Cooperative Task-Oriented Group Formation for Vehicular Networks

Huiye Liu
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia, USA
huiyeliu@gatech.edu

Douglas M. Blough
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia, USA
doug.blough@ece.gatech.edu

***Abstract*—As vehicles are embedded with an increasing number of sensors and more powerful processors, computation-intensive on-board applications are being deployed. Emerging cooperative processing capabilities among vehicles will increase computing capability even further. In this paper, we present a novel framework for task-oriented group formation, where groups of vehicles are tailored for a specific cooperative computation task to be performed. We use the framework to develop a vehicular group formation algorithm that improves the quality of the computation result while achieving a specified probability of successful task completion. A prototype of the group formation algorithm for a generic distributed learning application example is implemented and extensively evaluated. Results show that our approach is able to significantly increase the percentage of successfully completed tasks compared to two baseline approaches.**

***Index Terms*—cooperative computation, vehicular networks**

I. Introduction

With advancements in information and communication technologies in modern vehicles, the amount of data they generate and the computation capability they possess are both growing rapidly. For example, autonomous vehicles (AVs) can generate between 1.4 and 19 terabytes (TB) of data per hour [1]. Such a large amount of data brings not only opportunities but also challenges in various distributed computation tasks requiring cooperation [2], e.g. reinforcement learning based cooperative driving [3], distributed consensus based false information filtering [4], collaborative active learning [5], etc., since if conducted appropriately, the performance of the applications should increase with the number of participants [6]. For instance, a single vehicle may not be able to capture accurate and full perception data from its own sensors due to imprecision and limited view. Aggregating the diverse views from multiple vehicles can improve the results of many computational tasks by creating larger and richer data sets. Moreover, though on-board computing devices are becoming increasingly powerful [7], computing locally with a very large data set often requires enormous computation and memory resources, which hinders these applications on resource-constrained edge devices such as vehicles [1]. As a result, most proposed solutions have assumed that data is sent to a powerful central server for computation [8].

Nonetheless, the application requirements discussed above pose challenges for the conventional cloud computing paradigm. It is difficult to guarantee the stringent quality/experience requirements due to a large on-board memory requirement, high latency, and limited backhaul bandwidth [9]. To tackle these challenges, recent research has considered mobile edge computing (MEC) [10]–[12], vehicular fog computing (VFC) [13]–[15] and vehicular cloud computing (VCC) [16], where computation tasks are offloaded to surrounding vehicles with surplus resources, to address certain aspects of the problem.

There are three critical challenges that are not well addressed in this prior work. 1) Whether tasks being assigned to vehicles can be successfully completed is ignored and, instead, focused on internal dependencies between tasks and which tasks should be assigned to which vehicles. Tasks that are not successfully completed serve no useful purpose but waste valuable computational resources. 2) Cooperative task execution was not considered, i.e. they only considered the offloading of a task from one vehicle to another. As discussed earlier, cooperative task execution is required to not only deal with the very large storage and computational resources required to process the large amount of data generated by modern vehicles, but also to break the barrier of limited local views/perceptions by aggregating information from multiple vehicles. 3) Prior works ignored the trade-off between successful task completion and quality of the cooperative computation results. In general, the result quality of such tasks increases with the size of the cooperating group, however, the larger the group becomes, the less stable is the group's connection, which results in a lower probability of task completion.

In this paper, we propose a task-oriented group formation method addressing the above challenges. To our knowledge, this is the first work addressing computation task oriented group formation in vehicular networks. Contributions include:

- A framework addresses task-oriented group formation, based on the probability of successful task completion. We show that task completion probability is primarily dependent on two random variables – the stay time of the vehicular task group and its task completion time. We use this framework and a notion of result quality to formulate the problem of selecting the best task group for a particular computation.
- We present a two-stage algorithm that performs task-oriented group formation for cooperative computations. The

algorithm maximizes the size of groups in order to produce the best cooperative result while targeting a specified probability of task completion.
- We report on a prototype implementation of our group formation algorithm, which is based on distributed learning applications for autonomous vehicles. The prototype runs in a realistic environment built on top of Veins and SUMO. Evaluation results show that our algorithm achieves high task completion rates and good group sizes across a wide range of traffic scenarios.

II. Related Work Discussion

Immersive vehicular applications such as advanced driver assistants, safety improvement, and autonomous driving have heavy computational requirements [13] and cannot be offloaded to remote clouds due to delay and bandwidth constraints [17]. The computational requirements of many potentially useful applications of this type also exceed the capacity of individual vehicles [18]. Thus, computational task offloading at the edge is essential for intelligent networked vehicles to reach their full potential. The literature in mobile edge computing (MEC) [10]–[12], vehicular fog computing (VFC) [13]–[15], [19], and vehicular cloud computing (VCC) [16], [17] has widely studied the problem of task offloading including aspects such as task scheduling [20], resource allocation [21], and computation complexity [22]. However, this prior work focuses primarily on one-to-one task offloading and also relies on infrastructure support in the form of edge servers and/or roadside units. Herein, we focus on one-to-many task assignment without infrastructure support.

Clustering algorithms provide one possible approach to forming vehicular groups for one-to-many task offloading. Clustering work in dynamic networks began with studies on MANETs [23], [24]. Various clustering design goals, e.g. load balancing, cost of clustering, speed of cluster formation, and real-time requirement [23] were achieved through single or combined metrics of connectivity, mobility, power, maintenance cost, etc. [24]. As discussed in [25], MANETs and VANETs are different in many ways, especially in mobility patterns, network topology, and communication link lifetime. With a goal of maximizing stability, clustering studies in VANET have considered mobility more thoroughly [26], [27]. However, these works do not factor in the characteristics of the tasks to be processed when determining a good group of vehicles to cluster, which is the focus of our work. Moreover, as we demonstrate later, our work can be layered on top of existing clustering algorithms to select subgroups from a cluster that are well suited to execute a specific computation.

In addressing task offloading for many cooperative computation applications in vehicles, where the quality of the results increases with the number of participating vehicles [28], [29], there is an interesting interplay between result quality and successful task completion. This is because larger groups improve result quality but also reduce the group stay time, which makes it less likely that the cooperative computation will complete while the task group remains together. Therefore, neglecting the important relationship between group stay time and task completion time could result in low task completion rate. Our work explores this important relationship, which none of the prior works have considered. Though other works such as [13] did model the completion time of tasks, they did not consider the relationship between group stay time and task completion time. Instead, like many other works, [13] relies on RSUs and base stations to relay intermediate data and computation results when direct vehicle-to-vehicle communication is interrupted prior to task completion.

If the main goal of a cooperative execution environment on vehicles is to successfully complete as many tasks as possible, it is important to account for task characteristics when assigning the tasks to vehicles. Our contribution is a task-oriented group formation scheme to maximize task completion rate in highly dynamic vehicular networking environments. To do this, we demonstrate that accounting for both group stability (stay time) *and* task completion time is essential. To the best of our knowledge, this is the first work addressing task oriented group formation in vehicular networks that accounts for both of these characteristics when assigning tasks to vehicles.

III. System Model Overview

Our system model is based on a bidirectional multi-lane road scenario. We assume basic communication among vehicles is supported, where each vehicle discovers their neighboring vehicles through periodic beacon messages. As is common in vehicular networks' research, e.g. [22], [30], we assume the beacon messages exchanged among vehicles contain the vehicles' basic information, including location, velocity, and moving direction, so that each vehicle can estimate speed difference, relative distance, and traffic condition/density around it as introduced in [31]. We assume that all vehicles use the same physical mode for transmitting or receiving data, and the precise time is known and traceable. In addition to periodic beacon messages, event/application driven messages are also supported. Finally, we assume there are no malicious vehicles, in that all vehicles follow the protocol. However, abnormal behavior such as packet loss or abnormally long packet delay can still occur. Note that we use node and vehicle interchangeably in the rest of this paper.

IV. Problem Formulation

The problem that we consider herein is how to construct a task group that is well suited to carrying out a specific task. We call the vehicles that have computation task requests *Task Vehicles* (TVs), whereas the vehicles that can provide their data and surplus computational resources to the TVs are referred to as *Service Vehicles* (SVs). Note that a single vehicle can serve as a TV at one time and as a SV at a different time but cannot serve in both roles simultaneously. Given a group G that is a candidate to perform a cooperative task, there are two important quantities, which are both random variables: 1) *stay time*, denoted by T_G^{stay}, and 2) *task completion time*, denoted by T_G^{task}. T_G^{stay} is the length of time that all members of G remain in communication range of each other, and T_G^{task} is

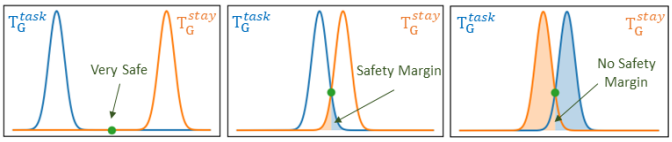

Fig. 1: Distribution of Stay Time and Task Completion Time. Gaussian distribution used as an illustration – framework does not assume any specific distribution.

the length of time that the vehicles of G need to complete the assigned task (assuming that the vehicles stay together for at least that amount of time).

As shown in Figure 1(a)-(c), how likely it is that the task will be completed successfully by a given group depends on the means of the T_G^{stay} and T_G^{task} distributions (as well as their shapes). If the mean stay time is much greater than the mean task completion time, e.g. Fig. 1(a), the task is very likely to be successfully completed. As the two distributions get closer together, e.g. Fig. 1(b), the probability of successful completion begins to decrease. Obviously, if the mean stay time becomes less than the mean task completion time, then it is very unlikely that the task will be successfully completed, e.g. Fig. 1(c). We consider a group viable if:

$$P(T_G^{stay} - T_G^{task} \geq 0) \geq 1 - \epsilon \ , \tag{1}$$

where $\epsilon \rightarrow 0$. While formally calculating this probability requires knowledge of the joint distribution of the two random variables, in practice any dependence between them is quite weak and they can be treated as independent random variables.

The size of a group is an important factor in the stay time and task completion time distributions and it can also impact the quality of the computational result. Larger groups tend to have a larger task completion time variance and higher communication cost, which for cooperative (non-parallel) computations, can also increase the expected task completion time. However, larger groups also tend to break apart more quickly. From these trends, we see that increasing the size of the group tends to drive the distributions from the Fig. 1(a) case (very small groups) to the Fig. 1(c) case (very large groups). Finally, we note that for many applications, e.g. the distributed learning example discussed in Section VI, larger task groups will produce higher quality results.[1]

Summarizing the above discussion, we tend to prefer larger task groups in order to improve the quality of the computational result. However, larger groups tend to be less likely to complete the task before they break apart. Thus, the goal of our task group formation algorithm, presented in the next section, is: **form as large a group as possible while ensuring that the group is very likely to successfully complete the task**, where "very likely to successfully complete the task" means that the group satisfies Inequality (1).

In practice, it is not possible to know the exact probability distributions of the stay time and task completion time of a group. Therefore, in our group formation algorithm presented in the next section, we introduce the notion of a *safety margin*, which is a safe separation between the estimated stay time and estimated task completion time for a group (see Figure 1). With an appropriate choice of safety margin, this can be considered as an approximation of the formal group viability condition specified by Inequality (1).

[1]For distributed learning, increasing the number of nodes creates a larger overall dataset and higher computational capability.

V. Task-Oriented Group Formation Algorithm

From the high-level perspective, we solve the problem formulated in the last section by a two-stage task-oriented group formation algorithm (ToG). As for the first stage, neighboring vehicles are clustered together based on relative mobility, connectivity and estimated stay time. The goal is to obtain stable clusters, in which the cluster members can not only stay together longer but also share good connectivity. In this work, we refer to this first stage as the parent clustering stage, where vehicles are clustered together and one parent cluster head (PCH) is selected for each parent cluster. The PCH is responsible for collecting and recording parent cluster members' (PCMs) statuses, position changes, and available resources. When a PCM is assigned a computation task, it becomes a TV and sends a task group formation request to its PCH. This triggers the second-stage – task group formation, where the PCH will select the best suited vehicles to form a task group based on certain criteria.

There are two main benefits of the proposed two-stage approach: a) parent clustering filters out the neighboring vehicles who have close geo-location but are less likely to stay together, thereby providing good candidates for task group formation; b) allowing a PCH to coordinate different task requests and select the qualified task group members for each of the requests increases efficiency and throughput. The details of the two-stage approach are presented next.

1) ***Parent Clustering****:* It is evident that a stable parent cluster scheme provides a strong basis for efficient task group formation, as the cluster members would stay together longer. Thus, for our parent clustering framework, we adopt the high-level clustering approach from [32], which achieves good cluster stability while maintaining relatively low communication overhead. Specifically, two main techniques are borrowed: 1) capability metrics, which are used to select the cluster head, and 2) substitute heads pre-selection before membership changes, so that both nodes that stay in the original cluster and nodes that are very likely to leave in a short period of time can smoothly settle down with limited effort, which increases the stability of clusters. The capability metrics of [32] that are used to select the cluster head are mainly focused on crossroads; therefore, we add another capability metric, which we refer to as relative distance metric (RDM), which is targeted at broader road types such as highways.

Next, we first define the capability metrics from [32], namely relative velocity metric (RVM) and power loss metric (PLM) and then we define our RDM metric. For vehicle v_i,

$$RVM(i) = \frac{1}{N}\sum_{j=1}^{N} log\left(\frac{v_{max}}{v_{max} - \Delta v_{i,j}}\right) \tag{2}$$

and

$$PLM(i) = \frac{1}{N}\sum_{j=1}^{N} log\left(\frac{P^t}{P^r_{i,j}}\right) \tag{3}$$

where, v_{max} is an upper bound on velocity, N denotes the number of direct neighbors of v_i, $\Delta v_{i,j}$ is the velocity difference between vehicles v_i and v_j, P^t is the unified transmission power of all nodes and $P^r_{i,j}$ denotes the received power of v_i from v_j. A smaller value of RVM indicates that the vehicle's velocity is similar to that of its direct neighbors. A smaller value of PLM means that the vehicle is more likely to have shorter communication distance and better channel quality with its direct neighbors.

Vehicles on a highway can have much higher speeds than in crossroads, and the distances between vehicles can be much longer, making the relative distance an important metric. While PLM is also related to distance, it is affected by other factors such as obstacles. Therefore, we add the relative distance metric, defined as:

$$RDM(i) = \frac{1}{N}\sum_{j=1}^{N} log\left(\frac{R}{R-\Delta d_{i,j}}\right) \tag{4}$$

where R is the communication range and $\Delta d_{i,j}$ represents the relative distance between node v_i and v_j. A smaller value of RDM indicates that a node is closer to the middle of its neighbors.

The combined capability metric we use is:

$$M(i) = RVM(i) + PLM(i) + RDM(i) \tag{5}$$

and the node with the smallest $M(i)$ among the cluster head candidates is selected as cluster head.

As parent cluster stability is the key to forming good task computation groups, we incorporate the neighbor sampling (NS) scheme from [33] to enhance the stability of parent clusters.[2] Additionally, we use a safe leaving distance σ to ensure that a PCH can plan in advance when a node is about to leave. A vehicle whose distance from its PCH is increasing must notify the PCH when it is within a distance σ of moving out of the PCH's transmission range. This safe leaving distance guarantees that the PCH can receive a notification about a leave before the connection is lost. Although we added a few other minor enhancements, the parent clustering procedure and maintenance strategy follow the general approach in [32].

*2) **Task Group Formation**:* Our task group formation scheme makes use of estimated stay time and estimated task completion time to choose groups that are well suited for the particular task to be executed. Good stay time and task completion time prediction schemes are the keys to efficient task group formation. We define $T^{stay}_{i,j}$ as the estimated stay time between parent cluster member v_i and v_j such that:

$$T^{stay}_{i,j} = \frac{|\Delta v_{i,j}|(\min\{R, D_{i,head}, D_{j,head}\}) - \Delta v_{i,j}\Delta D_{i,j}}{(\Delta v_{i,j})^2} \tag{6}$$

[2]Only vehicles within one hop vicinity moving in the same direction as well as a speed difference less than a threshold are considered as neighbor candidates. Refer to Algorithm 1 from [33] for details.

where R denotes the communication range of a vehicle, $D_{i,head}$ ($D_{j,head}$) denotes the distance between v_i (v_j) and the parent cluster head, and $\Delta D_{i,j}$ and $\Delta v_{i,j}$ represent the relative distance and velocity between vehicles v_i and v_j, respectively. In a task group, each vehicle is required to exchange data and computation results with each of the participants in a distributed fashion. Accordingly, a group G can be considered non-functional once the first pair of vehicles loses communication and we therefore estimate the stay time as:

$$\hat{T}^{stay}_G = \min_{i,j\in G}(T^{stay}_{i,j}) \ . \tag{7}$$

Different from stay time prediction, task completion time estimation is application specific. Instead of discussing the specific estimation scheme in this section, a distributed learning based application example is provided in Section VI-A. We will use the general notation $\hat{T}^{task}_G$ to represent estimated task completion time for a group G in this section. As different vehicles may be equipped with different computing capability, we use κ to represent the task computation rate of a vehicle. The larger the κ is, the faster a vehicle can compute its task. Let H denote the difference between estimated group stay time and task completion time. We only consider task groups with H larger than the safety margin T_{th}, i.e.

$$H = \hat{T}^{stay}_G - \hat{T}^{task}_G > T_{th} \tag{8}$$

Setting $T_{th} > 0$ accounts for the fact that $\hat{T}^{stay}_G$ and $\hat{T}^{task}_G$ are only estimates and we want to ensure that the task can be completed with high probability in its assigned group.

Vehicles in a parent cluster can be in one of three states: available, TV, or SV. PCHs can neither submit any computation request like a TV nor participate in any task computation as a SV. When a PCM has a request for computation assistance, it enters state TV and sends a request to its PCH. As discussed in Section III, a vehicle cannot act as a TV and a SV at the same time. Thus, once a vehicle changes its state from available to TV, it cannot service other tasks' computations until its current request is fulfilled or dropped. A vehicle that is currently serving a request cannot submit a computation request until it is done with its current request, i.e. a vehicle cannot change state directly from SV to TV. Rather, when the vehicle is done servicing a request, it changes its state to available and, only then, can it submit a computation request.

Algorithm 1 provides the details of the proposed task group formation algorithm. Each PCH maintains a member information table (MI) to keep track of PCMs' mobility and status information, periodically updated by intra-cluster messages. Upon receiving a task request, a PCH will check if there are available PCMs that can service tasks and attempts to form a task group. Initially, PCH starts from the assumption that all available PCMs within communication range R to the requester (TV) and $R-\sigma$ to the PCH are able to service the task, thus adding them to an empty group list G. Then, estimated stay time $\hat{T}^{stay}_G$ and estimated task completion time $\hat{T}^{task}_G$ are computed based on the information in MI. If H, as defined in Eq. 8, is larger than the safety margin T_{th}, the

Algorithm 1: Task Group Formation

```
1  PCM info table: MI, available PCM v_i ∈ MI, task vehicle: v^t;
2  while PCH receives a task request ∧ available PCMs do
3      initialize an empty group list G = [];
4      for all available v_i do
5          if D_{i,t} < R and D_{i,head} < R − σ then add v_i to G;
6      if |G| < C then
7          send task drop notification to v^t;
8          go back to the start of WHILE loop;
9      else calculate H;
10     while H < T_th do
11         G = G − argmax_{v_j ∈ G} H, recalculate H;
12     if |G| ≥ C then
13         PCH send group assignment notification;
14     else
15         send task drop notification to v^t;
```

PCH will send out a notification to TV and the selected SVs in G, notifying them of the formation of the group. However, if $H \leq T_{th}$, then the PCH removes the node with the largest M_G (metric calculated based on Eq. 5 with respect to G) or slowest task computation time, whichever can increase $\hat{T}_G^{stay}$ or decrease $\hat{T}_G^{task}$ the most. Lines 13-14 are repeated until a G with $H \geq T_{th}$ and size larger than the minimum allowed group size C is obtained. If no valid group can be found, the task is dropped and a notification is sent to the requester.

VI. Application Example and Evaluations

A. Application Example

As mentioned in the previous section, task completion time prediction is application specific. We consider a general distributed computation task [2] that requires multiple vehicles to cooperate by contributing combined resources of sensor data/local information, computing power, local decisions, etc, where examples are safety related applications [4] and on-board intelligence [2], [5]. In this section, we describe and evaluate an example on-board intelligence application - distributed learning across a vehicular group.

We assume a generalized distributed learning model, similar to the Federated Learning (FL) framework introduced in [34], but we do not address privacy concerns as FL does. In our application example, every car manufacturer has a own centralized server (CS) that can communicate to its manufactured vehicles through a Vehicle to Infrastructure (V2I) protocol. At each round of training, a CS selects a vehicle to initiate the process by sending it a task assignment. Following our proposed ToG algorithm, the selected vehicle will then submit the task request to its PCH and wait for a task group assignment. When the task is successfully completed, the task vehicle will send a task completion notification to its PCH and send model updates to its CS.

The task computation procedure for a given distributed learning task can be decomposed into four major stages: *1) Data Sharing:* for each task, a task group member shares its local data to other members, which are selected based on the task requirement with a fixed required length. *2) Task Computation:* each task group member collects other members' shared data, which totals to $|G|$ pieces of data (including its own piece). Once $|G|$ pieces of data are obtained, each member calculates and make predictions for the collected data set. *3) Results Sharing:* Once computation is done, each task group member multicasts its results with fixed required length to other group members. *4) Local Training:* Upon receiving computation results from other group members, a vehicle updates its local model by adding the unlabeled data points to its training set if the majority of the task group agree on the label[3]. After this process is completed, the TV syncs its update with the CS.

Let L_{Data} denote the fixed length of data needed to be exchanged by one task group member for a given task. Similarly, L_{Result} denotes the fixed length of result needed to be send out by one task group member after computing the task. As different models of vehicles may have different computing capability, we use operations per second κ to represent the computation rate of a given vehicle, and O_{Data} to represent total operations needed for given length of data L_{Data}. As the fourth stage – *Local Training* can be completed by a vehicle itself, it does not require stable connections among task group members. Hence, the task completion time as:

$$\hat{T}_G^{task} = \frac{|G| \times L_{Data}}{r_{min}^G} + \frac{|G| \times O_{Data}}{\kappa_{min}^G} + \frac{|G| \times L_{Result}}{r_{min}^G} \quad (9)$$

where $|G|$ represents the size of group G, r_{min}^G represents the minimum transmission rate among vehicles in G, and κ_{min}^G represents the minimum computation rate of the vehicles in G. By using the minimum transmission rate to estimate the data and result transmission times, we obtain an upper bound, which can be considered to compensate for additional time needed because of packet delays and losses.

By applying the above task completion time estimation scheme, we are able to follow Algorithm 1 to form groups tailored for distributed learning. An extensive simulation study based on this type of application and making use of real world maps is provided in the following subsections.

B. Simulation Set-Up

We implemented a prototype of ToG using C++ and Python, which can simulate not only different real maps but also different traffic scenarios by updating a small set of system parameters. It is built on top of Veins [36] which provides a comprehensive suite of models of IEEE 802.11p, DSRC/WAVE and obstacle shadowing. We add additional layers to simulate the communication between remote servers and TVs. TraCI from SUMO [37] is used to control the mobility of vehicles. About 4000 lines of code are written for the prototype. Real highway, branches and intersections

[3]Averaging local models may balance their contributions to produce an accurate joint model, though this is not guaranteed. Instead, we use tri-training [35], a classic method that reduces prediction bias on unlabeled data by using the agreement of multiple independently trained models.

Fig. 2: (a) 3.5 mile highway section. (b), (c) are captured traffic images on the same 120m highway section. Each small yellow triangle represents a vehicle.

are obtained from OpenStreetMap (OSM) [38]) with manual corrections referenced from Google Satellite.

Vehicles can communicate with the centralized servers and nearby vehicles through V2I and V2V, respectively. Figure 2(a) depicts a major highway section in Atlanta. Its main road is ∼3.5 miles long with 6 traffic lights controlling the connections to the branches, and 5 lanes in each major road of each direction. We simulate the example application as described in Section VI-A, where available vehicles are selected as TVs and receive a computation task assignment at random times. Upon receiving a task assignment, a vehicle submits a task request to its PCH asking for a task group formation.

The actual $\hat{T}_G^{Task}$ of a task consists of the actual communication cost and the actual task computation time. Communication cost can be well simulated in the current environment, but the task computation time is quite application specific and it can be affected by various factors, e.g., concurrent background processes, RAM size, etc. Therefore, to better evaluate the effectiveness of our proposed approach, we use two normal distributions $K \sim N(\mu_\kappa, \sigma_\kappa^2)$ and $T \sim N(\mu_\tau, \sigma_\tau^2)$ with adjustable mean and variance to cover a wide range of possible task computation times. As different vehicles may be equipped with different rated chips, at the initial stage, each vehicle entering the simulated area is assigned a computation capability rate κ (FLOPs), which follows the $K \sim N(\mu_\kappa, \sigma_\kappa^2)$ distribution. However, even if every vehicle has the same hardware, the actual task computation time is determined by various factors, and cannot be predicted precisely based on the rated computation capability. Thus, we use another normal distribution $T \sim N(\mu_\tau, \sigma_\tau^2)$ to model these variations in computation time. Therefore, the actual task computation time is the sum of the rated computation time chosen from the K distribution and the computation time variation chosen from the T distribution. To simulate different stay times, we included 6 categories of vehicles and varied the network flow as well as vehicle density by controlling the area throughput. For example, both Figure 2(b) and Figure 2(c) depict the same 120m segment in the simulated highway with different average numbers of vehicles, where the stay-time would be very different.

C. Evaluation of ToG Algorithm

For each assigned task, we defined three possible end states: completed, failed, and dropped. A task was counted as completed if the TV received computation results from all group members; alternatively, a task was considered as failed if the TV did not receive computation results from all group members before the assigned group broke apart; finally, if a PCH tried to form a task group but could not find any suitable combination (Eq. 8), the task was counted as dropped. Note that there were situations that a PCH or TV exited the simulated area while executing their task. As this was caused by the limited simulation area instead of failure of the algorithm, we ignored these cases in the evaluation results. The following metrics are used in our evaluation:

- *Task Execution Rate (TER):* The percentage of completed tasks over the number of completed, failed and dropped tasks. The higher the TER is, the higher the ratio of completed tasks versus task group formation attempts can be obtained, which is one of the ultimate goals of ToC.
- *Average Group Size (AGS):* The average group size of completed tasks. Since larger groups typically produce higher quality results for the cooperative tasks we are interested in, one of our goals is to maximize group size. Since task group size of failed tasks and dropped tasks give no indication on the quality of the formed groups, only completed tasks were counted in this metric.

Unless otherwise noted, the following parameters were used as default setting in all experiments: number of vehicles = 1800, $\mu_\kappa = 1.3$ TFLOPs[4], $\sigma_\kappa = 0.1$ TFLOPs, $\mu_\tau = 0.3$ min, $\sigma_\tau = 0.2$ min, $O_{Data} = 135$T operations, and beacon message frequency = 10Hz. The communication range among vehicles was set to $300m$ based on NHTSA's proposed rule [39]. Natural packet loss and delay were simulated such that messages were randomly dropped at receiving vehicles with a drop rate of 10% and packets were randomly delayed within the range of 50ms - 500ms. The minimum size of a task group is 2.

*1) **Choosing safety margin**:* We first evaluated how the safety margin T_{th} affects ToG's performance, to determine a suitable way to configure T_{th}. Recall that H is the difference between estimated group stay time and task completion time. Intuitively, a safety margin tolerates the gap between H and the actual difference of these quantities. Thus, one possibility is to make T_{th} a certain percentage of $\hat{T}_G^{Task}$ (Scenario A). However, a good T_{th} should also create a good separation between the distributions of task completion time and vehicle stay time. Then, the variance of task completion time should play an important role in choosing a proper safety margin. Thus, a second possibility is to make T_{th} proportional to σ_τ, the major variance factor in the actual T_G^{Task} (Scenario B).

Based on this, we did a comparison experiment where we varied T_{th} as both a percentage of the predicted task

[4]We set the mean computation rate as 1.3 TFLOPs based on the recently released Nvidia Drive AGX chip [7] designed for Level 2 as well as Level 3 autonomous vehicles. This is just for an example demonstration, the proposed algorithm does not rely on any specific hardware.

TABLE I: Task Execution Rate (TER), Task Completion Rate (TCR), and Average Group Size (AGS) for Different Ways of Choosing Safety Margin T_{th}. In each cell of the table, TER, TCR, AGS are presented.

(a) Scenario A: T_{th} as a Percentage of Predicted Task Completion Time T_G^{Task}

	5%	10%	15%	20%	25%	30%	35%
$\sigma_\tau = 0.2$	83.8,84.6,7.2	89.4,89.8,6.7	91.3,92.1,6.6	92.0,93.2,5.9	89.7,94.1,5.0	88.8,94.9,4.5	87.2,95.0,3.8
$\sigma_\tau = 0.4$	78.5,79.2,7.0	85.9,86.6,6.8	86.3,87.0,6.5	88.7,89.4,5.8	89.4,90.8,4.9	86.2,91.4,4.5	83.5,92.1,4.0
$\sigma_\tau = 0.6$	71.6,71.9,6.8	74.2,76.5,6.7	77.6,80.3,6.4	81.4,82.1,5.9	84.0,84.9,4.8	85.3,88.7,4.4	79.8,89.1,3.9
$\sigma_\tau = 0.8$	58.0,58.1,6.7	60.8,61.3,6.5	63.6,64.9,6.1	66.6,69.3,5.7	70.4,74.8,4.6	78.7,85.6,4.3	85.1,88.0,4.1

(b) Scenario B: T_{th} as a Linear Function of Computation Time Variance σ_τ

	$0.75\sigma_\tau$	$1.0\sigma_\tau$	$1.25\sigma_\tau$	$1.5\sigma_\tau$	$1.75\sigma_\tau$	$2.0\sigma_\tau$	$2.25\sigma_\tau$
$\sigma_\tau = 0.2$	85.6,85.9,7.0	90.8,91.5,6.6	92.0,92.8,6.5	92.3,93.1,6.4	92.5,93.5,6.2	92.6,94.0,5.9	90.8,94.6,5.6
$\sigma_\tau = 0.4$	86.0,86.4,6.9	91.6,91.8,6.5	91.9,92.7,6.4	92.1,92.9,6.2	92.4,93.3,5.9	92.2,93.8,5.7	90.2,94.3,5.3
$\sigma_\tau = 0.6$	86.7,87.0,6.2	90.3,90.5,6.1	90.9,92.4,6.1	90.4,92.8,5.7	90.1,93.4,5.6	89.8,93.9,5.3	88.5,94.1,5.1
$\sigma_\tau = 0.8$	87.2,88.6,4.1	88.5,89.0,4.8	86.3,89.2,5.4	82.4,89.6,5.6	76.8,89.9,4.8	70.2,90.0,4.5	62.7,90.0,4.1

completion time with 5% increments from 5% to 35%, and as multiples of σ_τ with 0.25 factor increments from $0.75\sigma_\tau$ to $2.25\sigma_\tau$. We ran simulations with the default setting for both scenarios, and varied the σ_τ from 0.2 to 0.8 with 0.2 increments for simulating different task computation time distributions. Thirty simulation runs were done for each parameter combination, where 200 computation tasks in total were distributed and tracked in each run. Every simulation run ends when all distributed computation tasks end in one of the three states (completed, failed, dropped) or the corresponding TV exits the map. Since TER may be heavily affected by large number of dropped tasks if there are no suitable members to form the group, we introduce a new metric to help choose T_{th}: Task Completion Rate (TCR). TCR is the percentage of completed tasks over the number of completed and failed tasks, where dropped tasks are not counted.

The results are reported in Table I. Table I(a) shows the Scenario A result for each combination of T_{th} and σ_τ in the format of (TER, TCR, AGS). Not surprisingly, for the same type of task computation time distribution (same σ_τ), TCR increases and AGS decreases as T_{th} grows. This is because a longer T_{th} increases the deviation that can be tolerated from the predicted H value. Thus, intuitively, the larger the T_{th}, the higher the TCR should be obtained. Note that different from TCR, TER drops as T_{th} becomes too high, because as T_{th} grows, PCHs may not be able to find enough suitable PCMs to form groups, thereby causing the drop in TER. Also, as σ_τ increases, the overall performance drops. This is due to the fact that, the distribution variance of the task computation time is not considered in this way of choosing T_{th}. The predicted $\hat{T}_G^{Task}$ is solely based on estimated communication cost among task group members and rated task computation time (see Eq. 9). Therefore, for larger σ_τ, higher T_{th} performs better, while for smaller σ_τ, lower T_{th} leads to better performance.

Table I(b) shows the result of Scenario B, in the same format as in Table I(a). As opposed to Scenario A, we see that making the safety margin proportional to σ_τ makes the results fairly consistent as σ_τ is varied. For any given T_{th} value, both TER and TCR are fairly stable for σ_τ in the range [0.2, 0.6]. It is only when σ_τ becomes 0.8 and the safety margin is large, that we see a significant drop-off in TER, which is the ultimate metric of interest. We note that, when $\sigma_\tau = 0.8$, the variance becomes unreasonably large for this simulation scenario and the poor performance is largely due to failed tasks arising from group members exiting the map before task completion.

This comparison shows that, if the value of σ_τ is known, T_{th} in the range of $[1.0\sigma_\tau, 2.0\sigma_\tau]$ give consistently good results on the ultimate metric, task execution rate, while also achieving good group sizes. Obviously, knowing σ_τ means that the task completion time distributions must be characterized for the tasks being executed. We briefly discuss ways in which this could be done in Conclusion. The Scenario A results show that, without knowledge of σ_τ, a larger safety margin might be needed to tolerate the possible range of σ_τ values and both TER and average group size will be somewhat lower than for Scenario B but good performance is still achieved.

Based on this discussion, in the remainder of the simulations, we assume σ_τ is known and we set $T_{th} = 1.25\sigma_\tau$ and $\sigma_\tau = 2$ as a representative scenario for further evaluations.

*2) **Impact of the number of vehicles**:* We also evaluated ToG's performance versus the number of vehicles, as this parameter has a large impact on the vehicle stay time distribution. For the number of vehicles ranging from 600 to 3400 in increments of 400, we repeated the simulation 30 times for each case. The results are reported in Figure 3. From the figure, we can see that as the number of vehicles increases, TER becomes closer to TCR and AGS increases from 2.6 to 14.9. Though when the number of vehicles is very small and vehicles move very freely, ToG is still able to achieve a TER above 87%, which shows the strong potential of ToG's estimation of stay time, making it robust to various traffic scenarios that produce widely different stay time distributions.

*3) **Impact of communication delay**:* As the communication environment plays an important role in affecting group

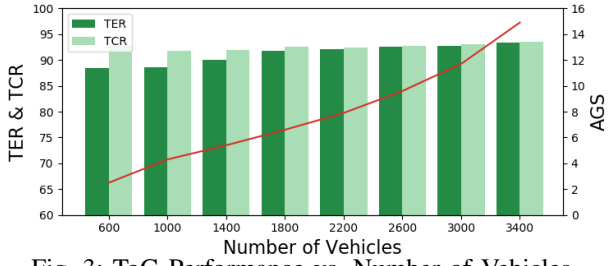

Fig. 3: ToG Performance vs. Number of Vehicles

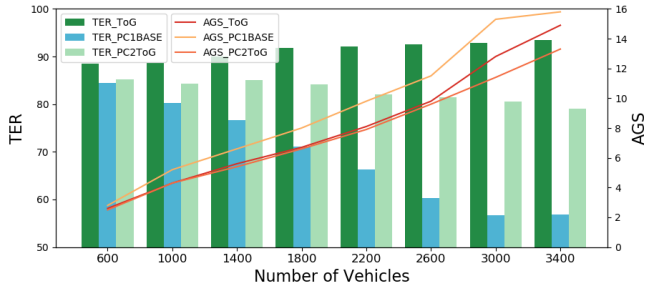

Fig. 4: Performance of 3 Approaches vs. Number of Vehicles

stability, we also evaluated performance over a wide range of packet delays[5], which was varied from [50,250) ms to [50, 1050) ms with 200 ms increments on the maximum delay. Thirty simulation runs were conducted for each case and the results are reported in Table II. From the table, we can see that as the packet delay range expands, TER, TCT and AGS show only small decreases until the packet delay reaches 1050 ms. Even in the range of [50,1050) ms, our proposed method is still able to achieve 87.3% TER with 5.4 AGS, which is still good performance. Thanks to the safety margin design and loosely bounded task completion time estimation, ToG shows good tolerance to communication delays while maintaining high TER, TCR and AGS.

D. Comparison of ToG Algorithm with Other Approaches

To better illustrate the benefits of the ToG Algorithm and how each stage affects the final performance, we compared its performance against two baseline approaches – 1) PC1Base: an approach that adopts the same first-stage parent clustering scheme as our proposed ToG scheme but uses a baseline second-stage task group formation methodology, which selects task group members that are within communication range of the TV and from which the PCH has received at least one beacon message update in the most recent 5 cycles, and 2) PC2ToG: an approach that uses a well cited clustering scheme [40] in the first stage and adopts the same task group formation method as the ToG Algorithm in the second stage. By choosing these alternative approaches, we hope to, in part, determine the relative impacts of the first stage (parent clustering) and second stage (task group formation) of our ToG approach. Figure 4 depicts the performance of the three algorithms versus different numbers of vehicles (from 600 to 3400 with increments of 400).

[5]Very long packet delay emulates the impact of packet loss.

TABLE II: ToG Performance vs. Packet Delay

Delay (ms)	[50,250)	[50,450)	[50,650)	[50,850)	[50,1050)
TER	92.5	92.1	91.4	90.2	87.3
TCR	92.8	92.6	92.0	90.9	88.5
AGS	6.7	6.6	6.3	6.1	5.4

We first compare the performance of ToG to that of PC1Base. It is observed that the TER of PC1Base drops quickly (from 84.4% to 56.8%) as the number of vehicles increases. Though PC1Base and ToG adopt the same parent clustering scheme, in the task group formation stage, ToG considers the distribution relationship between $\hat{T}_G^{Stay}$ and $\hat{T}_G^{Task}$, and selects PCMs with a better $\hat{T}_G^{Stay}$, $\hat{T}_G^{Task}$ distribution separation (through T_{th}) to form a task group, while PC1Base only considers distance and signal metrics and ignores the factors of different vehicles' computing capabilities as well as stay times of different groups. With PC1Base, as the number of vehicles increases, more PCMs within communication range and with good connection are not able to finish task computation before separation, causing the large decrease in TER. Nevertheless, as is expected, the AGS achieved by ToG is slightly smaller than that achieved by PC1Base, because ToG has more constraints in selecting task group members as a trade-off for better TER. Overall, this comparison shows the clear benefits of our task-oriented group formation in successfully completing a much higher percentage of tasks while achieving close to the same average group size as an approach that does not factor in the task requirements when choosing a computation group.

Next, we compare the performances of PC1Base and PC2ToG. Similar to PC1Base, PC2ToG's TER decreases as the number of vehicles increases, but not by as much (only from 85.2% to 79.3%). Recall that PC2ToG adopts the same task group formation scheme as ToG, but uses a different parent clustering scheme. This shows that, while both ToG's clustering scheme and its task group formation scheme improve the task execution rate, the task group formation scheme is the more important factor and using it with other parent clustering schemes still provides substantial benefits.

In the end, if we put an eye on performance among all three algorithms, we observe that only ToG's TER increases as the number of vehicles increases. This is because our parent clustering scheme both clusters vehicles that tend to stay longer together, which provides good candidates for second stage task group formation, and is robust across different vehicle densities. Then, in the second stage, only PCMs with higher potential of completing tasks together before losing connection are selected as task group members. Besides, it always tries to find the largest possible group to achieve better model training results. Therefore, we see that AGS grows as the number of vehicles increases. Moreover, although PC2ToG results in lower TER than ToG, it has better TER performance than PC1Base, and this shows that our second-stage task-oriented group formation algorithm can be combined with different parent clustering schemes to improve the success rate

of tasks completing.

VII. Conclusion and Future Work

Though we have demonstrated that our ToG approach has great potential in achieving high TER while maximizing AGS, it requires some prior knowledge of application-specific task completion time distributions. The estimation and statistical modeling of task completion time is an interesting area. Due to time and space limits, we were not able to address that aspect in this paper. However, in future work, we will consider the following high-level ideas to approach this problem: *1)* identify the important variables of a task affecting the task completion time, such as rated computation speed, operations required for the task, communication cost, etc.; *2)* collect data through simulations under different traffic settings, and build a baseline statistical model for a specific algorithm (techniques such as survival analysis [41] may be applied); and *3)* starting with the baseline model, apply online learning techniques [9] to tune the model during deployment.

References

[1] S. Dmitriev, "Autonomous cars will generate more than 300 tb of data per year," Jul 2020. [Online]. Available: https://www.tuxera.com/blog/autonomous-cars-300-tb-of-data-per-year/

[2] A. Alhilal, T. Braud, and P. Hui, "Distributed vehicular computing at the dawn of 5g: A survey," *arXiv preprint arXiv:2001.07077*, 2020.

[3] G. Wang, J. Hu, Z. Li, and L. Li, "Cooperative lane changing via deep reinforcement learning," *arXiv preprint arXiv:1906.08662*, 2019.

[4] H. Liu, C.-W. Lin, E. Kang, S. Shiraishi, and D. M. Blough, "A byzantine-tolerant distributed consensus algorithm for connected vehicles using proof-of-eligibility," in *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2019.

[5] H. Liu and D. Blough, "MultiVTrain: collaborative Multi-View active learning for segmentation in connected vehicles," in *2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS) (IEEE MASS 2021)*, 2021.

[6] S. Grigorescu, B. Trasnea *et al.*, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, 2020.

[7] NVIDIA Corporation. (2020) NVIDIA drive AGX developer kit. [Online]. Available: https://developer.nvidia.com/drive/drive-agx

[8] H. Ye, L. Liang, G. Y. Li, J. Kim, L. Lu, and M. Wu, "Machine learning for vehicular networks: Recent advances and application examples," *IEEE Vehicular Technology Magazine*, 2018.

[9] Y. Sun, X. Guo, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Learning-based task offloading for vehicular cloud computing systems," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018.

[10] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, 2017.

[11] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, 2017.

[12] C. Yang, Y. Liu *et al.*, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, 2019.

[13] Z. Zhou, H. Liao *et al.*, "Reliable task offloading for vehicular fog computing under information asymmetry and information uncertainty," *IEEE Transactions on Vehicular Technology*, 2019.

[14] G. Zhang, F. Shen, Y. Yang, H. Qian, and W. Yao, "Fair task offloading among fog nodes in fog computing networks," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018.

[15] C. Zhu, G. Pastor, Y. Xiao, Y. Li, and A. Ylae-Jaeaeski, "Fog following me: Latency and quality balanced task allocation in vehicular fog computing," in *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2018.

[16] M. LiWang, Z. Gao *et al.*, "Multi-task offloading over vehicular clouds under graph-based representation," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020.

[17] F. Sun, F. Hou, N. Cheng, M. Wang, H. Zhou, L. Gui, and X. Shen, "Cooperative task scheduling for computation offloading in vehicular cloud," *IEEE Transactions on Vehicular Technology*, 2018.

[18] S. Olariu, "A survey of vehicular cloud research: Trends, applications and challenges," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[19] W. Zhang, Z. Zhang, and H.-C. Chao, "Cooperative fog computing for dealing with big data in the internet of vehicles: Architecture and hierarchical resource management," *IEEE Communications Magazine*, 2017.

[20] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous vehicular edge computing framework with aco-based scheduling," *IEEE Transactions on Vehicular Technology*, 2017.

[21] G. Qiao, S. Leng, K. Zhang, and Y. He, "Collaborative task offloading in vehicular edge multi-access networks," *IEEE Communications*, 2018.

[22] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, 2019.

[23] J. Y. Yu and P. H. J. Chong, "A survey of clustering schemes for mobile ad hoc networks," *IEEE Communications Surveys & Tutorials*, 2005.

[24] R. Agarwal and D. Motwani, "Survey of clustering algorithms for MANET," *arXiv preprint arXiv:0912.2303*, 2009.

[25] M. Sood and S. Kanwar, "Clustering in MANET and VANET: A survey," in *2014 International Conference on Circuits, Systems, Communication and Information Technology Applications*. IEEE, 2014.

[26] C. Shea, B. Hassanabadi, and S. Valaee, "Mobility-based clustering in vanets using affinity propagation," in *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*. IEEE, 2009.

[27] C. Cooper, D. Franklin, M. Ros, F. Safaei, and M. Abolhasan, "A comparative survey of VANET clustering techniques," *IEEE Communications Surveys & Tutorials*, 2016.

[28] C. Allig and G. Wanielik, "Alignment of perception information for cooperative perception," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019.

[29] A. A. Abdellatif, C. F. Chiasserini, and F. Malandrino, "Active learning-based classification in automated connected vehicles," *arXiv preprint arXiv:2002.07593*, 2020.

[30] M. Ren, L. Khoukhi, H. Labiod, J. Zhang, and V. Veque, "A new mobility-based clustering algorithm for vehicular ad hoc networks (VANETs)," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2016.

[31] A. Daeinabi, A. G. P. Rahbar, and A. Khademzadeh, "VWCA: An efficient clustering algorithm in vehicular ad hoc networks," *Journal of Network and Computer Applications*, 2011.

[32] Y. Huo, Y. Liu, L. Ma, X. Cheng, and T. Jing, "An enhanced low overhead and stable clustering scheme for crossroads in VANETs," *EURASIP Journal on Wireless Communications and Networking*, 2016.

[33] M. Ren, J. Zhang, and etl al., "A unified framework of clustering approach in vehicular ad hoc networks," *IEEE Transactions on intelligent transportation systems*, 2017.

[34] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[35] Z.-H. Zhou and M. Li, "Tri-training: Exploiting unlabeled data using three classifiers," *IEEE Transactions on Knowledge and Data Engineering*, 2005.

[36] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road traffic simulation for improved IVC analysis," *IEEE Transactions on Mobile Computing (TMC)*, 2011.

[37] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO-simulation of urban mobility," *International Journal on Advances in Systems and Measurements*, 2012.

[38] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ," https://www.openstreetmap.org , 2017.

[39] NHTSA, "Federal motor vehicle safety standards; v2v communications," 2017. [Online]. Available: https://www.federalregister.gov/d/2016-31059

[40] M. Ren, L. Khoukhi, H. Labiod, J. Zhang, and V. Veque, "A mobility-based scheme for dynamic clustering in vehicular ad-hoc networks (VANETs)," *Vehicular Communications*, 2017.

[41] J. Wang, S. Faridani, and P. Ipeirotis, "Estimating the completion time of crowdsourced tasks using survival analysis models," *Crowdsourcing for Search and Data Mining (CSDM 2011)*, 2011.