On Leveraging the Computational Potential of Fog-Enabled Vehicular Networks

Ibrahim Sorkhoh
Concordia University
Montreal, Canada
e_sarkho@live.concordia.ca

Dariush Ebarhimi
University of Waterloo
Ontario, Canada
d2ebrahi@uwaterloo.ca

Sanaa Sharafeddine
Lebanese American University
Beirut, Lebanon
sanaa.sharafeddine@lau.edu.lb

Chadi Assi
Concordia University
Montreal, Canada
assi@ciise.concordia.ca

ABSTRACT

The advent of autonomous vehicles demands powerful processing capabilities of on-board units to handle the dramatic increase of sensor data used to make safe self-driving decisions. Those computational resources, being constantly available on the highways, represent valuable assets that can be leveraged to serve as fog computing facility to computational tasks generated from other vehicles or even from different networks. In this paper, we propose a fog-enabled system scheme that can be deployed on a road side unit (RSU) to schedule and offload requested computational tasks over the available vehicles' on-board units (OBUs). The goal is to maximize the weighted sum of the admitted tasks. We model the problem as a Mixed Integer Linear Programming (MILP), and due to NP-hardness, we propose a Dantzig-Wolfe decomposition method to provide a scalable solution. The experiment shows that our approach has a sufficient effectiveness in terms of both computational complexity and tasks acceptance rate.

KEYWORDS

Vehicular networks, fog computing, Dantzig-Wolfe decomposition

ACM Reference Format:
Ibrahim Sorkhoh, Dariush Ebarhimi, Sanaa Sharafeddine, and Chadi Assi. 2018. On Leveraging the Computational Potential of Fog-Enabled Vehicular Networks. In *MSWIM '19: The 22nd ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, November 25 – 29, 2019, Miami Beach, FL.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

In recent years, vast efforts from governments, industry and academia were all focused to establish the infrastructures of the fifth generation of communication technologies (5G). The federal budget in the United States dedicated to support 5G has reached 1.3 trillion dollars [11]. Companies like Intel, Qualcomm, Nokia and others are collaborating with researchers from different disciplines in the advancement of technologies to support and utilize 5G communication capabilities [9, 15]. Many fields including health-care, security, manufacturing, and of course transportation will benefit from these technologies to make a leap forward into a completely new era [9] by providing communication service far beyond the current wireless communication performance standards. Among the different fields, transportation in its evolution to include autonomous driving is taking a considerable attention due to its major implication on human safety on a daily basis. Deploying self-driving vehicles is one of the major artificial intelligence-based idea that have evolved to date leading to varying levels of self-driving capabilities. 3GPP provided a standard specification of autonomous driving levels that starts from no automation (level 0) to a fully-automated system (level 5) [13]. Increased level of autonomy imposes higher processing capabilities of OBUs that can efficiently run complex software to generate in-time decisions for safe driving in challenging road conditions. Over long times, processing capabilities of OBUs might be under utilized. Thanks to the 5G communication technologies, it becomes feasible to offload computation tasks over these OBUs in order to leverage their idle time. In addition to their high computation capabilities, the OBUs of these vehicles are usually deployed with complex artificial intelligence tools including speech recognition algorithms, data mining and machine learning models, computer vision and signal processing technologies. Vehicles that lack such resources can improve their driving automation potentials by offloading their computational tasks to nearby vehicles having these computational capabilities. One of the typical example in driving automation is augmented reality where the system creates 3D objects in order to support the driver safety and avoid traffic jams. Other example is performing commands through recognizing the driver speech to perform a certain task (e.g., controlling the multimedia system).

Besides these vehicles, some other entities might exploit that approach for many other usages. An internet of things device that tries to provide an overall road network status can request from vehicles to sense the surrounding environment, apply some artificial intelligence techniques and send certain information about the road status. A pedestrian who tries to apply a certain object recognition technique over a picture can leverage nearby vehicles to perform such an energy intensive and complex task instead of doing it

MSWIM '19, November 25 – 29, 2019, Miami Beach, FL
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

locally. Other examples include academic labs that require a study on the drivers behaviour. The detection of peculiar drivers on the road in order to take an action toward him/her. There are many other scenarios that support the idea of leveraging the vehicles computation capabilities and necessitates the establishment of a system scheme in order to realize this concept.

Providing a system that facilitates such a paradigm requires tackling several intricate challenges manifesting the need of an efficient and scalable task offloading scheme. First, the dynamism of the resource availability and the tasks' short lifetime make the scheduling process a cumbersome one. The vehicles availability on the RSU range is limited and any scheduler must consider this system variability. Second, although the vehicles OBUs do have sufficient resources, but a task offloaded to the system usually requires a high computational demands to ensure in-time processing. Managing these resources must consider that available computational capacity for public usage is limited though abundant as the vehicle prioritizes its own tasks on the OBU. Third, since the deadline of the tasks is usually in order of milliseconds, providing an adequate portion of the available radio spectrum should be efficient enough so the data to upload/download can arrive completely to its destination in a matter of milliseconds. Given all these challenges, it is critical to equip the RSU with a highly efficient scheduler to schedule tasks on OBUs.

The main contribution of this work is three fold: 1) We proposed a system to be deployed at RSUs that receives low-latency computation tasks and efficiently schedules them to be processed by incoming vehicles in a way to collect the task results before exiting the RSU coverage range. 2) We mathematically formulated the scheduling problem as mixed integer linear program while taking into account the dynamically available vehicles to process the tasks and their limited computational capacity, in addition to bandwidth limitations. 3) To combat the complexity of the problem after proving it NP hard, we proposed a solution based on Dantzig-Wolfe decomposition technique. Our solution decomposes the problem through a column generation algorithm that solves the linear master problem with barrier algorithm and the subproblems with a polynomial-time dynamic programming approach.

The paper is structured as follows: Section 2 is the related work. The system model is discussed in Section 3. The problem definition is presented in Section 4, followed by the mathematical model in Section 5. The Dantzig-Wolfe decomposition and the column generation algorithm are discussed in Sections 5 and 6 respectively. We test our approach and discuss the results in Section 8, and finally conclude in Section 9.

2 RELATED WORK

Vehicular fog computing is taking a considerable attention from academia leading to extensive studies addressing major concerns. In [6], authors used machine learning techniques to choose the best fog server deployed over a base station to be connected to a vehicle once it leaves a certain server range. The work considers the load and the location of the vehicles to accomplish an accurate prediction for the server. In [19], the authors suggested a machine learning algorithm that tries to utilize the vehicles mobility in order to minimize the delay of the tasks computation. The authors in [7] suggested a three layers scheme to manage the vehicles (parked and moving) considering them as computation fog nodes. They also suggested an offloading scheme for real-time traffic management and applied a real-world taxi-trajectory-analysis-based evaluation. A design principle for fog-enabled vehicular software-defined networking was suggested in [8]. The design was evaluated with the use case of traffic management system for fast traffic rescue using real traffic accident data. The work in [20] suggested an offloading scheme of tasks generated by the users equipment to the vehicles based on contract theory and matching theory. First, a contract is designed relating the required performance levels to the payments issued to vehicles offering this performance. Then a two-sided matching game approach is performed to assign the tasks to the vehicles. A framework to exploit the idle time in the vehicles computation resources is suggested in [4]. The work suggests a decentralized framework to utilize the computation resources by caching the jobs and assigning them using an Ant-colony based algorithm. The authors in [16] provided a mathematical model that studies quantitatively the vehicular fog computing capabilities. The work uses a realistic data acquired from tens of thousands of taxis. For data distribution application, [17] suggested a joint optimization of access mode selection and spectrum allocation while considering the randomness of the vehicular network, the edge cache and the content download delay.

The 3rd Generation Partnership Project (3GPP) provided communication standards for the vehicle-to-everything networks (V2X). These standards are supposed to support applications like vehicles platooning and remote driving [5, 14]. The standards specify levels of driving automation that vary from "no automation" (level 0) to "full automation" (level 5). The channels bandwidth can go up to 10 MHz in both ETSI and IEEE standards, hence, the bit rate can reach up to 27 Mb/s and 54 Mb/s. Such bit rate can not support autonomous driving which demands the latency to be no more than 100 ms. Hence, several works published recently proposed the utilization of mmWave communication technologies for vehicular networks [10, 12, 18]. The only obstacle was the highly dynamism of these kind of networks, and the major contribution of these works was to overcome the high speed of the vehicles. In conclusion, [18] suggested that mmWave can support vehicles speeds up to 140 km/h and the provided bit rate can reach 6.765 Gb/s. NOMA (Non-orthogonal multiple access) applicability for V2X was discussed in [3] and network slicing in [2].

Our work considers the most realistic situation where computation tasks are characterized in terms of processing time, deadline, and also an upload and download data size. All these properties are not predefined in any sense and assumed to be randomly distributed. The vehicle models are depicted by two randomly distributed values: its speed and its OBU processing capacity. We also consider the residence time of the vehicles in the range.

3 SYSTEM MODEL

As shown in figure 1, our system model consists of several base stations and road side units interconnected through a wired network in a dense urban area. Each of the RSUs is provided with LTE-based wireless communication capabilities that can utilize the 5G new radio (NR) channel ranges. We assume that system is using the first

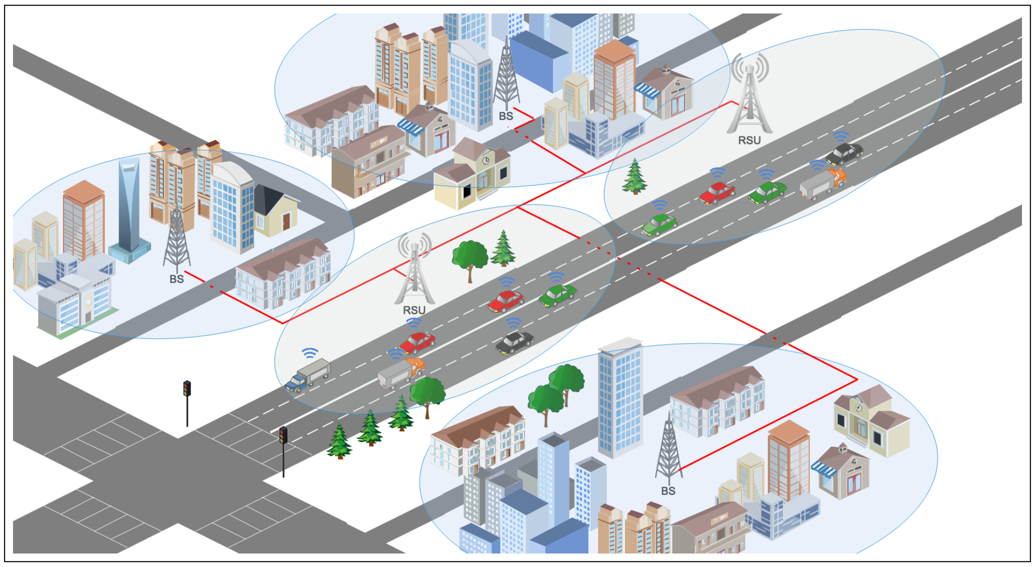

Figure 1: The system model

range of the new radios (≤ 5GHz) frequencies in its communications. To avoid any complication caused by the higher range (e.g., higher than 30 GHz) as these waves propagate solely through the line of sight and can be blocked by any obstacle. Each RSU receives computational tasks offloading requests through the wireless control channel, decides which task should be sent to which vehicle within its coverage range. These computational tasks might request a specific application that can be available on a certain vehicle. The aim of this work is to establish a framework that can receive tasks offloading requests, schedule them over certain available resource and returns the result before the task deadline.

3.1 Communication Model

Let B be the bandwidth assigned to a certain transmission, P_t be the transmission power, N_0 be 2 times the power spectral density, d be the distance between the source and destination, α be the path loss exponent and I be the interference. Then the bit rate can be calculated as follows:

$$rate(B) = B * log_2(1 + \frac{p_t * d^{-\alpha}}{I + N_0 * B}) \tag{1}$$

In a dense area, since the demand is usually high, the network designer usually decides to adopt the idea of micro-cells. Instead of providing one base station that covers the entire area, several low-power base stations can be distributed over the area splitting it into several micro-cells. In addition to increasing the frequency re-usability, this approach reduces the interference between micro-cells using the same channels. And by using efficient interference mitigation techniques, it is feasible to consider the interference effect to be negligible.

In order to provide an efficient scheduling scheme, it is necessary to provide the scheduler some flexibility on the resources allocation through time. A simple but inefficient technique is to lock a resource the entire time line with a certain task (i.e., reserving a certain channel with a certain bandwidth the entire time period for a given task). This will cause the resource to get wasted when task is not using it. Also, sometimes, it is preferable to reduce the amount of the resource used by a task in order to reassign this amount to another task. Such a dynamic resources allocation can be accomplished by splitting the time into several time units. By doing so, we can decide on the resources assignment and the amount assigned for the tasks in each of these time units. Another reason to split the time is the bit rate calculation. Notice that the bit rate is a function of the distance between the source and destination. Assume we do not split the time into time units. Let u be the data to be uploaded from a vehicle to the RSU. Then, the time to upload, $T_u = \frac{u}{r}$ is computed through the following integration:

$$r_i = \int_{t_0}^{T_u + t_0} r(t)dt \tag{2}$$

Here, t_0 is the time a vehicle starts uploading the task data. $r(t)$ contains the term $d(t)$, where $d(t) = \left[(a_x - t \times v)^2 + a_y^2\right]^{0.5}$. Here, v is the vehicle's speed, t is the time that has elapsed from the start of the upload time, and (a_x, a_y) are the 2D coordinates of the RSU, (0,0) being the entry point to the coverage of the RSU. This is a complex integration that can not be solved in online fashion. In order to avoid it, after splitting the time, we calculate the distance between the vehicle and the RSU in the middle of the time unit and calculate the rate accordingly.

3.2 Computation Model

Our system model assumes that there is a server available on each vehicle $j \in J$. The computational capacity of each vehicle is depicted by its CPU frequency f_j. For simplicity, we assume that each vehicle's OBU server contains only one CPU. Generalizing the work to include multi-core servers can be done easily, but will complicate the problem modeling without adding valuable elaboration. Each task $i \in I$ is characterized by five deterministic values: upload and download data sizes γ_i and σ_i respectively with their computation requirement c_i, weight ω_i, and deadline δ_i. The tasks are indivisible tasks, they can not be partitioned and should be computed by only one server. A server can not start computing a task unless it receives all the required data, and can not send back the calculated result unless it completes the task computation.

4 PROBLEM DEFINITION

The problem input is a set of tasks $I = \{1, ..., i, ..., I\}$ sent to the RSU and required to be offloaded to a set of J vehicles' OBU servers $V = \{1, ..., j, ..., J\}$. Task i requires a certain application available only on a subset of these servers, say J_i. The objective is to find the amount of frequency assigned on server j for task i, also the bandwidth $\alpha_{t,ij}$ (respectively $\beta_{t,ij}$) dedicated for task i to be uploaded (downloaded) to (from) vehicle j at time unit t, in order to maximize the weighted number of processed offloaded tasks. Note that the processing of a task can not start until all the required data is uploaded to the assigned vehicle. Similarly, a task cannot be downloaded until the processing of the task is finished. The distance between vehicle j and the RSU for task i at time unit t is indicated by d^t_{ij}.

Theorem 4.1. *The scheduling problem defined above is a NP-hard problem.*

Proof. Consider the well-known NP-hard scheduling problem $P|pmtn|\sum U_i$ [1], where there are N tasks with deadlines to be scheduled over M parallel identical machines in order to maximize the number of admitted tasks. It is easy to show that instances of this problem can be reduced to our problem. For each task in this problem, we may create a task for our problem having the same specifications without being implicitly downloaded or uploaded to a certain vehicle. Then we create M machines similar to the above known NP-hard problem to process maximum number of admitted tasks during the entire time line. Hence, our scheduling problem become similar to the NP-hard problem in [1]. □

In the next section, we mathematically formulate the problem as a mixed integer linear program, and due to the complexity, we propose a decomposition scheme based on Dantzig-Wolfe decomposition method in Section 6. This method divides the problem into a linear master problem and multiple pricing sub-problems, which can be solved in polynomial time through dynamic programming approach.

5 MATHEMATICAL MODEL

The list of symbols used in formulating the problem is listed in table 1. The objective is to maximize the number of admitted tasks :

$$maximize \sum_{i\in I}\sum_{j\in J_i} \omega_i x_{ij} \tag{Obj1}$$

Subject to the following constraints:

1) The sum of the computation resources assigned to the offloaded tasks in a vehicle can not exceed its maximum capacity.

$$\sum_{i\in I} f_{t,ij} \le f_j \qquad \forall j \in J \qquad \forall t \le T \tag{C1}$$

Table 1: Symbols used in formulating the problem

Symbol	Explanation
a) Parameters	
I	Set of offloaded tasks
J	Set of vehicles' OBU resources
J_i	Set of OBUs that can process task i
T	Maximum time segment
A	Total spectrum
c_i	Required number of clock cycles for task i
δ_i	Deadline of task i
γ_i	Upload data size of task i
σ_i	Download data size of task i
d^t_{ij}	Distance between vehicle j and the RSU
Δ	Time unit length
ω_i	Weight of task i
f_j	Maximum computation capacity of server j
N_0	Power spectral density
α	Path loss exponent
b) Variables	
$f_{ij} \in \mathbb{R}$	Frequency assigned for task i on vehicle j.
$us^t_i \in \{0,1\}$	Indicates whether task i is in upload stage at time t or not.
$ps^t_i \in \{0,1\}$	Indicates whether task i is in processing stage at time t or not.
$ds^t_i \in \{0,1\}$	Indicates whether task i is in download stage at time t or not.
$\alpha_{t,ij} \in \mathbb{R}$	Bandwidth assigned for task i to be uploaded to vehicle j.
$\beta_{t,ij} \in \mathbb{R}$	Bandwidth assigned for task i to be downloaded from vehicle j.
$x_{ij} \in \{0,1\}$	Indicates whether task i is assigned to vehicle j or not.
$rd_{t,ij} \in \mathbb{R}$	Download data rate of task i from vehicle j at time t.
$ru_{t,ij} \in \mathbb{R}$	Upload data rate of task i to vehicle j at time t

2) The computation resources assigned to a task from one vehicle should be sufficient to finish the task.

$$\sum_{t\le T} f_{t,ij} \times \Delta = x_{ij} c_i \qquad \forall i \in I \qquad \forall j \in J \tag{C2}$$

3) A task, if scheduled, is either in upload, compute, or download stage.

$$us^t_i + ps^t_i + ds^t_i = x_{ij} \qquad \forall i \in I \qquad \forall t < \delta_i \tag{C3}$$

4) The first time unit in the time segment of each task must be in the upload stage unless it is decided to be rejected.

$$us^1_i = x_{ij} \qquad \forall i \in I \tag{C4}$$

5) The last time unit before the deadline of a task must be in the download stage unless it is decided to be rejected.

$$ds^{\delta_i}_i = x_{ij} \qquad \forall i \in I \tag{C5}$$

6) In any time unit, a task can not be in an upload stage unless it was in the upload stage in the previous time unit.

$$us_i^t \leq us_i^{t-1} \qquad \forall i \in I \qquad : 2 \leq t \leq \delta_i \tag{C6}$$

7) In any time unit, a task can not be in a download stage unless it will be in the download stage in the next time unit.

$$ds_i^t \leq ds_i^{t+1} \qquad \forall i \in I \qquad \forall t < \delta_i \tag{C7}$$

8) After the deadline, no activities should be in progress.

$$us_i^t + ps_i^t + ds_i^t = 0 \qquad \forall i \in I \qquad \forall t \geq \delta_i \tag{C8a}$$

$$\sum_{\substack{i \in I \\ j \in J}} \beta_{t,ij} + \sum_{\substack{i \in I \\ j \in J}} \alpha_{t,ij} \leq A \qquad \forall t \leq T \tag{C8b}$$

9) In each time unit, a task can not be assigned a computation resource from a vehicle unless it is in the computation stage.

$$f_{t,ij} \leq f_j \times ps_i^t \qquad \begin{array}{l} \forall i \in I \\ \forall j \in J \\ \forall t \leq T \end{array} \tag{C9}$$

10) In each time unit, a task can not be assigned a bandwidth to download data unless it is in the download stage.

$$\beta_{t,ij} \leq A \times ds_i^t \qquad \begin{array}{l} \forall i \in I \\ \forall j \in J \\ \forall t \leq T \end{array} \tag{C10}$$

11) In each time unit, a task can not be assigned a bandwidth to upload data unless it is in the upload stage.

$$\alpha_{t,ij} \leq A \times us_i^t \qquad \begin{array}{l} \forall i \in I \\ \forall j \in J \\ \forall t \leq T \end{array} \tag{C11}$$

12) The transmission rate for task i to download data from vehicle j to RSU is calculated by:

$$r_{ij}^{dt} = \beta_{t,ij} \times log\left[1 + \frac{p_i \times (d_{ij}^t)^{-a}}{\beta_{t,ij} \times N_0}\right] \tag{C12}$$

The upload transmission rate r_{ij}^{ut} can be calculated similarly.

13) For each task, the bandwidth should be assigned for the entire data to be uploaded or downloaded.

$$\sum_{t \leq T} ru_{t,ij} \times \Delta = \gamma_i \times x_{ij} \qquad \forall i \in I \qquad \forall j \in J \tag{C13a}$$

$$\sum_{t \leq T} rd_{t,ij} \times \Delta = \sigma_i \times x_{ij} \qquad \forall i \in I \qquad \forall j \in J \tag{C13b}$$

To linearize the problem, we apply a well-known technique that is based on the calculation of the tangent of the bit rate function.

6 DANTZIG-WOLFE DECOMPOSITION

Dantzig-Wolfe decomposition technique is a utilization of an ILP property that several constraints of a problem contain only a subset of the variables (Blocks structure). These constraints, if separated, define an easy-to-solve subproblem. The approach uses the presentation theorem of a linear programming to encode all the feasible points of the original problem as an affine combination of the subproblem feasible points.

Now, looking into our problem model, we can infer that only two constraints that have variables of different tasks ($C1$ and $C8b$). All the other constraints contain variable of one task. Hence, using Dantzig-Wolfe decomposition, we can decompose the problem into one master problem and N pricing subproblems.

Let X^i be the set of points that satisfies all the constraints of task i except constraints in the sets $C1$ and $C8b$. Then any feasible solution for the problem can be written as affine combinations of the points in x_{ij} under the condition that it satisfies $C1$ and $C8b$. Let K_i be the number of integer points in the task i feasible space. Let $\mathbf{x}_i^k$ be feasible integer point in task i feasible space. Then the problem model can be re-written as follows:

$$maximize \sum_{i \in I} \sum_{j \in J_i} \omega_i \sum_{k \in K_i} \lambda_i^k x_{ij}^k$$

$$s.t.$$

$$\sum_{i \in I} \sum_{k \in K_i} \lambda_i^k f_{t,ij}^k \leq f_j \qquad \forall j \in J \quad \forall t \leq T \tag{C1'}$$

$$\sum_{\substack{i \in I \\ j \in J}} \sum_{k \in K_i} \lambda_i^k \left(\beta_{t,ij}^k + \alpha_{t,ij}^k\right) \leq A \qquad \forall t \leq T \tag{C8b'}$$

$$\sum_{k \in K_i} \lambda_i^k = 1 \qquad \forall i \in I \tag{Ca}$$

$$\lambda_i^k \in \{0, 1\}$$
$$\mathbf{x}_i^k \in X_i$$

As there are exponential number of points in each X_i, an efficient way to solve the problem is by relaxing the variables to linear ones and solve the problem through column generation. The next subsection will discuss our CG approach.

7 THE COLUMN GENERATION ALGORITHM

As discussed in the previous section, the possible number of columns of the master problem is exponential. The basic idea of column generation algorithm is to avoid including all the possible columns of a problem in the master model tableau. This is done by making the optimization of the master program to calculate the dual variables and feed it to the pricing subproblems. The role of the subproblems is to generate the columns with the minimum reduced cost using the given dual values. When the master model converges according to a certain criteria, the algorithm starts to solve the integer version of the master problem.

7.1 Initial solution

We chose to provide an initial solution with a constructive greedy heuristic. The algorithm starts by sorting the tasks according to their processing times in ascending order. Then for each task, it sorts the vehicles' OBUs according to their available computation resources. Then for each server, it tries to assign it that task . If it succeeds, it considers the task being scheduled. Otherwise, it resets the resources as that the task was not scheduled. The complexity of the algorithm is $O(|I|\big(|J|log|J| + |J|T\big))$.

7.2 Solving the Subproblems

Let ψ_{jt} be the dual variable corresponding to a constraint in the set $C1$, ϕ_t be the dual variable corresponding to constraint in set $C13a$, and ζ_i be the dual variable of constraint Ca for task i. Then the column generation pricing subproblem for each task can be modeled as follows:

$$\begin{aligned} minimize \sum_{\substack{j\in J\\ t\le T}} \psi_{jt} f_{t,ij} \\ + \sum_{t\le T} \phi_t \sum_{j\in J} \big(\alpha_{t,ij} + \beta_{t,ij}\big) \\ - \omega_i x_{ij} \\ + \zeta_i \\ s.t. \quad \mathbf{x}_{ij} \in X_i \end{aligned}$$

By looking into the subproblem objective function and constraints, we can state the subproblem as follows: *Given set of time units each with its own weights for being used only to upload, download or compute the task, specify the process (uploading, downloading or computing) and the amount of resource used in each time unit in order to minimize the total weight.* The solution space of this problem is polynomial in size. Hence, it is possible to solve the problem using a dynamic programming approach that can efficiently find the optimum solution without the need of any branch-and-bound based algorithm. We omit the details due to the limitation of the space.

8 NUMERICAL RESULTS

In this section we analyze and evaluate the performance of our proposed Dantzig-Wolfe decomposition method. First, we study the scheduling performance of our method versus the solution obtained from the MILP by considering the execution time and task rejection rate. We then study the overall weighted rejection rate for the whole system by considering different criteria such as: average number of cycles, task upload data size, and OBU server capacity. Finally, we compare our decomposition method with the greedy heuristic method by varying the same criteria mentioned above. We use CPLEX to solve our optimization model (MILP), and C++ to simulate the operation of our decomposition method. We generate results on CPU with Intel(R) Core(TM) i7-6700 CPU @ 2.7GHz, 16GB memory ram and 64-bit mac operating system. The simulation parameters with their values required to run our algorithms are shown in Table 2.

Table 2: Measurable factors used for the scheduling performance

Factors	Distribution	Mean	Variance
Tasks Arrival (tasks/s)	Exponential	200	-
Servers Capacity (GHz)	Gaussian	1	0.2
vehicle's Velocity (KM/H)	Trunc. Gaussian	50	40
Delay tolerant deadline (s)	Gaussian	0.1	0.03
Upload Data Size (KB)	Gaussian	500	1
Download Data Size (KB)	Gaussian	100	1
Tasks Weight	Gaussian	5	3
Number of Cycles (Millions)	Gaussian	20	5
Total Spectrum (GHz)	-	3	-

We let both vehicle and task arrivals follow Poisson process. As in traffic theory, the vehicle's speed follows a truncated Gaussian distribution. All task's specs are assumed to follow the Gaussian distribution. To generate problem instances, we simulate the vehicles arrival process until the system reaches the steady state. To reach the required number of vehicles, we choose the arrival rate that can easily reach such a number in 1000-meter range and keep running the simulation until the required number of vehicles is acheived.

Figures 2 and 3 show the execution time and rejection rate respectively of the MILP and Dantzig-Wolfe decomposition method for different number of arrival vehicles by varying the number of requested tasks. We limited the number of iterations for the Dantzig-Wolfe algorithm to 100 iterations. Note that if the linear relaxation of the master problem for the Dantzig-Wolfe decomposition method converges before 100 iterations, the algorithm, at that point, starts solving the integer version of the master problem model. Figure 2 shows that the Dantzig-Wolfe method is always surpassing the MILP in terms of execution time. For instance, when the number of vehicles is five and the number of requested tasks is ten, the MILP takes 20 seconds to execute, while the Dantzig-Wolfe method takes only 4 seconds. Whereas, for the same number of vehicles but for 15 tasks, the MILP takes 136 seconds compared to 20 seconds for the Dantzig-Wolfe method. It should be noted that the MILP failed to solve for more than 15 tasks for any number of arrived vehicles. For example, when the number of vehicles is 15 and the number of tasks is 35, the MILP after running for 20 minutes crashed due to memory failure. While, the Dantzig-Wolfe method toke only 315 seconds to obtain a feasible solution. As shown in Figure 3, the Dantzig-Wolfe shows high efficiency in solving large instances in terms of the rejection rate. For instance, for 5 vehicles and 35 tasks, the MILP failed to find a feasible solution, while the Dantzig-Wolfe method obtained a feasible solution with rejection rate of 24%. Another example is when the number of vehicles and tasks are 25 and 35 respectively, the rejection rate of the Dantzig-Wolfe method is less than 1%. The figure, for those numbers of tasks that the MILP could give feasible solutions (i.e., 5, 10, and 15 tasks), shows that the Dantzig-Wolf method rejects tasks at most 4% higher than the MILP.

In the remaining evaluation, since the MILP cannot obtain feasible solution for large instances, we utilize the greedy heuristic method used in generating initial solution for the Dantzig-Wolfe

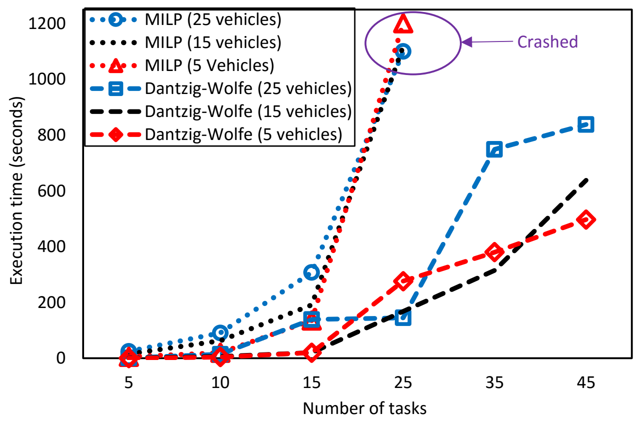

Figure 2: Execution time of the MILP and Dantzig-Wolfe decomposition method for different number of arrival vehicles by varying the number of requested tasks.

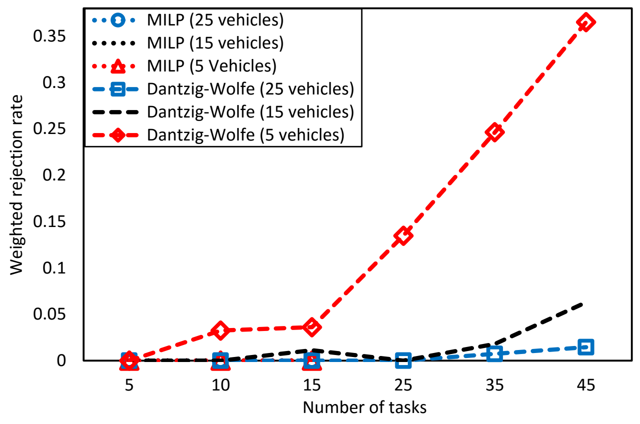

Figure 3: Evaluation of weighted rejection rate for our proposed methods by varying the number of tasks.

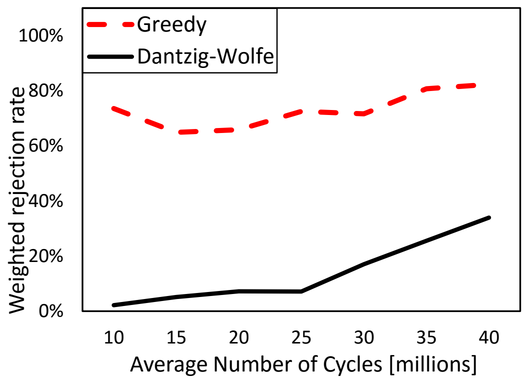

Figure 4: Rejection rate versus average number of cycles.

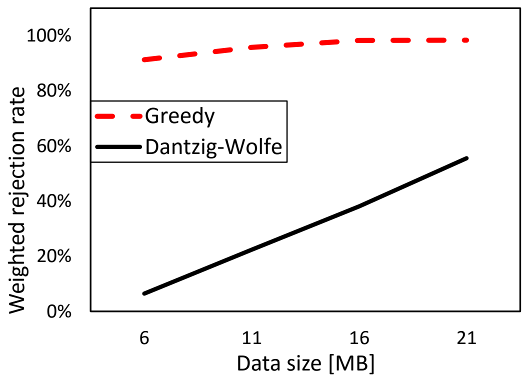

Figure 5: Rejection rate versus upload data size.

decomposition method. In Figure 4, we vary the average number of cycles for the system in order to study the impact of high computation-intensive tasks over the proposed scheduling methods. In this figure, the number of vehicles and tasks are considered 10 and 35 respectively. As shown in the figure, the Dantzig-wolfe outperforms the greedy method with lowest rejection rate of 2% and highest rejection rate of 34%. As expected, the rejection rate increases as the average number of cycles increases. The trend line becomes sharper when the number of cycles reaches 25 millions (the trend is almost flat before that while its slope is 2.7% after 25 millions). The reason can be explained in the OBUs capacity limitation which is been filled with the increasing number of cycles.

In order to study the effect of the data size on the system's rejection rate, in Figure 5, we increase the average uploaded data size of requested tasks. The number of vehicles and tasks used in this study is similar to Figure 4. To clearly illustrate the data size impact, we reduce the available radio spectrum to 0.5 Mhz. As shown in the figure, by increasing the uploaded data size, the rejection rate increases as well. Even when the rejection rate increases to more than 50%, the difference between the greedy and Dantzig-Wolfe performance stays significant. For example, for the average upload data size of 11 MB, the Dantzig-Wolfe method rejects 22% of the tasks, while the greedy method rejects around 96% of the tasks. For the larger size, say 21 MB, the Dantzig-Wolfe method rejects 56%, while the greedy method rejects upto 98%.

Figure 6 depicts the effect of increasing the size of OBU execution capacity. As expected, the rejection rate decreases with the increasing of the average computation speed of the OBU's servers. The performance of the greedy method stays low even with a high computation speed (the best case is around 60% rejection rate), while the rejection rate of the Dantzig-Wolfe method is 2%.

9 CONCLUSION

Fog computing is a promising paradigm that will allow an efficient utilization of computation resources available and accessible through wireless communication. 5G technologies with its low latency and high reliability promises can provide a platform to enable fog computing establishment upon various kinds of resources including smart vehicles. In this work, we proposed a system that can handle computation requests over vehicular network through

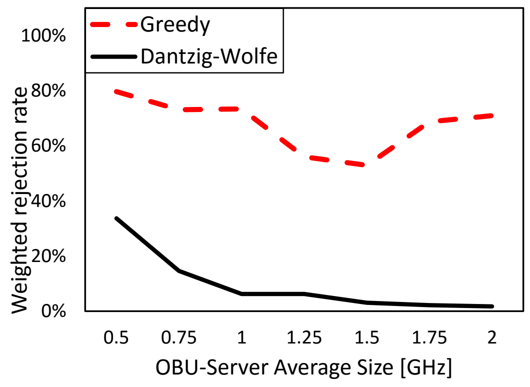

Figure 6: Rejection rate versus OBU server average size.

an efficient scheduling scheme that considers the available vehicle OBU computation servers and the limited radio spectrum in order to efficiently allocate them for the requested computational tasks. The problem was formulated as an MILP with the objective to maximize the weighted number of admitted tasks. Although the problem is an NP-hard one, we proposed a scalable decomposition scheme based on Dantzig-Wolfe scheme, which resulted in polynomial-time solvable subproblems and a linear master problem. The approach showed an efficient and scalable performance compared to a greedy heuristic and MILP. Several parameters were considered in the evaluation demonstrating the robustness of our approach to solve various kinds of the problem instances.

REFERENCES

[1] Peter Brucker. 2004. *Scheduling Algorithms.* SpringerVerlag.
[2] Claudia Campolo, Antonella Molinaro, Antonio Iera, and Francesco Menichella. 2017. 5G Network Slicing for Vehicle-to-Everything Services. *Wireless Commun.* 24, 6 (Dec. 2017), 38–45.
[3] B. Di, L. Song, Y. Li, and Z. Han. 2017. V2X Meets NOMA: Non-Orthogonal Multiple Access for 5G-Enabled Vehicular Networks. *IEEE Wireless Communications* 24, 6 (Dec 2017), 14–21. https://doi.org/10.1109/MWC.2017.1600414
[4] J. Feng, Z. Liu, C. Wu, and Y. Ji. 2017. AVE: Autonomous Vehicular Edge Computing Framework with ACO-Based Scheduling. *IEEE Transactions on Vehicular Technology* 66, 12 (Dec 2017), 10660–10675. https://doi.org/10.1109/TVT.2017.2714704
[5] A. Kousaridas, D. Medina, S. Ayaz, and C. Zhou. 2017. Recent advances in 3GPP networks for vehicular communications. In *2017 IEEE Conference on Standards for Communications and Networking (CSCN).* 91–97. https://doi.org/10.1109/CSCN.2017.8088604
[6] Salman Memon and Muthucumaru Maheswaran. 2019. Using machine learning for handover optimization in vehicular fog computing. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing.* ACM, 182–190.
[7] Zhaolong Ning, Jun Huang, and Xiaojie Wang. 2019. Vehicular fog computing: Enabling real-time traffic management for smart cities. *IEEE Wireless Communications* 26, 1 (2019), 87–93.
[8] Jéferson Campos Nobre, Allan M de Souza, Denis Rosario, Cristiano Both, Leandro A Villas, Eduardo Cerqueira, Torsten Braun, and Mario Gerla. 2019. Vehicular software-defined networking and fog computing: integration and design principles. *Ad Hoc Networks* 82 (2019), 172–181.
[9] Mansoor Shafi, Andreas F. Molisch, Peter J. Smith, Thomas Haustein, Peiying Zhu, Prasan De Silva, Fredrik Tufvesson, Anass Benjebbour, and Gerhard Wunder. 2017. 5G: A tutorial overview of standards, trials, challenges, deployment, and practice. *IEEE Journal on Selected Areas in Communications* 35, 6 (2017), 1201–1221. https://doi.org/10.1109/JSAC.2017.2692307
[10] G. H. Sim, S. Klos, A. Asadi, A. Klein, and M. Hollick. 2018. An Online Context-Aware Machine Learning Algorithm for 5G mmWave Vehicular Communications. *IEEE/ACM Transactions on Networking* (2018), 1–14. https://doi.org/10.1109/TNET.2018.2869244
[11] Yuxuan Sun, Jinhui Song, Sheng Zhou, Xueying Guo, and Zhisheng Niu. 2018. Task Replication for Vehicular Edge Computing: A Combinatorial Multi-Armed Bandit based Approach. *CoRR* abs/1807.05718 (2018).
[12] Andrea Tassi, Malcolm Egan, Robert J. Piechocki, and Andrew R. Nix. 2017. Modeling and Design of Millimeter-Wave Networks for Highway Vehicular Communication. *IEEE Transactions on Vehicular Technology* 66 (2017), 10676–10691.
[13] Xuyu Wang, Shiwen Mao, and Michelle X. Gong. 2017. An Overview of 3GPP Cellular Vehicle-to-Everything Standards. *GetMobile: Mobile Comp. and Comm.* 21, 3 (Nov. 2017), 19–25.
[14] Xuyu Wang, Shiwen Mao, and Michelle X. Gong. 2017. An Overview of 3GPP Cellular Vehicle-to-Everything Standards. *GetMobile: Mobile Comp. and Comm.* 21, 3 (Nov. 2017), 19–25.
[15] Wikipedia contributors. [n.d.]. 5G — Wikipedia, The Free Encyclopedia. [Online; accessed 29-August-2018].
[16] Xuefeng Xiao, Xueshi Hou, Xinlei Chen, Chenhao Liu, and Yong Li. 2019. Quantitative analysis for capabilities of vehicular fog computing. *Information Sciences* (2019).
[17] Shi Yan, Xinran Zhang, Hongyu Xiang, and Wenbin Wu. 2019. Joint Access Mode Selection and Spectrum Allocation for Fog Computing Based Vehicular Networks. *IEEE Access* 7 (2019), 17725–17735.
[18] H. Zhou, W. Xu, Y. Bi, J. Chen, Q. Yu, and X. S. Shen. 2017. Toward 5G Spectrum Sharing for Immersive-Experience-Driven Vehicular Communications. *IEEE Wireless Communications* 24, 6 (Dec 2017), 30–37. https://doi.org/10.1109/MWC.2017.1600412
[19] Sheng Zhou, Yuxuan Sun, Zhiyuan Jiang, and Zhisheng Niu. 2019. Exploiting Moving Intelligence: Delay-Optimized Computation Offloading in Vehicular Fog Networks. *arXiv preprint arXiv:1902.09401* (2019).
[20] Zhenyu Zhou, Pengju Liu, Junhao Feng, Yan Zhang, Shahid Mumtaz, and Jonathan Rodriguez. 2019. Computation Resource Allocation and Task Assignment Optimization in Vehicular Fog Computing: A Contract-Matching Approach. *IEEE Transactions on Vehicular Technology* (2019).