global variable t_{free} that indicates the next moment in time that the merge point is free. With $optimalVelocity$ we calculate the optimal speed for an agent s.t. it will arrive at t_{free} on the merge point plus some safe margin, or later if the agent cannot make it in time physically (line 3). If the agent is at the right lane of the main road and the optimal velocity is below a predefined threshold, then it is obliged to move to the left lane (line 5), otherwise it is obliged to adapt its velocity to the optimal velocity and pass the merge point on the right lane (line 7). An agent is sanctioned if it is not passing the merge point on the correct lane (lines 11–12, and 15–16). Otherwise, an agent can also be sanctioned if it did not achieve its predetermined velocity (lines 19–20).

Algorithm 2 Pseudocode for the merge norm scheme

$L \leftarrow merge(read(s_1), read(s_3))$
for all $agent \in L$ **do**
  $s \leftarrow optimalVelocity(agent, t_{free})$
  if $agent.lane = right \& s < v_{threshold}$ **then**
    $N \leftarrow N \cup \{(agent, ((left, v_{MAX}), high))\}$
  else
    $N \leftarrow N \cup \{(agent, ((right, s), high))\}$
$L \leftarrow read(s_4)$
for all $agent \in L$ **do**
  $N \leftarrow N \backslash \{(agent, ((left, v_{MAX}), high))\}$
  if $(agent, (right, s), low)) \in N$ **then**
    $sanction(agent, high)$
$L \leftarrow read(s_5)$
for all $agent \in L$ **do**
  if $(agent, (left, v_{MAX}, high)) \in N$ **then**
    $sanction(agent, high)$
    $N \leftarrow N \backslash \{(agent, ((left, v_{MAX}), high))\}$
  else if $(agent, (right, s), high)) \in N$ **then**
    if $s \neq agent.v$ **then**
      $sanction(agent, high)$
    $N \leftarrow N \backslash \{(agent, ((right, s), high))\}$

Appendix 2: Using Our Code

Our framework is open-source and available on-line on Github at https://github.com/baumfalk/TrafficMAS. It can be compiled from source, or it can be downloaded as a binary version.

How to Run It

Our framework can be run as follows. Assuming you use the binary JAR file, a scenario can be run with the following command:

```
java -jar TrafficMAS.jar ./scen/ scenario.mas.xml
path/to/sumo scenario.sumocfg [seed].
```

In this command `scen` is the directory the scenario is located in, `scenario.mas.xml` is the main configuration file for the scenario and `path/to/sumo` denotes the SUMO executable to use. The SUMO-GUI program can also be used. The parameter `scenario.sumocfg` denotes the SUMO configuration file used by the scenario. Finally, the parameter `seed` is used to prepare the random number generator, which is used to spawn vehicles in a probabilistic fashion. If no seed is provided, a random one is generated by the system.

How to Create Your Own Scenario

Our framework also allows for the creation of your own scenarios. A TrafficMAS scenario consists of several XML files:

- a global configuration file, containing the paths to the other XML files, as well as the simulation duration.
- a configuration file specifying which norms are used. In this file, the norms are also parameterized with scenario-specific information, such as road names.
- a configuration file which describes the norm-based traffic controllers. The file is used to define which controllers there are, which sensors they have access to and to which other controllers they are subscribed.
- a configuration file containing the vehicle profile distributions. This file contains the distributions of the various driver profiles and the traffic density of the different roads.
- various SUMO XML files: the XML file containing the nodes, the edges, the sensors and the routes.

References

1. Abdelkader G (2003) Requirements for achieving software agents autonomy and defining their responsibility. In: Proceedings of the autonomy workshop at AAMAS, vol 236
2. Ackerman E (2015) Tesla working towards 90 percent autonomous car within three years. http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/tesla-working-towards-90-autonomous-car-within-three-years. Accessed 29 June 2015
3. Alechina N, Dastani M, Logan B (2012) Programming norm-aware agents. In: Proceedings of the 11th international conference on autonomous agents and multiagent systems-volume 2, International foundation for autonomous agents and multiagent systems, pp 1057–1064
4. Alechina N, Dastani M, Logan B (2013) Reasoning about normative update. In: Proceedings of the twenty-third international joint conference on artificial intelligence. AAAI press, pp 20–26
5. Baines V, Padget J (2014) On the benefit of collective norms for autonomous vehicles. In: Proceedings of 8th international workshop on agents in traffic and transportation
6. Baines V, Padget J (2015) A situational awareness approach to intelligent vehicle agents. In: Modeling mobility with open data. Springer, Berlin, pp 77–103
7. Balke T, De Vos M, Padget J, Traskas D (2011) On-line reasoning for institutionally-situated bdi agents. In: The 10th international conference on autonomous agents and multiagent systems-volume 3, International foundation for autonomous agents and multiagent systems, pp 1109–1110
8. Baskar LD, De Schutter B, Hellendoorn J, Papp Z (2011) Traffic control and intelligent vehicle highway systems: a survey. IET Intell Transp Syst 5(1):38–52
9. Boella G, Van Der Torre L, Verhagen H (2006) Introduction to normative multiagent systems. Comput Math Organ Theory 12(2–3):71–79
10. Dastani M, Grossi D, Meyer J-JC, Tinnemeier N (2009) Normative multi-agent programs and their logics. In: Knowledge representation for agents and multi-agent systems. Springer, Berlin, pp 16–31
11. Hübner JF, Boissier O, Bordini RH (2011) A normative programming language for multi-agent organisations. Ann Math Artif Intell 62(1–2):27–53

12. Kavathekar P, Chen Y (2011) Vehicle platooning: a brief survey and categorization. In: ASME 2011 international design engineering technical conferences and computers and information in engineering conference. American society of mechanical engineers, pp 829–845
13. Krajzewicz D, Erdmann J, Behrisch M, Bieker L (2012) Recent development and applications of sumo–simulation of urban mobility. Int J Adv Syst Meas 5(3–4)
14. Krauss S, Wagner P, Gawron C (1997) Metastable states in a microscopic model of traffic flow. Phys Rev E 55(5):5597
15. Markoff J (2015) Google cars drive themselves, in traffic. http://www.nytimes.com/2010/10/10/science/10google.html. Accessed 29 June 2015
16. Meneguzzi F, Vasconcelos W, Oren N, Luck M (2012) Nu-BDI: Norm-aware BDI agents. In: Proceedings of the 10th European workshop on multi-agent systems. Dublin, Ireland
17. Testerink B, Dastani M, Meyer J-J (2014) Norm monitoring through observation sharing. In: Proceedings of the European conference on social intelligence, pp 291–304
18. Testerink B, Dastani M, Meyer J-J (2014) Norms in distributed organizations. In: Coordination, organizations, institutions, and norms in agent systems IX. Springer, Berlin, pp 120–135
19. Tinnemeier NA, Dastani M, Meyer J-J, Torre L (2009) Programming normative artifacts with declarative obligations and prohibitions. In: Web intelligence and intelligent agent technologies. WI-IAT'09. IEEE/WIC/ACM international joint conferences on 2009, vol 2. IET, pp 145–152
20. van Riemsdijk MB, Hindriks K, Jonker C (2009) Programming organization-aware agents. In: Engineering societies in the agents world X. Springer, Berlin, pp 98–112
21. Wang Z, Kulik L, Ramamohanarao K (2007) Proactive traffic merging strategies for sensor-enabled cars. In: Proceedings of the fourth ACM international workshop on vehicular ad hoc networks. ACM, pp 39–48