Decentralized Scheduling for Cooperative Localization With Deep Reinforcement Learning

Citation for the original published paper (version of record):
Peng, B., Seco-Granados, G., Steinmetz, E. et al (2019)
Decentralized Scheduling for Cooperative Localization With Deep Reinforcement Learning
IEEE Transactions on Vehicular Technology, 68(5): 4295-4305
http://dx.doi.org/10.1109/TVT.2019.2913695

Decentralized Scheduling for Cooperative Localization with Deep Reinforcement Learning

Bile Peng, Gonzalo Seco-Granados, *Senior Member, IEEE* Erik Steinmetz, Markus Fröhle, *Student Member, IEEE*, and Henk Wymeersch, *Member, IEEE*

***Abstract*—Cooperative localization is a promising solution to the vehicular high-accuracy localization problem. Despite its high potential, exhaustive measurement and information exchange between all adjacent vehicles is expensive and impractical for applications with limited resources. Greedy policies or hand-engineering heuristics may not be able to meet the requirement of complicated use cases. We formulate a scheduling problem to improve the localization accuracy (measured through the Cramér-Rao lower bound (CRLB)) of every vehicle up to a given threshold using the minimum number of measurements. The problem is cast as a partially observable Markov decision process (POMDP) and solved using decentralized scheduling algorithms with deep reinforcement learning (DRL), which allow vehicles to optimize the scheduling (i.e., the instants to execute measurement and information exchange with each adjacent vehicle) in a distributed manner without a central controlling unit. Simulation results show that the proposed algorithms have a significant advantage over random and greedy policies in terms of both required numbers of measurements to localize all nodes and achievable localization precision with limited numbers of measurements.**

***Index Terms*—Machine-learning for vehicular localization, cooperative localization, deep reinforcement learning, deep Q-learning, policy gradient.**

I. Introduction

Localization of vehicles has gained importance with the availability of increasingly automated vehicles. Modern vehicles can rely on a variety of sensors, including global positioning system (GPS), LIDAR, radar, and stereo cameras [1]. The use of radio technologies can play an important role as a redundant sensor, especially in the context of emerging 5G communication [2] and internet of vehicles [3]–[5] technologies. As 5G can be used both for communication and localization, it is a natural candidate for *cooperative localization*, where vehicles aid one another to determine their relative or absolute locations. Cooperative localization has shown to improve both coverage and accuracy [6]. Cooperation between vehicles comes at a cost in terms of resources (power, bandwidth), which need to be carefully optimized due to their scarce nature [7], [8]. In addition, cooperation leads to larger delays (and thus reduced update rates), due to (i) the measurement process, where inter-vehicle distance and angle measurements are collected; (ii) the information exchange (communication) during fusion of information, where measurements and a priori information are combined. Consequently, scheduling of transmissions for the cooperative localization problem is an important challenge [9], [10]. Often, the corresponding optimization problems do not have closed-form solutions and suffer from poor scalability, due to their combinatorial nature. If we take cooperation and long-term reward into account, the problem complexity would be prohibitive for traditional approaches. A recently (re-)emerging trend in the field of wireless communication is to rely on machine learning tools for providing novel solutions to outperform engineered methods [11].

Among the different branches in machine learning, deep reinforcement learning (DRL) is particularly attractive, as it combines reinforcement learning (RL) and deep neural network (DNN), can be applied to difficult Markov decision processs (MDPs) where labeled data may be expensive or not available, consider the interaction between agent and environment (i.e., the action of agent changes the environment state) and take long-term rewards into account [12]. Concisely, DRL involves agents observing states and acting in order to collect long-term rewards. The decisions are determined by a policy, which maps the state to an action. For complicated problems with large state and action spaces, the DNN is an possible implementation of the policy. So-called DRL has seen success across many areas [13]–[19].

DRL algorithms can be generally categorized into Q-learning and policy gradient (PG). The former estimates the expected long-term reward (defined as *Q-value*) of each action and selects the action with the highest Q-value (hence the algorithm estimates the Q-values explicitly and formulates the policy in an indirect way) [20], [21], whereas the latter optimizes the policy directly by improving the policy in direction of the gradient of the total reward with respect to the policy parameters [22]. A more detailed introduction to DRL can be found in Section III.

In context of wireless communication, DRL has been applied to a number of applications, e.g., in the areas of power and rate control [23]–[26]. In addition, distributed routing was investigated in [27] using the REINFORCE method (a policy gradient algorithm). A good overview of work up to 2012 can be found in [28]. More recently, [29] considers a deep Q-network (DQN) for multi-user dynamic spectrum access, and [30] applies a DQN for scheduling in a vehicular scenario where gateways aim to deliver data quickly without

B. Peng, E. Steinmetz, M. Fröhle, and H. Wymeersch, are with the Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, 41258 Gothenburg, Sweden (e-mail: {bile.peng, estein, frohle, henkw}@chalmers.se). E. Steinmetz is also with the Division of Safety and Transport at RISE Research Institutes of Sweden, Borås, Sweden. G. Seco-Granados is with the Department of Telecommunications and Systems Engineering, IEEC-CERES, Universitat Autonoma de Barcelona, 08193 Barcelona, Spain (email: gonzalo.seco@uab.cat). This work was supported in part by the Spanish Ministry of Science, Innovation and Universities under grant TEC2017-89925-R.

depleting their batteries. Power control was again considered in [31], where agents make decisions based on high-dimensional local information (interference levels to and from neighbors) with rewards given by spectral efficiency, penalized with interference. Recent advances in machine learning in the vehicular domain were covered in [32], and highlights intelligent wireless resource management based on DQN.

From a more abstract point of view, the above problems can be seen as either single-agent RL or multi-agent reinforcement learning (MARL) [33]. For a survey on MARL, we refer the reader to [34]. In contrast to single-agent RL, MARL presents a number fundamental challenges [35] as the multi-agent extension leads to a so-called stochastic game. When all agents observe the same global state, the problem reverts to an MDP, though with larger state and action spaces. In contrast, when independent agents are considered, the actions of other agents affect the observed environment by an agent, thus leading to a partially observable Markov decision process (POMDP) [33]. To make the system more Markovian, state histories are generally collected, often combined with recurrent DNNs [36]. Deep MARL is a current topic of research [37], [38], where agents may exchange information regarding rewards, policies or observations.

In this paper, we consider the cooperative localization problem from a MARL perspective, where each agent corresponds to an edge in the network. The problem is cast as a POMDP with a per-agent reward designed to localize all network nodes below a given uncertainty threshold as quickly as possible. The problem is then solved using DQN and PG. The obtained policies are then applied to the identical and larger scenarios, and provide performance improvements over both a random scheduling as well as a greedy algorithm. The main contributions of this paper are:

- The formulation of a cooperative localization scheduling problem in the context of a POMDP;
- The development of DQN and PG algorithms to solve the POMDP.

In the following part this paper, Section II introduces the system model and describes the calculation of the Cramér-Rao lower bounds (CRLBs), which is used as metric of localization precision. Section III presents the DRL algorithm for optimized decentralized scheduling. The simulation results are presented in Section IV and the conclusion is drawn in Section V.

II. System Model

In this section, we introduce the considered scenario, the approach to calculate the CRLB and formulate the scheduling problem.

A. Network Model

We consider a network graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ is the set of nodes (vehicles) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges (links) between nodes. Each node has a position $\mathbf{x}_i = (x_i, y_i)$, $i \in \mathcal{V}$ in a global frame of reference, where x_i and y_i are the coordinates in x and y directions, respectively. Each node is equipped with two types of sensors: a GPS-type of sensor that provides an estimate of $\mathbf{x}_i$ with covariance $\mathbf{\Sigma}_i^{\text{prior}}$ and a radar-type of sensor that provides relative location information for $(i, j) \in \mathcal{E}$. Finally, we assume that nodes can communicate with adjacent nodes.

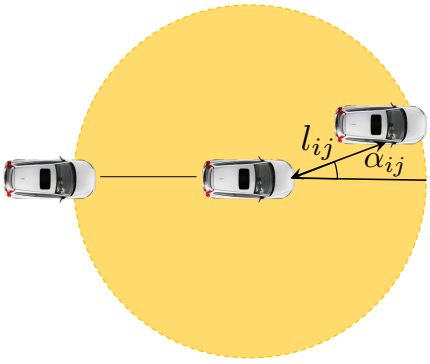


Fig. 1. Measurement between two nodes comprises relative position: distance l_{ij} and angle α_{ij}.

B. Measurement Model

We consider a radar-type measurement with multiple receiving antennas. The measured quantities are thus distance l_{ij} and angle α_{ij}, which are shown in Fig. 1 and calculated as

$$l_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{1}$$

and

$$\alpha_{ij} = \arctan\left(\frac{y_j - y_i}{x_j - x_i}\right) \tag{2}$$

respectively. The measurement can be expressed as

$$\begin{aligned} \mathbf{z}_{ij} &= \begin{pmatrix} l_{ij} \\ \alpha_{ij} \end{pmatrix} + \mathbf{n}_{ij} \\ &= \begin{pmatrix} \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \\ \arctan\left(\frac{y_j - y_i}{x_j - x_i}\right) \end{pmatrix} + \mathbf{n}_{ij}, \end{aligned} \tag{3}$$

where $\mathbf{n}_{ij}$ is Gaussian measurement noise, assumed to be of zero mean and diagonal covariance matrix $\mathbf{\Sigma}_{ij}$.

C. Objective

Our objective is to localize all nodes of the network (i.e., to reduce the uncertainty on the position of each node below some threshold) as quickly as possible (i.e., with as few measurements as possible). Let $a_{ij} \in \mathbb{N}$ be the number of times nodes i and j perform a measurement and $\mathbf{\Sigma}_i^{\text{pos}}$ denote the posterior position covariance of node i, then this problem can be formulated as follows:

$$\begin{aligned} \underset{\boldsymbol{A}}{\text{minimize}} \quad & \sum_{(i,j)\in\mathcal{E}} a_{ij} \\ \text{s.t.} \quad & \sqrt{\text{tr}[\mathbf{\Sigma}_i^{\text{pos}}(\boldsymbol{A})]} \leq \kappa, \ \forall i \end{aligned} \tag{4}$$

where $\boldsymbol{A}$ describes actions of all agents, $[\boldsymbol{A}]_{ij} = a_{ij}$ and κ is a threshold (in meters).

D. Network Localization Formulations

In this section, we describe two approaches to network localization and highlight the impact of measurement ordering.

1) Measure-then-Localize: Under this first approach, the network first performs all measurements between all pairs of nodes and then localizes all nodes. The localization uncertainty can be lower bounded [39] using the Fisher information matrix (FIM) $\mathbf{J}(\boldsymbol{A})$, which is a $2N \times 2N$ matrix, with 2×2 block

$$\mathbf{J}_{ij}(\boldsymbol{A}) = \begin{cases} \mathbf{J}_{ii}^{\text{prior}} + \sum_{k \neq i} a_{ik} \mathbf{J}_{ik}^{\text{meas}} & i = j \\ -a_{ij} \mathbf{J}_{ij}^{\text{meas}} & i \neq j, \end{cases} \tag{5}$$

in which $\mathbf{J}_{ii}^{\text{prior}}$ is the a priori information of node i (e.g., from GPS) and $\mathbf{J}_{ik}^{\text{meas}}$ is the amount of information a measurement between nodes i and k brings. From the model, it follows immediately that

$$\mathbf{J}_{ij}^{\text{meas}} = \boldsymbol{\Gamma}_{ij}^{\mathsf{T}} \boldsymbol{\Sigma}_{ij}^{-1} \boldsymbol{\Gamma}_{ij} \tag{6}$$

where

$$\boldsymbol{\Sigma}_{ij} = \text{diag}[\sigma_l^2 \quad \sigma_\alpha^2] \tag{7}$$

is the measurement covariance matrix with σ_l^2 and σ_α^2 the noise variances in distance and angle measurements, respectively, and the 2×2 Jacobian matrix of the range and angle measurements:

$$\boldsymbol{\Gamma}_{ij} = \begin{pmatrix} (\mathbf{x}_i - \mathbf{x}_j)^{\mathsf{T}} / l_{ij} \\ (\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j)^{\mathsf{T}} / l_{ij}^2 \end{pmatrix} \tag{8}$$

in which $\tilde{\mathbf{x}}_i = [-y_i \quad x_i]^{\mathsf{T}}$. Finally, the equivalent Fisher information matrix (EFIM) of node i $\mathbf{J}_i^{\text{E}}(\boldsymbol{A})$ is defined as the Schur complement of the block of $\mathbf{J}(\boldsymbol{A})$ without the 2 rows and 2 columns corresponding to node i, of the matrix $\mathbf{J}(\boldsymbol{A})$. As the FIM provides a lower bound under the error covariance, it follows, under regularity conditions, that $\boldsymbol{\Sigma}_i^{\text{pos}}(\boldsymbol{A}) \succeq (\mathbf{J}_i^{\text{E}}(\boldsymbol{A}))^{-1}$. We further introduce the positioning error bound (PEB) as

$$\text{PEB}_i = \sqrt{\text{tr}[(\mathbf{J}_i^{\text{E}}(\boldsymbol{A}))^{-1}]} \tag{9}$$

so that (4) can be approximated by

$$\begin{aligned} \underset{\boldsymbol{A}}{\text{minimize}} \quad & \sum_{(i,j) \in \mathcal{E}} a_{ij} \\ \text{s.t.} \quad & \text{PEB}_i \leq \kappa, \, \forall i \end{aligned} \tag{10}$$

While this problem in principle allows to find the optimal $\boldsymbol{A}$, it is generally hard to solve due to the high-dimensional and integer nature of $\boldsymbol{A}$, as well as the complex dependence of $\mathbf{J}_i^{\text{E}}(\boldsymbol{A})$ on $\boldsymbol{A}$.

Remark 1: In (10) the ordering of the measurements does not play a role: a measurement between two nodes with high a priori uncertainty is equally useful if it is scheduled first or last. Moreover, when multiple measurements are taken, each measurement contributes equally.

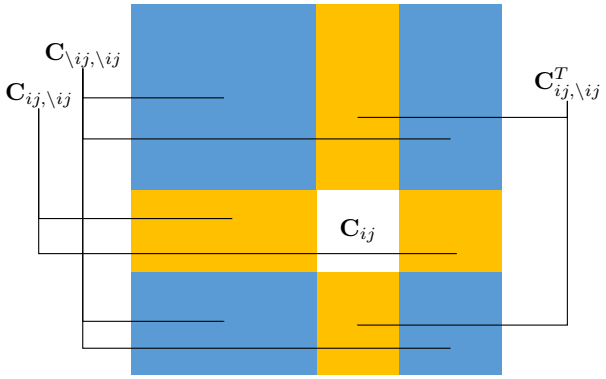


Fig. 2. Split of $\mathbf{C}$ for update using Kalman principle.

2) Localize-while-Measuring: A more practical way of performing network localization is to consider the perspective of a single link in the network (i, j), whereby a measurement is used immediately to update the location estimates of both nodes, but does not impact the location estimates of other nodes. Hence, this leads to a sequential decision making problem to progressively improve the EFIMs, whereby each link must decide whether or not to activate (i.e., measure and then update the location estimates) based on the current observable state of the network.

The evolution of the uncertainty is easily understood in the inverse FIM domain. Let $\mathbf{C}^{(0)} = (\mathbf{J}^{\text{prior}})^{-1}$ correspond to the a priori block-diagonal covariance for all nodes. By induction, we assume that we have performed a sequence of measurements $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(k)}$, where each $\boldsymbol{A}^{(k)} \in \mathbb{B}^{N \times N}$ contains zeros and a single one. Assume that $\mathbf{C}^{(k)} = \mathbf{C}$ is known and we wish to determine $\mathbf{C}^{(k+1)} = \mathbf{C}'$ after measurement $\boldsymbol{A}^{(k+1)}$, which involves nodes i and j.

1) We split the covariance matrix $\mathbf{C}$ into the following parts: (i) covariance matrix involved in the measurement $\mathbf{C}_{ij} \in \mathbb{R}^{4 \times 4}$, (ii) covariance matrix between involved quantities (ij) and not involved quantities (denoted by $\setminus ij$) $\mathbf{C}_{ij,\setminus ij}$ and (iii) covariance matrix of not involved quantities $\mathbf{C}_{\setminus ij,\setminus ij} \in \mathbb{R}^{4 \times (2N-4)}$. Note that $\mathbf{C}_i \in \mathbb{R}^{2 \times 2}$ will denote the covariance of the position of node i. Without loss of generality, we focus on the case $j = i + 1$, in which the split is illustrated in Fig. 2 (the general case can be obtained by reordering the node indices).
2) After a measurement, $\mathbf{C}_{ij}$ will change accordingly because the measurement increases covariance between the involved quantities; $\mathbf{C}_{ij,\setminus ij}$ is also affected because the measurement updates the involved quantities and their covariance with the uninvolved quantities decreases. $\mathbf{C}_{\setminus ij,\setminus ij}$ is unchanged since none of its respective position estimates are affected by the measurement.
3) We apply the principle of the sequential estimation to update the covariance. The Kalman gain for measurement

between nodes i and j is calculated as [39, pp. 249]

$$\mathbf{K}_{ij} = \mathbf{C}_{ij}\mathbf{T}_{ij}^{\mathsf{T}}\left(\mathbf{\Sigma}_{ij} + \mathbf{T}_{ij}\mathbf{C}_{ij}\mathbf{T}_{ij}^{\mathsf{T}}\right)^{-1}. \tag{11}$$

$$\mathbf{C}'_{ij} = \mathbf{C}_{ij} - \mathbf{K}_{ij}\mathbf{T}_{ij}\mathbf{C}_{ij} \tag{12}$$

$$\mathbf{C}'_{ij,\setminus ij} = \mathbf{C}_{ij,\setminus ij} - \mathbf{K}_{ij}\mathbf{T}_{ij}\mathbf{C}_{ij,\setminus ij} \tag{13}$$

$$\mathbf{C}'_{\setminus ij,\setminus ij} = \mathbf{C}_{\setminus ij,\setminus ij}, \tag{14}$$

where we have introduced

$$\mathbf{T}_{ij} = \begin{pmatrix}\mathbf{\Gamma}_{ij} & -\mathbf{\Gamma}_{ij}\end{pmatrix} \tag{15}$$

4) Finally, the whole updated matrix $\mathbf{C}'$ is then built again from the updated blocks.

It is to note that i and j are assumed to be adjacent in Fig. 2 for simplicity. If i and j are not adjacent, $\mathbf{C}$ has to be split into more pieces but all of them still falls into the above described three categories. It is also to note that $\mathbf{C}$ is symmetric. Therefore, only half of the elements need to be computed in order to determine the whole matrix.

Remark 2: From the above description, it follows that now the order of measurements plays a role, since different decision sequences $\boldsymbol{A}_a^{(1)}, \boldsymbol{A}_a^{(2)}, \ldots, \boldsymbol{A}_a^{(K)}$ and $\boldsymbol{A}_b^{(1)}, \boldsymbol{A}_b^{(2)}, \ldots, \boldsymbol{A}_b^{(K)}$ lead to different covariances $\mathbf{C}^{(k)}$, even when $\sum_{k=1}^{K}\boldsymbol{A}_a^{(k)} = \sum_{k=1}^{K}\boldsymbol{A}_b^{(k)} = \boldsymbol{A}$. Secondly, taking multiple measurements between two nodes will improve the relative positioning information, but will lead to more correlation. Hence, there is less benefit compared to the Measure-then-Localize approach of consecutively measuring multiple times between the same nodes.

The problem (4) can be approximated as follows:

$$\begin{aligned}\underset{K,\boldsymbol{A}^{(k)}}{\text{minimize}} \quad & \sum_{k=1}^{K}\sum_{(i,j)\in\mathcal{E}} a_{ij}^{(k)} \\ \text{s.t.} \quad & \sum_{(i,j)\in\mathcal{E}} a_{ij}^{(k)} = 1, \forall k \\ & \mathbf{C}^{(k+1)} = f(\mathbf{C}^{(k)}, \boldsymbol{A}^{(k+1)}) \\ & \mathrm{tr}[\mathbf{C}_i^{(K)}] \leq \kappa, \ \forall i\end{aligned} \tag{16}$$

in which the function $f(\cdot)$ executes the procedure listed above. While $\mathbf{C}^{(k)}$ can be calculated in closed form after each measurement, it is extremely difficult to find an optimal scheduling scheme to reduce uncertainty below κ with the smallest number of measurements. In particular, the long-term benefit is more difficult to consider than the instantaneous reduction in uncertainty. However, this type of problem in now in a form where RL can be applied.

III. Deep Reinforcement Learning for Scheduling Optimization

In this section, we formulate the original scheduling problem in the DRL framework and introduce the training algorithms with DQN and PG.

A. Problem Formulation

1) Single Agent Case: DRL is an area in machine learning that optimizes a policy of an agent (in the considered problem, an agent is a link between two nodes) when interacting with an environment with the objective to maximize the long term cumulative reward. The interaction between agent and environment is described as an MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $s \in \mathcal{S}$ is the state of the agent, $a \in \mathcal{A}$ is an action of the agent, $\mathcal{P}$ describes the transition density $p(s'|s,a)$ from the current to the next state, and $\mathcal{R}$ describes the instantaneous reward $r(s,a)$ (or more generally $r(s,s',a)$) and $\gamma \in [0,1]$ is the discount factor. In RL, an *agent* can take an action a according to a policy π given a state s. The agent obtains reward r as feedback of action a from the environment and updates state from s to s'. In summary, the data item (s,a,r,s') characterizes one interaction between agent and environment. In the next time step, a new action will be taken, given the state s'. In order to collect enough data to train the model, the training process involves many *episodes*. In each episode, the nodes begin with their initial PEBs (one anchor with low PEB and other nodes with high PEBs) and reduce their PEBs until every node achieves the objective.

2) Multiple Agent Case: If there are multiple agents interacting with the environment and the reward of each agent depends on the actions of other agents, the RL problem becomes an MARL problem. In our case, all agents behave individually but are governed by the same policy (as they have the same objective). In formulating the agent's policy (particularly for DQN), the other agents are considered as part of the environment. Therefore, if the policy changes, the environment changes as well. Since the agent's reward depends on the actions of other agents, the reward is issued before the next action of the same agent (i.e., after the actions of other agents). This is a crucial difference to the single-agent RL. When the agent does not have access to the environment state, the MDP becomes a POMDP, which is described by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \Omega, \mathcal{O})$, in which $o \in \Omega$ is the observation and $\mathcal{O}$ describes the observation probabilities $p(o|s)$. The action is then a function of the observed state (as well as the state history), not the true state. Such a situation is relevant in our context, as each agent has access only to local information, which in turn is affected by decisions of other agents.

B. Solution Strategies

The objective of this section is to develop algorithms that perform scheduling for cooperative localization (4), such that the constraint of objective PEB is satisfied for every node in the scenario with minimum number of measurements. As Section II-D points out, it is extremely difficult to solve this problem analytically. Therefore, we mention two standard solutions using DRL, which learn to make decisions according to the *experience* in the form of simulated data [40]. The two solutions are based on the two major categories of DRL, namely DQN and PG, which are elaborated as follows.

1) DQN: DQN focuses on estimation of the expected long-term reward of available actions, defined as Q-values, and the implicit policy π, is to choose the action that maximizes the Q-value. The Q-value given state s, action a and policy π in

time step T is expressed as

$$Q_T^{\pi}(s,a) = \mathbb{E}\left[\sum_{t=T}^{+\infty} \gamma^{t-T} r_t(s_t,a_t) | s_T = s, a_T = a, \pi\right] \quad (17)$$

where $\mathbb{E}(\cdot)$ is the expectation operator, T is the time step under consideration, s_t and a_t are state and action in time step t, respectively, $r_t(s_t,a_t)$ is the instantaneous reward at time step t and given state s_t and action a_t. The optimal Q-value is given by $Q^*(s,a) = \max_\pi Q^\pi(s,a)$ and satisfies the Bellman equation [21]:

$$Q^*(s,a) = \mathbb{E}_{s'}\left[r(s,a) + \gamma \max_a Q^*(s',a') | s,a\right]. \quad (18)$$

The optimal policy should choose the action that maximizes the Q value under every possible state, i.e.,

$$\pi^*(s) = \arg\max_a Q^*(s,a) \quad (19)$$

for any $s \in \mathcal{S}$.

If the number of available states and actions is small, we can use a look-up table to exhaustively list the expected Q-values for each (state, action) pair. However, if $\mathcal{S}$ or $\mathcal{A}$ is continuous or very high dimensional, which is the case of the addressed problem in this paper, the possible values cannot be presented in such a table. In this case, this look-up table can be approximated by a DNN with parameter set θ, denoted as $Q(s,a;\theta)$. This approach is referred to as DQN. DQN is an off-policy method, which allows it to use training data generated by a different policy than the one currently being optimized.

In episode i, θ_i is optimized to to minimize the mean square error (MSE) between output of the DNN and the Q-values calculated by the instantaneous rewards and Q-values obtained from the previous training. The loss of one data item is therefore calculated as

$$L_i(\theta_i) = \mathbb{E}_{s,a}\left[\left(y_i - Q(s,a;\theta_i)\right)^2\right] \quad (20)$$

where y_i is a target value given by

$$y_i = r + \gamma \max_{a' \in \mathcal{A}} Q(s',a';\theta_{i-1}), \quad (21)$$

with $y_i = r$ when s' is a terminal state. The expectation (20) is approximated by an average over a training database and the minimization is performed via a gradient descent method (ADAM in our work). As in other machine learning problems, DQN must take the exploitation-exploration trade-off [41] into account. Therefore, we apply ε-greedy in the training. Namely, we select a random action with the probability of ε and the action with the highest Q-value with the probability of $1-\varepsilon$, where ε is a small and decaying value with the episodes.

2) PG: While DQN estimates the Q-values and formulates the policy implicitly by choosing the action with highest Q-value or with the ε-greedy policy, PG optimizes the policy explicitly, which determines the action given a state. We represent the stochastic policy by a DNN parameterized by θ. Under a stochastic policy $\pi(a|s;\theta)$, there is a natural exploration. The parameter θ should be optimized to maximize the expected cumulative reward, defined as

$$J(\theta) = \mathbb{E}_{\tau\sim p(\tau;\theta)}[r(\tau)] = \mathbb{E}_{\tau\sim p(\tau;\theta)}\left[\sum_{t=0}^{H-1} r(s_t,a_t)\right] \quad (22)$$

where τ is the path of states, actions from s_0, a_0 to s_{H-1}, a_{H-1} with H the maximum number of time slots in an episode, $r(\tau)$ is the sum of rewards on path τ (defined as *path return*), $p(\tau;\theta)$ is the probability of path τ given policy θ, which is computed as

$$p(\tau;\theta) = p(s_0)\prod_{t=0}^{H-1} \pi(a_t|s_t;\theta) p(s_{t+1}|s_t,a_t) \quad (23)$$

with $p(s_0)$ the probability of initial state s_0, H the number of time slots, $\pi(a_t|s_t;\theta)$ the probability of choosing action a_t given state s_t and policy θ, $p(s_{t+1}|s_t,a_t)$ is the probability of state s_{t+1} in the next time step given current state s_t and action a_t. According to the REINFORCE algorithm [22], the gradient of $J(\theta)$ with respect to θ is calculated as

$$\begin{aligned}\nabla_\theta J(\theta) =& \mathbb{E}_{\tau\sim p(\tau;\theta)}[(r(\tau)-b)\nabla_\theta \log p(\tau;\theta)] \\ \approx& \frac{1}{N}\sum_{i=0}^{N-1}\sum_{\tau}(r_i(\tau)-b)\nabla_\theta \log \pi(a_{i,t}|s_{i,t};\theta)\end{aligned} \quad (24)$$

where b is an action-independent baseline and N is the sample size. In this paper, b is defined as the mean path return. In each iteration, the gradient ascent makes paths with high rewards more likely to appear in the future. This is equivalent to improving expected rewards with paths generated under the new policy.

C. Formulation of Network Localization as a RL Problem

We assume that the network has a baseline schedule, where each time an agent is scheduled, it needs to decide whether or not to measure. Not measuring takes no time, while measuring comes at a cost. We will now describe the network localization problem as a POMDP.

1) POMDP Description: In the problem considered in this paper, we assume that an agent cannot observe the global state in order to make the proposed algorithm more practical (hence the global state is *partially observable*). The state is therefore defined to contain the local state and a limited amount of global information (which is easy to observe). Formally, we introduce the following POMDP:

- *Agents:* The agents are the links $(i,j) \in \mathcal{E}$ in the network.
- *Actions:* $\mathcal{A} = \{0,1\}$, corresponding to the decision of not measuring ($a=0$) or measuring ($a=1$). Agents locally decide whether or not to measure, where not measuring takes up negligible time, while measuring takes significant time.
- *States:* $\mathcal{S}$ comprises the global state of the network, including the true locations of all nodes (say, $\mathbf{x}_i$), as well as the estimated locations ($\hat{\mathbf{x}}_i$) and the global covariance matrix $\mathbf{C}$.
- *State transitions:* $\mathcal{P}$ is determined by the evolution of the full network covariance, as described in Section II-D2. The evolution of the estimates depends on the specific localization algorithm. During training, the means are generated as $\hat{\mathbf{x}} = \mathbf{x} + \mathbf{C}^{1/2}\mathbf{w}$, in which $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{2N})$.
- *Observations:* Ω is the local observation, available to each agent, given by the projection of $\mathcal{S}$ onto the following vector

$$o = \{\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j, \mathbf{C}_{ij}, n_{ij}\} \quad (25)$$

where n_{ij} is the number of neighbors of the involved nodes that have not yet achieved the target PEB κ. This observation tells the agent how many nodes need its help and a large n_{ij} motivates the agent to measure. Please note that this definition of Ω fully determines $\mathcal{O}$. To remain consistent with the standard DRL terminology, we call denote the observation as the local state, since it does not introduce any ambiguity.

- *Rewards:* In this particular problem, we define $\mathcal{R}$ as follows. We first introduce an immediate (deterministic) reward:

$$r_{ij}^{\text{im}} = \begin{cases} 0 & a = 0 \\ m \cdot r^{\text{final}} - c^{\text{meas}} & a = 1 \end{cases} \tag{26}$$

where $c^{\text{meas}} \geq 0$ represents a fixed measurement cost, $r^{\text{final}} \geq 0$ is a *positioned* reward given once, when a node's uncertainty falls below the threshold, and $m \in \{0, 1, 2\}$ is the number of nodes that reduce their uncertainty below the threshold as a consequence of the action. Secondly, we introduce a long-term (stochastic) award:

$$r_{ij} = r_{ij}^{\text{im}} + \alpha \frac{\sum_{w=1}^{W} r_w^{\text{im}}}{W} \tag{27}$$

where W is the number of time slots in between the times when agent (ij) acts. Here r_w^{im} is the immediate reward of another agent ($w \neq (ij)$) that acted in between two consecutive actions of agent (ij), and $\alpha \geq 0$ is a parameter that encourages altruism in the agent.

Remark 3: The observable state only contains local information (i.e., information of nodes i and j), such that a decentralized decision process is possible. The observable state can be extended to include estimates with respect to one-hop or two-hop neighbors, and the covariances $\mathbf{C}_k$ of these neighbors. Note that the dimensionality of the observable must be made constant, so it should either compress the neighbor's information or consider a fixed number of neighbors (e.g., an upper bound).

Algorithm 1 DQN Training for Decentralized Scheduling

```
Initialize DNN with random θ
for episode e = 1, ..., M do
    Generate initial state s
    Initialize memory D
    for t = 1, 2, ... H do
        Select an agent (a link (ij))
        Observe state s_t of the agent
        Select a random action a_t with probability ε
        Otherwise select a_t = argmax_{a_t} Q(s_t, a_t; θ)
        Execute a_t and record r_t and s_{t+1}
        Save (s_t, a_t, r_t, s_{t+1}) in D
        if t mod P = 0 then                    ▷ gradient step
            Sample random minibatch from D
            Set y_i according to (21)
            Gradient step on (20) to update θ
        end if
    end for
end for
```

Algorithm 2 PG Training for Decentralized Scheduling

```
Initialize DNN with random θ
for episode e = 1, ..., M do
    for scenario s = 1, ..., S do
        Generate initial state s
        Initialize memory D
        for t = 1, 2, ... H do
            Select an agent (a link (ij))
            Observe state s_t of the agent
            Select action a_t ~ π(a|s_t; θ)
            Execute a_t and record r_{e,t,s} and s_{t+1}
        end for
        Compute r_{e,s} = Σ_{t=1}^{H} r_{e,t,s}
    end for
    Set baseline b = Σ_{t=1}^{H} Σ_{s=1}^{S} r_{e,t,s}/(HS)
    Gradient step on (24) to update θ
end for
```

D. Implementation Considerations

The implementation is provided in Algorithms 1 and 2. We note that we do not keep track of agent observation histories, as each agent implements the same policy. Since the state definition (25) is local, scenarios used for training and testing do not need to be identical, which broadens the generality of the algorithm, and can speed up training. As we will see later, it is possible to train on a small network and test on a larger network.

Since PG operates on the cumulative reward of an entire episode and each agent has the same policy, it is inherently global and cooperative behaviour can emerge even with $\alpha = 0$ in (26). In contrast, DQN with $\alpha = 0$ will not lead to any cooperation, as agents cannot see the benefit of their actions for the network as a whole. On the other hand, setting α to a large value will lead to large variations in the stochastic reward signal and can negatively affect learning. For that reason, we have found that PG was far easier to implement and optimize and led to more stable learning.

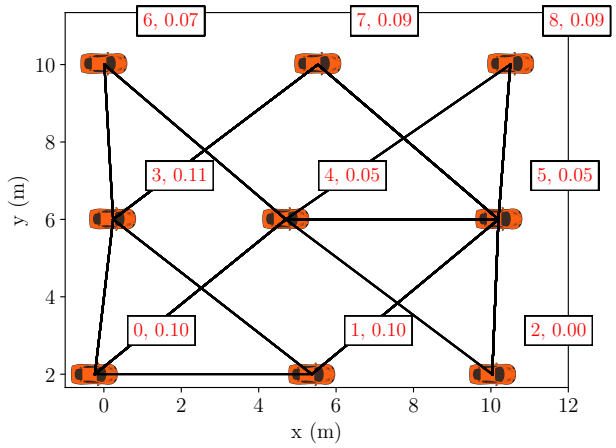
Another remark is that both algorithms are on-policy learning, i.e., they optimize the policy with data generated according to the current policy. The data are generated with the simulator described in Section II and according to the current policy. The description of the CRLB calculation in Section II and definitions of state, actions and reward in Section III provide sufficient information to reproduce the results presented in the next section.

IV. Simulation Results

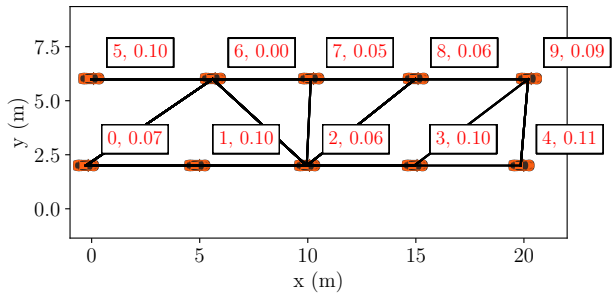
Simulation results are introduced in this section, which confirms the advantages of the proposed algorithms.

A. Setup

We defined two scenarios to train and test the proposed algorithms, which are depicted in Fig. 3: a highway scenario with 3 lanes, considering a network of 3 vehicles per lane (Fig. 3(a)) and a highway scenario with 2 lanes, considering 5



(a) Scenario of size 3×3, used for training and testing



(b) Scenario of size 2×5, used for testing

Fig. 3. Multi-lane multi-vehicle scenarios for training and testing of DQN and PG. The boxes next to each vehicle are the vehicle index and final PEB whereas the lines between vehicles are measurements.

vehicles per lane (Fig. 3(b)). To demonstrate the generalization capabilities of DRL, the first scenario is used for training, while both scenarios are used for testing. In Fig. 3, indices and PEBs of vehicles are denoted in the frames near the vehicle. For each scenario, we set a random vehicle as an anchor, with low initial uncertainty (the vehicle with the PEB of 0.00 in Fig. 3 is the anchor vehicle), while all remaining vehicles are nodes with high initial uncertainty (this high uncertainty is not shown in the figure). In order to achieve a balance between exploration and exploitation in DQN, ε-greedy is used during training, where ε reduces linearly from 1 (in episode 0) to 0 (in episode 350) and remains 0 until the end of the training.[1] Additional simulation parameters during training are provided in Table I. The DNN parameters for DQN and PG are listed in Table II and Table III, respectively. These parameter values were determined empirically. 1000 independent scenarios are applied in testing to make the results sound in a statistical sense.

During training we compute the FIM based on the true positions, while in testing the FIM is based on the estimated positions.

[1]Due to the MARL nature, the actions of other agents are part of the environment and influence the Q-values. Therefore, ε must be reduced to 0 at the end of the training to make the environment consistent with the environment in testing.

TABLE I
SIMULATION PARAMETERS

Parameter	Value
σ_l	0.1 m [1]
σ_α	0.1° [1]
γ (discounting factor)	0.75
Cost of measurement	0.1
Terminal reward	1.2
Initial PEB of normal vehicles (m)	3.4 [42]
Initial PEB of anchors (m)	0.0
Objective PEB κ (m)	0.12
Number of scenarios in PG training	100
Number of scenarios in testing	1000

TABLE II
DQN PARAMETERS

Parameter	Value
Number of layers	4
Number of neurons per hidden layer	100
Activation function	ReLU
Loss	MSE
α (altruism)	3
Optimizer	ADAM
Initial learning rate	5×10^{-5}
Batch size	128
Episodes	650
Number of scenarios in training	40

B. Performance Metrics and Benchmarks

The objective is for the vehicles to reduce their PEBs below a given threshold κ by means of cooperative localization, i.e., radar measurement of relative positions between the vehicles and information sharing of current position estimates, with minimum number of measurements. Hence, the performance metrics are:

1) *Efficiency:* the number of measurements needed to bring all PEBs below the objective and the number of vehicles that have achieved the objective.
2) *Outage probability:* the fraction of vehicles that fail to achieve the objective.
3) *Realized PEB:* the PEBs that is achieved by the vehicles upon completion of the methods.

Performance is evaluated for DQN and PG and two benchmarks: a random policy (which decides randomly whether to measure or not) and a greedy policy [43] (which chooses to measure if and only if the instantaneous reward defined in (26) is positive).

TABLE III
PG PARAMETERS

Parameter	Value
Number of layers	4
Number of neurons per hidden layer	100
Activation function	ReLU
α (altruism)	0
Optimizer	ADAM
Episodes	2000
Initial learning rate	1×10^{-4}
Number of scenarios in training	100

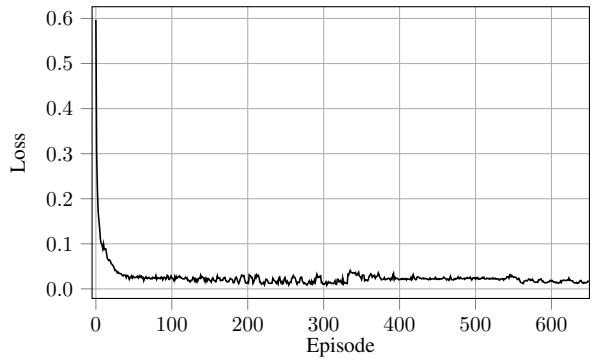


Fig. 4. Loss in DQN training as a function of the number of training episodes.

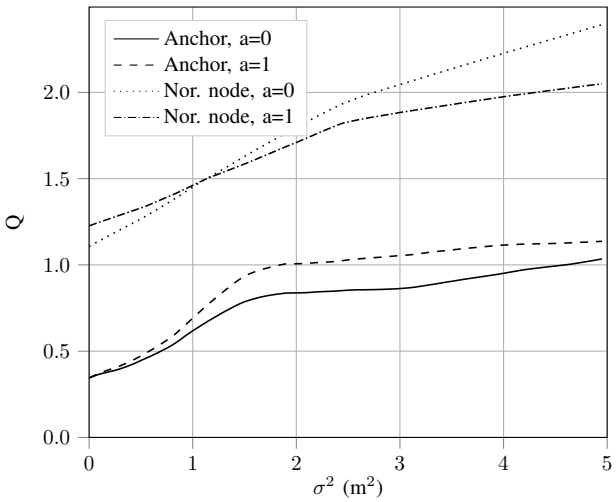


Fig. 5. Q-values as functions of priori location variance for a normal adjacent vehicle and an adjacent anchor.

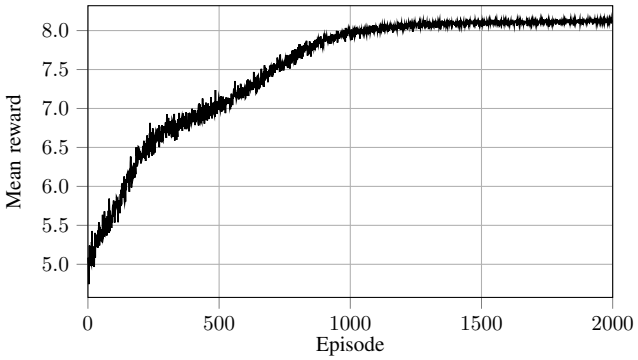


Fig. 6. Mean rewards in PG training.

C. Training Performance

1) DQN: The training of the DQN is carried out with Algorithm 1. The evolution of the loss (20) is shown in Fig. 4. The training loss (MSE) reduces from roughly 0.6 to less than 0.02 within 50 episodes and remains stable. We now look more detail to the Q-values as a function of variances of vehicles Fig. 5 shows Q-values of two actions for an agent between two normal vehicles (denoted as "nor." in the legend) and an agent between an anchor and a normal vehicle. The variances of one vehicle are assumed constant (PEB is 3.4 for normal vehicle and 0 for anchor) and the variances of the other vehicle are shown in the horizontal axis for both x and y directions. It can be observed that (i) except for very small values of σ^2, it is preferred to measure with an anchor vehicle. When $\sigma^2 < 0.01$ m^2 (corresponding approximately to the PEB of 0.12 m), then it is preferred not to measure; (ii) similarly, measuring with normal vehicles, is only performed if the variance is low enough such that the other vehicle can benefit from the measurement. If the variances of both vehicles are high, the information exchange between them would not bring significant advantage to compensate for the measurement cost; (iii) Q-values with a normal vehicles are higher than Q-values with an anchor, because the former has two chances to obtain rewards whereas the latter has only one.

2) PG: Unlike DQN, PG optimized the expected path return directly (Algorithm 2), which is approximated by the mean path return of 100 scenarios in the training, as shown in Fig. 6. We can observe that the mean reward is improving steadily in the training. After 2000 episodes, the probabilities of actions returned by the neural network are either close to 0 or close to 1, i.e., the stochastic policy reduces to (almost) deterministic policy. Therefore, the exploration stops and the policy can not be optimized further.

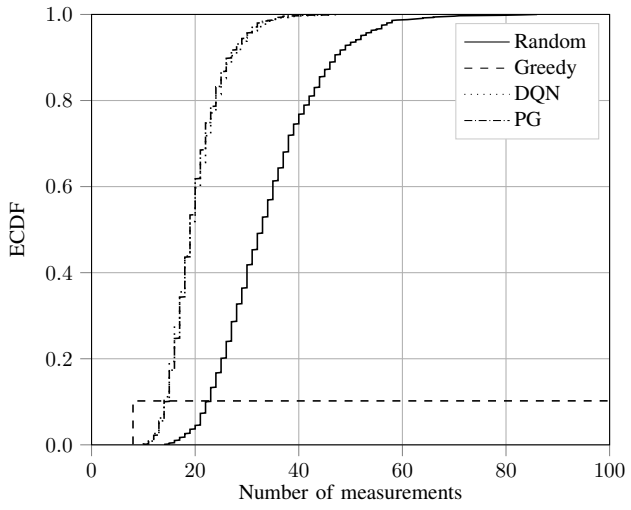
3) Comparison between DQN and PG: Comparing the parameters in Table II and Table III as well as the results, we can conclude that DQN is more sample-efficient than PG. However, the algorithm description in Section III shows that the algorithm complexity of DQN is higher than PG because we need to design the reward carefully to encourage cooperation between agents. On the other hand, PG is cooperative in its nature because all agents follow the same policy and the episode reward will increase when agents cooperate. In addition, DQN requires more careful parameter tuning than PG in our experience.

D. Testing Performance

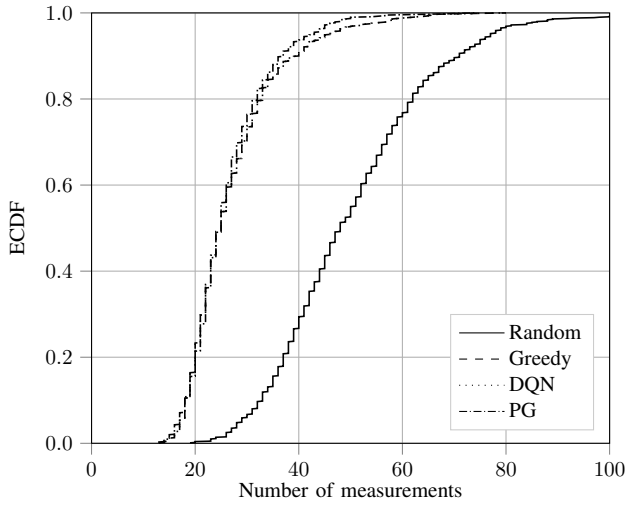
In this section, we present the simulation results to evaluate the performance of the DRL.

1) Detailed Example: We first consider the detailed example of Fig. 3, which shows the measurements (black lines) and the final PEBs of each vehicle (red text). The anchor vehicle has a PEB of 0 m and the normal vehicles have an initial PEB of 3.4 m. As depicted in the figure, the PEBs of normal vehicles have been successfully reduced below the objective of 0.12 m. Note that the order of measurements cannot be illustrated in Fig. 3 due to the limit of the paper length.

2) Statistical Analysis: In a more general and statistical perspective, Fig. 7 shows the empirical cumulative distribution functions (ECDFs) of the number of measurements needed to localize all the vehicles for the four methods for scenarios with sizes of 3×3 and 2×5. It turns out that except the greedy method, all methods are able to reduce the PEBs below the threshold. We can observe that DQN and PG perform almost equally well in both scenarios, while PG has a slightly shorter tail in the second scenario. Both RL algorithms



(a) 3×3 network



(b) 2×5 network – greedy policy is invisible because the greedy policy never meets the objective.

Fig. 7. ECDFs of numbers of measurements for different networks.

outperform the random policy considerably. Since the greedy policy only considers the immediate reward, it can only reduce the PEBs of all normal vehicles below the objective if the anchor vehicle is at the center of the first scenario (3×3), where all normal vehicles are adjacent to the anchor and the anchor can reduce their PEBs below the objective with one measurement. Therefore, the ECDF of the greedy policy has 8 measurements (one measurement for each normal vehicle) with the cumulative probability of 0.12 (roughly 1/9) and infinite number of measurements above it (because the objective is never achieved with other anchor positions) in the first scenario and the measurements are constantly infinite in the second scenario. On the other hand, the random policy decides whether to measure or not by chance. Therefore, given enough time (which is the case of our test simulation), the fraction with random policy can always tend to 1. However, this comes with a lot of unnecessary measurements, hence a very low efficiency. A further observation is that the advantages of both DRL algorithms are bigger in the second scenario, indicating

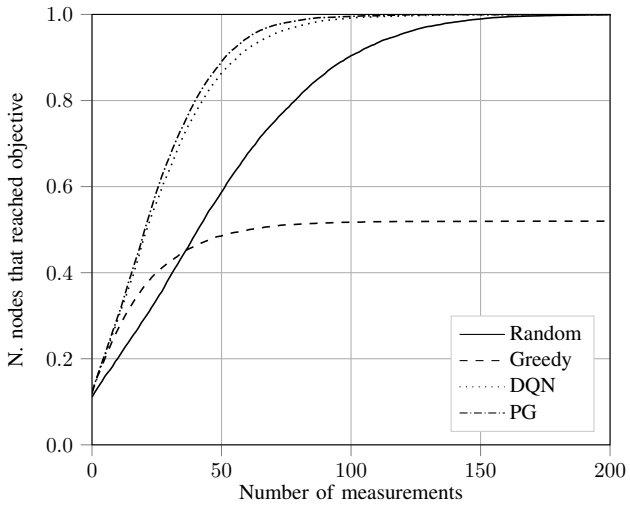


Fig. 8. Fraction of vehicles that have achieved the objective as a function of time.

that a more complicated scenario requires the sophisticated RL more than a simple scenario.

Fig. 8 shows the fraction of vehicles that have achieved the objective as a function of time for the four policies in the testing scenario. It is obvious that vehicles achieve the objective faster with both DRL algorithms than with random and greedy policies (PG performs slightly better than DQN). Due to the reason stated above, some vehicles never achieve the objective under the greedy policy.

To gain understanding in the realized PEB values of the 4 methods, Fig. 9 shows the ECDFs of PEBs after 10 measurements in the testing scenario. The figure provides an intuitive impression of achievable PEBs under resource constraint for limited numbers of measurements. Around 11% of the PEBs are 0 (bottom left corner of the graph), which are the anchors (1 out of 9 vehicles). Similar to the results shown before, the two DRL algorithms have similar and better performances than the random and the greedy policies. more than 80% of all PEBs are reduced below 0.3 m^2 with 10 measurements, which is considerably better than random and greedy policies. Besides, the random policy has a considerably higher spread than the other three policies due to its random nature. These facts demonstrate the advantage of the proposed algorithm in the resource-limited situation.

From the results above, we can conclude that the proposed decentralized scheduling algorithms with DRL are able to reduce all PEBs below the objective with fewer measurements (Fig. 7) and reduce the PEBs to lower levels with limited numbers of measurements (Fig. 9) than the random and greedy policies. The advantage is achieved by means of optimized scheduling, in particular, by maximizing the effect of one measurement (e.g., a measurement between two normal vehicles does not decrease the PEB much and should be avoided) and cooperation between agents (i.e., agents take rewards of other agents into account). Although the algorithms are trained in a specific scenario, they can operate in different scenarios, because they are designed to be decentralized and the vehicles require only local information for the optimal decision, which

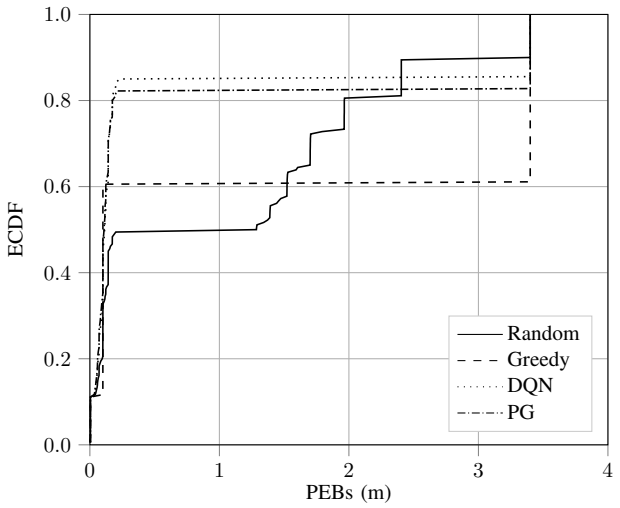


Fig. 9. ECDFs of PEBs after 10 measurements.

does not depend the global scenario. Our results show that the correlation between local state and actions is sufficient such that the proposed algorithms outperform the base lines (random and greedy policies) considerably. The reason is that the cooperation mechanism does not depend on the scenario setup. Despite that the global information is incomplete (e.g., how many vehicles depend on the considered vehicle globally to reduce their PEBs) and the agents do not have a global picture as a consequence, the trained models still prove themselves valid in a bigger scenario in a statistical sense, which is confirmed by the simulation results. The generality of the proposed algorithm is thus demonstrated.

V. Conclusion

This paper studied the problem of cooperative localization of vehicles in the context of multi-agent reinforcement learning. Cooperative localization is an important approach to improve localization precision and coverage. However, the measurement between nodes causes delays, which is particularly detrimental for vehicular applications. Hence, measurement scheduling is an important problem. We have proposed a novel formulation of the scheduling problem to account for measurement ordering and thereby can transform the cooperative localization problem as a POMDP, whereby state transitions and rewards are computed based on the PEB, which is as a general measure of localization accuracy. We have shown that the optimal scheduling problem is difficult to solve analytically especially in a decentralized manner, where the nodes make decisions based on the local information without coordination of a central unit. We propose to solve this problem with DRL, which optimizes the policy based on the rewards it obtains after executing an action according to the state. Two DRL algorithms, DQN and PG, are applied to solve the problem. Simulation results show that both methods outperform random and greedy policies in terms of required numbers of measurements. With limited number of measurements, the DRL algorithms also reduce PEBs considerably more than random and greedy policies. We found that DQN required more tuning of parameters and reward definition, while PG was able to perform well in its standard form.

Acknowledgment

The authors would like to thank Mr. Z. Zhao and Mr. Z. Xu for their helpful advices. TensorFlow [44] is applied for implementation of the DRL and the training is carried out with Swedish national infrastructure for computing (SNIC).

References

[1] F. de Ponte Müller, "Survey on ranging sensors and cooperative techniques for relative positioning of vehicles," *Sensors*, vol. 17, no. 2, p. 271, 2017.

[2] H. Wymeersch, G. Seco-Granados, G. Destino, D. Dardari, and F. Tufvesson, "5G mmwave positioning for vehicular networks," *IEEE Wireless Communications*, vol. 24, no. 6, pp. 80–86, 2017.

[3] M. Tao, W. Wei, and S. Huang, "Location-based trustworthy services recommendation in cooperative-communication-enabled internet of vehicles," *Journal of Network and Computer Applications*, vol. 126, pp. 1–11, 2019.

[4] Z. Ma, H. Yu, W. Chen, and J. Guo, "Short utterance based speech language identification in intelligent vehicles with time-scale modifications and deep bottleneck features," *IEEE transactions on vehicular technology*, 2018.

[5] A. Siddiqa, M. A. Shah, H. A. Khattak, A. Akhunzada, I. Ali, Z. B. Razak, and A. Gani, "Social internet of vehicles: Complexity, adaptivity, issues and beyond," *IEEE Access*, vol. 6, pp. 62 089–62 106, 2018.

[6] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 427–450, 2009.

[7] W. Dai, Y. Shen, and M. Z. Win, "Distributed power allocation for cooperative wireless network localization," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 1, pp. 28–40, 2015.

[8] T. Zhang, A. F. Molisch, Y. Shen, Q. Zhang, H. Feng, and M. Z. Win, "Joint power and bandwidth allocation in wireless cooperative localization networks." *IEEE Trans. Wireless Communications*, vol. 15, no. 10, pp. 6527–6540, 2016.

[9] T. Wang, Y. Shen, S. Mazuelas, and M. Z. Win, "Distributed scheduling for cooperative localization based on information evolution," in *Communications (ICC), 2012 IEEE International Conference on*. IEEE, 2012, pp. 576–580.

[10] S. Dwivedi, D. Zachariah, A. De Angelis, and P. Handel, "Cooperative decentralized localization using scheduled wireless transmissions," *IEEE Communications Letters*, vol. 17, no. 6, pp. 1240–1243, 2013.

[11] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Communications*, vol. 24, no. 2, pp. 98–105, 2017.

[12] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press, 1998.

[13] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[14] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 197–210.

[15] L. Chen, J. Lingys, K. Chen, and F. Liu, "Auto: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 2018, pp. 191–205.

[16] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang, "Experience-driven networking: A deep reinforcement learning based approach," *arXiv preprint arXiv:1801.05757*, 2018.

[17] Q. Qi, J. Wang, Z. Ma, H. Sun, Y. Cao, L. Zhang, and J. Liao, "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, 2019.

[18] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 44–55, 2018.

[19] Y. Zhang, B. Song, and P. Zhang, "Social behavior study under pervasive social networking based on decentralized deep reinforcement learning," *Journal of Network and Computer Applications*, vol. 86, pp. 72–81, 2017.

[20] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[22] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[23] C. Pandana and K. R. Liu, "Near-optimal reinforcement learning framework for energy-aware sensor communications," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 788–797, 2005.

[24] U. Berthold, F. Fu, M. van der Schaar, and F. K. Jondral, "Detection of spectral resources in cognitive radios using reinforcement learning," in *New Frontiers in Dynamic Spectrum Access Networks, 2008. DySPAN 2008. 3rd IEEE Symposium on*. IEEE, 2008, pp. 1–5.

[25] G. Naddafzadeh-Shirazi, P.-Y. Kong, and C.-K. Tham, "Distributed reinforcement learning frameworks for cooperative retransmission in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 8, pp. 4157–4162, 2010.

[26] N. Mastronarde and M. van der Schaar, "Fast reinforcement learning for energy-efficient wireless communication," *IEEE Transactions on Signal Processing*, vol. 59, no. 12, pp. 6262–6266, 2011.

[27] L. Peshkin and V. Savova, "Reinforcement learning for adaptive routing," in *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, vol. 2. IEEE, 2002, pp. 1825–1830.

[28] K.-L. A. Yau, P. Komisarczuk, and P. D. Teal, "Reinforcement learning for context awareness and intelligence in wireless networks: Review, new features and open issues," *Journal of Network and Computer Applications*, vol. 35, no. 1, pp. 253–267, 2012.

[29] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for dynamic spectrum access in multichannel wireless networks," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–7.

[30] R. F. Atallah, C. M. Assi, and M. J. Khabbaz, "Scheduling the operation of a connected vehicular network using deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[31] Y. S. Nasir and D. Guo, "Deep reinforcement learning for distributed dynamic power allocation in wireless networks," *arXiv preprint arXiv:1808.00490*, 2018.

[32] H. Ye, L. Liang, G. Y. Li, J. Kim, L. Lu, and M. Wu, "Machine learning for vehicular networks: Recent advances and application examples," *IEEE Vehicular Technology Magazine*, vol. 13, no. 2, pp. 94–101, June 2018.

[33] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.

[34] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008*, 2008.

[35] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems," *The Knowledge Engineering Review*, vol. 27, no. 1, pp. 1–31, 2012.

[36] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," *CoRR, abs/1507.06527*, vol. 7, no. 1, 2015.

[37] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," *arXiv preprint arXiv:1702.08887*, 2017.

[38] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," *arXiv preprint arXiv:1703.06182*, 2017.

[39] S. M. Kay, "Fundamentals of statistical signal processing," *Signal Processing*, 1993.

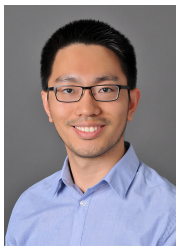
[40] L. P. Kaelbling, M. L. Littman, and A. W. survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[41] M. J. Benner and M. L. Tushman, "Exploitation, exploration, and process management: The productivity dilemma revisited," *Academy of management review*, vol. 28, no. 2, pp. 238–256, 2003.

[42] N. Alam, A. T. Balaei, and A. G. Dempster, "Relative positioning enhancement in VANETs: A tight integration approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 47–55, 2013.

[43] S. Van de Velde, G. T. de Abreu, and H. Steendam, "Improved censoring and NLOS avoidance for wireless localization in dense networks," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 11, pp. 2302–2312, 2015.

[44] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/



Bile Peng received the B.S. degree from Tongji University, Shanghai, China, in 2009, the M.S. degree from the Technische Universität Braunschweig, Germany, in 2012, and the Ph.D. degree with distinction from the Institut für Nachrichtentechnik, Technische Universität Braunschweig in 2018. He is currently a postdoctoral research in the Chalmers University of Technology, Sweden. His research interests include wireless channel measurement, modeling and estimation, massive MIMO signal processing, machine learning algorithms, in particular reinforcement learning for vehicular communication and localization.

Gonzalo Seco-Granados received the Ph.D. degree in telecommunications engineering from the Universitat Politecnica de Catalunya, Spain, in 2000, and the M.B.A. degree from the IESE Business School, Spain, in 2002. From 2002 to 2005, he was a member of the European Space Agency, where he was involved in the design of the Galileo System. In 2015, he was a Fulbright Visiting Professor with the University of California at Irvine, CA, USA. Since 2006, he has been with the Department of Telecommunications, Universitat Autònoma de Barcelona, where he has been Vice Dean of the Engineering School since 2011 and he is currently a Professor. His research interests include satellite and terrestrial localization systems. Since 2018, he has been serving as a member of the Sensor Array and Multichannel Technical Committee of the IEEE Signal Processing Society. He was a recipient of the 2013 ICREA Academia Award.

Erik Steinmetz received his M.Sc. degree in Electrical Engineering from Chalmers University of Technology, Sweden, in 2009. He is currently a research and development engineer with RISE Researche Institutes of Sweden. He is also affiliated with the Department of Electrical Engineering at Chalmers University of Technology, where he is working towards his Ph.D. degree. His research interests include positioning, sensor fusion, communication and controls applied within the fields of intelligent vehicles and cooperative automated driving.

Markus Fröhle (S'11) received the B.Sc. and M.Sc. degrees in Telematics from Graz University of Technology, Graz, Austria, in 2009 and 2012, respectively. He obtained the Ph.D. degree in 2018 in Signals and Systems from Chalmers University of Technology, Gothenburg, Sweden. From 2012 to 2013, he was a Research Assistant with the Signal Processing and Speech Communication Laboratory, Graz University of Technology. From 2013 to 2018, he was with the Communications Systems at the Department of Electrical Engineering, Chalmers University of Technology. His current research interests include signal processing for wireless multi-agent systems, and localization and tracking.

Henk Wymeersch (S'01, M'05) obtained the Ph.D. degree in Electrical Engineering/Applied Sciences in 2005 from Ghent University, Belgium. He is currently a Professor of Communication Systems with the Department of Electrical Engineering at Chalmers University of Technology, Sweden. Prior to joining Chalmers, he was a postdoctoral researcher from 2005 until 2009 with the Laboratory for Information and Decision Systems at the Massachusetts Institute of Technology. Prof. Wymeersch served as Associate Editor for IEEE Communication Letters (2009-2013), IEEE Transactions on Wireless Communications (2013–2018), and IEEE Transactions on Communications (2016-2018). His current research interests include cooperative systems and intelligent transportation.