Applied Intelligence (2022) 52:1808–1824
https://doi.org/10.1007/s10489-021-02449-5

Using ontology to guide reinforcement learning agents in unseen situations

A traffic signal control system case study

Saeedeh Ghanadbashi[1] · Fatemeh Golpayegani[2]

Accepted: 20 April 2021 / Published online: 29 May 2021

Abstract
In multi-agent systems, goal achievement is challenging when agents operate in ever-changing environments and face unseen situations, where not all the goals are known or predefined. In such cases, agents need to identify the changes and adapt their behaviour, by evolving their goals or even generating new goals to address the emerging requirements. Learning and practical reasoning techniques have been used to enable agents with limited knowledge to adapt to new circumstances. However, they depend on the availability of large amounts of data, require long exploration periods, and cannot help agents to set new goals. Furthermore, the accuracy of agents' actions is improved by introducing added intelligence through integrating conceptual features extracted from ontologies. However, the concerns related to taking suitable actions when unseen situations occur are not addressed. This paper proposes a new Automatic Goal Generation Model (AGGM) that enables agents to create new goals to handle unseen situations and to adapt to their ever-changing environment on a real-time basis. AGGM is compared to Q-learning, SARSA, and Deep Q Network in a Traffic Signal Control System case study. The results show that AGGM outperforms the baseline algorithms in unseen situations while handling the seen situations as well as the baseline algorithms.

Keywords Multi-agent Systems (MAS) · Goal generation · Autonomy · Reinforcement learning (RL) · Ontology · Traffic signal control system

1 Introduction

Multiple agents that interact in an environment and coordinate their behaviour to solve a complex task or achieve a goal are known as Multi-agent Systems (MAS). In dynamic environments, agents should be able to adapt to changes and self-generate goals based on the changing requirements autonomously. For example, in intelligent traffic signal control systems, traffic signals are controlled by autonomous and adaptive agents, and their goal is to minimise waiting time for all the vehicles. When an incident happens, it is expected that the system manages the traffic in a way that the ambulance could get to that location as soon as possible. In this situation, the traffic signal agents change their goals from "minimising the waiting time for all the vehicles" to "minimise the ambulance's waiting time". To do so, they need to identify the change in the environment, create a new goal, and take suitable actions accordingly.

Agents use several techniques to behave appropriately in an ever-changing environment [36]. Learning techniques are used to enrich agents' knowledge using big data or a long exploration period [12]. However, these techniques are not useful when real-time decision-making is desirable and not enough data is available. In the discussed example, if the current goal is defined as "minimising total waiting time", learning techniques cannot figure out how the goal should change when a new/unseen emergency situation happens unless they have a predefined goal to "minimise the ambulance waiting time".

✉ Saeedeh Ghanadbashi
saeedeh.ghanadbashi@ucdconnect.ie

Fatemeh Golpayegani
fatemeh.golpayegani@ucd.ie

1 Room G34, Computer Science Department, University College Dublin, Dublin, Ireland

2 Room 207, Computer Science Department, University College Dublin, Dublin, Ireland

Practical reasoning is also used to enable agents to infer knowledge from their environment and interaction with other agents [1]. Although reasoning allows agents to understand their environment, update their perceptions, and make decisions in real-time, they cannot help with setting new goals. In the example discussed earlier, the reasoning engine must have a predefined rule (e.g., context: emergency situation, action-specification: minimise the ambulance waiting time).

Goal generation approaches allow agents to choose a new goal from a pre-defined goal-set when the current goal should be changed [26]. Using Goal Reasoning (GR) techniques, agents continuously reason about the goals they are pursuing and if necessary, they adjust them according to their preferences [1]. There are two models of GR, Goal-Driven Autonomy (GDA) and Goal Refinement. In GDA, agents nominate a set of potentially suitable goals for the current situation. In Goal Refinement, agents generate one or more plans to achieve a given goal. When an unexpected event occurs, agents evaluate the goal and decide whether to pursue the goal, drop it, or resolve the detected event through one of the predefined strategies. Although GR agents can operate in complex systems with limited communications and adapt intelligently to changing conditions [21], they do not perform as well when they encounter an unseen situation.

The main research question this paper will address is how agents can adapt their behaviour when they experience an unseen situation. We propose an Automatic Goal Generation Model (AGGM) that allows agents to adapt their behaviour according to the changes in the environment by generating new goals to handle an unseen situation autonomously. In this model, agents continuously observe the state of their environment and evaluate the feedback they receive to decide whether there is a significant change in the environment and consequently if the current goal should be changed. A significant change can be interpreted either when there is a discrepancy with an expected reward, or in anticipation that a goal must be changed because of an unseen situation. To react to an identified significant change, the agent can replace the current goal with one of the predefined goals in its goal-set, and if there are not any suitable predefined goals, agents take actions that plausibly result in experiencing a previously known state. A traffic signal control system scenario is chosen as a case study for this paper, where AGGM's performance is compared to Q-learning, SARSA (State–Action–Reward–State–Action), and Deep Q Network (DQN). The results show that AGGM outperforms the baselines when handling unseen situations such as emergency and congestion cases.

The remainder of the paper is organised as follows. Section 2 presents a review of the relevant literature. Section 3 briefly presents the background knowledge, Section 4 presents AGGM. In Section 5, the case study is described and in Section 6 the experimental scenarios are defined. In Section 7 the results are analysed. Finally, our conclusion and future works are discussed in Section 8.

2 Related work

In an ever-changing environment, agents need to be adaptive and autonomously revise their understanding of their environment and other agents' behaviour to take suitable actions. Goal formation, goal generation, and goal reasoning techniques are used by an adaptive agent to handle their environments' unpredictability. Goal formation is a process of reaching from unachievable goals through instrumental beliefs to concrete ones [8]. In the goal generation process, goals are generated from conditional beliefs, obligations, intentions, and desires or motivations based on agent's preferences [5, 22], and [26]. Additionally, goals can be generated when an agent detects discrepancies between its sensory inputs and its expectations [27]. Goal reasoning is a case-based system that uses active and interactive learning to automatically select goals from a set of predefined goals [35]. In these works, a new goal is generated when an agent's belief is changed. However, the goal generation process when the environment's state is inconsistent with the agent's beliefs is not discussed.

In practical reasoning, agents' desires and preferences are used during goal generation [22] and [20]. Social reasoning is also used to enhance agents' understanding of others' goals and their dependencies [15]. Although reasoning is an effective approach for real-time decision-making, it requires predefined reasoning rules to handle unseen situations.

Learning techniques are also used to enhance agents' performance when having access to limited knowledge. In [12], a reverse method for gradual learning from a set of start states nearby the goal is described. In [11], agents learn the forward task and reset policy together, which resets the environment for a subsequent attempt. In [16], agents learn a world model and a self model simultaneously. The world model is used to predict the dynamic consequences of the agent's actions, and the self model estimates the world model's errors to be explored in the future.

When the reward function is unknown, agents face a difficult challenge [33]. Imitation Learning (IL) enables agents to reach any goal without any need for reward. This approach extracts additional information from the demonstrations, so it can leverage demonstrations that do not include expert actions [9]. Moreover, the Inverse Reinforcement Learning (IRL) agents can infer a reward function from the expert demonstrations and assume that the expert policy is optimal regarding the unknown reward function [17].

Although learning strategies can guide agents toward their goals using reward or fitness functions, they perform poorly when such functions are not available.

3 Background

This section briefly presents the required background information.

3.1 Reinforcement learning

In this paper, agents use RL as a learning method, however, AGGM does not depend on RL and other learning methods can be used instead. RL is a trial-and-error method in which the agents learn through interaction with the environment in such a way that an agent chooses the action according to its policy, which is subsequently sent to the environment, then the environment moves to a new state and an immediate reward will be sent to the agent. The agents use the action with the most positive reward repeatedly [10]. We use Q-learning, SARSA, and DQN as RL approaches with which AGGM's performance is compared.

Q-learning Q-learning is a fundamental RL method in which an agent computes the Q-value that estimates discounted cumulative reward of actions as displayed in (1). R_t^{Π} (see Appendix for description of symbols) is discounted cumulative reward at time step t under policy Π. γ is a discount rate that shows how much the future rewards contribute to the total reward, and r_τ is the reward at a time step τ. The agent becomes farsighted if γ values are closer to 1 and the agent becomes shortsighted if γ values are closer to 0 [40].

$$R_t^{\Pi} = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau, \gamma \in [0, 1) \tag{1}$$

Q-value for state s and action a is represented as $Q(s, a)$, and its value is calculated as shown in (2):

$$Q(s,a) = Q(s,a) + \rho[R(s,a,s') \\ + \gamma \max Q'(s',a') - Q(s,a)] \tag{2}$$

$Q(s, a)$ is current stored Q-value for applying action a in state s. $R(s, a, s')$ is the reward value the agent gets from the environment after doing action a in state s which takes the environment to the new state s'. $\max Q'(s', a')$ is the maximum expected future reward. The learning rate ρ determines in which degree the new information overrides the old one [14]. The Q-learning algorithm uses Q-table to store Q-values, then uses this data structure to calculate the maximum expected future rewards.

SARSA To compute Q-value, Q-learning computes the difference between $Q(s, a)$ and the maximum action value $Q'(s', a')$, while on-policy SARSA algorithm learns action values relative to the policy it follows [40] as shown in (3):

$$Q(s,a) = Q(s,a) + \rho[R(s,a,s') + \gamma Q'(s',a') - Q(s,a)] \tag{3}$$

Deep Q Network The DQN algorithm uses a deep neural network and the input of the neural network would be the state that the agent is in and the targets would be the Q-values of each of the actions [37].

The RL algorithm applies an exploration (i.e., exploring action space) and exploitation (i.e., performing the best action) scheme to compute a policy that maximises the payoff. The Epsilon-Greedy algorithm is used to balance the exploration and exploitation in choosing actions, and ε is the percentage dedicated to the exploration [40].

3.2 Traffic signal control systems

Traffic signal control is an important technical means using computer and information technology to adjust the signal timing parameters to improve the traffic flow [43]. In [24], the authors categorise traffic signal control methods based on artificial intelligence into the following categories: fuzzy control technology, Artificial Neural Networks (ANN), evolutionary algorithms (e.g., genetic algorithm, ant colony algorithm, and particle swarm optimisation), and RL methods.

Automatic traffic signal control can be managed by agents with learning capabilities controlling traffic signals [36] and [4]. Agents use interaction and communication protocols and ontologies to negotiate and make decisions [36, 42], and [4]. This system is particularly chosen as a case study for this paper as in traffic systems it is not possible to predict all the events and consequently facing unseen situations is inevitable.

3.3 Ontology

Ontology provides user-contributed, augmented intelligence, and machine-understandable semantics of data [13]. Ontologies are used to enhance agents' learning process by providing semantic models [31], augmented feedback loop to optimise their overall accuracy [6], and as a way to share knowledge between agents [41]. Additionally, agents can generate their own ontologies by observing others' actions and automatically derive logical rules that represent the observed behaviour [29] and [18].

We propose each agent represents its observation using a schema described by an ontology. The ontology development 101 strategy [34] and the ontology editing environment Protégé [32] can be used to develop an ontology. Also, ontology Verification & Validation (V&V) following the SABiO guidelines are used to evaluate an ontology (i.e., to identify missing or irrelevant concepts) [3]. In [18], Semantic Sensor Network (SSN) ontology is proposed to describe sensor resources, and the data they collect as observations. It bridges the gap between low-level data streams coming from sensors in real-time and high-level concepts used by agents to interpret an observation (i.e., high-level semantic representations). The ontology-based schema is used when a semantic description is needed and is composed of surrounding concepts and relations between concepts perceived by agents, which are modelled as observations. These relations enable inheritance between concepts and automated reasoning. We define $L^t_{g_i}$ as the schema describing the data observed by agent g_i at time step t. C represents the set of concepts, and M represents the set of relations over these concepts (see (4)). The domain and range of a relation determine what kind of instances it can be used for (i.e., domain) and what kind of values it can have (i.e., range).

$$L^t_{g_i} = \{C^t_{g_i}, M^t_{g_i}\} \tag{4}$$

A graph representation of our ontology is generated using OntoGraf plug-in in Protégé, and the inference rules are expressed using Semantic Web Rule Language (SWRL) [19]. There are two methods of reasoning when using inference rules [38]:

- **Forward reasoning:** It starts from state observation and applies inference rules to extract more facts until reaches the goal. For example, we can conclude from "A" and "A implies B" to "B".
- **Backward reasoning:** It starts from the goal and chaining through inference rules to find the required facts that support the goal. For example, we can conclude from "not B" and "A implies B" to "not A".

To model the concepts in traffic signal control context, we use the ontology proposed in [30] (see Fig. 1).

4 Automatic goal generation model

In this paper we assume our agents to be adaptive and use the RL algorithm. When unseen situations occur, the agents continuously observe their environment, identify the changes, and have access to a mechanism that enables them to decide whether to continue with their RL process, change their current goal to a pre-defined goal, or generate a new goal. AGGM enables agents to make such decisions in the following stages: in the *Observe* stage, agents continuously observe the environment state, update their own state, and the reward associated with their previous action. The agents then evaluate these states and the received rewards in the *Evaluate* stage, and in the *Significant Change Identification* stage agents decide whether or not the current goal needs

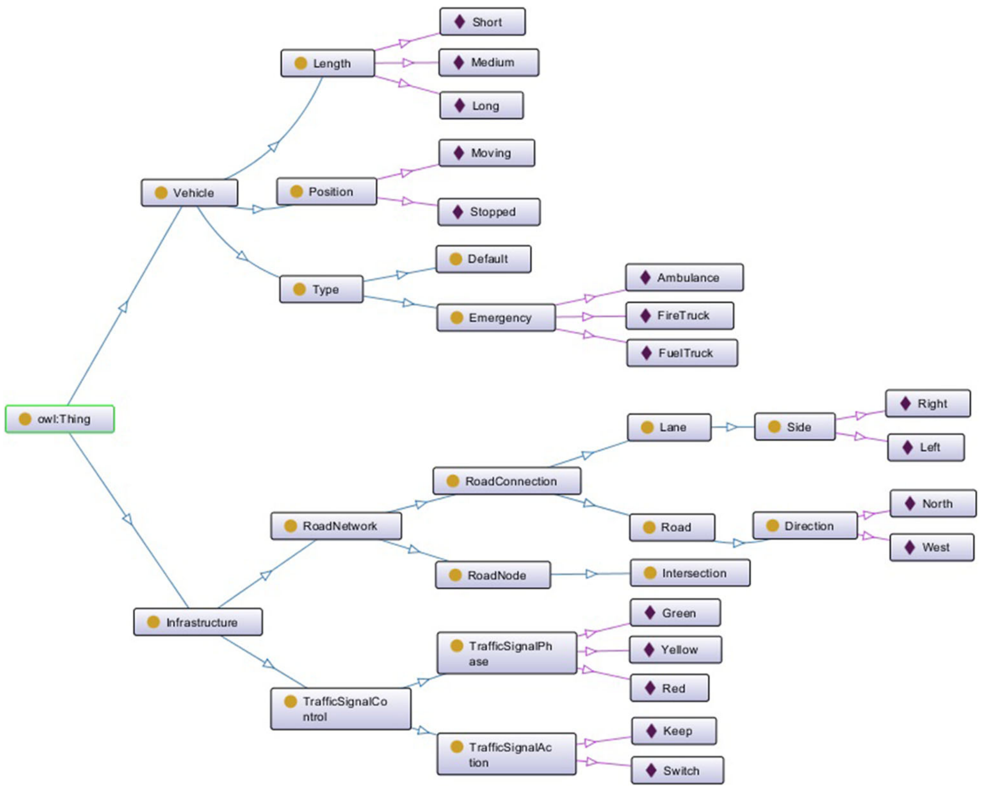

Fig. 1 Ontology for traffic signal control, as represented by OntoGraf

to be revised. If so, in the *Reasoning* stage the current goal is changed to a predefined goal, or a new goal will be generated. The output from this stage results in a new reward function that will be sent to the *Generate Action* stage to generate suitable actions accordingly. Then these actions will be executed in the *Execute Action* stage. This process continues by sending the next state and reward of current action back to the agents until the environment sends a terminal state or a specific number of iterations happens (see AGGM in Fig. 2 and algorithmic procedure in Algorithm 1). The details of the model components are explained below.

Algorithm 1 Automatic Goal Generation Model decision process.

```
1: while $Stop\ condition\ has\ not\ reached$ do
2:     $o^t_{g_i}$ ←Observe()
3:     $Q^t_{g_i} \leftarrow Q(s^t_{g_i}, a)$
4:     $D^t_{g_i} \leftarrow |V_{s^{t-1}_{g_i}} - V_{s^t_{g_i}}|$
5:     $iw^t_{g_i} = \sum_{x \in s^t_{g_i}} iw_c(x)$
6:     $\{iw^t_{g_i}\} = \{iw_c(x) \mid \forall x \in s^t_{g_i}\}$
7:     if Significant-Change-Identification($Q^t_{g_i}, D^t_{g_i}, iw^t_{g_i}$) = True then
8:         $I$ ←Match-with-Predefined-Goals($s^t_{g_i}$,goal-set)
9:         if $I$ ! = ∅ then
10:            $G$ ←Choose-Most-Appropriate-Goal($P_{g_i}, \{iw^t_{g_i}\}, I$)
11:            $B^t_{g_i}$ ←Update-Problem-specific-Reward($G$)
12:        else
13:            $J^t_{g_i}$ ←Generate-State-Similarity-Reward($L^t_{g_i}, s^t_{g_i}, s^{t-1}_{g_i}$)
14:        end if
15:    end if
16:    $a$ ←Generate-Action($s^t_{g_i}$,F($B^t_{g_i}, J^t_{g_i}$),$A$)
17:    Execute-Action($a$)
18: end while
```

4.1 Observe

The agents constantly observe the environment to identify new changes and update their perception of the environment, the states, and the reward they received for their previous actions. S^t is the individual states of all n agents in the system at time step t. For example, $s^t_{g_i}$ is the state of agent g_i at time step t.

$$S^t = \{s^t_{g_1}, s^t_{g_2}, \ldots, s^t_{g_i}, \ldots, s^t_{g_n}\} \tag{5}$$

Agent g_i's observation $o^t_{g_i}$ is a tuple including its state $s^t_{g_i}$ and the reward $r^t_{g_i}$ received from the environment at time step t. When an observation is received, it will be sent to the *Evaluate* stage.

$$o^t_{g_i} = \left(s^t_{g_i}, r^t_{g_i}\right) \tag{6}$$

4.2 Evaluate

Agent g_i evaluates its observation using the Q-value, the state distance, and the importance of the observation.

Q-value Using the reward $r^t_{g_i}$ received from the environment, the agent calculates the Q-value, $Q^t_{g_i}$ (line 3 of Algorithm 1).

The state distance The agent computes $D^t_{g_i}$ the absolute difference between the current state $s^t_{g_i}$ and the previous state $s^{t-1}_{g_i}$. To do so, we define $V_{s^t_{g_i}}$ to be a quantifying value that describes $s^t_{g_i}$ (line 4 of Algorithm 1).

The importance of observation Each agent g_i observes the environment at time step t based on its ontology $L^t_{g_i}$, so the importance of each observation $s^t_{g_i}$ is determined based on the importance of the concepts $C^t_{g_i}$ involved (line 5 of Algorithm 1). A concept weighting function is used to quantify the degree of importance of each concept $x \in C^t_{g_i}$ in a domain using an iweighting indicator [28] (see (7)):

$$iw_c(x) = 1/|M(x)| \sum_{m \in M(x)}^{|M(x)|} iw^{(x,y)}_{M_m} \tag{7}$$

The iweighting indicator, denoted by $iw_c(x)$ is a numerical value derived from weighting the local context of concept x based on its outgoing edges (i.e., relations to other concepts). To compute concepts' weight, the relations

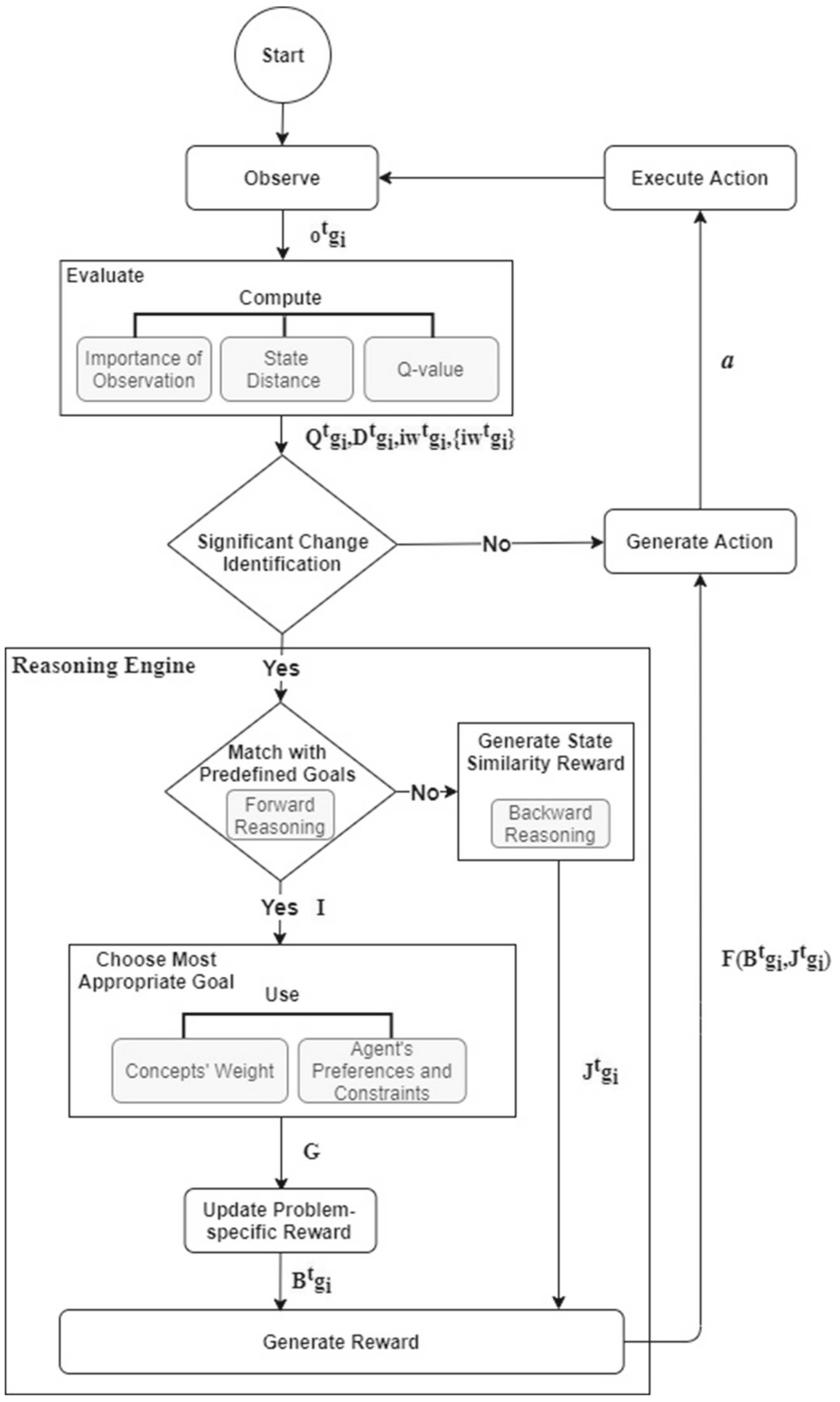
Start
Observe
Execute Action
$o^t g_i$
Evaluate
Compute
Importance of Observation
State Distance
Q-value
a
$Q^t g_i, D^t g_i, iw^t g_i, \{iw^t g_i\}$
Significant Change Identification
No
Generate Action
Reasoning Engine
Yes
Match with Predefined Goals
Forward Reasoning
No
Generate State Similarity Reward
Backward Reasoning
Yes
I
$F(B^t g_i, J^t g_i)$
Choose Most Appropriate Goal
Use
Concepts' Weight
Agent's Preferences and Constraints
$J^t g_i$
G
Update Problem-specific Reward
$B^t g_i$
Generate Reward

Fig. 2 Automatic Goal Generation Model

are initially weighted manually by ontology engineers during the ontology development process, and $iw_c(x)$ is calculated based on the average importance weights of the relations $m \in M^t_{g_i}$ of domain concept x constrained by their particular range y. For example, in traffic context, “Vehicle” (i.e., domain) “has type” (i.e., relation) and can be an “Emergency” one (i.e., range), and a “Highest Importance” value (i.e., importance weight) can be assigned to a vehicle of emergency type (i.e., relation and its domain/range combination). There are five degrees of importance weights, which can be converted to numerical values using predefined mappings (see Table 4): “Lowest Importance”, “Low Importance”, “Middle Importance”, “High Importance”, and “Highest Importance”.

4.3 Significant change identification

Based on the output from the *Evaluate* stage an agent decides whether there is a need to change its current goal or create a new one (see the algorithmic procedure in Algorithm 2), so three cases are possible:

- **Case 1.** When $Q^t_{g_i}$ is out of the predefined range ($Q^t_{g_i}$ < Discrepancy-Low-Threshold or $Q^t_{g_i}$ > Discrepancy-High-Threshold) (line 2 of Algorithm 2). Discrepancy-Low-Threshold and Discrepancy-High-Threshold define the minimum and maximum values of Q-values that can be received.
- **Case 2.** When $D^t_{g_i}$ is bigger than the predefined State-Difference-Threshold which is the maximum difference between successive environment states (line 4 of Algorithm 2).
- **Case 3.** When a concept x with a high importance weight $iw_c(x)$ appears in an environment or the importance of observation $iw^t_{g_i}$ becomes higher than a predefined Importance-Weight-Threshold which is the maximum importance weight that has been experienced for the current goal (line 6 of Algorithm 2).

Algorithm 2 Significant-Change-Identification ($Q^t_{g_i}$, $D^t_{g_i}$, $iw^t_{g_i}$).

```
1: result ← False
2: if $Q^t_{g_i}$ > Discrepancy-High-Threshold || $Q^t_{g_i}$ < Discrepancy-Low-Threshold then
       result ← True
3: end if
4: if $D^t_{g_i}$ > State-Difference-Threshold then
       result ← True
5: end if
6: if $iw^t_{g_i}$ > Importance-Weight-Threshold then
       result ← True
7: end if
8: Return result
```

4.4 Reasoning

Using the *Reasoning Engine*, agent g_i continuously reasons about the goals it is pursuing, when it is required to change or generate a new goal, two cases are possible:

- **Choosing a predefined goal.** An agent uses inference rules to deduce a predefined goal from a goal-set through the forward reasoning (line 8 of Algorithm 1). The goal-set specifies tuples of (S, G) where G is a goal that can be adopted when observation S is observed. When more than one goal in the goal-set is consistent with $s^t_{g_i}$, agent's preferences and constraints P_{g_i} or the concepts' weight $\{iw^t_{g_i}\}$ will be used as decision criteria (line 10 of Algorithm 1). Finally, the problem-specific reward function $B^t_{g_i}$ is updated with the selected goal G (line 11 of Algorithm 1). For example, consider the two following alternatives rules in the goal-set:

  - **rule 1:** $c_1 = k_1, c_2 = k_2, ...c_5 = k_5, ..., c_n = k_n- > g_1$
  - **rule 2:** $c_1 = k_1, c_2 = k_2, ...c_6 = k_6, ..., c_n = k_n- > g_2$

  c_i is a concept and k_i is its value in observation S. The agent's observation $s^t_{g_i}$ is consistent with both rules. If iweighting indicator $iw_c(c_6)$ is higher than iweighting indicator $iw_c(c_5)$ then g_2 will be selected.
- **Creating a new goal.** When agent g_i cannot find a suitable goal, a state similarity reward function $J^t_{g_i}$ which is the inverse of the difference between $s^t_{g_i}$ and $s^{t-1}_{g_i}$ is defined (line 13 of Algorithm 1). Reducing the difference between $s^t_{g_i}$ and $s^{t-1}_{g_i}$ leads to an increase in the state similarity reward (see (8)).

$$J^t_{g_i} = 1/|V_{s^{t-1}_{g_i}} - V_{s^t_{g_i}}| \tag{8}$$

  Agent g_i uses backward reasoning to maximise $J^t_{g_i}$. Suppose an unseen situation occurs when ambulance a enters intersection s monitored by agent g_i, according to the inference rules shown in Table 1, the agent maximises the position coordinates of ambulance a until it passes through the intersection, thereby, the environment will be reverted to a known previous state. Table 2 shows another example of using backward reasoning to reduce congestion in road *r1* by maximising the position coordinates of all instances of vehicle b on the road *r1* (i.e., hasPosition(?b, Moving)). A schematic overview of the backward reasoning process is shown in Fig. 3.

Finally, agent g_i selects an action based on the recommendation of the function that combines the two reward functions $B^t_{g_i}$ and $J^t_{g_i}$ (line 16 of Algorithm 1). Depending on the problem, various combinations can be defined for these two reward functions. In this paper, we define a priority function which prioritises $J^t_{g_i}$ over $B^t_{g_i}$. Therefore, agent g_i prioritises maximising the state similarity reward over the problem-specific reward optimisation, and takes actions that can contribute to experiencing its previous known state (see Fig. 4). To do so, agent g_i selects actions that minimise the difference between $s^t_{g_i}$ and $s^{t-1}_{g_i}$.

4.5 Generate and execute action

Using the function F($B^t_{g_i}$, $J^t_{g_i}$), based on the current state $s^t_{g_i}$, agent g_i selects the appropriate action a from action space A and executes it (lines 16 and 17 of Algorithm 1).

Table 1 An example of inference rules, inferring maximising the position coordinates of the ambulance a as a parameter in the state similarity reward function J^t_{gi}

Inference rules
TrafficSignalControl(?i), Intersection(?s), Road(?r1), Lane(?l1), Vehicle(?a),
isOn(?a, ?l1), consistOf(?r1, ?l1), atIntersection(?i, ?s), isRegulatedBy(?r1, ?i),
hasPosition(?a, Stopped)
− >
atIntersection(?a, ?s)

In Semantic Web Rule Language (SWRL), variables are indicated using the standard convention of prefixing them with a question mark

5 Traffic signal control case study

In the multi-agent traffic signal control systems, the traffic signal at each intersection is controlled by an independent agent. Agents observe and analyse the traffic collected data and decide their actions accordingly. In the remainder of this section, we present how our model is tested using a traffic micro-simulator, Simulation of Urban MObility (SUMO) which provides a microscopic real-time traffic simulation [25].

5.1 Simulation setting

SUMO is employed to evaluate the performance of AGGM in a traffic signal control case study. The whole simulated traffic network is a $750m \times 750m$ area. Each intersection is a $300m \times 300m$ area. Thus, the total number of intersections is 16 (see Fig. 5). At each intersection, we have two incoming roads and two exit roads. Each road is marked with a name such as 0to1 and includes two lanes, for example, road 0to1 includes two lanes 0to1_0 and 0to1_1. So, we have eight lanes at each intersection in which vehicles drive. The lane length is 120 meters. The vehicles in incoming roads from west-to-east are allowed to take right-turn and pass through traffic. The vehicles in incoming roads from north-to-south are allowed to take left-turn and pass through traffic. The minimal gap between the two vehicles is 2.5 meters. We have four types of vehicles in the simulation: default, ambulance, fuel truck, and trailer truck. The length of default and ambulance vehicles is 5 meters and the length of the fuel truck and trailer truck is 10 meters. The default vehicles arrive in the environment following a random process, and the arrival rate of every lane is the same, one per second. The arrival rate of other types of vehicles is according to scenarios discussed in Section 6. Vehicles are discarded if they could not be inserted. For all types of vehicles, the max speed is 55.55 m/s, which is equal to 200 km/h. Also, the max accelerating acceleration is 2.6 m/s^2 and the decelerating acceleration is 4.5 m/s^2. SUMO uses the Krauss Following Model [23], which guarantees safe driving on the road. The duration of yellow phase is set to 2 seconds. The minimum duration of green phase is set to 5 seconds and the maximum one is set to 100 seconds (Max-Green-Time). The number of simulation seconds ran before learning begins is set to 300 seconds. The number of simulated seconds on SUMO is set to 1,000 seconds. The simulation seconds between actions are set to 5 seconds. We used the interface to instantiate RL environments with SUMO for traffic signal control provided by [2] to interact with the traffic signal-controlled intersections.

We perform the simulation through 10 runs for each scenario. One run is an episode of 1,000 seconds. The reward is accumulated in an episode. The goal in our network is to maximize the reward in each 1,000 seconds episode by modifying the traffic signals' phases. The simulation results show the average values obtained from 10 runs and are compared to the baseline algorithms. The parameters of the network are shown in Table 3.

Table 2 An example of inference rules, inferring maximising the position coordinates of all instances of vehicle b as a parameter in the state similarity reward function J^t_{gi}

Inference rules
TrafficSignalControl(?i), Intersection(?s), Road(?r1), Lane(?l1), Vehicle(?b),
isOn(?b, ?l1), consistOf(?r1, ?l1), atIntersection(?i, ?s), isRegulatedBy(?r1, ?i),
hasPosition(?b, Stopped)
− >
hasCongestion(?r1)

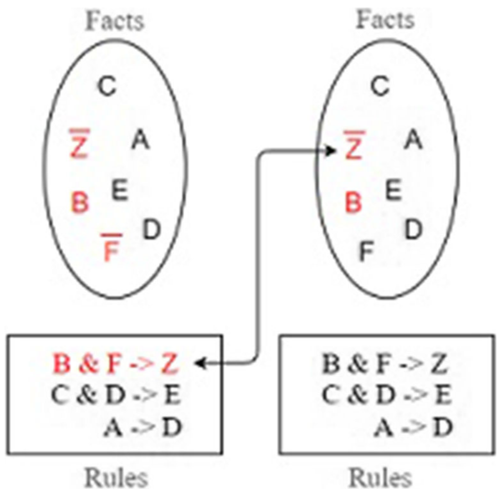

Fig. 3 An example of the backward reasoning process, Z shows the fact atIntersection(?a, ?s) and $\overline{\text{F}}$ shows the inferred fact hasPosition(?a, Moving). B shows the other facts in Table 1 such as consistOf(?r1, ?l1), atIntersection(?i, ?s), and isRegulatedBy(?r1, ?i)

5.2 Baseline algorithms

To build the traffic signal control system using RL, we need to define the states, actions, and rewards. The three elements are defined in the following:

- *States*: We model information of each state $s^t_{g_i}$ for traffic signal g_i at time step t as follows:

$$s^t_{g_i} = \{\Phi^t_{g_i}, e^t_{g_i}, q^t_{l_i}, z^t_{l_i}, y^t_{v_i}, b^t_{v_i}, w^t_{v_i}\} \tag{9}$$

  - Yellow, red and green *phase indicators* are shown as $\Phi^t_{g_i}$ for the intersection monitored by g_i at time step t.
  - Current *phase elapsed time*, is shown as $e^t_{g_i}$ for the intersection monitored by g_i at time step t and is computed by (10). u is the time duration from start of the current phase up to now.

$$e^t_{g_i} = u/\text{Max-Green-Time} \tag{10}$$

  - Current *lane queue*, the number of vehicles waiting in each lane divided by the lane capacity, is shown as $q^t_{l_i}$ for lane l at the intersection monitored by g_i at time step t and is computed using (11). h_l is the total number of halting vehicles for the last time step on lane l (a speed of less than 0.1 m/s is considered a halt), e_l is the length of lane l in meters and f is the sum of the vehicle length and the minimum gap.

$$q^t_{l_i} = \min(1, (h_l/(e_l/f))) \tag{11}$$

  - Current *lane density*, the number of vehicles in each lane divided by the lane capacity, is shown as $z^t_{l_i}$ for lane l at the intersection monitored by g_i at time step t and is computed using (12). n_l is the number of vehicles on lane l within the last time step.

$$z^t_{l_i} = \min(1, (n_l/(e_l/f))) \tag{12}$$

  - The *type of vehicle* is shown as $y^t_{v_i}$ for vehicle v at the intersection monitored by g_i at time step t.
  - The *position coordinates of vehicle* along the lane (the distance from the front bumper to the start of the lane) is shown as $b^t_{v_i}$ for vehicle v at the intersection monitored by g_i at time step t.
  - The *waiting time of a vehicle* counts the number of seconds a vehicle has a speed of less than 0.1 m/s and is shown as $w^t_{v_i}$ for vehicle v at the intersection monitored by g_i at time step t.
- *Action Space*: Traffic signal phases in the action space include green, yellow, and red phases (i.e., 1, 2, and 3 indicating green, yellow, and red respectively). The green phase is the period during which vehicles are permitted to cross. The yellow phase is required between two neighbouring phases to guarantee safety. The red phase is the period during which vehicles are not allowed to cross.
- *Rewards*: The main goal here is to increase the efficiency of the intersection, by minimising the vehicles' waiting time. Therefore, the reward will be the amount of change in the cumulative waiting time between two consecutive cycles (see (13)). During the training period, the RL algorithm tries different signal control schemes and eventually converges to an optimal scheme which yields a minimum average waiting time.

$$B^t_{g_i} = \sum w^{t-1}_{v_i} - \sum w^t_{v_i} \tag{13}$$

Unseen situations We define the arrival of ambulances, fuel trucks, and trailer trucks and a line of more than 10 vehicles at an intersection as unseen situations.

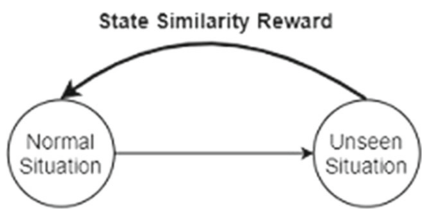

Fig. 4 Handling an unseen situation in the Automatic Goal Generation Model, the functionality of state similarity reward

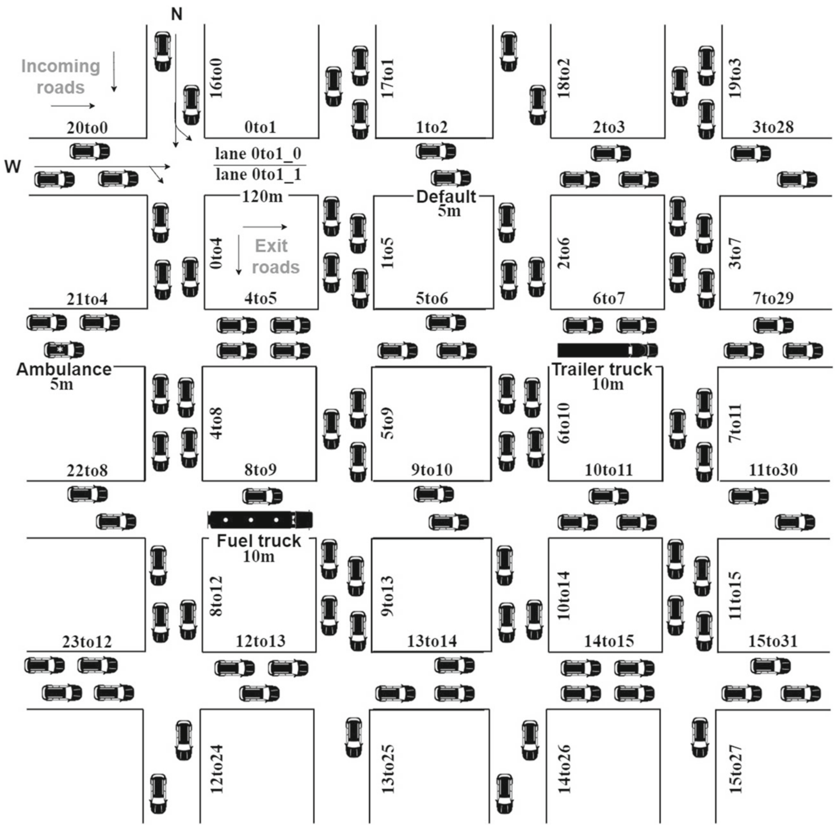

Fig. 5 Simulated traffic network in SUMO

5.3 AGGM in SUMO

The stages of AGGM in the traffic signal control system are as follows:

- *Observe*: We model each traffic signal's observation as a tuple which includes the traffic signal's state $s^t_{g_i}$ (see (9)) and the traffic signal's reward $r^t_{g_i}$ received from the environment at time step t. The reward functions include the problem-specific reward $B^t_{g_i}$ which is explained in (13) and the state similarity reward $J^t_{g_i}$ as displayed in (14). Agent g_i uses backward reasoning over its ontology's inference rules (see Tables 1 and 2) and deduces the state similarity reward as maximising the position coordinates of vehicles.

$$J^t_{g_i} = \sum (b^t_{v_i} - b^{t-1}_{v_i}) \tag{14}$$

So, the traffic signal's reward is a set which includes reward computed using $B^t_{g_i}$ and $J^t_{g_i}$ (15).

$$r^t_{g_i} = \{B^t_{g_i}, J^t_{g_i}\} \tag{15}$$

We defined a combination of the two reward functions $B^t_{g_i}$ and $J^t_{g_i}$ in the form of the priority function by giving more priority to the output of the state similarity reward function.

- *Evaluate and Significant Change Identification*: The traffic ontology includes the importance weights of the relations and their domain/range combination (see Table 4). Traffic signal agent g_i computes the importance weight $iw_c(r_i)$ of the road r_i by $\sum_{v_i \in r_i} iw_c(v_i)$, to identify whether there has been a significant change in the environment (see line 6 of Algorithm 2). When congestion happens or an important vehicle (e.g., an ambulance, a fuel truck,

Table 3 Parameter settings of the baseline algorithms

Parameter	Value
ε	0.05 (Q-learning, SARSA, Deep Q Network)
γ	0.99 (Q-learning), 0.95 (SARSA), 0.99 (Deep Q Network)
ρ	1e−1 (Q-learning), 1e−9 (SARSA), 1e−3 (Deep Q Network)

Table 4 Relations between concepts in traffic signal control ontology

Relation	Domain	Range	Importance weight	Numerical value
hasPosition	Vehicle	Moving	Lowest Importance	0
hasPosition	Vehicle	Stopped	Low Importance	1
hasType	Vehicle	Default	Lowest Importance	0
hasType	Vehicle	Emergency	Highest Importance (e.g., ambulance)	10
			High Importance (e.g., fuel truck)	5
hasLength	Vehicle	Short	Lowest Importance	0
hasLength	Vehicle	Medium	Lowest Importance	0
hasLength	Vehicle	Long	Middle Importance (e.g., trailer truck)	2

or a trailer truck) enters the intersection on road r_i in the current state at time step t, the $iw_c(r_i)$ value becomes larger than Importance-Weight-Threshold, and the *Reasoning Engine* will be triggered.

- *Reasoning and Execute Action*: The state similarity reward $J^t_{g_i}$ is defined as the sum of the relocation of the vehicles in two consecutive states (see (14)). The greater the state similarity reward is the closer the vehicles that have led to the significant change get to the intersection. AGGM along with the RL algorithm are used to generate the traffic signal's action (see Algorithm 3):
  - *AGGM:* When there is a significant change, agent g_i will select an action a which maximises the value of the state similarity reward, and discards the action suggested by the RL algorithm (line 5 of Algorithm 3).
  - *RL:* Action a will be suggested based on the problem-specific reward function (line 7 of Algorithm 3).

Algorithm 3 Generate-Action.

```
1: $iw_c(r_i) \leftarrow \sum_{v_i \in r_i} iw_c(v_i)$
2: $r_{max} \leftarrow \arg\max_{\forall r_i \in s^t_{g_i}} iw_c(r_i)$
3: if $iw_c(r_{max}) >$Importance-Weight-Threshold then
4:     $J^t_{g_i} \leftarrow \sum_{v_i \in r_{max}} (b^t_{v_i} - b^{t-1}_{v_i})$
5:     $a \leftarrow \arg\max_{\forall a \in A} J^t_{g_i}$
6: else
7:     $a \leftarrow \arg\max_{\forall a \in A} B^t_{g_i}$
8: end if
```

6 Experimental scenarios

Vehicles' **average waiting time**, is used to measure the efficiency of AGGM and baseline algorithms (i.e., Q-learning, SARSA, and DQN), and it is calculated as follows:

$$w^t_{g_i} = 1/|v_i| \sum w^t_{v_i} \tag{16}$$

The evaluation scenarios used to test the performance of AGGM are listed in Table 5. **Scenarios 1, 2, and 3** generate important vehicles (i.e., ambulances, fuel trucks, and trailer trucks) in specified or random roads in *Low* and *High* frequencies. In *Low* frequency setting, 3 important vehicles are generated per 2 minutes, and in *High* frequency, 5 important vehicles are generated per 2 minutes. **Scenario 4** creates congestion at specific roads per 3 minutes.

7 Results and discussion

The results report the average waiting time for all types of vehicles in 10 runs in each scenario. As shown in Fig. 6, AGGM performs almost as well as the baseline algorithms when the average waiting time of default vehicles is compared in scenarios 1, 2, and 3, and outperforms the baseline algorithms in scenario 4. As shown in Figs. 7, 8, and 9, AGGM significantly decreases the average waiting time of ambulances, fuel trucks, and trailer trucks compared to the baseline algorithms.

As shown in Table 6, the decrease in average waiting when AGGM is applied is significant compared to the baseline algorithms, however, AGGM is more effective in low-frequency settings in more complex scenario (i.e., scenario 3).

As reported in Table 6, we observe that the improvement of AGGM compared to the baseline algorithms in scenario 1 where ambulances, fuel trucks, and trailer trucks are generated in parallel roads is more than scenario 2 where they are generated in intersecting roads. This is because the important vehicles entering in intersecting roads creates a more complex situation. Also, our approach shows a better performance in more complex scenarios where unseen situations can happen simultaneously in multiple parallel or intersecting roads (i.e., scenario 3) and it does not perform as well in simple scenarios where congestion happens with a line of more than 10 vehicles at an intersection (i.e., scenario 4).

From Table 6, we observe that AGGM decreases the waiting time of ambulances more than the other types of

Table 5 Evaluation scenarios

Scenario	Description	Freq.
1	Generates important vehicles in four parallel roads:	Low
	0to1, 4to5, 9to10, 14to15.	High
2	Generates important vehicles in three pairs of intersecting roads:	Low
	(0to4, 21to4), (5to9, 8to9), (10to14, 13to14).	High
3	Generates important vehicles at random roads.	Low
		High
4	Creates traffic congestion at even-numbered intersections (e.g., 0, 2, ...).	

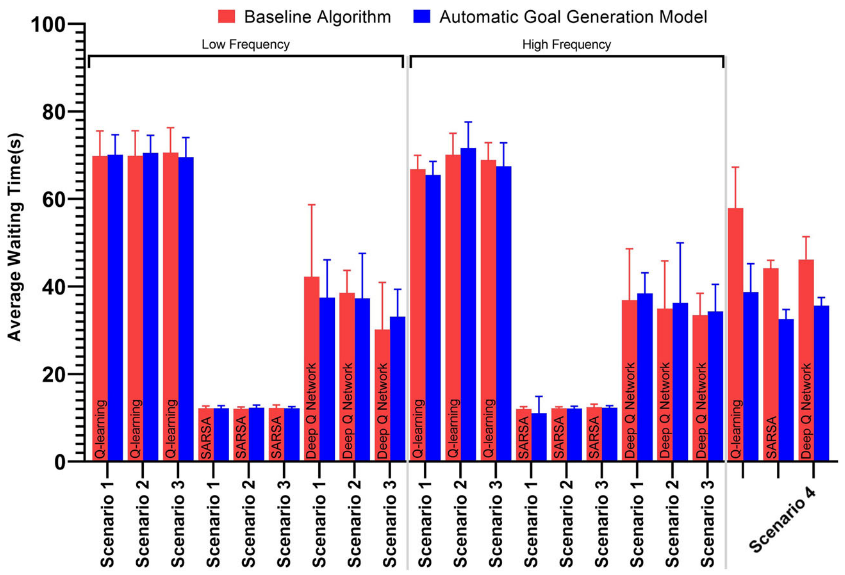

Fig. 6 The average waiting time of default vehicles –AGGM and the three baselines

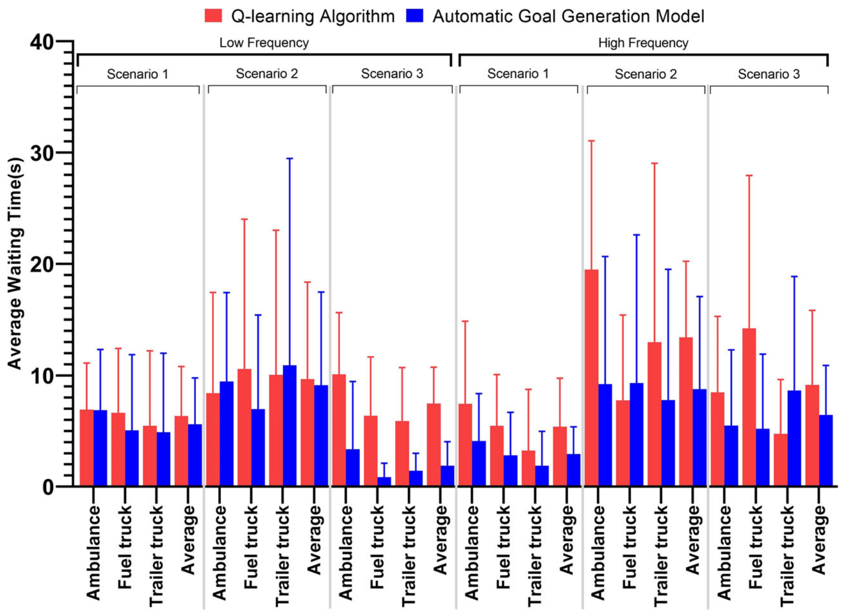

Fig. 7 The average waiting time of important vehicles – AGGM and Q-learning

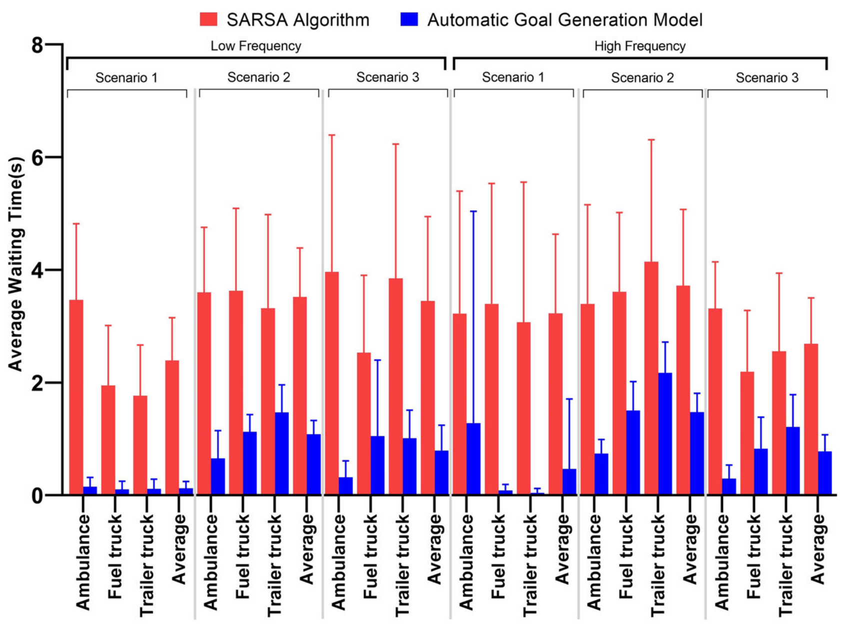

Fig. 8 The average waiting time of important vehicles – AGGM and SARSA

vehicles in scenario 3. Also, in scenario 2, the improvement for fuel trucks is higher than the other types of important vehicles. Since in scenarios 2 and 3, important vehicles can enter the environment in intersecting roads, the higher priority is given to the road that has the most important vehicles. Therefore, when an ambulance or a fuel truck is observed on one road and a trailer truck on another road, the road with the ambulance or fuel truck will have a higher priority and AGGM reverts the road to a familiar previous state using the state similarity reward function.

Finally, from waiting time results for all types of vehicles in the system, we can conclude that the performance of the default vehicles is not compromised to increase the performance of ambulances, fuel trucks, and trailer trucks in the first three scenarios (see Fig. 6). This is particularly an important result as AGGM can handle unseen situations while keeping the performance of the system for seen situations.

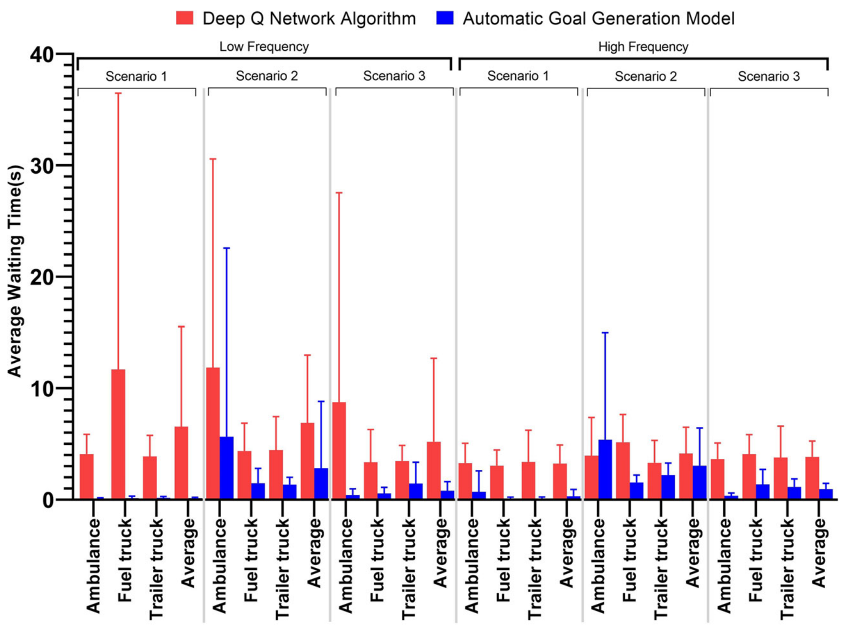

Fig. 9 The average waiting time of important vehicles – AGGM and DQN

Table 6 Decrease in average waiting time – AGGM and baseline algorithms

Typeofvehicle	Scenario	Freq.	Q-learning	SARSA	DQN	Avg.
Ambulance	1	Low	1%	96%	98%	65%
		High	45%	60%	79%	61%
	2	Low	–12%	82%	52%	41%
		High	53%	78%	-37%	31%
	3	Low	67%	92%	95%	85%
		High	35%	91%	91%	72%
Fuel truck	1	Low	24%	95%	99%	73%
		High	49%	98%	97%	81%
	2	Low	34%	69%	66%	56%
		High	-20%	58%	70%	36%
	3	Low	87%	58%	84%	76%
		High	63%	63%	67%	64%
Trailer truck	1	Low	11%	94%	96%	67%
		High	42%	99%	97%	79%
	2	Low	-9%	56%	69%	39%
		High	40%	48%	33%	40%
	3	Low	76%	74%	59%	70%
		High	-82%	53%	70%	14%
Avg. Ambulance, Fuel truck, and Trailer truck	1	Low	12%	95%	98%	68%
		High	46%	85%	91%	74%
	2	Low	6%	69%	59%	45%
		High	35%	60%	26%	40%
	3	Low	75%	77%	85%	79%
		High	30%	71%	75%	59%
Default	4	–	33%	26%	23%	27%

8 Conclusion

In real-world environments, it is not possible to predict all possible events in advance, therefore when an agent faces unseen situations or events it will not be able to efficiently show suitable behaviour. In this paper, we attempted to address such environments where agents might face unseen situations. We proposed an Automatic Goal Generation Model in which agents are enabled to detect unseen situations and handle them automatically. In such situations, agents either replace their current goal with another predefined goal or generate a new goal. AGGM is evaluated in a traffic signal control system case study with varying levels of frequency of unseen situation occurrence. The results are compared to several baseline algorithms including Q-learning, SARSA, and DQN and the results show that our method performs significantly better in detecting and handling unseen situations (e.g., emergency and congestion situations).

This paper can be extended in several directions. The state similarity reward was defined particularly for the signal control system case study using the position coordinates of vehicles as a primary parameter. However, defining the state similarity reward might be more complicated when multiple types of unseen events are possible in an environment. To address this issue, we used ontology models to formulate states and their distinguishing parameters. So, AGGM's performance depends on the accuracy and completeness of the used ontology. Additionally, operating in dynamic environments requires the ontology to evolve and update frequently. Ontology evolution techniques [39] can be used to address this issue. The new goal generation model only concerns about reverting the environment to a familiar state that has been experienced before, however, this might be more challenging when there exist multiple states including familiar and inexperienced states that can contribute to addressing an unseen situation. To address this issue ontology evolution techniques can be used. High-level policies help agents adapt their goals according to different situations of the environment. Generative Policy-based Models (GPM) can be used to enable agents to observe, learn, and adapt high-level policy models [7].

Appendix Symbols description

Symbol	Description
R_t^{Π}	Discounted cumulative reward at time step t under policy Π.
γ	Discount rate.
r_τ	Reward at time step τ.
$Q(s, a)$	Q-value for applying action a in state s.
$R(s, a, s')$	Reward value the agent gets from the environment after doing action a in state s which takes the environment to the new state s'.
$\max Q'(s', a')$	Maximum expected future reward.
ρ	Learning rate.
A	Action space.
s	Environment state.
ε	Probability to choose random action in Epsilon-Greedy algorithm.
$L_{g_i}^t$	Ontology used by agent g_i at time step t.
$C_{g_i}^t$	The set of concepts observed by agent g_i at time step t.
$M_{g_i}^t$	The set of relations used by agent g_i at time step t .
S^t	Individual states of all the agents in the system at time step t.
$s_{g_i}^t$	State of agent g_i at time step t.
$r_{g_i}^t$	Reward received from the environment by agent g_i at time step t.
$o_{g_i}^t$	Observation of agent g_i at time step t.
$Q_{g_i}^t$	Q-value for applying action a in state $s_{g_i}^t$.
$D_{g_i}^t$	Absolute difference between the current state $s_{g_i}^t$ and the previous state $s_{g_i}^{t-1}$.
$V_{s_{g_i}^t}$	Quantifying value that describes $s_{g_i}^t$.
$iw_{g_i}^t$	The importance of observation $s_{g_i}^t$.
$\{iw_{g_i}^t\}$	The importance of concepts $C_{g_i}^t$ involved in $s_{g_i}^t$.
I	The goals in goal-set which are consistent with the agent's observation $s_{g_i}^t$.
G	A selected goal in goal-set.
P_{g_i}	Preferences and constraints of agent g_i to select goal.
c_i	A concept in observation S.
k_i	A concept's value in observation S.
$B_{g_i}^t$	Problem-specific reward for agent g_i at time step t.
$J_{g_i}^t$	State similarity reward for agent g_i at time step t.
$\mathrm{F}(B_{g_i}^t, J_{g_i}^t)$	Combination of the two reward functions $B_{g_i}^t$ and $J_{g_i}^t$.
$\Phi_{g_i}^t$	Phase indicator for the intersection monitored by agent g_i at time step t.
$e_{g_i}^t$	Phase elapsed time for the intersection monitored by agent g_i at time step t.
u	Time duration from start of current phase up to now.
$q_{l_i}^t$	Lane queue for lane l at the intersection monitored by agent g_i at time step t.
h_l	Total number of halting vehicles for the last time step on lane l.
e_l	Length of lane l in meters.
f	Sum of the vehicle length and the minimal gap.
$z_{l_i}^t$	Lane density for lane l at the intersection monitored by g_i at time step t.
n_l	Number of vehicles on lane l within the last time step.
$y_{v_i}^t$	Type of vehicle v at the intersection monitored by agent g_i at time step t.
$b_{v_i}^t$	Position coordinates of vehicle v at the intersection monitored by agent g_i at time step t.
$w_{v_i}^t$	Waiting time of vehicle v at the intersection monitored by agent g_i at time step t.
$w_{g_i}^t$	Average waiting time of vehicles at the intersection monitored by agent g_i at time step t.

Funding Open Access funding provided by the IReL Consortium.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Aha DW (2018) Goal reasoning: Foundations, emerging applications, and prospects. AI Mag 39(2):3–24
2. Alegre LN (2019) SUMO-RL https://github.com/LucasAlegre/sumo-rl
3. Almeida Falbo R, Menezes CS, Rocha ARC (1998) A systematic approach for building ontologies. In: Ibero-american conference on artificial intelligence (IBERAMIA). Springer, pp 349–360
4. Bailey JM, Golpayegani F, Clarke S (2019) Comasig: a collaborative multi-agent signal control to support senior drivers. In: IEEE Intelligent transportation systems conference (ITSC). IEEE, pp 1239–1244
5. Broersen J, Dastani M, Hulstijn J, van der Torre L (2002) Goal generation in the BOID architecture. Cognit Sci Quarter (CSQ) 2(3-4):428–447
6. Caruana G, Li M, Liu Y (2013) An ontology enhanced parallel SVM for scalable spam filter training. Neurocomputing 108:45–57
7. Cunnington D, Manotas I, Law M, de Mel G, Calo S, Bertino E, Russo A (2019) A generative policy model for connected and autonomous vehicles. In: IEEE Intelligent transportation systems conference (ITSC). IEEE, pp 1558–1565
8. Dignum F, Conte R (1997) Intentional agents and goal formation. In: International workshop on agent theories, architectures, and languages (ATAL). Springer, pp 231–243
9. Ding Y, Florensa C, Abbeel P, Phielipp M (2019) Goal-conditioned imitation learning. In: Conference on neural information processing systems (NIPS), pp 15,298–15,309
10. Dorri A, Kanhere SS, Jurdak R (2018) Multi-agent systems: A survey. IEEE Access 6:28,573–28,593
11. Eysenbach B, Gu S, Ibarz J, Levine S (2017) Leave no trace: Learning to reset for safe and autonomous reinforcement learning. Computing Research Repository (CoRR). arXiv:1711.06782
12. Florensa C, Held D, Wulfmeier M, Zhang M, Abbeel P (2017) Reverse curriculum generation for reinforcement learning. In: Annual conference on robot learning (coRL). PMLR, pp 482–495
13. Fong ACM, Hong G, Fong B (2019) Augmented intelligence with ontology of semantic objects. In: International conference on contemporary computing and informatics (IC3i). IEEE, pp 1–4
14. François-Lavet V, Fonteneau R, Ernst D (2015) How to discount deep reinforcement learning: Towards new dynamic strategies. Computing Research Repository (CoRR). arXiv:1512.02011
15. Golpayegani F, Dusparic I, Clarke S (2019) Using social dependence to enable neighbourly behaviour in open multi-agent systems. ACM Trans Intell Syst Technol (TIST) 10(3):1–31
16. Haber N, Mrowca D, Fei-Fei L, Yamins DL (2018) Learning to play with intrinsically-motivated, self-aware agents. In: Conference on neural information processing systems (NIPS), pp 8388–8399
17. Hadfield-Menell D, Milli S, Abbeel P, Russell SJ, Dragan A (2017) Inverse reward design. In: Conference on neural information processing systems (NIPS), pp 6765–6774
18. Haller A, Janowicz K, Cox SJ, Lefrançois M., Taylor K, Le Phuoc D, Lieberman J, García-castro R, Atkinson R, Stadler C (2019) The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation. Semantic Web 10(1):9–32
19. Horrocks I, Patel-Schneider PF, Boley H, Tabet S, Grosof B, Dean M (2004) SWRL: A Semantic web rule language combining OWL and ruleML. W3C Member Submission 21(79):1–31
20. Jaidee U, Muñoz-Avila H, Aha DW (2011) Integrated learning for goal-driven autonomy. In: International joint conference on artificial intelligence (IJCAI). IJCAI/AAAI, pp 2450–2455
21. Johnson B, Floyd MW, Coman A, Wilson MA, Aha DW (2018) Goal reasoning and trusted autonomy. In: Foundations of trusted autonomy. Springer, Cham, pp 47–66
22. Kondrakunta S, Gogineni VR, Molineaux M, Munoz-Avila H, Oxenham M, Cox MT (2018) Toward problem recognition, explanation and goal formulation. In: Goal reasoning workshop at IJCAI/FAIM
23. Krauß S (1997) Towards a unified view of microscopic traffic flow theories. Int Federat Autom Control (IFAC) Proc 30(8):901–905
24. Liu Z (2007) A survey of intelligence methods in urban traffic signal control. Int J Comput Sci Netw Secur (IJCSNS) 7(7):105–112
25. Lopez PA, Behrisch M, Bieker-Walz L, Erdmann J, Flötteröd YP, Hilbrich R, Lücken L, Rummel J, Wagner P, WieBner E (2018) Microscopic traffic simulation using sumo. In: IEEE Intelligent transportation systems conference (ITSC). IEEE, pp 2575–2582
26. Luck M, d'Inverno M (1995) Goal generation and adoption in hierarchical agent models. In: Australasian joint conference on artificial intelligence (AJCAI). World scientific
27. Maynord M, Cox MT, Paisner M, Perlis D (2013) Data-driven goal generation for integrated cognitive systems. In: AAAI Fall symposium series. AAAI Press
28. Mazak A, Schandl B, Lanzenberger M (2010) Iweightings: Enhancing structure-based ontology alignment by enriching models with importance weighting. In: International conference on complex, intelligent and software intensive systems (CISIS). IEEE, pp 992–997
29. Monticolo D, Lahoud I, Bonjour E (2012) Distributed knowledge extracted by a MAS using ontology alignment methods. In: International conference on computer & information science (ICCIS). IEEE, pp 386–391
30. Morignot P, Nashashibi F (2012) An ontology-based approach to relax traffic regulation for autonomous vehicle assistance. Computing Research Repository (CoRR). arXiv:1212.0768
31. Motta JA, Capus L, Tourigny N (2016) Vence: a new machine learning method enhanced by ontological knowledge to extract summaries. In: Science and information (SAI) computing conference. IEEE, pp 61–70
32. Musen MA (2015) The protėgė project: A look back and a look forward. AI Matters 1(4):4–12
33. Nguyen TT, Nguyen ND, Nahavandi S (2018) Deep reinforcement learning for multi-agent systems: A review of challenges,

solutions and applications. Computing Research Repository (CoRR). arXiv:1812.11794
34. Noy NF, McGuinness DL et al (2001) Ontology development 101: A guide to creating your first ontology. Tech. rep., Stanford Knowledge Systems Laboratory. https://protege.stanford.edu/publications/ontology_development/ontology101.pdf
35. Powell J, Molineaux M, Aha DW (2011) Active and interactive discovery of goal selection knowledge. In: International florida artificial intelligence research society (FLAIRS) conference. AAAI Press
36. Rezzai M, Dachry W, Moutaouakkil F, Medromi H (2018) Design and realization of a new architecture based on multi-agent systems and reinforcement learning for traffic signal control. In: International conference on multimedia computing and systems (ICMCS). IEEE, pp 1–6
37. Sewak M (2019) Deep Q Network (DQN), double DQN, and dueling DQN. In: Deep reinforcement learning. Springer, pp 95–108
38. Sharma T, Tiwari N, Kelkar D (2012) Study of difference between forward and backward reasoning. Int J Emerg Technol Adv Eng (IJETAE) 2(10):271–273
39. Stojanovic L (2004) Methods and tools for ontology evolution. Ph.D. thesis, Karlsruhe Institute of Technology, Germany. http://digbib.ubka.uni-karlsruhe.de/volltexte/1000003270
40. Sutton RS, Barto AG (2018) Reinforcement learning: An introduction. MIT Press, Cambridge
41. Thanh-Tung D, Flood B, Wilson C, Sheahan C, Bao-Lam D (2006) Ontology-MAS for modelling and robust controlling enterprises. In: International conference on theories and applications of computer science (ICTACS), pp 116–123
42. Tomás VR, Garcia LA (2005) A cooperative multiagent system for traffic management and control. In: International joint conference on autonomous agents and multiagent systems (AAMAS). ACM, pp 52–59
43. Wang Y, Yang X, Liang H, Liu Y (2018) A review of the self-adaptive traffic signal control system based on future traffic environment. J Adv Transport (JAT) 1–12

Saeedeh Ghanadbashi is a Ph.D. candidate at the School of Computer Science, University College Dublin. She received the MS.c degree in Computer Science from Sharif Technical University. Her areas of interest include intelligent agent-based models, reinforcement learning, ontology, smart city, and autonomous vehicles. Currently, her research project aims to study and identify the research gaps of the current goal generation/evolution, goal reasoning, and cognitive reasoning techniques in the context of open multi-agent systems.

Fatemeh Golpayegani received the MS.c degree from Sharif Technical University and the Ph.D. degree in Computer Science from Trinity College Dublin. She is currently an Assistant Professor in the School of Computer Science, University College Dublin, where she leads the Multi-agent Systems and Sustainable Solutions research group. Her research interests are in intelligent agent-based models for dynamic and autonomous decision-making in large-scale and mobile environments.