Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Smart vehicular traffic management: An edge cloud centric IoT based framework

Sahil[a,*], Sandeep Kumar Sood[b]

[a] Department of Computer Science and Engineering, Guru Nanak Dev University Regional Campus, Gurdaspur, India
[b] Department of Computer Science and Informatics, Central University of Himachal Pradesh, Dharamshala, India

A R T I C L E I N F O

Article history:
Received 13 February 2019
Revised 6 November 2019
Accepted 14 November 2019
Available online 20 November 2019

Keywords:
Urbanization
Big data
Advanced driver assistive technologies (ADAT)
Internet of things (IoT)
Cloud computing
Edge computing
V2I (Vehicle to infrastructure) communication
Vehicular ad hoc network (VANET)

A B S T R A C T

Urbanization is imposing many challenges, and vehicular traffic management is one such challenge. It hampers smooth traffic flow, wastes time, and threats of road safety. Moreover, it also impacts the environment, economy, health, and other essential services. The root cause of traffic mismanagement is the dynamic nature of the traffic on roads and the incapability of legacy systems to interpret such dynamics in real-time. The data analysis services based on Edge Cloud frameworks in technology outfitted urban spaces provide real-time robust and smart solutions to ever facing challenges from urbanization. Hence, Edge Cloud-based traffic management can be used to manage urban vehicular traffic in real-time. In this paper, Edge Cloud-centric IoT based smart traffic management system is developed for traffic inflow prediction and time optimized smart navigation of the vehicles. The traffic inflow prediction adapts the traffic movement phase time accordingly and avoids long waiting queues and congestions at intersections. The smart navigation enables the optimal distribution of traffic to possible paths and subsequently improves road safety at intersections. Baseline classifiers are used to predict the traffic inflow, and the statistical analyses acknowledge the prediction efficiency of the J48 decision tree as compared to other utilized classifiers. Edge Computing is used for time optimized smart navigation of vehicles and, subsequently, optimal traffic load balancing in real-time. The road safety perspective of vehicles at intersections is also get benefited from the optimal traffic load balancing. The results depict the efficiency of the proposed system for smart navigation, optimal traffic load balancing, and improved road safety at intersections.

1. Introduction

The urbanization is changing the demographic structure of the world. More than 50% of the total human population on earth is now living in urban areas. This figure is projecting to rise to 67% in developing countries and to 86% in the developed countries by 2050 [1]. The urbanization has impacted all phases of life from its people to the environment in many ways. On the brighter side, it has improved the mode of transportation, health services, education, and other facilities. But on its darker side, it has presented new challenges like vehicular traffic mismanagement, garbage mismanagement, water pollution, air pollution, and increased energy consumption.

* Corresponding author.
E-mail addresses: sahil.neelam@hotmail.com (Sahil), san1198@gmail.com (S.K. Sood).

https://doi.org/10.1016/j.iot.2019.100140
2542-6605/© 2019 Elsevier B.V. All rights reserved.

Since many walks of life are highly dependent on the mode of road transportation, traffic mismanagement causes a negative impact on the economy, health, and other essential services. This scenario is even worse in urban areas, where people are more reliant on road transportation for most of the activities. A report on urban mobility [2] by Victoria Transport Policy Institute consists of a study on the money and time wasted due to traffic mismanagement, which accounts for a 2.9 billion gallon fuel loss and 5.5 billion hours of time delay in traffic jams during the period of 2000 to 2010.

The recent developments in advanced driver assistive technologies (ADAT) have led to improved road safety, optimized navigation, real-time data exchange, and vehicle responsiveness. However, traffic management doesn't get many benefits from the advancements in ADAT. Consequences of which, various traffic errors, heavy traffic jams, and congestions happen and plague the transportation system, and road safety around the world [3]. This paper addresses the above-stated issues by proposing an Edge Cloud-centric IoT based smart traffic management system. The proposed system is designed to provide a ubiquitous, real-time, and area-wide solution to traffic management using adaptive traffic phase time planning, time optimized smart navigation and traffic load balancing for optimized traffic flow, and improved road safety at intersections in urban spaces.

The advancement in wireless communications, internet technology, and miniaturization of sensor hardware has enabled the information processing a pervasive process. One such category of applications is the intelligent transport system (ITS), where vehicles carry sensing power, storage capabilities, onboard computing facilities, and communication systems. The idea behind ITS was to solve the issues raised by the legacy systems efficiently. Vehicle ad hoc Network (VANET) is the core component of ITS. It [4,5] enables the vehicles in a given proximity to communicate with each other as well as with the roadside units (RSU). VANET has the capacity to address many traffic management related challenges [6,7].

The immense potential of Big Data Analytics, Cloud Computing, and the Internet of Things (IoT) is capable of providing robust and smart solutions to ever facing challenges from urbanization. It became possible only because of technology outfitted urban spaces, where urban spaces are now equipped with various sensing capabilities and large computing infrastructures. They are now capable of producing and processing huge volume and variety of big data [8,9] like traffic data, human mobility data, geographical data, etc. One such field of interest is transportation, where the increasing use of vehicles and the use of smart vehicular technology in vehicles are generating a huge amount of data in continuous and on a real-time basis. Such big data requires the power of Big Data Analytics for yielding useful insights and using the same for smart traffic management in real-time. The Internet of things (IoT) [10] and cloud computing [11] are potentially providing sensing and computing infrastructure to acquire and process such big data.

Traffic movement phase design, navigation, and load balancing are considered as the three prime foundation pillars for a traffic management system. Adaptive traffic phase time planning based on the traffic inflow can accommodate the traffic in a way to avoid long waiting queues at intersections. Smart navigation based on the real-time traffic scenario can guide a vehicle to drive into an optimized path and subsequently can optimally distribute traffic to all possible paths. Such real-time situation-aware features of a smart traffic management system can help in optimized traffic flow and consequently improved road safety at intersections. This can be only possible if the traffic management system can interpret various traffic dynamics and can build an area-wide traffic scenario in real-time. The computer-assisted technologies like the internet of things, cloud computing and big data analytics provide traffic management system such smart dimensions to interpret traffic dynamics and synthesize appropriate solutions to manage traffic on roads efficiently.

The proposed system consists of three subsystems: user, cloud, and action. In the user subsystem, the internet of things (IoT) helps in acquiring vehicle mobility data. In the cloud subsystem, cloud computing provides the storage and computing power to handle such big data. The action subsystem provides the synthesized analytics to the various stakeholders at the user subsystem. For time-sensitive and less latency demanding services in smart traffic management, a layer of edge computing [12] called reverse edge layer (REL) is used between the user subsystem and the action subsystem. It avoids the communication overhead to process data in cloud. It analyzes the time sensitive data at the edge of the data origin. Edge computing provides quality of services assurance, low latency and location awareness with localized and immediate computing to time sensitive services.

The ubiquitous and real-time sensing of mobility data is highly significant for smart traffic management. Hence, the data acquisition utilizes onboard sensors, OBU (onboard unit), infrastructure-based sensors, and V2I (vehicle to infrastructure) communication model with the help of VANET [13,14]. The adaptive traffic phase time planning analyzes the mobility data at the cloud subsystem to prepare adaptive traffic phase time plans. The smart navigation uses the real-time traffic scenario and road network geometry on the reverse edge layer (REL) to choose time optimized path in real-time and, subsequently, optimally balance the traffic load to all possible paths.

Table 1 presents the comparative analysis of smart traffic management and other related works based on various classifications like the technologies it employed: cloud computing (CC), internet of things (IoT), and fog or edge computing (FC/EC). The study also reviewed the cited works' approaches based on other parameters like real-time aspect (RTA) of the approach, service provisioning (SP) to people, area-wide analysis (AWA), and type of traffic estimation (ToTE) [15]. The analysis contrasts various works based on values: Y means yes (the work includes that parameter or technology); N means no (the work doesn't include that parameter or technology); LW means lane-wise (this value signifies that the traffic analysis is based on lane-wise unlike of area-wise); IFB means infrastructure based traffic estimation (where, the traffic is analyzed on infrastructure devices like RSUs or centralized processing units at intersections); IFF means infrastructure free traffic estimation (where traffic parameters are sensed and relayed to some kind of server processing like in cloud computing), and HYB means the traffic estimation employs both IFB and IFF approaches.

Table 1
Comparative analysis of smart traffic management related works .

Author(s)	Description	CC	IoT	FC/EC	RTA	SP	AWA	ToTE
Sood et al. [16]	Social collaboration based Cloud-IoT architecture is proposed to monitor flood related Big Data and predict the state of flood for a particular region.	Y	Y	FC	Y	Y	Y	N
Feng et al. [13]	Mobility data from vehicles are analyzed at RSUs and send to signal controlling system to adapt green signal accordingly.	N	Y	N	Y	N	N	IFB
Whaiduzzaman et al. [14]	Component study of vehicular cloud computing and its applications is presented.	Y	Y	N	Y	N	Y	IFF
Bedogni et al. [17]	A framework for battery consumption analysis of electric vehicles and pre-reservation of slots at charging stations, is proposed.	Y	Y	N	Y	Y	Y	IFF
Abadi et al. [18]	Traffic flow is predicted using vehicle mobility data.	Y	Y	N	Y	N	Y	IFF
Pérez et al. [19]	IoT acquired mobility data is analyzed at different level of computing: fog and cloud, for traffic flow prediction.	Y	Y	FC	Y	N	Y	IFF
Chang and Park [20]	A model is designed for vehicles in a group, to directly exchange traffic flow data through a leader vehicle with the signal controlling system.	N	Y	N	Y	N	N	IFB
Wan et al. [21]	A computation load balancing approach is designed for edge nodes to facilitate optimized edge computing without any performance degradation in Internet of Vehicles using 5G networks	Y	Y	EC	Y	N	N	IFB
Collotta et al. [22]	WSN is used to acquire and transmit the data from vehicles to the isolated central signal controlling unit for traffic queue assessment and calculation of green phase time span for signals.	N	N	N	Y	N	N	IFB
Wan et al. [23]	A speed advisory system is proposed to guide vehicles for reaching at intersection only during green phase, using the traffic phase time sequences.	N	N	N	Y	Y	N	IFB
HomChaudhuri et al. [24]	A vehicle-based localized computing framework is designed to regulate the speed of the vehicle for avoiding the waiting queue at the intersection. It increases the fuel economy by utilizing traffic phase information and communicates the same to the trailing vehicles recursively and manages traffic.	N	N	N	Y	Y	LW	IFB
Zhang et al. [25]	Deep neural network based learning model is proposed to classify the speed and transportation mode of moving objects.	Y	Y	N	N	Y	Y	IFF
Liu et al. [26]	A computation offloading model using reverse auction based on endowment effect, is proposed to balance the computation load of the edge nodes in Internet of Things based vehicular network.	Y	Y	EC	Y	N	N	IFB
Proposed System	Cloud-based traffic inflow prediction and edge-based traffic navigation and load balancing are proposed. It manages traffic in real-time and improves road safety at intersections.	Y	Y	EC	Y	Y	Y	HYB

The real-time monitoring and management of vehicular traffic for optimized traffic flow and road safety at intersections is not feasible in the legacy systems. Hence, the proposed system is focused on providing smart solutions in traffic management with the following contributions.

- To develop a smart traffic management system using Edge Cloud-centric IoT for providing area-wide traffic management solutions in real-time.
- To efficiently predict and adapt the time duration of the green phase of traffic signal point based on the traffic inflow.
- To provide time optimized navigation service based on the area-wide traffic scenario.
- To optimize traffic load balancing on roads.
- To optimize traffic flow and road safety at intersections.

This paper is divided into four sections. In Section 2, the proposed methodology for smart traffic management is explained. In Section 3, the experimental setup and evaluation, along with the result analyses of the proposed system are discussed. The Section 4 concludes this paper.

2. Proposed system

The proposed conceptual model is shown in the Fig. 1. The entire system is based on the urban area, where the road network consists of four-leg intersections, and each road is having two lanes: left lane for upstream traffic and right lane for downstream traffic. Each lane is equipped with RSUs, loop detectors, and having a traffic signal point down the lane at the intersection. At every intersection, a Signal Controlling System (SCS) is present, which adapts the time duration of different

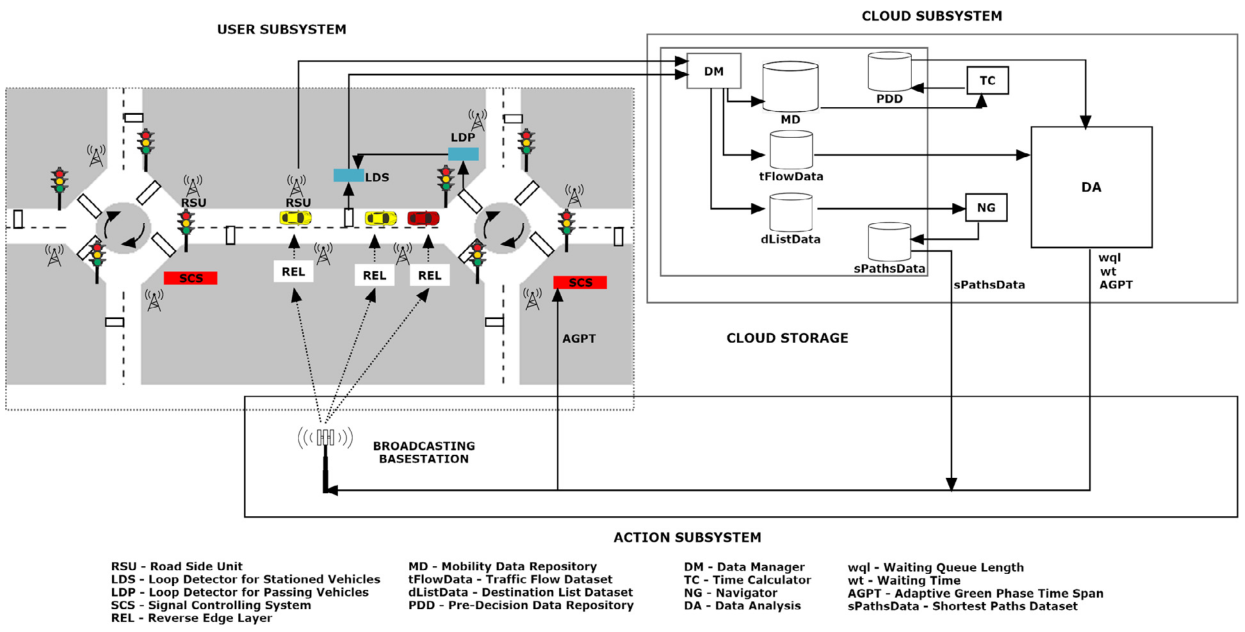
USER SUBSYSTEM
CLOUD SUBSYSTEM
DM
MD
PDD
TC
tFlowData
dListData
NG
sPathsData
DA
wql
wt
AGPT
sPathsData
CLOUD STORAGE
RSU
RSU
LDS
LDP
SCS
SCS
REL
REL
REL
AGPT
BROADCASTING BASESTATION
ACTION SUBSYSTEM
RSU - Road Side Unit
LDS - Loop Detector for Stationed Vehicles
LDP - Loop Detector for Passing Vehicles
SCS - Signal Controlling System
REL - Reverse Edge Layer
MD - Mobility Data Repository
tFlowData - Traffic Flow Dataset
dListData - Destination List Dataset
PDD - Pre-Decision Data Repository
DM - Data Manager
TC - Time Calculator
NG - Navigator
DA - Data Analysis
wql - Waiting Queue Length
wt - Waiting Time
AGPT - Adaptive Green Phase Time Span
sPathsData - Shortest Paths Dataset

Fig. 1. Edge Cloud centric IoT based smart traffic management conceptual model.

phases of traffic movement at various traffic signal points at that intersection. The phase time adaptation is based on the respective predicted green phase time span received from the cloud.

The proposed system is divided into three subsystems: user, cloud, and action. In the user subsystem, the real-time mobility data from vehicles is acquired by the RSUs, and the vehicle count is acquired by the loop detectors. The acquired data is relayed to the cloud. In the cloud subsystem, the data manager (DM) component of cloud storage classifies the acquired data from RSUs and loop detectors in different datasets based on their sources. The time calculator (TC) component of the cloud calculates time to reach (TTR) of every vehicle to the signal point down the lane. The navigator (NG) component of the cloud calculates the set of shortest paths to destinations of the vehicles. The data analysis (DA) component of the cloud subsystem analyzes the respective datasets to predict the green phase time span by predicting waiting queue length and waiting time at respective signal points. In the action subsystem, the predicted green phase time, waiting time and waiting queue length at signal points, and calculated shortest paths dataset are broadcasted to the entire set of vehicles in that urban area using cellular network-based navigation system. The predicted green phase time span is also separately sent to the respective SCS, which adapts the green phase at the respective signal point accordingly. For real-time time optimized path identification, the reverse edge layer (REL) is present between the user and the action subsystem. It uses the real-time traffic scenario (waiting queue lengths, waiting times and green phase time span at various signal points), and shortest paths dataset, to identify the time optimized shortest path to the destination, at the origin of the data by avoiding communication overhead to process data in the cloud. The reverse edge layer enables the time optimized smart navigation of the vehicles and, subsequently, optimal distribution of traffic load to all possible paths. The proposed system helps in optimized traffic flow and, subsequently, improved road safety at the intersections. Each subsystem of the proposed system is explained in detail as follows.

2.1. User subsystem

In the user subsystem, RSUs acquire the mobility data: vehicle id, velocity, acceleration/deceleration, vehicle location, and the destination location of the journey. To avoid any unauthorized access to the vehicle's location, the location is shared with due privacy [27]. Loop detectors for stationed vehicles (LDS) acquire data regarding the number of vehicles that passed through a particular cross-section or point of the road and are stationed at the signal point. The RSUs are available alongside the roads at a certain distance apart, hence acquire every possible change in the mobility of the vehicle in real-time. LDSs are present before the signal point and sense the data purposed for how many vehicles are driven into the signal point and are waiting for the green signal to move further. The acquired data from vehicles are relayed to the cloud. The OBUs of the vehicles continuously calculate the acceleration/deceleration based on the initial and current velocity of the vehicle, and the time is taken from the last RSU to the current location (refer Eq. (1)). The vehicle communicates the updated acceleration/deceleration value when an RSU is approached. The last communicated velocity becomes the initial velocity, and current velocity becomes the final velocity. The acquired data through sensors are relayed to the cloud.

$$a = \frac{v - u}{T} \tag{1}$$

Table 2
Datasets maintained at cloud storage .

S.No.	Dataset	Attributes	Sensors & Componenets used
1.	Mobility (mData)	Vehicle id, Velocity, Acceleration/Deceleration, Location of Vehicle	Onboard sensors, OBUs, GPS sensors, RSUs
2.	Destination List (dListData)	Destination Location	OBUs, RSUs
3.	Traffic Flow (tFlowData)	Lane id, Vehicle Count	LDSs
4.	Pre-decision (pDecData)	Vehicle id, Time to reach	TC
5.	Shortest Paths (sPathsData)	Destination Location, Set of weighted shortest paths	NG

where a is the acceleration/deceleration of the vehicle. v is the final velocity or current velocity of the vehicle. u is the initial velocity or the last communicated velocity of the vehicle. (in case of the start of the journey, $u = 0$). T is the time taken to cover the distance from last encountered RSU to the current location (in case of start of the journey, T is calculated from the beginning of the journey to current location).

2.2. Cloud subsystem

The data from various RSUs and LDSs arrive at the cloud. The data from RSUs contains: Source Identifier (scrId), Lane Id (lanId), Vehicle Id (vId), Velocity (v), Acceleration/Deceleration (a), Vehicle Location (vLoc) and Destination location (desLoc)). Source Identifier depicts the source of data i.e., whether its from RSU or LDS. Lane Id is the unique identification number for a lane, and the same lane id is given to all RSUs of that lane. It is used to classify the mobility data of vehicles lane-wise in cloud storage. The vehicle id helps the cloud to identify a vehicle uniquely on a lane. Velocity determines the velocity sensed by the onboard sensors of the vehicle. The rate of change in velocity between two consecutive RSUs or between two points is communicated in the form of acceleration/deceleration. The GPS device embedded in the vehicle communicates the location of the vehicle. The location of the vehicle is used by the cloud in the form of the distance of the vehicle from the signal point down the lane by calculating the relative distance between the location of vehicle and location of traffic signal point. The destination location depicts the destination of the journey. This parameter is also required to be monitored in real-time to calculate all possible shortest paths to the destination.

The data from LDSs contains Source Identifier (scrId), Lane Id (lanId), and vehicle count (vCount). Lane Id is the unique identification number of the lane and is given to the LDS of that lane. So, the records in the traffic flow data dataset can be maintained lane wise. Vehicle Count depicts the number of vehicles that have been crossed a particular cross-section or point of the road and are now waiting for the green signal to move further.

2.2.1. Cloud storage

The data relays to the cloud is stored in cloud storage. The data arrives to cloud is not in a well-classified form. It is the responsibility of the data manager (DM) component of cloud storage to classify various datasets (refer Table 2) based on their sources (refer Algorithm 1). The Time Calculator (TC) and Navigator (NG) of components of the cloud use the datasets classified by DM, to create new datasets in cloud storage. The data sets managed in cloud storage are explained as follows.

1. **Mobility datasets (mData**) DM classifies the data from RSUs in various datasets based on their Lane Id (lanId). These mobility datasets (mData) of different lanes are stored under repository mobility data (MD). The mData includes the vehicle id (vId), velocity (v), acceleration/deceleration (a), location of the vehicle (vLoc) and distance (d) of the vehicle from the intersection signal point down the lane. The velocity of the vehicle is captured by onboard sensors, whereas the acceleration/deceleration of the vehicle is processed by OBU. The distance is derived from the location of the vehicle (vLoc) and the location of the signal point down the lane. The location of the vehicle is communicated by the GPS embedded in the vehicle, whereas the location of the signal point is determined using the respective Lane Id (lanId). The mobility datasets are kept on updating with every new packet coming from RSUs.
2. **Destination List dataset (dListData)** DM classifies a dataset comprises of all distinct destination locations (desLoc) to which the vehicles are heading. The driver of the vehicle feeds the destination in the OBU of the vehicle at the start and during the journey (if requires). This destination location is acquired by the RSUs along with the mobility data. The destination list dataset (dListData) is kept on updating with every new packet coming from RSUs.
3. **Traffic flow dataset (tFlowData)** DM classifies the data from various LDSs in a dataset called as traffic flow dataset (tFlowData). It includes the lane id (lanId) and respective vehicle count (vCount). The number of vehicles stationed at a signal point is captured by the loop detector for stationed vehicles (LDS). The traffic flow dataset (tFlowData) is kept on updating after every interval of time.
4. **Pre-decision datasets (pDecData)**
   The Time Calculator (TC) component of the cloud calculates time to reach (TTR) of every vehicle to the respective signal point down the lane using the corresponding mData (refer Table 3). The newly calculated lane-wise pre-decision datasets (pDecData) are stored under different repository pre-decision data (PDD). The TTR is calculated as follows.
5. **Shortest paths dataset (sPathsData)**
   The navigator (NG) component of the cloud subsystem uses the dListData and online map service [28] to calculate all shortest paths (having shortest distance) to the individual destination in the dListData. The set of all shortest paths

Algorithm 1: Data Classification by Data Manager component.

```
Input  : Set of Values (Data from RSUs and LDSs)
Output: Sets of Records (Mobility Datasets(mData), Destination List Dataset (dListData)and Traffic Flow
        Dataset(tFlowData))
// Determine the source of data and if source is from RSU
if scrId==RSU then
    // For classifying mobility data
    // Search the respective mobility dataset of lane with identity lanId=i in Mobility Data repository
    search($mData_i$, MD)
    // If required mobility dataset set is found
    if $mData_i$ found then
        // Search the record of vehicle with identity vId in found mobility dataset
        search(vId, $mData_i$)
        // If record of vehicle with vId is found
        if Record found then
            // Update the velocity, acceleration/deceletation and location of vehicle in found record
            update(v, a, vLoc)
        end
        // If record is not found, create a new record with vId, v, a, vLoc and append it to found mobility
           dataset
        else
            INSERT INTO $mData_i$(vId, v, a, vLoc)
        end
    end
    // If searched mobility dataset set is not found, create new mobility dataset of lane with identity
       lanId=i in Mobility Data repository and create a new record with values
    else
        CREATE TABLE $mData_i$ in MD
        INSERT INTO $mData_i$(vId, v, a, vLoc)
    end
    // For classifying destination list dataset
    // Search destination location record in destination list dataset
    search(desLoc, dListData)
    // If searched record is not not found, create a new record with desLoc value, otherwise ignore it
    if Record not found then
        INSERT INTO dListData(desLoc)
    end
end
// For classifying traffic flow dataset
// if data source is from LDS, search a record with lanId in traffic flow dataset
else
    if scrId==LDS then
        search(lanId, tFlowData)
        // If searched record is found, update the corresponding vehicle count
        if Record Found then
            update(vCount)
        end
        // Otherwise create a new record with lane id and respective vehicle count and append into tFlowData
           dataset
        else
            INSERT INTO tFlowData(laneId, vCount)
        end
    end
end
```

Table 3
Time To Reach (TTR) determination in proposed methodology .

TTR derivation using distance (d), acceleration/deceleration (a) and velocity (v)
We Know,the acceleration/deceleration (a) is the rate of change of velocity (v) w.r.t time (t) as, $$a=\frac{dv}{dt}$$ We also know, the intergral of acceleration/deceleration (a) over time (t) is the integral of velocity between the limits u (initial velocity) and v (final velocity) as, $$\int_u^v dv = a\int dt$$ $$\Big\vert v \Big\vert_u^v = at + c$$ $$v - u = at$$ $$u = v - at$$ The basic formula for distance over velocity and time is, $$\text{distance} = \text{velocity} * \text{time}$$ Because of variable nature of velocity, the velocity can be taken as the mean of velocities as, $$d = \frac{v+u}{2} * t$$ By using value of u in above equation as, $$d = \frac{v+(v-at)}{2} * t$$ $$d = \frac{2v-at}{2} * t$$ $$t = \frac{2d}{2v-at}$$ $$2vt - at^2 = 2d$$ $$at^2 - 2vt + 2d = 0$$ $$t = \frac{2v \pm \sqrt{4v^2-8ad}}{2a} = TTR$$

to corresponding destinations is maintained in a new dataset shortest paths dataset (sPathsData). In sPathsData, each record comprises of a destination location and set of all shortest paths, which leads to the corresponding destination along with their distances. The NG employs Dijkstra' s [29] single-destination shortest path algorithm for determining the all shortest paths to the given destination (refer Algorithm 2).

Algorithm 2: Shortest paths determination in proposed methodology.

```
Input  : Set of Values (D, G)
Output: Set of Records (Set of weighted Shortest paths)
// Initialize destination D with 0 and all others vertices with ∞ in W, of G
Initialize Single destination (G,D)
// Initially no vertices´~shortest path is determined in S
S=∅
// build a min heap Q with V vertices of Graph G
Q=G.V
// Until Q heap get empty
while Q≠∅ do
    // Extract the minimum calculated shortest distant vertex from Q
    u=extract_min(Q)
    // add u to S
    S=S∪{u}
    // for each vertex adjacent to u
    for each vertex v∈G Adj[u] do
        // Relax the edge uv with weight w
        relax(u, v, w)
            // If the distance from destination to u via v is shorter, take that path
        if d(u) > d(v)+w then
            d(u)=d(v)+w
        end
        return d(u)
    end
end
```

The Algorithm 2 uses the following terminologies for shortest paths determination.

- Destination (D): D is the destination, for which all possible shortest paths will be calculated.
- Weighted graph (G): G is the directed graph of the entire area having distance weighted edges.
- Cost matrix (W): W is the cost matrix having the distance of each vertex to the destination.
- Calculated paths (S): S is the set of vertices, whose shortest paths to the destination have been calculated.

- Function extract_min(Q): extract_min(Q) extracts the minimum calculated distant vertex from heap Q.
- Distance from destination d(u): d(u) is the distance from destination to a vertex u.

$$TTR = \frac{2v \pm \sqrt{4v^2 - 8ad}}{2a} \tag{2}$$

where d is the current distance of the vehicle from the signal point. v is the velocity of the vehicle. a is the acceleration/deceleration of the vehicle.

2.2.2. Data analysis

Based on the respective datasets in traffic flow data (TFD) and pre-decision data (PDD) repositories, the waiting queue lengths, waiting times, and adaptive green phase time spans at the signal points, are predicted by the data analysis (DA) component of the cloud subsystem. The methodology to predict the waiting queue length and waiting time, and using the same to predict the adaptive green phase time span for respective signal points, is explained through the Algorithm 3 with

Algorithm 3: Data analysis methodology for predicting green phase time span.

```
Input  : Set of Values (lanId, vId, TTR, vCount)
Output: Sets of Records (wql, wt, AGPT)
// initialize wql to vehicle count sensed by respective LDS
wql=vCount
vehCons= $\emptyset$
// repeat for every vehicle until DA phase is over
while DA phase$\neq$over do
    // calculate TTS using time stamps $t_G$ and $t_C$
    TTS=$t_G$-$t_C$
    // if vehicle seems to reach in time
    if TTR<TTS then
        // Check the consideration of vehicle in waiting queue. if yes, simply ignores it
        if vehicleCondiderd(vId) then
            continue
        end
        // if vehicle is not considered already, include it into waiting queue
        else
            wql=wql+1
            // mark the vehicle into considered set
            vehCons=vehCons$\cup${vid}
        end
    end
    // if vehicle doesn't seems to reach in time
    else
        // remove the already considered vehicle from the set and decrement the wql, otherwise ignore
        if vehicleConsidered(vid) then
            wql=wql-1
            vehCons=vehCons-{vid}
            continue
        end
        continue
    end
    // Calculate time for crossing k meters of signal point area
    $\tau$=2k/$v_S$
    // Calculate the time taken to clear traffic queue
    TCT= (wql*$\tau$)
    // calculate total waiting time at the signal
    wt=TTS+TCT
end
// calculate the final adapted green phase time
AGPT=wt
```

following postulations.

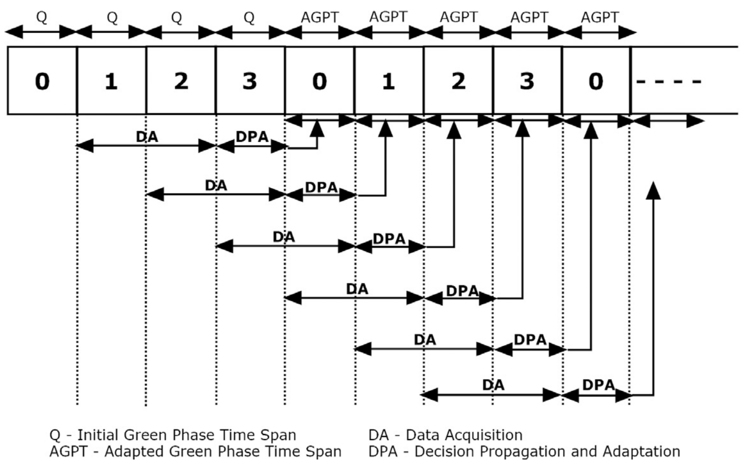
Q
AGPT
0
1
2
3
DA
DPA
Q - Initial Green Phase Time Span
AGPT - Adapted Green Phase Time Span
DA - Data Acquisition
DPA - Decision Propagation and Adaptation

Fig. 2. Adaptive green phase prediction timeline. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

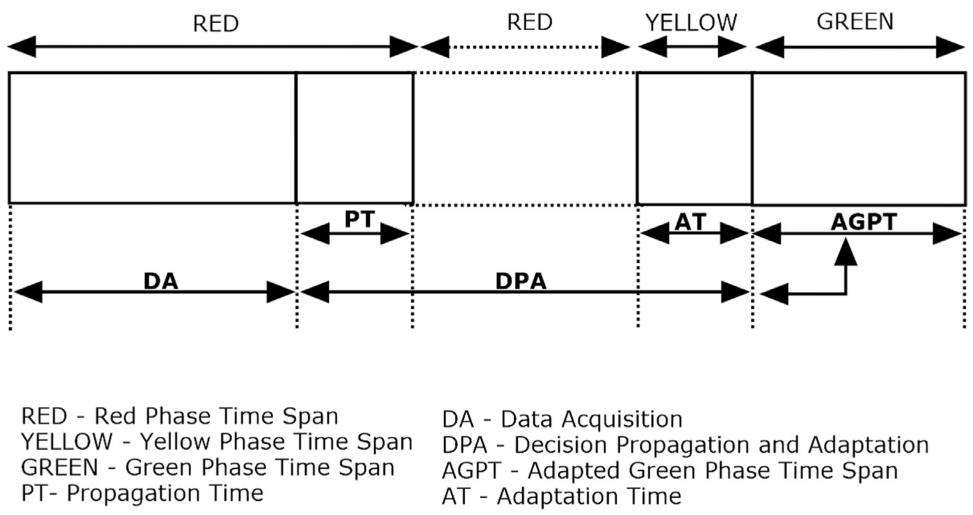
RED
RED
YELLOW
GREEN
PT
AT
AGPT
DA
DPA
RED - Red Phase Time Span
YELLOW - Yellow Phase Time Span
GREEN - Green Phase Time Span
PT- Propagation Time
DA - Data Acquisition
DPA - Decision Propagation and Adaptation
AGPT - Adapted Green Phase Time Span
AT - Adaptation Time

Fig. 3. Traffic movement phase distribution. (For interpretation of the references to color in text, the reader is referred to the web version of this article.)

- The traffic movement has three phases at a signal point: (a). Red, when traffic halts at signal point (b). Yellow, the time span between the red phase and green phase, meant to get ready for crossing the signal point (c). Green, when the traffic is allowed to cross the signal point.
- The initial time span for the green phase at each signal point is Q seconds, and signal points at each intersection are numbered in a sequence as 0, 1, 2, and 3.
- The green phase rotation on signal points is in a round-robin fashion at each intersection.
- The Data is acquired and waiting time is calculated continuously for the prediction of the next green phase time span at ith signal point, during the green phase at ((i+1) mod4)th and ((i+2) mod4)th signal points. It is represented by the data acquisition (DA) phase (refer Fig. 2).
- During the green phase at ((i+3) mod4)th signal point, the predicted Adapted Green Phase Time (AGPT) span is sent to respective SCS for the next green phase time span adaptation at ith signal point. It is shown as a decision propagation and adaptation (DPA) phase.

The traffic movement phase distribution at a traffic signal point based on the discussed methodology (refer Fig. 2) is depicted in Fig. 3. The red phase comprises the entire DA phase, and the adaptation time period excluded DPA phase. The Yellow phase comprises of the adaptation time (AT) period of the DPA phase. The DPA phase is comprised of at least the propagation time (PT) and AT periods because at least this much time is required for propagation of the predicted AGPT from cloud to the SCS and for the adaptation of the next green phase time span. But, the DPA phase can be longer than the minimum time because of the AGPT of the ((i+3) mod4)th signal point. Hence, the extra time in DPA phase is entitled to the red phase. The green phase is comprised of the predicted AGPT. Every phase at ((i) mod4)th signal point is comprised of the Green phases of the other signal points (refer Fig. 2). Hence, every green phase should be at least equivalent to the combined period of PT and AT to accommodate DPA phase at other signal points.

The data analysis algorithm uses the following terminologies for predicting the waiting queue length, waiting time, and adapted green phase time span for a signal point (refer Algorithm 3).

- Waiting queue length (wql): wql depicts the number of vehicles that are and will wait at the signal point for the next green phase.
- Vehicle considered set (vehCons): vehCons is the set of vehicles on a lane, which is considered for wql.
- Time to reach (TTR): TTR determines when the vehicle will reach to the signal point down the lane.
- Time to signal (TTS): TTS initially determines the time left for the start of next green phase at the respective intersection, but subsequently, during the convergence of DA phase it extends to the time left for the start of next adapted green phase specifically for that respective signal point. TTS enables the DA phase to consider only those vehicles which are reaching the ((i) mod4)th signal point up to the end of the green phase at ((i+3) mod4)th signal points.
- Green Phase Timestamp(t_G): t_G initially depicts the time stamp for the start of next green phase at the respective intersection, but subsequently during the convergence of DA phase, it depicts the start of next adapted green phase specifically for that respective signal point. t_G enables the DA phase to calculate the TTS.
- Current Timestamp(t_C): t_C is the timestamp for the current time.
- Function vehicleConsidered(vId): vehicleConsidered is a function that determines whether the vehicle with vId is already considered in vehCons set or not.
- Signal crossing time (τ): τ is the time requires to pass a particular cross-section of the road from the signal point during the green phase.
- Speed limit at Signal (v_S): v_S is the maximum speed limit at the signal crossing, which can be attained from an initial speed of 0 km/h.
- Time to clear traffic (TCT): TCT is the time taken to clear the entire traffic, which was under consideration for adaptive green phase time planning.
- Waiting time (wt): wt is waiting time for a vehicle to cross signal point because of the wql at that signal point.
- Adapted green phase time (AGPT): AGPT is the final predicted waiting time at a signal point.

The Algorithm 3 monitors all vehicles which are arriving on a lane. It considers all vehicles which are already waiting at the traffic signal point and only those arriving vehicles which are supposed to arrive at traffic signal up to the end of Data Acquisition phase (DA). The Algorithm 3 predicts the reachability of arriving vehicles based on their TTR. For real-time reachability analysis and subsequently, the consideration for the waiting queue, the mobility data is monitored continuously. After calculating the vehicle queue length, Time required to Clear Traffic (TCT) is calculated using the Waiting Queue Length (wql) and the time (τ) required to cross a particular cross-section of the road to assure that the vehicle has left the signal point. Based on the TCT and Time to Signal (TTS), the total waiting time (wt) is calculated continuously for depicting real-time traffic scenario. At the end of the DA phase, this wt is considered as Adapted Green Phase Time (AGPT).

For crossing a distance of k meters from the signal point to assure that the vehicle has left the signal point, the time (τ) considers the speed limit v_s at the signal point. Because the vehicle is stationary before crossing, so initial velocity (u) is considered 0 km/h, and the final velocity (v) can be a maximum of v_s km/h. τ is calculated (refer Eq. (3)) using the derivation of Table 3 as follows.

$$\tau = 2k/v_S \tag{3}$$

After data analysis at the cloud subsystem, the predicted wql, wt, and AGPT, along with the entire shortest paths dataset (sPathsData), are forwarded to the action subsystem.

2.3. Action subsystem

In this subsystem, the AGPT is propagated to the respective signal controlling system (SCS). The AGPT is propagated to respective SCS in a fixed time called as propagation time (PT). The SCS adapts the green phase time span in another fixed time called as adaptation time (AT), and this period is called as the yellow phase. In the minimal extreme time, the DPA comprises of PT and AT (refer Fig. 3). The predicted waiting time and waiting queue length at signal points, and the entire dataset for the shortest paths for every destination (sPathsData) is broadcasted to the entire set of vehicles in that urban area through the cellular network-based navigation system. Now, the navigation system is capable of presenting an area-wide view of traffic scenario. Based on the area-wide view of traffic scenario and available set of shortest paths to the destination, the reverse edge layer (REL) helps in performing localized analysis for time optimized path identification and traffic load balancing.

2.3.1. Reset phase

This phase happens at the end of the predicted green phase time span. In this phase, the mobility data dataset (mData) and pre-decision dataset (pDecData) of the respective lane are deleted to repopulate with new data. A loop detector for passed vehicle (LDP), which is present at the signal point, counts the number of vehicles (vCountLDP) passed during the green phase. It communicates the vehicle count to respective signal point's LDS, to update the vehicle count (vCount). Now, the updated vehicle count at LDS depicts the vehicles which are still present at the signal point. After that, the LDP reset its count to zero (refer Algorithm 4).

Algorithm 4: Reset phase.

```
Input  : Set of Values (lanId)
Output: Empty mobility and pre-decision datasets and updated vehicle count at LDS
// delete mData and pDecData datasets of lane with identity lanId=i
delete($mData_i$, $pDecData_i$)
// send vCountLDP of LDP at $i$th lane to LDS at $i$th lane
send($vCountLDP_i$, $LDS_i$)
// recieve $vCountLDP_i$ and update the vCount of LDS at $i$th lane
$vCountLDP_i$=recieve()
$vCount_i$=$vCount_i$-$vCountLDP_i$
// reset the vCountLDP of LDP at $i$th lane
$vCountLDP_i$=0
```

2.4. Reverse edge layer

The edge computing provides localized and low latency computing at the edge of the data origin. It provides computing [30] to the data that relays from user to cloud. The same concept is used but in a novel way to provide localized and low latency computing by analyzing the data analytics that relays in reverse direction i.e., from cloud to user. Hence, the reverse edge layer provides localized, and low latency mobile computing [31] by analyzing data analytics that arrived from the cloud at the OBU. In the proposed system, it identifies an optimized path by computing and comparing the total journey time of each shortest path with the chosen path. This layer, based on the area-wide traffic scenario i.e., waiting queue lengths and waiting times at intersections, and average journey time of possible shortest paths to the destination, run a smart navigation and traffic load balancing algorithm (refer Algorithm 5). This algorithm chooses a time optimized shortest

Algorithm 5: Smart Navigation and Load balancing.

```
Input  : Set of Values (Set of k shortest paths to a destination, chosenPath, Area-wide Traffic scenario)
Output: selectedPath
// calculate combined weight of every $j$th shortest path from the set of k shortest paths
for $j = 1$, $j \leq k$, $j$++ do
    // calculate the non-queud distance of the $j$th path
    $NQD_j$=$TD_j$ - $\sum wql_j$
    // calculate the average time taken to cover $NQD_j$
    $NPT_j$ = $NQD_j/\bar{v}$
    // calculate the combined weight of the $j$th path
    $w_j$= $NPT_j$ + $\sum wt_j$
    // save the calculated $w_j$ in an weightedPaths list
    weightedPaths [j]= $w_j$
end
// choose best path, which posses minimum combined weight
bestPath=min(weightedPaths)
// compare the journey time of best path with the chosen path
if $bestPath < chosenPath$ then
    // redirect to bestPath, if its journey time is lower, otherwise continue on the chosen path
    selectedPath = bestPath
end
```

path to the destination in such a manner that the combined weight of traffic waiting time and the average time to cover the distance, remains least. This algorithm enables the balanced traffic distribution to all possible paths along with smart navigation through time optimized paths.

The proposed smart navigation and traffic load distribution works on the minimum weighted path approach, where the path possesses a combined weight of the aggregate waiting time because of traffic waiting queues at intersections and average journey time to reach the destination. The algorithm uses the following terminologies for identifying an optimal path and enabling the balanced distribution of traffic on all possible paths.

- Total distance (TD_j): TD_j is the total distance of the jth path.
- Aggregate Waiting Queue Length ((Σwql_j): Σwql_j depicts the aggregate queue length at jth path.
- Non Queued Distance (NQD_j): NQD_j is the waiting queues excluded distance of jth path.
- Average velocity ($\bar{v}$): $\bar{v}$ is the average of all communicated velocities by the vehicle in the current journey.
- Non-queued Path Time (NPT_j): NPT_j is the average time required to cover the distance NQD_j.
- Aggregate Waiting time (Σwt_j): Σwt_j depicts the aggregate waiting time at the jth path because of waiting queues at intersections.
- Combined weight (w_j): w_j is the combined weight of aggregate waiting time and average journey time to reach the destination on jth path.
- Weighted Path List (weightedPaths [ ]): weightedPaths [ ] is the list of calculated combined weights of j shortest paths.
- Function min(weightedPaths): min(weightedPaths) determines the minimum value from the weightedPaths list.
- Best Path (bestPath): bestPath is the time of the path having least weight from the weightedPaths list.
- Chosen Path (chosenPath): chosenPath is the journey time of the chosen path.
- Selected Path (selectedPath): selectedPath is the variable which signifies the time optimized selected path.

3. Experimental evaluation

The proposed system comprises two modules: adaptive traffic phase time planning, and smart navigation and load balancing. Hence, the experimental evaluation is conducted into two stages as follows.

3.1. Adaptive traffic phase time planning experimental evaluation

Due to the non-availability of mobility data of interest, the data is generated systematically for the experimental evaluation of the adaptive traffic phase time planning in the proposed methodology. For this, we have taken the help of a mathematical model to generate mobility dataset aligned with the proposed algorithm (refer Algorithm 3). Hence, the traffic phase time planning experimental evaluation is explained through two phases: data generation and data classification efficiency.

3.1.1. Data generation

For vehicle mobility data, we have taken a mathematical model aligned with the proposed algorithm (refer Algorithm 3) to generate the dataset. It is assumed that the RSUs are present at every 500 m distance apart on the lanes, and each lane is 5 km in length. The speed limit is a very crucial parameter to consider for traffic management, especially in urban spaces. Hence, it is assumed that the speed limit is 80 km/h. For asserting acceleration range, we have determined the minimum and maximum time to cover a distance between two consecutive RSUs using Eq. (4).

$$T_c = \frac{d_c}{v} \tag{4}$$

where d_c is the distance between two consecutive RSUs or two points. v is the velocity of the vehicle. T_c is the time taken to cover the distance between two consecutive RSUs or two points.

To cover a distance in minimum time, the velocity should be maximum, i.e., $v = 80$ km/h (because of the speed limit). To cover a distance in maximum time, the velocity should be minimum, i.e., $v = 1$ km/h. Based on the minimum and maximum time to cover a distance between two consecutive RSUs, the acceleration is determined based on the Eq. (1). But, because the acceleration is dependent on two factors: change in velocity and time taken, the acceleration can vary in different cases. So, we assumed three extreme cases based on the extreme values of change in velocity and time taken to determine the range of acceleration as follows.

I If the change in velocity ($\triangle v$) is maximum For maximum change in velocity, initial velocity (u) = 0 km/h and final velocity (v) = 80 km/h, and acceleration is calculated further in two sub cases:
  (a) If minimum time is taken to cover a distance between two consecutive RSUs
  (b) If maximum time is taken to cover a distance between two consecutive RSUs

II If the change in velocity ($\triangle$v) is minimum For minimum change in velocity, initial velocity (u) = 0 km/h and final velocity (v) = 1 km/h or initial velocity (u) = 79 km/h and final velocity (v) = 80 km/h, or anything, which has $\triangle v = 1$ km/h and acceleration is calculated further in two sub cases:
  (a) If minimum time is taken to cover a distance two consecutive RSUs
  (b) If maximum time is taken to cover a distance two consecutive RSUs

III If there is no change in velocity ($\triangle v$) = 0
  In this case, $u = v$ and $\triangle v = 0$ km/h. So, the acceleration remains 0 m/s^2 in every situation.

Hence, the acceleration range is determined using cases I, II, and III. Similarly, for deceleration, the range is also determined in the same way. Now the mathematical model has velocity range, acceleration/deceleration range, and the distance range.

The dataset is generated using random values for velocity, acceleration, and distance from the logically well-taken ranges to simulate the randomness and real-world vehicle mobility (refer Algorithm 6). The TTR is calculated based on the Eq. (2).

Algorithm 6: Mobility data generation .

```
Input  : Set of Values (Range of velocity values (V), Range of accelearation/deacceleration values (A), and Range of
         distance values (D))
Output: Set of Records (Mobility dataset)

// Specify the number of required mobility records (n) and initial value of iteration number (i)
int n, i=0
float mData[n,3]
// Runs untill the specified number of mobility records are not generated
while i<n do
    // Assign random values from the well defined ranges of velocity, acceleration/deceleration and
       distance
    float v=rand(V)
    float a=rand(A)
    float d=rand(D)
    // create a temporary record
    float temp[ ]={v, a, d}
    // Search the temporary record (temp) in mobility dataset
    search(temp, mData)
    // If searched record is found in mobility dataset, ignore it
    if record found then
        continue
    end
    // If searched record is not found in mobility dataset, append the temp record in the mobility dataset
    else
        mData[i,3]=temp
        i++
    end
end
```

A dataset with TTR of every vehicle on a particular lane is considered as the final data readings at the end of the data acquisition (DA) phase (refer Fig. 2).

3.1.2. Data classification efficiency

The data classification techniques categorized the TTR values into two classes: True and False, to determine whether the vehicle would reach before the start of the green signal or not. For analyzing the reachability prediction of the vehicles, four baseline classification approaches: J48 Decision tree (DT), NaiveBayes (NB), Support vector machine (SVM), and K-nearest neighbor (KNR) are utilized using weka 3.8.2 [32]. Ten folds cross-validation is used to synthesize optimal results. It means the entire dataset is divided into ten parts from which nine parts are used as training data, and the tenth part is used as a testing data. The batch size for every classifier is set to 100. For SVM, Polykernel E 1.0 C 250,007 is used for kernel function. Four data samples of 400, 600, 800, and 1000 instances are used. To compare the performance of utilized classifiers, four statistical measures: Accuracy, Sensitivity, Specificity, and F-measure are observed.

Result analysis

Choosing an appropriate classifier for prediction, especially in real-time situations, is very critical because it improves the performance of the system. Hence, the same sized data sample is used for all four classifiers to predict the reachability of the vehicles based on the acquired mobility parameters: velocity, acceleration, and distance. The comparison of the statistical results of the utilized data classifiers is shown in Fig. 4(a)–(d). Based on statistical result comparisons, the J48 decision tree outperforms the other three classifiers. The accuracy (correct prediction rate) of the J48 Decision tree is highest, with an average rate of 0.99. The average sensitivity (true positive rate) of J48 is 0.95, and it is the highest of all. Similarly, the specificity and F-Measure of J48 are also highest with an average rate of 1 and 0.97, respectively. As far as Accuracy, Sensitivity, Specificity, and F-measure are concerned, the J48 is the best performing classifier. The statistical results depict that the J48 decision tree can efficiently classify the mobility data for predicting vehicle reachability.

3.2. Smart navigation and traffic load balancing experimental evaluation

For smart navigation and traffic load balancing experimental evaluation, a code is designed based on the proposed algorithm (refer Algorithm 5) in c environment. A particular destination is taken with a set of weighted shortest paths to that destination. The paths are assumed as single-edged. The experiment uses four different sized traffic samples of 100, 200,

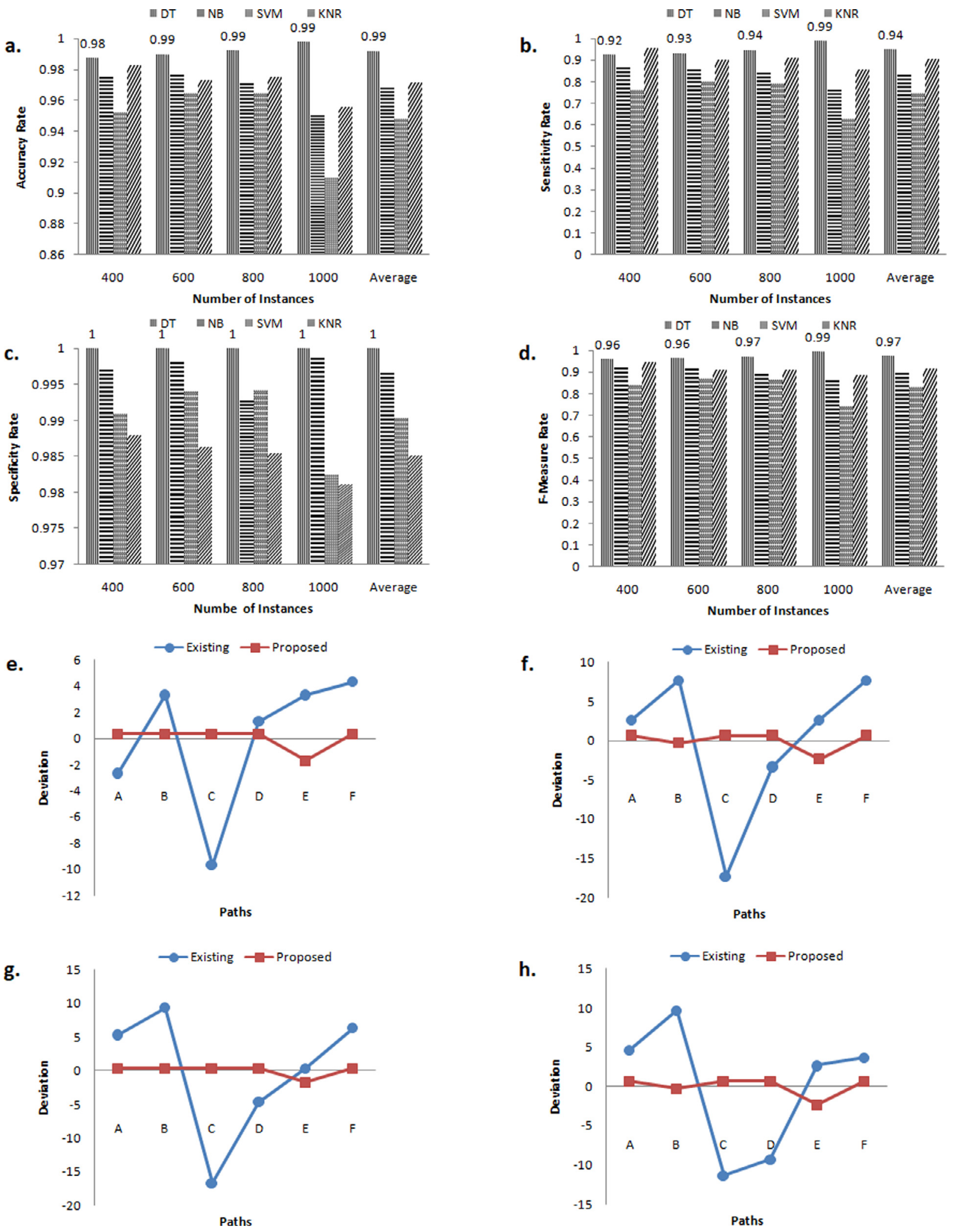

Fig. 4. Result Comparison: classification efficiency of mobility data (a) Accuracy (b) Sensitivity (c) Specificity (d) F-measure; load balancing efficiency (e) Load balancing of 100 vehicles (f) Load balancing of 200 vehicles (g) Load balancing of 400 vehicles (h) Load balancing of 500 vehicles.

400, and 500 vehicles. For simulating the existing traffic distribution scenario [22], the vehicles are allowed to proceed in any path. It is simulated using a random function. The traffic volume at each path is assumed as the waiting queue because of the single-edged paths.

For proposed traffic distribution, a combined weight of the waiting time due to traffic queues at intersection and time taken to cover the distance of every shortest path to the destination is determined using smart navigation methodology. Now, a vehicle is only allowed to proceed to a particular path, which has the minimum combined weight of all the paths. The traffic volume distribution to each path in both existing and proposed system is statistically analyzed using standard deviation (refer Eq.(5)) as follows.

$$\sigma = \sqrt{\frac{\sum x_k - \bar{x}}{k-1}} \tag{5}$$

where σ is the standard deviation. x_k is the volume of the kth path. $\bar{x}$ is the mean of the traffic volumes of k paths.

3.2.1. *Traffic load balancing evaluation*

The standard deviation is a very significant statistical representation to analyze the effectiveness of a traffic load balancing. The deviation in statistics is a difference of an observed value from another value (in our case, this value is a mean value). It shows how far the value deviates from its mean value. If the traffic load evenly distributes to each path, then the mean of all traffic volumes equals a single volume, and the deviation of every volume from its mean value converges to zero. Hence, more close to zero means more even values and lesser close to zero means lesser even values. This can be well depicted from the Fig. 4(e)–(h), where the deviation of traffic in the proposed system is not far from the zero on all six paths. But, in the existing system, the traffic deviates farther from the zero as compared to the proposed system. On average, the standard deviation of the existing system is 8.17, and the proposed system is 1.01. The statistical analyses of the experimental results show that the distribution of traffic load by the proposed system is more optimal as compared to the existing non-smart system.

3.2.2. *Road safety analysis*

The increase in volume and subsequently increase in density of traffic affects road safety at intersections. With higher traffic volumes, the road lanes congest. It results in more rear-end crashes at the tail of the vehicle queues at the intersections. The reason for the increase in such crash rates is the flow compression, and the velocity difference between the waiting queue's rear-end and inflow traffic. It tends the resultant headway so small that the error compensation or crash avoidance becomes least or impossible. This can be explained from the kinematic flow theory [33], where the average distance between vehicles can be defined as a function of density under operational conditions z (refer Eq. (6)) and the distance required to stop a vehicle, when it decelerates using the basic motion equation (refer Eq. (7)) can be explained as follows.

$$d_z = c\frac{1}{\rho_z} \tag{6}$$

$$d_r = \frac{{s_i}^2 - {s_e}^2}{2a} \tag{7}$$

where d_z is the average distance between vehicles under operational conditions z. c is the constant that approximately accounts for the distance occupied by vehicles. ρ_z is the density of vehicles under operational conditions z. d_r is the distance required to decelerate from s_i to s_e. s_i is the initial velocity. s_e is the end velocity. a is the rate of deceleration.

Under safe operational conditions, the distance required to stop a vehicle to avoid a crash with the vehicle ahead must be less than the average available distance between the vehicles (refer Eq. (8a)). Assume, Se = 0 to stop the inflow vehicle completely because of the stationary rear-end vehicle in traffic waiting queue (refer Eq. (8b)).

$$\frac{{s_i}^2 - {s_e}^2}{2a} < c\frac{1}{\rho z} \tag{8a}$$

$$\frac{{s_i}^2}{2a} < c\frac{1}{\rho z} \tag{8b}$$

$$\frac{{s_i}^2}{2a} \propto \frac{1}{\rho z} \tag{8c}$$

$$SafetyIndex_j \propto \frac{1}{\rho z} \tag{8d}$$

where SafetyIndex$_j$ reflects the required distance to decelerate the vehicle to stationery and subsequently to avoid rear-end crash at the intersection of the jth path.

It is depicted from the Eq. (8c) that the density of vehicles on a lane should be inversely proportional to the distance required to decelerate the vehicle to stationery. Using the comparison theory, it can be observed that the increase in density

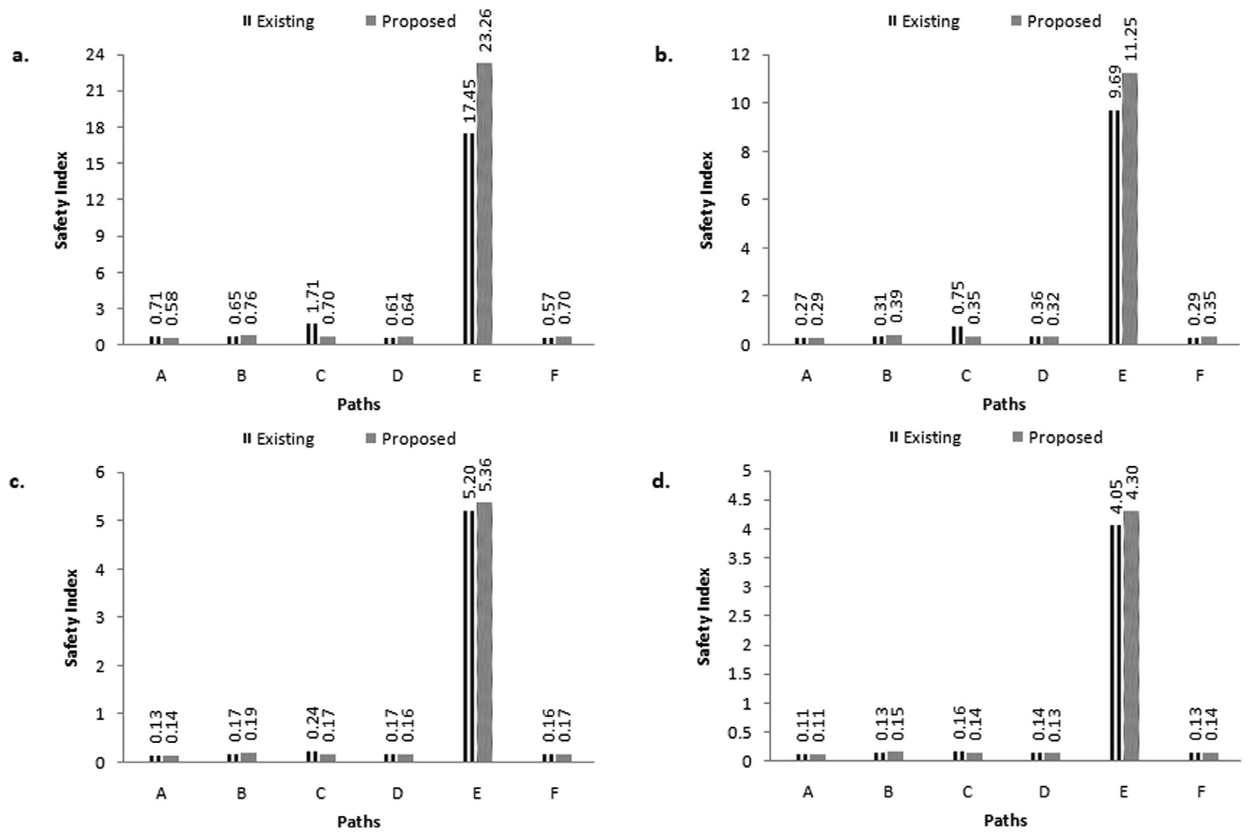

Fig. 5. Safety index comparison of the paths with traffic of (a) 100 vehicles (b) 200 vehicles (c) 400 vehicles (d) 500 vehicles.

decreases the average distance between the vehicles, which further decreases the distance required to decelerate the vehicle to avoid collision (refer Eq. (8a)). Hence to stop a vehicle fully under safe condition, the deceleration should be more. But under the same operating conditions z, the decelerations remains the same. Hence, there's more probability of crashes between the oncoming vehicle and the rear-end of the stationed vehicle at the signal queue, if the density of vehicles on the lane is higher. To validate the above-stated point, a safety index of every lane is calculated using Eq. (8d). The safety index comparison of paths under the existing and proposed system is depicted in Fig. 5(a)–(d). The result analyses acknowledge the fact that more paths having a higher safety index in the proposed system as compared to the existing system. For a specific path E (refer Fig. 5(a)–(d)), the safety index in both existing and proposed systems is much higher than the safety index for other paths in respective systems. The reason for these kinds of results is the distance for path E is much higher than the distance of other paths, which results in lesser traffic density on this path and subsequently, a higher safety index. But, comparatively, the safety index of path E is higher in the proposed system than the existing system. The overall average safety index of all paths is 2.12 in the proposed system as compared to the 1.84 in the existing system. Hence, it can be proved that, with the optimal (or balanced) distribution of traffic on all possible paths, the road safety perspective for vehicles at intersections, also gets improved.

4. Conclusion

Vehicular traffic management is very crucial for many essential services, especially in urban spaces. Hence, in this paper, Edge Cloud-centric IoT based smart traffic management system is developed for traffic inflow prediction and smart navigation of vehicles based on the current traffic scenario. The traffic inflow prediction enables the traffic movement phases at intersections to adapt according to the traffic inflow and avoid long waiting queues and congestions at intersections. The edge computing-based smart navigation, based on predicted waiting queues and waiting times at intersections lying on the chosen route, guide the vehicles to follow time optimized routes to the destinations in real-time. The smart navigation also enables the optimal distribution of traffic load to all possible paths and improves the road safety perspective for vehicles at intersections. Four classifiers: J48 Decision Tree, NaiveBayes, SVM, and k-nearest neighbor are utilized to analyze the efficiency of the best classifier to predict the traffic inflow at the intersection. The statistical analyses found that J48 Decision Tree outperformed all the remaining three in predicting the traffic inflow. The results' comparison and discussion of smart navigation based traffic load distribution and subsequently the road safety at intersections indicates the optimality of the proposed system. The literature review, results, and comparison of the proposed system with the existing system acknowledge its effectiveness and comprehensiveness in predicting traffic flow, enabling time optimized smart navigation, optimal traffic load balancing, and improved road safety for vehicles at intersections. This paper has a huge future scope for managing ever-increasing traffic in technology-equipped urban spaces. VANET can also be explored in the domain of energy-efficient vehicular traffic management systems.

Declaration of Competing Interest

None.

CRediT authorship contribution statement

Sahil: Conceptualization, Data curation, Formal analysis, Methodology, Writing - original draft, Writing - review & editing. **Sandeep Kumar Sood:** Conceptualization, Methodology, Supervision, Validation.

References

[1] UN, World Population Prospects, 2012.
[2] T. Litman, Congestion Costing Critique: Critical Evaluation of the “Urban Mobility Report”, 2013.
[3] D. Schrank, B. Eisele, T. Lomax, TTIS 2012 Urban Mobility Report, 2012.
[4] G. Sun, L. Song, H. Yu, V. Chang, X. Du, M. Guizani, V2V routing in a vanet based on the autoregressive integrated moving average model, IEEE Trans. Veh. Technol. 68 (1) (2019) 908–922.
[5] G. Sun, M. Yu, D. Liao, V. Chang, Analytical exploration of energy savings for parked vehicles to enhance vanet connectivity, IEEE Trans. Intell. Transport.Syst. (99) (2018) 1–13.
[6] C.-C. Hung, H. Chan, E.H.-K. Wu, Mobility pattern aware routing for heterogeneous vehicular networks, in: Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE, IEEE, 2008, pp. 2200–2205.
[7] R.C. Eberhart, Y. Shi, Computational Intelligence: Concepts to Implementations, Elsevier, 2011.
[8] M. Batty, Big data, smart cities and city planning, Dialogues Hum. Geogr. 3 (3) (2013) 274–279.
[9] A. Wahid, M.A. Shah, F.F. Qureshi, H. Maryam, R. Iqbal, V. Chang, Big data analytics for mitigating broadcast storm in vehicular content centric networks, Fut. Gener. Comput. Syst. 86 (2018) 1301–1320.
[10] P. Verma, S.K. Sood, Cloud-centric IoT based disease diagnosis healthcare framework, J.Parallel Distrib. Comput. 116 (2018) 27–38.
[11] S.K. Sood, I. Mahajan, A fog-based healthcare framework for chikungunya, IEEE Internet Things J. 5 (2) (2018) 794–801.
[12] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: vision and challenges, IEEE Internet Things J. 3 (5) (2016) 637–646.
[13] Y. Feng, K.L. Head, S. Khoshmagham, M. Zamanipour, A real-time adaptive signal control in a connected vehicle environment, Transport. Res. Part C 55 (2015) 460–473.
[14] M. Whaiduzzaman, M. Sookhak, A. Gani, R. Buyya, A survey on vehicular cloud computing, J. Netw. Comput. Appl. 40 (2014) 325–344.
[15] T. Darwish, K.A. Bakar, Traffic density estimation in vehicular ad hoc networks: a review, Ad Hoc Netw. 24 (2015) 337–351.
[16] S.K. Sood, R. Sandhu, K. Singla, V. Chang, IoT, big data and HPC based smart flood management framework, Sustain. Comput. 20 (2018) 102–117.
[17] L. Bedogni, L. Bononi, M. Di Felice, A. D’Elia, R. Mock, F. Morandi, S. Rondelli, T.S. Cinotti, F. Vergari, An integrated simulation framework to model electric vehicle operations and services, IEEE Trans. Veh. Technol. 65 (8) (2016) 5900–5917.
[18] A. Abadi, T. Rajabioun, P.A. Ioannou, et al., Traffic flow prediction for road transportation networks with limited traffic data, IEEE Trans. Intell. Transport. Syst. 16 (2) (2015) 653–662.
[19] J.L. Pérez, A. Gutierrez-Torre, J.L. Berral, D. Carrera, A resilient and distributed near real-time traffic forecasting application for fog computing environments, Fut. Gener. Comput. Syst. 87 (2018) 198–212.
[20] H.-J. Chang, G.-T. Park, A study on traffic signal control at signalized intersections in vehicular ad hoc networks, Ad Hoc Netw. 11 (7) (2013) 2115–2124.
[21] S. Wan, X. Li, Y. Xue, W. Lin, X. Xu, Efficient computation offloading for internet of vehicles in edge computing-assisted 5g networks, J. Supercomput. (2019) 1–30, doi:10.1007/s11227-019-03011-4.
[22] M. Collotta, L.L. Bello, G. Pau, A novel approach for dynamic traffic lights management based on wireless sensor networks and multiple fuzzy logic controllers, Expert Syst. Appl. 42 (13) (2015) 5403–5415.
[23] N. Wan, A. Vahidi, A. Luckow, Optimal speed advisory for connected vehicles in arterial roads and the impact on mixed traffic, Transport. Res. Part C 69 (2016) 548–563.
[24] B. HomChaudhuri, R. Lin, P. Pisu, Hierarchical control strategies for energy management of connected hybrid electric vehicles in urban roads, Transport. Res. Part C 62 (2016) 70–86.
[25] R. Zhang, P. Xie, C. Wang, G. Liu, S. Wan, Classifying transportation mode and speed from trajectory data via deep multi-scale learning, Comput. Netw. 162 (2019) 106861–106873.
[26] J. Liu, W. Wang, D. Li, S. Wan, H. Liu, Role of gifts in decision making: an endowment effect incentive mechanism for offloading in the IoV, IEEE Internet Things J. 6 (4) (2019) 6933–6951.
[27] G. Sun, V. Chang, M. Ramachandran, Z. Sun, G. Li, H. Yu, D. Liao, Efficient location privacy algorithm for internet of things (IoT) services and applications, J. Netw. Comput. Appl. 89 (2017) 3–13.
[28] R. Li, C. Cheng, M. Qi, W. Lai, Design of dynamic vehicle routing system based on online map service, in: Service Systems and Service Management (ICSSSM), 2016 13th International Conference on, IEEE, 2016, pp. 1–5.
[29] S. Pallottino, M.G. Scutella, Shortest path algorithms in transportation models: classical and innovative aspects, in: Equilibrium and advanced transportation modelling, Springer, 1998, pp. 245–281.
[30] P. Verma, S.K. Sood, Fog assisted-IoT enabled patient health monitoring in smart homes, IEEE Internet Things J. 5 (3) (2018) 1789–1796.
[31] B. Huang, Z. Li, P. Tang, S. Wang, J. Zhao, H. Hu, W. Li, V. Chang, Security modeling and efficient computation offloading for service workflow in mobile edge computing, Fut. Gener. Comput. Syst. 97 (2019) 755–774.
[32] W. Fitriani, A.P.U. Siahaan, Comparison between Weka and Salford system in data mining software, Int. J. Mobile Comput.Appl. 3 (4) (2016) 1–4.
[33] J. Kononov, C. Durso, D. Reeves, B.K. Allery, Relationship between traffic density, speed, and safety and its implications for setting variable speed limits on freeways, Transport. Res. Rec. 2280 (1) (2012) 1–9.