Continuous learning and cooperative prediction for traffic dynamics by Adaptive Multi-Agent Systems

Ha Nhi Ngo, Elsy Kaddoum, Marie-Pierre Gleizes, Jonathan Bonnet, Anaïs Goursolle

Continuous learning and cooperative prediction for traffic dynamics by Adaptive Multi-Agent Systems

Ha Nhi Ngo
IRIT
Computer Science Research Institute of Toulouse
Continental Digital Services France
ha-nhi.ngo@irit.fr

Elsy Kaddoum
IRIT
Computer Science Research Institute of Toulouse
elsy.kaddoum@irit.fr

Marie-Pierre Gleizes
IRIT
Computer Science Research Institute of Toulouse
marie-pierre.gleizes@irit.fr

Jonathan Bonnet
Continental Digital Services France
jonathan.bonnet@continental.com

Anaïs Goursolle
Continental Digital Services France
anais.goursolle@continental.com

***Abstract*—Prediction of traffic dynamics plays a significant role in many Intelligent Transportation Systems (ITS). Nonetheless, accurate and real-time traffic prediction is always a difficult task. The classical models are challenged by the complex spatiotemporal relationships of the road network that raises unsolved questions relating the reliability and the feasibility of prediction models. Nowadays, the development of localization and communication technologies in transportation has led to massive data collected by on-board sensors known as *floating car data (FCD)*. These data sets open up a new direction for traffic prediction using big data analysis methods. In this paper, we propose to address the traffic dynamics prediction problem using a self-adaptive multi-agent system that aims at continuously processing vehicle trajectory data to detect and learn different traffic dynamics and thus predict traffic evolution. The proposed system includes two processes: *local learning*, which distributes learning tasks at the agent level, and *prediction process*, which enables accurate traffic prediction using cooperative interactions among agents. The conducted experiments underline the performance of our system compared to the well-known models in the traffic prediction domain.**

***Index Terms*—Traffic prediction, dynamic clustering, Multi-Agent System, continuous learning, self-adaptive mechanisms**

I. Introduction

Nowadays, traffic congestion has become a big concern for urban mobility because of its inconveniences such as time and consumed energy waste, negative impacts on physical and emotional health of drivers, and especially high risk of collision at the end of jam queue. Providing drivers with accurate information, especially relating traffic jams can improve traffic safety. One promising approach widely used in many services of Intelligent Transportation Systems (ITS) for traffic control, planning, safety, and guidance, has been referred to *traffic dynamics prediction*. By observing future traffic dynamics on the road network estimated by this approach, we can predict the position of the traffic jam, its queue, and its propagation property.

However, estimating accurate traffic dynamics remains challenging due to complex and long-term spatiotemporal dependencies. Indeed, on the one hand, prediction models must consider the temporal dependencies of traffic data as non-linearity, non-stationarity, or seasonality, and the spatial correlations in the road network. On the other hand, the model complexity should be reasonable to adapt to large-scale applications. This trade-off is not easy to balance. In addition, the changes in the drivers' behavior when they are provided with traffic dynamics predictions must be considered to update continuously learned models. Indeed, drivers can change their itineraries given updated traffic information inducing changes in the learned temporal regularities. To deal with this problem, the learning process needs to continuously adapt and integrate all changes in the driving environment into the model. A learning method that is explainable and easy-to-update is appreciated in this case. Moreover, the recently widespread deployment of GPS localization and V2X (Vehicle-to-Everything) connectivity also brings a new generation of traffic data mining. ***Floating car data (FCD)*** along car's trajectory allow us to develop novel prediction models based on learning from data without model assumption *a priori*.

This paper presents ADRIP - a self-adaptive multi-agent system for traffic dynamics prediction. Its architecture is based on the road network description to decentralize the prediction process at the level of road segments composing the network. Agents representing road segments in the MAS implement each two parallel and autonomous processes. The first one, a **continuous learning process using dynamic clustering**, aims at dynamically learning from vehicle data streams the clusters representing the traffic dynamics on a given road segment. As collected vehicle data raises the question of specific behaviors of some vehicles that do not correspond to the observed traffic dynamics, the learning process is provided with a method enabling the detection of such singular behaviors. The second process, based on cooperation between neighboring road segments, computes **real-time traffic predictions**. This process enables cooperative behaviors of agents including self-adaptation and self-correction that help them to overcome the

conflicts during interactions and improve prediction.

II. State of the art

Many existing studies have focused on traffic dynamics prediction by estimating the mean speed, volume, or density of traffic for a given time horizon. The well-known approaches are categorized into two main groups: parametric models and non-parametric models [1].

A. Parametric models

Parametric models refer to prediction models using a finite set of parameters and a fixed structure. Data used in these models satisfy different assumptions as normality, stationarity, or no outliers existence. The most common parametric models suited for traffic prediction are the family of ARMA-based models. They show the linear dependency of future values on the previous values (Auto-Regressive - AR) and the random noise series (Moving Average - MA) with stationary assumption. ARIMA - an extended version of ARMA applied in [2] can deal with non-stationary data. Some later models can address spatial relationships such as the multivariate models called VARMA (Vector Auto-Regressive Moving Average) and STARIMA (Space-Time Autoregressive Integrated Moving Average). The results in [3] demonstrated the improvement of prediction performance by multivariate models applied to large networks with large amounts of sensors.

Parametric models are very explicit and clear to understand the relationships between parameters and outputs. The implementation and execution of these models do not require high computational capacities. They can achieve good performance in regular traffics. However, they can only solve linear problems with the strong imposed assumptions on traffic data that is not suited for complex applications and irregular traffic data.

B. Non-parametric models

Non-parametric models do not fix the dependencies between variables, the set of parameters and the model structure can change adapting to arrival data. This property brings high flexibility for models but requires training them on large and diverse data sets to calibrate parameters. Thus, vehicle trajectory data are very suited for non-parametric models. The well-known model first applied for traffic prediction is the K-nearest neighbor model (KNN). KNN finds the K closest clusters to the current traffic state and estimates the predictions by observing the next states of its k-neighbors. An adaptive spatiotemporal KNN model (ST-KNN) was developed in [4] that considered the spatial heterogeneity of road traffic. Another application of KNN was presented in [5] for traffic state prediction under special events.

Feed Forward Neural Network (FFNN) - a simple neural network, was used in [6] to estimate multiple steps of next traffic flows on multiple road segments. Paper [7] applied an improved NN with a stack of hidden layers to capture the complex dependencies between the input and output variables. The obtained MSE (Mean Squared Error) decreased by 14% compared to traditional NN with one hidden layer. However, adding multiple hidden layers to capture long-term dependencies leads to increase model complexity.

RNN-based models (Recurrent Neural Networks) such as Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU) are specially designed to address long-term dependency issues. The comparison in [8] showed that RNN-based models outperformed ARIMA by reducing errors by 10%. [9] also underlined their higher performance on traffic congestion prediction. However, RNN-based models sometimes encounter problems caused by vanishing or exploding gradients [10].

Recent approaches including Convolutional Neural Networks (CNN) and Graph Convolutional Networks (GCN) aim to fill the gap relating to spatial correlation modeling in previous models. In [11], the input traffic data were sent to CNN as images in which spatiotemporal traffic dynamics were converted to a two-dimensional time-space matrix. Work in [12] and [13], combining LSTM and CNN models, outperformed the original models since it benefits from both temporal and spatial dependency modelings. STGCN (spatiotemporal graph convolutional networks) [14] and DCRNN (Diffusion convolutional recurrent neural network) [15] showed the performance of GCN on graph structure data using spatial information without restricting models to only process on grid structure data (images, videos) as CNN. Some latter works aim at improving the performance of GCN by integrating the ATtention mechanism in ATGCN [16] for dynamic spatial-temporal correlations, adopting the mechanism of iterative prediction for Long-Short terms prediction in LSGCN [17] and including a continual learning mechanism for streaming traffic flow forecasting in [18].

C. Discussion

TABLE I: Comparison of traffic prediction models

Model	Modeling ability	Spatial consideration	Online learning	Model simplicity
ARIMA	Linearity	-	-	++
SARIMA	Linearity, seasonality	-	-	+
VARMA, STARIMA	Linearity, multi-variate	++	-	-
KNN	Non-linearity	+	-	++
FFNN	Non-linearity	+	-	-
RNN	Non-linearity, long-term dependency	-	-	–
CNN	Non-linearity, long-term dependency	++	-	–
GCN	Non-linearity, long-term dependency	+++	-	–

Table I highlights the characteristics of learning approaches to build and train models for traffic predictions regarding four criteria: *modeling ability*, *spatial consideration*, *online learning* and *model simplicity*. These criteria are chosen based on the need of balancing between high-level modeling ability for spatiotemporal relations and dynamic properties for large-scale

and real-time applications. Temporal dependencies covered by all the studied models are not considered in this table. Two points are underlined in this comparison. First, **learning phase of all models is offline**, so that it does not update the learned model when driving environments and traffic data evolve. Second, **the consideration of long-term temporal dependency and spatial correlation increases model complexity** that limits their usage on large-scale applications.

The first limit which we aim to address is the lack of continuous learning by using **dynamic clustering** for the learning process. Continuous learning requires a process capable of integrating arrival observations and continuously updating the model with restriction of memory and time. Dynamic clustering technique is proven to be efficient for dealing with infinite and evolved data thanks to flexible structural changes. These properties refer to the following behaviors: (1) create new clusters when detecting novel data behaviors, (2) adjust or update the centroid of clusters when aggregating new data, (3) merge clusters when they approach each other and (4) split dispersed clusters. The well-known methods include CluStream [19], DenStream [20], the family of topological structures based on Self-Organizing Maps (SOM) and Growing Neural Gas (GNG) introduced in [21], [22] and [23]. These models have been widely applied in network intrusion detection, financial transaction, phone recording, web click stream processing, $etc.$ To our knowledge, their application for traffic dynamics prediction is novel as in [24] that presents a system for Online Real-time Unsupervised Network Anomaly Detection Algorithm. This system outperformed DBSCAN and PCA (Principal Component Analysis) in detecting anomalies in traffic networks.

The second point to consider is **maintaining a reasonable complexity level of system**. For that, we consider decentralized and cooperative decisions of self-adaptive systems based on multi-agent system (MAS) approach for the prediction process. According to [25], MAS consists of multiple autonomous entities known as *agents* which can perceive information from their environment, make decisions according to their perceptions and knowledge, and act to cooperatively reach their local goals. MAS aims at decentralizing the function by distributing the control at the agent's level so that each agent owns only a partial view of the environment and a local decision-making algorithm to obtain its individual goals. To achieve the global objective of the system, agents need to cooperate with others. MAS-based systems allow us to integrate the self-adaptive mechanisms to deal with open and complex applications such as life-long supervised machine learning [26], traffic control [27],$etc.$ In the following, we present ADRIP - a self-adaptive multi-agent system that explores the dynamic clustering to self-adapt to dynamic changes in traffic and the self-correction mechanism to provide more accurate traffic prediction.

III. System description

This section formalizes the problem considered in this work, and defines the corresponding system architecture.

A. Problem description

Previous traffic prediction models presented in the state of the art are based on the road network topology as it enables to integrate spatial dependencies. For example, spatial dependencies can be evaluated by the correlations between neighboring sensors from observed data. The traffic prediction problem is then described as follows, given :

- a set of vehicles $V = v_1; v_2; ...; v_n$, each following an itinerary I segmented into a sequence of road segments noted $I = \{rds_1, \ldots, rds_d\}$ and communicating its *mobility profile* to each crossed road segment;
- a set of road segments determined according to the road network in Open Street Map (OSM), its starting and ending points located by GPS devices;

conceive a system able to learn and predict future traffic dynamics at the level of each road segment.

Previous studies define traffic dynamics prediction as the prediction of mean speed, volume, or density. These macroscopic traffic parameters cannot express the variation of vehicle information on a road segment over time. For example, the predicted mean speed at 30km/h can refer to both cases: a constant speed of a vehicle on a considered road segment and a homogeneous change of vehicle speeds from 0km/h to 60km/h. To fill this gap, we introduce **Mobility Profile (MP)** computed by each vehicle when crossing a road segment.

Definition 1. ***Mobility Profile (MP)***
The MP is defined as the distribution of travel time at different speed ranges on a given road segment.

Figure 1 illustrates the MP of a vehicle crossing a segment of 72m length with maximum speed of 30km/h using 7 speed ranges.

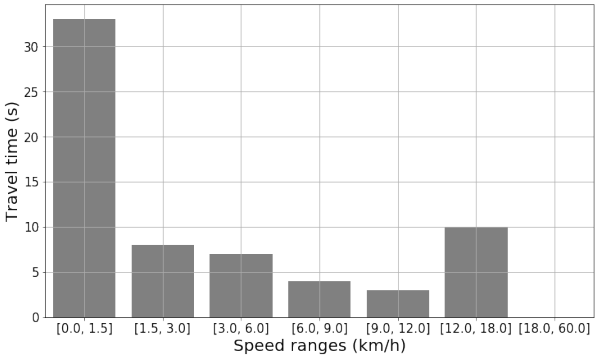

Fig. 1: Illustration of MP

Given a MP, perspicuous information as total travel time, mean speed, or speed variation, can be communicated to drivers. In addition, by its definition, a MP is enough succinct to adapt to memory and calculation time restrictions for continuous learning and real-time prediction.

B. System architecture

From the problem description and the AMAS approach (Adaptive Multi-Agent System), two types of agents have been

defined :

- **Vehicle Agents (VA)**: representing each a vehicle, they aim to share their MPs along the crossed road segments of their trajectory.
- **Segment Agents (SA)**: associated each with a road segment of the road network, their aim is to locally predict future traffic dynamics on their corresponding road segment. To do that, SA's behavior is divided into two processes: a learning process and a prediction process (cf.Fig 2).

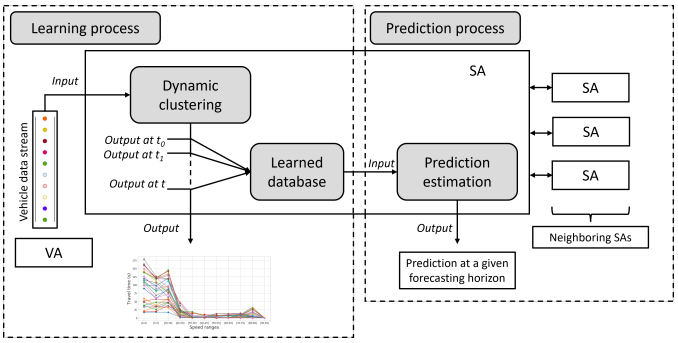

Fig. 2: System architecture

At each SA, the learning process continuously classifies the different MPs perceived from VAs into clusters representing different traffic dynamics and memorizes them with their associated timestamp. The output of the learning process is a database of traffic dynamics clusters, each described by a centroid MP and a list of **Ranges of Use (RUs)** indicating the moments when vehicles crossed that road segment with this MP. The clustering decisions and structure are updated at each received MP. That enables the system to locally self-adapt to traffic dynamics evolution. In this process, only the communications between VAs and corresponding SAs are established to exchange MPs. The learning process addresses the temporal dependencies by observing the transitions between clusters.

Simultaneously with the learning process, SAs perform a prediction process that provides traffic dynamics estimations until a given time horizon. To relate to the fact that traffic dynamics on a given segment are impacted by the traffic of neighboring segments, each SA cooperates with its neighboring SAs defined by the road network to compute the predictions. This cooperative interaction between SAs is the key of the prediction process since it assesses the propagation of traffic dynamics. Given that, the prediction process achieves the integration of spatial dependencies.

Finally, in order to provide accurate predictions, SA compares its predictions to true perceived traffic information and deploys its self-correction mechanism to improve its predictions.

IV. Segment Agent behavior

A. Learning process

All SAs perform the same dynamic clustering algorithm (cf. Algorithm 1) to cluster the perceived MPs on their associated road segment. Nonetheless, depending on the stream of perceived MPs, each SA possesses a different cluster structure and learned database. This distribution of clustering mechanism at the SA level enables several benefits. First, SAs function simultaneously and independently from each other to facilitate system control, mitigate the potential damage of a centralized processing node and reduce the calculation time by enabling parallel processing on multiple processors. Second, it enhances the openness and flexibility of the system since SAs can be easily added or removed when having some changes in the road network. Third, it self-adapts to the continuous update when traffic dynamics change on the road network. For example, when the correlations between road segments evolve, the system only needs to update the relationships between related road segments rather than recomputing for the entire road network.

Algorithm 1 Dynamic clustering algorithm of SA

```
percMP ⟵ getMPFromVA()
isSingular ⟵ NoiseDetection(percMP)
if isSingular == False then
    SimClus ⟵ similarClusters (LearnedClusters, percMP, α)
    if SimClus ≠ ∅ then
        if card(SimClus) ≥ 2 then
            MostSimClus ⟵ mergeClus(SimClus, α)
        else
            MostSimClus ⟵ SimClus[0]
        end if
        MostSimClus.centroid ⟵ adjustGiven(percMP)
        MostSimClus.RUs ⟵ updateGiven(percMP)
    else
        SA.createNewClus(percMP)
    end if
else
    Ignore perceived MPs
end if
```

Algorithm 1 details the learning process performed by a SA. After perceiving a MP ($percMP$) from a VA, SA verifies whether the new MP describes a singular vehicle behavior (*NoiseDetection()*). If it is not the case, SA searches for existing clusters similar to $percMP$ by assessing the similarity between $percMP$ and the centroids of learned clusters (*similarClusters()*). If more than one similar cluster is found, SA evaluates if they have to merge together (*mergeClus()*). Then, the most similar one is chosen to assign $percMP$ and is updated. Otherwise, a new cluster is created by SA for $percMP$. If a singular behavior is detected, SA ignores the new perceived MP.

*1) **Detection of singular behaviors**:* Using trajectory data of vehicles as representative of traffic dynamics requires consistency. Indeed, we can only deduce the dynamics of traffic from the time series of vehicle speeds when the behaviors of vehicles correctly reflect what happens on road segments (no-outlier existence assumption). This required consistency is

not always guaranteed, for example with emergency vehicles moving with particular priorities or vehicles moving with individual behaviors in free-flow traffic. In such cases, the learning process is disturbed since it may consider such behaviors as new clusters of traffic dynamics while these discrepancies are due to the diversity of individual vehicle behaviors.

Our noise detection method is constructed as a multi-criteria method to detect singular behaviors according to each established situation. In this paper, we explore the detection of singular behaviors in free-flow traffic. For that, we compare the distance between the position of the vehicle communicating $percMP$ and its leading vehicle to the stopping sight distance (SSD). SSD ([28], [29]) is defined as the minimum distance required on a roadway to enable a vehicle traveling at or near the design speed to stop before reaching a stationary object in its path. If the computed distance is greater than the SSD of the road segment, that vehicle may have singular behavior since it is not constrained by the security restrictions and the new $percMP$ is thus considered as noise. Other situations will be studied and added to further versions of ADRIP.

*2) **Searching for similar clusters**:* The similarity between a MP and a cluster is evaluated by the similarity between this MP and the centroid of the cluster.

Definition 2. ***MPs similarity***

The distance measure between two MPs, regarding the time by speed range, is defined as an array whose elements are the absolute differences in time travel of respective speed ranges of both MPs. Two MPs are similar to each other if each element of the obtained array is less than a threshold α (cf. equation 1).

$$MPDiff(MP^i, MP^l) = [|MP^i_j - MP^l_j|] \leq \alpha_j \quad \forall j \in \{1, \ldots, N\} \tag{1}$$

where N is the number of speed ranges, MP^i_j and MP^l_j are the values of time travel corresponding to the j^{th} speed range. Noting that both MPs must have the same list of speed ranges sorted in increasing order. α_j is a time threshold adapting to each speed range j. Indeed, the travel times with different speed intervals have different ranges ($eg.$ longer travel time for lower speed). Thus, using the adaptive similarity thresholds for different speed ranges has the same aim as data rescaling techniques such as data normalization or standardization.

3) Merging similar clusters: When more than one cluster is similar to the perceived MP, SA evaluates if they have to merge together. That allows the clustering structure to adapt to new arrival data without depending on initial data. *mergeClus()* method (cf. Algorithm 2) focuses on the two most similar clusters to the perceived MP. Thus, the merging of clusters is a local process that avoids costly calculations. If the difference between the two most similar clusters satisfies equation 1, they merge together. The new centroid is computed as the mean of their centroids and the list of RUs is the aggregation of their lists of RUs.

Algorithm 2 mergeClus(SimClus, α)

```
OrderedSimClus ← SimClus ordered increasingly by the distance to percMP
c1 ← OrderedSimClus[0]
c2 ← OrderedSimClus[1]
d ← MPDiff(c1.centroidMP, c2.centroidMP)
if d ≤ α then
    c1.centroidMP = (c1.centroidMP + c2.centroidMP)/2
    c1.listRUs.append(c2.listRUs)
end if
Update SA's database
return c1
```

4) Integrating perceived MP to learned database : At this step, if the set of similar clusters is empty, SA creates a new cluster with $percMP$ as the centroid. Otherwise, the information from $percMP$ is used to adjust the centroid of the most similar cluster using γ as an adjustment coefficient (cf. equation 2). In this algorithm, γ plays a similar role as *learning rate* in the optimization algorithms as in the gradient descent that makes the chosen cluster gradually move towards the new $percMP$.

$$MP^{sim}_{adjusted} = MP^{sim} + \gamma * \text{sign}(percMP - MP^{sim}) \tag{2}$$

B. Prediction process

The prediction process aims to compute the chain of the next changes of MPs for a required time horizon. Thus, the predicted data is denoted as: $\mathcal{P} = \{(predMP_1, Ts_1) \longrightarrow (predMP_2, Ts_2) \longrightarrow \cdots \longrightarrow (predMP_H, Ts_H)\}$. Ts_i is the predicted timestamp where the traffic on a given SA changes to $predMP_i$. $predMP_1, Ts_1$ is predicted using current traffic state. A prediction $(predMP_i, Ts_i)$ is computed using the previous prediction $predMP_{i-1}, Ts_{i-1}$.

To complete the criteria highlighted in table I, the prediction process addresses two points: spatial dependency and real-time update.

First, for including spatial dependency, SAs are based on the fact that the traffic dynamics on a given segment are impacted by its neighboring road segments (spatial correlation). Each SA cooperates with the neighboring SAs (SAs associated with direct upstream and downstream road segments as defined by OSM) to collect the historical information required for its predictions (cf Algorithm 3). Through these interactions, each SA computes the predictions based on how its traffic dynamics in the past have been influenced by its neighboring SAs. The information of considered SA and its neighbors constitute a **Configuration** of traffic.

Definition 3. ***Configuration***

The configuration at an instant T under the point of view of a SA is the set of observed MPs with their corresponding RU at T on itself and on its neighboring SAs.

Second, the real-time update in the prediction process is firstly guaranteed by the real-time extension of predicted data to ensure that they are always available up to the required

time horizon. When the current prediction horizon is shorter than the required one, SAs use the farthest prediction to compute the next one. This extension mechanism is continuous through time. However, as the following prediction is computed based on the previous one, the bad performance of the previous step can propagate and make the prediction accuracy degrade quickly. Thus, the real-time update also includes a self-correction mechanism using real-time observed data. This ability allows to enhance the availability and reliability of the prediction process.

Algorithm 3 Prediction algorithm performed by each SA for a horizon H

```
currentTs ⟵ Current timestamp of prediction process
Ts ⟵ currentTs
while Ts < currentTs + H do
    Config_Ts ⟵ buildConfig(SA, neighborSAs,Ts)
    listRUs ⟵ getRUs(SA.MP_Ts)
    histConfigs = []
    for RU ∈ listRUs do
        histConfigs.add(buildConfig(SA, neighborSAs, RU.start))
    end for
    mostSimConfig ⟵ evalutateConfigSim(Config_Ts, histConfigs)
    predMP, RU_predMP ⟵ getFollowingMP(mostSimConfig)
    Ts ⟵ Ts + RU_predMP
end while
```

1) Prediction algorithm: Algorithm 3 details this prediction process starting from the current timestamp with the current observed configuration, until the desired prediction horizon H. The main principle is, given a *configuration* constituted by the observed MP of SA and the observed MPs of its neighbors at a given timestamp, to figure out the most similar configuration in the past called *historical configuration*. The prediction of SA is then defined as the succeeding MP that was observed after this historical configuration. This prediction lasts as long as the selected MP lasts in the past (RU_predMP)

At each timestamp Ts of the prediction process, SA builds (buildConfig()) the configuration $Config_{Ts}$ from its MP and by asking the MPs from its neighbors. Then, SA extracts the RUs ($listRUs$) at Ts (getRUs()) and constructs, for each RU, the historical configuration containing the MPs of its neighbors at the beginning of the RU. SA then compares each historical configuration with $Config_{Ts}$ (*evalutateConfigSim()*) based on two ordered criteria:

1) the number of neighboring SAs with different MPs in both configurations to address the difference in traffic dynamics;
2) the time gap between the beginnings of RUs in both configurations to address the difference in dynamic propagation time.

The most similar historical configuration ($mostSimConfig$) is then defined as the historical configuration minimizing both criteria. Then, SA gets the next MP and its RUs of this configuration as the predictions for the next minutes from Ts. SA computes the achieved prediction horizon by forwardly shifting Ts for a time interval equal to the RUs of predicted MP. If the desired horizon H is not reached yet, all described steps are repeated.

2) Cooperation failures: During interactions between SAs, some specific situations can occur and disturb the functioning of SAs. That requires SAs to self-adapt. Thus, a set of cooperative behaviors allowing SAs to overcome those situations and maintain the adequate functioning of the system has been defined.

First, during the construction of traffic configuration, SA faces an issue where it does not receive responses from its neighbors for their MPs' demands. As a result, SA is unable to establish the complete configuration. This happens as some neighbors haven't estimated yet their prediction at the requested timestamp, preventing them from sending this information to demanding SA. In such cases, SAs with shorter prediction horizons launch their calculation first. The remaining SAs will wait until the required predictions are computed. Through this mechanism, SAs self-organize their execution based on different situations.

Second, in cases where the current MP on a SA has not been observed yet, SA does not have the historical information necessary for the prediction process. To address this lack of information, SA will use this MP as the prediction until the required prediction horizon. When SA detects a significant difference between the predicted MP and the observed MP, it activates the *self-correction mechanism* to correct the prediction. Noting that, in parallel, the learning process will cluster this new MP into the SA database allowing the prediction process to perform better in the future.

Third, when several historical configurations are similar to the configuration at time Ts, SA must select the most accurate one for its prediction. In such cases, the configurations are considered as equivalent, SA will opt for the most recent one. This choice is coherent with the objective of continuous learning, emphasizing the significance of recent changes in the driving environment.

3) Self-correction mechanism: The goal of the self-correction mechanism is to detect when the predicted MP is different from the observed MP and correct it. To do that, when a change of MP is detected, SA compares the perceived MP with the predicted MP. If the difference between them is larger than the similarity threshold (cf.Equation 1), SA relaunches the prediction process using its current configuration. This mechanism allows to reduce the degradation of prediction performance due to the dependence of following predicted steps on previous ones.

V. Experiments

A. Data generation

Traffic data applied in the experiment are generated by GAMA platform (GIS Agent-based Modeling Architecture) [30]. GAMA is a simulation platform widely used for building

explicit agent-based simulations, including traffic simulation with the illustration of many interactions between agents (vehicle, road, infrastructure, people, etc). GAMA allows to perform simulations using real road networks by importing external road networks from shape files or OSM files. For this evaluation, we chose as a scenario a road network consisting of 63 road segments (cf. figure 3) with various lengths including roundabouts and different intersection types. However, many road segments among them are divided by the crosswalks, making them very short and unremarkable for this study. After eliminating those segments, it remains 30 segments used for this comparison.

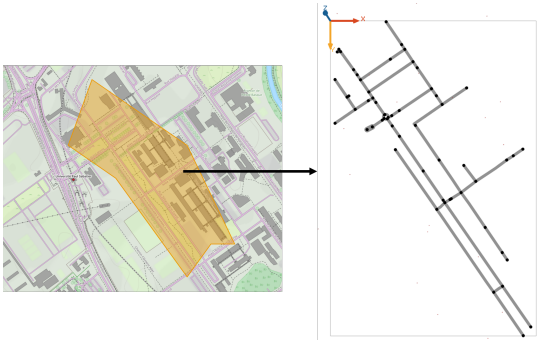

Fig. 3: The scenario from OSM (left) and the projection of chosen zone in GAMA

The behaviors of vehicles in this simulation follow Advanced Driving Skill [31] which is inspired by the Intelligent Driver Models and the MOBIL model for lane change. To get enough diversity in traffic dynamics, the number of vehicles at every instant of the simulation is inhomogeneously varied between 50 and 200 vehicles. The starting position of vehicles and the destination are randomly chosen. Then, GAMA computes all possible paths between these two points and picks the optimal one considered as the trajectory of the vehicle. We simulated the traffic and registered the obtained data for 3 hours, totalizing approximately 9300 vehicle trajectories. The generated data include the GPS position, speed, and distance to the closest leading vehicle at every second for every vehicle.

B. Evaluation metrics

To evaluate the prediction performance, we adopt 2 metrics (equation 3): MAE (Mean Absolute Errors) and RMSE (Root Mean Squared Error).

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y_i}|; \quad RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2} \tag{3}$$

where y_i is the i^{th} true value, $\hat{y_i}$ is the i^{th} predicted value and N is the number of data points.

C. Parameter setting

ADRIP has two parameters to be defined *a priori* which are the adjustment coefficient γ fixed at 0.05 and the similarity threshold vector α defined as 20% of the time required to cross the segment with the average speed of each speed range. We compare ADRIP with five models presented in the state of the art. Their implementations are found in Python modules with the parameters set as follows:

- ARIMA ($statmodel$): Order = (30,1,1), numbers of lags = 30.
- KNN ($sklearn$): k = 18
- FFNN ($keras.layers$): Hidden layers = 2, units = 256, learning rate = $1e^{-3}$, dropout rate = 0.1, decay rate = $1e^{-2}$, batch size = 256, optimizer: stochastic gradient descent algorithm.
- RNN (LSTM, GRU) ($keras.layers.recurrent$): Hidden cells with 64 units, dropout layer = 0.2.

D. Results and analysis

The conducted experiment compares our 2 versions of ADRIP: without and with the self-correction mechanism with ARIMA, KNN, FFNN, RNN models (LSTM, GRU) for **prediction of travel time on a road segment**

We train ARIMA, KNN, FFNN, RNN models, and ADRIP with the data set corresponding to the first 2 hours of simulation. Learned models are used to estimate future data for the next hour. All predicted travel time for the next hour will be compared with data from the simulation. Noting that these models do not update themselves during the testing phase while the learning process of ADRIP is continuously updated through time.

1) Time travel prediction: The inputs of compared models are the time series of travel times of all crossing vehicles on each road segment for every 10 seconds. Those models predict the next value by analyzing its dependence on the previous 5-minute observations. During the testing phase, the vectors of 5-minute previous observations are used to estimate the predictions for the next 5 minutes. Data rescaling transformations such as normalization, standardization, etc are not applied since it is impractical for continuous learning to set up the rescaling parameters due to the dynamic arrival and generation of data [32].

Concerning ADRIP that learns and predicts MPs, the travel time is the sum of all elements of the predicted MP as $T = \sum_{j=1}^{N} MP_j$. The prediction process computes in real-time the next MP changes until the next 5 minutes equivalent to 30 next points (a prediction is provided every 10 seconds to allow comparison with other models).

During experiments, we notice that some road segments have a high diversity of traffic dynamics as they are located at the main entries and exits of the considered area, thus many vehicles crossed them. As a result, travel time data on those segments exhibit high variance. Table 4 shows the boxplots and standard deviations of travel time data on studied road segments. Some of them show the values mostly constant as their value range showed by the boxplot is small and their standard deviation is insignificant. Meanwhile, others express high variation of data due to many outliers on the boxplots and big standard deviations.

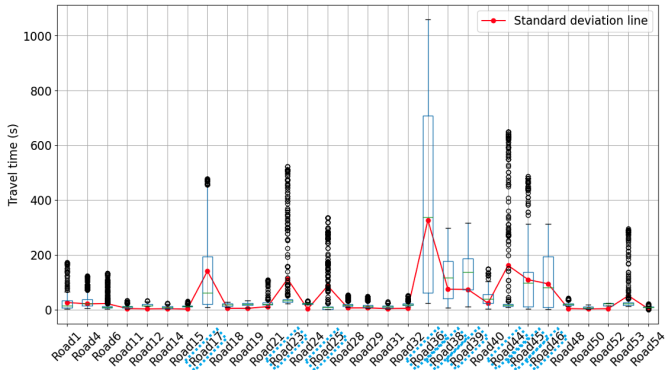

Fig. 4: Boxplots and standard deviation of travel time data. Roads framed by a dotted line express high variation of data.

Table II shows the comparison results on road segments with low variation of traffic data and highlights four key points. First, ARIMA, with its linear modeling limitation and strict data assumption, obtains significant prediction errors. Second, KNN, FFNN, LSTM, and GRU with their capacity of capturing long-term temporal dependencies and non-linear modeling have good performance in travel time prediction for low variation data. High accuracy achieved in KNN is due to the benefits of static clustering, which analyzes the data structure for clustering from a complete database thus having the better understanding to detect clusters. In contrast, dynamic clustering attempts to detect data structure from small samples and gradually update it. That leads to potentially inadequate clustering structures initially, requiring more data and time to improve. Results obtained by both versions of ADRIP are close to the compared models. In addition, ADRIP gains in explicability since we can determine which historical configuration brings this prediction and also the impacts of each neighboring road segment on the predictions. That helps to better analyze and understand the traffic evolution. We note that the self-correction mechanism does not enhance prediction accuracy. This is due to the fact that when dealing with low variation data, ADRIP does not frequently launch the self-correction mechanism since no MP change is detected. ADRIP can misunderstand that its predictions are good. Further tests and analysis are required to identify the situations where SAs must launch this mechanism.

TABLE II: Prediction errors in second of travel time prediction on road segments with low variations of traffic

Criteria	MAE	RMSE
ARIMA	21.52	26.75
KNN	8.75	14.80
FFNN	10.76	16.71
LSTM	9.05	14.90
GRU	9.11	15.18
ADRIP without self-correction	10.64	16.87
ADRIP with self-correction	11.82	18.08

In table III, the comparison results between ADRIP and other models on road segments with high variation of traffic data are presented (the road segments framed by a dotted line in figure 4). First, ARIMA remains obtaining the worst performance. Second, we remark that ADRIP without self-correction obtains worse accuracy compared to KNN, FFNN, LSTM, and GRU. That is resulted from the emergence of new behavioral patterns of the high dynamics of traffic. The functioning of the prediction process in SAs is often disturbed since many new MP are created and SAs do not have historical information about them. Thus, the prediction is quickly degraded since the following prediction step depends on the previous one. In such cases, ADRIP proposes the prediction as current MP and relies on the self-correction mechanism to correct them if errors are detected. Indeed, ADRIP with the self-correction mechanism outperforms most compared models in both criteria, except for a slightly higher MAE than FFNN. However, compared to FFNN, our proposed model achieves a lower RMSE. Since RMSE penalizes the large errors more significantly than MAE, we can deduce from this phenomenon that, ADRIP makes more small-scale errors but fewer large-scale errors than FFNN. In summary, the comparison results have proven the advantages of the self-correction mechanism and its importance when dealing with highly dynamic information.

TABLE III: Prediction errors in second of travel time prediction on road segments with high variations of traffic

Criteria	MAE	RMSE
ARIMA	149.29	190.68
KNN	91.49	122.02
FFNN	90.14	126.23
LSTM	110.24	145.07
GRU	108.06	141.71
ADRIP without self-correction	134.53	181.08
ADRIP with self-correction	91.07	120.10

VI. Conclusion and Perspective

In this paper, we introduced a traffic dynamics prediction model based on dynamic clustering and multi-agent systems. The proposed system consists of two processes: continuous learning with local dynamic clustering, and real-time prediction based on cooperative decisions.

ADRIP's architecture satisfies the requirement of traffic prediction problem: temporal dependencies are considered in the learning process while spatial correlations are handled by the prediction process. Our system can fill the gap in studied methods for continuous learning thanks to dynamic clustering and MAS. From our knowledge, dynamic clustering for traffic prediction is novel. This paper demonstrated its adequacy to address the considered problem due to its flexible structure adapting to new arrival data and reasonable complexity in responding to time and memory restrictions for online updates. Our system can leverage the advantages and solve the limitations of vehicle trajectory data. Indeed, the rich information from onboard sensors allows us to provide the prediction

of ***Mobility Profile*** which is more representative for traffic dynamics than macroscopic parameters. Besides, the vehicle trajectory data stream is non-stationary, i.e. its distribution may change over time and arrive continuously and potentially unboundedly through time. Continuous learning is necessary since storing all the arrival data is not feasible. Our system also addresses the detection of singular vehicle behaviors improving the prediction quality. On the other hand, the prediction process integrates several real-time update mechanisms including self-organization, self-adaptation, and self-correction allowing to enhance the availability and reliability of the obtained results.

The conducted experiment on generated data by GAMA has shown promising results. ADRIP with self-correction mechanism outperforms other compared models in both MAE and RMSE criteria.

For further works, many possibilities are interesting to explore. First, the detection of singular behaviors of vehicle data will be investigated with different cases. For example, an online anomaly detection mechanism for traffic network data presented in [24] is interesting to explore. Second, the comparison criteria between configurations can also consider the seasonality of traffic, the fading of outdated data, etc to filter the redundant configurations and improve cooperation. Furthermore, a case test on real-world data will be conducted.

REFERENCES

[1] A. M. Nagy and V. Simon, "Survey on traffic prediction in smart cities," *Pervasive and Mobile Computing*, vol. 50, pp. 148–163, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574119217306521

[2] D. Billings and J.-S. Yang, "Application of the arima models to urban roadway travel time prediction - a case study," in *2006 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, 2006, pp. 2529–2534.

[3] Y. Kamarianakis and P. Prastacos, "Forecasting traffic flow conditions in an urban network: Comparison of multivariate and univariate approaches," *Transportation Research Record*, vol. 1857, no. 1, pp. 74–84, 2003.

[4] S. Cheng, F. Lu, P. Peng, and S. Wu, "Short-term traffic forecasting: An adaptive st-knn model that considers spatial heterogeneity," *Computers, Environment and Urban Systems*, vol. 71, pp. 186–198, 2018.

[5] H. Yu, N. Ji, Y. Ren, and C. Yang, "A special event-based k-nearest neighbor model for short-term traffic state prediction," *IEEE Access*, vol. 7, pp. 81 717–81 729, 2019.

[6] Y. Gao, S. Sun, and D. Shi, "Network-scale traffic modeling and forecasting with graphical lasso," in *Advances in Neural Networks – ISNN 2011*, D. Liu, H. Zhang, M. Polycarpou, C. Alippi, and H. He, Eds. Springer Berlin Heidelberg, 2011, pp. 151–158.

[7] N. G. Polson and V. O. Sokolov, "Deep learning for short-term traffic flow prediction," *Transportation Research Part C: Emerging Technologies*, vol. 79, pp. 1–17, 2017.

[8] R. Fu, Z. Zhang, and L. Li, "Using lstm and gru neural network methods for traffic flow prediction," in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2016, pp. 324–328.

[9] S. Sun, J. Chen, and J. Sun, "Traffic congestion prediction based on gps trajectory data," *International Journal of Distributed Sensor Networks*, vol. 15, no. 5, p. 1550147719847440, 2019.

[10] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, p. III–1310–III–1318.

[11] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, 2017.

[12] T. Bogaerts, A. D. Masegosa, J. S. Angarita-Zapata, E. Onieva, and P. Hellinckx, "A graph cnn-lstm neural network for short and long-term traffic forecasting based on trajectory data," *Transportation Research Part C: Emerging Technologies*, vol. 112, pp. 62–77, 2020.

[13] Z. Duan, Y. Yang, K. Zhang, Y. Ni, and S. Bajgain, "Improved deep hybrid networks for urban traffic flow prediction using trajectory data," *IEEE Access*, vol. 6, pp. 31 820–31 827, 2018.

[14] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, ser. IJCAI'18, 2018, p. 3634–3640.

[15] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations*, 2018.

[16] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 922–929, Jul. 2019.

[17] R. Huang, C. Huang, Y. Liu, G. Dai, and W. Kong, "Lsgcn: Long short-term traffic prediction with graph convolutional networks," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 2355–2361, main track. [Online]. Available: https://doi.org/10.24963/ijcai.2020/326

[18] X. Chen, J. Wang, and K. Xie, "Trafficstream: A streaming traffic flow forecasting framework based on graph neural networks and continual learning," in *International Joint Conference on Artificial Intelligence*, 2021.

[19] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*. VLDB Endowment, 2003, p. 81–92.

[20] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proceedings of the 2006 SIAM international conference on data mining*. SIAM, 2006, pp. 328–339.

[21] M.-R. Bouguelia, Y. Belaïd, and A. Belaïd, "An adaptive incremental clustering method based on the growing neural gas algorithm," in *ICPRAM*, 2013.

[22] B. Fritzke, "A growing neural gas network learns topologies," in *Proceedings of the 7th International Conference on Neural Information Processing Systems*. MIT Press, 1994, p. 625–632.

[23] M. Ghesmoune, M. Lebbah, and H. Azzag, "Clustering over data streams based on growing neural gas," in *Advances in Knowledge Discovery and Data Mining*, T. Cao, E.-P. Lim, Z.-H. Zhou, T.-B. Ho, D. Cheung, and H. Motoda, Eds. Cham: Springer International Publishing, 2015, pp. 134–145.

[24] J. Dromard and P. Owezarski, "Integrating short history for improving clustering based network traffic anomaly detection," in *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2017, pp. 227–234.

[25] J. Ferber and G. Weiss, *Multi-agent systems: an introduction to distributed artificial intelligence*. Addison-wesley Reading, 1999, vol. 1.

[26] N. Verstaevel, J. Boes, J. Nigon, D. d'Amico, and M.-P. Gleizes, "Lifelong machine learning with adaptive multi-agent systems," in *9th International Conference on Agents and Artificial Intelligence (ICAART 2017)*, vol. 2, 2017, pp. pp–275.

[27] M. Guériau, F. Armetta, S. Hassas, R. Billot, and N.-E. El Faouzi, "A constructivist approach for a self-adaptive decision-making system: Application to road traffic control," in *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2016, pp. 670–677.

[28] D. B. Fambro, K. Fitzpatrick, and R. J. Koppa, *Determination of stopping sight distances*. Transportation Research Board, 1997, vol. 400.

[29] R. Layton and K. Dixon, "Stopping sight distance," *Kiewit Center for Infrastructure and Transportation, Oregon Department of Transportation*, 2012.

[30] A. Grignard, P. Taillandier, B. Gaudou, D. A. Vo, N. Q. Huynh, and A. Drogoul, "Gama 1.6: Advancing the art of complex agent-based modeling and simulation," in *PRIMA 2013: Principles and Practice of Multi-Agent Systems*, G. Boella, E. Elkind, B. T. R. Savarimuthu, F. Dignum, and M. K. Purvis, Eds. Springer Berlin Heidelberg, 2013.

[31] P. Taillandier, "Traffic simulation with the gama platform," in *Eighth International Workshop on Agents in Traffic and Transportation*, 2014, pp. 8–p.

[32] S. Ross, P. Mineiro, and J. Langford, "Normalized online learning," in *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, 2013, pp. 537–545.