Cost-Effective Sensor Data Collection from Internet-of-Things Zones Using Existing Transportation Fleets

Fangqi Liu[1], Quixi Zhu[1], Md Yusuf Sarwar Uddin[1], Cheng-Hsin Hsu[2], and Nalini Venkatasubramanian[1]

[1]Department of Computer Science, University of California Irvine, CA

[2]Department of Computer Science, National Tsing Hua University, Taiwan

***Abstract*—Modern IoT devices are equipped with media-rich sensors that generate a heavy burden to local access networks. To improve efficiency of data collection, we introduce the concept of "IoT zones" as geographically-correlated clusters of local IoT devices with well connected wireless networks that may have limited access to the Internet. We develop techniques to create a cost-effective data collection network using existing transportation fleets with predefined schedules to collect sensor data from IoT zones and upload them at locations with better network connectivity. Specifically, we provide solutions to the upload point placement and upload path planning problems given tradeoffs between collection quality, timing needs (QoS), and installation cost. We evaluate our approaches using a real-world bus network in Orange County, CA and study the applicability and efficiency of the proposed method as compared to several other approaches. The trace-driven simulations reveal that our best-performing algorithm significantly outperforms others, e.g., in one of the scenarios with 160 total cost, our algorithm achieves sub-21 sec data transfer time (30+ times improvement), sub 3.2% late delivery ratio (about 14 times improvement), and above 96% data delivery ratio (about 50% improvement). In addition, it achieves the above performance without excessive installation cost: even when a cost limit of 640 is given, our algorithm opts for a solution with about 160 total cost (versus 640 from others).**

I. Introduction

Internet-of-Things (IoT) deployments are giving rise to smart cities and communities worldwide. The next generation urban planning is moving towards the design of smart instrumented spaces beneficial to citizens. For example, the City of Barcelona is exploring the concept of *superblocks* to limit transportation within cityblocks [1]. A large number of interesting urban smart greenspaces efforts are being initiated in cities throughout the US [2], the goal being to support increased environmental sustainability and improved quality of life for citizens. Studies [3] also point out that the market for short-range IoT wireless technologies (Wi-Fi, BlueTooth, Zigbee) has overtaken those for long-range connectivity (LoRa, Sigfox) which can probably be attributed to the lower hardware and spectrum costs associated with the short-range networks. In the future, we envision that such innovative design will give rise to localized **IoT zones**, where geographically-correlated clusters of IoT devices are interconnected via a variety of wireless networks (short and long-range).

Modern IoT devices are often equipped with media-rich sensors, such as microphones, cameras, and Radar/Lidar sensors, which generate tremendous volumes of sensor data. Data obtained from these devices must be fused, analyzed and interpreted – solutions that require on-board storage, computation, and communication for complex analytics in every device is prohibitively expensive. The ability to create local collection points is plausible within a zone using M2M technologies and simple analytics can be executed at these network edges. Enabling high bandwidth networks to backhaul data at the device level to big data processing backends for more comprehensive analysis is both expensive and difficult. Communications of large IoT sensor data, such as surveillance camera footage (for further analysis or archival) or real-time social media sharing is challenging since access networks usually have limited bandwidth and are vulnerable to network congestion. This paper deals with the problem of getting data from local edges to the backend cost-effectively.

Our proposed solution is to create a hierarchy where end-devices communicate to a local node that we refer to as a **Rendezvous Point (RP)**. Creation of such local edge components or RPs is becoming increasingly possible, e.g. through smart streetlights, smart transit stops, etc. Data make their way from the local RP to a larger processing backend through intermediate **Upload Points (UPs)** scattered at suitable locations where better network connectivity and bandwidths are available. Two key questions need to be answered: (a) where should Upload Points (UPs) be located (*upload point placement problem*) and (b) how do data to be uploaded get from local RPs to UPs (*upload path planning problem*)?

Our key intuition is to exploit existing transport possibilities to implement this hierarchy, i.e. by leveraging city transit infrastructures that currently transport people to now transport data from devices to a big data processing backend. Such traffic fleets with predefined schedules (routes and times) are common in urban communities today, which include buses, mail vehicles, and garbage trucks with regular schedules and stops. We plan to leverage fixed city transit infrastructures, in particular, bus stops (or traffic lights) as potential RPs and UPs in our hierarchy. Since existing transportation fleets (e.g. city buses) have predefined schedules and paths, we will design solutions where data gathering (from RPs), transport (to UPs), and upload (from UPs to backend) can be made to occur along these paths. A rationale solution is to colocate RPs and UPs with transit stops, because vehicles are typically required to stop at these points, providing the needed time for reliable wireless transmission of data to/from vehicles.

The following are key contributions of this paper.

- We develop an integrated approach to address the problems of *upload point placement* and *upload path planning* (Sec. II).

- We develop a modeling framework and formulate the upload point placement problem, which is NP-hard (Sec. III).
- We develop a greedy heuristic for optimized upload point placement that trades of the spatial coverage, upload deadlines, and deployment feasibility for the target IoT zones under an installation cost limit (Sec. IV).
- We develop an upload path planning algorithm that utilizes the upload point placement map generated in the previous step for minimizing delay, given an IoT deployment with data generation patterns, communication needs, and vehicle schedules (Sec. IV).
- We validate our approach and compare it to a variety of solution choices using real-world transportation networks–i.e. road map and transit schedules for Orange County, CA (Sec. V).

II. Sample Scenario and Problem Statements

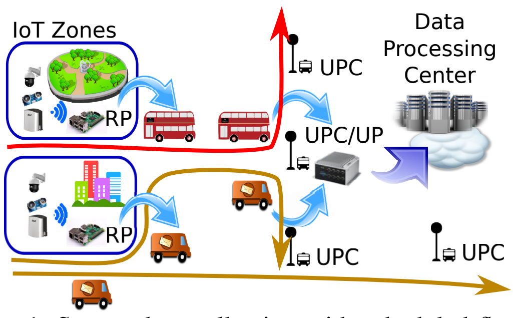


Fig. 1: Sensor data collection with scheduled fleets.

Fig. 1 is a simplified usage scenario with two IoT zones (park, business district) each with devices that are well connected, and communicate sensor data to local RPs. RPs possess sufficient storage to buffer incoming sensor data, until a scheduled pickup vehicle on a *trip* arrives. Sensor data items are associated with a *delay tolerance*, i.e. a tolerable time difference between arriving at RPs and at a *data processing center*. Each vehicle contains a computing device with storage; vehicles in transit pick up data buffered at RPs and transport them to an *Upload Point* (UP) from where sensor data is delivered to data processing center. Different from RPs, whose locations are given, UPs can be set up at various locations, such as bus stops, road intersections, street lamps, and roadside buildings. We refer to the possible locations for setting up UPs as *Upload Point Candidates* (UPCs). Every UPC has its own *installation cost*, which depends on practical factors, such as type of access network technologies, distance to Internet access points, and availability of power supply. We address two core research problems in this setting:

- *Upload point placement.* Selection of an optimal set of UPs from all possible UPCs to ensure spatial coverage and satisfy delay tolerance, under a budget of UP installation cost.
- *Upload path planning.* Plan for each vehicle and RP that schedules for pick-up and drop-off of data gathered at RPs, given the transportation fleet schedules.

In Fig. 1, among the three potential UPCs, the center candidate maximizes the number of RPs served with lower delays and is hence selected as an upload point. In practice, the choice of a upload point will take into consideration installation costs of individual UPCs and transit vehicle paths and schedules. Several usage scenarios of this infrastructure can be envisioned. Consider public parks instrumented with surveillance cameras and environmental sensors that communicate data to a local collection point; data are transported to suitable upload points for further analysis to provide live situational awareness of public spaces (occupancy), detect public safety threats (fires, protests) or to promote community recreational events. Similarly, residents in assisted living facilities can send personal health and community activity related information to data processing centers via mail/carrier trucks.

Related Work: We discuss key related work in delay tolerant networking (DTN), wireless mesh networks, vehicular nets and community-scale data collection. DTNs and data mules have exploited mobility for data transfer [4], [5], [6]; these approaches often assume very limited access to communication infrastructure and aim to meet data transfer deadlines using multi-hop networks [7]. Techniques for proactive and adaptive data transmission in this setting have been designed with crowdsensing applications and energy efficiency [8], [9], [10], [11], [12]. Similar work in the context of WSN makes use of mobile sinks [13], [14], [15] for improved communication. Techniques to exploit public bus transportation for DTN-based data dissemination include utilizing busline patterns [16], accounting for encounter frequency of bus routes [17] and using bus stops as communication relays.

Planning and deployment are often network specific; techniques for AP and gateway placement in wireless mesh networks [18], [19] are useful for community scale networking. Modeling using set cover based formulation [20], and techniques for network deployment [21] and performance [22] have been explored. Facility placement mechanisms for data upload in delay-tolerant crowdsensing and content distribution have been formulated and studied [23], [24]. Combining facility placement with transport logistics [25] for an integrated provisioning is an approach similar to ours, albeit for goods transport. Related literature from the vehicular networking community includes work on deployment and operation of vehicular networks [26], their graph properties [27], and methods for more realistic modeling of mobility [28], [29], [30]. More recent efforts have explored the use of public transport to collect IoT data in cities [31], [32]. Other related efforts include approaches to utilize Wi-Fi-enabled buses for non-urgent communications [33] and geocast based mechanisms to improve delivery reliability and timing [34], [35] and monitor the urban environment [36].

Much of the earlier literature uses statistical estimates of traffic flow and encounter probabilities. In contrast, we leverage the knowledge of exact routes and transit schedules (especially in urban settings) to generate data routing plans and handle heterogeneity of IoT traffic while satisfying communication QoS needs.

III. Problem Formulation

In this section, we formulate our upload point placement and upload path planning into one combined formulation.

A. Symbols and Notations

The transportation/transit fleet is described by a set of *trips*, $\mathbf{V}$, where each trip v denotes a bus/vehicle moving through a sequence of *stop points* (i.e., bus stops). When the vehicle reaches its terminal stop and starts again, it is considered as a different trip even though it may be the same vehicle/bus. Let $\mathbf{N}$ be the set of all stops where all these trips go by. Since we assume all data gets accumulated in these stop points and data also gets uploaded via them, this set, $\mathbf{N}$, is the superset of all RPs and UPCs as well. We, in fact, can eliminate those stops from $\mathbf{N}$ that are neither RPs nor UPCs because their presence do not affect our placement and planning. In that, $\mathbf{N}$ becomes the set of all RPs and UPCs. For each stop point n, let R_n, D_n, and C_n be the data rate, delay tolerance, and installation cost of n, respectively ($R_n > 0$ and $D_n > 0$ if n is an RP, otherwise 0, and $C_n = \infty$ if n is not a UPC).

We assume that the schedule of all trips are known apriori. Hence, we denote $A_{n,v}$ as the arrival time of trip v at stop point n ($A_{n,v} = -1$ if trip v does not go by n). The schedule matrix, $\mathbf{A} = [A_{n,v}]$ contains the complete schedule of all trips. The schedule holds for a certain duration (e.g., for a day) and then perhaps repeats itself. We assume that IoT data collection happens sometime within this interval and let T_s and T_e be the start and end time of this data collection interval.

For the *upload point placement* problem, we define a binary array $\mathbf{Y} = \{y_1, y_2, \ldots, y_N\}$ to denote UP placement. In particular, stop point $m \in \mathbf{N}$ is chosen as an UP iff $y_m = 1$. For the *upload path planning* problem, we define a three-dimensional auxiliary matrix $\mathbf{X}$ of size $N \times (V+1) \times N$, where $x_{n,v,m} = 1$ iff trip $v \in \mathbf{V}$ is used to carry data from stop point (RP) n to another stop point (UP) m; and $x_{n,v,m} = 0$ otherwise. Moreover, we use index $v = 0$ to capture the corner cases where an RP/UPC is selected as a UP. In that case, the sensor data are directly uploaded by the UP whenever they reach the RP. Concretely, we let $x_{n,0,m} = 1$ iff $n = m$ and $y_m = 1$; $x_{n,0,m} = 0$ otherwise. In the following, if not otherwise stated, we use n to denote an RP stop point, v for trip, and m for a UPC stop point.

We note that $m \in \mathbf{N}$ should be selected as a UP in $\mathbf{Y}$, if at least one vehicle trip decides to dump sensor data at m in $\mathbf{X}$. This is, $y_m = 1$ if and only if there exists at least one $x_{n,v,m} = 1$ for some n and v. Moreover, each vehicle trip that picks up data at n uploads the data at a single UP. That is, $\sum_m x_{n,v,m} \in \{0, 1\}$ for each n and v.

Data Transfer Time: As data chunks are moved from an RP to a UP via a trip, the chunk experiences some delay. For each $x_{n,v,m} = 1$, there is an associated data transfer time, $d_{n,v,m}$, that denotes the amount of time it takes to transfer data from RP n to UP m via trip v. The transfer time is a function of $\mathbf{X}$ and it has two parts: wait time (waiting for the bus to arrive at the RP) and travel time (the time to reach the planned UP). The wait time depends on when was the last time data was carried by any trip passing by this stop point (since then, the RP is waiting for a bus to show up). Let $B_{n,v}(\mathbf{X})$ be last pick up time preceding trip v. This is:

$$B_{n,v}(\mathbf{X}) = \max\{A_{n,v'} \sum_m x_{n,v',m} | A_{n,v'} < A_{n,v}\}. \quad (1)$$

If no preceding trip exists, $B_{n,v}$ is set to T_s. So, the wait time becomes $A_{n,v} - B_{n,v}(\mathbf{X})$. Once loaded into bus trip v, the travel time to each UP m is given by: $A_{m,v} - A_{n,v}$. Adding the above two terms and canceling the common term, we obtained the data transfer time as follows:

$$d_{n,v,m}(\mathbf{X}) = A_{m,v} - B_{n,v}(\mathbf{X}). \quad (2)$$

Data volume: The volume of data carried by each vehicle trip affects the decisions, in the sense that the on-time delivery of a vehicle trip carrying more data should be more critical. We let $S_{n,v}(\mathbf{X})$ be the data volume carried by vehicle trip v from stop point n, which can be written as:

$$S_{n,v}(\mathbf{X}) = R_n[\min(T_e, \max(A_{n,v}, T_s)) - \min(T_e, B_{n,v}(\mathbf{X}))] \sum_m x_{n,v,m}. \quad (3)$$

Here, $\min(T_e, \max(A_{n,v}, T_s))$ and $\min(T_e, B_{n,v}(\mathbf{X}))$ represent the last and first sensor data bits that are buffered at RP n, which will be picked up by vehicle trip v. The right-most summation is a binary value indicating if trip v picks up sensor data at stop point n or not.

Penalty function: The overall objective of data collection and transfer is to upload as much data as possible with lower transfer delay. Hence, we introduce the notion of a *penalty* function that accounts for both the volume of data as well as the delay the transfer experiences. That is, the data chunks transferred with larger delays incur more penalty than the ones that are transferred with smaller delays. There is also a penalty for data being not uploaded at all at the end of the operation (after T_e). Therefore, the overall penalty, denoted as $\hat{P}(\mathbf{X})$, measures how good a certain upload plan X is and thereby quantifies the service quality of sensor data collection. The penalty function is the sum of the following two terms:

Penalty due to transfer delay: This part measures the total accumulated transfer delay weighted by the respective volume of data, which is defined as:

$$P_l(\mathbf{X}) = \frac{1}{V} \sum_{n,v,m} f_n(d_{n,v,m}(\mathbf{X})) \cdot S_{n,v}(\mathbf{X}) \cdot x_{n,v,m}, \quad (4)$$

where $V = (T_e - T_s)\sum_n R_n$ is the total volume of data generated and $f_n(d)$ is used to normalize delay within [0, 1], which is defined as $f_n(d) = 1 - \exp\left(\frac{d}{D_n}\right)^3$ with $f_n(0) = 0$ and $f_n(\infty) \to 1$.

Penalty due to data not being uploaded: This part accounts for sensor data being left on RPs beyond the end of each schedule (say, overnight). Let L_n be the last time (within T_e) when data is carried from stop point n by any trip. That is, $L_n(\mathbf{X}) = \max\{A_{n,v} \sum_m x_{n,v,m}\}$. We have:

$$P_u(\mathbf{X}) = \frac{1}{V} \sum_n R_n \cdot (T_e - L_n(\mathbf{X})) \cdot (1 - x_{n,0,n}). \quad (5)$$

The term $(1 - x_{n,0,n})$ takes care of the corner cases where an RP is also chosen as a UP.

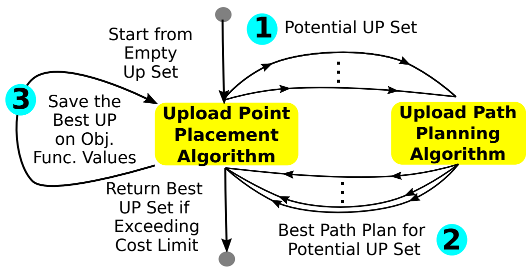
B. Formulation

We write the upload point placement and upload path planning problem (finding $\mathbf{Y}$ and $\mathbf{X}$ simultaneously) as follows:

$$\min \quad \hat{P}(\mathbf{X}) = P_l(\mathbf{X}) + P_u(\mathbf{X}) \tag{6a}$$
$$\text{s.t.} \quad x_{n,v,m} \le y_m, \forall n \in \mathbf{N}, v \in \mathbf{V} \cup \{0\}, m \in \mathbf{N}; \tag{6b}$$
$$A_{n,v} x_{n,v,m} \ge 0, \forall n, m \in \mathbf{N}, v \in \mathbf{V}; \tag{6c}$$
$$A_{m,v} x_{n,v,m} \ge 0, \forall n, m \in \mathbf{N}, v \in \mathbf{V}; \tag{6d}$$
$$(A_{m,v} - A_{n,v}) x_{n,v,m} \ge 0, \forall n, m \in \mathbf{N}, v \in \mathbf{V}; \tag{6e}$$
$$\sum_{m \in \mathbf{N}} x_{n,v,m} \le 1, \forall n \in \mathbf{N}, v \in \mathbf{V}; \tag{6f}$$
$$\sum_{m \in \mathbf{N}} y_m C_m \le \Theta; \tag{6g}$$
$$x_{n,v,m} \text{ and } y_n \in \{0,1\}, \forall n \in \mathbf{N}, v \in \mathbf{V} \cup \{0\}, m \in \mathbf{N}. \tag{6h}$$

The objective function in Eq. (6a) is to minimize the penalty function value. The constraints in Eq. (6b) connect $\mathbf{Y}$ with $\mathbf{X}$, so that $y_m = 1$ if *at least* a vehicle trip $v \in \mathbf{V} \cup \{0\}$ is determined to pick up sensor data at n, i.e., $x_{n,v,m} = 1$; $y_m = 0$ otherwise. The constraints in Eqs. (6c) and (6d) prevent any vehicle trip v that doesn't pass stop point n ($A_{n,v} = -1$) from retrieving sensor data from n. The constraints in Eq. (6e) guarantee that vehicle trip v always picks up sensor data before dropping off them. Eq. (6f) makes sure that if RP n sends data to vehicle trip v, v only drops off the data at a single UP. Eq. (6g) caps the UP installation cost at Θ, which is an input.

Our problem is NP-hard, which can be shown through a polynomial-time reduction from the uncapacitated facility location problem (UFLP) [37] to it. The UFLP problem deploys *facilities* to serve *clients*, with the objective of minimizing the sum of the facility cost and distance between each client and the closest facility. We can map facilities to our UPs and clients to our RPs, although our problem is more comprehensive, e.g., our penalty function is more complex than fixed distances. Details of the proof are omitted due to the space limitations.



Start from Empty Up Set
1
Potential UP Set
3
Save the Best UP on Obj. Func. Values
Upload Point Placement Algorithm
Upload Path Planning Algorithm
Return Best UP Set if Exceeding Cost Limit
Best Path Plan for Potential UP Set
2

Fig. 2: Our approach with two collaborating algorithms.

IV. Solution Approach and Algorithms

We propose to iteratively solve the problems using two alternative algorithms:

- *Upload point placement algorithm*, which produces the optimal upload point placement $\mathbf{Y}$ based on multiple invocations of the next algorithm.

Algorithm 1 Upload Point Selection Algorithm

```
Input:
— UPC set $\mathbf{N}$, schedule $\mathbf{A}$, rate $\mathbf{R}$, cost $\mathbf{C}$, delay $\mathbf{D}$
— RP set $\{n \mid n \in \mathbf{N}; R_n > 0\}$
Output:
— UP placement $\mathbf{Y}$ and upload path plan $\mathbf{X}$
1: $\mathbf{Y} \leftarrow \{0\}_N$
2: $\mathbf{X} \leftarrow \{0\}_{N \times (V+1) \times N}$
3: while $\sum_{i \in N} C_i y_i < \Theta$ do
4:   for all $n \in \{n \mid y_n = 0, C_n \ne \infty\}$ do
5:     $\mathbf{Y}'(n) \leftarrow \mathbf{Y}$; $y'_n = 1$
6:     $\mathbf{X}'(n) \leftarrow \mathbf{Assign}(\mathbf{A}, \mathbf{Y}'(n))$
7:     calculate the cost effectiveness for $n$
         $E(n) = \frac{\hat{P}(\mathbf{A},\mathbf{X}) - \hat{P}(\mathbf{A},\mathbf{X}'(n))}{C_n}$.
8:   $i \leftarrow arg\ max_n\ E(n)$
9:   $\mathbf{Y} \leftarrow \mathbf{Y}'(i)$
10:  $\mathbf{X} \leftarrow \mathbf{X}'(i)$
11: return $\mathbf{Y}$ and $\mathbf{X}$
```

- *Upload path planning algorithm*, which is invoked by the above algorithm to compute the optimal upload path plan $\mathbf{X}$ for a given UP set $\mathbf{Y}$.

Fig. 2 illustrates the interactions between these two algorithms. The algorithm systematically *tries* different potential UP sets, which do not exceed the cost limitation. It only sees the high-level picture (set $\mathbf{Y}$), and relies on the upload path planning algorithm to compute the low-level details (plan $\mathbf{X}$). In particular, for each iteration, the upload point placement algorithm generates multiple potential UP sets and invokes the upload path planning algorithm multiple times (step ①). The upload path planning algorithm computes the best plan for each potential set (step ②). Towards the end of each iteration, the upload point placement algorithm selects the best UP set based on the objective function values (step ③). If the cost limit is exceeded (Eq. (6g)), the algorithms stop; otherwise the upload point placement algorithm moves to the next iteration.

A. Upload Point Placement Algorithms

We propose four upload point placement algorithms below. Intuitively, selecting UPCs with more passing trips from RPs will increase the chance of data uploads which in turn can decrease data loss and data transfer time. Based on this observation, we propose *coverage maximization* (COV) algorithm, in which we greedily select UPCs based on their coverage of RPs. UP m is said to cover RP n if there are at least one trip v passing by n and arrives at m later, i.e., arrival time $A_{n,v} < A_{m,v}$ and $A_{n,v} \ne -1$. According to transit schedule $\mathbf{A}$, we get RP cover set $\mathbf{Cov}_m$ for each UPC m. We then greedily choose the UPCs in the decreasing order of their $\frac{|Cov_m|}{C_m}$ until the cost exceeds the cost limit Θ.

Volume-maximization (VOL) algorithm adopts the same heuristic method but uses the sum of the data rates of all those RPs instead of using only the count. The method then chooses UPCs in the decreasing order of volume to installation cost ratio until the cost hits the limit.

We also propose a *genetic algorithm* (GA) based solution where we create the initial populations using the solutions of COV and VOL. For each individual (a subset of UPC), we get its corresponding trip assignment using our **Assign** Algorithm, and calculate the penalty value according to Eqs. (4) and (5). We use the negative of penalty value as the fitness score to rank all populations. Then, we use three basic rules to create the next generation: (i) selection rules select a subset of individuals with highest fitness value as parents that contribute to the population at the next generation, (ii) crossover rules combine two parents to form children for the next generation, and (iii) mutation rules apply random changes to individual parents to form children. We also set the constraint of GA as the total cost of individuals shouldn't exceed the cost limit. In our work, we set the population size as 50 and the maximum number of iterations as 100.

The *upload point selection* (UPS) algorithm, as outlined in Algorithm 1, greedily finds UPs according to their reduction in penalty per unit of cost. More specifically, in each iteration, the algorithm computes the predicted upload path plan $\mathbf{X}'(n)$ after adding each candidate n into the current UP set by using the subroutine $\mathbf{Assign}(\mathbf{A}, \mathbf{Y}'(n))$ (line 6). It then calculates the cost effectiveness of this assignment (line 7) which is the ratio between the decrease of the total setting penalty after adding n and the cost of n. We add the UPC that maximizes the cost effectiveness to the UP set and continue to the next iteration if the cost limit has not been reached.

Algorithm 2 Assign($\mathbf{A}$, $\mathbf{Y}$) — DM algorithm

```
Input:
— Schedule $\mathbf{A}$, UP placement $\mathbf{Y}$, data rate $\mathbf{R}$
Output:
— Upload Path Plan $\mathbf{X}$
Step 1: Adding all available trips
1: $\mathbf{X} \leftarrow \{0\}_{N\times(V+1)\times N}$
2: for all UP $m \in \{m \mid y_m = 1\}$ do
3:   for all RP $n \in \{n \mid n \in \mathbf{N}, R_n > 0\}$ do
4:     if $n == y$ then
5:       set $x_{n,0,n} \leftarrow 1$
6:     else
7:       for all $v \in \{v \mid v \in \mathbf{V}, A_{m,v} \neq -1\}$ do
8:         if $A_{n,v} \neq -1$ and $A_{n,v} < A_{m,v}$ then
9:           set $x_{n,v,y} \leftarrow 1$
Step 2: Trimming
10: for all RP $n \in \{n \mid n \in \mathbf{N}, R_n > 0\}$ do
11:   if $x_{n,0,n} == 1$ then
12:     set $x_{n,v,m} \leftarrow 0 \;\forall\; v \neq 0$
13:   if $\sum_{m\in N} x_{n,v,m} > 1$ then
14:     for all $m \in \{m \mid m \in \mathbf{N}, x_{n,v,m} = 1\}$ do
15:       if $m = \underset{m\in N}{arg\ min} A_{m,v}$ then
16:         $x_{n,v,m} \leftarrow 1$
17:       else
18:         $x_{n,v,m} \leftarrow 0$
Step 3: Removing trips
19: for RP $n \in \{n \mid n \in \mathbf{N}, R_n > 0\}$ do
20:   get all trips passing $n$: $\mathbf{V_n} = \{V_{n,v,m} \mid \sum_m x_{n,v,m} = 1\}$
21:   sort $\mathbf{V_n} = [V_n[1], V_n[2], ...]$ by their arrival time $A_{n,v}$
22:   while $updated$ do
23:     for each trip $V_n[i] : V_{n,v,m}$ do
24:       get travel time $t_n[i] = A_{m,v} - A_{n,v}$
25:       get arrival time $a_n[i] = A_{n,v}$.
26:     for each trip in $V_n[i]$ do
27:       if $t_n[i] > 2 \times (a_n[i+1] - a_n[i]) + t_n[i+1]$ then
28:         remove trip $V_n[i]$: $x_{n,v,m} \leftarrow 0$
29:         $updated \leftarrow True$
30: return $\mathbf{X}$
```

B. Upload Path Planning Algorithms

We propose two upload path planning algorithms: (i) *First Contact* (FC): every RP sends buffered data through all passing-by vehicles that then drop the data to the first UPs they encounter, and (ii) *Delay minimization* (DM) algorithm, which performs local search for optimal upload path plans. The algorithm works as follows (shown in Algorithm 2). For each RP, the algorithm finds the the optimal subset of trips that should carry data and upload them to the nearest UP they encounter. One can argue that an RP can send data through *all* passing trips and transfer a little chunk of data at each encounter. But it turns out that choosing all trips may not be the best, rather skipping some trips can generate better results (produce lower total transfer time/delay). Particularly, the trips that take long travel time to reach their nearest UPs can be skipped. The following lemma (proof is omitted due to the space limit) establishes the condition.

Lemma 1 (Removing Trips)**.** *Let a stop point n see two successive trips v_1 and v_2 with the corresponding wait times as w_1 and w_2 and travel times to their respective nearest UPs as t_1 and t_2. If $t_1 > 2 \times w_2 + t_2$, then trip v_1 can be skipped which will decrease the overall data transfer time.*

Proof. The proof follows from the fact that we want to minimize the weighted data transfer delay. If the RP chooses to send data through both trips, the first trip transfers $w_1.R_n$ volume of data with transfer delay $w_1 + t_1$ and the second trip carries $w_2.R_n$ amount of data with delay $w_2 + t_2$. So, the total weighted delay is $w_1.R_n.(w_1 + t_1) + w_2.R_n.(w_2 + t_2)$. If the first trip is skipped then the sum becomes $(w_1 + w_2).R_n.(w_1 + w_2 + t_2)$, which will be smaller than the former one if $t_1 > 2.w_2 + t_2$ (the condition to remove v_1). $\square$

Algorithm 2 shows the subroutine $\mathbf{Assign}(\mathbf{A}, \mathbf{Y})$, which is our DM algorithm. This algorithm includes three phases: (i) adding all available assignments: find all trips going from RPs to UPs in the current UP set; (ii) assignment trimming: remove all useless trip assignments which map to the trips passing stops contain both RP and UP, and the trips of multiple uploading choices for data from one specific RP; and (iii) reducing lateness: remove the trip assignments that have long travel time to reduce total data transfer time per Lemma 1.

TABLE I: Considered Algorithms

	Upload Point Placement			
Upload Path Planning	**COV**	**VOL**	**GA**	**UPS**
First Contact (FC)	COV_F	VOL_F	GA_F	UPS_F
Delay Minimization (DM)	COV_D	VOL_D	GA_D	UPS_D

V. Evaluations

We perform simulations to evaluate our proposed algorithms. Our simulation setup consists of four components: (i) data preprocessor, (ii) upload point placement algorithms, (iii) upload path planning algorithms, and (iv) ONE simulator. The data preprocessor (in Python) converts open transportation datasets into proper formats. We have implemented four

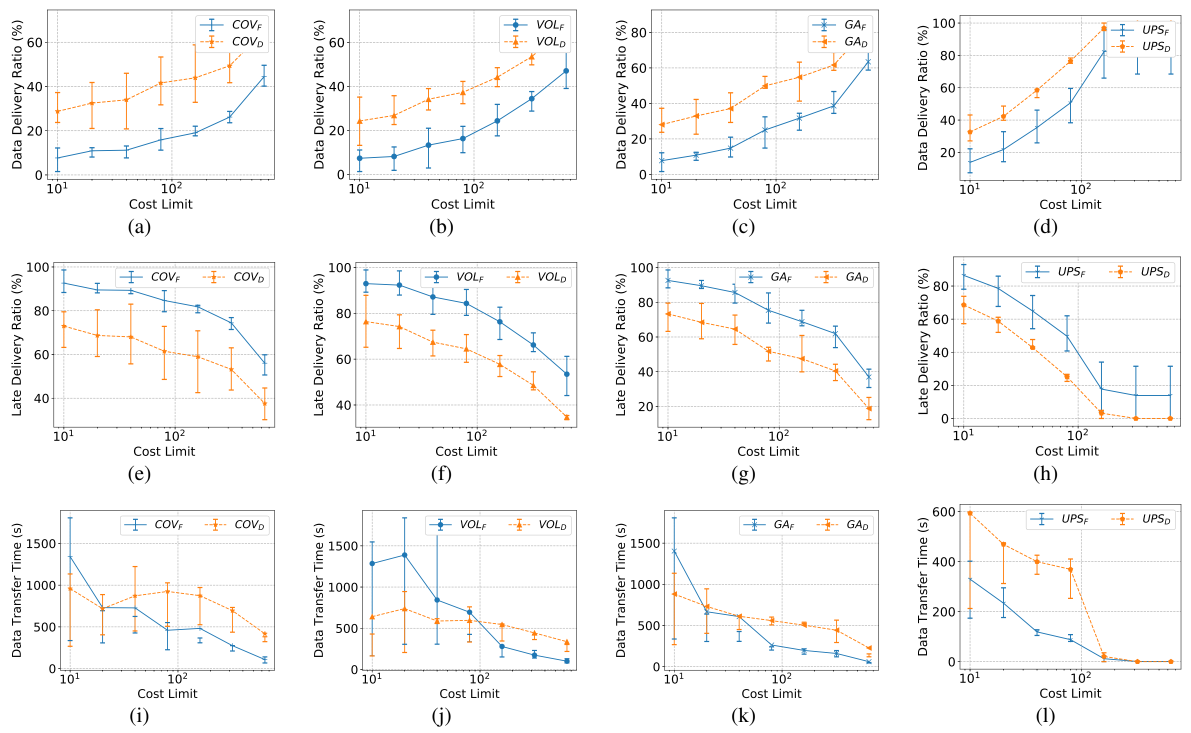


Fig. 3: Comparisons between the FC and DM algorithms under different cost limitations: (a)–(d) data delivery ratio, (e)–(h) late delivery ratio, and (i)–(l) data delivery ratio; (a), (e), (i) COV, (b), (f), (j) VOL, (c), (g), (k) GA, and (d), (h), (l) UPS.

upload point placement algorithm in Python: COV, VOL, GA, and UPS algorithms. We have also implemented two upload path planning algorithms in Python: FC and DM algorithms. All considered algorithms are summarized in Table I. Once the algorithms produce the upload point placement and upload path planning solutions, we put them into a detailed simulator: Opportunistic Network Environment (ONE) simulator [29]. We modify the ONE simulator to route the sensor data following the solutions from our algorithms and keep track of statistics. We consider the following performance metrics:

- *Penalty value:* The objective function value of the upload point placement algorithm.
- *Data delivery ratio:* The ratio between the sensor data volumes delivered at UPs and sent by RPs.
- *Late delivery ratio:* The fraction of sensor data that exceeds their delay tolerances.
- *Data transfer time:* The time difference between sensor data arrive at an RP and at a UP.
- *Total cost:* The total UP installation cost.
- *Number of UPs:* The number of placed UPs.

A. Scenarios

We employ the public transit dataset made public by the Orange County government [38]. The dataset contains bus stop locations, trip schedules, and routes. We focus on the seven bus routes around the UCI campus. In particular, our data preprocessor extracts the schedules and bus stop locations for our simulations. The resulting schedule spans over a weekday from 6:09 a.m. to 9:09 a.m., which consists of 99 vehicle trips and 551 bus stops in total. The average vehicle trip duration is 64 minutes, while the minimum (maximum) duration is 18 (109) minutes. On average, each vehicle trip traverses through 20.38 stops, and each stop has 35.01 vehicle trips passing by.

We take all bus stops as our UPCs. The dataset, however, does not contain RPs, nor their data rate, installation cost, and delay tolerance. For each simulation run, we randomly select RPs from all UPs. We then overlap the dataset with OpenStreetMap to systematically determine the parameters associated with each RP. In particular, for a given RP, we set its data rate to be proportional to the surrounding density of public facilities, with a mean of 200 kB/s. We set the installation cost to be proportional to the distance to the closest public facility, with a mean of 5. We set the delay tolerance to be proportional to the distance to the closest critical infrastructure, such as police stations and hospitals, with a mean of 30 mins. We vary the cost limitation between 10 and 640. We consider a small scenario with 20 RPs and a large one with 40 RPs. Simulations with the same inputs and parameters are repeated with all considered algorithms. We run each simulation 5 times, and report the average results. In addition, we plot the 1st/3rd quartiles as errorbars whenever possible.

B. Comparison Results

Our DM algorithm leads to better performance than the FC algorithm with the same upload point placement algorithms. We compare the two upload path planning algorithms with various upload point placement algorithms under cost limitation between 10 and 640. Sample results from 20 RPs are reported; while results with more RPs are similar. We plot the results in Fig. 3. In this figure, each column represents an

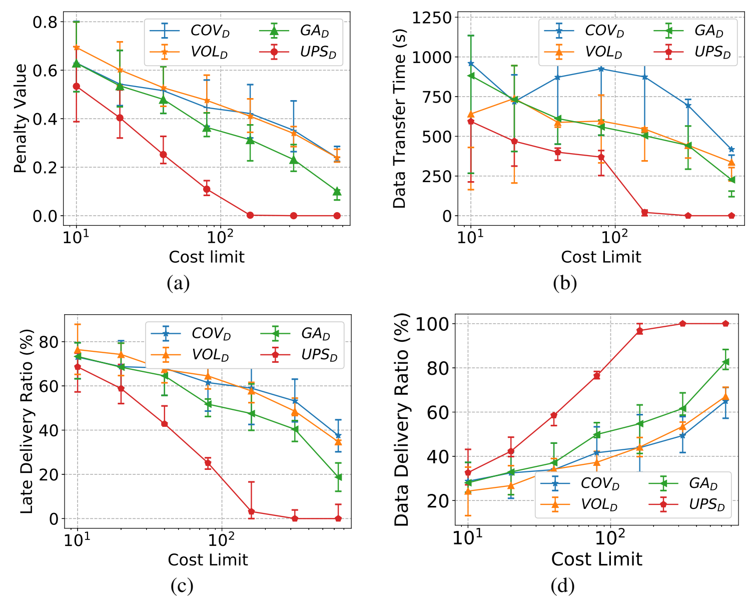


Fig. 4: Performance of the four upload point placement algorithms under different cost limitations: (a) penalty value, (b) data transfer time, (c) late delivery ratio, and (d) data delivery ratio.

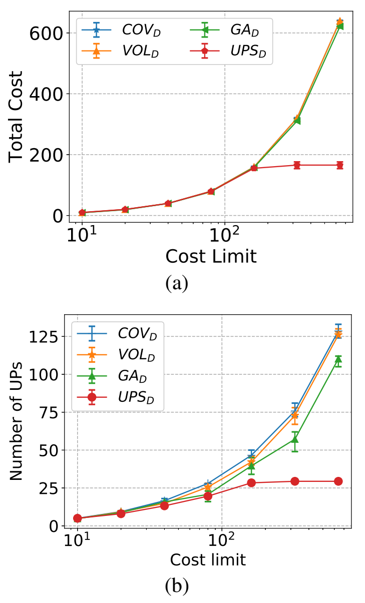


Fig. 5: Installation cost of the four upload point placement algorithms under different cost limitations: (a) total cost and (b) number of UPs.

upload point placement algorithm; while each row represents a key performance metrics. We notice that the penalty value is an output of the upload point placement algorithms, which is common with either upload path planning algorithm. Hence we do not report the penalty value (objective function value) in the figure. We make a few observations on this figure. First, Figs. 3(a)–3(d) give the data delivery ratio, which show that our DM algorithm always delivers more data: more than 20% increases are observed. Next, we check if the delivered data are late by looking into the late delivery ratio in Figs. 3(e)–3(h). It can be seen that our DM algorithm constantly results in lower late delivery ratio: 20+% average reduction is possible.

Last, the data transfer time of *delivered data* is given in Figs. 3(i)–3(l). These figures depict that the FC algorithm may lead to shorter data transfer time than the DM algorithm. This is because the FC algorithm makes greedy decisions without proper planning, which may occasionally lead to shorter data transfer time. Nonetheless, such difference doesn't change the fact that our DM algorithm delivers: (i) more data and (ii) less late data than the FC algorithm, as shown above. Thus, we no longer consider the FC algorithm in the rest of this paper.

Our UPS algorithm outperforms other upload point placement algorithms under different total costs. We plot sample results from the four upload point placement algorithms with 20 RPs in Fig. 4. In Fig. 4(a), we observe our proposed UPS algorithm significantly outperforms other algorithms in terms of the objective function value: as high as 30% gap, compared to the GA algorithm is observed. Moreover, as the cost limit increases, UPS algorithm's penalty value descends at a much higher rate than other algorithms, including the GA algorithm. We then check other performance results from the simulators: data transfer time in Fig. 4(b), late delivery ratio in Fig. 4(c), and data delivery ratio in Fig. 4(d). In all these figures, our UPS algorithms outperforms other algorithms, and the performance gap becomes nontrivial even with a moderate cost limitation. For example, with a cost limit of 160, compared to other algorithms, our UPS algorithm achieves sub-21 sec data transfer time (30+ times improvement), sub 3.2% late delivery ratio (about 14 times improvement), and above 96% data delivery ratio (about 50% improvement). Among the three other algorithms, GA performs better than COV and VOL, but still worse than our algorithm. One way to boost the performance of GA is to increase the number of iterations. This however is not feasible, because of its long running time: as high as 8 hrs for solving a problem with GA.

Our UPS algorithm results in cost-effective upload point placement. We observe above that our UPS algorithm achieves better performance with a rather small increase of cost limitation. We next dig a bit deeper and plot the total cost of the four algorithms from 20 RPs in Fig. 5(a). This figure shows that our UPS algorithm only consumes a total cost of about 180, even when the cost limitation is beyond that. Fig. 5(b) also demonstrates that the UP placement decisions are almost frozen beyond the cost limitation of 160. These two figures demonstrate that our UPS algorithm makes cost-effective placement decisions; on top of its superior performance. In contrast, three other algorithms continue to use up all the cost limitation yet deliver inferior performance.

Our UPS algorithm delivers prominent performance gains over other algorithms in larger/heavier scenarios. We next report performance *gain* of our UPS algorithm over other algorithms, which is defined as the performance improvement normalized to UPS' value. Notice that, our UPS algorithm may achieve zero late delivery ratio. In that case, we put 0.01% in the denominator to be conservative. Fig. 6 shows the

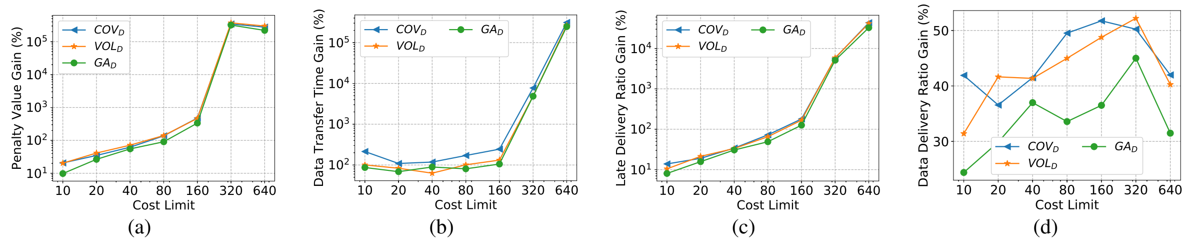


Fig. 6: The performance gains of UPS comparing with the other algorithms on: (a) penalty value, (b) data transfer time, (c) late delivery ratio, and (d) data delivery ratio under different cost limitations in the scenario with 40 RPs.

sample performance gains from the larger scenario. This figure confirms our above observations on the smaller scenario are also applicable to larger scenario. Even through with more RPs (i.e., heavier traffic), the performance gains remain significant across the considered cost limitations. For example, at the cost limitation of 160, our UPS algorithm gets at least 478% gain in penalty value, 244% gain in data transfer time, 180% gain in late delivery ratio, and 45% gain in data delivery ratio.

Hence, we recommend the combination of UPS and DM algorithms for solving the upload point placement and upload path planning problems.

VI. Conclusion

In this paper, we studied the use of scheduled transportation fleets to enable cost-effective, reliable and timely data collection in urban IoT settings with limited backhaul connectivity. We illustrated the value of a hierarchical approach that include: (a) the creation of locally connected "IoT zones" with planned collection points (RPs), (b) careful positioning of limited upload points from which data is uploaded to backend data processing centers, and (c) intelligent planning of data movement from RPs to UPs using already scheduled transportation fleets. In the future, we plan to expand the range of urban scenarios to include extreme conditions (such as fires, earthquakes), which can damage the sensing, communication and transport fabric in smart communities. The use of alternate data transfer methods, e.g. drones is a topic of further study as well. The flexibility of our hierarchical approach will become more crucial as the data collection needs evolve.

References

[1] "https://www.vox.com/2016/8/4/12342806/barcelona-superblocks."

[2] "https://smartgrowth.org/6-urban-green-space-projects-that-are-revitalizing-u-s-cities/."

[3] Report Linker. (2017) Internet of Things (IoT) networks: Technologies and global markets to 2022. [Online]. Available: https://www.reportlinker.com/p05273319

[4] T. Black *et al.*, "Using Autonomous Mobile Agents for Efficient Data Collection in Sensor Networks," in *WAC '06.*, 2006.

[5] M. Zhao *et al.*, "Mobile Data Gathering with Load Balanced Clustering and Dual Data Uploading in Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 4, pp. 770–785, Apr. 2015.

[6] W. Zhao *et al.*, "A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks," in *MobiHoc '04*. ACM, 2004.

[7] D. Kim *et al.*, "Minimum latency multiple data mule trajectory planning in wireless sensor networks," *IEEE TMC*, vol. 13, no. 4, 2014.

[8] N. D. Lane *et al.*, "Piggyback CrowdSensing (PCS): Energy Efficient Crowdsourcing of Mobile Sensor Data by Exploiting Smartphone App Opportunities," in *SenSys '13*. ACM, 2013.

[9] H. Xiong *et al.*, "CrowdTasker: Maximizing coverage quality in Piggyback Crowdsensing under budget constraint," in *PerCom '15*, 2015.

[10] J. Ma *et al.*, "Pso-based proactive routing in delay tolerant network," in *CCT '14*. IET, 2014, pp. 1–4.

[11] C. Raffelsberger *et al.*, "A hybrid manet-dtn routing scheme for emergency response scenarios," in *PerCom Workshops '13*. IEEE, 2013.

[12] A. Petz *et al.*, "A Framework for Evaluating DTN Mobility Models," in *ICST '09*, 2009, pp. 94:1–94:8.

[13] C. Konstantopoulos *et al.*, "A Rendezvous-Based Approach Enabling Energy-Efficient Sensory Data Collection with Mobile Sinks," *IEEE TPDS*, vol. 23, no. 5, pp. 809–817, 2012.

[14] A. W. Khan *et al.*, "A Comprehensive Study of Data Collection Schemes Using Mobile Sinks in Wireless Sensor Networks," *Sensors*, pp. 2510–2548, 2014.

[15] P. Kuila *et al.*, *Clustering and routing algorithms for wireless sensor networks: Energy efficiency approaches*. Chapman and Hall/CRC, 2017.

[16] M. Sede *et al.*, "Routing in large-scale buses ad hoc networks," in *WCNC '08*, pp. 2711–2716.

[17] L. Li *et al.*, "R2R: Data forwarding in large-scale bus-based delay tolerant sensor networks," pp. 27 – 31, 12 2010.

[18] F. Xhafa *et al.*, "Locals search algorithms for efficient router nodes placement in Wireless Mesh Networks," in *NBiS '09*, 2009.

[19] ——, "Solving mesh router nodes placement problem in Wireless Mesh Networks by Tabu Search algorithm," *JCSS*, vol. 81, no. 8, 2015.

[20] D. Buezas, "Constraint-based modeling of Minimum Set Covering : Application to Species Differentation Constraint-based modeling of Minimum Set Covering : Application to Species Differentation," Ph.D. dissertation, 2010.

[21] S. Chu *et al.*, "Deployment of a connected reinforced backbone network with a limited number of backbone nodes," *IEEE Transactions on Mobile Computing*, vol. 12, no. 6, pp. 1188–1200, 2013.

[22] T. Oda *et al.*, "Analysis of mesh router placement in wireless mesh networks using Friedman test," in *AINA '14*, 2014, pp. 289–296.

[23] C. Song *et al.*, "Deployment mechanism design for cost-effective data uploading in delay-tolerant crowdsensing," in *ISPA/IUCC '17*, Dec 2017.

[24] B. Behsaz *et al.*, "New Approximation Algorithms for the Unsplittable Capacitated Facility Location Problem," in *Algorithm Theory - Scandinavian Symposium and Workshops*, 2016, pp. 237–248.

[25] R. Ravi *et al.*, "Approximation Algorithms for Problems Combining Facility Location and Network Design," *Operations Research*, vol. 54, no. 1, pp. 73–81, 2006.

[26] K. Xiong, "Solving the Performance Puzzle of DSRC Multi-Channel Operations," in *ICC '14*, 2014.

[27] L. Quintero-Cano *et al.*, "Bus networks as graphs: new connectivity indicators with operational characteristics," *Canadian Journal of Civil Engineering*, 2014.

[28] P. Cao *et al.*, "Solving configuration optimization problem with multiple hard constraints: An enhanced multi-objective simulated annealing approach," Tech. Rep. 860, 2017.

[29] A. Keränen, "Opportunistic network environment simulator," 2008.

[30] A. Ker *et al.*, "Simulating Mobility and DTNs with the ONE," *JOURNAL OF COMMUNICATIONS*, vol. 5, no. 2, pp. 92–105, 2010.

[31] L. Kang, "A Public Transport Bus as a Flexible Mobile Smart Environment Sensing Platform for IoT," in *IE '16*, 2016, pp. 1–8.

[32] N. Indra Er *et al.*, "On the Performance of VDTN Routing Protocols with V2X Communications for Data Delivery in Smart Cities," in *IWSSS'17*, 2017, pp. 1–2.

[33] U. Acer *et al.*, "Timely data delivery in a realistic bus network," in *IEEE INFOCOM '11*, April 2011, pp. 446–450.

[34] F. Zhang *et al.*, "On geocasting over urban bus-based networks by mining trajectories," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 6, pp. 1734–1747, June 2016.

[35] ——, “Cbs: Community-based bus system as routing backbone for vehicular ad hoc networks,” *TMC*, vol. 16, no. 8, pp. 2132–2146, 2017.

[36] L. Kang *et al.*, “A public transport bus as a flexible mobile smart environment sensing platform for iot,” in *IE ’16*, Sep. 2016, pp. 1–8.

[37] M. Mahdian *et al.*, “Universal facility location,” in *Algorithms - ESA 2003*. Springer Berlin Heidelberg, 2003, pp. 409–421.

[38] J. Matute. (2018) Publicly-accessible public transportation data. [Online]. Available: https://www.transitwiki.org/TransitWiki/index.php/Publicly-accessible˙public˙transportation˙datam