IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received August 11, 2020, accepted August 24, 2020, date of publication September 10, 2020, date of current version September 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3023263

Cost-Effective V2X Task Offloading in MEC-Assisted Intelligent Transportation Systems

ANDREY BELOGAEV[1,2], ALEXEY ELOKHIN[2], ARTEM KRASILOV[1,2], EVGENY KHOROV[1,3], (Senior Member, IEEE), AND IAN F. AKYILDIZ[1], (Fellow, IEEE)

[1]Wireless Networks Laboratory, Institute for Information Transmission Problems, Russian Academy of Sciences, 127051 Moscow, Russia
[2]Telecommunication Systems Laboratory, National Research Institute Higher School of Economics, Moscow Institute of Electronics and Mathematics, 123458 Moscow, Russia
[3]School of Radio Engineering and Computer Technology, Moscow Institute of Physics and Technology, 117303 Moscow, Russia

Corresponding author: Andrey Belogaev (belogaev@wireless.iitp.ru)

This work was done at IITP RAS and supported by the Russian Government under Contract 14.W03.31.0019.1

ABSTRACT Intelligent Transportation Systems (ITS) will become an essential part of every city in the near future. They should support various vehicle-to-everything (V2X) applications that improve road safety or even enable autonomous driving. Recently, the European Telecommunications Standards Institute (ETSI) introduced a multi-access (mobile) edge computing concept as a promising solution to satisfy the V2X delay and computational requirements. Based on this concept, the tasks generated by V2X applications can be offloaded to servers at the edge of the radio access network (RAN). There is a need for a task offloading algorithm that minimizes the ITS operator expenses connected with the servers deployment and maintenance while satisfying the requirements of the V2X applications. Most of the existing papers in the literature do not pay much attention to queuing delays at servers. In this paper, the queuing delays are analyzed by considering a general-type task computational time distribution. A non-linear optimization problem is formulated to minimize the ITS operator expenses subject to delays and computational resources constraints. The flexibility is also improved by considering that a delay constraint is satisfied with a given probability. To solve this problem, a method for linearization of the problem is proposed, and consequently, an algorithm based on Integer Linear Programming (ILP) is designed. A heuristic algorithm called Cost-effective Heuristic Algorithm for Task offloading (CHAT) is also introduced that provides close to optimal results and has much lower computational complexity than the ILP algorithm. The efficiency of the CHAT algorithm is studied in several scenarios in terms of the computational time, delays, and the total server energy consumption as the cost function. The results show that the CHAT algorithm satisfies the requirements of the V2X applications in all the considered scenarios and reduces the ITS operator expenses over twice compared with other algorithms proposed in the literature.

INDEX TERMS V2X, vehicular networks, integer linear programming, intelligent transportation systems, mobile edge computing, multi-access edge computing, task offloading.

I. INTRODUCTION

Intelligent transportation systems (ITS) attract much attention in both academia and industry in recent years. The deployment of these systems provides many benefits: the improvement of the driver quality of experience (QoE), the decrease of the accident numbers, the reduction of CO_2 emissions, etc. [1]. The key elements of any ITS are vehicle-to-everything (V2X) applications that use V2X communications to allow vehicles to exchange information with other cars and the surrounding infrastructure for maneuvers coordination. A significant part of these applications requires billions of computational operations to be executed within a very short time. Hence, these applications should use servers that are close to the road. To shorten the communication delays and improve the coverage of the V2X applications, an operator can deploy special infrastructure gateways called road-side units (RSUs) along the road [2]. Then, all capital (CAPEX) and operational (OPEX) expenses related to such deployment place a considerable burden on an operator. Therefore, we need an algorithm that minimizes the operator's expenses while satisfying the requirements of applications.

According to the 3rd Generation Partnership Project (3GPP) technical report [3], various V2X applications should be supported by the next generation of mobile systems (5G).

The associate editor coordinating the review of this manuscript and approving it for publication was Najah Abuali.

TABLE 1. V2X applications and requirements.

Application	Message size, bytes	Frequency, Hz	E2E latency, ms
Vehicle platooning	50 – 1200	30 – 50	20
Intersection safety	400 – 500	50	10
Environment perception	1600	5 – 10	3

These applications have different quality of service (QoS) requirements and traffic parameters. In Table 1, we list three applications considered in this paper with a corresponding message size, a message transmission frequency, and a required vehicle-to-infrastructure (V2I) end-to-end latency, i.e., the time interval between the transmission of a message at the source and the reception of the message at the destination [3]:

- *Vehicle platooning* is an application that allows vehicles to move in a closely linked manner. The vehicles can dynamically form a platoon and select one vehicle to be a platoon leader. The platoon leader reports information about the platoon parameters, e.g., its speed, route, lane, and road conditions to an RSU. Using the information obtained from surrounding vehicles and its own sensors, the RSU can advise the platoon members to modify their motion parameters accordingly.
- *Intersection safety information provisioning* is an application that is aimed at preventing accidents at intersections. For that, the vehicles and RSUs exchange information about surrounding vehicles and pedestrians and decide how to cross the intersection, i.e., determine priority, speed, lane selection, etc.
- *Collective perception of environment* is an application that allows sensing the road conditions, e.g., detecting objects on the road, such as obstacles or vehicles that do not support V2X communications. Based on this information, RSUs advise vehicles to change their speed, make a maneuver, e.g., passing, overtaking, lane change, etc.

Besides very strict delay constraints, the novel V2X applications require computational resources to find optimal trajectories of autonomous vehicles. To satisfy both delay and computational requirements simultaneously, the multi-access (mobile) edge computing (MEC) concept was introduced by the European Telecommunications Standards Institute (ETSI) [4]. According to this concept, the computational resources are moved close to the edge of the radio access network (RAN), i.e., closer to the vehicles. Many papers [5]–[12] have proposed architectures for MEC-assisted ITS recently. In Fig. 1, we illustrate a typical MEC-assisted ITS architecture. Generally, it consists of V2X-capable vehicles, RSUs, and cellular base stations (BSs). Whereas RSUs are deployed to provide assistance and connectivity to vehicles on the roads, BSs, as a part of metropolitan cellular system infrastructures, serve much larger areas with many other users in addition to vehicles. Computational resources are provided by servers that have all the necessary capabilities for V2X applications, i.e., CPU, memory, storage, etc. These servers can be deployed: (i) at RSUs; (ii) at BSs; (iii) in an operator core network; (iv) on the Internet, e.g., rented from cloud services. Vehicles and RSUs are equipped with various sensing devices, such as RADARs, LIDARs, cameras, GPS, etc., and use them to collect information about the current situation on the road. To make a recommendation for vehicles (speed/acceleration tuning, passing, overtaking, lane change, etc.), V2X applications at RSUs generate tasks using the collected information as task input parameters. Following the MEC paradigm, the servers are deployed at RSUs, allowing access to computational resources with low latency. However, the remote servers at BSs, in a core network, or on the Internet can have a much higher performance. So, for each task, an RSU has a choice: make all calculations using its own computational resources or offload a task to a remote server, which is more efficient and/or less loaded.

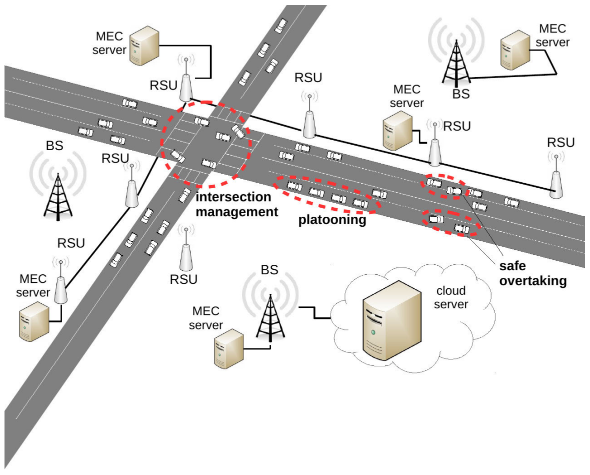


FIGURE 1. MEC-assisted intelligent transportation system architecture.

In this paper, we consider the problem of task offloading that minimizes the operator's expenses, i.e., CAPEX/OPEX, subject to delay constraints of various V2X applications given the network topology and the location of computational resources. The main contributions of the paper are as follows:

- We provide a new mathematical formulation of the task offloading problem as a non-linear integer optimization problem. In contrast to the existing works, we consider that the task computational time at a server has a general-type distribution.
- We show how to linearize the problem and propose a new algorithm based on integer linear programming to solve it.
- We design a greedy heuristic algorithm called CHAT that selects servers with the minimal cost per task. This algorithm significantly reduces the computational complexity while providing close to optimal results.

TABLE 2. Overview of the existing solutions.

Papers	Delay constraints	CAPEX/OPEX optimization	Number of applications	Solution complexity	Considering queuing delays
[6]	✓	✓	multiple	low	M/M/1 model
[13]	✓	×	multiple	low	×
[14]	✓	×	multiple	low	×
[15]	✓	×	multiple	low	×
[16]	✓	×	multiple	low	×
[17]	✓	×	multiple	high	game theory approach
[18]	✓	×	multiple	low	×
[19]	✓	✓	single	low	×
[20]	✓	✓	multiple	high	M/M/1 model
[21]	✓	✓	single	low	×
[22]	✓	✓	multiple	high	×
[23]	✓	✓	multiple	low	×
this paper	✓	✓	multiple	low	M/G/1 model

- We carry out an extensive performance evaluation of the proposed algorithms and compare their performance with the existing ones. We show that the proposed algorithms reduce the operator's expenses up to several times while satisfying heterogeneous requirements of V2X applications in all the considered scenarios.

The rest of the paper is organized as follows. In Section II, we review the existing studies related to the V2X task offloading problem. In Section III, we provide a mathematical statement of this problem. In Section IV, we propose the algorithms aimed at solving the formulated problem. In Section V, we evaluate the performance of the proposed algorithms and compare them with the existing ones. Finally, Section VI concludes the paper.

II. RELATED WORK

The problem of V2X task offloading attracted much attention in the last few years. As shown in Table 2, most of the studies formulate the problem as an optimization problem of minimizing a particular cost function subject to various constraints, such as delay requirements for various applications or finite computational resources at servers. The cost functions considered in various studies can be divided into two groups: delay-based and CAPEX/OPEX-based functions.

First, let us consider the delay-based functions. In [13], the authors formulate a binary integer linear programming (ILP) problem aimed at minimizing the average task processing delay for all tasks. Three types of V2X applications generate the tasks: cooperative awareness, decentralized environmental notification, and media downloading/streaming. The authors propose a greedy algorithm that achieves close to the optimal performance with a much lower computational complexity. Another heuristic algorithm is proposed for a similar problem in [14]. We consider both algorithms as the baseline for performance evaluation in Section V. Besides the average delay, the delay-based cost function may contain additional terms corresponding to vehicle quality of experience indicators. In particular, many papers consider energy consumed for a task calculation (locally at vehicles) or a task data transmission to MEC servers. In particular, the papers [15]–[17] consider the weighted sum of the average delay and the consumed energy as an objective function. The problem is solved by means of convex optimization [15], iterative heuristic algorithm [16], or game theory [17]. In [18], the authors propose to use energy harvesting at vehicles with computational capabilities on board. Since both the task computation and offloading consume energy, tasks can be dropped when the remaining energy is not enough. The task drop rate adds a penalty to the cost function. To solve the problem, the authors propose a Lyapunov-optimization-based algorithm and prove its asymptotic optimality.

Although the average task processing delay is a significant performance indicator, it does not include the operator's expenses connected with server deployment and maintenance. Hence, many papers [19]–[23] consider CAPEX/OPEX-based cost functions. In [19], the authors consider the cost function consisting of two terms: the number of used servers, and the number of required physical links. The delay constraint is formulated in terms of the number of hops. To reduce the computational complexity of the algorithm, the authors propose a hybrid approach that consists of two phases. First, the servers are selected with a heuristic algorithm. Then, the optimization problem is solved with an additional constraint on the servers that can be used. The significant drawback of the proposed framework is that it provides server selection without considering the demands of the computational tasks and the intensity of the task flows. Moreover, the number of hops constraint does not guarantee the satisfaction of the delay requirements because the delay varies with the server load and the task computational time.

In [20], the authors consider a computational resource cost as a cost function and formulate the problem as a finite-player game. The authors compare two offloading methods, one of which transmits tasks directly from a vehicle to an RSU while the other uses vehicle-to-vehicle communications to deliver tasks. They show that the latter method allows reducing the total cost for high traffic density. The solution for the task offloading problem is represented by a Nash equilibrium that is proved to exist, but its computing has very high complexity. Hence, in a large scale scenario with many vehicles, this algorithm can be inapplicable. In [21], the authors propose an algorithm for platooning tasks offloading, where each task is modeled as a sequence of subtasks. Each subtask can be calculated locally, offloaded to a neighboring vehicle, or offloaded to a MEC server. Similarly to [20], the authors use the computational resource cost that is different for platoon members and the MEC server. The task offloading problem is formulated as a shortest path in a directed acyclic graph problem and solved via the Lagrange relaxation. The proposed algorithm has low computational complexity. However, it considers only one sequence of platooning subtasks, which makes unclear the extension of the algorithm to scenarios with multiple tasks of various types. In [22], the authors consider the price of the used spectrum for task data transmission as a cost function. They propose a convex optimization based task offloading scheme for two scenarios: with independent and cooperative MEC servers. In this scheme, they utilize the knowledge of the vehicle speed and direction to offload tasks with a high data rate when a vehicle is close to an RSU. A similar mobility-aware approach is utilized in [23], where the authors propose a heuristic algorithm to determine the portions of each task that are computed locally at a vehicle, offloaded to another vehicle, or computed at a MEC server. However, for very low latency constraints of the V2X applications, the mobility awareness of the algorithms cannot provide a meaningful cost reduction, since vehicles only slightly change their positions during such short intervals. The study [6] does not use any objective function directly, but it considers both server utilization and delay. Specifically, the authors propose a multi-level MEC structure where higher layers serve tasks that cannot be served by lower layers within the delay constraint. They show that their solution allows improving the utilization of resources with no quality of service degradation. Unfortunately, this study does not pay attention to the computational cost.

All the discussed papers above take into account the delay requirements of tasks. When a task is offloaded to a server, the selected server can be busy computing other tasks. In this case, the task will wait in the queue until the server is able to accept and start computing a new task. However, only few papers [6], [17], [20] consider queuing delays at MEC servers in a problem statement. Specifically, in [17], game theory is used to address the problem. Although various game-theoretic approaches [24, Chapter 4] have been proposed in the literature for vehicular networks, they usually find a solution by computing a Nash equilibrium that is known as PPAD-complete problem and believed to be hard [25]. In [6], [20], the MEC server is modeled as an M/M/1 queuing system. However, in some cases, the assumption that the task computational time has an exponential distribution results in a wrong estimation of the required task computation time and, thus, violation of the delay requirements. Hence, in our paper, we consider a general M/G/1 model, which is valid for any computation time distribution.

Although there are many papers devoted to V2X task offloading, all of them consider short-term task management, i.e., task offloading decision is made for every task generated by a vehicle. However, when the number of vehicles and tasks is large, such an approach can be inefficient because of the long additional delays needed for task offloading decisions. In this paper, we consider long-term task management, which means that the recalculation of mapping between tasks and servers is done only when the traffic intensity significantly changes, e.g., depending on the time of the day, season, etc.

III. PROBLEM STATEMENT

The network is modeled as an undirected graph $G(\mathbb{V}, \mathbb{E})$ (see Fig. 2), where $\mathbb{V}$ is a set of vertices, and $\mathbb{E}$ is a set of edges. Each vertex corresponds to a network node (i.e., a BS, an RSU, or a remote server in an operator's core network or on the Internet), and each edge corresponds to a physical link between two nodes. The connections to remote servers are modeled as virtual links with a given end-to-end latency. The MEC servers are placed in a subset $\mathbb{K} \subset \mathbb{V}$ of RSUs and BSs. For convenience, we list all the notations in Table 3.

Each RSU executes several V2X applications that coordinate trajectories of the connected vehicles. Specifically, using wireless technologies, an RSU collects information about the current state of the connected vehicles (e.g., their coordinates, speeds, road conditions, etc.). Using this information as input data, the RSU generates computational tasks. For example, the RSU can compute trajectories of the vehicles crossing an intersection. Let $\mathbb{F}$ be the set of types of V2X tasks that can be generated at RSUs. When a task is generated at an RSU, this RSU can use its computational resources or offload the task to another node. To offload a task, the RSU transmits a request message with the task input data to the target node (i.e., the node at which the MEC server is installed) over wired links. When the target node finishes computation, it sends back a response message with the computed results. Based on the computed results, the RSU sends recommendations/commands (e.g., change of speed, lane, etc.) to the corresponding vehicles over the wireless channel.

In the paper, we assume that the wireless technologies/protocols employed at RSUs are configured to provide delivery of data with the given delay and reliability. In particular, 5G wireless technologies shall provide Ultra-Reliable and Low-Latency Communications (URLLC) service, i.e., the data shall be delivered within 1 ms with the reliability higher than 99.999%. For the detailed description of the solutions enabling URLLC and aiming at reducing the wireless network energy consumption, the interested reader can refer to [26]–[29].

We assume that the V2X tasks are generated based on the information obtained from a high number of vehicles that work independently of each other. Thus, according to the basic results of queuing theory [30], we model the process of generating tasks at each node (e.g., at RSU) as a Poisson process. Let σ_{if} be the rate of task flow of a type $f \in \mathbb{F}$ generated at a node i. If the node i does not generate tasks of type f, $\sigma_{if} = 0$.

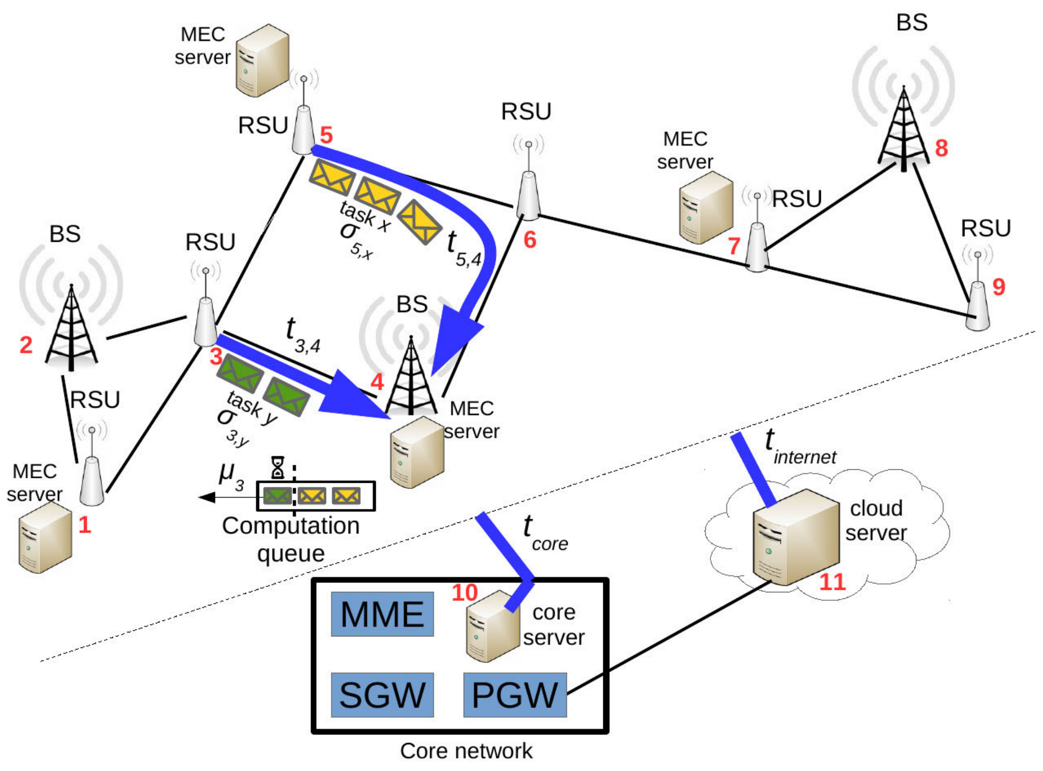


FIGURE 2. Network graph.

The assignment of task flows to servers is expressed by the array $\boldsymbol{Z} = \{z_{ifk}\}$: if a task flow of a type f generated by a node i is executed remotely on a server at node k, the corresponding element $z_{ifk} = 1$. Otherwise, $z_{ifk} = 0$. For convenience, we introduce the vector $\boldsymbol{Y} = \{y_k\}$: if a server at a node k (further server k) is used for task computing, the corresponding element y_k equals one. Otherwise, it equals zero. Variables $\boldsymbol{Z}$ and $\boldsymbol{Y}$ are connected with the following inequalities:

$$y_k \geq z_{ifk}, \quad \forall i \in \mathbb{V}, \ \ \forall f \in \mathbb{F}, \ \ \forall k \in \mathbb{K}. \tag{1}$$

Since the task flows of a type f are computed within some information context, all task flows of this type generated at a node i are computed at the same server.

$$\sum_k z_{ifk} = 1, \quad \forall i \in \mathbb{V}, \ \ \forall f \in \mathbb{F}. \tag{2}$$

The computational time of a task is a random value that depends on the server used for computation. For a server k, the computational time has the average value $1/\mu_k$ and the variance v_k. Note that the total incoming load on server k shall not exceed μ_k in order to avoid server overload.

$$\sum_{i,f} z_{ifk}\sigma_{if} < \mu_k, \quad \forall k \in \mathbb{K}. \tag{3}$$

The delay t_e experienced by a task request/response traversing a link $e \in \mathbb{E}$ consists of:

1) A store-and-forward switching delay, i.e., a delay caused by packet forwarding at PHY and MAC layers at transmitting node;
2) Propagation delay caused by the finite speed of light;
3) Transmission delay with a given data rate.

Note that the store-and-forward delay is the main factor in the metropolitan networks with high-rate links between devices [31]. Therefore, we neglect propagation and transmission delay terms. For core and Internet servers, we assume that the delay on a virtual link equals t_{core} and $t_{internet}$, respectively. In addition, we assume that the amount of control traffic associated with V2X applications is much lower than that of non-V2X traffic. Hence, we can prioritize V2X traffic and do not consider load balancing, i.e., servers and RSUs always use the shortest path for information exchange. We denote the delay on the shortest path as $t_{ik} = \sum_{e \in \mathbb{E}_{ik}} t_e$, where $\mathbb{E}_{ik}$ is the set of links on the shortest path from a node i to a server k.

The total delay d_{ifk} of a task of a type f generated by a node i and offloaded to a server k (if $k = i$, the task is computed without offloading) consists of a two-way offloading delay, a queuing delay at the server and a computational time. Let us denote the sum of queuing delay and computational time, i.e., the sojourn time, as T_k. Then,

$$d_{ifk} = 2\, t_{ik} + T_k.$$

Various V2X application have different QoS requirements. We assume that the delay constraint D_f for a task of a type f is satisfied if the total delay does not exceed it with probability q:

$$P(d_{ifk} < D_f) \geq q. \tag{4}$$

TABLE 3. List of used variables.

Parameter	Meaning
$\mathbb{V}$	set of vertices in network graph
$\mathbb{E}$	set of edges in network graph
$\mathbb{K}$	subset of vertices with deployed servers
$\mathbb{F}$	set of types of tasks
σ_{if}	intensity of task requests of type f generated by node i
$1/\mu_k$	average task computational time at server k
v_k	variance of task computational time at server k
c_k	variation coefficient of task computational time at server k
y_k	indicator $I\{$server k is used for computing$\}$
z_{ifk}	indicator $I\{$task flow of type f generated by node i is computed at server $k\}$
$x_{ifi'f'k}$	auxiliary variables $z_{ifk}z_{i'f'k}$
h_k	total intensity of incoming requests at server k
t_e	end-to-end delay on link e
t_{core}	end-to-end delay from RSU to server in core network
$t_{internet}$	end-to-end delay from RSU to server on the Internet
t_{ik}	delay between node i and server k
$\mathbb{E}_{ik}$	set of links on the shortest path from node i to server k
W_k	queuing delay at server k
T_k	sum of queuing delay and computational time of one task at server k
d_{ifk}	total delay experienced by task of type f generated by node i and offloaded to server k
D_f	delay constraint for task of type f
q	target probability for delay constraint satisfaction
S_k	cost function value for server k
$g(k)$	cost per task for server k
β_k	cost of server k in idle state
γ_k	cost per task of server k in loaded state
α_k	$\gamma_k - \beta_k$

In this paper, we aim at minimizing the ITS operator's expenses subject to delay constraints of applications. Assuming that the expenses corresponding to a particular server depend linearly on a server computational utilization, we introduce the cost function $S_k = \alpha_k h_k/\mu_k + \beta_k$, where $h_k = \sum_{i,f} z_{ifk}\sigma_{if}$ is the incoming request intensity, $\alpha_k > 0$ and $\beta_k > 0$ are given coefficients that depend on server location, performance and other parameters.

Finally, we state the V2X task offloading problem as follows:

$$\begin{cases} \sum\limits_k \left[y_k\beta_k + \frac{\alpha_k}{\mu_k} \sum\limits_{i,f} z_{ifk}\sigma_{i,f} \right] \to \min \\ \text{s.t. } (1)-(4). \end{cases} \tag{5}$$

IV. ALGORITHMS DESCRIPTION

A. COMPUTATIONAL TIME DISTRIBUTION

Since the estimation of the sojourn time T_k depends on task computational time distribution, we consider two cases: (i) exponential distribution, and (ii) general-type distribution, and obtain the closed-form inequalities from delay constraint (4) for each case.

1) EXPONENTIAL CASE

Since tasks are generated at each node according to Poisson process, for the exponentially distributed service time we can consider a server as an M/M/1 queue [30]. Hence, the sojourn time is exponentially distributed with mean [30, Chapter 3]

$$E[T_k] = \frac{1}{\mu_k - h_k} = \frac{1}{\mu_k - \sum\limits_{i',f'} z_{i'f'k}\sigma_{i'f'}}.$$

Since the q-quantile of the exponential distribution $p(x) = \lambda exp[-\lambda x]$ with the mean $1/\lambda$ equals $(1/\lambda)\ln[1/(1-q)]$, the constraint (4) transforms into

$$2t_{ik}z_{ifk} + \frac{1}{\mu_k - \sum\limits_{i',f'} z_{i'f'k}\sigma_{i',f'}} \ln\left[\frac{1}{1-q}\right] \leq D_f, \quad \forall i \in \mathbb{V}, \quad \forall k \in \mathbb{K}, \; \forall f \in \mathbb{F}. \tag{6}$$

2) GENERAL-TYPE CASE

For general-type distribution, we can consider a server as an M/G/1 queue. Hence, the average waiting time, i.e., the amount of time the task is expected to wait in the queue before its service, can be calculated using the Pollaczek–Khinchine formula [30, Chapter 5]:

$$E[W_k] = \frac{h_k}{\mu_k - h_k}\frac{1+c_k^2}{2}\frac{1}{\mu_k},$$

where $c_k^2 = \mu_k^2 v_k$.

According to [32, Chapter 2], the cumulative distribution function of the waiting time $F(x_k)$ is bounded as follows:

$$F(x_k) \geq 1 - e^{-\frac{x_k}{s_k}},$$

where $s_k = E[W_k]/(h_k/\mu_k)$.

Therefore, the upper bound of the q-quantile of the waiting time equals $s_k \ln(1/(1-q))$. To calculate the estimation of the sojourn time q-quantile, we add the average calculation time $1/\mu_k$ to s_k. Then, we obtain estimation of sojourn time q-quantile: $(s_k + 1/\mu_k)\ln(1/(1-q))$. Hence, similar to (6), the delay constraint (4) can be expressed as follows:

$$2t_{ik}z_{ifk} + \left(\frac{1}{\mu_k} + \frac{c_k^2+1}{2(\mu_k - \sum\limits_{i',f'} z_{i'f'k}\sigma_{i'f'})} \right) \cdot \ln\left[\frac{1}{1-q}\right] \leq D_f, \quad \forall i \in \mathbb{V}, \; \forall k \in \mathbb{K}, \; \forall f \in \mathbb{F}. \tag{7}$$

Since the M/M/1 model is a special case of the M/G/1 one, further we consider only M/G/1, unless otherwise explicitly stated. The optimization problem (5) for the M/G/1 model transforms into:

$$\begin{cases} \sum\limits_k \left[y_k\beta_k + \frac{\alpha_k}{\mu_k} \sum\limits_{i,f} z_{ifk}\sigma_{if} \right] \to \min \\ \text{s.t. } (1)-(3), \; (7). \end{cases} \tag{8}$$

B. PROPOSED ALGORITHMS

We introduce two algorithms to solve the problem (8): the Integer Linear Programming (ILP) algorithm and the Cost-effective Heuristic Algorithm for Task offloading (CHAT) based on a greedy approach.

1) INTEGER LINEAR PROGRAMMING

The original optimization problem (8) is non-linear because of the constraint (7). To linearize the problem, we multiply both parts of (7) by $(\mu_k - \sum_{i',f'} z_{i'f'k}\sigma_{i'f'})$. Then, we obtain

$$\begin{aligned}&2\mu_k t_{ik} z_{ifk} - 2t_{ik}\sum_{i',f'} z_{ifk} z_{i'f'k}\sigma_{i'f'} \\ &\quad + D_f \sum_{i',f'} z_{i'f'k}\sigma_{i'f'} \\ &\quad + \ln\left[\frac{1}{1-q}\right]\left(1+\frac{c_k^2+1}{2} - \frac{1}{\mu_k}\sum_{i',f'} z_{i'f'k}\sigma_{i'f'}\right) \\ &\leq \mu_k D_f \quad \forall k \in \mathbb{K},\ \forall i \in \mathbb{V},\ \forall f \in \mathbb{F}.\end{aligned}$$

Let us introduce new auxiliary variables:

$$z_{ifk} z_{i'f'k} = x_{ifi'f'k},$$

where $x_{ifi'f'k}$ are also binary variables.

The new variables shall satisfy the following constraints:

$$x_{ifi'f'k} \leq z_{ifk}, \quad \forall k \in \mathbb{K},\ \forall i, i' \in \mathbb{V},\ \forall f, f' \in \mathbb{F}, \tag{9}$$

$$x_{ifi'f'k} \leq z_{i'f'k}, \quad \forall k \in \mathbb{K},\ \forall i, i' \in \mathbb{V},\ \forall f, f' \in \mathbb{F}, \tag{10}$$

$$x_{ifi'f'k} \geq z_{i'f'k} + z_{ifk} - 1, \quad \forall k \in \mathbb{K},\ \forall i, i' \in \mathbb{V},\ \forall f, f' \in \mathbb{F}. \tag{11}$$

So, the constraint (7) can be converted to the linear form as follows:

$$\begin{aligned}&2\mu_k t_{ik} z_{ifk} - 2t_{ik}\sum_{i',f'} x_{ifi'f'k}\sigma_{i'f'} + D_f \sum_{i',f'} z_{i'f'k}\sigma_{i'f'} \\ &\quad + \ln\left[\frac{1}{1-q}\right]\left(1+\frac{c_k^2+1}{2} - \frac{1}{\mu_k}\sum_{i',f'} z_{i'f'k}\sigma_{i'f'}\right) \\ &\leq \mu_k D_f \quad \forall k \in \mathbb{K},\ \forall i \in \mathbb{V},\ \forall f \in \mathbb{F}.\end{aligned} \tag{12}$$

Finally, the optimization problem (5) transforms into an ILP problem:

$$\begin{cases}\sum_k \left[y_k\beta_k + \frac{\alpha_k}{\mu_k}\sum_{i,f} z_{ifk}\sigma_{if}\right] \to \min, \\ \text{s.t.}\ (1)-(3),\ (9)-(12).\end{cases}$$

The problem can be solved using the classical Branch-and-Cut method [33]. Specifically, in this paper, we use the PuLP Python 3 library [34].

2) CHAT

Since solving an ILP problem is computationally hard, there is a need to design a heuristic algorithm that can solve the problem with less complexity and provide close to optimal results. Consequently, we propose an iterative algorithm, called CHAT, which is based on a greedy approach that selects servers with the minimal cost per task. The pseudocode of the algorithm is presented in Algorithm 1.

For simplicity, in this section, the task flows are presented as pairs (i, f), where i is a network node, and f is a task type. Let us consider a server as *free* when it does not participate in task computing. On each iteration, the algorithm selects one server from the set of free servers and assigns task flows to it. Each iteration consists of the following steps. First, for each free server k, the algorithm constructs a set $\mathbb{S}_k$ of task flows that can be computed on this server without violation of delay constraint. Specifically, the set $\mathbb{S}_k$ is constructed as follows (lines 15–25):

1) The algorithm constructs the set of task flows $\{i, f\}_k$ that can be *separately* assigned to server k without violation of constraints (3) and (7) (see function CONSTRUCT at line 2).
2) If the set $\{i, f\}_k$ is non-empty, then the task flow $(i, f) \in \{i, f\}_k$ that corresponds to the minimal incoming traffic intensity σ_{if} (lines 20–21) is included in the set $\mathbb{S}_k$. After that, the set of task flows $\{i, f\}_k$ is reconstructed (line 23).
3) If the set $\{i, f\}_k$ is empty, then the set $\mathbb{S}_k$ is considered to be constructed, and the procedure stops.

If these sets are empty for all the considered servers, the algorithm assumes that there is no solution for the task offloading optimization problem.

Second, we choose the server k^* with the minimal cost per task among all servers (lines 29–30):

$$k^* = \arg\min g(k) = \arg\min \frac{S_k}{\sum_{(i,f)\in\mathbb{S}_k} z_{ifk}\sigma_{if}}.$$

Finally, we exclude the server k^* from the set of free servers (line 30) and assign the task flows from $\mathbb{S}_{k^*}$ to the server k^* (lines 31–32), and the next iteration starts. The algorithm stops when the number of unassigned task flows reaches zero or until there are no free servers left.

Let us estimate the algorithm's worst-case computational complexity as a function of the network graph size, i.e., the number of RSUs and servers ($|\mathbb{V}|$ and $|\mathbb{K}|$, respectively). In the worst case, the algorithm needs to consider all the servers for task offloading, and therefore the external while loop (lines 14–34) repeats for $|\mathbb{K}|$ times. On each iteration of this while loop, one server is removed from the set $\mathbb{T}$ of unused servers. Hence, the total repetition number of the actions within the inner for loop (lines 15–25) equals $|\mathbb{K}|(|\mathbb{K}| + 1)/2$, which is $O(|\mathbb{K}|^2)$. The complexity of the CONSTRUCT function equals $O(|\mathbb{R}|)$, where $\mathbb{R}$ is the second parameter of the function. Since the cardinality of the sets $\mathbb{M}$ and $\mathbb{U}$ is bounded by $|\mathbb{V}|$, the complexity of the CONSTRUCT functions calls at lines 17 and 23 equals $O(|\mathbb{V}|)$. The number of iterations for the inner while loop (lines 19–24) is bounded by the constant value $(\max_{k\in\mathbb{K}} \mu_k)/(\min_{i\in\mathbb{V}, f\in\mathbb{F}} \sigma_{i,f})$. So, the complexity of the actions into the for loop equals $O(|\mathbb{V}|)$.

Algorithm 1 CHAT Algorithm

```
Input: $G(\mathbb{V}, \mathbb{E})$, $\mathbb{F}$, $\mathbb{K}$, network parameters
Output: $Z, Y$
1: ▷ $k$ - server, $\mathbb{R}$ - set of task flows
2: function construct($k$, $\mathbb{R}$)
3:     $\{i,f\}_k \leftarrow \emptyset$
4:     for all $q \in \mathbb{R}$ do
5:         if $q$ can be assigned to server $k$ then
6:             $\{i,f\}_k \leftarrow \{i,f\}_k + \{q\}$
7:         end if
8:     end for
9:     return $\{i,f\}_k$
10: end function
11:
12: $\mathbb{M} \leftarrow \mathbb{V} \times \mathbb{F}$ ▷ $\mathbb{M}$ - set of unassigned task flows
13: $\mathbb{T} \leftarrow \mathbb{K}$ ▷ $\mathbb{T}$ - set of unused servers
14: while $\mathbb{M} \neq \emptyset$ do
15:     for all $k \in \mathbb{T}$ do
16:         $\mathbb{S}_k \leftarrow \emptyset$
17:         $\{i,f\}_k \leftarrow$ construct($k$, $\mathbb{M}$)
18:         $\mathbb{U} \leftarrow \mathbb{M}$
19:         while $\{i,f\}_k \neq \emptyset$ do
20:             $a \leftarrow (i,f)$ with minimal $\sigma_{if}$
21:             $\mathbb{S}_k \leftarrow \mathbb{S}_k + \{a\}$
22:             $\mathbb{U} \leftarrow \mathbb{U} - \{a\}$
23:             $\{i,f\}_k \leftarrow$ construct($k$, $\mathbb{U}$)
24:         end while
25:     end for
26:     if $\mathbb{S}_k = \emptyset \; \forall k$ then
27:         break ▷ no solution
28:     end if
29:     $k^* \leftarrow \arg\min g(k)$
30:     $y_{k^*} \leftarrow 1$
31:     $z_{ifk^*} \leftarrow 1 \; \forall (i,f) \in \mathbb{S}_{k^*}$
32:     $\mathbb{M} \leftarrow \mathbb{M} - \mathbb{S}_{k^*}$
33:     $\mathbb{T} \leftarrow \mathbb{T} - \{k^*\}$
34: end while
```

Finally, the worst-case computational complexity of the algorithm equals $O(|\mathbb{K}|^2|\mathbb{V}|)$.

V. PERFORMANCE EVALUATION

In this section, we carry out extensive performance evaluation of the proposed algorithms and compare them with the existing algorithms proposed in the literature. Specifically, in Section V-A, we describe the scenarios; in Section V-B, we introduce a cost function; in Section V-C, we describe the baseline algorithms; in Section V-D, we analyze the obtained numerical results.

A. SCENARIOS

We consider two urban scenarios presented in Fig. 3. Black lines correspond to RSU-to-RSU and RSU-to-BS connections, while BSs are not directly connected with each other. The first scenario, shown in Fig. 3(a), forms a regular Manhattan grid. The second scenario, shown in Fig. 3(b), is based on a real map of an area in Moscow, Russia. Positions of BSs are taken from the OpenCellid database [35].

For these scenarios, we consider small scale and large scale cases. Specifically, we consider one small scale and two large scale scenarios:

1) Manhattan small scale scenario with 12 RSUs and 3 BSs;
2) Manhattan large scale scenario with 49 RSUs and 7 BSs;
3) Moscow large scale scenario with 78 RSUs and 32 BSs.

The set $\mathbb{F}$ consists of three task types specified in Section I. Since Table 1 contains ranges for parameter values, to resolve ambiguity, we consider the following parameters for V2X applications:

- Vehicle platooning: $\nu_1 = 40$ Hz, $d_1 = 20$ ms;
- Environment perception: $\nu_2 = 10$ Hz, $d_2 = 3$ ms;
- Intersection safety: $\nu_3 = 50$ Hz, $d_3 = 10$ ms.

Here ν and d correspond to the number of generated tasks per second and the one-way end-to-end latency constraint, respectively. The total delay experienced by a task includes a round-trip delay on a wireless link between a vehicle and an RSU. Assuming that this delay does not exceed $\tau = 4$ ms [27], we calculate the round-trip delay constraints $D_f = 2\,d_f - \tau$. Thus, $D_1 = 36$ ms, $D_2 = 2$ ms, $D_3 = 16$ ms. The end-to-end delay on each link is drawn from the uniform distribution on the interval [200, 500] μs. The delays t_{core} and $t_{internet}$ of virtual links from RSUs to core and Internet servers are set to 5 ms and 10 ms, respectively.

To estimate the number of tasks σ_{if} generated at an RSU per second, we consider a road section that is served by this RSU. We assume that each RSU serves a road section of length $l = 200$ m, whereas all RSUs are placed along the road with constant distance l between them. If the road section has n lanes ($n = 4$ for Manhattan scenario and $n = 6$ for Moscow scenario) and $\xi \in [0, 1]$ is the traffic load, then the task intensity can be estimated as follows:

$$\sigma_{i,f} = I\{f \in \mathbb{F}_i\}\xi n \frac{l}{w} \nu_f,$$

where $I\{f \in \mathbb{F}_i\}$ is the indicator function that returns one if tasks of type f are generated by vehicles on the considered road section, and zero, otherwise. Parameter $w = 10$ m is the effective length of one vehicle (including the guard interval between vehicles). In this paper, we assume that vehicle platooning and environment perception tasks are generated by all vehicles. In contrast, intersection safety tasks are generated only by vehicles on the road sections close to intersections.

B. COST FUNCTION

As an example of a cost function in the paper we consider the energy consumption of the servers. The energy consumption of a server depends on its utilization [36], wherein the energy consumption is non-zero even in an idle state, i.e., when a server does not compute any task. Let the energy consumption

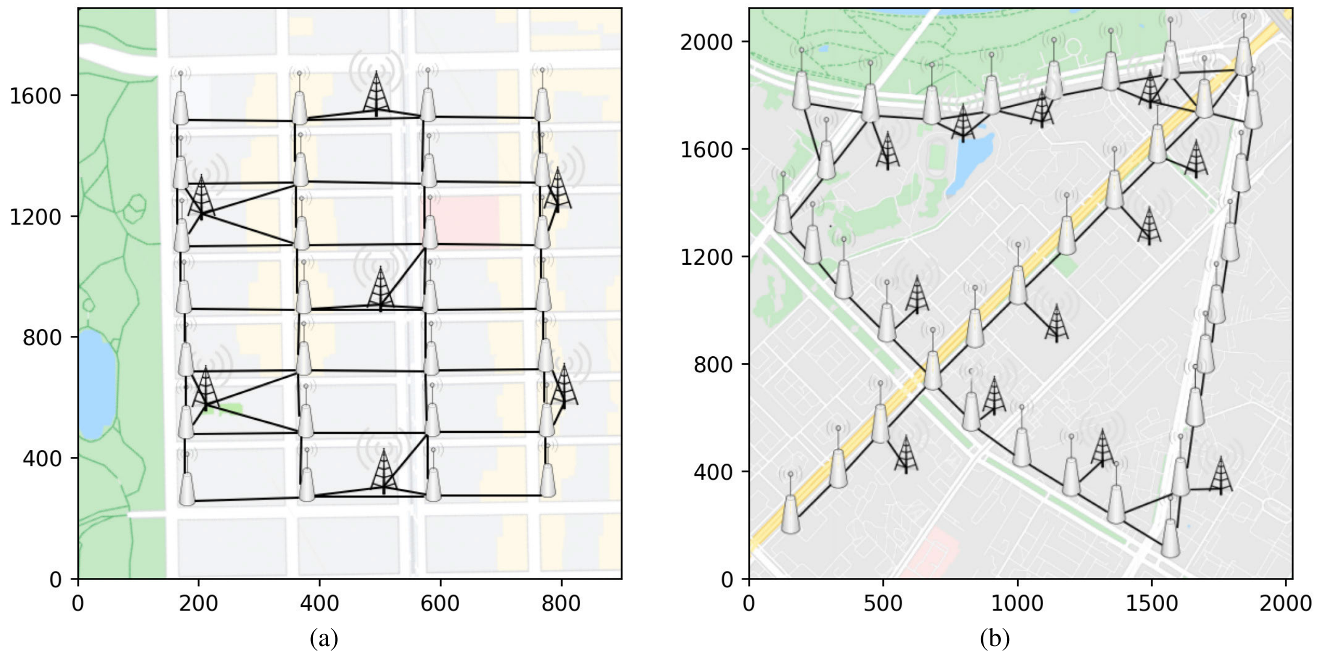


FIGURE 3. Scenarios: (a) Manhattan; (b) Moscow.

of a server k in an idle and a loaded state be β_k and γ_k, respectively. The share of time when a server k is loaded (server utilization) equals h_k/μ_k. Then, the average energy consumption S_k of a server k equals

$$\begin{aligned} S_k &= \gamma_k \frac{h_k}{\mu_k} + \beta_k(1 - \frac{h_k}{\mu_k}) \\ &= \beta_k + (\gamma_k - \beta_k)\frac{h_k}{\mu_k} = \beta_k + \alpha_k \frac{h_k}{\mu_k}, \end{aligned}$$

where $\alpha_k = \gamma_k - \beta_k$.

So, we can see that server energy consumption is a linear function of the server computational utilization in accordance with the assumption in Section III. Each server is characterized by three variables: its performance μ_k and energy consumption (β_k, γ_k) in idle and loaded state, respectively. The computing time for each task is drawn from log-normal distribution with average $1/\mu_k$ and variation coefficient $c_k = \sqrt{v_k}\mu_k = 2$. In all experiments, we consider five types of servers (see Table 4).

TABLE 4. Servers parameters.

Server type	μ_k, tasks/s	β_k, W	γ_k, W
RSU server	23000	5	30
BS server 1	92000	3.5	70
BS server 2	161000	5	100
Core server	230000	5	120
Internet server	690000	7.5	150

Each RSU has its own server that can be used for task computation. Each BS has a server of Type 1 or Type 2 with a probability of 0.25 each, or no server with a probability of 0.5. For the small-scale scenario, we assume that servers are deployed only at RSUs and BSs. In contrast, in the large scale scenarios, we deploy one server in the operator's core network and one server on the Internet.

C. BASELINE ALGORITHMS

In this paper, we consider the following baseline algorithms:

1) the algorithm from [13] that considers tasks in ascending order of delay constraint and selects for offloading the nearest (in terms of delay) servers;
2) the algorithm from [14] that randomly selects a server for each task with considering its delay constraint;
3) the optimization algorithm that minimizes the weighted sum of delay and cost (e.g., from [15]);
4) the Heuristic algorithm from [19] that considers tasks one by one and prioritizes (for task offloading) already selected servers for previously considered tasks.
5) the Hybrid algorithm adapted from [19] that consists of two phases: (i) the CHAT algorithm is used for a preliminary task offloading decision; (ii) the ILP algorithm with additional constraint is used for fine-tuning.

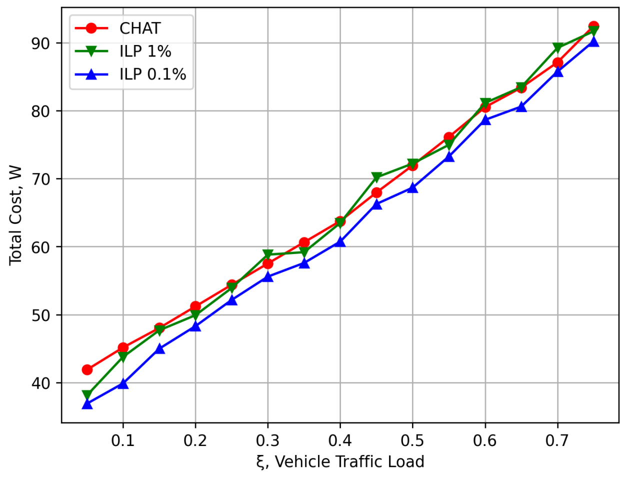
D. ANALYSIS OF THE RESULTS

In this section, we analyze the obtained numerical results and compare the performance of various algorithms. First, we examine the proposed ILP and CHAT algorithms. Since the ILP algorithm has high complexity, we evaluate its performance only in the small scale scenario.

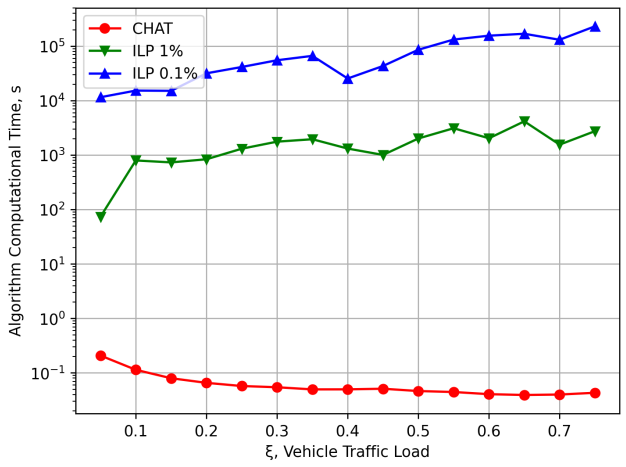
Second, we analyze the computational complexity of the CHAT algorithm and compare it with the worst-case complexity obtained in Section IV-B2.

Third, we investigate the applicability of the M/M/1 model in the case of non-exponential distribution of task computational time. For that, we compare the performance of the ILP and CHAT algorithms with different delay constraints (i.e., equation (6) for the M/M/1 model and (7) for the M/G/1 model) in the small scale scenario.

Finally, we compare the CHAT algorithm with the baseline algorithms from related papers in all scenarios.



(a)



(b)

FIGURE 4. CHAT algorithm validation in the Manhattan small scale scenario.

1) CHAT VALIDATION

Figure 4 shows the results obtained in the Manhattan small scale scenario for two algorithms: CHAT and ILP with different accuracy settings, 0.1% and 1%. According to Fig. 4(a), the total cost value linearly increases with traffic load. The cost provided by the CHAT algorithm is almost the same as for the ILP algorithm. Fig. 4(b) shows that the CHAT algorithm has multiple orders of magnitude less complexity than the ILP algorithm for both accuracy settings. By decreasing the accuracy from 0.1% to 1%, we can reduce the computational time by 1…2 orders, but the accuracy affects the total cost. Specifically, the ILP algorithm with 1% accuracy provides even higher cost value than the CHAT algorithm for some ξ values, which makes further accuracy decrease meaningless.

2) ALGORITHM COMPUTATIONAL TIME EVALUATION

In Section IV-B2 we show that worst-case computational complexity of the CHAT algorithm is $O(|\mathbb{K}|^2|\mathbb{V}|)$. Let us analyze the complexity of the algorithm in a linear scenario where all RSUs and BSs are equally spaced along a straight line. For this scenario, the number of servers $|\mathbb{K}|$ is proportional to the number of RSUs $|\mathbb{V}|$. Hence, the computational time dependency on the number of RSUs in this scenario is expected to be approximately cubic. Fig. 5 shows that a cubic function can approximate the computational time of CHAT.

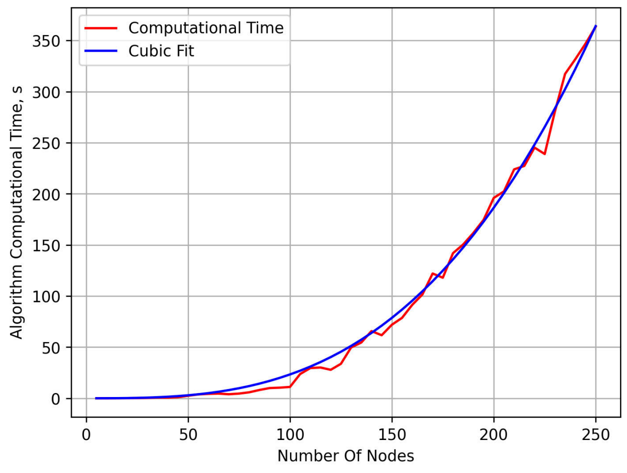
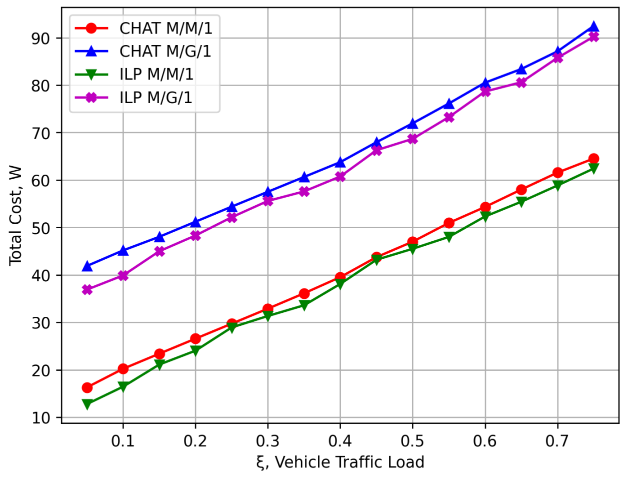


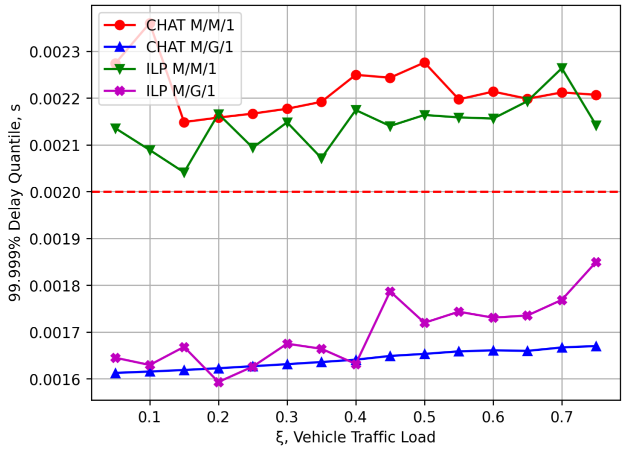
FIGURE 5. Algorithm computational time and its cubic fit.

3) APPLICABILITY OF THE M/M/1 MODEL

In this section, we investigate the applicability of the M/M/1 model in scenarios with non-exponential task computational time distribution. For that, we apply equation (6) for the M/M/1 model and equation (7) for the M/G/1 model, and consider the proposed ILP and CHAT algorithms. Using the task offloading solution obtained with the algorithms, we run simulations to estimate the delay quantile. Although Fig. 6(a) shows, that the algorithms with the M/M/1 model provide lower total cost, they violate the delay constraint (see Fig. 6(b)). Hence, the M/M/1 model is inapplicable in the considered scenario because it underestimates task execution time at the server. We conduct experiments with different task computing distributions and conclude that the accuracy of the M/M/1 model significantly depends on the distribution, while the M/G/1 model is valid for all the considered distributions.

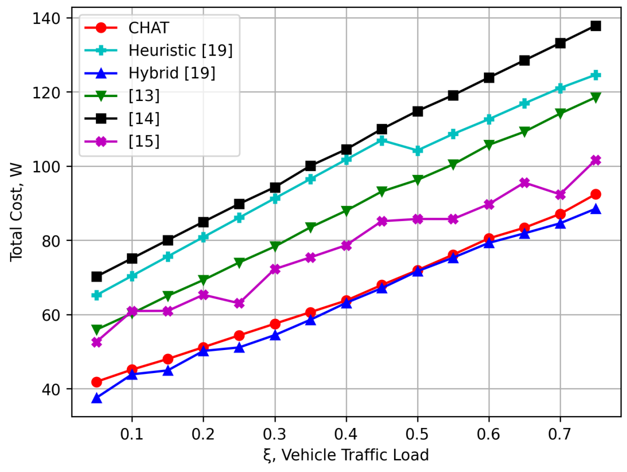


(a)

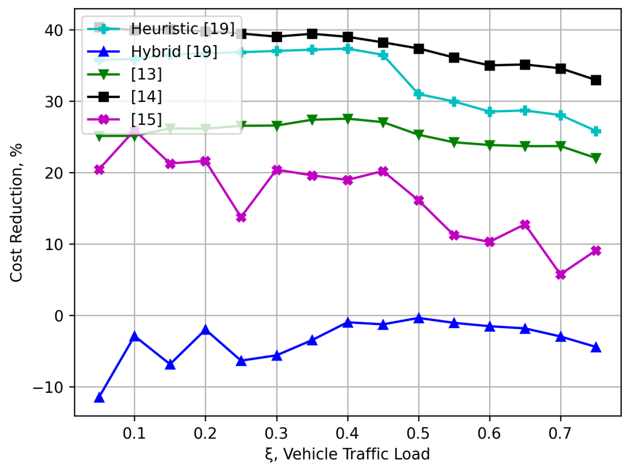


(b)

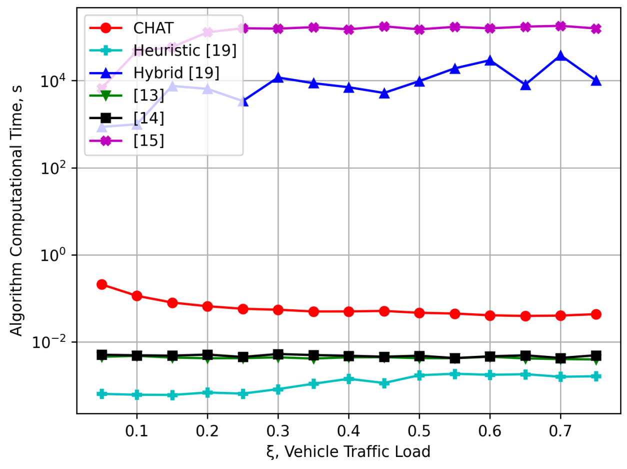
FIGURE 6. Applicability of the M/M/1 model in the Manhattan small scale scenario.



(a)



(b)



(c)

FIGURE 7. Evaluation of algorithms in the Manhattan small scale scenario.

4) PERFORMANCE EVALUATION OF CHAT

Let us compare the performance of the CHAT algorithm with the baseline algorithms described in Section V-C. We provide results for three scenarios, one Manhattan small scale scenario and two large scale scenarios (Manhattan and Moscow).

a: SMALL SCALE SCENARIO

First, we compare the algorithms in the Manhattan small scale scenario. Fig. 7(a) shows that the CHAT and the Hybrid algorithms reduce the operator's expenses compared with other algorithms. However, according to Fig. 7(c), the computational complexity of the Hybrid algorithm is several orders of magnitude higher than the complexity of the CHAT algorithm. Note that the total costs provided by the algorithms from [15] and [19] are non-monotonic functions of the traffic load, while the other curves are monotonic. Specifically, the cyan curve corresponding to the algorithm from [19] consists of two linear parts and a transition part near $\xi = 0.5$ because by design the algorithm prefers to compute all tasks using servers at RSUs and at $\xi \sim 0.5$, the algorithm starts using other servers, which are more efficient. The non-monotonic behavior of the magenta curve corresponding to

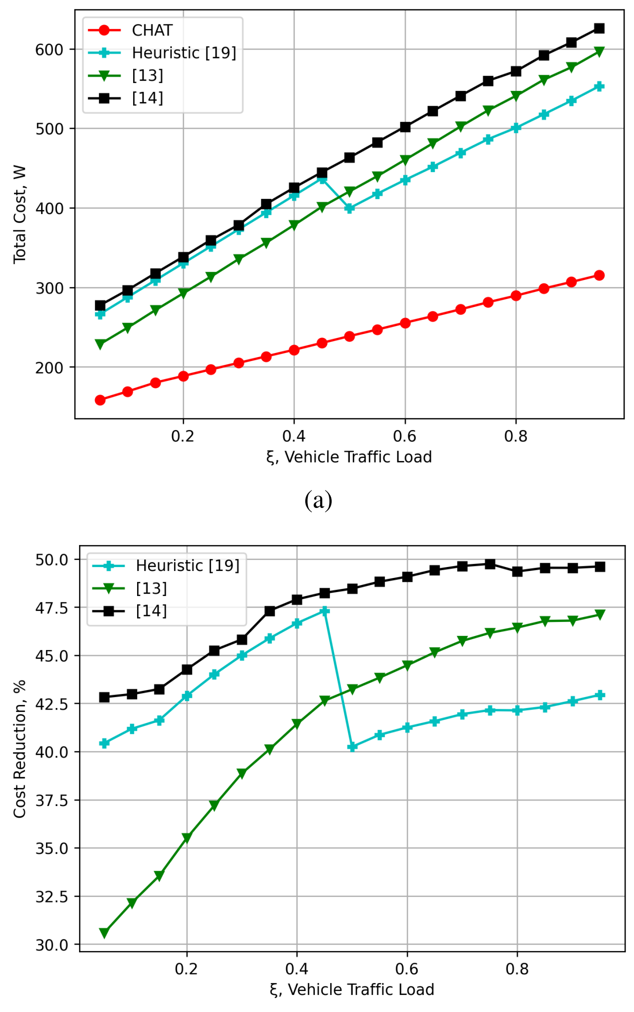


FIGURE 8. **Evaluation of algorithms in the Manhattan large scale scenario.**

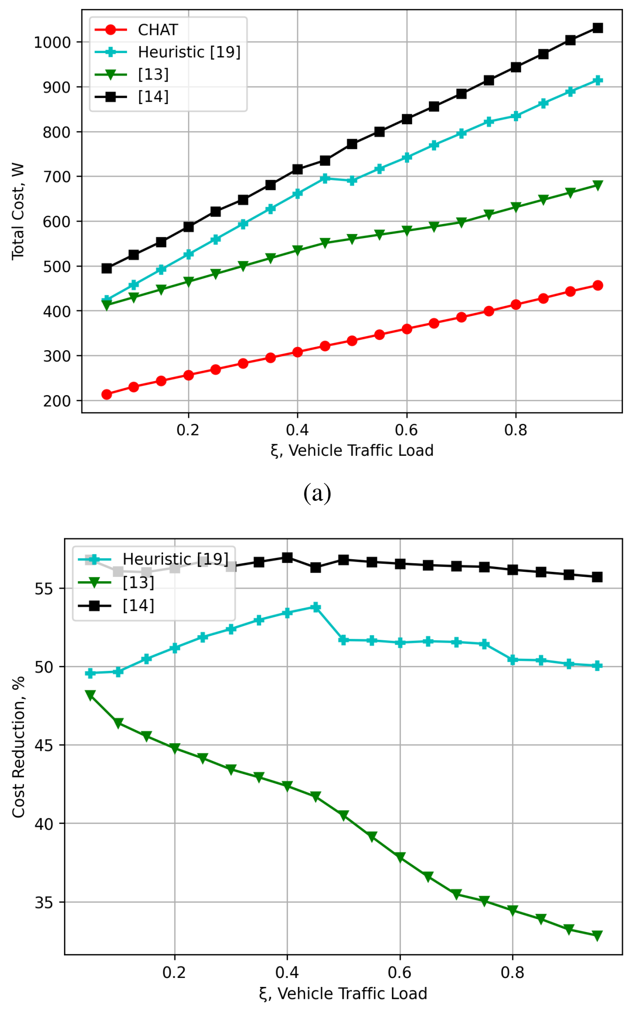


FIGURE 9. **Evaluation of algorithms in the Moscow large scale scenario.**

the algorithm from [15] is explained by the choice of the cost function that includes the average delay, which results in the prioritization of energy-inefficient servers that are deployed close to the road. This algorithm provides the least cost among all the baseline algorithms from the literature. However, it is still $10-20\%$ higher than for the CHAT, and its computational complexity is even higher than the complexity of the Hybrid algorithm. Hence, we do not consider this algorithm as well as the Hybrid algorithm for comparison in the large scale scenarios. The next best alternative, i.e., the algorithm from [13], provides $\sim 25\%$ higher cost than the CHAT algorithm.

The algorithm from [14] shows the worst performance, i.e., up to 40% higher cost than the CHAT algorithm because of the random selection of servers.

b: LARGE SCALE SCENARIOS

Finally, we compare the algorithms in more complex scenarios. Figs. 8 and 9 show the results obtained in the Manhattan and Moscow large scale scenarios, respectively. By the same reason as in the discussed above Manhattan small scale scenario, the cyan curve corresponding to the algorithm from [19] shows non-monotonic behavior at $\xi \sim 0.5$. Note that for high vehicle traffic load values, the algorithm from [13] performs much better in the Moscow scenario than in the Manhattan scenario. This effect follows from the topological differences between these scenarios. In particular, in the Moscow scenario, the average number of hops between RSUs and BSs is large, which increases the utilization of more cost-effective servers in the core network and on the Internet. Nevertheless, the CHAT algorithm reduces the cost by $\sim 40\%$ compared with the algorithm from [13] and more than twice compared with the algorithm [14] that selects servers randomly.

Assuming that a big city (e.g., Moscow) has approximately $N = 3 \cdot 10^5$ vehicles on roads every day, we can recalculate the cost reduction ΔS in terms of the annual city energy consumption: $\Delta S \cdot N \cdot 365 \cdot 24/N_{scenario}$, where $N_{scenario}$

is the number of vehicles in the considered scenario. For both scenarios, the CHAT algorithm reduces the city energy consumption by up to 10^6 kWh per year. So, our algorithm significantly cuts off the operator's energy expenses while satisfying V2X applications requirements.

VI. CONCLUSION

In the paper, we have formulated the V2X task offloading problem as a non-linear optimization problem aimed at minimizing a cost function subject to delay constraints of various V2X applications. In contrast to other studies, we consider queuing delays at servers, which can have general-type distribution. We have developed two algorithms to solve the problem: (i) the ILP algorithm based on integer linear programming and (ii) the CHAT algorithm based on a greedy approach. The obtained results show that the CHAT algorithm provides close to the optimal results and has several orders lower computational complexity than the ILP algorithm.

We have compared the proposed CHAT algorithm with other algorithms proposed in the literature by considering the total energy consumption of servers as the cost function. The numerical results demonstrate that the proposed algorithm reduces energy consumption up to several times while satisfying heterogeneous requirements of V2X applications in all the considered scenarios. Thus, the application of the proposed algorithm in practice can significantly reduce ITS operator expenses.

We see the following directions for future research. First, we are going to consider other cost functions, both linear and non-linear ones. For example, we plan to take into account (i) leasing expenses of ITS operator when third-party computational resources are used, (ii) energy consumption of vehicles spent both for local task computation and transmission of tasks over the wireless channel. Second, we will extend our communication model by considering the distribution of time and reliability of delivering a computational task from a vehicle to RSU and back over the wireless channel.

REFERENCES

[1] Z. MacHardy, A. Khan, K. Obana, and S. Iwashina, "V2X access technologies: Regulation, research, and remaining challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1858–1877, 3rd Quart., 2018.

[2] A. B. Reis, S. Sargento, and O. K. Tonguz, "On the performance of sparse vehicular networks with road side units," in *Proc. IEEE 73rd Veh. Technol. Conf. (VTC Spring)*, May 2011, pp. 1–5.

[3] *Study on Enhancement of 3GPP Support for 5G V2X Services*, document TR 22.886, V16.2.0, 3GPP, Dec. 2018.

[4] *Multi-Access Edge Computing (MEC); Study on MEC Support for V2X Use Cases*, document GR MEC 022, V2.1.1, ETSI, Sep. 2018.

[5] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the Internet of vehicles," *Proc. IEEE*, vol. 108, no. 2, pp. 246–261, Feb. 2020.

[6] M. Khayyat, A. Alshahrani, S. Alharbi, I. Elgendy, A. Paramonov, and A. Koucheryavy, "Multilevel service-provisioning-based autonomous vehicle applications," *Sustainability*, vol. 12, no. 6, p. 2497, Mar. 2020.

[7] F. Giust, V. Sciancalepore, D. Sabella, M. C. Filippou, S. Mangiante, W. Featherstone, and D. Munaretto, "Multi-access edge computing: The driver behind the wheel of 5G-connected cars," *IEEE Commun. Standards Mag.*, vol. 2, no. 3, pp. 66–73, Sep. 2018.

[8] Y. Wang, J. Wang, Y. Ge, B. Yu, C. Li, and L. Li, "MEC support for C-V2X system architecture," in *Proc. IEEE 19th Int. Conf. Commun. Technol. (ICCT)*, Oct. 2019, pp. 1375–1379.

[9] J. Liu, J. Wan, B. Zeng, Q. Wang, H. Song, and M. Qiu, "A scalable and quick-response software defined vehicular network assisted by mobile edge computing," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 94–100, Jul. 2017.

[10] S. Zhou, P. P. Netalkar, Y. Chang, Y. Xu, and J. Chao, "The MEC-based architecture design for low-latency and fast hand-off vehicular networking," in *Proc. IEEE 88th Veh. Technol. Conf. (VTC-Fall)*, Aug. 2018, pp. 1–7.

[11] C. Storck and F. Duarte-Figueiredo, "A 5G V2X ecosystem providing Internet of vehicles," *Sensors*, vol. 19, no. 3, p. 550, Jan. 2019.

[12] F. Zhou, R. Q. Hu, Z. Li, and Y. Wang, "Mobile edge computing in unmanned aerial vehicle networks," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 140–146, Feb. 2020.

[13] A. Moubayed, A. Shami, P. Heidari, A. Larabi, and R. Brunner, "Edge-enabled V2X service placement for intelligent transportation systems," *IEEE Trans. Mobile Comput.*, early access, Jan. 10, 2020, doi: 10.1109/TMC.2020.2965929.

[14] H. Guo, J. Zhang, and J. Liu, "FiWi-enhanced vehicular edge computing networks: Collaborative task offloading," *IEEE Veh. Technol. Mag.*, vol. 14, no. 1, pp. 45–53, Mar. 2019.

[15] J. Wang, D. Feng, S. Zhang, J. Tang, and T. Q. S. Quek, "Computation offloading for mobile edge computing enabled vehicular networks," *IEEE Access*, vol. 7, pp. 62624–62632, 2019.

[16] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2633–2645, Aug. 2018.

[17] M. Deng, H. Tian, and X. Lyu, "Adaptive sequential offloading game for multi-cell mobile edge computing," in *Proc. 23rd Int. Conf. Telecommun. (ICT)*, May 2016, pp. 1–5.

[18] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[19] F. Tonini, B. Khorsandi, E. Amato, and C. Raffaelli, "Scalable edge computing deployment for reliable service provisioning in vehicular networks," *J. Sens. Actuator Netw.*, vol. 8, no. 4, p. 51, Oct. 2019.

[20] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive offloading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.

[21] X. Fan, T. Cui, C. Cao, Q. Chen, and K. S. Kwak, "Minimum-cost offloading for collaborative task execution of MEC-assisted platooning," *Sensors*, vol. 19, no. 4, p. 847, Feb. 2019.

[22] C. Yang, Y. Liu, X. Chen, W. Zhong, and S. Xie, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, vol. 7, pp. 26652–26664, 2019.

[23] S. Raza, W. Liu, M. Ahmed, M. R. Anwar, M. A. Mirza, Q. Sun, and S. Wang, "An efficient task offloading scheme in vehicular edge computing," *J. Cloud Comput.*, vol. 9, no. 1, pp. 1–14, Dec. 2020.

[24] Y. Zhang and M. Guizani, *Game Theory for Wireless Communications and Networking*. Boca Raton, FL, USA: CRC Press, 2011.

[25] M. Wooldridge, *An Introduction to Multiagent Systems*. Hoboken, NJ, USA: Wiley, 2009.

[26] H. Yang, K. Zheng, K. Zhang, J. Mei, and Y. Qian, "Ultra-reliable and low-latency communications for connected vehicles: Challenges and solutions," *IEEE Netw.*, vol. 34, no. 3, pp. 92–100, May 2020.

[27] K. Lee, J. Kim, Y. Park, H. Wang, and D. Hong, "Latency of cellular-based V2X: Perspectives on TTI-proportional latency and TTI-independent latency," *IEEE Access*, vol. 5, pp. 15800–15809, 2017.

[28] E. Khorov, A. Krasilov, I. Selnitsky, and I. Akyildiz, "A framework to maximize the capacity of 5G systems for ultra-reliable low-latency communications," *IEEE Trans. Mobile Comput.*, early access, Feb. 24, 2020, doi: 10.1109/TMC.2020.2976055.

[29] C. Singhal and S. De, *Resource Allocation in Next-Generation Broadband Wireless Access Networks*. Hershey, PA, USA: IGI Global, 2017.

[30] L. Kleinrock, *Queueing Systems: Theory*, vol. 1. New York, NY, USA: Wiley, 1975.

[31] S. M. Rumble, D. Ongaro, R. Stutsman, M. Rosenblum, and J. K. Ousterhout, "It's time for low latency," in *Proc. 13th USENIX Conf. Hot Topics Oper. Syst. (HotOS)*. Berkeley, CA, USA: USENIX Association, 2011, pp. 1–5.

[32] L. Kleinrock, *Queueing Systems: Computer Applications*, vol. 2. New York, NY, USA: Wiley, 1976.

[33] J. E. Mitchell, ''Branch-and-cut algorithms for combinatorial optimization problems,'' in *Handbook of Applied Optimization*. Oxford, U.K.: Oxford Univ. Press, 2002, pp. 65–77.

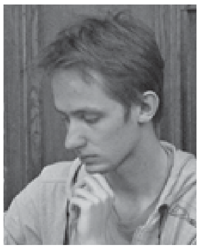
[34] S. Mitchell, M. Osullivan, and I. Dunning, ''PuLP: A linear programming toolkit for Python,'' *Optim. Online*, pp. 1–12, Sep. 2011. [Online]. Available: http://www.optimization-online.org/DB_FILE/2011/09/3178.pdf

[35] *OpenCellid—Largest Open Database of Cell Towers & Geolocation—By Unwired Labs*. Accessed: May 2020. [Online]. Available: https://opencellid.org

[36] L. A. Barroso and U. Hölzle, ''The case for energy-proportional computing,'' *Computer*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

ANDREY BELOGAEV received the B.S. degree in applied mathematics and physics and the M.S. degree in telecommunications from the Moscow Institute of Physics and Technology, in 2014 and 2016, respectively. He is currently pursuing the Ph.D. degree in telecommunications with the Moscow Institute of Physics and Technology.

He has been a Researcher with the Network Protocols Research Laboratory and Wireless Networks Lab (both Institute for Information Transmission of the Russian Academy of Sciences, Moscow, Russia) since 2014 and 2018, respectively. His research interests include enabling novel URLLC (including V2X-related) applications in 5G, control information dissemination, and overhead reduction in wireless networks, QoS/QoE provisioning.



ALEXEY ELOKHIN received the B.S. degree in physics from Moscow State University, in 2019. He is currently pursuing the master's degree with the Computer Science Department, Higher School of Economics, HSE Moscow Institute of Electronics and Mathematics, Moscow, Russia. He is currently a Junior Researcher with the Telecommunication Systems Laboratory, HSE Moscow Institute of Electronics and Mathematics.

ARTEM KRASILOV received the M.S. and Ph.D. degrees in telecommunications from the Moscow Institute of Physics and Technology, in 2010 and 2013, respectively.

He is currently a Senior Researcher with the Institute for Information Transmission Problems, Russian Academy of Sciences. He has been involved in various national and international research projects supported by both academic foundations (FP7 programme, Russian Government) and industrial partners. His professional interests are related to SDN and NFV-based architectures for the next-generation wireless networks, 5G, QoE/QoS provisioning in wireless networks, development of analytical and simulation tools for performance evaluation, and optimization of various wireless access technologies. He has coauthored over 30 research papers.

EVGENY KHOROV (Senior Member, IEEE) is currently the Head of the Wireless Networks Laboratory, Institute for Information Transmission Problems, Russian Academy of Sciences. He has led dozens of national and international projects sponsored by academia funds and industry. Being a voting member of the IEEE 802.11, he has contributed to the 802.11ax standard as well as to real-time applications TIG with many proposals. He has authored more than 100 articles. His main research interests are related to 5G and beyond wireless systems, next-generation Wi-Fi, protocol design, and QoS-aware cross-layer optimization. He was a recipient of the Russian Government Award in Science and Technology, several Best Papers Awards, and the Scopus Award Russia 2018. In 2015, 2017, and 2018, Huawei RRC awarded him as the Best Cooperation Project Leader. He gives tutorials and participates in panels at large IEEE events. He chairs TPC of the IEEE GLOBECOM 2018 CA5GS Workshop and IEEE BLACKSEACOM 2019. He was awarded as the Editor of the Year 2020 of Ad Hoc Networks.

IAN F. AKYILDIZ (Fellow, IEEE) has been the Megagrant Research Leader of the Institute for Information Transmission Problems, Russian Academy of Sciences, Russia, since 2018. He is the Ken Byers Chair Professor in Telecommunications with the School of Electrical and Computer Engineering, the Director of the Broadband Wireless Networking Laboratory, and the Chair of the Telecommunication Group, Georgia Institute of Technology, Atlanta, GA, USA. He received numerous awards from the IEEE and ACM, and many other organizations. His H-index is 125, and the total number of citations is above 119K as per Google scholar as of July 2020. His current research interests are in 5G wireless systems, nanonetworks, Terahertz band communications, and wireless sensor networks in challenging environments. He is a Fellow of the ACM (1997).