A MARL Approach for Optimizing Positions of VANET Aerial Base-Stations on a Sparse Highway

BOTE JIANG, SIDNEY N. GIVIGI, JR., (Senior Member, IEEE), AND JEAN-ALEXIS DELAMER, (Member, IEEE)

1School of Computing, Queen’s University, Kingston, ON K7L 3N6, Canada
2Department of Computer Science, St. Francis Xavier University, Antigonish, NS B2G 2W5, Canada

Corresponding author: Bote Jiang (13bj7@queensu.ca)

This work was supported in part by the Natural Sciences and Engineering Council of Canada under the Discovery Grant Program.

ABSTRACT A Vehicular Ad-Hoc Network (VANET) helps vehicles send and receive environmental and traffic information, making it a crucial component towards fully autonomous roads. For VANETs to serve their purpose, there has to be sufficient coverage, even in areas where there is less demand. Moreover, a lot of the safety information is time-sensitive; excessive delay in data transfer can increase the risk of fatal accidents. Unmanned Aerial Vehicles (UAVs) can be used as mobile base-stations to fill in gaps of coverage. The placement of these UAVs is crucial towards how well the system performs. We are particularly interested in the placement of mobile base-stations for a rural highway with sparse traffic, as it represents the worst-case scenario for vehicular communication. Instead of heuristic or linear programming methods for optimal placement, we use multi-agent reinforcement learning (MARL). The main benefit of MARL is that it allows the agents to learn model-free through experience. We propose a variation of the traditional Deep Independent Q-Learning. The modifications include an observation function augmented with information directly shared between neighbouring agents as well a shared policy scheme. We also implement a custom sparse highway simulator that is used for training and testing our algorithm. Our testing shows that the proposed MARL algorithm is able to learn the placement policies that produce the maximum rewards for different scenarios while adapting to the dynamic road densities along the service area. Our experiments show that our model is scalable, allowing the number of agents to increase without any modifications to the code. Finally, we show that our model can be generalized as the algorithm can be directly used on an industry standard simulator with similar performance. Future experiments can be performed to improve the realism and complexity of the highway models as well as to test the method on real-world data.

INDEX TERMS Automated highways, autonomous agents, unmanned aerial vehicles, end-to-end delay, multi-agent reinforcement learning, reinforcement learning, drones, VANET, vehicular network.

I. INTRODUCTION

By 2030, experts predict that autonomous vehicles will displace most human driving [1]. Fully autonomous roads will require sophisticated networking solutions to facilitate vehicular communications [2]. One popular solution is Vehicular Ad-Hoc Network or VANET technology. This not only encompasses vehicle-to-vehicle (V2V) communications but also vehicle-to-infrastructure (V2I) communications. Infrastructures are often in the form of roadside units (RSU) which include radio towers, dishes, or even small cellular base-stations. The purpose of VANETs is to help vehicles send and receive crucial environmental and traffic information as well as provide internet connection for passengers. This is important for enhancing safety features such as collision avoidance, road surface prediction, and improving efficiency of travel. For VANETs to function, there has to be sufficient coverage even in less populated areas where there is less demand. Moreover, a lot of the safety information is particularly time-sensitive, making excessive delays potentially fatal. For safe and efficient operation, there is usually a maximum delay limit that has to be satisfied. As a result, the placement of these RSUs (radio towers) is crucial towards the effectiveness of a VANET system.

The delay in message transmission usually occurs during its way to the RSUs [3]. We can theoretically minimize the delay by populating the entire highway segment with sufficient RSUs. However, its high deployment costs make this option infeasible, especially in rural regions where there is low usage demand [4]. There have been a number of heuristic
and meta-heuristic algorithms that tackle the problem of optimal placement. In [5], the authors model the placement problem as a set-covering problem. They proposed an algorithm called “Greed Set Cover” to solve the problem in polynomial-time. In [6], the problem was reformulated as an integer linear program. The authors proposed a center particle swarm optimization approach, in which a center particle is adopted for increasing the convergence speed. These works, however, all consider RSUs as fixed units that cannot be moved during runtime.

Unmanned aerial vehicles (UAVs) or drones can be equipped with communication devices. Integration of UAVs in wireless communication systems has started to attract attention. Their main benefit is the flexibility in positioning. Ideas like this have already been implemented, for example, UAVs have been used to assist the transmission and relay of cellular wireless networks [7]. When implemented in VANETs, UAVs can act as mobile base-stations.

A special case is with highways in rural regions with sparse road traffic. In rural areas, highways and regional roads are less likely to be in coverage of an RSU due to high costs. Furthermore, low vehicle densities will mean that the cars cannot rely on vehicle-to-vehicle (V2V) communications and would have to depend on vehicle-to-infrastructure (V2I) connections to transmit and receive crucial safety information. Seliem et al. [8] studied how delay can be modelled under a sparse environment and proposed a Drone Active Service (DAS) to adjust the distance between the drone base-stations to provide sufficient coverage. For a highway service area with multiple aerial base-stations, DAS would place these drones equidistant from each other. This is acceptable if road density is consistent across the entire highway segment. However, if different parts of the service area had different road densities that also varied from one time-step to another, there will be the need of a more complex algorithm that can adjust to the different density zones.

Reinforcement learning (RL) is a category of machine learning algorithms that has garnered a lot of attention over the past decade [9]. In RL, a controllable entity or agent interacts with its environment and receives information in return in the form of states and rewards [10]. Through training, the agent will map actions to states and will try to maximize long-term rewards. Deep Reinforcement Learning uses neural networks to approximate these mappings. This allows RL to be utilized on problems with more advanced state and action representations. As a result, an agent can learn to accomplish tasks through experience and without supervision. RL has been used to solve various types of VANET problems. Xiao et al. [11] used a policy-hill based algorithm to learn a UAV relay strategy to help with VANET jam resistance. Wu et al. [12] proposed a routing protocol for urban VANETs called RSU-assisted Q-learning-based Traffic-Aware Routing (QTAR).

VANETs can leverage the use of multiple agents. Multi-agent reinforcement learning (MARL) is, as its name suggests, RL for more than one agent. However, separate agents can only see part of the of the environment and have to rely on their own observations to make actionable decisions. A major challenge is for agents to adapt to the changing behaviour of other agents due to their partial observability.

MARL has been successfully applied towards vehicular network problems. For example, MARL has been used in urban traffic light control [13], resource allocation [14], and efficient routing of messages to minimize packet delay [15]. MARL has also been used for cellular networks to improve coverage and quality of service for users [16].

A. MOTIVATION AND CONTRIBUTION

Drones equipped base-stations can be optimally spread out along a highway with uniform density, as proposed by the Drone Active Service (DAS) [8]. However, instead of using a heuristic algorithm, a MARL approach can be applied to allow the drones to adjust their positioning through experience, even on a highway with dynamically changing road densities. Despite the works about MARL in vehicular networks, there have been no published results, to our knowledge, that directly uses MARL to find optimal positions of drone base-stations along a sparse highway.

The main contribution of this work is a MARL-based algorithm that uses limited traffic and road information to minimize end-to-end delay of a segment of highway (also called a service area). We evaluate our model in four ways:

1) Feasibility: The agents should be able adapt to the traffic situation and place themselves in a way that maximizes the segments of the highway that satisfy a delay constraint of 55 seconds.
2) Scalability: We also show that increasing and decreasing the number of agents in the system can be accomplished with only needing minimal re-training.
3) Robustness: We show that the system is able to quickly re-stabilize after single-point and dual-point failures.
4) Generalizability: We show that our MARL algorithm can learn similarly as well on an industry standard simulator as it does on our custom simulator.

It is important to note that this study does not test the algorithm’s readiness for deployment as doing so would require access to real-world data, something that we do not have. As a result, we used simulated traffic. Therefore, our work is primarily a proof of concept to show the feasibility and benefits of MARL on this category of problem.

B. PAPER STRUCTURE

This paper is organized as follows. In Section II, we begin by going over some of the related works in RL. Then in Section III, we define the problem and the notation to be used throughout this paper. In Section IV, we introduce the proposed MARL algorithm for solving this problem. Subsequently, in Section V, we present the simulation design. In Section VI, the results of the four experiments demonstrating the feasibility of the algorithm are provided. Finally, the conclusions and future work are discussed in Section VII.
II. RELATED WORKS

The simplest way to introduce more agents into the problem is to create a centralized entity that can control all of the agents collectively using a singular policy. While simple in theory, it introduces several problems. First, it requires a global state that is accessible to the central agent, something that is rarely achievable in practice. Also, the action space will grow exponentially as the number of actionable agents grow, leading to the curse of dimensionality [17].

A more scalable approach is treating each agent as independently learning entities. Independent Q-Learning (IQL) [18] is one of the most popular MARL approaches due to its similarities with single agent Q-learning. In this algorithm, each agent will have its own Q-network and each agent will condition its policy on its own partial observation of the global state and action. In works such as [12], [18] and [19], tabular IQL was used due to the simplistic representation of the states. To leverage the benefits of Deep Reinforcement Learning, Foerster et al. [20] have tried MARL variations of DQN. Like its single agent counterpart, Deep IQL also utilizes an experience replay buffer. However, since there are multiple agents learning independently of each other, the learning target is always moving. Therefore, a sample from the replay buffer will no longer represent the state and agents’ policies. This creates the problem of non-stationarity.

There have been many ways to alleviate the problem of non-stationarity. Foerster et al. [21] have tried to disable the replay buffer while Leibo et al. [22] have tried to selectively use more recent samples. Both of these methods were not sample efficient, which counteracts one of the main benefits of having a replay buffer. With IQL as a basis, [20] also proposed two methods to allow the agents to adjust towards non-stationary data. The first method uses importance sampling to decay old data and the second method labels the samples with the iteration number (or epsilon value) to disambiguate a sample’s age.

Another type of algorithm is explored by the works of [23] and [24] in which there is centralized learning and decentralized execution. This uses actor critic methods as a starting point in which the actor will select an agent’s actions while the critic will critique the policy that is followed by the actor. In this camp of MARL algorithms, each agent will only know a piece of the global state while a centralized controller, which knows the entire state space, will act as the critic. In Multi-Agent Deep Deterministic Policy gradients MADDPG, there is an individual critic for each agent, while in counterfactual multi-agent policy gradients (COMA), there is one large critic. A downside is that there has to be a centralized system that is capable of receiving and transmitting information to and from the agents, and also the input space of the critics dynamically changes with the number of agents in the system, limiting the scalability of the system.

While there have been no previous works applying MARL on optimizing drones for sparse road networks, we can look at other MARL vehicular and cellular network problems. In vehicular networks, Calvo and Dusparic [13] used Deep IQL to optimize traffic flow by controlling traffic lights. Their MARL involves Dueling Double Deep Q-Networks (DDDQNs) with prioritized experience replay and uses fingerprinting as way to mitigate non-stationarity. Wu et al.’s work minimized end-to-end delay on urban roads by routing the vehicles so that it efficient used the coverage of pre-placed RSUs [25]. Their MARL approach used Q-tables, did not require an experience replay and did not require any specific steps to deal with non-stationarity. While not for VANETs, Hammami et. al’s work involved controlling drone positions to optimize cellular networks. Since their environment only contained 6 cells to service, their representation was small enough to also use a tabular Q-learning method [16].

While it is not the focus of this work to produce a realistic environment to test readiness for real-world deployment, it is worthwhile to ensure that the highway and delay model used in our experiments is representative of what is seen in the literature.

For example, a number of papers [26], [27] have dealt with UAV-assisted VANETs in urban areas. It is valuable to look at the model design for other problems in this domain. Like our work, these papers all used short-range UAVs to enhance the connectivity of a VANET system. Furthermore, these works also assumed that the drones have high line of sight probability and disregard the degradation of the signal from the UAVs to the traffic. The main difference is in road design. While their work used a grid $4 \times 4$ km$^2$ in size, our work uses a straight highway. However, this is due to the different setting of the problem (urban vs. rural). Like our work, the Oubatti et. al used simulated traffic data and not real-world data. For performance metrics like end-to-end delay, Oubatti et. al used Network Simulator 2 (NS2) to generate values rather than estimations [27].

Other work [8], [28], [29] are more in line with our study, proposing different UAV assisted VANET solutions on sparse highway conditions. These works employed a simplistic highway design comprised of two lanes (forward and backward), assuming constant density throughout the entire service area (highway segment). These works also used SUMO to generate the constant density of road vehicles at a static highway speed. Our work is different in that, the road densities vary before and after every junction, therefore creating multiple density zones. For end-to-end delay, Sellem et al. [8] proposed an estimation scheme that uses numerical parameters such as distance, density and junction frequency. These parameters are detected in real-time from the traffic simulator and it is shown that the delay estimations produce analogous results to those generated by a network simulator such as NS-2.

III. PROBLEM STATEMENT

In this section, we will go over how the problem is designed.

A. HIGHWAY DESIGN

In this study, the highway model design is inspired by the work of [8]. We consider a 2D bi-directional straight highway.
The position units of the highway is discretized into 100 m bins. The segment of that highway that needs service has length $l$. There are two RSUs binding this area with positions of 0 and $l$. In Fig. 1, the RSUs are represented by red circles.

On the highway, there are also road junctions that are distributed randomly. In Fig. 1, the junctions are represented by vertical black lanes. Vehicles may leave or join the highway causing the road density to vary between different segments before and after a junction. In addition, we assume that the number of road junctions within the highway follows a Poisson distribution with a parameter $\lambda_c$ and a vehicle can leave the highway at any road junction with a probability $P_c$. For this study, every junction has a $P_c$ of 0.1.

For this problem, it is considered that there are $N$ number of infrastructure drone base-stations capable of receiving real-time information from the vehicles on the road. In Fig. 1, drones are represented as white circles. It is assumed that the drones and vehicles communicate using the DSRC 802.11p protocol. These drones are also assumed to be low altitude platform (LAP) multi-rotor drones with communication ranges $d_r$ of 500 m. We can assume that the transmission between a vehicle and a drone happens as soon as it enters the drone’s communication range. The actual latency of transmission once the vehicle is in communication range is negligible. We assume that the altitude for the drones is fixed at an arbitrary height $h$. As for the distribution of the drones, the $N$ drones can be distributed any way along the length of the road as long as they satisfy several constraints:

1) All drone-drone and drone-RSU distances must be at least $2d_r$ apart.
2) Positions are also discretized into 100 meter bins.

The drones can change their placement and the inter-drone distance based on the current highway state. Their movement is limited to discretized steps of step size $\text{step}$ and is constrained so that the position of drone $i$ is always between drone $i - 1$ and drone $i + 1$. It is also assumed that the drones can be placed directly on top of the highway, thereby simplifying the range and pathloss models.

Following the literature [8], we consider a rural setting with a sparse road network (less than 10 vehicles per kilometer). Both the literature and this study considers a sparse network because it estimates the worst-case scenario for data packet delivery [8]. The highway segment is in a straight line and of consistent width. For the sake of simplicity, we assume that all vehicles are moving at the same velocity $v$.

We define a subsegment as the region between two drones (Fig. 2). One important parameter of a subsegment is road density. In our work, we represent the vehicle density for a subsegment for the forward and backward directions collectively as $\lambda$.

We assume that all vehicles considered are VANET enabled and can send and receive data to base-stations or RSUs. The source of vehicular data packets is any moving vehicle, and the destination is an infrastructure drone which has access to the internet and other infrastructure systems. A vehicle acting as the packet source sends an exact copy of the packets in the opposite direction of the highway. Therefore, a packet is transferred from an arbitrary vehicle to a base-station by one of two ways:

1) The Vehicle of Focus relies on itself to deliver the message by driving within range of a base-station.
2) Rely on a carrier vehicle moving in the opposite direction of the highway. The source vehicle will duplicate the data-packet to an oppositely moving vehicle, the opposite vehicle can then transmit that data to its nearest base-station (in the backward direction of the source vehicle).

Delay is individually calculated for every subsegment between the drones. For a subsegment between two drones, the delay is modelled using an arbitrary vehicle of focus (VOF). The quickest way to deliver the information to a
base-station will be considered for the delay analysis. Since we are assuming a sparse road network, we assume there are no multi-hop connections in the same directions, meaning an arbitrary vehicle cannot use a vehicle in front of it to help deliver the data packet quicker. This is shown visually in Fig. 5.

Fig. 3 shows a 3D representation of the problem. For traffic, there is one lane per direction. The service area is flanked by two RSUs. The UAVs/agents can move their horizontal positions in order to maximize the portions of the service area that satisfy a delay constraint.

**B. DELAY MODEL**

The vehicle-to-drone packet delivery delay calculation is based on the work in [8]. The probability that the delay of an arbitrary vehicle of focus (VOF) is below a certain threshold \( d \) can be broken into the conditions \( \Phi = 1 \) and \( \Phi = 0 \). For \( \Phi = 1 \), there is one replica of the packet as the source vehicle sends the packet in the forward and opposite directions (considers both directions). In the second case (\( \Phi = 0 \)), there is no replication of the packets and a source carrier vehicle sends the packets only in the forward direction. The resulting output of the equation is the probability that the delay of a subsegment \( D \) is less than a specified delay amount \( d \):

\[
Pr(D \leq d) = Pr(D \leq d, \Phi = 1) + Pr(D \leq d, \Phi = 0).
\]  

(1)

We can use a single equation which will be instantiated differently to describe the probability distribution functions in the right side term of (1):

\[
Pr(D \leq d, \Phi) = \int_0^\infty \int_0^\infty a^{-2d_f} Pr(D \leq d|\Phi, x, y) Pr(\Phi|x, y) \cdot f(x, y) \, dy \, dx,
\]  

(2)

where \( x \) represents the distance between the source vehicle and the first vehicle in the opposite direction, and \( y \) is the distance between the source vehicle and the drone in the forward direction. We can limit \( y \) to be between 0 and \( a - 2d_f \), but \( x \) can be between 0 and \( \infty \). When we consider both sides, which ever side has least amount of delay is used for overall delay. Now, we particularize for each value of \( \Phi \in \{0, 1\} \).

When \( \Phi = 0 \), we only consider the time taken for a message to get to the forward drone and the probability distribution function takes the form

\[
Pr(D \leq d|\Phi = 0, x, y) = u(d - \frac{y - d_f}{v_f})
\]  

(3)

where \( v_f \) is the speed of the forward lane, and \( u(\cdot) \) is the Heaviside step function. For \( \Phi = 1 \), we have to consider both sides of the highway and the probability distribution function becomes

\[
Pr(D \leq d|\Phi = 1, x, y) = u(d - \min(\frac{a - y - d_f + x}{v_f}, \frac{y - d_f}{v_f}))
\]  

(4)

where \( v_f \) is the speed of the vehicles in the backward lane.

The second term of (2) expresses the exit probability of the packet carrier vehicle from the opposite lane. When \( \Phi = 0 \), \( Pr(\Phi|x, y) \) represents the probability that the Vehicle of Focus will exit the highway before reaching a drone’s communication range.

\[
Pr(\Phi = 0|x, y) = 1 - e^{Pr(x - dy + \lambda x)}. \quad (5)
\]

For \( \Phi = 1 \), \( Pr(\Phi|x, y) \) is the probability that the Vehicle of Focus will stay on the highway until it has reached a drone’s communication range.

\[
Pr(\Phi = 1|x, y) = e^{Pr(x - dy + \lambda x)}. \quad (6)
\]

The third term of (2) is the joint probability density function of \( x \) and \( y \),

\[
f(x, y) = \frac{\lambda}{a - 2d_f} (e^{-\lambda x}), \quad (7)
\]

which depends on the sum of the densities from both lanes (\( \lambda \)), the distance between the drones (\( a \)) and the range of each drone (\( d_f \)).

The resulting output is the cumulative distribution of the packet delivery delay which is plotted against the delay in Fig. 6. In this work, we define the delay value as the amount corresponding to 85 % probability on the CDF curve. In Fig. 6, the CDF curve is calculated with \( a = 4000, \lambda = 0.005, \lambda_x = 0.0004 \). In this case, the delay value we take is 52.49 seconds.

**C. MARL**

For MARL problems, we consider an extension of the single-agent MDP called a Markov Game [30]. A Markov Game can be defined by a tuple \((S, U, P, R, Z)\) with \( N \) agents identified by \( i \in I = 1, \ldots, N \). The environment containing all \( N \) agents has a global state \( s \in S \). At each timestep, each agent
will select an individual action \( u_i \in U \). The actions for all \( N \) agents will form joint action \( u \in U \). After receiving a joint action, the environment will undergo a transition using the state transition function \( P(s'|s, u) : S \times U \times S \rightarrow \mathbb{R} \). The agent \( i \) receives a reward \( r_i \in R \) where \( r_i = r(s, u, s', i) \) is the reward function and \( R : S \times U \times S \times \mathbb{I} \rightarrow \mathbb{R} \).

In MARL, full observability is usually not possible. An agent with full observability can condition its policy on observations of the global state. Observation \( z_i \) is the observation of agent \( i \) and is derived from the observation function \( O(s, i) : S \times \mathbb{I} \rightarrow Z \) where \( Z \) is the observation set. For this problem, individual agents have a 6-parameter observation. For agent \( i \):

\[
z_i = [a_{\text{left}}, a_{\text{right}}, \lambda_{\text{left}}, \lambda_{\text{right}}, \lambda_{\text{c.left}}, \lambda_{\text{c.right}}]
\]  

(8)

We can denote the joint observation as \( z = [z_1, z_2, \ldots, z_N] \). This represents the necessary information required to calculate the delay values for both sides of a drone. \( a_{\text{left/right}} \) is the distance to the left/right neighbouring drone/RSU respectively. \( \lambda_{\text{left/right}} \) is the traffic density of the road between the left/right neighbouring agents respectively. Likewise, \( \lambda_{\text{c.left/right}} \) is for junction density. These parameters are detected by sensors on-board the drone or fed to the drone by other signal sources.

In this study, we augment \( z_i \) with information from neighbouring agents \( i - 1 \) and \( i + 1 \), if available,

\[
z_{i}^{\text{aug}} = [z_{i-1} \cup z_i \cup z_{i+1}].
\]

(9)

Like with the original observations, we can create a joint observation variable \( z^{\text{aug}} = [z_1^{\text{aug}}, z_2^{\text{aug}}, \cdots, z_N^{\text{aug}}] \).

If the agent has a neighbour that is an RSU, all missing parameters will be replaced with 0s. The new observation of a single agent is then 12 parameters (combining the non-overlapping parameters from 3 agents).

The joint action \( u \) is the collection of individual actions from \( N \) agents. The agents can maneuver by selecting movements in either the left or right directions. At each time-step, available actions are:

\[
u_i = [-\text{step}, 0, +\text{step}]
\]

(10)

An action is selected through feeding the modified observation \( z_{i}^{\text{aug}} \) into the policy of agent \( i, \pi_i \), where \( \pi = [\pi_1, \pi_2, \cdots, \pi_N] \).

The step size is predefined as a hyper parameter. As mentioned earlier, any action that would result in UAV overlapping ranges or exchanging order with any adjacent neighbours will be considered an illegal action.

Since all agents in this system work cooperatively to minimize the delay of the service area, there is a shared reward. The shared reward uses the global state \( S \) to generate a singular numerical reward \( r \). Recall that a subsegment is a piece of the service area between two agents or between an agent and an RSU. The primary object of the reward function is to maximize the number of satisfied subsegments that satisfy the delay constraint of 55 seconds. We chose 55 seconds because it is a commonly used value in related literature [8], [28] [29]. The reward scheme is an exponential function of the number of satisfied subsegments. The higher number of satisfied portions, the greater the increase in reward. This encourages the agents to move out of potential local maximums. An example would be the agents preferring having 6 subsegments satisfied but each with 54 second delays rather than having 5 subsegments satisfied with 30 second delays but 1 with a 70 second delay. We added a factor of 2 to the exponent in order to emphasize the exponential growth.

**Algorithm 1 Reward Scheme**

\[
\text{Total} = 0 \\
\text{Initialize array } A \text{ with the delays of all the zones between each drone} \\
\text{for delay value } D \text{ in } A \text{ do} \\
\text{if } D \leq 55 \text{ then} \\
\text{Satisfied } += 1 \\
\text{end if} \\
\text{end for} \\
\text{Return } 2^\text{Satisfied}
\]

**IV. ALGORITHM**

The reinforcement learning algorithm used for this problem is a variation of Independent Q-learning explored by [18]. Unlike the work by [21], this approach does not modify the experience replay buffer to deal with non-stationarity. Instead, we take advantage of several key properties of this particular problem.

With the configuration and constraints of the agents mentioned in the previous section, we know that for agent \( i \), the subsegment between agent \( i \) and agent \( i + 1 \) as well as the space between agent \( i \) and agent \( i - 1 \) is shared. This means all parameters relating to density, distance, and delay are identical. Furthermore, since each agent can share its information with its neighbouring agent, one drone now has...
full observability of a 3-agent sub-problem. And because the neighbouring agents are also aware of its own neighbours, an agent can gain insight to the policies of neighbouring agents. This should improve the sample efficiency of the replay buffer compared to traditional independent Q-learning. Furthermore, there is now a fixed input size for the Q-Network. To simplify the problem, we neglect any communication delays between neighbouring agents. Since the agents are not making decisions at every time step, a small amount of delay between inter-drone information sharing should be tolerable.

Another way to leverage the shared information between agents is to share policy networks whenever possible. For example, given a 4-agent system, agents 1 and 4 are unique, since they only have one moveable neighbour; agent 1 has an RSU on its left and agent 4 has an RSU on its right. RSUs cannot move, therefore cannot contribute to non-stationarity. On the other hand, agents 2 and 3 are identical in the information that they can see; they both see a 3-agent sub-problem where the agent is surrounded by two moveable neighbours. Therefore, agents 2 and 3 can share the same policy network since they use the same structure of information to make decisions. Overall, in a system with three or more agents, where the agent is surrounded by two moveable neighbours, agents 2 and 3 can share the same policy network where the agent is surrounded by two moveable neighbours. Therefore, agents 2 and 3 can share the same policy network since they use the same structure of information to make decisions. Overall, in a system with three or more agents, where the agent is surrounded by two moveable neighbours, agents 2 and 3 can share the same policy network.

Akin to a regular Deep IQL, there are two neural networks used per agent, a policy network and a target network. The networks are all simple feedforward neural networks with two hidden layers. The inputs are the 12 features, the first hidden layer has 128 nodes while the second hidden layer has 64 nodes. The output is an array with the probability of each of three possible actions. The agents initialize their respective networks with consideration of network sharing as previously detailed. Upon initialization, the policy networks clone their weights into their respective target networks. For each episode, each step will have the agents simultaneously select an action based on their observations and policies. The next observation as well as the rewards will be added into a tuple with the current observation and action. This tuple will be stored in the replay buffer. After the replay buffer has enough entries for a specified batch size, a batch will be sampled. The policy network will be updated using the backpropagation rules seen in DQN. After a specified number of steps, the target networks are updated with the parameters of the policy networks.

**Algorithm 2 Deep IQL With Shared Policies**

Initialize environment with $N$ agents

Initialize policy networks $\theta_{\pi} = [\theta_{\pi_1}, \theta_{\pi_2}, \ldots, \theta_{\pi_N}]$ and duplicate for target networks $\theta_{\pi}^T = [\theta_{\pi_1}^T, \theta_{\pi_2}^T, \ldots, \theta_{\pi_N}^T]$

for $e = 1 \ldots \max$ episode number do

Reset environment and randomize positions for agents

if $N < 0$ then

Let highway populate with data

end if

for $t = 1 \ldots \max$ episode length do

observe joint augmented observation $z^{aug}$

select joint action $u$ w.r.t to the current policy and exploration rate of each agent

Execute joint action $u$ and observe shared reward $r$

and new observations $z^{aug}$

Store ($z^{aug}, u, r, z^{aug}$)in replay buffer

end for

if buffer can provide $s$ samples then

Sample a random mini-batch of size $s$ as: ($z^{aug}, u, r, z^{aug}, j+1$)

for each agent $i$ do

$y_j^i = r + \gamma \text{ argmax } Q(z_i^{aug}, u_j^i, \theta_{\pi_i}^T)$

Perform gradient descent on ($y_j^i - Q(z_i^{aug}, u_j^i, \theta_{\pi_i})$)

end for

end if

Update target network with policy network parameters every $\tau$ episodes

end for

produces a straight 2D highway segment that has sparse traffic. This means that the vehicle densities along the highway range from 0 to 0.007 vehicles per meter. Along the highway, there are also predefined road junctions. For this experiment, the entire highway is 35,000 m long and junctions are located at: [1500, 2500, 4500, 6000, 8000, 10500, 13500, 16000, 17000, 18500, 21500, 25000, 30000, 32500] (position in m starting from 0). Each junction has an exit probability of $P_e = 0.1$. A road junction has the ability to change the road density by $\pm 1$ car per kilometer, meaning the road density before and after a junction can vary by 1 car per kilometer. The junctions’ densities are limited from 0.0001 to 0.00015 junctions per meter. The service area is defined by two RSUs, which are represented by red circles in the simulator. For the purposes of our experiments, we placed the left-most RSU at position 0 of the entire highway.

The UAVs/agents are operating with a range of $d_r = 500$ m and an altitude of $300$ m. These UAVs can be placed anywhere along the highway given certain constraints:

1) agents positions are discretized into 100 m bins;
2) agents must be at least 1000 m apart and;
3) agents cannot switch positions.

**V. SIMULATION**

The simulation was conducted on a custom-made highway simulator based on similar metrics seen in [8]. The simulator
At each time step, the UAVs can move along the highway in discretized steps. For the experiments, a drone step size of 200 m is used.

As for velocity, vehicles on the road are all travelling at 27 m/s (around 97 km/h), which is a typical highway speed, and also the value used in [8].

The possible density and junction density value ranges were based on values generated from Simulation of Urban MOBility (SUMO). SUMO is an open traffic simulation package that can produce realistic traffic patterns [31]. Along with its traffic control interface (TraCI), it allows for the detection of traffic properties “on-line”. The issue with directly training with SUMO and TraCI is that it does not contain a feature for real-time density detection. Another problem is that while traffic can be manipulated easily, the highway itself cannot. The structure of the highway is, for the most part, hard-coded prior to training. This means editing the structure of the road is difficult after the initial design. The custom map provides the ability to color code different segments of the map depending on road density. Furthermore, using a custom map is also less resource intensive resulting in faster training times compared to its SUMO counterpart. There was also only a slight increase in training time when the GUI is activated, which is not the case with SUMO. This allowed us to train with the GUI, showing real time behaviour of the agents.

Despite the benefits of the custom environment, there are some potential drawbacks when compared to SUMO and TraCI. The custom environment simulates traffic through generating directly the density of vehicles, while SUMO simulates the actual vehicles on the road. This means that SUMO can provide accurate vehicle positions and allow for more realistic behaviour under more dense traffic. This experiment addresses these problems in two ways. The first is that the delay model is only tested with sparse traffic not dense urban traffic. The second is that, a model trained on the custom map is also tested on a similar SUMO map.

To save time while training, delay values were pre-sampled on the parameter ranges for distance, vehicle density and junction density using Eq. 1 and Eq. 2. The sampled delay values were exported into a.csv file and during execution and the retrieval of the delay value only requires a table lookup.

At each new episode, the agents randomize their positions along the service area while retaining their existing order. This will lessen the chance that the agents will settle at a local minimum.

VI. EXPERIMENTS

We conducted a four-part experiment to demonstrate the potential of our solution. Experiment I (feasibility) is comprised of the training and execution of different numbers of agents. Experiment II (scalability) tests how well a policy trained for a setup with less agents can perform when used on a setup with more agents. Experiment III (robustness) tests the robustness of the learned policies by incrementally removing agents from the system. Experiment IV (generalizability) tests how well the model trained on the custom map can be generalized for use on an industry standard simulator (SUMO).

A. EXPERIMENT I: FEASIBILITY

Experiment I tests the feasibility of using MARL on a UAV-assisted VANET highway. We trained six setups (1, 2, 3, 4, 6 and 8 agents) on three different highway lengths, 20 km, 30 km and 35 km. For Experiment I, we trained each setup for 3500 episodes. The agents were initialized at random positions along the highway at the beginning of each episode. The positions of the agents adhere to the rules elaborated in Section III, which include a minimum inter-drone distance of 1000 meters and inability to switch order. The reason for this is to promote exploration in the training process. Starting at the same position every episode will increase the chance of the agents falling into a local maximum. The densities of the road segments between the junctions will also vary by a magnitude 0.001 vehicles per meter every 300 episodes. By having dynamic road densities, we can ensure that the agents are moving based on detected features (densities and distances) and not overfitting to specific positions along the highway. We chose 300 episodes as the interval through trial-and-error; less than 300 is not enough time to stabilize, more than 300 can cause the agents to over-fit.

The performance of the MARL agents is expressed by the number of subsegments that satisfy the delay limit of 55 seconds. Recall that the number of subsegments is \( N + 1 \) where \( N \) is the number of agents. For visual clarity, we used “subsegments satisfied” for the y-axis instead of the “exponential function of the subsegments satisfied” for the reward function. When plotting the results, we used a sliding window technique with an aperture of 50 episodes to obtain a moving average line. This is plotted in Fig. 7 using a black line starting at episode 51 in order to have enough samples for the first window position.

Every setup has a theoretical maximum performance where every subsegment is satisfied. However, this is often not achievable since there are not enough agents to span the service area. The theoretical maximum is only achieved when the highway is saturated with UAVs so that the inter-drone distances are low, or if one of the subsegments experiences a period of no vehicles. In most cases, the system can only achieve a target of \( N \) number of subsegments satisfied (all but one). In all of the figures of Section VI, we represent the theoretical maximum as a green bar and the target score as a pink bar.

We also introduced a baseline performance as a lower bound for the agents. The baseline rewards are calculated by the agents equidistantly placed across the service area. Their positions are fixed throughout the duration of the training or execution. While in certain circumstances, there might exist positions with even lower rewards, we chose equidistant placement because it is the most basic placement strategy and what is used in other works [8]. In all of the figures of Section VI, this is portrayed as a red dashed line.
For training, the $1-3$ agent setups all managed to converge around the target line. This can be seen in Figs. 7a, 7b and 7c where the black line meets the pink line. During training, there were no episodes where the agents exceeded $N$ number of subsegments satisfied. This is because even in the best-case scenario, there would be one segment where the inter-drone distance is too large. The baseline performance for $1$ and $2$ agents are at $0$ meaning when statically placed equidistant from each other, no subsegment has a delay that is less than $55$ seconds. The $3$-agent setup does have a slight period of a non-zero baseline but the majority is still at $0$. For the $1-3$ agent experiments, the average lines begin to reach the target line at around $700-800$ episodes. Throughout the entire training span, the $1-3$ agent average lines are always above the baseline, this means that the MARL method is producing better results than using a static placement method.

In the $4$-agent setup, in Fig. 7d, the agents mostly achieve a maximum of $N$ subsegments satisfied. However, there are roughly $300$ episodes where the agents reach the theoretical maximum, meaning every part of the service area satisfies
the delay requirement. This is likely due to the dynamic road densities randomly producing a traffic pattern where all subsegments can be satisfied by the agents. However, as we see in the results, this is unlikely to occur. The baseline for 4-agent training is mostly greater than 0, with a maximum of 3 subsegments satisfied. During the entire training process, the MARL algorithm always outperformed above the baseline, with an average of one more subsegment satisfied.

For the 6 and 8-agent setups, we used longer service areas of 30 km and 35 km respectively in order to give the agents more room to move. The results are shown in Fig. 7e and 7f. For 6-agents on a 30 km highway, the training converged at N subsegments satisfied at around 2200 episodes. The performance temporarily dropped after initially reaching the target line likely due to the dynamically changing densities on the highway producing a state that was drastically different to the agents. The baseline for 6 agents has a maximum of 4 subsegments satisfied, however during training, the average line was mostly higher than the baseline. The 8-agent setup on a 35 km service area is unique because the black converges towards...
the theoretical maximum. This means that there are enough agents for the length of the given service area. It approaches $N + 1$ subsegments satisfied at around 2800 episodes. The baseline also performs better as it reaches a maximum of 7 subsegments satisfied. However, from the training results, our MARL method still outperforms the baseline at every episode.

For testing the policies after training, we placed the agents, loaded with their respective policies in their same service area lengths used for training and ran the system continuously for 3000 steps. At every step, the agents pick a joint action similar to how they did during training. However, for testing, we did not allow the agents to learn. The inter-junction densities could also change by 0.001 vehicles per meter every 100 steps. Like with learning, the performance is measured by the number of subsegments satisfied and judged by the same 3 criteria (theoretical maximum, target maximum and the baseline). The results are shown in Fig. 8.

For testing, 1 − 4 agent setups all constantly performed at the target maximum of $N$ number of segments satisfied with the baseline always below the MARL policies' performance. This can be seen in Figs. 8a, 8b, 8c and 8d where the black line is at the pink line while the red dashed line is below. For the 6-agent testing, the policies’ performance dips below the target occasionally; in Fig. 8e, the black line falls from the pink line to 5 subsegments satisfied. A possible explanation for the dips is that the 6-agent case ran on a longer highway/service area (30 km). A longer service area not only increases the number of possible positions for the agents, but also increases the number of potential road density combinations. However, it still, on average, satisfies 3 − 4 subsegments more than the baseline suggesting it is worth using a MARL algorithm over a static equidistant placement strategy. For the 8-agent case, the MARL performed mostly at the theoretical maximum line. This means that every subsegment was satisfied. This outperformed the baseline which had a maximum of 7 subsegments satisfied. This is seen in Fig. 8f.

**B. EXPERIMENT II: SCALABILITY**

Experiment II tests the scalability of the MARL method. A UAV placement algorithm should be able to easily adapt to increases in the number of agents during execution and should not require a complete re-train when changing the problem size. For scalability testing, we imported base-policies into higher-agent count setups. We plotted the results of 3500 continuous steps without additional training and compared it to the execution performance after 500 episodes of re-training with the imported policies. This is shown in Fig. 9.

We experimented with two base polices, a 1-agent policy and a 3-agent policy sets. Both base-policies were trained on 20 km service areas in Section VI-A. For the 1-agent baseline policy, we imported it on 2, 3, 4, 6 and 8-agent setups. For a 1-agent base-policy test, we imported the same policy into all the agents. The 2, 3 and 4-agent setups were tested on 20 km service areas while the 6 and 8-agent setup was on 30 km and 35 km service areas respectively. For the 3-agent base-policy, we imported it into 4, 6 and 8-agent setups on 20 km, 30 km, and 35 km highways respectively. The 3 component base-policy was imported 1-to-1 into the testing agents.

For the 1-agent base-policy, the performance of directly importing the policies was constantly 1 − 2 subsegments below the target line. This is seen in Fig. 9 where the blue line is below the pink line. In the case of the 4-agent setup, (Fig. 9c), the performance was sometimes the same as the baseline. It is likely that the 1-agent base-policy has no experience when dealing with neighbouring agents, therefore performs sub-optimally. After 500 episodes of training with the base-policy, the setups with more agents start performing at or near the target line, except for the 6 and 8-agent setups, both of which perform below the target line. This is shown by the black line in Fig. 9d and 9e. An explanation for this is that a 1-agent base-policy expects no moving neighbours, and as a result, does not expect non-stationarity. In every other case, every agent has at least one moveable neighbour, therefore, needs to account for other actions in order to deal with non-stationarity. The higher the number of agents in the system, the greater amount of non-stationarity. It is likely for 2, 3, and 4-agents, there was not enough non-stationarity and the agents were still able to perform well. However, with 6 or 8-agents, there was too much variability in the system to find the globally optimal positions.

The 3-agent base-policy performs better both from a direct import and re-training. This is seen in Fig. 10. This is because the fundamental policy structure is the same for setups above 3 agents. Without re-training, the 4, 6 and 8-agent setups all perform at or only 1 subsegment below the target line. With 500 episodes of re-training, the agents perform at the target line. After 500 episodes of re-training, the model performs equally or better than the policies after direct import. In general, we see that the MARL method is scalable for agents that share the same 3-policy structure, while also constantly outperforming the baseline.

**C. EXPERIMENT III: ROBUSTNESS**

An important quality for group-UAV control algorithms is that it should be robust. In real-life, agents may experience technical issues and would be unable to continue operating. The system should be able to quickly adapt to failures.

For robustness testing, we used a 6-agent setup on a 30 km service area to start, and then introduced either single point or dual point failures. When a failure happens, the agent is deleted from the agent list in the back-end and all neighbour connections are recreated. A non-zero exploration rate is introduced into the system to promote a faster re-learn. Furthermore, the agents cannot look for and import previously learnt policies from setups with less agents; all agents have to continue using their present policies. The results are shown in Fig. 11.

For the single point failures, the system drops out 1 agent every 1000 episodes until there is a single agent left. For
For the dual point failures, the system drops out 2 agents every 1000 episodes until there are 2 agents left. For consistency, we also drop out two agents every 1000 episodes for the baseline. The results for dual point failures are shown in Fig. 11b. After every drop in the number of agents, the system was able to stabilize very close to the target line. After the first 2 failures, the reward line stabilizes near 4 subsegments satisfied, and after 4 failures, the reward line stabilizes near 2 subsegments satisfied. The second dual point failure is unique in that the system goes from 4 agents to 2. Recall that in setups...
more than or equal to 3 agents, there are 3 unique policies while there are only 2 unique policies with a 2-agent system. The results show that even when the policy setup is changed from 3 to 2 unique policies, the system was able to stabilize close to the target line. After all the agents are turned back on, the system returns to the target of 6 subsegments satisfied.

Overall, the results show that the method is resistant to outages in the UAV fleet. Every time an agent is deactivated,
FIGURE 12. Experiment 4: Generalizability training results.

D. EXPERIMENT IV: GENERALIZABILITY

As noted in previous sections, most of the experiments in Section VI are performed on a custom highway simulator, which generates simplified, yet sufficient traffic information. We do this to improve the training and testing speed. However, we also show that the MARL method also learns well when performed on an industry standard simulator like SUMO.

For generalizability tests, we created a highway that was identical in design to our custom simulator. We also placed the junctions at the same positions as the ones on the custom map. The SUMO simulator also produces dynamic densities ranging from 0 to 0.007 vehicles per meter. The main difference is that the road density is not generated by a random number generator, but rather by detecting and counting
vehicles on the road. Because the vehicles are moving, the road densities are subject to more frequent but smaller changes.

We trained the same 1, 2, 3, 4, 6 and 8-agent setups as we did in the feasibility tests and compared the results (in Fig. 12). Likewise, we judged the performance of the MARL algorithm (black line) based on the equidistant static placement baseline (red line), the target value with all but one subsegments satisfied (pink line) and the theoretical maximum value with all subsegments satisfied (green line). In most instances, the agents in the SUMO environment performed comparably to that in the custom simulator with the latter producing slightly better results at each episode. Since the difference is minute, we believe that this is due to the different density states in the two simulators and not a significant difference in the quality of performance. The only case where the SUMO simulator training was outperformed by the custom simulator training was with 8 agents on a 30 km highway (in Fig. 12f). In the custom simulator, the results converged towards the theoretical maximum. Under the SUMO simulator, there were episodes where the agents reached theoretical maximum, however it stabilized around the target line. An explanation for this could again be the way that SUMO generates density values. As SUMO needs to count the individual vehicles on the road, it is more susceptible to smaller and yet frequent changes. These frequent changes could have made it more difficult for the agents to stabilize at particular positions, hence producing a slightly lower performance.

When training, the main advantage for the custom simulator is the low resource intensity. On SUMO, the 8-agent setup took over 190 minutes to train 3500 episodes while the custom simulator only took 20 minutes. An explanation for this phenomenon is that SUMO must generate the dynamics for all the individual vehicles while the custom simulator directly generates densities between junctions. While SUMO’s data is more realistic and can potentially be useful for other tasks, for the purpose of our experiments, the data produced from our custom simulator produces similar results with the benefit of much lower training times.

VII. DISCUSSION AND CONCLUSION

In this paper, we studied the feasibility and benefits of using MARL to find the optimal positions of UAVs along a straight sparse highway. The goal is to maximize the number of subsegments that satisfy the delay limit of 55 seconds. We proposed a Deep IQL MARL method with a modified observation function and sharing of policies. For testing, we implemented our own custom environment.

Experiment I (Section VI-A) showed that the proposed MARL algorithm is able to learn the positions that satisfy the maximum reward without full observability. This suggests that advanced actor-critic methods with mixed levels of observability are not always required for a multi-agent problem, and depending on the question, augmenting observations may be sufficient to provide global coordination and effective at mitigating the problem of non-stationarity.

Experiment II (Section VI-B) tested scalability. The results showed that when imported with base-policies trained with a fewer number of agents, the system was still able to produce near optimal results after some re-training. We did find that the 1-agent base-policy performed worse at scalability compared to a 3-agent base-policy. This is because the inherent policy structure is different. Compared to other options like MADDPG, where the global observation will grow with the number of agents, our approach requires less effort to scale up.

Experiment III (Section VI-C) showed that the system can easily adapt to single and dual point failures. Starting from a 6 agent setup, the policies only took less than 300 episodes for each progressive failure. This is an important feature to have in the real world as extensive re-training can cause long service delays which can affect the safety of drivers on the road.

Experiment IV (Section VI-D) showed that our algorithm is transferable to other simulation environments. When the algorithm was used on a similar map in SUMO, the training results were comparable; they converged at similar rates to the same maximums. It suggests our method will work on other environments that provide necessary parameters such as distance to neighbour, road density, and junction density.

As autonomous vehicles become more prevalent on roads, the need for VANET infrastructure becomes greater. UAVs equipped with base-stations can use their mobility advantage to fill-in gaps in coverage based on demand. However, their placement has to be carefully coordinated in order to maximize their efficiency. The use of MARL brings the benefits of not needing a model and the ability to learn non-trivial placement rules for each drone individually.

For future work, we can test the execution performance of directly importing policies from the custom simulator into SUMO. For potential use in urban scenarios with greater traffic density, we would have to modify the algorithm to incorporate more parameters such as vehicle-to-vehicle connections, UAV speed and trajectories, and UAV line-of-sight. Our method will also have to accept more complex road structures (winding highways, more lanes, more complex on and off ramps). For now, through testing on a simple highway structure with sparse traffic, our work shows the promising potential of using MARL for this category of problems.

REFERENCES

[1] P. A. Hancock, I. Nourbakhsh, and J. Stewart, “On the future of transportation in an era of automated and autonomous vehicles,” Proc. Nat. Acad. Sci. USA, vol. 116, no. 16, pp. 7684–7691, Apr. 2019.

[2] B. Förber, “Communication and communication problems between autonomous vehicles and human drivers,” in Technical, Legal and Social Aspects. Berlin, Germany: Springer, 2016, pp. 125–144.

[3] C. Liu, H. Huang, H. Du, and X. Jia, “Optimal rsus deployment in vehicular networks,” in Web-Age Information Management, Y. Chen, W.-T. Balke, J. Xu, W. Xu, P. Jin, X. Lin, T. Tang, and E. Hwang, Eds. Cham, Switzerland: Springer, 2014, pp. 236–246.
[4] M. Al-shareeda, A. Khalil, and W. Fahs, “Realistic heterogeneous genetic-based RSU placement solution for V2I networks,” Int. Arab J. Inf. Technol., vol. 16, pp. 540–547, Jan. 2019.

[5] A. Aloisio, V. Izzo, and S. Rampone, “FPGA implementation of a greedy algorithm for set covering,” in Proc. 14th IEEE-NPSS Real Time Conf., 2005, p. 5.

[6] C.-C. Lin and D.-J. Deng, “Optimal two-lane placement for hybrid VANET-sensor networks,” IEEE Trans. Ind. Electron., vol. 62, no. 12, pp. 7883–7891, Dec. 2015.

[7] I. Bor-Yaliniz and H. Yankomerolu, “The new frontier in RAN heterogeneity: Multi-tier drone-cells,” IEEE Commun. Mag., vol. 54, no. 11, pp. 48–55, Nov. 2016.

[8] H. Selimi, R. Shahidi, M. H. Ahmed, and M. S. Shehata, “Drone-based highway-VANET and DAS service,” IEEE Access, vol. 6, pp. 20125–20137, 2018.

[9] S.-M. Hung and S. N. Givigi, “A Q-learning approach to flocking with UAVs in a stochastic environment,” IEEE Trans. Cybern., vol. 47, no. 1, pp. 186–197 Jan. 2017.

[10] H. Raslan, H. Schwartz, and S. Givigi, “A learning invader for the, guarding a territory” game: A reinforcement learning problem,” J. Intell. Robot. Syst., vol. 83, no. 1, pp. 55–70, 2016.

[11] L. Xiao, X. Lu, D. Xu, Y. Tang, L. Wang, and W. Zhuang, “UAV relay in VANETs against smart jamming with reinforcement learning,” IEEE Trans. Veh. Technol., vol. 67, no. 5, pp. 4087–4097, May 2018.

[12] J. Wu, M. Fang, H. Li, and X. Li, “RSU-assisted traffic-aware routing based on reinforcement learning for urban VANets,” IEEE Access, vol. 8, pp. 5733–5748, 2020.

[13] J. A. Calvo and I. Dusparic, “Heterogeneous multi-agent deep reinforcement learning for traffic lights control,” in Proc. Irish Conf. Artif. Intell. Cogn. Sci., 2018, pp. 2–13.

[14] H. Ye, G. Y. Li, and B.-H. F. Juang, “Deep reinforcement learning based resource allocation for V2V communications,” IEEE Trans. Veh. Technol., vol. 68, no. 4, pp. 3163–3173, Apr. 2019.

[15] V. G. Rolla and M. Curado, “A reinforcement learning-based routing for delay tolerant networks,” Eng. Appl. Artif. Intell., vol. 26, no. 10, pp. 2243–2250, Nov. 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197613001450

[16] S. E. Hammami, H. Affifi, H. Moangla, and A. Kamel, “Drone-assisted cellular networks: A multi-agent reinforcement learning approach,” in Proc. IEEE Int. Conf. Commun. (ICC), May 2019, pp. 1–6.

[17] G. Qu, Y. Lin, A. Wierman, and N. Li, “Scalable multi-agent reinforcement learning for networked systems with average reward,” 2020, arXiv:2006.06626. [Online]. Available: https://arxiv.org/abs/2006.06626

[18] M. Tan, “Multi-agent reinforcement learning: Independent vs. cooperative agents,” in Proc. 10th Int. Conf. Mach. Learn. Learn. San Mateo, CA, USA: Morgan Kaufmann, 1993, pp. 330–337.

[19] L. Buonocore, R. Babuska, and B. De Schutter, “Multi-agent reinforcement learning: An overview,” in Innovations in Multi-Agent Systems and Applications. Springer, 2010, pp. 183–221.

[20] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson, “Stabilising experience replay for deep multi-agent reinforcement learning,” 2017, arXiv:1702.08887. [Online]. Available: https://arxiv.org/abs/1702.08887

[21] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, “Learning to communicate with deep multi-agent reinforcement learning,” 2016, arXiv:1605.06676. [Online]. Available: https://arxiv.org/abs/1605.06676

[22] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, “Multi-agent reinforcement learning in sequential social dilemmas,” 2017, arXiv:1702.03037. [Online]. Available: https://arxiv.org/abs/1702.03037

[23] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, “Counterfactual multi-agent policy gradients,” Comput. Res. Reposit., vol. abs/1705.08926, 2017. [Online]. Available: http://arxiv.org/abs/1705.08926 and https://dblp.org/rec/journals/corr/FoersterFAN17.bib

[24] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” in Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS), Long Beach, CA, USA, Red Hook, NY, USA: Curran Associates, 2017, pp. 6382–6393.