Sustainable Vehicular Edge Computing Using Local and Solar-Powered Roadside Unit Resources

Yu-Jen Ku, Sujit Dey
Mobile Systems Design Laboratory
Department of Electrical and Computer Engineering
University of California, San Diego, CA, USA
Email:{yuku, dey}@ucsd.edu

Abstract— **In this paper, we explore a sustainable solution to growing vehicular computing and communication needs by utilizing edge computing and communication resources of a network of Solar-powered Roadside Units (SRSUs), along with any vehicular computing resources available. The solution ensures no additional grid energy expended while minimizing any QoS loss for the vehicle users (VUs). An SRSU consists of a small cell base station (SBS) and a road-edge computing (REC) node, is powered by a low-cost solar system. VUs can offload their vehicular application tasks to SRSUs to receive high throughput and low latency services. However, the limited capacity of solar energy, REC computing, and bandwidth resources may cause service disruption and affect the Quality of Service (QoS) that VUs receive. To minimize such QoS loss, we formulate a dynamic offloading QoS loss minimization problem, where the different subtasks of a VU application are optimally executed either locally using the VU computing resource or remotely using the REC resource, considering the energy, computing and bandwidth constraints of the SRSU network. We then propose to solve it by a heuristic algorithm which jointly makes the optimal user association, subtask offloading, and SRSU resource allocation decisions. To evaluate the proposed algorithm, we build a simulation framework consisting of a dense SRSU network using real-world solar energy generation and urban vehicular traffic data. The simulation results show that our proposed approach can significantly reduce QoS loss compared to other best effort strategies.**

Keywords— Solar Energy, Computation Offloading, Quality of Service, Roadside Unit

I. INTRODUCTION

Evolving vehicles have witnessed a rapidly growing need for computing and communication to fulfill the emerging demands of compute-intensive and time-sensitive Adaptive Driver Assistance Systems (ADAS) functionalities, along with multi-media entertainment functions. The reliance on vehicular computing and communication will accelerate more with the realization of level 4 and level 5 self-driving applications [1]. A promising solution to support this growing need is using Roadside Units (RSU), with a RSU consisting of a small cell base station (SBS) and a road-edge computing (REC) server. REC servers can be more powerful than each individual vehicle's local computing resource. By offloading compute-intensive vehicular user (VU) applications like object detection and collision prediction to RSUs, VU can receive better service quality with safer and more improved driving experience.

However, dense deployment of RSUs to support VUs will significantly worsen the carbon footprint and energy consumption of the cellular networks. Therefore, in our previous work [2], we proposed the use of Solar-powered RSU (SRSU), which consists of SBS, REC server, and self-sustained solar system.

Note that in SRSU, the generated solar energy is limited and fluctuating. If solar energy cannot meet SRSU's power demand, SRSU will reduce its capacity of computing and communication. Some of the VUs thus cannot offload their applications to the REC server due to the lack of computing and communication resources at the SRSU. This leads to Quality of Service (QoS) loss, in terms of service outage and disruption, for vehicular applications. Furthermore, the problem becomes more difficult for the vehicular network, where the location and density of VUs are frequently changing.

In our previous work [2], we attempted to address the loss of QoS in SRSU network by optimally scheduling the use of solar energy with the help of an additional battery at the SRSU, and make VUs' SRSU association (i.e., user association) as well as resource allocation decisions in advance. We showed with extensive simulations that for realistic solar energy generation and vehicular traffic scenarios, the proposed algorithm could significantly reduce the QoS loss.

However, the proposed algorithm requires accurate predictions of vehicular application offloading demand and solar energy generation profiles to optimize QoS loss, both of which can be difficult to realize. Moreover, it does not consider the opportunity posed by modern vehicles to have spare computing capacity, which can also be used to meet the overall vehicular computing needs to minimize any QoS loss. In this paper, we consider an online optimization problem, where the VU task offloading, user association, and resource allocation decisions are made in real-time with current VU application requests and solar energy generation profile. To keep SRSU design complexity and cost low, we assume there is no battery in SRSU. Hence, SRSU's energy availability is limited by the solar energy harvested currently, and the current decisions made will not affect the optimality of the future.

We assume that each VU has some computation resource which can be supplemented with REC resource to execute the VU application. In this way, SRSU's computation and communication resources can be utilized optimally according to their solar energy generation variability to serve more VUs and further reduce QoS loss. We consider a dynamic offloading

978-1-7281-1220-6/19/$31.00 ©2019 IEEE

scenario, where different subtasks of an application can be chosen to be either executed locally by VU computing resource or offloaded to the REC server of the associated SRSU.

To minimize QoS loss, which will be defined in detail in Section III, we aim to find the optimal decisions of user association, the set of offloaded subtasks, and SRSU's communication and computing resource allocation at any given time. The decision is made by a network controller of the SRSU network and considers current SRSU resource availabilities, channel conditions, VU locations, and the offloading requests (including requirements to complete the application).

The rest of this paper is organized as follows. We will review related work in Section II. Section III describes the system model and formulates the problem. Section IV introduces our methodology for QoS loss minimization. We will present experiment results in Section V and conclude our work in Section VI.

II. Related Work

Computation offloading challenges in energy constrained cellular networks have been previously addressed. Mao *et al*. [3] and Xu *et al*. [4] address the challenges of utilizing Renewable Energy (RE) for computation task offloading. However, they focus on a single user scenario. In our work, computation tasks from multiple VU users have to be jointly considered and offloaded to adequate SRSUs.

In [5], the authors minimize the number of computation tasks and downlink data traffics that are dropped by RE-powered SBSs. However, the considered computation task and downlink data traffic in [5] are independent. In this paper, we consider a more challenging scenario where a task has both requirements on computation and data transmission, which share a joint delay constraint. Moreover, the above work cannot be used in scenarios when both user (VU) and SBS (REC) are capable of executing computation tasks.

There is a large amount of research on techniques of computation offloading considering subtask offloading and scheduling, as well as collaboration between edge computing sever and local computing resource. For example, in [6], [7], and [8], users can locally execute part of the task, and in the meantime, compete for the limited computation resources in the edge server to execute the rest of the task. They model a task by sequentially or parallelly dependent subtasks. They propose task partitioning and scheduling algorithms to minimize the energy consumption of users [7][8] or computation delay of tasks [6][7]. However, the above work did not consider energy and communication resource constraints at the edge server. To our best knowledge, we are the first to consider not only the collaboration of local computing and REC server, but the challenge of utilizing limited communication, computing, and energy resources of SRSU.

III. System Model and Problem Formulation

In this section, we will first introduce our system model. Then we define QoS loss and formulate the QoS loss minimization problem.

A. Network and Channel Model

We consider an SRSU network with a set of SRSUs $\boldsymbol{\mathcal{B}}$. Each SRSU has a communication module SBS and a computation module REC server. We use SBS b and REC b to represent the SBS and REC server in SRSU $b \in \boldsymbol{\mathcal{B}}$, respectively. Each SBS b has $K_{D,b}$ downlink subcarriers and $K_{U,b}$ uplink subcarriers. Each REC b has U_b computing capacity, which is in terms of the number of machine instructions the processors of REC b can execute in one time slot. The total operation time is equally divided into T time slots, with duration Δ. Let $\boldsymbol{I}^t = \{1,2,\dots,\ell\}$ be the set of VUs in the network at the t^{th} time slot, where $\ell = |\boldsymbol{I}^t|$ is the number of VUs. We denote the location of VU $i \in \boldsymbol{I}^t$ as a_i^t and its local computing capacity as U_i.

We denote the transmit power of SBS and VU as p_B and p_I, respectively. For simplicity, we assume that the channel gains of all the subcarriers are the same. Let $\eta_{D,bi}^t$ be the signal-to-interference-noise ratio (SINR) of downlink transmission from SBS b to VU i. $\eta_{D,bi}^t$ is given by,

$$\eta_{D,bi}^t = \frac{p_B\, g_{bi}^t}{N_0 + \sum_{b' \neq b} p_B g_{b'i}^t}\,, \tag{1}$$

where g_{bi}^t is the channel gain and N_0 is the noise level. Let $r_{D,bi}^t$ be the achievable downlink transmission rate from SBS b to VU i per subcarrier,

$$r_{D,bi}^t = W\, log_2(1 + \eta_{D,bi}^t)\,, \tag{2}$$

where W is the bandwidth per subcarrier. Similarly, the uplink transmission rate from VU i to SBS b per subcarrier is,

$$r_{U,bi}^t = W\, log_2(1 + \frac{p_I g_{bi}^t}{N_0})\,, \tag{3}$$

where the interference from other VUs is negligible with frequency reuse and bandwidth allocation techniques [9].

B. Vehicular Application Task Model

As discussed before, the evolution of vehicles is leading to compute-intensive vehicular applications, like ADAS or autonomous driving applications (e.g., object detection and tracking, emerging driver guidance and warning applications). At each time slot, every VU will request to offload a (application) task to the REC server. We denote d_i^t as the maximum tolerable delay of the task. We assume that the task consists of M sequential subtasks, as shown in Fig. 1. Let $\boldsymbol{M} = \{1,2,\dots M\}$ be the set of all subtasks. For each subtask $m \in \boldsymbol{M}$, we assume ω_{mi}^t is the input data size and $\omega_{(m+1)i}^t$ is the output data size. Note that $\omega_{(m+1)i}^t$ will also be the input data size of subtask $m+1$. We also assume that c_{mi}^t is the computing resource required to complete subtask m, which is quantized as the number of machine instructions. Let $\boldsymbol{y}_i^t \subset \boldsymbol{M}$ denotes the *offloading set*, which is the set of subtasks that will be offloaded from VU i at time slot t. Note that there are 2^M possibilities of $\boldsymbol{y}_i^t$. $\boldsymbol{y}_i^t$ will affect the amount of computing resource that SRSU needs to provide and the size of data which need to be transmitted between VU and SRSU. For example, for subtasks $m \in \boldsymbol{y}_i^t$, and $m-1, m+1 \notin \boldsymbol{y}_i^t$, the input of m (with data size ω_{mi}^t) needs to be uploaded to REC after $m-1$ is executed by VU. Also, the output of m (with data size $\omega_{(m+1)i}^t$) needs to be downlink transmitted to VU after m is executed. For each $\boldsymbol{y}_i^t$, we denote $\varpi_i^t(\boldsymbol{y}_i^t)$ as the size of data which need to be uploaded to SRSU, $C_i^t(\boldsymbol{y}_i^t)$ as the required machine instructions, and $\delta_i^t(\boldsymbol{y}_i^t)$ as the size of data which needs to be downlink transmitted to VU i.

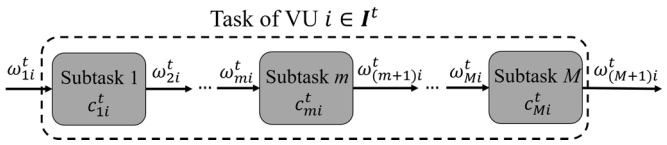

Fig. 1. Breakdown of a vehicular application task.

We denote the user association decision at the t^{th} time slot as a binary matrix $\boldsymbol{X}^t = \{x_{bi}^t\}_{b\in\boldsymbol{B},i\in\boldsymbol{I}^t}$, where $x_{bi}^t = 1$ if VU i is associated with SRSU b (VU i can thus offload its subtasks to SRSU b), and $x_{bi}^t = 0$ otherwise. At each time slot, we assume each VU can only offload subtasks to one SRSU. REC server, though, can serve tasks from different VUs. Also, note that the task of a VU i cannot be offloaded to multiple SRSUs at the same time slot.

C. SRSU Resource and Utilization

To satisfy the delay requirement the offloaded tasks, SRSU needs to allocate adequate amounts of computing and communication resources to the associated VUs. At the t^{th} time slot, let u_{bi}^t be the computing capacity, which is quantized as instructions per second, of the virtual machine server created for VU i by REC b. Let $k_{U,bi}^t$ and $k_{D,bi}^t$ respectively be the number of uplink and downlink subcarriers allocated to VU i by SBS b. For simplicity, we assume that u_{bi}^t, $k_{U,bi}^t$ and $k_{D,bi}^t$ are reserved and active for VU i within the whole time slot. Therefore, for VU i, the allocation of the above resources should satisfy the following,

$$\sum_{b\in\boldsymbol{B}} x_{bi}^t \left(\frac{\varpi_i^t(\boldsymbol{y}_i^t)}{r_{U,bi}^t k_{U,bi}^t} + \frac{C_i^t(\boldsymbol{y}_i^t)}{u_{bi}^t} + \frac{\delta_i^t(\boldsymbol{y}_i^t)}{r_{D,bi}^t k_{D,bi}^t} \right) \le \sum_{b\in\boldsymbol{B}} x_{bi}^t d_i^t(\boldsymbol{y}_i^t), \quad (4)$$

where $d_i^t(\boldsymbol{y}_i^t) = d_i^t - \frac{\sum_{m\in\boldsymbol{M}} c_{mi}^t - C_i^t(y_i^t)}{U_i}$ is the remaining tolerable delay for the offloaded subtasks after deducting local execution delay.

Also, the computing and communication resources of each SRSU are limited and constrained by,

$$\sum_{i\in\boldsymbol{I}^t} x_{bi}^t u_{bi}^t \le U_b, \quad (5)$$

$$\sum_{i\in\boldsymbol{I}^t} x_{bi}^t k_{U,bi}^t \le K_{U,b}, \quad (6)$$

$$\sum_{i\in\boldsymbol{I}^t} x_{bi}^t k_{D,bi}^t \le K_{D,b}. \quad (7)$$

D. Power consumption of SRSU

The power consumption of each SRSU consists of the power consumption of its REC and SBS. At the t^{th} time slot, let $P_{S,b}^t$ be the power consumption of REC b. $P_{S,b}^t$ linearly increases with the utilized processor's computing capacity [10]. We denote $p_{M,b}$ as the idle power of REC b and $p_{C,b}$ is the power consumption when REC b is fully utilized. $P_{S,b}^t$ can then be represented by the following equation,

$$P_{S,b}^t = p_{M,b}\left(1 - \frac{1}{U_b}\right) + \frac{p_{C,b}}{U_b} \sum_{i\in\boldsymbol{I}^t} x_{bi}^t u_{bi}^t. \quad (8)$$

For SBS, the power consumption of uplink transmission is the circuit power for demodulation and baseband processing. It increases linearly with the number of active subcarriers [11]. Secondly, operating downlink transmission consumes circuit and RF related power; both are linearly increasing with the number of active downlink subcarriers [12]. Hence, the power consumption of SBS can be expressed as:

$$P_{X,b}^t = \sum_{i\in\boldsymbol{I}^t} x_{bi}^t \left(p_{D,b} k_{D,bi}^t + p_{U,b} k_{U,bi}^t \right) + p_{N,b}, \quad (9)$$

where $p_{N,b}$ is the idle power of SBS b, $p_{U,b}$ is the circuit power consumption per active uplink subcarrier, and $p_{D,b}$ is the circuit and transmission power consumption per active downlink subcarrier. Note that $p_{D,b}$ needs to consider the efficiency of the power amplifier. The overall power consumption of SRSU b at the t^{th} time slot is, therefore, represented as: $P_b^t = P_{S,b}^t + P_{X,b}^t$.

At the t^{th} time slot, let S_b^t be the amount of energy harvested from the solar panel of SRSU b. The power consumption of SRSU b thus needs to satisfy the following, $S_b^t \ge P_b^t$.

E. QoS Model

The evaluation of QoS loss of VU in this paper is defined in terms of the occurrence of service outage, service disruption, and *local computing ratio*.

1) Service Outage

Because the energy, computing, and communication resources are limited, SRSU may not be able to serve a VU in a way that satisfies its task delay requirement (4). Service outage happens when a task cannot be completed by neither SRSUs nor local computing resource. We denote the number of such tasks at the t^{th} time slot as L_d^t,

$$L_d^t = \sum_{i\in\boldsymbol{I}^t} \left(1 - \sum_{b\in\boldsymbol{B}} x_{bi}^t\right). \quad (10)$$

2) Service Disruption

Service disruption happens to a VU task when it is being handed over to another SRSU by currently associated SRSU. The handover can take place when a VU is leaving a SRSU's coverage or when its associated SRSU has to be actively changed due to computing or communication resource constraints. VU can only offload its task to new SRSU after the handover is completed, leading to service disruption. We denote the number of such tasks at the t^{th} time slot as L_h^t,

$$L_h^t = \sum_{i\in\boldsymbol{I}^t} \left(\sum_{b\in\boldsymbol{B}} x_{bi}^t\right)\left(1 - \sum_{b\in\boldsymbol{B}} x_{bi}^t x_{bi}^{t-1}\right). \quad (11)$$

3) Local Computing Ratio

Note that local computation resource is more unstable and less powerful than a dedicated REC server. In each VU, there may be other onboard applications (e.g., multimedia entertainment functions) sharing the local computing resource, leading to larger-than-expect local execution delay for the task. Therefore, to reduce such risk, REC needs to execute as many subtasks as possible. For VU i, we denote *local computing ratio* as the ratio of locally executed machine instructions to the overall required machine instructions of its task.

$$L_l^t = \sum_{i\in\boldsymbol{I}^t} \left(\sum_{b\in\boldsymbol{B}} x_{bi}^t\right)\left(1 - \frac{C_i^t(\boldsymbol{y}_i^t)}{\sum_{m\in\boldsymbol{M}} c_{mi}^t}\right). \quad (12)$$

The above three cases have different impacts on VU. In the first case, VU will be left unserved during the whole time slot. In the second case, if the duration of handover disruption is small, VU can still be served for the rest of the time slot. Finally, in the third case, the application task of VU can be served for the whole time slot with the risk that delay constraint maybe violated. Therefore, we introduce weighted factors κ and σ on L_h^t and L_l^t, respectively. The QoS loss of the network is thus,

$$\mathcal{L}^t = L_d^t + \kappa L_h^t + \sigma L_l^t. \quad (13)$$

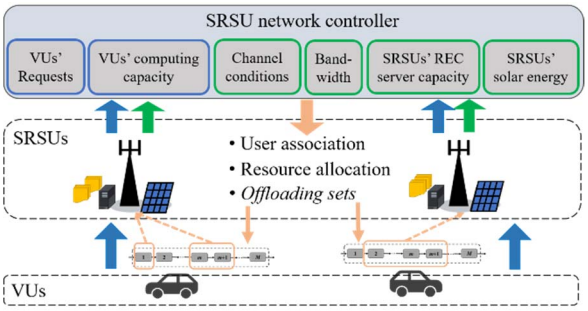

Fig. 2. Overview of proposed SRSU-assisted vehicular computing eco-system, including request and decision flows.

F. Problem Formulation

We assume at the beginning of a time slot, each VU will send an offloading request, which includes information of subtask parameters and available local computing capacity, to the SRSU it associates with. We assume a central SRSU network controller will take the above information and make user association, *offloading set* (for dynamic offloading), and resource allocation decisions. The decisions will be sent to SRSUs and each SRSU will inform their associated VUs the offloading sets. Fig. 2 depicts the whole process. In Fig. 2, blue arrows show the flow of task offloading requests and local computing capacity information (listed in blue boxes) from VU to the network controller; green arrows show the flow of channel, bandwidth, solar energy, and REC computing availability (listed in green boxes) information from SRSU to the network controller; orange arrows indicate the flow of decisions from the network controller to SRSU and VU.

The objective of the SRSU network controller is to determine for VU i the user association x_{bi}^t, *offloading set* decision $\boldsymbol{y}_i^t$, and the resource allocation u_{bi}^t, $k_{U,bi}^t$, $k_{D,bi}^t$, to minimize the QoS loss for each time slot t. The decision is made at the beginning of the time slot based on the current solar energy generation, the channel conditions, VUs' current association, location, task offloading demand, and local computing resource.

The optimization problem for the t^{th} time slot can, therefore, be formulated as

$$\mathbf{P1}: \min_{x_{bi}^t, \boldsymbol{y}_i^t, k_{U,bi}^t, k_{D,bi}^t, u_{bi}^t \, \forall i \in \boldsymbol{I}^t} \mathcal{L}^t$$

$$s.t. \ (4), (5), (6), (7)$$

$$\sum_{b\in\boldsymbol{\mathcal{B}}} x_{bi}^t \le 1, \quad \forall i \in \boldsymbol{I}^t, \tag{14}$$

$$x_{bi}^t = \{0,1\}, \quad \forall i \in \boldsymbol{I}^t, \tag{15}$$

$$S_b^t \ge P_b^t, \quad \forall b \in \boldsymbol{\mathcal{B}}, \tag{16}$$

$$k_{U,bi}^t, k_{D,bi}^t, u_{bi}^t \in \mathbb{Z}^+ \cup \{0\}, \forall i \in \boldsymbol{I}^t, \forall b \in \boldsymbol{\mathcal{B}}, \tag{17}$$

$$\sum_{b\in\boldsymbol{\mathcal{B}}} x_{bi}^t \eta_{D,bi}^t \ge \sum_{b\in\boldsymbol{\mathcal{B}}} x_{bi}^t \eta_{th}, \forall i \in \boldsymbol{I}^t. \tag{18}$$

Constraint (14) together with (15) avoid the task to be offloaded to multiple SRSUs simultaneously. (16) is the energy consumption constraint of each SRSU. (17) states that resource allocation should be 0 or positive integer. Moreover, constraint (18) limits VUs to associate only with the SRSU that provides enough downlink SINR, with the threshold being set by η_{th}.

IV. Solution Methodology

P1 is a nonlinear integer programming problem and therefore is NP-hard [13]. We propose to solve **P1** by investigating its dual problem. We will calculate the dual cost for each possible VU-SRSU association and *offloading set*. The dual cost will be used in a heuristic algorithm to determine the priority of VU and its optimal *offloading set* when associating VU with any SRSU.

We start by rewriting the objective function $\mathcal{L}^t$,

$$\mathcal{L}^t = L_d^t + \kappa L_h^t + \sigma L_l^t$$
$$= \ell + \sum_{i\in\boldsymbol{I}^t}\sum_{b\in\boldsymbol{\mathcal{B}}} \left(-1 + \kappa + \sigma - \kappa\Omega\left(x_{bi}^t, x_{bi}^{t-1}\right) - \frac{\sigma C_i^t(\boldsymbol{y}_i^t)}{\sum_{m\in\boldsymbol{M}} c_{mi}^t}\right) x_{bi}^t, \tag{19}$$

where $\Omega(x_{bi}^t, x_{bi}^{t-1})$ returns 1 if current user association indicator x_{bi}^t equals the VU i 's previous user association indicator, or otherwise returns 0. $\ell = |\boldsymbol{I}^t|$ is the number of VUs. Let $z_{bi}^t(\boldsymbol{y}_i^t) = -1 + \kappa + \sigma - \kappa\,\Omega\left(x_{bi}^t, x_{bi}^{t-1}\right) - \frac{\sigma C_i^t(\boldsymbol{y}_i^t)}{\sum_{m\in\boldsymbol{M}} c_{mi}^t}$, the dual problem of **P1** can then be formulated as,

$$\mathbf{P1_{LD}}: \max_{\lambda_b^t,\mu_b^t,\theta_b^t,\rho_b^t\in\mathbb{R}_+, b\in\boldsymbol{\mathcal{B}}} \ \min_{x_{bi}^t, \boldsymbol{y}_i^t, b\in\boldsymbol{\mathcal{B}}, i\in\boldsymbol{I}^t} \sum_{i\in\boldsymbol{I}^t}\sum_{b\in\boldsymbol{\mathcal{B}}} z_{bi}^t(\boldsymbol{y}_i^t) * x_{bi}^t$$
$$+ \sum_{b\in\boldsymbol{\mathcal{B}}} \lambda_b^t\left(\sum_{i\in\boldsymbol{I}^t} x_{bi}^t u_{bi}^t - U_b\right) + \sum_{b\in\boldsymbol{\mathcal{B}}} \mu_b^t\left(\sum_{i\in\boldsymbol{I}^t} x_{bi}^t k_{U,bi}^t - K_{U,b}\right)$$
$$+ \sum_{b\in\boldsymbol{\mathcal{B}}} \theta_b^t\left(\sum_{i\in\boldsymbol{I}^t} x_{bi}^t k_{D,bi}^t - K_{D,b}\right) + \sum_{b\in\boldsymbol{\mathcal{B}}} \rho_b^t\left(\sum_{i\in\boldsymbol{I}^t} x_{bi}^t P_b^t - S_b^t\right)$$
$$s.t. \ (4), (14), (15), (17), (18).$$

where λ_b^t, μ_b^t, θ_b^t and ρ_b^t are the Lagrangian multipliers for dualizing constraints (5)-(7) and (16). The optimality of $\mathbf{P1_{LD}}$ (as for **P1**) depends on the values of λ_b^t, μ_b^t, θ_b^t, and ρ_b^t. For simplicity, we update the values of λ_b^t, μ_b^t, θ_b^t, and ρ_b^t for the t^{th} time slot as following,

$$\lambda_b^t = \tau\frac{\sum_{i\in\boldsymbol{I}^{t-1}} x_{bi}^{t-1}u_{bi}^{t-1}}{(U_b)^2}, \mu_b^t = \tau\frac{\sum_{i\in\boldsymbol{I}^{t-1}} x_{bi}^{t-1}k_{U,bi}^{t-1}}{(K_{U,b})^2},$$
$$\theta_b^t = \tau\frac{\sum_{i\in\boldsymbol{I}^{t-1}} x_{bi}^{t-1}k_{D,bi}^{t-1}}{(K_{D,b})^2}, \rho_b^t = \tau\frac{P_b^{t-1}}{(S_b^{t-1})^2}, \tag{20}$$

where τ is a scaling factor. The reason is to make a larger cost for assigning a VU to the SRSU whose resources are more likely to be fully utilized from the observation of the last time slot.

Let $\boldsymbol{Y}$ be the set of all possibilities of the subtask set $\boldsymbol{y}_i^t$ that will be offloaded to REC server at time i, and n be the index of the n^{th} possible $\boldsymbol{y}_i^t$ in $\boldsymbol{Y}$. Solving $\mathbf{P1_{LD}}$ is then equivalent to finding the SRSU b and the *offloading set* $\boldsymbol{y}_i^t$ which minimize $q_{b,i,n}^t = z_{bi}^t(n) + \lambda_b^t u_{bi}^t + \mu_b^t k_{U,bi}^t + \theta_b^t k_{D,bi}^t + \rho_b^t P_b^t$ for each VU i. Since u_{bi}^t, $k_{U,bi}^t$, and $k_{D,bi}^t$ are also decision variables in **P1**, we propose to decide the values of these variables by solving the delay constraint (4) with the following equation,

$$\frac{u_{bi}^t}{U_b} = \frac{k_{U,bi}^t}{K_{U,b}} = \frac{k_{D,bi}^t}{K_{D,b}}. \tag{21}$$

Note that u_{bi}^t, $k_{U,bi}^t$, and $k_{D,bi}^t$ need to satisfy SRSU resource capacity and energy constraints (5)-(7), and (16). If these constraints are violated, we set $q_{b,i,n}^t$ to be ϵ, which is an arbitrarily large number.

Based on the above discussion, we propose the Dynamic offloading User association and Resource allocation for QoS loss minimization (DURQ) algorithm, with its pseudo-code shown below. First, we initialize at line 3 a B-by-ℓ-by-υ matrix $\boldsymbol{Q}^t \leftarrow \{q_{b,i,n}^t\}_{b\in\boldsymbol{\mathcal{B}}, i\in\boldsymbol{I}^t, n\in\boldsymbol{Y}}$ using the definition of $q_{b,i,n}^t$ as we discussed above. Note that B is the number of SRSUs in $\boldsymbol{\mathcal{B}}$, υ is the size of $\boldsymbol{Y}$. Then we start to pick a VU for user association. For every VU which has not been associated with any SRSU,

Fig. 3. A neighborhood in Brooklyn, NY and SRSU deployment studied in this paper.

we find the best *offloading set* index n^* and SRSU b^* that corresponds to the minimum $q^t_{b,i,n}$ for this VU. Second, we find the *offloading set* index n' and the SRSU $b' \neq b^*$ which provides the second best value of $q^t_{b,i,n}$ for this VU. Among yet to be associated VUs, we pick the VU which has the largest difference between its best and second best $q^t_{b,i,n}$ and try to associate it to SRSU b^*. As shown in lines 5-9 of Algorithm DURQ. During the association, the uplink, downlink, and computing resources allocated to the VU are calculated by using constraint (4) and Eq. (21). If constraints (5)-(7), (16), and (18) cannot be satisfied by the calculated resource allocation decision, we will set $q^t_{b^*,i,n^*}$ to ϵ. Then we restart from the beginning of the VU picking step (line 5). The above steps are executed iteratively in a *whole* loop from line 4 to line 19 until all the VUs are associated or $\boldsymbol{Q}^t \leftarrow \{\epsilon\}_{b\in\boldsymbol{\mathcal{B}},i\in\boldsymbol{I}^t,n\in\boldsymbol{Y}}$. Finally, DURQ will return the heuristic decisions for user association, *offloading sets*, and SRSU resource allocation.

Assume υ is the size of all possible subtask offloading sets, ℓ is the number of VUs at current time slot, and B is the number of SRSUs. Lines 5-8 has time complexity O($\upsilon\ell B$), which is the most complicated step in the *while* loop. For the worst case, the *while* loop needs to be executed $\upsilon\ell B$ times. Therefore, the complexity of DURQ is O($\upsilon^2\ell^2B^2$). We will show in the next section that the complexity of DURQ is small enough to be executed in real-time for realistic scenarios.

Algorithm DURQ (Dynamic offloading User Association and Resource Allocation for QoS loss minimization)

Inputs:
1) The solar energy generation $S^t_b,\ \forall\ b \in \boldsymbol{\mathcal{B}}$
2) VU computing capacity $\{U_i\}$, task $\{\omega^t_{mi}, c^t_{mi}, d^t_i, m \in [1, M]\}\ \forall i \in \boldsymbol{I}^t$
3) VU location $\{a^t_i\}$ and Channel conditions $\{g^t_{bi} |\ i \in \boldsymbol{I}^t, b \in \boldsymbol{\mathcal{B}}\}$
4) System Parameters η_{th}, $K_{D,b}$, $K_{U,b}$, and $U_b, \forall b \in \boldsymbol{\mathcal{B}}$
5) Previous association status $x^{t-1}_{bi}, \forall i \in \boldsymbol{I}^t, \forall b \in \boldsymbol{\mathcal{B}}$
7) Lagrangian multipliers λ^t_b, μ^t_b, θ^t_b and ρ^t_b, , $\forall b \in \boldsymbol{\mathcal{B}}$
7) An arbitrarily large number ϵ

Output:
1) User association $\boldsymbol{X}^t$,
2) Resource allocation $\{u^t_{bi}, k^t_{U,bi}, k^t_{D,bi}\}\ \forall i \in \boldsymbol{I}^t, \forall b \in \boldsymbol{\mathcal{B}}$
3) Offloading sets $\boldsymbol{y}^t_i\ \forall i \in \boldsymbol{I}^t$

1: **Initialization:**
2: *visit* ← 0
3: $\boldsymbol{Q}^t \leftarrow \{q^t_{b,i,n}\}_{b\in\boldsymbol{\mathcal{B}},i\in\boldsymbol{I}^t,n\in\boldsymbol{Y}}$ by the definition of q^t_{bin}
4: **while** *visit* $\leq \ell$ && $\exists \boldsymbol{Q}^t_{b,i,n} \neq \epsilon$ **do**
5: **for** $\forall\ i \in \boldsymbol{I}^t$ **do**
6: $\{b^*_i, n^*_i\} \leftarrow argmin_{b\in\boldsymbol{\mathcal{B}},n\in\boldsymbol{Y}}\ \boldsymbol{Q}^t_{b,i,n}$
7: $\{b'_i, n'_i\} \leftarrow argmin_{\ b\in\boldsymbol{\mathcal{B}}\setminus\{b^*_i\},n\in\boldsymbol{Y}}\ \boldsymbol{Q}^t_{b,i,n}$
8: **end for**
9: $i^* \leftarrow \max_i \boldsymbol{Q}^t_{b'_i,i,n'_i} - \boldsymbol{Q}^t_{b^*_i,i,n^*_i}$, $b^* \leftarrow b^*_{i^*}$, $n^* = n^*_{i^*}$
10: **calculate** $u^t_{b^*i^*}, k^t_{U,b^*i^*}, k^t_{D,b^*i^*}$ by using (4) and (21)
11: **if** $u^t_{b^*i^*}, k^t_{U,b^*i^*}, k^t_{D,b^*i^*}$ satisfy constraint (5)-(7),(16),and (18)
12: $x^t_{b^*i^*} \leftarrow 1$, *visit* ← *visit*+1
13: $\boldsymbol{y}^t_{i^*} \leftarrow$ *offloading set according to* n^*
14: $\boldsymbol{Q}^t_{b,i^*,n} \leftarrow \epsilon, \forall b \in \boldsymbol{\mathcal{B}}, \forall n \in \boldsymbol{Y}$
15: $\{u^t_{bi}, k^t_{U,bi}, k^t_{D,bi}\}_{i=i^*,b=b^*} \leftarrow \{u^t_{b^*i^*}, k^t_{U,b^*i^*}, k^t_{D,b^*i^*}\}$
16: **else**
17: $\boldsymbol{Q}^t_{b^*,i^*,n^*} \leftarrow \epsilon$
18: **endif**
19: **end while**
20: **return** $\boldsymbol{X}^t, \{u^t_{bi}, k^t_{U,bi}, k^t_{D,bi}\}\ \forall i \in \boldsymbol{I}^t, \forall b \in \boldsymbol{\mathcal{B}}$, and $\boldsymbol{y}^t_i\ \forall i \in \boldsymbol{I}^t$

V. Experimental Results

We will first introduce our simulation framework and parameters. Then, we will present the performance comparison of DURQ with another two best effort techniques and an algorithm without dynamic offloading.

A. Simulation Framework

Our simulation framework is MATLAB-based. The objective is to evaluate the QoS loss performance of different strategies for user association, *offloading set* decision, and resource allocation with real-world scenarios. We consider a 450*450 (meters) rectangular area in Brooklyn, New York City, as shown in Fig. 3, to simulate realistic VU movement and topology. We use historical vehicular traffic data in this area collected by the New York State Department of Transportation [14]. The dense deployment of 13 SRSUs used in the network is also shown by the dark triangles in Fig. 3.

The duration of each time slot is set as 1 second. We set $\kappa = 0.1$ because the duration of the handover process in LTE-A can be less than 100 ms. Moreover, we assume the probability distribution that a subtask is disrupted by other local vehicular applications is uniform between 0 and 1; hence we set σ as its expected value 0.5. Total simulation time is 7 hours, starting from 10 AM to 5 PM. The QoS loss performance shown in this section is calculated by averaging the QoS loss over each VU and over the 7-hour operation time.

The subcarriers are allocated to VUs in groups, and each group has 12 subcarriers (i.e., 180 kHz/group). Multiple groups of subcarriers can be allocated to the same VU simultaneously. We assume each SRSU can utilize 460 subcarrier groups concurrently for each direction of transmission. We assume N_0 is -174 dBm/Hz and the efficiency of power amplifier is 0.15. We list the parameters of the SBS power model in Table I.

We assume each SRSU is equipped with 5 Nvidia Jetson Tx 2 modules [15] as REC server. The total computing speed U_b is 104150 MIPS and the power parameters are, $p_{M,b}$ (idle power) = 20W, and $p_{C,b}$ (power consumption when fully utilized) = 75W. For each VU, the local computing capacity U_i follows a uniform distribution from 500 to 5000 (MIPS).

For solar energy generation profile, we use the data collected at multiple sites in UC San Diego [16]. We normalize the solar energy data and assume the solar panel size is 1.6 m^2 for each SRSU.

At the beginning of each time slot, VUs enter at both ends of each street following a Poisson distribution with rate Θ. Each VU travels with predetermined routes and speed so that the average traffic volume of each street satisfies the historical data.

We consider VU's task as an object detection application, which consists of 2 sequential subtasks, namely video encoding as subtask 1 and object detection as subtask 2. We assume the

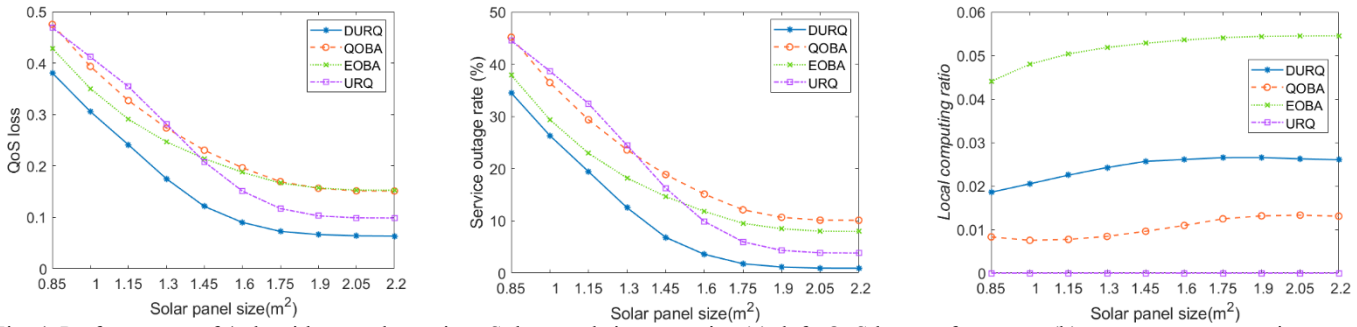

Fig. 4. Performances of 4 algorithms under various Solar panel size scenarios (a). left, QoS loss performance, (b) center, average service outage rate per VU, and (c) right, average *local computing ratio*.

delay requirement d_i^t is 0.1s for each VU i. Noting different vehicles may have cameras with different video resolutions, we consider two different video resolutions in our simulation. We assume there is a 0.75 probability that a VU uses 640 x 480 (pixels) raw video frames (ω_{1i}^t = 922KB/frame) as the input of subtask 1. In this case, subtask 1 will require c_{1i}^t = 59.35 Million Instructions Per Frame (MIPF) and output an H.264 Blu-ray format (ω_{2i}^t =28KB/frame) encoded video [17]. In the meantime, subtask 2 will require c_{2i}^t = 680.4 MIPF [18] and output a ω_{3i}^t = 5KB/frame results. On the other hand, VU will have 0.25 probability to use 1280 x 720 (pixels) raw video frames (ω_{1i}^t = 2.76MB/frame) as the input of subtask 1. In this case, c_{1i}^t =190 MIPF, ω_{2i}^t = 84KB/frame, c_{2i}^t = 2041 MIPF, and ω_{3i}^t = 10KB/frame.

To demonstrate how DURQ optimize QoS loss through user association, we compare DURQ with the following two best effort user association algorithms. These algorithms associate VUs with the SRSU which provides the best signal strength. The first algorithm, denoted as QoS loss Oriented Best Effort User Association algorithm (QOBA), uses the same $q_{b,i,n}^t$ definition as DURQ. For each SRSU b, it greedily chooses the VU and *offloading sets* $\boldsymbol{y}_i^t$ which correspond to the minimum $q_{b,i,n}^t$. Then QOBA uses constraint (4) and Eq. (21) to compute SRSU resource allocation. SRSU b keeps associating with the VUs until the capacity constraints (5)-(7) or energy constraint (16) can no longer be satisfied.

The second algorithm is Energy Oriented Best Effort User Association algorithm (EOBA). It uses the same way as QOBA to allocate communication and computing resources for VU. Each SRSU will greedily choose the VU and its *offloading set* that correspond to the minimum energy consumption. Similarly, SRSU stops associating with new VU when the capacity and energy constraints (5)-(7), (16) cannot be satisfied.

To evaluate the benefit of utilizing local VU computing resource besides REC resource, we also compare with a User Association and Resource Allocation for QoS loss minimization (URQ) algorithm. URQ works the same as DURQ, except it asks VU to offload all its subtasks to the REC server. Namely, $\boldsymbol{y}_i^t$ can only be {1,2, ... ,M} for URQ.

TABLE I: KEY PARAMETERS IN SIMULATION FRAMEWORK

Parameter	Value
$p_{U,b}$ (uplink circuit power/subcarrier)	0.0067 W
$p_{D,b}$ (downlink circuit power/subcarrier)	0.14 W
$p_{N,b}$ (SBS idle power)	10 W
η_{th} (SINR threshold)	0 dB
p_B (SRSU transmit power density)	30 dBm/10MHz
p_I (VU transmit power density)	23 dBm/10MHz
Parameter	Value
g_{bi}^t	Path loss: 128.1 + 37.6log10(R), R is the distance between SRSU and VU (kilometers) Shadowing: 8 dB

B. *Simulation Results*

We implement DURQ in Matlab on a computer which has a 4 GHz CPU. For the above simulation settings, the run-time is 330 ms for a time slot in the worst case. Therefore, DURQ can be executed in real-time.

Next, we present QoS loss performance comparison of our proposed algorithm with URQ, QOBA, and EOBA under different solar panel sizes, and REC computing capacities.

Fig. 4(a) shows the QoS loss performance of the above 4 algorithms under various solar panel sizes. The solar panel size affects the solar energy availability of SRSU. With the original size of the solar panel (1.6 m^2), the QoS loss of DURQ is 40%, 54%, and 52% lower than URQ, QOBA, and EOBA, respectively. DURQ is always better than the other techniques, but its performance stops improving after solar panel size exceeds 2 m^2 because the bottleneck becomes the REC capacity.

Fig. 4(b) shows the corresponding service outage rate per VU, which is calculated by averaging the overall service outage occurrence over the number of VUs. The service outage rate of URQ is worse than both QOBA and EOBA when the panel size is small, but it performs better when the panel size is large. This is because URQ relies only on REC server to execute tasks, and the REC capacity is reduced by SRSU when solar energy availability is low. Therefore, URQ is affected the most by the deficiency of solar energy among other algorithms.

Moreover, we can observe the benefit of utilizing VU computing resource by comparing the performance of DURQ and URQ in large solar panel size situation. URQ has 0.1 QoS loss and 3.8% outage rate when solar energy is abundant. This is because REC capacity is limited. On the contrary, by utilizing some VU computing resource, DURQ can achieve 0.06 QoS loss and 0.9% outage rate with reasonable sized and cost solar panel. Fig. 4(c) shows that DURQ only needs very low ratio of local computing to achieve the above performance. The *local computing ratio* of DURQ is less than 0.03, which means no more than 3% of overall tasks' machine instructions are executed by VU computing resource.

Fig. 5(a) shows the QoS loss performance of the above algorithms under various computing capacities of a REC server. We assume the overall computing capacity of a REC server is the summation of the capacity of the NV Jetson Tx 2 modules it has. With 5 Jetson modules inside the REC server, the QoS loss

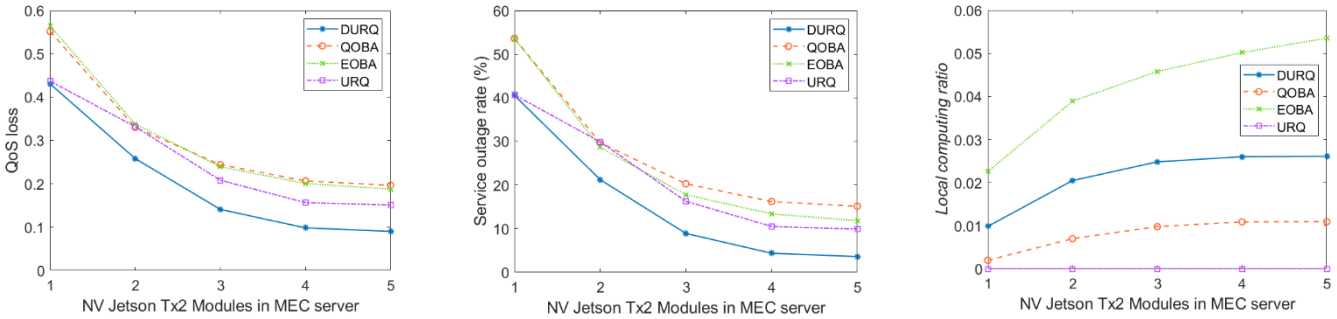

Fig. 5. Performances of 4 algorithms under various REC server computing capacities (a). left, QoS loss performance, (b) center, average service outage rate per VU, and (c) right, average *local computing ratio*.

of DURQ is 40% lower than URQ, 54% lower than QOBA, and 52% lower than EOBA.

Fig. 5(b) shows the corresponding service outage rate per VU. With only 1 Jetson module, DURQ performs the same as URQ, where most of the tasks are experiencing service outage because of the low capacity at REC server despite the algorithm. However, DURQ can significantly reduce service outage compared to the other techniques with an increase in Jetson modules, to a 4% rate with 5 Jetson modules. In Fig. 5(c), the *local computing ratio* increases with the number of Jetson modules because with more computing capacity of REC server, SRSU can serve more tasks from VUs and in the meantime leverage their local computing capacity to reduce the overall QoS loss. Fig. 5(c) also shows that once the computing capacity of the REC server increases to a certain level (more than 4 modules), there is no need for DURQ to keep increasing the utilization of VU's computing resource.

The results in Fig. 4 and 5 also demonstrate the tradeoff between the QoS loss performance and different solar panel size and REC specifications. We believe this enables the service providers to identify what might be the best configurations of SRSU for expected solar energy generations and vehicular traffic and application offloading demands.

VI. Conclusions and Future Work

In this paper, we propose DURQ, a QoS loss minimization algorithm for the SRSU network to support the dynamic offloading of computation intensive and delay sensitive vehicular applications. DURQ jointly considers the user association, *offloading sets*, and SRSU resource allocation decisions. Our simulation shows that DURQ can significantly reduce the QoS loss compared to other algorithms. The simulation is based on real-world solar energy generation and vehicular traffic data. Our proposed algorithm and simulation framework can help service providers and city planners to find optimal configuration and deployment of Solar-powered Roadside Units to enable SRSU-assisted vehicular applications.

Note that VUs will have high QoS loss despite our algorithm when the generation of solar energy is very low, like in bad weather conditions or in the evening. Therefore, in the future, we plan to include the use of battery in SRSU, and also consider utilizing other RE resources (e.g., wind energy), to ensure low QoS loss performance for SRSU-assisted VU computing.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1619184.

References

[1] B. Schoettle and M. Sivak, "A survey of public opinion about autonomous and self-driving vehicles in the U.S., the U.K., and Australia," Transp. Res. Inst., Univ. Michigan, Ann Arbor, MI, USA, Tech. Rep. UMTRI-2014-21, 2014.

[2] Y. Ku, P. Chiang and S. Dey, "Quality of Service Optimization for Vehicular Edge Computing with Solar-Powered Road Side Units," in *2018 IEEE 27th ICCCN*, Hangzhou, 2018, pp. 1-10.

[3] Y. Mao, *et al.*, "Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices," in *IEEE Journal on Selected Areas in Communications*, vol.34, no.12, pp.3590-3605, Dec. 2016.

[4] J. Xu and S. Ren, "Online Learning for Offloading and Autoscaling in Renewable-Powered Mobile Edge Computing," *2016 IEEE GLOBECOM*, Washington, DC, 2016, pp. 1-6.

[5] H. Wu, *et al.*, "Online Geographical Load Balancing for Energy-Harvesting Mobile Edge Computing," *2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO, 2018, pp. 1-6.

[6] L. Yang, J. Cao, H. Cheng and Y. Ji, "Multi-User Computation Partitioning for Latency Sensitive Mobile Cloud Applications," in *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2253-2266, 1 Aug. 2015.

[7] M. Deng, H. Tian and X. Lyu, "Adaptive sequential offloading game for multi-cell Mobile Edge Computing," *2016 23rd International Conference on Telecommunications (ICT)*, Thessaloniki, 2016, pp. 1-5.

[8] W. Liu, W. Gong, W. Du and C. Zou, "Computation offloading strategy for multi user mobile data streaming applications," *2017 19th International Conference on Advanced Communication Technology (ICACT)*, Bongpyeong, 2017, pp. 111-120.

[9] X. Mao, A. Maaref and K. H. Teo, "Adaptive Soft Frequency Reuse for Inter-Cell Interference Coordination in SC-FDMA Based 3GPP LTE Uplinks," *2008 IEEE GLOBECOM*, New Orleans, LO, 2008, pp. 1-6.

[10] Y. Mao, *et al.*, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322-2358, Fourthquarter 2017.

[11] N. T. Ti and L. B. Le, "Computation offloading leveraging computing resources from edge cloud and mobile peers," *2017 IEEE International Conference on Communications*, Paris, 2017, pp. 1-6.

[12] Z. Xu, *et al*., "Energy-Efficient Configuration of Spatial and Frequency Resources in MIMO-OFDMA Systems," in *IEEE Transactions on Communication*s, vol. 61, no. 2, pp. 564-575, Feb. 2013

[13] R. Hemmecke, *et al*, "Nonlinear Integer Programming," *50 Years of Integer Programming 1958-2008: The Early Years and State-of-the-Art Surveys*, Springer-Verlag, 2009.

[14] "NYS Traffic Data Viewer" [Online]. Available: https://gis3.dot.ny.gov/html5viewer/?viewer=tdv

[15] "NVIDIA Jetson TX2 Series System-on-Module", NVIDIA Co., CA, USA, Datasheet v.1.2, 2014 [Online]. Available: https://developer.nvidia.com/embedded/jetson-tx2

[16] P. Chiang, *et al.*, "Forecasting of Solar Photovoltaic System Power Generation using Wavelet Decomposition and Bias-compensated Random Forest," in *Proc. of the 9th Annual IEEE Green Technologies Conference*, Denver, CO, Mar. 2017, pp. 260-6.

[17] El Haj Ahmed G., *et al.*, "System-on-Chip Evaluation for the Implementation of Video Processing Servers" in *World Conference on Information Systems and Technologies*, Madeira, Portugal, Apr. 2017, pp. 267-274.

[18] R. Benenson, M. Mathias, R. Timofte and L. Van Gool, "Pedestrian detection at 100 frames per second," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, 2012, pp. 2903-2910