Evaluation Mechanism for Decentralised Collaborative Pattern Learning in Heterogeneous Vehicular Networks

Cheng Qiao*, Jing Qiu*, Zhiyuan Tan†, Geyong Min‡, Albert Y. Zomaya, *Fellow, IEEE*,§ and Zhihong Tian, *Senior Member, IEEE* *

* Cyberspace Institute of Advanced Technology, Guangzhou University, China
† School of Computing, Edinburgh Napier University, Edinburgh, UK
‡ High Performance Computing and Networking, University of Exeter, UK
§ School of Computer Science, University of Sydney, Australia
mcheng.qiao@gmail.com, qiujing@gzhu.edu.cn, Z.Tan@napier.ac.uk, G.Min@exeter.ac.uk, albert.zomaya@sydney.edu.au, tianzhihong@gzhu.edu.cn

***Abstract*—Collaborative machine learning, especially Federated Learning (FL), is widely used to build high-quality Machine Learning (ML) models in the Internet of Vehicles (IoV). In this paper, we study the performance evaluation problem in an inherently heterogeneous IoV, where the final models across the network are not identical and are computed on different standards. Previous studies assume that local agents are receiving data from the same phenomenon, and a same final model is fitted to them. However, this "one model fits all" approach leads to a biased performance evaluation of individual agents. We propose a general approach to measure the performance of individual agents, where the common knowledge and correlation between different agents are explored. Experimental results indicate that our evaluation scheme is efficient in these settings.**

***Index Terms*—Internet of Vehicles, Distributed algorithm, Clustering, Similarity measurement**

I. Introduction

IoV is gaining more and more attention for it has the potential to improve driving comfort, traffic efficiency, and road safety [1, 2, 3]. Those benefits rely on a service with acceptable delay and reliable delivery of data. However, IoV is usually characterized by unreliable and limited bandwidth, which makes it a challenge to provide such a service under these rigid constraints [4, 5]. In particular, such service is crucial in some safety-related applications (i.e., traffic jam warnings) [6]. Taking collision avoidance as an example, the chances of having a collision are low if the altering of collision avoidance service is informed on time.

Federated Learning, is widely used to build high-quality ML models in the setting where many agents, each having access to their own personal data, work in collaboration to learn the desired ML models [7, 8]. This setting fits well with the IoV, where the agents are distributed in space and they work together towards the aforementioned benefits (i.e., road safety). However, it assumes that a trusted party (i.e., data sharing platform Ocean Protocol) [9]) aggregates data from those agents and distributes an updated ML model to each agent. Note that the network is prone to high-communication cost if it is orchestrated by a server or trusted third party [10]. Moreover, it is a challenge to find such a reliable and powerful orchestrator in many practical applications [11].

To this end, we consider a decentralised FL framework for heterogeneous networks and focus on clustering algorithms as (a) the shortage of labeled data [12]; (b) complex models can not be used on those agents for their high requirement on computing [13]. Taking the deep learning computation as an example, it requires high-performance computing resources with GPUs, each of which is made up of thousands of core processing units [14]. In contrast, clustering algorithms are widely used in IoV for its efficiency and computational fast benefits [15, 16].

Although there are many works concentrating on the distributed learning with clustering as methods to compute local model [17, 18, 19, 20, 21], few works have considered the performance aspects of ML models in an inherently distributed network, especially when there are sub-groups of agents receiving data from different phenomena.

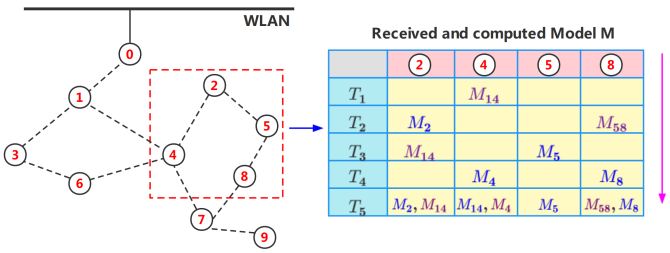

Fig. 1: An toy example for network with multiple standards: the left part shows that the basic network topology. The right part shows the models received from neighbours (denoted with purple color) and models computed by agent itself (denoted with blue color). At time stamp T_5, multiple provisional models are available to agent A_2, A_4 and A_8.

At some timestamps T_i, some agents may have learned

†† The corresponding author: Jing Qiu(qiujing@gzhu.edu.cn) and Zhihong Tian (tianzhihong@gzhu.edu.cn)

their provisional global models or received model from their neighbours. This leads to a fact that agents across the network calculated their models based on different standards (see figure 1). The diversity of models makes it a challenge to evaluate the performance of ML models in such a distributed network. The percentage of membership mismatch (PMM) is used to measure the performance of ML models by comparing the assignment of data points with the centralised baseline [22, 23]. However, this method assumes that all agents are receiving data from the same resource. When there are multiple patterns, the correctness of how agents are put into the right pattern poses a critical effect on the PMM.

It motivates us to find a new measurement that works even there are multiple standards in the network. To the best of our knowledge, there is no previous work that succeeds in measuring the performance of networks with multiple patterns. So, our question is, *how to evaluate the performance of a network with multiple patterns?* In this paper, we proposed three different methods to measure the performance of heterogeneous network, where the common knowledge and correlation between different agents are considered.

To better evaluate the efficiency of proposed methods, we compute the accuracy of agent assignment, which measures how correct the agents are put into the right pattern, as well as PMM. Specially, we measure the number of patterns and accuracy of agent assignment. Simply counting the number of patterns is not enough, since agents may be assigned to the wrong sub-patterns. Empirical experimental results show that the new methods perform better in measuring the accuracy against patterns. Moreover, we show that the right prediction improves the accuracy against data points by as much as 9%. The major contributions of this paper are summarized below:

1) We propose a privacy-preserving scheme to share information with neighbours, where the summary description of models, rather than the details, are shared.
2) We propose three different methods to measure the performance of ML models in multiple-patterns vehicular networks. These methods consider the common knowledge and correlation between different agents.
3) Empirical results show that the accuracy of agents can be improved by as much as 9%.

In the remainder of the paper, we survey related work in the next section, we then define our methods, followed by the experimental set-up, and finally we present and discuss the results of the experiments.

II. Related works

A. Federated and decentralised learning

FL aims to train a global model without centralising agents' individual data [24]. Owing to the benefits of privacy and security, FL has become the most prominent framework of distributed learning [25]. It is widely used in Cyberattack detection [26], Vehicular networks [27], Medical diagnosis system [28] and 5G wireless networks [29, 30], and zhang et al. [31] presented a comprehensive survey of various applications of FL.

However, traditional FL is orchestrated by a server or trusted third party, which is impractical in most real-world applications. This motivates the introduction of decentralised Learning [32]. Decentralized learning assumes that a central server does not exist and agents (i.e., edge devices) communicate peer-to-peer during the learning steps [33]. The learning process usually comprises: 1) a local learning step where each agent defined a summary that best describes the locally learned pattern; 2) an update step where each agent refines its local estimation based on information received from neighbours; and 3) a converge step where agents stop updating their local estimation when certain conditions, for instance, a finite number of iterations, are met.

For inferring the local model, K-means is widely used as the clustering algorithm [23, 34, 35]. Each agent runs K-means on its local data, and transmits a description of its centroids and the number of data points associated with them to its neighbours. Once an agent has received an update from all of its neighbours, it proceeds to the next round. However, the exchange of just centroids and counts also discards information about the shape of the cluster. When clusters have significantly different shapes, this may again cause problems for convergence.

The weighted average method is a prominent way of updating the model [23, 34, 36]. Finally, agents stop updating the local estimation whenever the change before and after updating is below a predefined threshold or the maximum number of iterations of updating approaches a fixed threshold.

B. Pattern detection in IoV

For the aforementioned benefits, IoV is explored in a wide range of applications, including Safety-related applications, comfort and infotainment, traffic efficiency and management, and healthcare applications [37, 38, 39, 40]. Security is another challenging issue in IoV. IoV is exposed to various types of attacks and threats, and a vehicle, which is controlled by a hacker, is more likely to cause a fatal traffic accident. Sharma et al. presented an excellent survey on security requirements, security challenges, and security attacks in IoV [41].

Clustering algorithms are widely used in IoV to identify traffic patterns. Nezerenko et al. identified countries with similar trends in the field of transportation and the main reason lead to these identified trends. Data from Baltic Sea Region (BSR) during the period 2004–2011 was used to evaluate the performance of Hierarchical Cluster Analysis (HCA) based methodology [15]. Ona et al. believed that offering high-quality services are important for public transport administrations, and they can be improved by cluster analysis [16]. The taxi-cab trips with origins and destinations in Beijing City were explored by a density-based clustering method [42]. Zhu et al. proposed data mining methods to model the dynamic traffic flow, which includes agglomerative hierarchical clustering and the k-nearest neighbor method [43].

III. Framework

As we mentioned before, the learning process usually comprises three main steps: computing local models, sharing

models and updating the provision models. In this section, we briefly describe the methods to learn the global models in multiple-pattern networks first (Section III-A), followed by a description of how to share the ML models in that network (Section III-B). Then, the measurement is given (Section III-C).

A. Learning the global models

When there are multiple patterns in the network, an agent has to run two basic procedures: identify the sub-patterns and compute the models.

1) Identify the sub-patterns: It has to run a similarity test on the models whenever it is expected to combine multiple models into one or more, or to update a model with some new information. Based on the similarity measurement, it infers the number of patterns and runs separate processes (i.e., clustering, updating and sharing) for each pattern.

Note that the number of clusters inside the model learned by different agents may be different, so the widely used L1-norm and L2-norm Euclidean distance can no longer be used. To address this problem, we propose an EMD-based distance metric [44], named weighted EMD, to measure the similarity between agents, which mitigates the high computation cost, but preserves the advantages of EMD (i.e., robust to noise and globally shape-aware). Then a similarity matrix is built on the distance metric, and the hierarchical clustering is applied on this similarity matrix. A cut-off value is carefully chosen to split the agents into different groups [1].

2) Compute the models: After the identification of patterns, agents have to compute the provisional global models for each sub-patterns. For models in the same sub-patterns, models are updated in a weighted averaged way.

Example 1. Suppose in a 10-agent network, datasets owned by 5 agents A_0, A_2, A_3, A_4 and A_8 are sampled from one distribution (also known as a pattern) and the rest of agents are sampled from another distribution. Figure 2 shows the hierarchical clustering result. The horizontal axis of the dendrogram represents the objects and clusters, and the vertical axis represents the distance between clusters. The horizontal bar between the two clusters is labeled with the distance (dissimilarity) of these two clusters. It shows that the dissimilarity between the new cluster C_1 (merged by A_0, A_2, A_3, A_4 and A_8) and new cluster C_2 (merged by A_1, A_5, A_6, A_7 and A_9) is 4.95, more than 2 times larger than its last merge 2.05. It suggests that C_1 and C_2 are dissimilar and they should not merge into a new cluster. So the identified patterns in this network are:

$$[[A_0,\ A_2,\ A_3,\ A_4,\ A_8], [A_1,\ A_5,\ A_6,\ A_7,\ A_9]]$$

B. Sharing the ML models

If an agent has learned the number of patterns and inferred the patterns, it has to decide how to share this information. To respect the privacy, it may restrict shared information to just

[1]The details of this weighted EMD method could be found in another concurrent paper. We only describe the basic procedure and main results here.

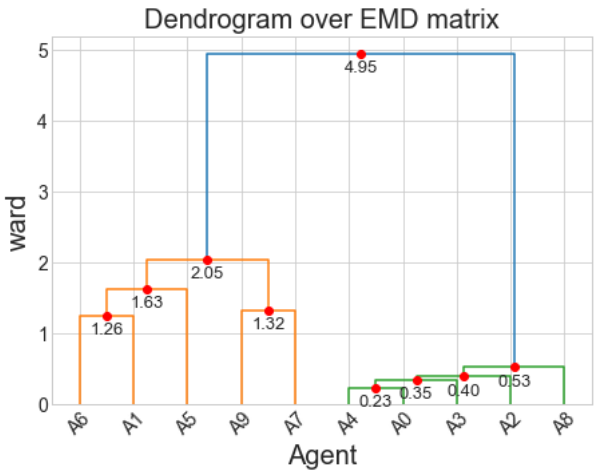

Fig. 2: Identification of patterns by Hierarchical clustering with weighted EMD as similarity metrics

the number of patterns and corresponding summary description of each sub-pattern, or it could also tell its neighbours the details of which particular sub-pattern every agent belongs to. So two different requirements are assumed at the start (see figure 3):

S_1: Each agent is told the number of patterns and summary description of each pattern, but not the individual agent assignments to the patterns.

S_2: Each agent is told the number of patterns, pattern description and details of the assignment of all agents.

Figure 3 shows the difference between these two schemes by a toy example. Suppose that agent A_4 and A_2 learned the global models (denoted as pink and blue circle, respectively). Then these two models are shared with neighbours. If agents share all details with neighbours, then all agents will learn two different global models eventually. Agent A_3, A_4 and A_5 learn the global model $[[A_1, A_2], [A_3, A_4, A_5]]$ and agent A_2 learns a model as same as A_1 ($[[A_1, A_2, A_3], [A_4, A_5]]$). The right part of figure shows the final models if the other strategy $\boldsymbol{S_2}$ is applied. The final models are the same as the previous scenario as long as they made the right prediction. Since a wrong prediction is made, agent A_5 learns a model $[[A_1], [A_2, A_3, A_4, A_5]]$, which differs from the model produced by A_4.

Although S_1 preserves some privacy for the agents, the disadvantage is that agents have to infer which sub-pattern they belong to based on their own computation. Note that this computation may introduce errors that were not present in the original inferred model that was communicated to the network. If S_2 is applied, agents are told their assignment, but its assignment is also known to other agents.

Next, we describe the sharing strategies and process steps for scenario S_1 and S_2. When agents know the group assignment of all agents (S_2), both the basic description for each pattern and the assignment of agents to patterns must be shared. In this case, each agent simply reads its assignment. If agents are not told the group assignment of all agents (S_1), each agent must infer which sub-pattern it belongs to by:

V_1: comparing its basic model with patterns received. Weighted EMD is used to measure the difference between agent and sub-patterns. Each agent chooses the pattern

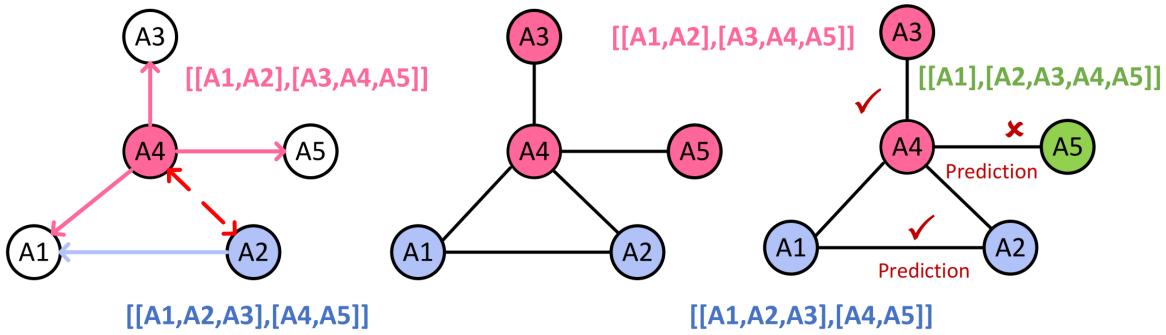

Fig. 3: The left part shows that agent A_4 and A_2 learned the global models at some timestamp t_i. These two models are not necessarily identical, so we denote them with different colours. If strategy $\boldsymbol{S_1}$ is applied, agent A_3 and A_5 received model from A_4 and agent A_1's model is produced by agent A_2 (the central part). The right part of shows a different case if strategy $\boldsymbol{S_2}$ is considered (different models are separated by colours).

with the smallest distance as its final assignment. Or

V_2: comparing its raw data with patterns received. Similarly, Standard EMD between raw data points and sub-patterns is used to measure the difference. Each agent chooses the pattern with the smallest distance as its final assignment.

C. Measurement

In this section, we introduce a general method to estimate the performance of ML models. Especially, both known and unknown assignment (S_1 and S_2) have been considered.

1) Accuracy against patterns with the known assignment (R_2): Accuracy against pattern is an important quota to measure how accurately the algorithm puts agent into the right groups against ground truth patterns. Note that predicting the right number of patterns only is not enough. It is possible that agents are put into the wrong sub-patterns, but the right number of patterns has been preserved. When the agent knows the details of the assignment of all agents (S_1), we can compute the accuracy in a pair-wise way. The accuracy against pattern is computed as:

$$Acc_p = \frac{\sum_{i=1}^{n}(Acc_p{}^i)}{n} \quad (1)$$

Where $Acc_p{}^i$ denotes the estimation by agent i. For each agent i, it computes the estimation by comparing the inferred pattern with the ground truth pattern directly, and it could be computed by the following two methods:

Me_1: For agent i and other agents in the same inferred group, are they assigned in the same group as ground truth? Count the number of pairs $\{i, j\}$ ($j \in G_i$ and $j \neq i$) that are assigned in the same group as ground truth [2], or

Me_2: For all agents in the learned pattern, are they assigned in the same group as ground truth? Count the number of pairs $\{i, j\}$ in sub-group G_m ($j \in G_m$, $i \in G_m$ and $j \neq i$) that are in the same sub-group as ground truth.

The difference between these two methods is that Me_1 only considers an agent and its neighbours in agent's group while all pairwise agents in the learned patterns are considered in Me_2.

Example 2. Suppose the pattern learned by A_0 is $[[A_0, A_1, A_2], [A_3, A_4]]$. If Me_1 method is used, all pairwise agents in the first sub-group are considered:

$$[A_0, A_1], [A_0, A_2] \textit{ and } [A_1, A_2]$$

If Me_2 is used instead, then all possible pairwise agents are considered:

$$[A_0, A_1], [A_0, A_2], [A_1, A_2] \textit{ and } [A_3, A_4]$$

The pair $[A_3, A_4]$ is not considered in measurement Me_1 but in Me_2.

2) Accuracy against patterns without known assignment (R_1): When each agent only knows the number of patterns and which sub-pattern it belongs to, how should we measure the accuracy? It is a challenge since the prediction of the assignment for agents may be based on different models. Suppose that in a 5-agent network (A_i, A_j, A_k, A_l, A_n), agent A_i and A_j produced final models (m_i and m_j) at some timestamp t_r. Then these two different models are broadcasted to the other agents.

When agents must finish with an identical model, the centralising agent A_i may have put agent A_j and A_k into the same group, but agent A_k has put itself into a different group. The accuracy measurement depends on which agent's view is used. Things are even worse when agents may finish with different models. Agent A_k receives model m_i, which puts agent A_i, A_j and A_k into a group G_1, and agent A_k puts itself in that group. Agent A_l receives model m_j, which had a group G_2 contains A_i, A_l and A_n, and A_l puts itself in that group. Now the accuracy measurement depends on all centralising agents' views.

To address the problem, we propose three different potential solutions to measure the accuracy against pattern:

1) Building a symmetric table based on agents' prediction, where agents who make the same prediction are grouped together (denoted by MD1).

[2] G_i denotes the a sub-group where agent i is in.

2) Considering both the agents' prediction and the centralised method to build an asymmetric table (denoted by MD2).
3) Using the model produced by the centralised agent directly (denoted by MD3).

2.1) MD1: symmetric table. The first one is to build a symmetric table based on agents' predictions. It could be used in scenario with an identical model or different models. Based on the prediction, agents who make the same prediction are grouped together. By this method, the common knowledge of predictions is considered, and agents who make the same prediction are grouped together.

Example 3. Figure 4 shows one example to build the symmetric matrix in a 5-agent network. Agent A_4 produced the final model and broadcast its model to the rest of agents. Then each agent makes a prediction based on the model received. Agent A_4 produced the model, so it knows which sub-pattern it is in. Agent A_2, A_3 and A_5 predict they are in sub-pattern P_2, so they are grouped together. Agent A_1 predict it is in sub-pattern P_1, so it is alone. Thus, the final table is :

$$[[A_1],\ [A_2,\ A_3,\ A_5],\ [A_4]]$$

Then, this table is compared to ground truth patterns:

$$[[A_1],\ [A_3,\ A_4],\ [A_2,\ A_5]]$$

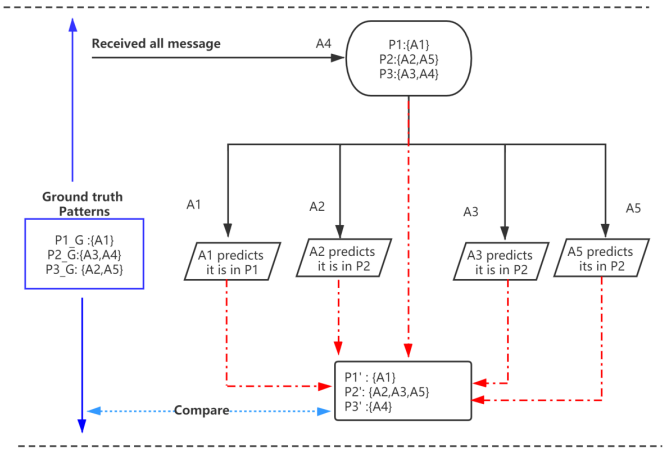

Fig. 4: Symmetric table built by agents' prediction

2.2) MD2: asymmetric table. The first method is not working if it is further extended to the case when there are multiple different models in the network, and it inspires us to propose a new method. The second method is to build an asymmetric table (denoted by MD2).

Algorithm 1 shows the algorithm to build an asymmetric table. First, the actual sub-pattern s of patterns computed by centralising agent where this agent i belongs to is obtained. Then this index s is compared to the prediction p. If agent i made a correct prediction ($p = s$), then agents that together with and apart from this agent within model M are stored. However, agent i may make the wrong prediction. In this case, agent i is inserted to the p^{th} sub-pattern of model M. Actually, agent i is put in another sub-pattern $\bar{p}$ by the centralising agent. So agent i is removed from sub-pattern $\bar{p}$. Again, the agents that together with and apart from agent i are stored. Then the asymmetric table is built when all agents find out who they are together with and apart from.

Algorithm 1: Method to estimate the relationship with other agents for individual agent

Input: Prediction p, Model received M;
Output: T_s: set of agents together with agent i, T_a: set of agents apart from agent i ;
Find the index s of agent i in the received Model M;
if $s == p$ **then**
    Add agents that together with agent i to T_s ;
    Add agents that apart from agent i to T_a;
else
    Insert agent i in the pth pattern of model M;
    Remove agent i in other patterns (not in pth pattern) of model M;
    Add agents that together with agent i to T_s ;
    Add agents that apart from agent i to T_a;

By this method, the correlation between agents is ignored. Two agents that made the same prediction based on some models may have different estimations. However, different from the symmetric table, both agent's prediction and the centralised method are considered. It is suitable for the case that there are multiple different patterns in the network.

Example 4. Figure 5 shows one example to build an asymmetric table. Agent A_2 and A_4 computed the final mode m_2 and m_4. Then model m_2 is sent to agent A_1 and m_4 is sent to agent A_3 and agent A_5. Agent A_1 predicts that it is in the sub-pattern P1. Based on the centralised agent A_2, the prediction is right and the agents that together with and apart from A_1 should be consistent with that in model m_2 (together with $[A_3]$ and apart from $[A_2,\ A_4,\ A_5]$). However, agent A_3 made the wrong prediction: it predicts it is in second sub-pattern, but actually it is in the third sub-pattern of model m_4. By inserting agent A_3 to the predicted sub-pattern and deleting the actual location of agent A_3, now agent A_3 is assumed to together with $[A_2,\ A_5]$ and apart from $[A_1,\ A_4]$, which differs significantly from the model m_4.

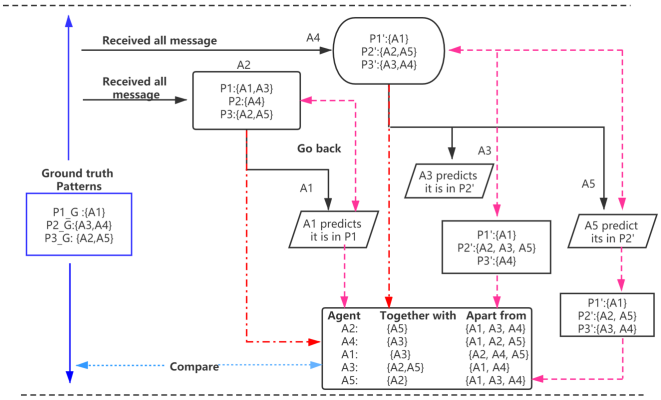

Fig. 5: Asymmetric table built by agents' prediction

2.3) MD3: Table learned by centralised agent. The last method compares the patterns learned by centralised agent with the ground truth directly (denoted by MD2). By this method, the errors introduced by a prediction by agents have been avoided.

Example 5. Take agents in Figure 5 as an example, agents A_2 and A_4 learned the global patterns

$$[[A_1,\ A_3],\ [A_4],\ [A_2,\ A_5]]$$

and

$$[[A_1],\ [A_2,\ A_5],\ [A_3,\ A_4]]$$

respectively. So we compare them with the ground truth:

$$[[A_1],\ [A_3,\ A_4],\ [A_2,\ A_5]]$$

To evaluate how accurately the algorithm puts agents into the same group as ground truth, methods MD1 and MD2 are used.

IV. Experimental results

To simulate the operation of the algorithms on a Vehicular Network, we implement an Asynchronous Message Delay Simulator, based on [45]. This experiment is tested on a 10-agent network and repeated for 20 runs. The 10-agent network is described as a connected graph, where vertices could share messages if they are connected. We assume that the message delivery is reliable, but this may require re-transmissions on failure, and so there is a message-specific delay for any transmission, which is generated uniformly from the range [0.5,1.0] [23]. Agents begin their work at a time randomly selected in [0,0.1s]. The underlying datasets are generated from fully symmetric Gaussians and K-means is used as the clustering algorithm. The initial raw data is generated in k clusters, sampled from a two-dimensional normal distribution, scaled to the range [0,1] before being assigned randomly to network agents. In the experiment, we only focus on two different patterns of networks: two patterns (5:5, 7:3, 9:1) and three patterns (5:3:2). Raw data of agents in the same sub-pattern is generated from the same normal distribution [23]. Taking the division 7:3 as an example, the first 7 agents are generated raw data (i.e., 5 cluster) from the corresponding 5 normal distributions.

To begin with, we measure the performance when the group assignment is known. Then compare two different methods to infer the group assignment when group assignment is not told, followed by building the similarity tables in three different ways. After that, we compare the performance of the proposed method with the state-of-the-art method. The notation is summarised in Table I.

Figure 6 shows the overall pipeline for performance evaluation. The three proposed methods MD1, MD2 and MD3, are measured over both the assignment is known and not known. Then two different methods M_{e1} and M_{e2} are used to compute the accuracy against patterns. After that the accuracy against data points is measured, and it is compared to the PMM (see table II).

V_1	Compare its basic model with patterns received
V_2	Compare its raw data with patterns received
M_{e1}	Count the number of pairs that are assigned in the same group as ground truth
M_{e2}	Count the number of pairs in sub-groups that are in the same sub-group as ground truth
MD1	Build a symmetric table based on agent's prediction
MD2	Build an asymmetric table that agent go back to the models received and check who it is together with and apart from
MD3	Compare the patterns learned by a centralised agent with the ground truth directly

TABLE I: Notation of different methods and measurements

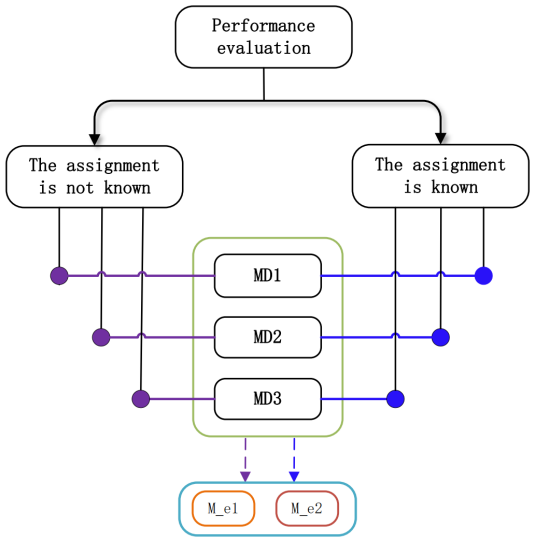

Fig. 6: The overview pipeline for performance evaluation

A. Evaluation with known assignment

In this section, we evaluate the performance of the matrix with known assignments. We compare the proposed two methods to measure how accurate the algorithm assigns agents to the same group as ground truth. The figure on the left of Figure 7 shows the results when the patterns are similar. There are few differences between these two methods, and the difference only occurs in the network with a split of 7:3, where M_{e2} performs better than M_{e1}. The most possible reason lies in the fact that it is a challenge for the proposed algorithm to predict the right group assignment when different patterns are similar. When the patterns inside the network are different, M_{e2} outperforms M_{e1} in all cases. The accuracy against patterns achieved by M_{e2} (right side of Figure 7) is higher (as much as 10%). That is because it is not difficult for the proposed algorithm to identify the group assignments for all agents if the patterns are dissimilar.

Figure 8 shows the performance when an asymmetric table is built to measure the similarity between agents. When the patterns in the network are similar, M_{e2} outperforms M_{e1} by as much as 14%. Again, the accuracy achieved by M_{e2} is higher if the patterns are different. We expected that results since the asymmetric table can describe the complex relationship between agents when there are multiple patterns in the network.

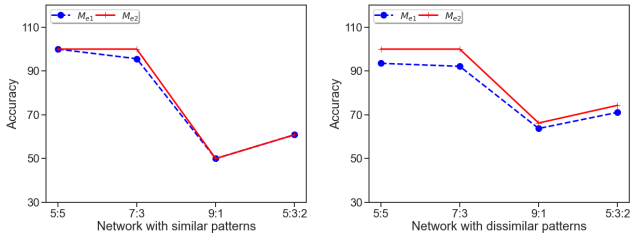

Fig. 7: Comparison results when the symmetric table is employed

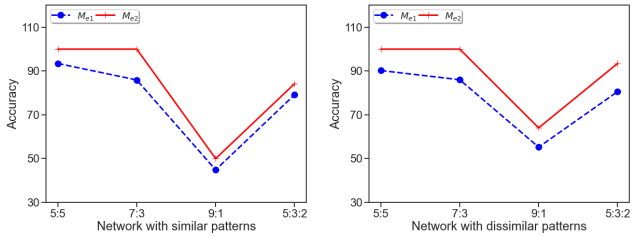

Fig. 8: Comparison results when the asymmetric table is employed

B. *Inferring the pattern with unknown assignment*

In this section, we show the comparison result on the two proposed methods (V_1 and V_2) to predict which sub-pattern the agent belongs to if the group assignment is not available. When agents are not told the group assignment of all agents, agents have to compute the similarity between their own model and the cluster model inside the patterns. It could compare the cluster model with the cluster model of patterns (V_1), or compare the raw data points with the cluster model of patterns (V_2).

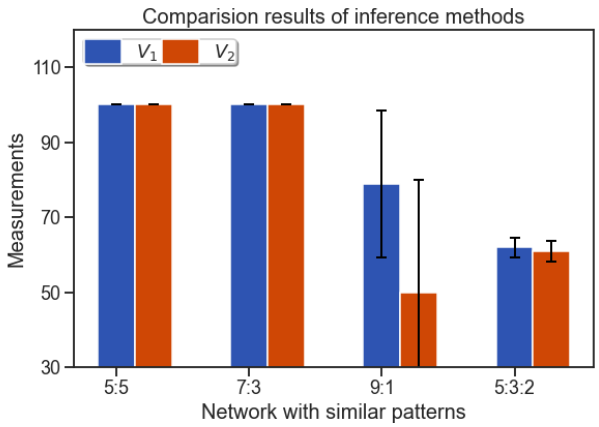

Fig. 9: Performance with symmetric tables (1)

Figure 9 shows the result when the ground truth distributions to generate data points in different patterns are similar. In terms of two patterns, there is little difference except in the case of singleton agent (9:1), where V_1 outperforms V_2 (by as much as 26%). When there are three patterns (5:3:2), the accuracy is low (below 65%). The most possible reason is that it is a challenge to predict the right group assignment if the patterns are similar.

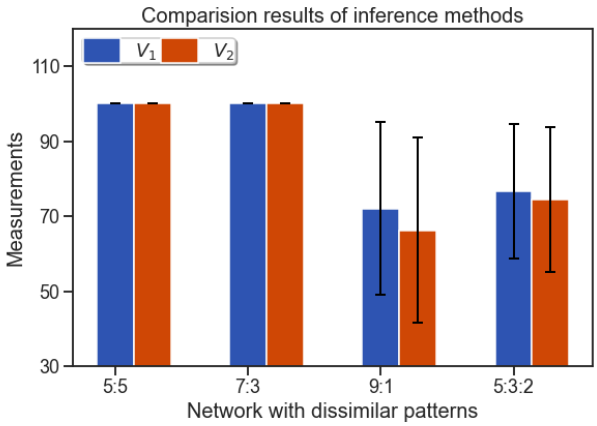

Fig. 10: Performance with symmetric tables (2)

Figure 10 shows the result when the ground truth distributions to generate data points in different patterns are dissimilar instead. As before, the only difference occurs when there is a singleton agent, where V_1 outperforms V_2. Again, method V_1 achieved higher accuracy than V_2 when there are three patterns. Since the patterns now are different, we expect a higher accuracy than that with a similar pattern (see Figure 9). The accuracy reaches 74% when there are three patterns.

Note that when there is a singleton agent, the accuracy is low even if the patterns in the network are different. That is because most clustering algorithms prefer the large size of clusters, and the predicted group assignment is wrong. But still the V_1 outperforms V_2 in that case.

Figure 11 and 12 show the results when an asymmetric table is built to measure the similarity between agents. Again, V_1 performs better than V_2. If there is only one pattern in the network and all agents infer the pattern assignment based on the same model, we expected that accuracy achieved by symmetric table is much more accurate. Since we only consider network with multiple patterns, we expected a higher accuracy. Compared to Figure 9, the accuracy against pattern, except for singleton agent, in Figure 11 is much higher.

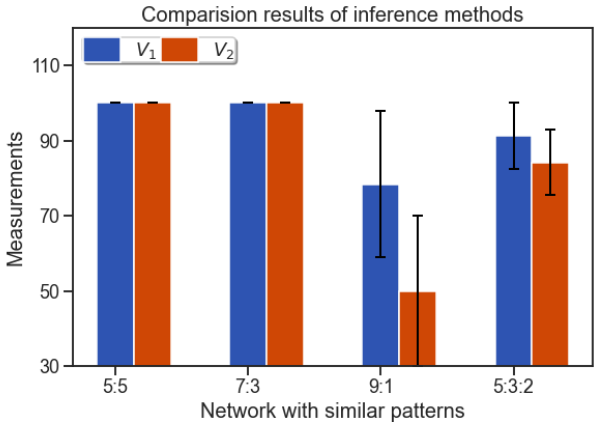

Fig. 11: Performance with asymmetric tables (1)

TABLE II: Comparison of accuracy against data points

Method	5:5	7:3	9:1	5:3:2
PMM(S)	81.71 ± 1.34	86.06 ± 3.30	88.74 ± 3.79	88.14 ± 2.01
$\widetilde{PMM}(S)$	**89.04** ± 3.63	**91.27** ± 5.21	**90.96** ± 4.18	**95.8** ± 2.92
PMM(D)	94.69 ± 2.26	94.96 ± 2.52	91.25 ± 3.53	90.04 ± 2.71
$\widetilde{PMM}(D)$	**99.17** ± 0.61	**99.49** ± 0.63	**97.98** ± 1.86	**99.18** ± 1.45

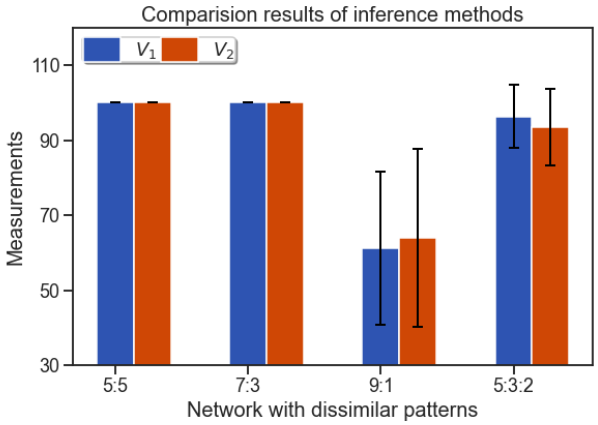

Fig. 12: Performance with asymmetric tables (2)

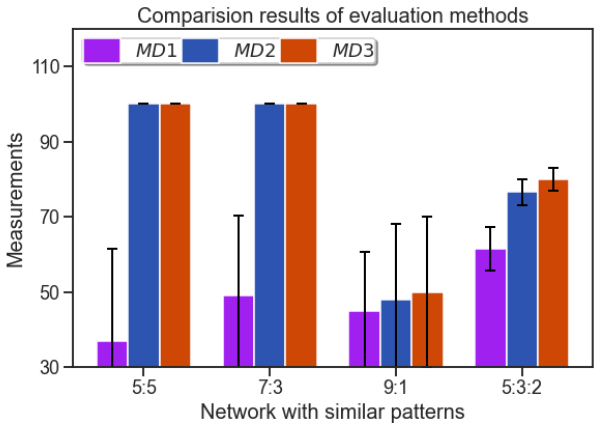

Fig. 13: Performance evaluation with similar patterns

To summarise, when agents are not told the group assignment of all agents and have to predict its group assignment, measuring its cluster models (V_1) with patterns received performs better. In addition, there is little difference in these two methods when there are only two patterns without a singleton agent.

C. Evaluation with unknown assignment

In this section, we evaluate the performance of different methods for predicting the group assignment if the group assignment is not known. After the confirmation of prediction of agents' assignment, we count the number of pairs in sub-groups that are in the same sub-group as ground truth (M_{e2}) and compute the final accuracy against patterns.

Figure 13 shows the performance when the patterns are similar in the network. MD3 performs better than the other methods. We expected this result since we compare the results, produced by centralised agent, with the ground truth directly. By this method, the prediction error has been avoided. Note that there is little difference between MD2 and MD3, which suggests that MD2 is also a potential method to measure the accuracy of patterns. MD1 performs the worst, and that is because we only consider networks with multiple patterns in this experiment.

Figure 14 shows the performance when the patterns are dissimilar in the network. Compared to the Figure 13, the accuracy against pattern is higher. The most possible reason lies in the fact that it is much easier for the proposed algorithm to detect the patterns. Again, MD3 performs the best among all methods and MD1 performs the worst.

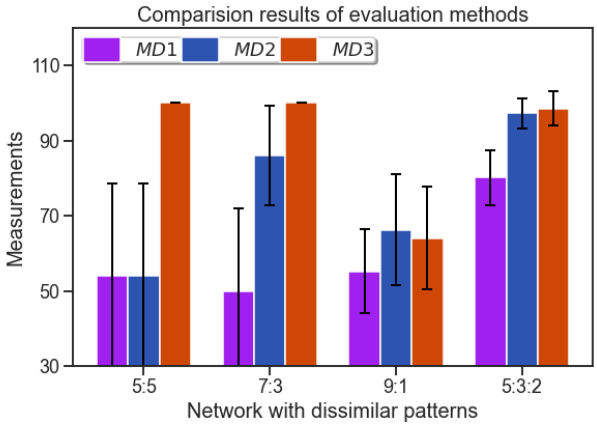

Fig. 14: Performance evaluation with dissimilar patterns

D. Comparison with the state-of-the-art

In this section, we measure the accuracy against data points, which computes how accurate the algorithm puts data points into the right clusters as ground truth, and compare the accuracy achieved by our methods $\widetilde{PMM}$ with the state-of-the-art method PMM. We consider two different types of networks: similar patterns and different patterns (denoted by the S and D in the bracket respectively). When there are multiple patterns in the network, we expect a higher accuracy against data points if the agents are put into the right patterns as ground truth. Table II shows the accuracy achieved by the proposed method $\widetilde{PMM}$ outperforms that by PMM in all cases, by as much as 9%.

V. Conclusion and future directions

In this paper, we assume that there are multiple different patterns of data being observed by different subgroups of vehicles of IoV. The aim is for vehicles to be able to identify the different patterns and their associated clusters, rather than combine the multiple patterns into a single description.

We proposed an evaluation matrix to measure the performance of group assignments for agents. Experimental results show that, the method where all possible pairs of agents are considered (known as M_{e2}) performs better in measuring the performance of agents in scenarios with single and multiple patterns. Moreover, measuring the cluster model achieves a better performance in inferring the agent assignment even the group assignment is not known in advance. Compared to the state-of-the-art, the accuracy against data points can be improved by as much as 9% if the group of agents is predicted correctly.

For a more comprehensive conclusion, we will extend the evaluation, to consider larger networks and different data distributions. We will address the problems where the distributions change over time and the requirements for personalised models are needed.

Acknowledgment

This research was supported by the National Natural Science Foundation of China under Grant No. U20B2046, Guangdong Province Key Area RD Program of China under Grant No. 2019B010137004 and Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2019).

References

[1] V. E. F. C. Borges, D. F. S. Santos, A. Perkusich, and K. M. Malarski, "Survey and evaluation of internet of vehicles connectivity challenges," in *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2020, pp. 1–6.

[2] K. Sjoberg, P. Andres, T. Buburuzan, and A. Brakemeier, "Cooperative intelligent transport systems in europe: Current deployment status and outlook," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 89–97, 2017.

[3] Y. Liu, J. Wang, J. Li, S. Niu, and H. Song, "Machine learning for the detection and identification of internet of things devices: A survey," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 298–320, 2021.

[4] Y. Sun, H. Song, A. J. Jara, and R. Bie, "Internet of things and big data analytics for smart and connected communities," *IEEE access*, vol. 4, pp. 766–773, 2016.

[5] D. Jiang, L. Huo, Z. Lv, H. Song, and W. Qin, "A joint multi-criteria utility-based network selection approach for vehicle-to-infrastructure networking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 10, pp. 3305–3319, 2018.

[6] M. Alam, J. Ferreira, and J. A. Fonseca, "Introduction to intelligent transportation systems," 2016.

[7] H. Claver, " Data sharing key for AI in agriculture," https://www.futurefarming.com/Tools-data/Articles/2019/2/Data-sharing-key-for-AI-in-agriculture-389844E/, 2019, [Online].

[8] J. J. E. Conway, "Artificial intelligence and machine learning: Current applications in real estate," Ph.D. dissertation, Massachusetts Institute of Technology, 2018.

[9] "Ocean protocol: A decentralized substrate for ai data and services," Ocean Protocol Foundation, Tech. Rep., 2019.

[10] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, 2012, p. 1223–1231.

[11] P. Vanhaesebrouck, A. Bellet, and M. Tommasi, "Decentralized Collaborative Learning of Personalized Models over Networks," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 509–517.

[12] M. S. Hadi, A. Q. Lawey, T. E. El-Gorashi, and J. M. Elmirghani, "Big data analytics for wireless and wired network design: A survey," *Computer Networks*, vol. 132, pp. 180–199, 2018.

[13] J. Park, S. Samarakoon, A. Elgabli, J. Kim, M. Bennis, S.-L. Kim, and M. Debbah, "Communication-efficient and distributed learning over wireless networks: Principles and applications," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 796–819, 2021.

[14] S. Ahn, J. Kim, E. Lim, W. Choi, A. Mohaisen, and S. Kang, "Shmcaffe: A distributed deep learning platform with shared memory buffer for hpc architecture," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, 2018, pp. 1118–1128.

[15] O. Nežerenko, O. Koppel, and T. Tuisk, "Cluster approach in organization of transportation in the baltic sea region," *Transport*, vol. 32, no. 2, pp. 167–179, 2017.

[16] R. de Oña, G. López, F. J. D. de los Rios, and J. de Oña, "Cluster analysis for diminishing heterogeneous opinions of service quality public transport passengers," *Procedia - Social and Behavioral Sciences*, vol. 162, pp. 459–466, 2014.

[17] M. Bateni, A. Bhaskara, S. Lattanzi, and V. Mirrokni, "Distributed balanced clustering via mapping coresets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. MIT Press, 2014, p. 2591–2599.

[18] O. Bachem, M. Lucic, and A. Krause, "Scalable k -means clustering via lightweight coresets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, ser. KDD '18, p. 1119–1127.

[19] A. Soliman, S. Girdzijauskas, M.-R. Bouguelia, S. Pashami, and S. Nowaczyk, "Decentralized and adaptive k-means clustering for non-iid data using hyperloglog counters," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer,

2020, pp. 343–355.

[20] D. Dennis, T. Li, and V. Smith, "Heterogeneity for the win: One-shot federated clustering," in *ICML*, 2021.

[21] C. Qiao, K. N. Brown, F. Zhang, and Z. Tian, "Federated adaptive asynchronous clustering algorithm for wireless mesh networks," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021.

[22] M. Bendechache, A.-K. Tari, and M.-T. Kechadi, "Parallel and distributed clustering framework for big spatial data mining," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 34, no. 6, pp. 671–689, 2019.

[23] G. Di Fatta, F. Blasa, S. Cafiero, and G. Fortino, "Fault tolerant decentralised k-means clustering for asynchronous large-scale networks," *Journal of Parallel and Distributed Computing*, vol. 73, no. 3, pp. 317–329, 2013.

[24] T. Li, A. K. Sahu, A. S. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, pp. 50–60, 2020.

[25] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016.

[26] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues," *Concurr. Comput.: Pract. Exper.*, vol. 28, no. 10, p. 2991–3005, 2016.

[27] W. Wang, G. Srivastava, J. C.-W. Lin, Y. Yang, M. Alazab, and T. R. Gadekallu, "Data freshness optimization under caa in the uav-aided mecn: A potential game perspective," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2022.

[28] W. N. Price and I. G. Cohen, "Privacy in the age of medical big data," *Nature medicine*, vol. 25, no. 1, pp. 37–43, 2019.

[29] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 46–51, 2020.

[30] B. D. Deebak, F. H. Memon, S. A. Khowaja, K. Dev, W. Wang, and N. M. F. Qureshi, "In the digital age of 5g networks: Seamless privacy-preserving authentication for cognitive-inspired internet of medical things," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2022.

[31] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[32] R. A. Kontar, N. Shi, X. Yue, S. Chung, E. Byon, M. Chowdhury, J. Jin, W. Kontar, N. Masoud, M. Nouiehed, C. E. Okwudire, G. Raskutti, R. Saigal, K. Singh, and Z. Ye, "The internet of federated things (ioft): A vision for the future and in-depth survey of data-driven approaches for federated learning," *ArXiv*, vol. abs/2111.05326, 2021.

[33] A. Nedic, "Distributed gradient methods for convex machine learning problems in networks: Distributed optimization," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 92–101, 2020.

[34] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *2010 IEEE International Symposium on Information Theory*, 2010, pp. 1753–1757.

[35] M. Bendechache, A.-K. Tari, and M.-T. Kechadi, "Parallel and distributed clustering framework for big spatial data mining," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 34, no. 6, pp. 671–689, 2019.

[36] R. Xin, S. Pu, A. Nedić, and U. A. Khan, "A general framework for decentralized optimization with first-order methods," *Proceedings of the IEEE*, vol. 108, no. 11, pp. 1869–1889, 2020.

[37] A. Kumar, N. Anusha, and B. S. S. V. Prasad, "Automatic toll payment, alcohol detection, load and vehicle information using internet of things & mailing system," in *2017 International Conference on Intelligent Computing and Control (I2C2)*. IEEE, 2017, pp. 1–5.

[38] Y. Wang, Z. Tian, Y. Sun, X. Du, and N. Guizani, "Locjury: An ibn-based location privacy preserving scheme for iocv," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5028–5037, 2021.

[39] C. Chen, L. Liu, T. Qiu, K. Yang, F. Gong, and H. Song, "Asgr: An artificial spider-web-based geographic routing in heterogeneous vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 5, pp. 1604–1620, 2019.

[40] W. Li and H. Song, "Art: An attack-resistant trust management scheme for securing vehicular ad hoc networks," *IEEE Transactions on Intelligent Transportation systems*, vol. 17, no. 4, pp. 960–969, 2015.

[41] S. Sharma and B. Kaushik, "A survey on internet of vehicles: Applications, security issues solutions," *Vehicular Communications*, vol. 20, p. 100182, 2019.

[42] T. Pei, W. Wang, H. Zhang, T. Ma, Y. Du, and C. Zhou, "Density-based clustering for data containing two types of points," *International Journal of Geographical Information Science*, vol. 29, no. 2, pp. 175–193, 2015.

[43] Z. Jiang, S. Li, and X. Liu, "Parameters calibration of traffic simulation model based on data mining," *Journal of Transportation Systems Engineering and Information Technology*, vol. 12, no. 6, pp. 28–33, 2012.

[44] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, pp. 99–121, 2004.

[45] R. Zivan and A. Meisels, "Message delay and discsp search algorithms," *Annals of Mathematics and Artificial Intelligence*, vol. 46, no. 4, pp. 415–439, 2006.