8-2020

Resource Optimization in Support of IoT Applications

Ihab Ahmed Mohammed
Western Michigan University, eihabmurjan@yahoo.com

RESOURCE OPTIMIZATION IN SUPPORT OF IOT APPLICATIONS

by

Ihab Ahmed Mohammed

A dissertation submitted to the Graduate College
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
Computer Science
Western Michigan University
August 2020

Doctoral Committee:

Ala Al-Fuqaha, Ph.D., Chair
Ajay Gupta, Ph.D.
Alvis Fong, Ph.D.
Mohammad Salahuddin, Ph.D.

RESOURCE OPTIMIZATION IN SUPPORT OF IOT APPLICATIONS

Ihab Ahmed Mohammed, Ph.D.

Western Michigan University, 2020

With the rise of the Internet of Things (IoT) and smart communities, managing computation and communication resources required by billions of smart devices becomes a concern. To tackle this problem, we develop algorithms for resource management to ensure better Quality of Service (QoS), safety, and performance. We focus our efforts on three problems.

In the first problem, we studied the strict QoS requirements of applications and differentiated service requirements in different situations of vehicular networks. We propose a generic prioritization and resource management algorithm that can be used to prioritize the processing of received packets in vehicular networks. We formulate the generic severity-based prioritized packet processing problem as Penalized Multiple Knapsack Problem (PMKP) and prove that it is an NP-Hard problem. We thus develop a real-time heuristic that utilizes a relaxed version of the formulation. The relaxed formulation executes in polynomial time and guarantees a minimum delay per severity-level while respecting the processing rate constraint. To measure the performance of the proposed heuristic, real traffic data is used in a small-scale experiment. The proposed heuristic is tested against the PMKP solution and results show a small degradation of up to 4% in profit for the heuristic compared to the PMKP solution. Also, the proposed heuristic is tested against a non-prioritized processing algorithm that works using first come first served policy. Results show that the proposed heuristic gains 9% to 67% more profit than the non-prioritized processing algorithm in moderate and high congestion scenarios.

In the second problem, we explored the utilization of existing vehicles on roads as "message ferries" for the transport data for smart community applications to avoid the cost of installing new communication infrastructure. We propose an opportunistic data ferry selection algorithm that strives to select vehicles that can minimize the overall delay for data delivery from a source to a given destination. Our proposed opportunistic algorithm utilizes an ensemble of online hiring algorithms, which are run together in passive mode, to select the online hiring algorithm that has performed the best in recent history. The proposed ensemble-based algorithm is evaluated empirically using real-world traces from taxies plying routes in Shanghai, China, and its performance is compared against a baseline of four state-of-the-art online hiring algorithms. A number of experiments are conducted and our results indicate that the proposed algorithm can reduce the overall delay compared to the baseline by an impressive 13% to 258%.

In the third problem, we solve the problem of optimizing accuracy in stateful federated learning with a budgeted number of candidate clients by selecting the best candidate clients in terms of test accuracy to participate in the training process. We formulate the problem of maximizing the probability of selecting the best candidate clients based on test accuracy as a secretary problem then analytically analyze the performance and provide proofs. Next, we propose an online stateful FL heuristic to find the best candidate clients. Additionally, we propose an IoT client alarm application that utilizes the proposed heuristic in training a stateful FL global model based on IoT device classification to alert clients about unauthorized IoT devices in their environment. To test the efficiency of the proposed online heuristic, we conduct several experiments using a real dataset and compare the results of the proposed online heuristic against state-of-the-art algorithms. Our results indicate that the proposed heuristic performance is comparable to the performance of the best offline algorithm and outperforms the online random algorithm with up to 27% gain in accuracy.

ACKNOWLEDGEMENTS

I dedicate this humble work to my mother and my aunt Ezdihar, who dedicated their lives to take care of me. To my wife, who sacrificed her time and energy so that I complete this work. To my uncle Sadiq, who raised me like his son. To my uncle Safaa, who inspired me and supported me. To the beautiful minds (friends and colleagues), who supported me through this journey. To Prof. Ala (my supervisor), who challenged my mind, orchestrated my efforts, sharpened my skills, and showed me the path to success in academia, industry, and life. I also would like to express sincere thanks to:

* Committee members Dr. Ajay Gupta, Dr. Alis Fong, and Dr. Mohammad Salahuddin.

* Faculty and Staff of Computer Science Department, WMU.

* The Graduate College, WMU.

* Faculty and Staff of Al-Nahrain University, Baghdad - Iraq.

* Staff of the Higher Committee of Education Development in Iraq (HCED).

Ihab Ahmed Mohammed

TABLE OF CONTENTS

Table of Contents—Continued

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

IoT	**I**nternet of **T**hings
IoMT	**I**nternet of **M**edical **T**hings
IoAT	**I**nternet of **A**nimal **T**hings
IoWasteT	**I**nternet of **W**aste **T**hings
IoBT	**I**nternet of **B**attlefields **T**hings
IoUGT	**I**nternet of **U**nderground **T**hings
IoUWT	**I**nternet of **U**nderwater **T**hings
IoNT	**I**nternet of **N**ano **T**hings
IoMobT	**I**nternet of **M**obile **T**hings
VANETs	**V**ehicular **A**d-hoc**N**etworks
QoS	**Q**uality of **S**service
MKP	**M**ultiple **K**napsack **P**roblem
PMKP	**P**enalized **M**ultiple **K**napsack **P**roblem
RPKP	**R**elaxed **P**enalized **K**napsack **P**roblem
DSRC	**D**edicated **S**hort-**R**ange **C**ommunications
WAVE	**W**ireless **A**ccess in **V**ehicular **C**ommunications
SCH	**S**ervice **C**hannel
CCH	**C**ontrol **C**hannel
EDCA	**E**nhanced **D**istributed **C**hannel **A**ccess
MAC	**M**edium **A**ccess **C**ontrol
ACs	**A**ccess **C**ategories
BSMs	**B**asic **S**afety **M**essages

ECDSA	**E**lliptic **C**urve **D**igital **S**ignature **A**lgorithm
OBU	**O**n-**B**oard **U**nit
D2D	**D**evice **2** **D**evice
FCFS	**F**irst **C**ome **F**irst **S**erve
ICT	**I**nformation **C**ommunications **T**echnologies
WSNs	**W**ireless **S**ensor **N**etworks
CPS	**C**yber-**P**hysical **S**ystems
MF	**M**essage **F**errying
SCMC	**S**mart **C**ommunity **M**anagement **C**enter
TMC	**T**raffic **M**anagement **C**enter
LCB	**L**ocal **C**ommunity **B**roker
SVaaS	**S**mart **V**ehicle as a **S**ervice
DTN	**D**elay **T**olerant **N**etwork
ML	**M**achine **L**earning
DL	**D**eep **L**earning
FD	**F**ederated **L**earning
AI	**A**rtificial **I**ntelligence
EHR	**E**lectronic **H**ealth **R**ecord
VEC	**V**ehicular **E**dge **C**omputing
DDoS	**D**istributed **D**enial **o**f **S**ervice
CSV	**C**omma **S**eparated **V**alues
JSON	**J**avaScript **O**bject **N**otation
DDN	**D**eep **N**eural **N**etwork

CHAPTER 1

INTRODUCTION

1.1. Introduction

The Internet started as a mean to connect people and provide electronic services such as email, web, e-commerce, and social networks. Nevertheless, as more heterogeneous devices (i.e. laptops, tables, phones, sensors, etc) are connected to the internet, the concept of the Internet of Things (IoT) is realized. IoT refers to devices, sensors, or things in general that are intelligent, have a unique address, and autonomous [1]. Moreover, IoT have many applications and benefits. In fact, IoT is used in developing countries to obtain Sustainable Development Goals (SDGs) of the United Nations [2]. The authors in [3] classified application in IoT as follows:

- **Internet of Medical Things (IoMT)** with things are medical wearable devices to control diseases such as Parkinson, diabetes or collect information such as heart rate;
- **Internet of Animal Things (IoAT)** with things such as smart cattle collars, RFID ear tags, and sound analyzers;
- **Internet of Waste Things (IoWasteT)** with things like smart garbage bins and RFID tags;
- **Internet of Battlefield Things (IoBT)** with things like ammunition, weapons, vehicles, and human-wearable sensors;
- **Internet of Underground Things (IoUGT)** with things such as buried soil sensors and seismometers;

- **Internet of Underwater Things (IoUWT)** with things such as underwater sensors and asmrt buoys;
- **Internet of Nano Things (IoNT)** with things such as nano-sensors and actuators;
- **Internet of Mobile Things (IoMobT)** with things like mobile personal devices.

It is estimated that each person will use 6 to 7 devices by the year 2020 and 500 billion devices will be connected to the Internet by 2030 [1]. Consequently, connecting all those devices while providing high quality of service along with privacy and security is a challenge. Realization of IoT vision faces many challenges mostly related to availability, reliability, mobility, performance, management, scalability, interoperability, and security and privacy [4]. In this work, we address some of these challenges by designing real-time and online algorithms to optimize the use of resources in IoT applications.

1.2. Statement of the Problem

Managing communication and computation resources utilized by billions of smart devices in IoT applications is a challenge. The lack of efficient algorithms that control the usage of resources results in economic problems, safety threatening situations, and security and privacy concerns.

1.3. Purpose of the Research

In this work, we investigate the use of efficient real-time and online algorithms in managing communication and computation resources in IoT applications.

1.4. Significance of the Study

On one study, we show that the utilization of the proposed real-time algorithm to opportunistically prioritize the processing of data flows based on a generic severity notion in vehicles helps drivers avoid collisions, which saves lives. On another study, we reduce the cost of setting up smart cities significantly by using vehicles as message ferries instead of setting up a costly infrastructure. Additionally, we propose opportunistic data ferry selection algorithm that strives to select vehicles that can minimize the overall delay for data delivery from a source to a given destination. Finally, we optimize the accuracy of stateful federated learning used in IoT applications.

1.5. Contributions

This dissertation is based on the following publications:

- **Paper I:** A. Al-Fuqaha, I. Mohammed, S. J. Hussini, and S. Sorour, "*Severity-Based Prioritized Processing of Packets in VANETs*," in IEEE Transactions on Mobile Computing, vol. 19, no. 2, pp. 484-496, 1 Feb. 2020.

- **Paper II:** I. Mohammed, S. Tabatabai, A. Al-fuqaha, and J. Qadir, "*Opportunistic Data Ferrying in Areas with Limited Information and Communications Infrastructure*," 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall), Honolulu, HI, USA, 2019, pp. 1-6.

- **Paper III:** I. Mohammed, S. Tabatabai, A. Al-fuqaha, F. El Bouanani, J. Qadir, and B. Qolomany, "*Budgeted Online Selection of Candidate Clients to Participate in Federated Learning*," to be submitted, IEEE Internet of Things Journal, July 2020.

The work in this dissertation also contributes to the following publications:

- S. Tabatabai, I. Mohammed, A. Al-fuqaha, and J. Qadir, "*Opportunistic Selection of Vehicular Data Brokers as Relay Nodes to the Cloud*," 2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2020, pp. 1-6.

- S. Tabatabai, I. Mohammed, A. Al-Fuqaha, and M. A. Salahuddin, "*Managing a Cluster of IoT Brokers in Support of Smart City Applications*," 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), Montreal, QC, 2017, pp. 1-6.

1.6. Structure of the Dissertation

The overall structure of the dissertation is as follows:

- Chapter I: Introduction
- Chapter II: Paper I
- Chapter III:Paper II
- Chapter IV: Paper III
- Chapter V: Conclusions and future work

CHAPTER 2

Severity-Based Prioritized Processing of Packets with Application in VANETs

2.1. Introduction

2.1.1. Motivation

Dedicated short-range communications (DSRC) and Wireless Access in Vehicular Communications (WAVE) have become the de-facto vehicular communications technologies in VANETs. IEEE 802.11p and IEEE P1609 form the bases of the WAVE protocol as shown in Fig.2.1[5]. To address the unique challenges in VANETs (i.e., dynamic topology, short communication periods and application requirements), WAVE uses multi-channel operations to increase channel utilization, and differentiated service categories to provide Quality of Service (QoS). To provide multi-channel operations, WAVE utilizes the IEEE 1609.4 standard, which defines six Service Channels (SCH) and one Control Channel (CCH) for use in VANETs [6]. These channels have different frequencies, maximum transmission power, and applications. The CCH for instance is used for transmitting safety and control data in order to insure fast and prioritized delivery of time-critical data. Furthermore, to provide differentiated services, WAVE relies on IEEE 802.11p, which uses Enhanced Distributed Channel Access (EDCA) at the Medium Access Control (MAC) layer to provide differentiated QoS. The IEEE 802.11p EDCA specifies four different Access Categories (ACs): AC_VO (Voice or AC3), AC_VI (Video or AC2), AC_BE (Best effort or AC1) and AC_BK (Background or AC0), each of which has a different priority for accessing the communication medium [6]–[8]. This categorization offers low latency communications and differentiated services for applications. An overall schematic of multi-channel access and ACs assignment is shown in Fig.2.2. We have

to emphasize that the IEEE 802.11p scheme for prioritization only applies to applications during data transmission and does not affect the processing of data after its receipt. For more details on WAVE, please refer to [5], [9], [10].

The use of multi-channel access and data prioritization via ACs, as detailed in the EDCA standard, only guarantees prioritized access to the communications medium for broadcasting and sending data. However, this may not necessarily lead to improved system-level performance, because each vehicle in the network has to process the received data based on its own context, not the transmitting vehicle's context. Assume the scenario shown in Fig.2.3, where five different vehicles are involved. If vehicles B and C collide, then both vehicles would try to inform their neighbors by sending data using the highest priority AC. This flow of data is very important for vehicle "A", but not as important for vehicles "D" and "E". In such a scenario, vehicle "A" must prioritize the processing of packets that it receives from vehicle "B" (and/or vehicle "C"). On the other hand, vehicles "D" and "E" do not necessarily benefit from processing the high priority safety application data flow(s) from vehicles "B" and/or "C" ahead of other flows with lower ACs, since the accident does not have an impact on their mobility. In such scenarios, it is hence imperative that each vehicle processes the received data flows, not just based on their ACs, but also based on the flows impacts on the vehicles' mobility.

Prioritization of received data and management of computation resources are important tasks because autonomous and semi-autonomous vehicles have to process data from a wide range of sensors (e.g., LIDAR, GPS, compass, radar, infrared cameras) at any given moment, which in turn puts more computation overhead over the on-board computational resources. Beyond processing raw sensor data, the on-board computational resources must execute complex algorithms to perform proximity understanding and vehicle control.

Another example to illustrate the importance of the proper prioritization of received data

flows is the security performance of basic safety messages (BSMs). To verify and sign BSMs, the standard recommends using the Elliptic Curve Digital Signature Algorithm (ECDSA). However, using ECDSA increases the processing overhead [11] and degrades the performance that could result in the loss of safety-critical BSMs. This can become problematic especially in environments with a higher density of vehicles. To solve this issue, the authors of [12] propose re-prioritizing signature verification of received BSMs based on location proximity of the sender's vehicle, such that nearby vehicles are assigned higher priority. This example demonstrates the importance of prioritized processing of data flows on the receiver's side in order to increase the overall performance of the system.

From the discussion of the two previous scenarios, it is evident that achieving ideal performance in VANETs depends, not only on efficient algorithms and techniques for sending data but also on a better prioritization of packets on the receiver side. This prioritization must be done according to a certain *severity metric* that corresponds to the nature of the application. The proposed algorithm prioritizes the processing of received data based on their impact on the safety of a given vehicle.

2.1.2. Contributions

In this paper, we aim to design a real-time algorithm to opportunistically prioritize the processing of data flows based on a generic *severity* notion, such that the benefit to the overall system is maximized. Severity can thus be defined according to the application, as a metric (e.g., vehicle speed, direction, position) that underlines the importance of the flow for the overall benefit of the system. The proposed algorithm models flow prioritization by classifying them according to their severity into multiple queues with different priority and capacity levels. For this proposed approach, we derive an upper bound on the delay of service for the flows classified in each queue using network calculus. We then formulate

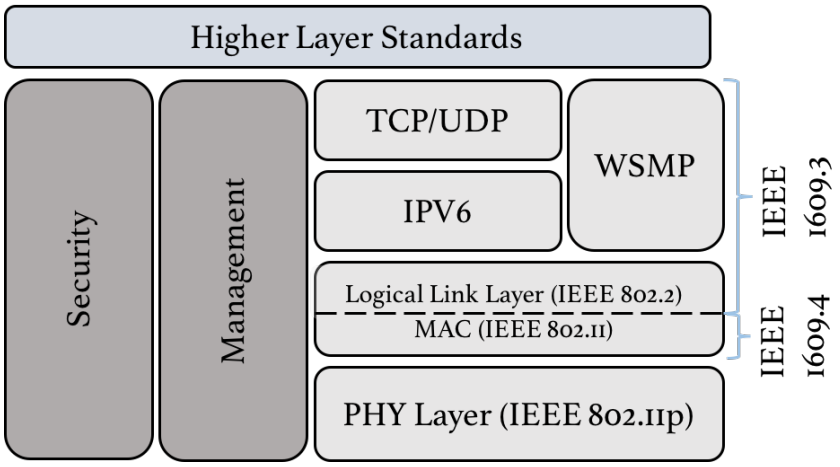


Figure 2.1: WAVE stack.

the problem of maximizing the severity of the admitted flows into the queue system as a penalized multiple knapsack problem with a service delay constraint on each of the queues. Being an NP-hard problem, we propose a real-time heuristic that divides the problem into sub-problems, each finding the optimal admission of flows into each of the queues such that the total severity of the admitted flows is maximized. Finally, both the optimal and heuristic solutions are tested using real traffic data and compared to un-prioritized first come first serve approaches.

2.1.3. Examples of Applications

The potential applications of our proposed algorithm can be numerous depending on the application's definition of *flow severity*. In the simplest form, flow severity can be defined as a QoS requirement that needs to be fulfilled. The dynamic topology of VANETs imposes strict QoS requirements on applications, especially the safety applications [13]. If severity is defined as application's QoS requirement, then the algorithm would strive to assign flows to queues such that flows with the most strict QoS requirements get higher priority. If severity is defined as the sending vehicle's proximity, then the algorithm would strive to provide

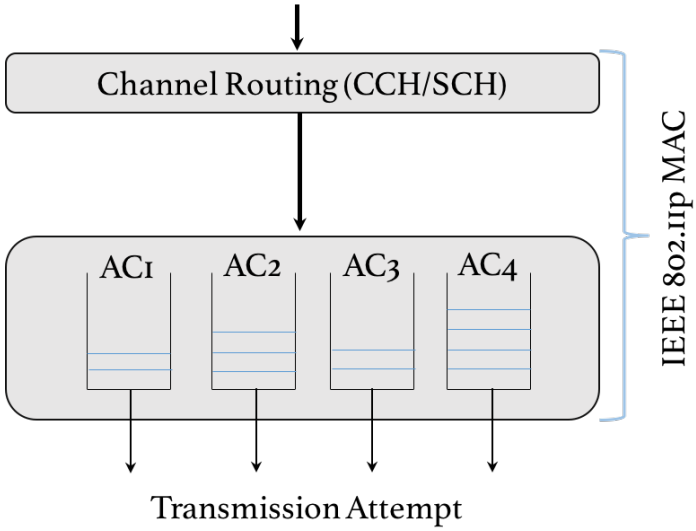


Figure 2.2: User priority service in WAVE.

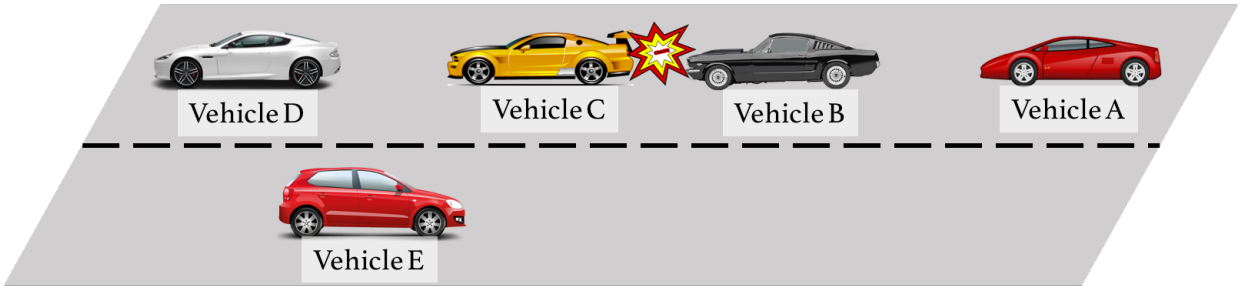


Figure 2.3: Illustrative scenario.

prioritized service to flows that belong to the closest vehicles. In all these use cases, our algorithm acts as an admission control system, whereby it assesses the processing resources and incoming flows and strives to service the most severe flows first in order to achieve the best performance for the overall system.

Another important use of our proposed severity-based prioritization algorithm is to provide adaptive security in VANETs. To provide security in VANETs, WAVE relies on cryptographic solutions [10]. In order to have better security, it is important that more robust cryptographic solutions are used. However, using more robust cryptographic solutions leads

to increased computations, increased cryptographic loss [11], and can have severe impact on the achieved QoS of applications [11], [14], [15]. To overcome this problem, adaptive security measures can be used, where a vehicle can determine the robustness of the cryptographic solution it uses for each flow based on the load and performance of the overall system. In such scenarios, our algorithm can classify four different cryptographic solutions ranked from the most (i.e., requires more computations) to the least robust, and determine which cryptographic solution would result in the best performance based on the flow, application QoS requirements, and performance of the overall system.

In conclusion, we are proposing an algorithm that can be adopted to support any application benefiting from intelligent prioritized processing of requests. In the rest of this paper, we focus on prioritized packet processing in VANETs.

2.1.4. Organization

The remainder of the paper is organized as follows: The related literature is reviewed in Section 2.2. The proposed prioritization model and system parameters are described in Section 2.3. In Section 2.4, we derive the delay upper bounds for the service of each of the queues. Section 2.5 shows the formulation of the problem in terms of the penalized knapsack problem, while Section 2.6 provides a heuristic solution. In Section 2.6.4, we provide the upper bound delay proof. Section 2.7 describes our experiments and results while conclusions are drawn in Section 2.8.

2.2. Related Work

WAVE QoS, service prioritization, and performance improvements are studied extensively in the literature. Generally, the studies and solutions offered can be divided into two broad

groups: (1) studies that offer solutions from the *sender's* point of view and (2) studies that tackle the issue from the *receiver's* point of view. In the first group, the offered solutions range from spectrum sharing in order to prioritize safety message delivery [16]–[20] to MAC layer enhancements and admission control techniques in order to improve QoS and channel access [21]–[29]. Furthermore, there have been studies [30]–[32] that use vehicle's contextual information (i.e., vehicle's position) in order to ensure effective utilization of resources and provide prioritized services. The major difference between our work and these studies is that we tackle the issue from the receiver's point of view. We consider a receiver's perspective (as opposed to the sender's perspective) to achieve prioritized processing at the receiver (as opposed to prioritized transmission from the sender). Our algorithm deals with data that is received by a vehicle with the objective of opportunistically prioritize the processing of the received data for the overall benefit of the VANET.

While there have been numerous studies to improve QoS, service prioritization, and performance of WAVE from the sender's perspective, in the second group, there have not been many studies that address these issues on the receiver's side. The few studies that have been conducted in this area focus on security performance. Verification of time-sensitive BSMs in order to decrease cryptographic loss, by prioritizing the verification of BSMs based on the physical position of the sender, has been studied in[12], [33]. The authors of [34] have proposed to reduce the verification time of messages at the receiver side in dense areas by assigning different priority levels to nearby vehicles based on their physical parameters after verifying those vehicles. Our study in this paper extends these special cases in the aforementioned sources to a more general prioritized processing of the received packets in VANETs, as a function of a generic severity metric, in order to maximize the overall profit to the system while respecting its QoS constraints. The generic severity metric can be defined based on the application, such as physical proximity of vehicles as in [34] or more complex

settings as in [32]. Therefore, our proposed solutions can apply to a variety of prioritized packet processing applications by properly defining its corresponding severity metric.

2.3. System Model

2.3.1. Prioritization Model

In this paper, we propose a real-time algorithm that allows vehicles in VANETs to intelligently prioritize the processing of the received packets based on their severity. Upon receipt of data flows, each flow is assigned a severity-level and passed on to the next phase for flow policing. This policing prevents flooding attacks as it regulates each flow via a token bucket filter to a predefined processing rate. The last phase is assigning the flows to one of N queues based on both the flow's severity and the capacity of the queue. These queues have different priority levels and capacities, such that "Queue N" has the highest priority and smallest capacity, while "Queue 1" has the lowest priority and the largest capacity. The size of each queue is determined by the computational load and delay requirements of it's assigned flows, and it dictates the maximum delay that will be incurred by the flows assigned to that queue. While assigning flows to queues, the algorithm strives to assign the highest severity flows to the highest priority queue to ensure their fast processing. The other flows are passed to their corresponding queues according to their severity levels. Once the capacity of a queue is reached, the algorithm proceeds to place the next batch of unassigned flows of the same severity to the subsequent lower priority queue. This process continues until all flows are assigned to queues or queues are all filled. If some flows are left out after all the queues were filled, the algorithm discards these remaining flows as these cannot be serviced in a timely manner. Although it's possible that some flows are discarded, our algorithm makes sure to serve the highest severity flows first while guaranteeing performance bounds for the

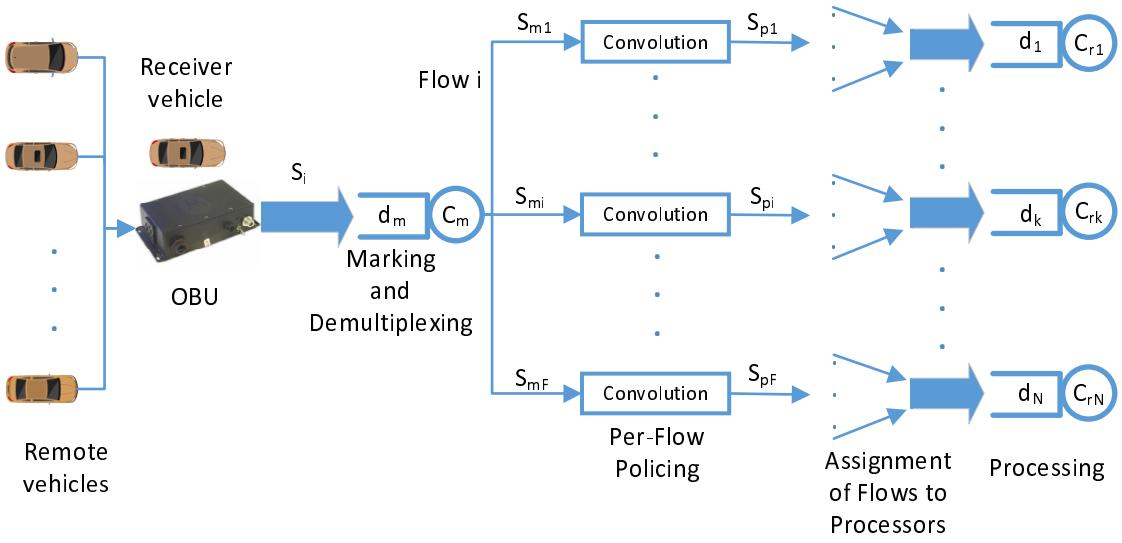


Figure 2.4: System model.

accepted flows. The process of assigning flows to queues happens in snapshots. During every snapshot, which is done every Δt, the algorithm takes the available flows and assigns them to queues. The frequency of generating flow snapshots and the addition and removal of the flows to the snapshots are not handled in our algorithm. Rather, it assumes the availability of the snapshots and focuses on prioritizing flows available during each snapshot.

2.3.2. System Parameters

The system model and parameters are shown in Fig.2.4. Assuming a VANET with V vehicles, where one vehicle $V_r \in V$ receives packets of length L (measured in bytes), sent from a set of sending vehicles $V_s = V \setminus V_r$ within its communications range. V_r receives all these packets via its On Board Unit (OBU), which has a channel reception rate R_b (measured in Mbps) and maximum burst length B (measured in packets). Let R be the rate in packets per second of the OBU (i.e., $R = \frac{R_b}{L}$). The received packets are then passed to the marking queue of the marking processor, which is assumed to be the On-Board

Unit (OBU) processor. The marking processor assigns to each incoming packet a priority (severity) level based on the receiving vehicle's perspective. The marking service requires simple computations (e.g., marking the packets based on the type of application, relative position, speed and direction of the sending vehicle); thus, induces a very small and almost negligible processing delay. The marking processor uses the packets' source MAC address in order to demultiplex the incoming packets into a set of flows $\mathcal{F}$ (i.e., a flow is defined between a source-destination pair of vehicles). The marking process applied to the scenario shown in Fig.2.3, would be as follows: When vehicles B and C collide, they transmit data about the collision to their surrounding vehicles. Since vehicle A is affected by this collision, when it receives data about the collision, based on its context severity, its marking processor will mark the received data about the collision with the highest severity category and processes it accordingly. But since the collision doesn't affect vehicles D and E, when they receive this data, their marking processors would mark it with a lower severity level (severity level 2,3, or 4 based on the defined context severity). Let S_i be the severity level of the packets in flow i, $i \in \{1, \ldots, |\mathcal{F}|\}$. We assume M severity levels. Packets are assigned to flows in order to enforce per-flow policing, which is an important measure in order to protect the system from flow-based unfair exploitation (e.g., flooding attacks). The policing phase limits the per-flow rate and burstiness based on the severity level. Flow i, $i \in \{1, \ldots, |\mathcal{F}|\}$, with severity level S_i has a maximum rate and burstiness of $(\rho^i_{max}, \sigma^i_{max})$. Regardless of the traffic from neighboring vehicles, all flows are thus forced to respect the per-severity limits such that flows violating the limits are dropped. Finally, the last phase is the processing phase where flows are assigned to N queues. "Queue N" has the highest priority and the least capacity, which is appropriate for serving high severity flows. On the other end, "Queue 1" has the lowest priority and highest capacity. Although, "Queue N" may best serve flows of high severity, but this queue may admit flows of low severity depending on the available

flows in order to maximize the utilization of system resources. Processor k, $k \in \{1, \ldots, N\}$, has a fixed processing rate of C_k and queue k, $k \in \{1, \ldots, N\}$, has a guaranteed maximum delay of d_{qk}. Additionally, the total delay induced by processing all accepted flows admitted to queue k must be less than or at most equal to the guaranteed maximum delay d_{qk} as explained later in equation (9). The objective is to serve the maximum number of requests while providing the highest quality of service and maintaining guaranteed delay bounds in all queues.

2.4. Delay Upper-Bounds

Since the OBUs are the sources of data and as long as the maximum transfer rate is known, the output of the i-th transmitting OBU can be upper bounded, using network calculus tools, by an affine arrival curve (A_i) that is defined as [35]:

$$A_i \sim (B_i, R_i) \tag{2.1}$$

Consequently, the i-th sending OBU has burstiness B_i and rate R_i. The sum of burstiness of all OBU flows equals the received burstiness at the destination OBU as shown in the following equation:

$$B = \sum_{i=1}^{F} B_i$$

Also, the sum of the rates of all OBU flows equals the received rate at the destination OBU as shown in the following equation:

$$R = \sum_{i=1}^{F} R_i$$

Thus, the input to each receiving OBU D_i is upper bounded by an affine arrival curve:

$$D_i \sim (B, R) \tag{2.2}$$

Since the input to the marking process is bounded by (2.2) and the marking service rate is C_m, then the delay for the marking phase d_m is:

$$d_m = \frac{B}{C_m - R}$$

The output of the marking phase D_{m_i} consists of $|\mathcal{F}|$ flows that is upper bounded by:

$$D_{m_i} \sim \left(B_i + R_i \cdot \frac{B}{C_m - R}, R_i\right)$$

which can be reduced to the following form:

$$D_{m_i} \sim \left(\frac{B_i\left(C_m - R\right) + R_i B}{C_m - R}, R_i\right) \tag{2.3}$$

The policing process of the ith flow D_{p_i} is expressed by the following bound:

$$D_{p_i} \sim \left(\left(\frac{B_i\left(C_m - R\right) + R_i B}{C_m - R}, R_i\right) * \left(\sum_{j=1}^{M} \sigma_{max}^{j} f_i^j, \sum_{j=1}^{M} \rho_{max}^{j} f_i^j\right)\right) \tag{2.4}$$

where f_i^j is a flag that is set to 1 to indicate that the ith flow belongs to severity level j; zero otherwise. Since there exist M severity levels and each flow is assigned to only one of them, we have the following constraint on these flags:

$$\sum_{j=1}^{M} f_i^j = 1 \quad \forall i$$

The $*$ operator in (2.4) represents the min-plus convolution operator, which is an infimum operation over the addition of the bursitness and the service. Consequently, Equation (2.4) can be rewritten as:

$$D_{p_i} \sim \alpha(t) = \min \left\{ \frac{B_i \left(C_m - R\right) + R_i B}{C_m - R} + R_i t,\ \sum_{j=1}^{M} \sigma_{max}^{j} f_i^j + \sum_{j=1}^{M} \rho_{max}^{j} f_i^j t \right\}$$

which is equivalent to:

$$D_{p_i} \sim \left(\min \left(\frac{B_i \left(C_m - R\right) + R_i B}{C_m - R}, \sum_{j=1}^{M} \sigma_{max}^{j} f_i^j \right),\ \min \left(R_i, \sum_{j=1}^{M} \rho_{max}^{j} f_i^j \right) \right)$$

Since we are looking for the upper bound, we can assume the maximum traffic-per-flow. In other words, only the maximum burstiness and rate per severity are selected from equation (2.5). Thus, the output of the policing process is upper-bounded by:

$$D_{p_i} \sim \left(\sum_{j=1}^{M} \sigma_{max}^{j} f_i^j, \sum_{j=1}^{M} \rho_{max}^{j} f_i^j \right) \tag{2.5}$$

Each flow i is assigned to one of the N prioritization queues of the system. The flag g_i^k is used to indicated these assignments, such that g_i^k is set to 1 to indicate that the ith flow is assigned to Queue k; otherwise, it's set to zero. So, for any flow i we have:

$$\sum_{k=1}^{N} g_i^k \leqslant 1 \qquad \forall i$$

The delay during the processing phase d_{r_k} is per-queue, so the upper-bound on the delay for Queue k is computed using all flows that are processed by Queue k as:

$$d_{r_k} = \frac{\sum_{i=1}^{|\mathcal{F}|} \sum_{j=1}^{M} \sigma_{max}^{j} f_i^j g_i^k}{C_k - \sum_{i=1}^{|\mathcal{F}|} \sum_{j=1}^{M} \rho_{max}^{j} f_i^j g_i^k} \tag{2.6}$$

The total delay per-queue d_{t_k} is the sum of the marking delay and the processing delay as shown below:

$$d_{t_k} = \frac{B}{C_m - R} + \frac{\sum_{i=1}^{|\mathcal{F}|} \sum_{j=1}^{M} \sigma_{max}^{j} f_i^j g_i^k}{C_k - \sum_{i=1}^{|\mathcal{F}|} \sum_{j=1}^{M} \rho_{max}^{j} f_i^j g_i^k} \tag{2.7}$$

Note that "Queue N" has the highest priority and can handle high severity flows faster than the other queues.

2.5. Problem Formulation

As was mentioned earlier, our objective is to serve the maximum number of flows (i.e., vehicles) prioritized based on their severity while at the same time striving to provide the highest quality of service. However, these two goals might not always be in harmony. If we try to provide the highest quality of service (e.g., using the most robust cryptographic solution), it may result in some flows not being serviced within the required delay constraint. So, we have to find a balance between the quality of service that can be provided and the number of flows that can be serviced with priorities based on their severities. Thus, we formulate this problem as a Penalized Multiple Knapsack Problem (PMKP) with the goal of maximizing the profit (i.e., the total severity of the admitted flows). The next sub-sections introduce the formulation of the problem of maximizing the profit of the system in terms of the total sum of severities of admitted flows while fulfilling their corresponding delay constraints. In sub-Section C, we show that the problem is NP-Hard.

2.5.1. Derivation of the Delay Constraints

Each queue has a delay d_{q_k}, and accepted flows must satisfy the condition that the total processing delay of accepted flows in Queue k must be less than the delay of that queue.

This can be written mathematically as:

$$d_{t_k}=\frac{B}{C_m-R}+\frac{\sum_{i=1}^{|\mathcal{F}|}\sum_{j=1}^{M}\sigma_{max}^{j}f_i^j g_i^k}{C_k-\sum_{i=1}^{|\mathcal{F}|}\sum_{j=1}^{M}\rho_{max}^{j}f_i^j g_i^k}\leqslant d_{q_k}$$

which states that the total delay-per-queue d_{t_k} must not exceed the kth queue delay d_{q_k}. The above inequality can be reduced to the following form:

$$\sum_{i=1}^{|\mathcal{F}|}\left(\sum_{j=1}^{M}\sigma_{max}^{j}f_i^j+d_{q_k}\sum_{j=1}^{M}\rho_{max}^{j}f_i^j-\frac{B}{C_m-R}\sum_{j=1}^{M}\rho_{max}^{j}f_i^j\right)\cdot g_i^k\leqslant C_k\left(d_{q_k}-\frac{B}{C_m-R}\right) \quad (2.8)$$

2.5.2. PMKP Formulation

Before presenting the formulation, we define ψ as the penalty term that represents the highest severity among the dropped flows and order the set $\mathcal{F}$ based on severity in non-increasing order.

The problem can thus be expressed as a PMKP as follows:

$$\max\sum_{k=1}^{N}\sum_{i=1}^{|\mathcal{F}|}S_i\cdot g_i^k\cdot k-\psi \quad (2.9)$$

s.t.

$$S_i\left(1-g_i^k\right)\leqslant\psi\quad\forall\ i,k \quad (2.10)$$

$$\sum_{i=1}^{|\mathcal{F}|}\left(\sum_{j=1}^{M}\sigma_{max}^{j}f_i^j+d_{q_k}\sum_{j=1}^{M}\rho_{max}^{j}f_i^j-\frac{B}{C_m-R}\sum_{j=1}^{M}\rho_{max}^{j}f_i^j\right)\cdot g_i^k\leqslant C_k\left(d_{q_k}-\frac{B}{C_m-R}\right)\quad\forall\ k \quad (2.11)$$

$$\sum_{k=1}^{N} g_i^k \leqslant 1 \quad \forall\, i \tag{2.12}$$

$$g_i^k \in \{0,1\} \quad \forall\, i,k \tag{2.13}$$

Constraint (2.10) indicates that the penalty term is the highest severity among the dropped flows. The idea is to deduct a penalty from the overall profit for every dropped flow. The second constraint (2.11) ensures that the total delay of each queue does not exceed the guaranteed queue delay. The third constraint in (2.12) specifies that every flow i cannot be assigned to more than one queue. However, it is possible that flow i is not assigned to any of the queues; meaning it is dropped. The last constraint in (2.13) indicates that this optimization problem is an integer linear programming problem, where the assignment of flows to queues can be either zero or one (i.e., no fractional assignment of flows to queues).

2.5.3. Proof of NP-Hardness

The problem discussed in this paper is a $PMKP$ and is $NP-Hard$ in the weak sense. To prove it, we only need to show that this problem is actually a Multiple Knapsack Problem (MKP). In fact, MKP is a special case of the $PMKP$ when the penalty term ψ is known.

2.6. Heuristic Solution

To devise a real-time heuristic, we divide the overall problem into N sub-problems, where each problem $k \in N$ finds the optimal flows that fit in Queue k capacity such that the total profit (i.e., admitted severity) of the system is maximized. To ensure the maximum system profit and the best quality of service for the received flows, the sub-problems are solved starting from Queue N down to Queue 1. The solution of the sub-problems is inspired by

the work of Ceselli and Righini [36].

Before presenting the algorithm, we define $\mathcal{F}_k$ as the set of input flows to Queue k, i.e., flows available to be admitted in Queue k. This set $\mathcal{F}_k$ of input flows to Queue k includes all the unassigned flows after solving the flow assignment problems to the higher priority queues (i.e., $\mathcal{F}_k$ includes all flows from the original set $\mathcal{F}$ except those selected by the higher priority Queues $\{k+1, ..., N\}$). For example, the input set of flows $\mathcal{F}_{N-1}$ excludes flows selected by Queue N from input set $\mathcal{F}_N$. Furthermore, we define $\mathcal{X}_k$, (with $|\mathcal{X}_k| = |\mathcal{F}_k|$) as the vector of flow selection flags from $\mathcal{F}_k$, where a value of 1 in the i-th flag $\mathcal{X}_k[i]$ indicates that the i-th flow is selected for admission in Queue k, while 0 means that the i-th flow is not selected by Queue k. Consequently, the set of input flows to Queue k can be mathematically expressed as:

$$\mathcal{F}_k = \mathcal{F}_{k+1} \setminus \bigcup_{\forall\ i|X_{k+1}[i]=1} f_i \qquad \forall\ k = N-1, \ldots, 1$$

where $\mathcal{F}_N$ is the set of input flows to Queue N, which is initialized to all $\mathcal{F}$.

The proposed heuristic algorithm proceeds as follows:
For $k = N$ down to 1, do the following steps:

- Order the set of input flows $\mathcal{F}_k$ for Queue k based on their severity.
- Find the initial flow selection vector $\mathcal{X}_k$ for admission to Queue k using the lower bound solution (See Sec. 2.6.1). Let Z_k be the sum of the severity of admitted flows in Queue k given the lower bound solution.
- Iteratively find better flow admission solutions for Queue k using the upper bound solution (See Sec. 2.6.2). If any flow admission solution $\widetilde{\mathcal{X}}_k$ with the sum-severity $\widetilde{Z}_k > Z_k$, let $Z_k = \widetilde{Z}_k$ and $\mathcal{X}_k = \widetilde{\mathcal{X}}_k$.

- $\mathcal{X}_k$ contains the optimal flow admission solution for Queue k having the maximum severity level Z_k, while respecting the flow constraints.

In the next subsections, we will explain how the lower and upper bound solutions are obtained. The pseudo-code in Algorithm 1 details the steps of the entire process.

2.6.1. Lower Bound Solution

The input flows are admitted to Queue k in descending order of flow severity. Such flows are added until the flow with index l, for which its addition will exceeds the capacity C_k of Queue k (as illustrated in Fig. 2.5). Thus, the range of flows $\{1, ..., l-1\}$ can be admitted and processed by Queue k without violating the capacity constraint, and without penalty, since no flow is dropped from this range of flows. This implies that the elements $\mathcal{X}_k[j]$ will have one values for $j \in \{1, ..., l-1\}$, and zero otherwise. We will refer to flow f_l^k as the initial leading flow, which represents the flow with the highest severity among the dropped flows in the range $\{l, ..., |\mathcal{F}_k|\}$. Consequently, the lower bound solution Z_k is initialized by adding the severity of those flows on the left side of flow l as shown in the following equation:

$$Z_k = \sum_{i=1}^{l-1} S_i$$

This initial solution is a feasible solution. Next, we define $\mathcal{L}_k$ as the set of possible leading flows for Queue k. The set $\mathcal{L}_k$ is initialized with all the flows to the left side of the leading flow f_l^k (i.e., f_i^k, $\forall i \in \{1, \cdots l-1\}$) as illustrated in Fig. 2.5. Each leading flow is used to find an upper bound solution to the sub-problem.

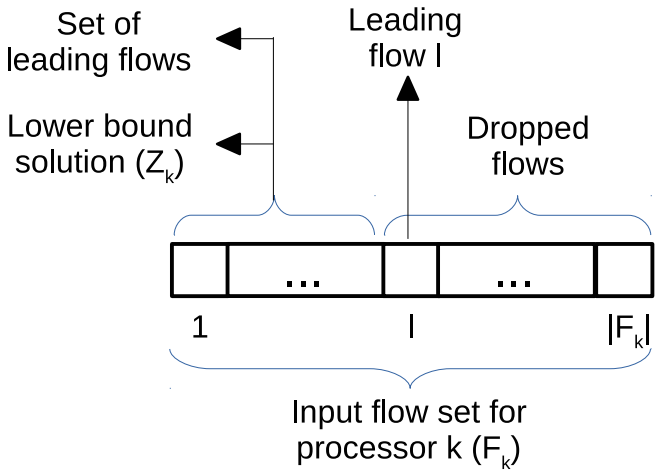


Figure 2.5: Input flow set divisions.

2.6.2. Upper Bound Solution

In this part, we solve $|\mathcal{L}_k|$ Relaxed Penalized Knapsack Problems ($RPKP$) and compare their solutions with the lower bound solution to find the solution with the maximum profit. The following formulation represents the problem of finding the upper bound solution for Queue k's flow selection problem:

$$\max \sum_{i=1}^{|\mathcal{F}_k|} S_i \cdot g_i^k \cdot k - \psi_k \tag{2.14}$$

s.t.

$$S_i \left(1 - g_i^k\right) \leqslant \psi_k \quad \forall\ i \tag{2.15}$$

$$\sum_{i=1}^{|\mathcal{F}_k|} \left(\sum_{j=1}^{M} \sigma_{max}^j f_i^j + d_{q_k} \sum_{j=1}^{M} \rho_{max}^j f_i^j - \frac{B}{C_m - R} \sum_{j=1}^{M} \rho_{max}^j f_i^j \right) \cdot g_i^k \leqslant C_k \left(d_{q_k} - \frac{B}{C_m - R} \right) \tag{2.16}$$

$$0 \leqslant g_i^k \leqslant 1 \quad \forall i \tag{2.17}$$

$$\widetilde{\mathcal{X}_k}[i] = 1 \quad \forall\ i = 1 \ldots l-1 \tag{2.18}$$

$$\widetilde{\mathcal{X}_k}[l] = 0 \tag{2.19}$$

The upper bound solutions are computed using the following $|\mathcal{L}_k|$ iterative steps for every leading flow $i \in \mathcal{L}_k$ in descending order of severity (as detailed in the pseudo-code of Algorithm 1):
While $l \geq 1$ do:

- Set the penalty term $\psi_k = S_l$; i.e. set it to the severity of the leading flow.
- Mark the leading flow l as dropped and all flows to its left as selected. This is basically equivalent to eleminating the current leading flow from $\mathcal{F}_k$ in the current iteration of the upper bound solution. Marking is represented by constraints (2.18) and (2.19).
- Solve the problem as an $RPKP$ with the penalty $\psi_k = S_l$.
- Fix the fractional results of $RPKP$ to 1s and 0s using the proposed fixing heuristic (See Sec. 2.6.3). Compute the corresponding upper bound solution $\widetilde{\mathcal{X}_k}$ and its sum-severity $\widetilde{Z}_k$ as the severity of flows with $x_i = 1,\ \forall\ i$.
- If the current upper bound solution has a greater sum-severity than Z_k, update the solution $\mathcal{X}_k = \widetilde{\mathcal{X}_k}$ and its sum-severity $Z_k = \widetilde{Z}_k$.
- Select the new current leading flow l from the set $\mathcal{L}_k$ to be the one just to the left of the previous leading flow. In other words, set $l \leftarrow l - 1$.

The core philosophy of this solution is to eliminate one of the selected flows or the initial leading flow from the lower bound solution at a time, starting from the lowest severity among them and on. This opens room for selecting flows on the right of the current leading flow of each iteration, as long as this will maximize the profit of Queue k (i.e. the sum-severity of its final admitted flows). To do this, the problem is solved as an $RPKP$. Now since the penalty term ψ_k is known to be the severity of the leading flow, then the problem is reduced to an RKP. However, solving the problem using the relaxed version might result in a fractional solution. Thus, we use the proposed heuristic in section 2.6 for fixing the relaxed solution to get a feasible integer solution.

2.6.3. Fixing Upper Bound Solution

The solution produced by solving the relaxed problems (i.e, $RPKP$) in each of the $|\mathcal{L}_k|$ iterations of the upper bound solution results in values between 0 and 1. Such values do not constitute a feasible solution as they do not indicate which flows are assigned to Queue k. To resolve this problem, we propose an algorithm for rounding the fractional values to integer values.

The algorithm iteratively searches for a flow with strictly fractional value (i.e. between 0 and 1 neither including 0 nor 1) starting from the left side of the set $\widetilde{\mathcal{X}_k}$ and fixes it to 1 (i.e., select that flow). However, the $RPKP$ solution is optimal, and rounding a value to 1 means accepting more portion of that flow. Consequently, this may result in more delays, which could violate the delay constraint. Thus, the next step is to search for a set of flows with strictly fractional values starting from the right side of the set $\widetilde{\mathcal{X}_k}$ and fix them to 0 (i.e drop these flows) in order to regain the balance and satisfy the delay constraint. The pseudo-code in Algorithm 1 details these steps, where D_t is the total delay for selected flows ($\mathcal{X}_k[i] = 1, \forall i$) and D_k^{max} is the maximum delay for Queue k.

Algorithm 1 Severity-Based Prioritized Processing Heuristic

Set $\mathcal{F}_k = \mathcal{F}$ $k = N$ downto 1
Sort $\mathcal{F}_k$ in descending order based on severity
Find the minimum l such that flows $1, \ldots, l-1$ fit in C_k
Set $\mathcal{X}_k[i] = 1 \; \forall \; i\{1, \ldots, l-1\}$
Set $Z_k = \sum_{i=1}^{l-1} S_i \; l \geqslant 0 \; l \geq 1$
Set $\widetilde{\mathcal{X}_k}[l] = 0 \; l \geq 2$
Set $\widetilde{\mathcal{X}_k}[i] = 1 \;\; \forall i \in \{1, \ldots, l-1\}$
Set $\widetilde{\mathcal{X}_k}[j] = RPKP$ solution, $j \in \{l+1, \cdots, |\mathcal{X}_k|\}$.
Set $a = l + 1$
Set $b = |\mathcal{X}_k|$ a < b $\widetilde{\mathcal{X}_k}[a] > 0$ AND $\widetilde{\mathcal{X}_k}[a] < 1$
Round $\widetilde{\mathcal{X}_k}[a]$ to 1
do$\widetilde{\mathcal{X}_k}[b] > 0$ AND $\widetilde{\mathcal{X}_k}[b] < 1$
  Round $\widetilde{\mathcal{X}_k}[b]$ to 0
  Compute D_t
  $b = b - 1$
while $D_t > D_k^{max}$ AND a < b $D_t > D_k^{max}$
Undo any changes on $\widetilde{\mathcal{X}_k}$ for this iteration
Round $\widetilde{\mathcal{X}_k}[a]$ to 0
$a = a + 1$
$\widetilde{Z}_k = \sum_{i|\widetilde{\mathcal{X}_k}[i]=1} S_i \; \widetilde{Z}_k > Z_k$
Set $Z_k = \widetilde{Z}_k$
Set $\mathcal{X}_k = \widetilde{\mathcal{X}_k}$
$l = l - 1$
Set $\mathcal{F}_{k-1} = \mathcal{F}_k \setminus \bigcup_{\forall \; i|\mathcal{X}_k[i]=1} f_i$

2.6.4. Upper Bound Solution Proof

First, we define the term $KP(l)$ to denote the optimal value of the binary knapsack with flows indices in the range $(1, \ldots, l-1)$ as selected for Queue k. The binary knapsack problem is defined in the following formulation:

$$KP(l) = \max \sum_{i=1}^{|\mathcal{F}_k|} S_i \cdot g_i^k \cdot k$$

s.t.

$$\sum_{i=1}^{|\mathcal{F}_k|}\left(\sum_{j=1}^{M}\sigma_{max}^{j}f_i^j+d_{q_k}\sum_{j=1}^{M}\rho_{max}^{j}f_i^j-\frac{B}{C_m-R}\sum_{j=1}^{M}\rho_{max}^{j}f_i^j\right)\cdot g_i^k\leqslant C_k\left(d_{q_k}-\frac{B}{C_m-R}\right)$$

$$g_i^k\in\{0,1\}\quad\forall\ i$$

$$\widetilde{\mathcal{X}_k}[i]=1\quad\forall\ i=1\ldots l-1$$

Taking flow l as the leading flow, we define the term KP_l which is the optimal value of the binary KP with flows indices in the range $(1,\ldots,l-1)$ as selected for Queue k and flow l as the flow with the highest severity among all dropped flows. The definition is expressed below:

$$KP_l=\max\sum_{i=1}^{|\mathcal{F}_k|}S_i\cdot g_i^k\cdot k$$

s.t.

$$\sum_{i=1}^{|\mathcal{F}_k|}\left(\sum_{j=1}^{M}\sigma_{max}^{j}f_i^j+d_{q_k}\sum_{j=1}^{M}\rho_{max}^{j}f_i^j-\frac{B}{C_m-R}\sum_{j=1}^{M}\rho_{max}^{j}f_i^j\right)\cdot g_i^k\leqslant C_k\left(d_{q_k}-\frac{B}{C_m-R}\right)$$

$$g_i^k\in\{0,1\}\quad\forall\ i$$

$$\widetilde{\mathcal{X}_k}[i] = 1 \quad \forall\ i = 1 \ldots l-1$$

$$\widetilde{\mathcal{X}_k}[l] = 0$$

Note that KP_l is a special case of $KP(l)$ when the l-th flow is eliminated from the optimization problem. Next, we define the term PKP_l as the optimal value of the binary PKP with a penalty of subtracting the severity of the l-th flow as the leading flow as shown below:

$$PKP_l = KP_l - S_l \tag{2.20}$$

Finally, we use RKP_l to denote the relaxed knapsack problem after eliminating the l-th flow (i.e. the leading flow), which is defined as follows:

$$RKP_l = \max \sum_{i=1}^{|\mathcal{F}_k|} S_i \cdot g_i^k \cdot k$$

s.t.

$$\sum_{i=1}^{|\mathcal{F}_k|} \left(\sum_{j=1}^{M} \sigma_{max}^j f_i^j + d_{q_k} \sum_{j=1}^{M} \rho_{max}^j f_i^j - \frac{B}{C_m - R} \sum_{j=1}^{M} \rho_{max}^j f_i^j \right) \cdot g_i^k \leqslant C_k \left(d_{q_k} - \frac{B}{C_m - R} \right)$$

$$0 \leqslant g_i^k \leqslant 1 \quad \forall\ i$$

$$\widetilde{\mathcal{X}_k}[i] = 1 \quad \forall\ i = 1 \ldots l-1$$

$$\widetilde{\mathcal{X}_k}[l] = 0$$

Ordering the set of input flows $\mathcal{F}_k$ based on severity in non-increasing order results in the following inequalities:

$$S_l \geqslant S_{l+1} \tag{2.21}$$

$$KP(l) \geqslant KP(l+1) \tag{2.22}$$

$$KP(l) \geqslant KP_l \tag{2.23}$$

$$RKP_l \geqslant KP_l \tag{2.24}$$

The following represents the computation of the upper bound solutions for all possible leading flows:

$$u_i = RKP_i - S_i \quad \forall\ i = 1 \ldots l \tag{2.25}$$

Where the flows in the range $\{1, \ldots, l\}$ represent the list of leading flows as explained previously.

Proposition 1.

Let $\mathcal{S}$ be the set of $RKP_i - S_i$ for leading flow i $\forall i = 1 \ldots l$. Also, let $\mathcal{P}$ be the set of solutions of PKP_i for leading flow i $\forall i = 1 \ldots l$. Then, u is an upper bound for the optimal value of

the original PKP if

$$u \geqslant \max(\mathcal{S}) \geqslant \max(\mathcal{P}) \geqslant PKP \tag{2.26}$$

Proof.

From definitions (2.20), (2.24), and (2.25), and since

$u \geqslant u_i = RKP_i - S_i \geqslant KP_i - S_i = PKP_i \geqslant PKP \quad \forall i = 1 \dots l$

then

$u \geqslant \max(\{u_1 \dots u_l\}) \geqslant \max(\{PKP_1 \dots PKP_l\}) \geqslant PKP$ □

The core of the proposed heuristic (cf. Algorithm 1) is based on this proposition.

2.7. Experimental Results

In this section, we apply our proposed severity-based service prioritization scheme to VANETs. We first describe the VANET traffic data used in our experiments and experiment settings and the environment. Furthermore, we will also discuss the achieved results.

2.7.1. VANET Traffic Characterization

As explained earlier, our proposed heuristic works on any metric defined to serve as the *severity*. For our experiments, we define the severity to represent the type of application generating the VANET data. Data on safety applications (e.g., BSMs) are marked to have higher severity than data of infotainment applications (e.g., video streaming, congestion warning). Furthermore, we use the Bologna Ring-way dataset [37] to incorporate real vehicle mobility in our VANET simulations.

Table 2.1: D2D Experimental results

	Lecture Video		Rogue One Trailer	
	MPEG-2	MPEG-4	MPEG-2	MPEG-4
Average	38.9	23.1	78	84.1
Peak	92	29	97	97.6
Burst	2.3	1.2	1.2	1.1

Average/Peak values are in packets per second

For all safety applications, the data rate is set to 10 packets per second as recommended by the standard for safety periodic messages (i.e., BSMs) [38]. Moreover, the burstiness and rate values for infotainment applications were acquired experimentally. In these experiments, a computer is set up as a VLC [39] server that streams videos over a Motorola Vehicle Mounted Modem or OBU acting as a device-to-device (D2D) communications medium. Another computer with VLC software acts as a client. While the VLC client is receiving the video stream, the network traffic is analyzed using WireShark [40] to determine the rate and burstiness of the traffic. It should be noted that two different types of videos, each with two different video encodings, were used for our experiments. The maximum rate of infotainment applications was set to the maximum rate of the WAVE protocol suite, while the burstiness value from our previous experiment was used as the average burstiness of infotainment applications. The results of the experiments are shown in Table 2.1.

To measure the performance of the proposed heuristic, we conducted two experiments. In the first experiment, we compared the results from the PMKP formulation in (2.9) against our proposed heuristic. In the second experiment, we compared our results against a baseline non-prioritized processor that operates using first come first serve (FCFS).

To conduct the experiments, we simulate the system model shown in Fig. 2.4 using a marking processor with a capacity (rate) of 9000 packets per second. The capacity of the marking processor is set to a large value that induces minimum delay during the marking

Table 2.2: Policing queue settings

Severity Level	Maximum Burst	Maximum Rate
1	2	50
2	2	50
3	1	10
4	1	10

Table 2.3: Queue settings

# of Queue	Capacity	Guaranteed Delay
1	130	0.4
2	110	0.3
3	90	0.2
4	70	0.1

service. The policing processor configuration is shown in Table 2.2, which specifies for each severity level the maximum rate in packets per second and maximum burst size in packets.

The PMKP and our proposed heuristic use four queues with the configurations shown in Table 2.3, where the capacity is reported in packets per second and the guaranteed delay is reported in seconds. Severity level 1 is for safety messages while severity levels 2-4 are for infotainment messages. The baseline non-prioritized processor uses one queue without guaranteed delay (i.e., non-prioritized processing).

To evaluate the performance of the proposed heuristic against the baseline non-prioritized processor, we set the capacity of the baseline queue to equal the combined capacity of the four queues. Such configuration is more advantageous to the baseline no-prioritized processing algorithm as it will have one large queue with a rate that equals the rate of the four queues of the proposed prioritized processing algorithm. To validate the results, we run two other tests and set the baseline queue capacity to 50% of the combined capacity of the four queues in one experiment, and 150% of the combined capacity of the four queues in the other.

Obtaining the simulation data and the results of the two experiments are discussed next.

2.7.2. Simulation Settings

In order to test the proposed heuristic, the Bologna Ring-way dataset was used to generate a basis of test data. The Bologna Ring-way dataset results were analyzed and a snapshot of the data was taken every 200 milliseconds. The snapshots provided us with an accurate number of flows for each vehicle during the simulation. It was revealed that a vehicle can have no more than 50 flows at a given time and most of the vehicles have 10 flows at some point during the simulation. Equipped with this data, we generated five data sets with 10, 20, 30, 40 and 50 flows. To simulate real-world traffic for each of these data sets, we further generated different percentages of safety-to-infotainment application data. Safety applications have the highest severity level, while infotainment and non-safety applications have three different severity levels. For each data set, five different combinations of safety-to-infotainment messages, which are $20\% - 80\%$, $40\% - 60\%$, $50\% - 50\%$, $60\% - 40\%$, and $80\% - 20\%$ were generated. The first two scenarios of 10 and 20 flows represent a light congestion case. A moderate congestion case is presented in the third scenario with 30 flows. To test high congestion traffic scenarios, the fourth and fifth scenarios with 40 and 50 flows are used. The number of flows in these scenarios represents the number of vehicles that a certain vehicle is receiving data.

2.7.3. Comparison Study: Proposed Heuristic Versus PMKP

In this experiment, we executed the PMKP optimization and run the proposed heuristic on the 25 datasets. The testing results are shown in Fig.2.6. As Fig.2.6 (a) indicates, the PMKP has a small gap of up to 4% in profit gain over the proposed heuristic. As for the number of accepted flows, the PMKP and the proposed heuristic have similar results with the PMKP having more flows in some cases as illustrated in Fig.2.6 (b). The PMKP achieves

a better delay compared to the proposed heuristic, as clearly illustrated in Fig.2.6 (c).

In this small-scale experiment that involves a high congestion scenario, the maximum number of flows is set to 50. This value can be higher in real-life traffic congestion scenarios. However, when raising the number of flows over 40, the optimal PMKP requires more processing time, unlike the proposed heuristic, which makes it useless in solving this real-time problem. This is due to the combinatorial nature of embedded integer linear programming in PMKP. Thus, this experiment demonstrates that the proposed heuristic performs almost at the same level as that of the PMKP in terms of profit and quality of service. This is the case when the number of sending vehicles is between 10 and 50. Our proposed heuristic outperforms the PMKP in terms of its execution time in all cases especially when the number of sending vehicles is over 40.

2.7.4. Comparison Study: Proposed Heuristic Versus Non-Prioritized Processing

To compare the performance of the four queues when the proposed heuristic is employed with that of the single queue non-prioritized processing approach, we conduct three experiments. In these experiments, we set the baseline non-prioritized processor queue capacity to 50%, 100%, and 150% of the combined capacity of the four queues of the proposed heuristic, respectively. Experimenting with these various capacities aims to give more confidence to our claims on the superiority of our proposed prioritized processing heuristic.

One way to compare the proposed approach with the non-prioritized approach is to sum the severity of the accepted flows for each approach and compare the two results. However, in this paper, we gave more advantage to the non-prioritized approach (i.e., baseline) over the proposed approach for the sake of fairness. Thus, in the proposed approach, the cumulative severity is the sum of the severity of accepted flows. However, for the non-prioritized

approach, the cumulative severity is the sum of the severity of all accepted flows multiplied by a factor based on the total delay as shown below:

$$profit = \begin{cases} cumulative\ severity \cdot 1 & \text{if } d_t \geqslant 0.3, \\ cumulative\ severity \cdot 2 & \text{if } 0.2 \leqslant d_t < 0.3, \\ cumulative\ severity \cdot 3 & \text{if } 0.1 \leqslant d_t < 0.2, \\ cumulative\ severity \cdot 4 & \text{if } d_t < 0.1. \end{cases}$$

where d_t is the total delay. Basically, the profit of the baseline non-prioritized approach is multiplied by the equivalent queue number k of the proposed heuristic that offers an equivalent delay.

Profit-wise, the proposed heuristic outperforms the baseline non-prioritized processing approach in moderate and high congestion scenarios as shown in Fig.2.7. When the capacity of the baseline non-prioritized processor queue is increased to 150%, the proposed heuristic still collects more profit than the baseline non-prioritized processing approach, as shown in Fig.2.7 (c). Furthermore, Fig.2.7 (a), illustrates the superior performance of the proposed heuristic when the baseline non-prioritized processing queue capacity is reduced by half.

The non-prioritized processor accepts almost the same or sometimes more flows compared to the proposed heuristic as indicated in Fig.2.8. However, the extra flows processed by the baseline non-prioritized approach can be ignored, since the proposed heuristic still gets more profit.

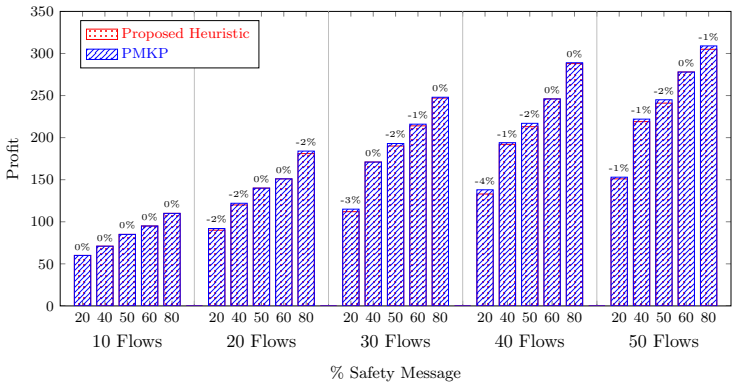
To measure the quality of service, the worst-case delay is observed, which is the maximum delay for every queue be it prioritized or non-prioritized. Fig.2.9 shows that all four queues of the proposed heuristic guarantee a delay level for all flows based on their severity. This delay level never exceeds the required QoS limit. Contrary to our approach, the non-prioritized

approach provides no delay guarantees, which results in high severity flows suffering long processing delays. In other words, all flows regardless of their severity levels encounter similar processing delays when the baseline non-prioritized approach is used. Conversely, our proposed prioritized processing heuristic fulfills its promise of processing flows based on their severity level and processing them within the QoS and time requirements.

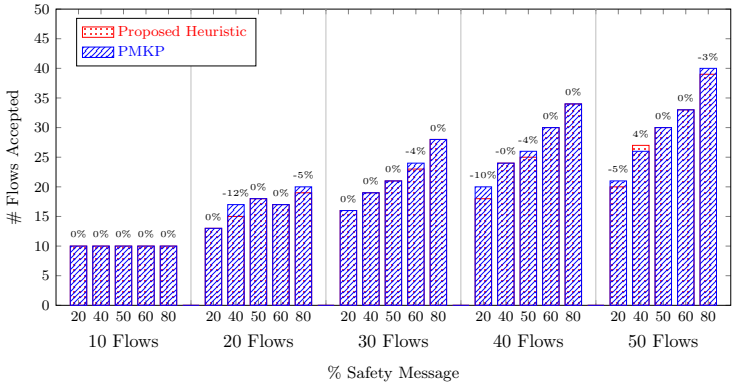
2.8. Future Directions and Conclusion

In this paper, we presented a generic real-time heuristic that provides differentiated services based on a given set of flows and their corresponding *severity* metric. Furthermore, we demonstrated that the proposed heuristic can be used to offer differentiated services and improve QoS in VANETs. The proposed heuristic intelligently prioritizes the processing of flows in VANETs based on their corresponding severity metric. The problem is formulated as a PMKP, which is proved to be NP-Hard. Due to the complexity of the PMKP, a polynomial-time algorithm based on a relaxed version of the PMKP formulation is proposed to perform the desired prioritization in real-time. The proposed heuristic is tested against the PMKP solution and a baseline non-prioritized processing approach. Results obtained through simulations with real traffic data demonstrated a minor difference in performance between the proposed heuristic and the PMKP. On the other hand, the proposed heuristic surpasses the baseline non-prioritized approach by 9% to 67% more profit in moderate and high congestion scenarios. As the results suggest, differentiated services are not required when the system has resources to satisfy all the requests, but rather when the system is under higher loads. In such scenarios, results show that our proposed prioritized processing heuristic is superior and provides better performance.

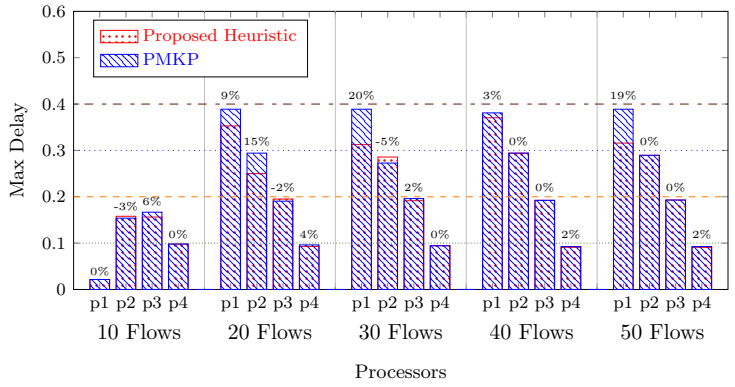
In our future research, we plan to pursue applications of our proposed approach beyond VANETs. Specifically, we plan to explore the potential use of our approach in support of



(a) Profit

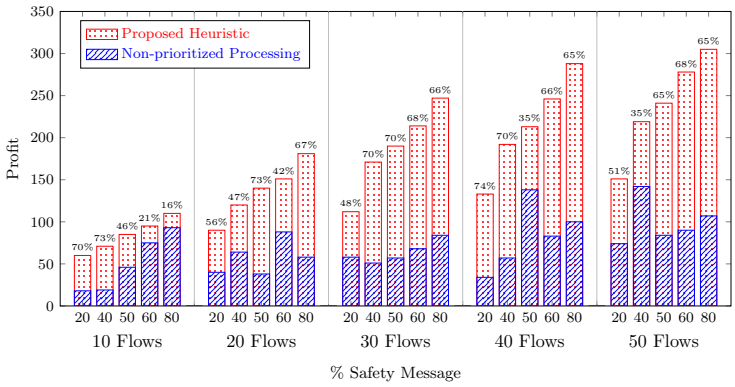


(b) Accepted flows

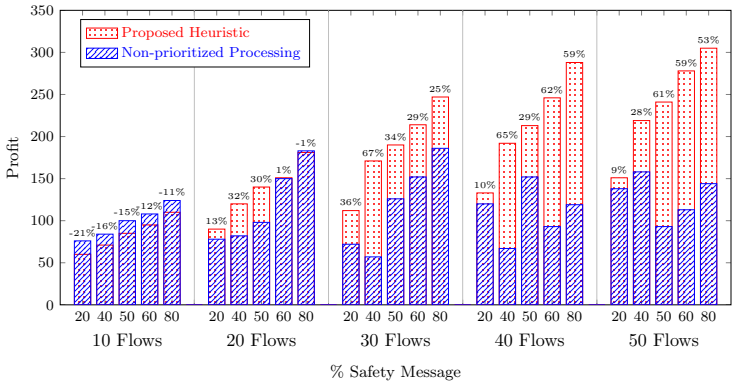


(c) Average load

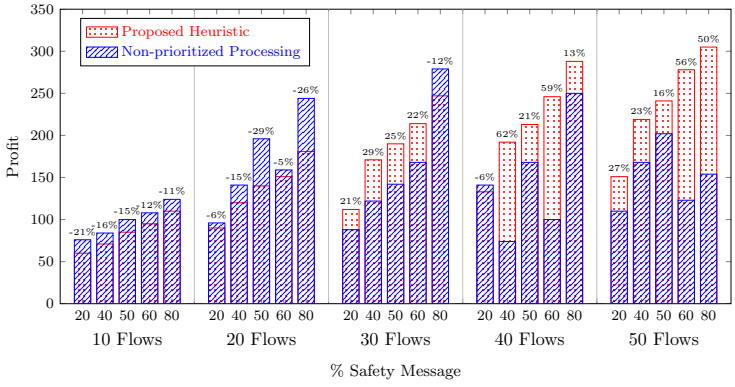
Figure 2.6: Performance of the proposed heuristic compared to the PMKP.



(a) Non-prioritized processor capacity is 50% of the combined proposed heuristic capacity

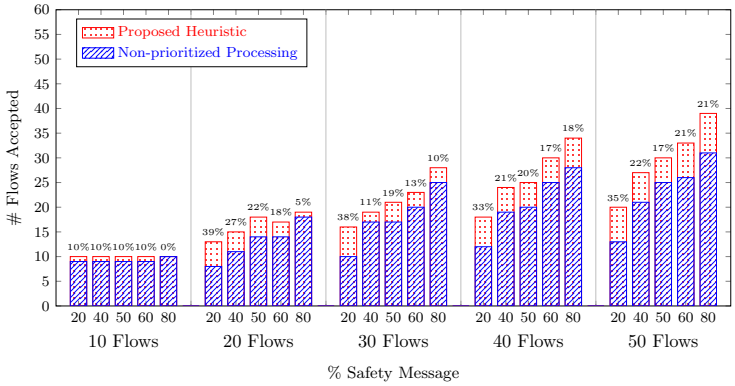


(b) Non-prioritized processor capacity is the same as the combined proposed heuristic capacity

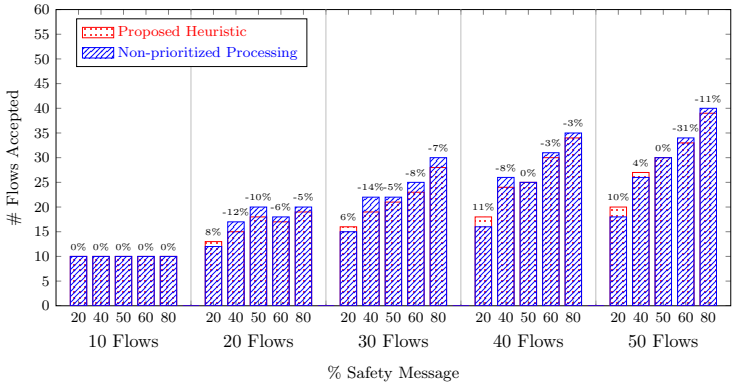


(c) Non-prioritized processor capacity is 150% of the combined proposed heuristic capacity

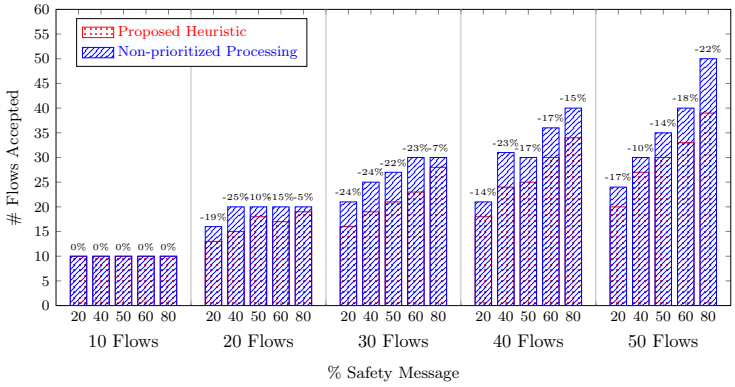
Figure 2.7: Profit of the proposed heuristic compared to the non-prioritized processing algorithm.



(a) Non-prioritized processor capacity is 50% of the combined proposed heuristic capacity

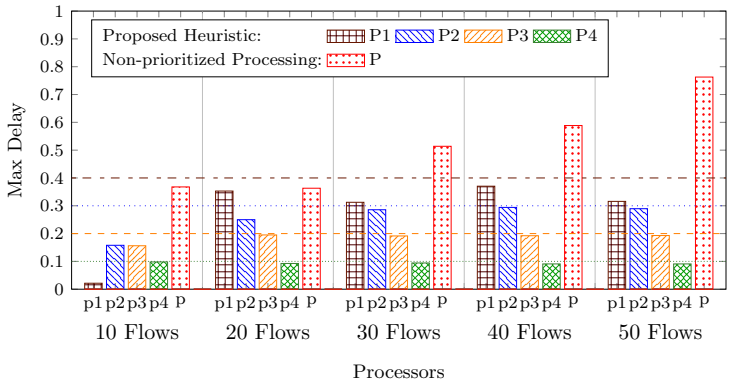


(b) Non-prioritized processor capacity is the same as the combined proposed heuristic capacity

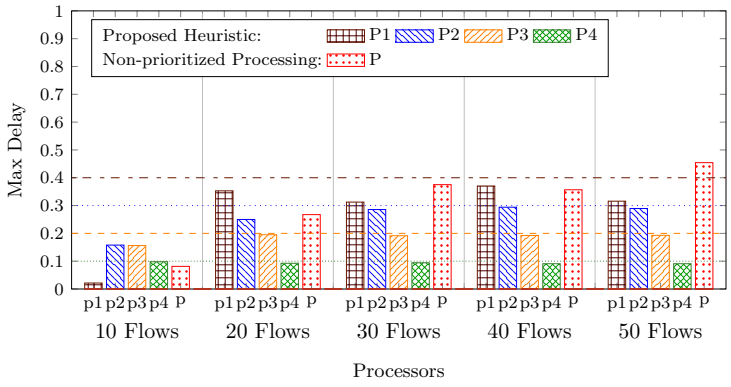


(c) Non-prioritized processor capacity is 150% of the combined proposed heuristic capacity

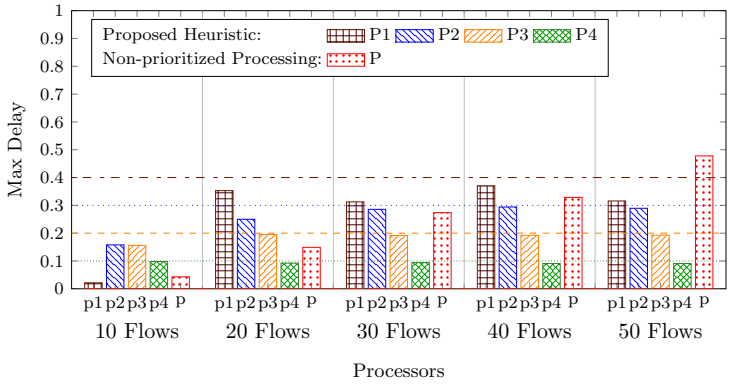
Figure 2.8: Number of accepted flows of the proposed heuristic compared to the non-prioritized processing algorithm.



(a) Non-prioritized processing capacity is 50% of the combined proposed heuristic capacity



(b) Non-prioritized processing capacity is the same as the combined proposed heuristic capacity



(c) Non-prioritized capacity is 150% of the combined proposed heuristic capacity

Figure 2.9: Maximum delay of the proposed heuristic compared to the non-prioritized processing algorithm.

Industrial IoT applications (IIoT) with real-time QoS constraints below 10 ms.

2.9. Acknowledgment

This publication was made possible by NPRP grant # [7-1113-1-199] from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

CHAPTER 3

Opportunistic Data Ferrying in Areas with Limited Information and Communications Infrastructure

3.1. Introduction

It is estimated that about 60% of the world's population will live in cities by 2030 [41]. Additionally, by the year 2020, around 20.4 billion devices are expected to be connected to the Internet [42]. To cope with the trend of people moving to urban centers and to provide high-quality services to their residents, municipalities are increasingly turning to Information and Communications Technologies (ICT)—such as cloud computing, Internet of Things (IoT), Wireless Sensor Networks (WSNs), and Cyber-Physical Systems (CPS) [43]—for the deployment of smart community applications that provide value-added services in diverse fields such as healthcare, transportation, entertainment, and governance [44].

But such smart community deployments are often prohibitively expensive, especially for smaller communities where the deployment of smart community applications is hindered by the unavailability of appropriate communication infrastructure. One way to address this concern is to exploit existing infrastructure in innovative ways. In particular, modern vehicles that abundantly ply the roads of urban cities can be exploited to obviate the need for an expensive communications infrastructure. In recent times, with increased interest in vehicular ad-hoc networks (VANETs) and self-driving cars, vehicles are increasingly becoming more sophisticated and it is expected that by the year 2020, 90% of vehicles will be equipped with a hardware-based On-Board Unit (OBU) [45] that has processing and communications capabilities. Therefore developing an approach for opportunistically accessing these smart vehicles in an efficient delay-tolerant manner becomes a promising approach

towards the deployment of cost-effective and efficient smart community applications with limited ICT infrastructure overhead. For example, rural areas that lack the funds to deploy ICT infrastructure can benefit from our proposed approach.

One approach of delivering data in sparse networks is Message Ferrying (MF). In this approach, devices are classified as *message ferries* (or *ferries*) or *regular nodes*. Ferries are mobile devices that move around to collect messages from regular nodes to deliver them to their destination [46]. In this paper, we use this approach and utilize vehicles as ferries to transfer data collected from the smart devices (i.e., regular nodes) to the Smart Community Management Center (SCMC). One example of the SCMC is the Traffic Management Center (TMC), which is used for managing traffic in support of intelligent transportation applications.

In this paper, we envision a smart community application architecture where the service area is divided into blocks, each having at least a single Local Community Broker (LCB), where an LCB is a cloudlet that is deployed in the block or hosted on a vehicle that has processing and communications capabilities. Each LCB serves as a block manager responsible for selecting vehicles to transfer data collected from IoT devices scattered across the block to the Smart Community Management Center (SCMC). When a vehicle passes through the block, the block manager (i.e., LCB) acquires the estimated delivery delay from the vehicle and decides on whether to utilize it as a ferry to transfer collected data to the SCMC. Consequently, the vehicles themselves are used as ferries to transfer data between the different blocks of the service area. Actually, sparsely deployed LCBs are the only required infrastructure in our proposed architecture, and no communications infrastructure (optical, microwave, or 5G base stations) is required to relay the data from the LCBs to the SCMC.

The main challenge in this architecture lies in the 'intelligent' selection of vehicles. If

the block manager is not selective, it may end up using vehicles with high delivery delay as any vehicle might be selected even if it has a high delivery delay—delivery delay is the time taken to transfer a data bundle on a vehicle selected to serve as a data ferry from the LCB to the SCMC. On the other hand, if it is very selective and picks vehicles that have a short delivery delay, it would incur a high waiting delay as it needs to wait longer to find such vehicles—waiting delay is the time taken to select a vehicle to serve as a data ferry. In fact, this problem is of an online nature because once a vehicle leaves the block, the block manager can no longer utilize the vehicle. Consequently, the block manager may regret its decision when the delivery delay of future vehicles is shorter than that of previous ones.

To solve this problem, we propose an online algorithm that utilizes an ensemble of online hiring algorithms. The proposed algorithm opportunistically selects the best performing algorithm from its ensemble based on the recent observed performance. The average overall delay is the sum of the average waiting delay and the average delivery delay as detailed later in the paper. The proposed algorithm selects one algorithm from its ensemble to be active with the objective of minimizing the average overall delay. All other algorithms in the ensemble are set to be passive (i.e., inactive). This way, the active algorithm alone selects the vehicles to be used as data ferries. Although other algorithms in the ensemble will not be utilized to select the data ferries, their performance is utilized in order to choose the best performing algorithm in the ensemble by choosing the one that demonstrated the lowest average overall delay in the recent history (i.e., greedy approach). Consequently, different hiring algorithms from the ensemble might be utilized over time.

To the best of our knowledge, this is the first research effort that utilizes online hiring algorithms for the selection of data ferries in the specific setting of smart community applications. We perform a thorough evaluation of our work and show that our proposed algorithm performs better than other state-of-the-art online hiring algorithms in a wide variety

of settings (including for different traffic volumes).

3.2. Related Work

The literature is rich with research that deals with network issues in smart communities. Jawhar et al. [43] presented the networking requirements for different smart city applications and additionally presented network architectures for different smart city systems. In [44], the authors discussed the networking and communications challenges encountered in smart cities. Paradells et al. [47] state that deploying wireless sensor networks along with the aggregation network in different locations in the smart city is very costly and consequently propose an infrastructure-less approach in which vehicles equipped with sensors is used to collect data.

Bouroumine et al. [48] present a system where public and semi-public vehicles are used for transporting data between stations distributed around the city and the main server. Aloqaily et al. [49] introduce the concept of Smart Vehicle as a Service (SVaaS). They predict the future location of the vehicle in order to guarantee a continuous vehicle service in smart cities. In another work [50], the authors indicate that cars will be the building blocks for future smart cities due to their mobility, communications, and processing capabilities. They propose Car4ICT, an architecture that uses cars as the main ICT resource in a smart city. The authors in [51] propose an algorithm for collecting and forwarding data through vehicles in a multi-hop fashion in smart cities. They proposed a ranking system in which vehicles are ranked based on the connection time between the OBU and the RSU. The authors claim that their ranking system results in a better delivery ratio and decrease the number of replicated messages.

In [52], the authors state that existing network infrastructure in smart cities can not sustain the traffic generated by sensors. To overcome this problem, an investment in telecommunication infrastructure is required. However, the authors proposed to exploit buses in a Delay Tolerant Network (DTN) to transfer data in smart cities. In [53], the authors introduce mobile cloud servers by installing servers on vehicles and use them in relief efforts of large-scale disasters to collect and share data. These mobile cloud servers convey data among isolated shelters while traveling and finally returning to the disaster relief headquarters. Vehicles exchange data while waiting in the disaster relief headquarters, which is connected to the Internet.

Bonola et al. [54] conduct a study on using taxi cabs as oblivious data mules for data collection and delivery in smart cities. They have no guarantee of data communications since they are using taxi cabs without any selection criteria. They use real taxi traces in the city of Rome and divide the city into blocks of size 40×40 meter^2. Depending only on opportunistic connections between vehicles and nodes, the authors claim to achieve a coverage of 80% of the downtown area over a 24 hour period.

The aforementioned papers mostly utilize multiple relays for transferring data between source-destination locations. Furthermore, these papers do not approach the ferry selection problem from an online perspective. Conversely, in this paper we propose an approach where each vehicle transfers a data bundle from source to destination without having to use relays and decisions are made in an online fashion—these assumptions are practical as more vehicles utilize OBU and GPS units that provide exact or probabilistic information about the path of the vehicle. Additionally, this paper considers online hiring algorithms for data ferry selection.

3.3. System Model

To utilize data ferrying in a given area, we divide the service area (e.g., city) into B blocks. Each block has a number of smart devices (e.g., sensor nodes) that generate data. In addition, one of these blocks hosts the SCMC as shown in Figure 3.1. Vehicles are used to transfer data collected from smart devices to the SCMC. Also, each block has one LCB positioned near the center of the block. Any vehicle that enters a given block is within the communications range of its LCB.

In this paper, we make two assumptions about incoming vehicles. First, it is known if the vehicle will pass through the SCMC in the future. Second, for vehicles that pass through the SCMC, the expected arrival time is known. Our assumptions are based on many research efforts that appeared in the recent literature [55]–[58] to predict those parameters.

Each LCB serves as a block manager that contacts incoming vehicles to acquire two pieces of information; namely, whether the vehicle is going to pass through the SCMC block, and at what time (i.e., expected arrival time at the SCMC block). If a vehicle is going to visit the SCMC block, the block manager computes its expected arrival time. Using the expected arrival time, the block manager computes the delivery delay d, which is the time required by the vehicle to transfer the data from the current block to the SCMC block. Besides, the block manager computes the waiting delay w, which is the time between the selection of the last vehicle and the selection of the current vehicle. In other words, w is the time the data must wait until a vehicle is selected to serve as a data ferry.

When a vehicle passes through a block, the block manager, running the proposed algorithm, makes a decision on whether to accept or reject the current vehicle to serve as a data ferry. The block manager computes the delivery delay d and the waiting delay w and passes them to the proposed algorithm. Once the proposed algorithm makes a decision, it

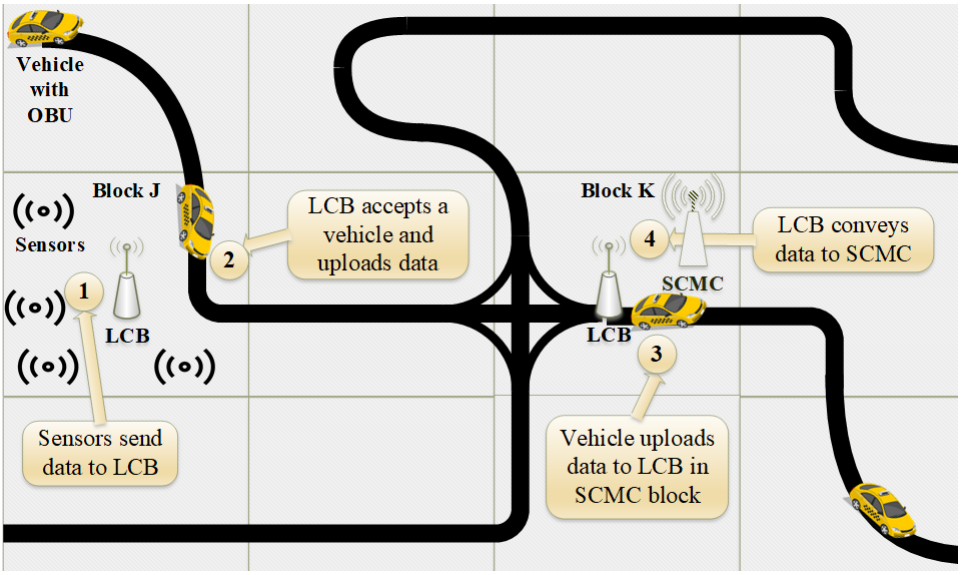


Figure 3.1: Ferrying data from different blocks to the SCMC block in a community using vehicles.

is impossible for that decision to be altered. The proposed algorithm utilizes an ensemble of online hiring algorithm as explained in Section 3.5. If more than one vehicle exists in a block, and these vehicles are going to pass through the SCMC block sometime in the future, the block manager considers only the vehicle with the minimum d.

Once a vehicle is selected, the LCB uploads data of block j to the vehicle. Next, the vehicle continues its trip and eventually passing through block k, which has the SCMC. Once in block k, the OBU of the vehicle uploads data collected from block j to the LCB of block k, which in turn conveys it to the SCMC. These steps are illustrated in Fig 3.1.

3.4. Online Hiring Algorithms

Many companies around the world use hiring algorithms to select employees instead of the traditional manual selection process. Actually, there are many flavors of the hiring algorithm. In [59] [60], researchers investigate the performance of the different heuristics

for the *classical secretary problem* that select the best candidate out of multiple candidates. The problem involves an interviewer interviewing n candidates one at a time for a position and then deciding after each interview if the interviewee is the best candidate. The overall goal in this problem that seeks to decide under uncertainty is to maximize the probability of choosing the best candidate. To select the best candidate, the authors introduce three hiring algorithms [59] including (1) hire above a threshold, (2) hire above minimum or maximum, and (3) hire above mean or median (Lake Wobegon). We provide a brief description of the algorithms used in this work next.

- *Hiring Above a Threshold*: In this version of the hiring algorithm, vehicle i is selected only if the delivery delay d_i is less than or equal to a fixed threshold τ.

- *Hiring Above the Mean*: In this hiring algorithm version, vehicle i is selected only if the delivery delay d_i is less than a_b (the average delivery delay of selected vehicles in block b). Initially, the algorithm accepts the first vehicle that enters block b and then sets a_b to d_0, and subsequently, a_b decreases gradually as the algorithm accepts more vehicles.

- *Hiring Above the Median*: Hiring above the median uses m_b (the median of the delivery delay of all selected vehicles in block b). Like hiring above the mean, this algorithm initially accepts the first vehicle that enters block b. This algorithm needs an odd number of selected vehicles before recomputing m_b since m_b is the value in the middle after sorting. Therefore, after selecting a vehicle, if the number of selected vehicles is even, the algorithm does not update m_b postponing the update of m_b to the situation where the number of vehicles is odd.

3.5. Heuristic Solution

In this work, we propose an algorithm that strives to minimize the average overall delay for transporting data from one block to the SCMC block. The overall delay is the sum of the waiting delay and the delivery delay. The idea is simply to run an ensemble of N online hiring algorithms in passive mode while selecting only one of them to be active at any point in time. By passive, we mean an algorithm makes a decision for whether a given vehicle should be selected to serve as a data ferry but the decision is not executed. This is done in order to collect performance metrics needed to compare the performance of the different algorithms in the ensemble.

The proposed algorithm utilizes four hiring algorithms; namely, low threshold, high threshold, mean, and median. These algorithms can only consider the delivery delay and cannot take the waiting delay into consideration. In other words, they cannot make a decision based on the overall delay which includes the waiting delay. This stems from the fact that the waiting delay can be larger than the threshold used by those algorithms. Consequently, those algorithms will reject all requests after that time and will be stuck in this state forever. For example, if the threshold of the low threshold algorithm is set to 50 minutes and the algorithm is waiting for more than 50 minutes (i.e., waiting delay is greater than 50) then the overall delay will always be greater than 50 even if the delivery delay is zero. Thus, the low threshold algorithm will reject vehicles from serving as data ferries indefinitely. However, the proposed algorithm is capable of analyzing the history of all algorithms in its ensemble in terms of the overall delay. Moreover, to be efficient, the proposed algorithm has the chance to switch between the four algorithms in its ensemble every S time units in case the performance of the already selected algorithm deteriorates.

Algorithm (2) shows the three parts of the proposed algorithm. One of the four algorithms

Algorithm 2 Proposed algorithm for selecting ferries

Input: vehicle arrival time A.
Output: decision (accept or reject)
Initialization *(executed once at time 0)*:
1: Set all algorithms as passive
2: Set activeAlgorithm = select an algorithm randomly.
On vehicle arrival:
3: Set $d = A-$ current time
4: Set w = current time - last time a vehicle was accepted
5: **for** each of the 4 algorithms **do**
6: Run the algorithm
7: **if** Decision is accept **then**
8: Save overall delay as $d + w$
9: **if** this algorithm is activeAlgorithm **then**
10: Accept the vehicle
11: **end if**
12: **else if** this algorithm is activeAlgorithm **then**
13: Reject the vehicle
14: **end if**
15: **end for**
Executed every S ***minutes:***
16: Compute average overall delay based on saved data
17: Set bestAlgorithm = algorithm with minimum delay
18: **if** activeAlgorithm $\neq$ bestAlgorithm **then**
19: Set activeAlgorithm = bestAlgorithm
20: **end if**

in the ensemble is selected randomly to serve as the active algorithm in the initialization part, which is executed once when the algorithm starts as indicated in lines 1 and 2. The second part is executed whenever a vehicle arrives and a decision needs to be made on whether to select it as a data ferry. In the second part, the four algorithms in the ensemble are executed and the overall delay of each one is saved for later analysis in the third part. Moreover, the decision made by the active algorithm is committed while the decisions of other algorithms are ignored. Lines 3 to 15 represent the second part. The last part is only executed after S time units had passed, which is represented by lines 16 to 20. Furthermore, the average overall delay of all algorithms is computed based on saved data and the algorithm with the minimum average overall delay is set as the active algorithm while setting the other algorithms as passive.

Table 3.1: Performance (average overall delay) of the proposed algorithm compared to the two online hiring algorithms

Time	Algorithm A	Algorithm B	Proposed	
	Avg. delay	Avg. delay	Avg. delay	Selection
t_0	0	0	0	Algorithm B
t_1	8	10	10	Algorithm A
t_2	12	22	14	Algorithm A
t_3	18	26	20	Algorithm B
t_4	30	28	22	Algorithm B

3.6. Illustrative Example

Let Algorithm A and Algorithm B be two online hiring algorithms with Algorithm A initialized in passive mode and Algorithm B initialized in active mode (i.e., selected randomly by the proposed algorithm) at t_0. Table 3.1 shows the average overall delay per algorithm recorded every S minutes (see Algorithm 2), which is computed as the average of overall delays in the period t_i to t_j, where $j = i + 1$. The performance of the proposed algorithm is not the best at t_1 since it randomly selects Algorithm B as the active algorithm between t_0 and t_1, which performs worst than Algorithm A. However, the proposed algorithm switches to Algorithm A at t_1 and got an average overall delay of 4 minutes between t_1 and t_2, which is the same value gained by Algorithm A. Between t_2 and t_3, Algorithm B performs better than Algorithm A leading the proposed algorithm to get 6 minutes instead of 2 minutes. In t_3, the proposed algorithm switches to Algorithm B and get the minimum average overall delay of 2 minutes between t_3 and t_4. By t_4, the proposed algorithm achieves less average overall delay compared to both algorithms.

3.7. Experimental Results

In this section, we describe the dataset used in our experiments; explain the experiments' settings; evaluate the performance of the proposed online algorithm by comparing it with

four baseline online hiring algorithms using real vehicular traces; and finally, discuss the results and present the major insights learned from our experiments.

3.7.1. Dataset & Experimental Settings

In our experiments, we make use of the Shanghai dataset consists of taxi traces collected in the city of Shanghai in China. Each taxi has a GPS unit and a GPRS wireless communications modem. Vehicles send their GPS location along with other information to a data center every minute. Around 2,109 taxis participated in this dataset in 2007. Information sent by taxis includes ID, timestamp, longitude and latitude, speed, and heading direction [61].

In order to utilize the Shanghai dataset, we have encoded the geographical location that encompasses longitude and latitude into a string of 7 characters using the `GeoHashing` method [62]. Every string represents a grid (i.e., block) of the city. Actually, we used a `GeoHashing` of 7 characters because this allows for the division of the globe into blocks, each of 153×153 meters, which is within the communications coverage of a typical LCB. Using these blocks, we position the SCMC in the block that is mostly visited by vehicles. Additionally, we filtered the dataset to remove blocks that have no traffic activity and focus on the active blocks. The dataset is based on a one-day observation. However, one day is a very short period for the proposed algorithm to work effectively. Therefore, we replicate the one-day data a number of times to have datasets for 5, 10, 15, 20, and 25 continuous days.

To study the performance of the proposed algorithm under different traffic scenarios, we divide the city into three areas based on the traffic volume. We computed the average number of vehicles per block along with the standard deviation and found that the standard deviation is greater than the average. Therefore, we categorized each block based on N_b, the number of vehicles in block b, as follows:

- *Light traffic area*: $N_b <$ average
- *Medium traffic are*a: average $\leq N_b \leq$ standard deviation
- *High traffic area*: $N_b >$ standard deviation

We set S to 30 minutes in all of the experiments to be consistent. Also, to derive the low and high threshold values for the threshold algorithms in every block, we used a percentile of the delivery delay in the block. The dataset used in the experiment has a heavy-tailed distribution and particularly a long-tail distribution and we resort to extremely low and high threshold values—2^{nd} percentile for the low threshold algorithm and 95^{th} percentile for the high threshold algorithm—to fully explore the space of values in such a distribution.

3.7.2. Results Discussion

3.7.2.1. Evaluation of the proposed algorithm for different traffic volumes Considering the four baseline hiring algorithms only, it can be clearly seen that a different one outperforms in each area depending on the traffic volume. The mean algorithm is the best in light traffic areas, the high threshold is the best in medium traffic areas, and the low threshold is the best in high traffic areas as shown in Fig. 3.2.

The low threshold algorithm suffers from a high overall delay in the light traffic areas because the waiting delay is very high since it is very selective. However, the low threshold algorithm outperforms in the area of the high traffic since it only selects vehicles with low delivery delay and there are plenty of vehicles to pick from. On the other hand, the high threshold algorithm performs best in the medium traffic areas, which provide a balance between waiting delay and delivery delay. Since the high threshold algorithm accepts the majority of vehicles, it benefits from this balance. As for the mean and median algorithms,

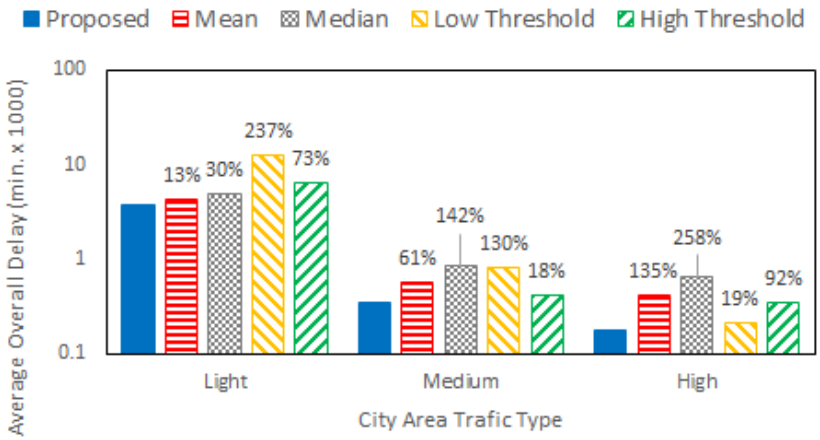


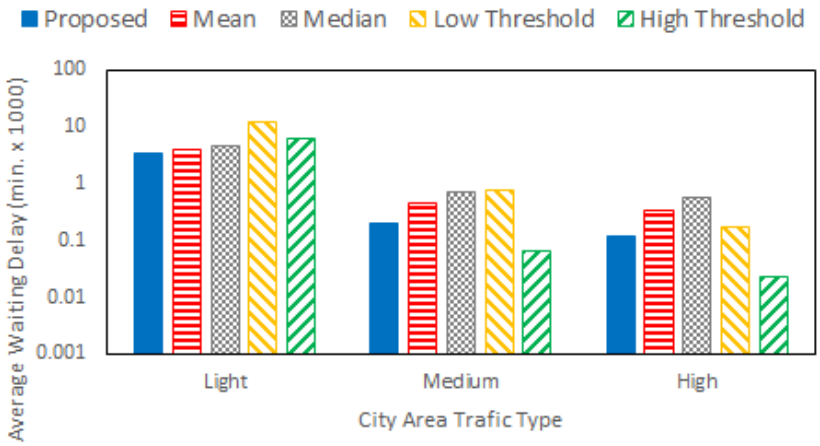
Figure 3.2: Average overall delay per block. *Our proposed algorithm achieves minimal overall delay per block regardless of the traffic volume.*

their performance is best in the areas of light traffic. This is because the thresholds of these algorithms decrease with more vehicles. The more these algorithms accept, the more greedy they become towards a lower threshold.

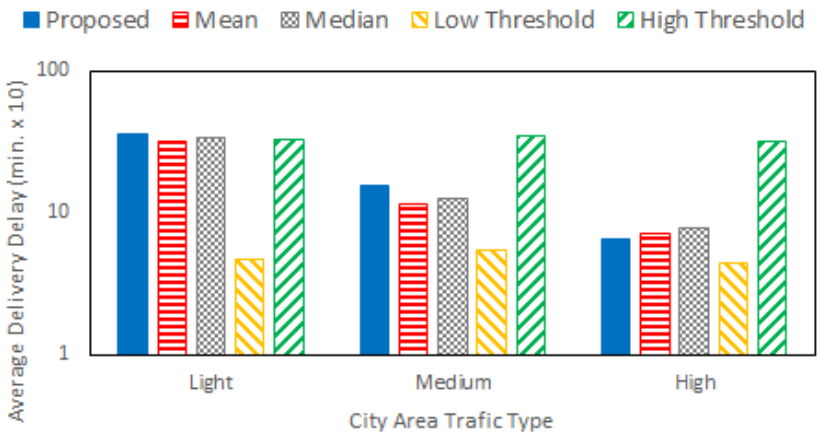
The proposed algorithm achieves the best results in all areas with up to 258% less overall delay albeit at a cost. To understand this cost, we focus on the 10 days results and record the number of selected vehicles, average delivery delay, average waiting delay, and average overall delay.

The proposed algorithm outperforms the baselines algorithms regardless of the traffic volume by either performing better on the waiting delay or on the delivery delay but not both as indicated in Fig. 3.3.

It should be noted that the proposed algorithm does not only perform better in terms of the average overall delay but it also accepts more vehicles to serve as data ferries as shown in Fig. 3.4 (with the exception of the high threshold algorithm since it accepts the majority of vehicles in all areas).



(a) Average waiting delay per block.



(b) Average delivery delay per block.

Figure 3.3: Average waiting & delivery delay per block. *Our proposed algorithm performs the best either with minimal average waiting delay or average delivery delay per block, but not with both.*

3.7.2.2. Evaluation of the proposed algorithm for different time periods To assess the performance of the proposed algorithm relative to the four baseline hiring algorithms, we run the algorithms for a different number of days. Results are collected in terms of the average overall delay in each of the three different areas as shown in Fig. 3.5. The figure shows the consistent behavior of the proposed algorithm regardless of the number of days.

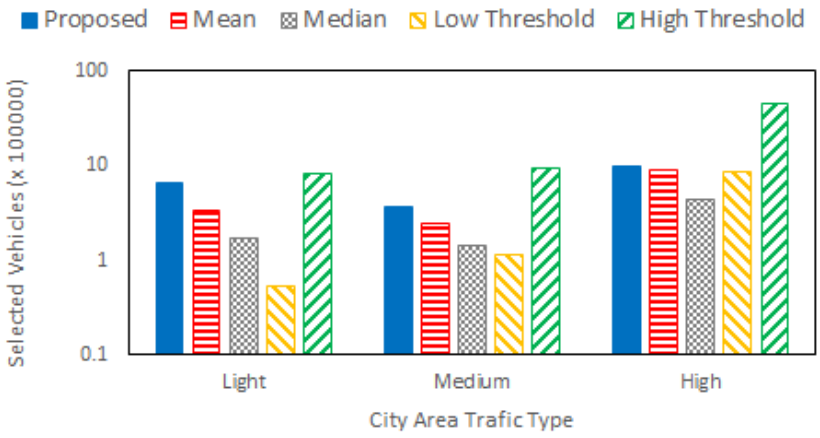
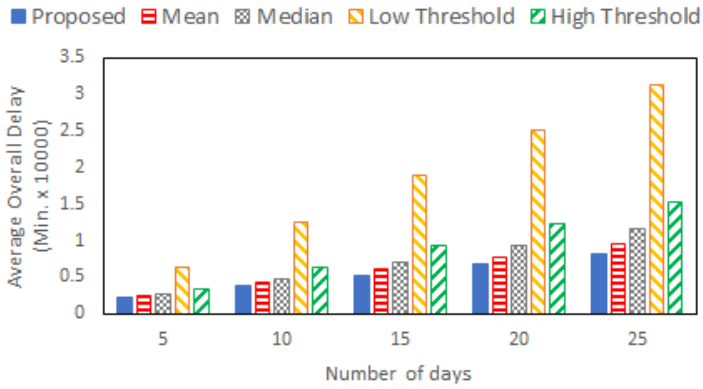


Figure 3.4: Number of selected vehicles. *Our proposed algorithm hires the most number of vehicles with the exception of the high threshold algorithm.*

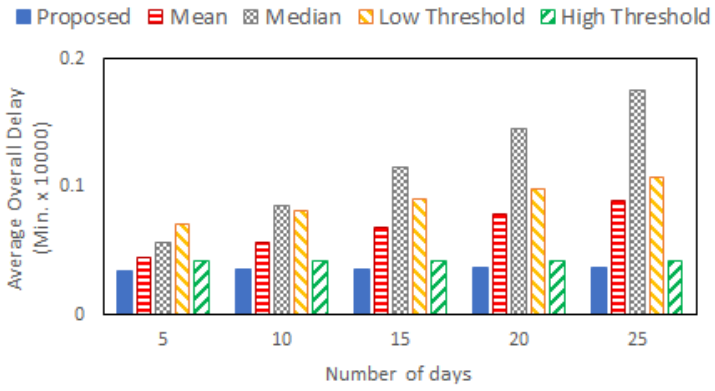
3.7.2.3. Switching activity of the proposed algorithm To show the switching activity of the proposed algorithm, we record the average overall delay every hour for one block over 10 days as illustrated in Fig. 3.6. The figure shows how some algorithms perform better for a period of time and how the proposed algorithm follows the one with the minimum average overall delay based on performance collected from recent history.

3.8. Conclusions and Future Work

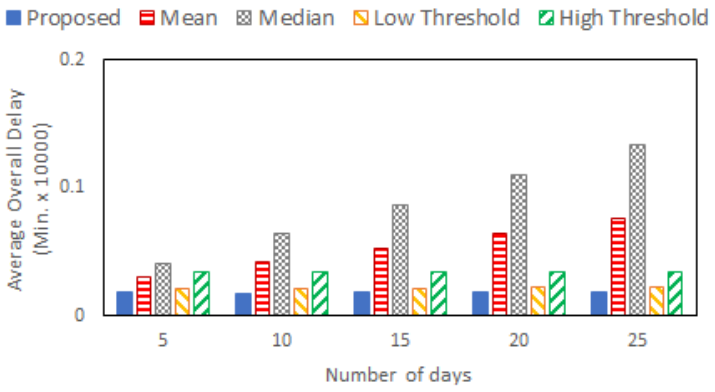
In this paper, the problem of selecting vehicles to serve as data ferries in support of smart community applications is considered. The selection process strives to achieve the minimum average overall delay. An online algorithm is proposed that utilizes four online hiring algorithms by running all of them together in passive mode and selecting the one that has performed the best in recent history. The proposed algorithm is evaluated using real taxi traces from the city of Shanghai in China and compared against a baseline of four online hiring algorithms. Experiments with these traces demonstrate that the proposed algorithm outperforms online hiring algorithms presented in the literature regardless of the



(a) Light traffic area.



(b) Medium traffic area.



(c) High traffic area.

Figure 3.5: Average overall delay using different algorithms for the various traffic scenarios (light, medium, high) over different days. *Our proposed algorithm achieves the minimal overall delay for different number of days results regardless of the traffic volume.*

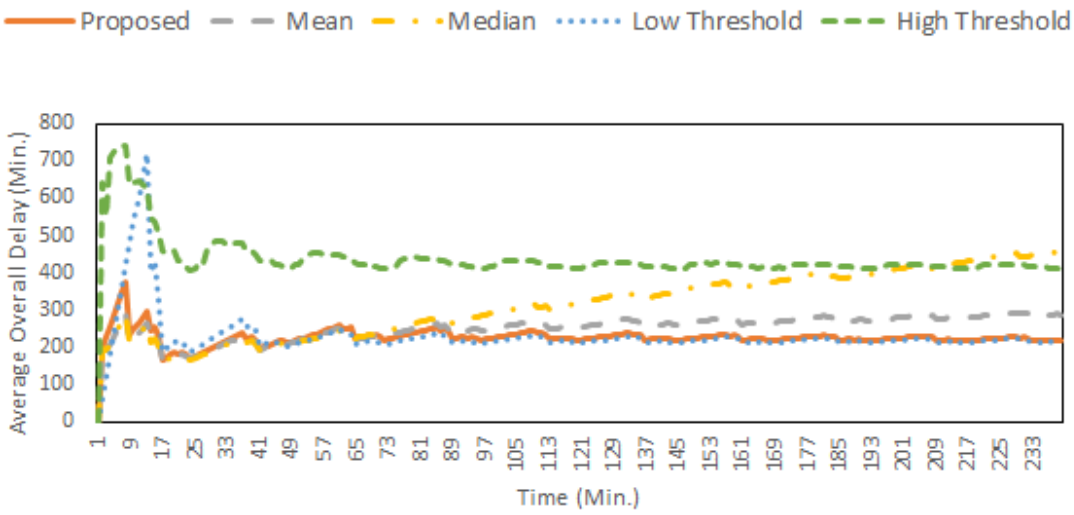


Figure 3.6: Performance of algorithms in one block over 10 days. *Our proposed algorithm switches between different hiring algorithms to achieve minimal overall delay.*

traffic volume by either performing better on the waiting delay or on the delivery delay but not both.

In the future, we plan to evaluate the proposed algorithm analytically to provide performance guarantees, in terms of competitive ratio, in worst-case scenarios.

CHAPTER 4

Budgeted Online Selection of Candidate Clients to Participate in Federated Learning

4.1. Introduction

The fourth industrial revolution (Industry 4.0) promises the provisioning of smart services that enhance the manufacturing process by utilizing emerging technologies such as Internet of Things (IoT) and Artificial Intelligence (AI) [63] [64]. In particular, most of the recent advances in Industry 4.0 and AI are driven by Machine Learning (ML), a branch of AI, and more specifically by Deep Learning (DL) [63] [65] [66].

The ML and DL techniques however require a large amount of data for the training of their models. In particular, serious privacy and security concerns crop up when we have to collect and process data scattered over different organizations and users [67] [68]. For instance, the prediction of patient mortality using Electronic Health Record (EHR) data dispersed over many hospitals is a complex undertaking due to the various privacy, security, regulatory, and operational issues [69]. Additionally, the communication of potentially large amounts of data from the clients to a central server is costly and can choke the networks when limited bandwidth is available [70]. Such bottlenecks can be observed in Vehicular Edge Computing (VEC) where vehicles have to send their data such as images to roadside servers to build models, which results in the networks being greatly burdened [71].

To address the issues of security, privacy, and excessive communication cost, the technique of Federated Learning (FL) [72], a distributed ML approach that runs on a server and multiple clients, was proposed. The server and the clients use the same model architecture. The server initiates the global model (i.e., the server model) and executes the following steps

over several communication rounds [70] [72]:

- The server sends the global model's parameters to some (or if possible all) clients;
- Every participating client uses the received global parameters to train the local model using the local dataset;
- Every participating client sends the local model parameters to the server;
- The server aggregates the local parameters received from the clients to update the global model;
- Eventually, the accuracy of the global model converges to some threshold.

In FL, the server has no access to the client's local dataset since only the local model parameters are shared with the server. Consequently, privacy and security are preserved and communication cost is reduced. However, FL suffers from the following two problems [73]:

- Convergence may take a long time, which increases the communication cost.
- Clients have different computation, storage, and communication resources and different dataset sizes, which makes the task of selecting clients a challenge.

FL can be *stateful* or *stateless*. In stateful FL, a candidate client can participate in each of the communication and computation rounds used in training the global model and thus the state is preserved between rounds. Nevertheless, in stateless FL, a candidate client will likely participate in one communication and computation round to train the global model, which means in each round, new fresh candidate clients are utilized [74].

In this paper, we propose a stateful FL model with a budgeted number of candidate clients to overcome communication and computation constraints. In other words, from a total of N

candidate clients, we select the best $R < N$ candidate clients to participate in training the global model. Now, some candidate clients become available while other candidate clients become offline or out of communication range over time. Also, we assume that not all candidate clients are available at the same time. Meaning that the problem of selecting R candidate clients is an online problem. As a result, the selection of candidate clients is a challenge. In offline problems, information about all candidate clients are well known in advance rendering the problem of selecting the best R candidate clients trivial. However, in online problems, once a candidate client becomes available then an irrevocable decision must be made on the selection of this candidate client without any prior knowledge about coming candidate clients. Consequently, we propose a budgeted online selection algorithm that selects the best R candidate clients based on their evaluated test accuracy. The proposed algorithm is inspired by the solution of the secretary problem.

The proposed algorithm can be used in different applications, particularly for online applications with intermittently available mobile clients. Once a client is available, a decision must be made on whether to utilize the client or not since the client may become unreachable like out of communication range or offline. However, once the client is selected, the client will be utilized. Furthermore, decisions cannot be revoked in online applications but might be regretted.

Detection and identification of unauthorized IoT devices are very important especially with the increase in the number of attacks on IoT devices [75]. Therefore, we propose a clients' alarm application that alerts clients about unauthorized IoT devices in their environment. Each client uses a local machine (i.e. server) to monitor the traffic generated by IoT devices in the environment and extract features based on IoT device behavior. Extracted features are used to identify the IoT device type by training an ML model on those features. This is known as *IoT device type classification*. However, clients can not identify unknown

IoT devices in their environment depending only on the local dataset. Therefore, clients subscribe to the alarm service provided by the server on the cloud that utilizes the proposed algorithm. Clients share their model's parameters with the server to train a global model capable of identifying unauthorized IoT devices.

The salient *contributions of this paper* are:

- We propose a model for optimizing accuracy in stateful federated learning by selecting the best candidate clients based on test accuracy. We formulate the problem of maximizing the probability of selecting the best R candidate clients based on test accuracy from N total candidate clients as a secretary problem and analytically analyze the performance and provide proofs.

- We propose an online heuristic solution for optimal budgeted client selection based on test accuracy inspired by the secretary problem that works in stateful FL settings. This is the first work as far as we know that utilizes online resources selection in federated learning.

- We propose a client alarm application for identifying unauthorized IoT devices using the proposed algorithm and IoT device type classification. We conduct many experiments to evaluate the performance of the proposed heuristic against other state-of-the-art algorithms. Results show an improvement of up to 27% in accuracy compared with the online random algorithm and an accuracy gain of approximately 10% compared with the offline best algorithm.

The *organization of the remainder of the paper* is as follows. Background regarding FL is discussed in section 4.2. Related literature is reviewed in Section 4.3. The proposed system is described in Section 4.4 while Section 4.5 provides the heuristic solution. Section 4.6 provides performance proofs including analysis for the worst-case scenario. Experimental

results are provided in Section 4.7, where we discuss the application, the dataset (and its preprocessing phases), and the conducted experiments. A discussion of the results and the salient lessons learned are provided in Section 4.8. Finally, the paper is concluded in Section 4.9 by summarizing this work and identifying future directions of work.

4.2. Background

To understand the concepts of FL systems, Li et al. [68] provide a comprehensive study of FL systems. They categorize FL systems based on six features including machine learning model, communication architecture, data partition, privacy mechanism, motivation of federation, and scale of federation. Additionally, the authors present a summary of a comparison that includes 42 studies based on the six proposed features. Researchers [67] and [68] categorize FL based on data distribution as:

- *Horizontal Federated Learning*: datasets of clients share the same feature space but with a small intersection in regards to the sample space.
- *Vertical Federated Learning*: datasets of clients share the same sample space but with a small intersection in regards to the feature space.
- *Hybrid Federated Learning* (*Federated Transfer Learning*): datasets of clients have a small intersection in regards to both the feature and sample spaces.

The main challenges in implementing FL, as described in [73] and [76], are:

- *Communication cost*: there could be many clients (millions) and the system may execute many rounds before converges to the required level of accuracy, which imposes an overload on the network.

- *Clients heterogeneity*: the system is heterogeneous and has clients with varying computation, storage, and communication capabilities. Also, the client datasets may differ in features and samples (i.e., the datasets may have statistical heterogeneity).

- *Privacy and security*: FL already protects clients' data by only sharing models' parameters. However, sensitive information may be revealed.

Researchers [76] and [77] have highlighted the importance of client selection for enhancing the performance of FL systems since it contributes to both communication cost and resource allocation.

Existing research on enhancing performance in FL follow one of the following approaches:

- *Algorithm Optimization*: optimize the FL algorithm and perform more computation on clients to reduce the convergence time by reducing the number of rounds on the expense of more computation [78]–[85].

- *Selective Updates*: select only important updates from the clients or select the best clients in regards to the clients' resources and data size [86]–[91].

- *Model Compression*: reduce the amount of data exchanged between clients and the server [79], [92]–[94].

4.3. Related Work

FL is a hot research area that grabs the attention of many researchers. In this section, we list different approaches for enhancing the performance and discuss studies in each approach. Next, we discuss studies related to online resource selection based on the optimal stopping theory and particularly the secretary problem.

4.3.1. Algorithm Optimization

Some researchers work on optimizing the algorithm used in FL to reduce convergence time and thus reduce the generated traffic on the network. Replacing the minibatch Stochastic Gradient Descent (mb-SGD) optimization model with Adam has been studied in [79]. The authors propose CE-FedAvg, an algorithm that uses Adam optimization and compresses models before uploading to the server. The authors claim that using Adam optimization along with model compression reduces the convergence time by reducing the number of rounds and the amount of data exchanged between clients and the server. Using a multi-objective evolutionary algorithm with neural networks in FL has been studied in [80]. The authors use the Elitist Nondominated Sorting Genetic Algorithm (NSGA-II) to minimize the communication cost at the expense of higher computation cost. In [81], researchers propose Momentum Federated Learning (MFL), which uses Momentum Gradient Descent (MGD) in every step of local updates rather than the first-order gradient descent. Authors state that since MGD consider preceding iteration, it converges faster than the traditional FL system.

Other researchers proposed algorithms that utilize the computation power on clients' machines to speed up the convergence process. To reduce the number of rounds, Liu et al. [82] propose to use Federated Stochastic Block Coordinate Descent (FedBCD) algorithm in vertical FL, which let clients do multiple local model updates before syncing with each other. Authors in [83] claim that using two models in every client instead of a single model can reduce the number of rounds. Besides training the global model received from the server, each client trains another local model and uses the Maximum Mean Discrepancy (MMD) between the output of the two models. Using agents on edge nodes between clients and the server are studied in [84] and [85]. Multiple agents perform partial model aggregation before communicating with the server to reduce the communication cost between clients and the

server.

Other researchers study the trade-off between the number of iterations performed by clients to minimize the loss function and the frequency of global aggregation done by the server. In [78], the authors compute the convergence bound of the gradient-descent algorithm then designed an algorithm that finds the best frequency of global aggregation based on system dynamics, model characteristics, and data distribution to minimize the consumed computation and communications.

4.3.2. Selective Updates

In [86], the authors formulate a client selection and resource allocation optimization problem for FL in wireless networks to minimize the value of the loss function. They first derive an equation to represent the expected convergence rate of the FL algorithm. Next, they simplified the optimization problem as a mixed-integer nonlinear programming problem. Then for a given uplink resource block allocation and client selection, authors compute the optimal transmit power. Finally, they transform the problem into a bipartite matching problem and use the Hungarian algorithm to find the optimal client selection and resource block allocation. Nishio and Yonetani [87] propose a new FL protocol named FedCS to enhance the efficiency of FL. The basic idea of the proposed protocol is to select clients based on their computation/communication capabilities and their data size instead of picking clients randomly. To reduce the communication overload, authors in [88] propose an approach that identifies clients with irrelevant updates and prevent those clients from uploading their updates to the server. In [89] and [90], the authors proposed the selection of clients based on the consumed energy in model's transmission and training, clients' distance from the server, and channel availability using Deep Reinforcement Learning (DRL) approach. Yoshida et al. [91] propose a hybrid FL approach based on the assumption that

some clients share and upload their data to the server to improve the accuracy and mitigate the degradation resulted from non-independent-and-identically-distributed (non-IID) data. However, uploading clients' data to the server violates the rules of FL.

4.3.3. Model Compression

Sattler et al. [92] proposed a new compression framework named Sparse Ternary Compression (STC). Authors claim that their compression framework performs better than other proposed methods in the literature in bandwidth-constrained learning environment. In [93], the authors propose to use structured updates (low rank and random mask) and force models to use these structures and also sketched updates with lossy compression before sending models to the server. On the other hand, Caldas et al. [94] apply lossy compression on the model sent from the server to the clients.

The related research discussed up till now have a high computational cost. The clients' intensive computation and algorithm optimization approach requires intensive computation. In addition to the extra computation required by the compression approach, it is best applied to models with large parameter vector such images or models with many hidden layers. The presented studies using the selective updates approach either are too difficult to train (especially when a large number of clients are used as in DRL) or have no analysis and proofs for convergence. In contrast, our proposed algorithm, which uses the selective updates approach, does not require intensive computations or a large parameter vector, and we also provide analysis and proofs demonstrating convergence.

4.3.4. Secretary Problem

The secretary problem, which is also known as the marriage problem, dowry problem, beauty contest problem, or Googol is a class of the optimal stopping decision problems. The secretary problem was first introduced by Martin Gardner back in 1960 [95]. The classical secretary problem focuses on the selection of a secretary from a pool of candidates adhering to the following rules [95] [96]:

- The number N of candidates is known,
- Only one candidate is to be chosen,
- Candidates are interviewed sequentially in random order,
- Each candidate must be accepted or rejected before interviewing the next one (with no provision for recalling rejected candidates later),
- Candidates are ranked from best to worst and the decision of accepting or rejecting a candidate depends on the relative ranks of candidates interviewed so far,
- The problem focuses on maximizing the probability of selecting the best candidate.

The solution of the secretary problem is for some integer $1 \leq \alpha < N$, reject the first α candidates then select the first candidate with rank better than of those observed candidates. The goal is to find the optimal α that maximizes the probability of selecting the best candidate. Actually, it has been proven that the optimal value for α is 0.367879 with optimal probability of $\frac{1}{e}$ [95]. In other words, the probability of finding the best candidate is 37% when rejecting the first 37% of candidates and selecting the first candidate with ranking better than those observed ones.

The authors in [97] reviewed the extensions and generalizations of the secretary problem. They indicate that some researchers focus on the secretary problem when the number of candidates is unknown. Other researchers assume that candidates' ranks follow a specific distribution such as Poisson. Additionally, they show that some studies focus on selecting R candidate instead of one.

In this paper, we are interested in studies of the secretary problem where R top candidates are selected. In [98], the authors provide many variations of the secretary problem studied under different assumptions and one of these cases is for selecting R candidates with one of the candidates as the best candidate. Kleinberg proposed an algorithm to maximize the sum of ranks of the R selected candidates [99]. The algorithm has two stages. In the first stage, the classical secretary algorithm is recursively applied on roughly the first half of candidates to select $l = R/2$ best candidates. In the second stage, the rank of the lth selected candidate in the first stage is used as a threshold for selecting $R/2$ candidates from the second half of candidates. The author states that the algorithm has a competitive ratio of $1 - O(\sqrt{q/R})$. In [100], the authors propose an algorithm to maximize the sum of the R selected candidate. The algorithm rejects the first $\lfloor n/e \rfloor$ candidates and records the R highest rankings in set S. Next, when a candidate with a rank higher than the minimum rank in S is encountered, the candidate is selected and the minimum rank in S is removed. This is repeated until either S is empty or all candidates are reviewed. The authors indicate that the algorithm has a competitive ratio no worse than e for all values of R.

The work in this paper is inspired by the aforementioned studies. However, this work is different in that we find the optimal stopping position α, which we call α^*, to maximize the probability of selecting the R top candidates. We reject the first α^* candidates and record the best rank. Then, we use the best rank as a threshold in selecting the top R candidates.

4.4. System Model

We assume N candidate clients and one server. Also, we assume a budget of R candidate clients. The nature of the proposed model is online since some clients become available while others become unreachable or offline over time. Consequently, the server must make an irrevocable decision to accept (i.e. select) or reject a candidate client once a candidate client becomes available. The server runs the proposed heuristic (explained in the next section), which initializes the global model, selects the R best candidate clients based on their test accuracy, and then train the global model using the selected candidate clients in K communication rounds. Each selected candidate client trains the local model in E epochs using the local dataset but with the global model parameters. Moreover, we assume the datasets of candidate clients are different in size. Therefore, we use the terms *fat clients* and *thin clients* to point to candidate clients with different sizes of datasets. We note that in some literature, the terminology of elephants (instead of fat) and mice (instead of thin) is used instead [101]. For the convenience of the readers, we have listed the main mathematical notations used in this paper in Table 4.1.

4.5. Proposed Client Selection Solution

The problem we tackle in this paper is to select the best set of candidate clients that provide higher test accuracy when training the global model using their local dataset. This problem is similar to the famous *secretary problem*, which aims to maximize the probability of selecting the maximum element from a randomly ordered sequence [102]. However, instead of selecting one element, in this problem, R elements must be selected. The secretary problem is one scenario of the *optimal stopping theory*. In the secretary problem, an employer wants to hire a secretary and there are N candidates. The employer cannot assess the quality

Table 4.1: Summary of mathematical notations

Notation	***Definition***
N	Total number of candidate clients arriving until time T. Each candidate client is identified by an index in the interval $1..N$
K	Number of communication rounds
E	Number of epochs
R	Number of required best candidate clients
$(\mathcal{C}_\ell)_{1\leq\ell\leq N}$	Set of candidate clients
α	An index in the interval $1..N$
α^*	The optimal value of α
$\mathcal{C}_M$	The best candidate client in $[1..\alpha]$, i.e., $1 \leq M \leq \alpha$
$(i_m)_{1\leq m\leq R}$	The set of R positions corresponding to the top R best candidate clients in the interval $[\alpha+1..N]$, better than $\mathcal{C}_M$, such that $\alpha+1 \leq i_m \leq N$ and $\mathcal{C}_{i_m}$ is worst than $\mathcal{C}_{i_{m-1}}$ for all $2 \leq m \leq R$
$\left\{\mathcal{A}^{(i)}\right\}_{\alpha+1\leq i\leq N}$	The set of events where "the ith candidate better than $\mathcal{C}_M$ is selected"
$\left\{\mathcal{B}^{(m,i)}\right\}_{1\leq m;\alpha+1\leq i\leq N}$	The set of events where "the ith candidate is the mth best one"
$\mathcal{E}_R$	The event "The R best candidates in $[\alpha+1..N]$ better than $\mathcal{C}_M$ are selected" occurring with the probability $\Pr(\mathcal{E}_R)$; Obviously, $\bigcap_{\ell=1}^{R} \mathcal{E}_R = \bigcup_{i_R\leq i_{R-1}..\leq i_1} \bigcap_{\ell=1}^{R} \left\{\mathcal{A}^{(i_\ell)} \cap \mathcal{B}_R^{(\ell,i_\ell)}\right\}$ with i_1 and i_R correspond to the best and the worst combinations, among R selected ones, respectively
$\mathcal{P}^{(r_1,r_2)}$	Probability to select the R best candidates $(i_m)_{1\leq m\leq R}$ where $r_1 \leq R \leq r_2$ and $r_1, r_2 < N$. $\mathcal{P}^{(r_1,r_2)} = \sum_{R=r_1}^{r_2} \Pr(\mathcal{E}_R)$.

of a candidate until after the end of the interview and have to make an irrevocable hiring decision. Thus, the employer may end up hiring a candidate before interviewing the rest of the candidates and the hiring of the best candidate is not guaranteed.

Our solution is inspired by the *secretary problem*. The quality of a client is determined by its test accuracy. We evaluate the test accuracy (i.e. quality) of the first α^* (see section 4.6) clients and reject them all. Then, select the next R clients with test accuracy better

than the best test accuracy of the first α^* clients, and if none is found then select the last clients.

4.5.1. Proposed Algorithm

The proposed heuristic identifies the best test accuracy among the first few available candidate clients then use this test accuracy as a threshold for accepting or rejecting candidate clients available later. The heuristic accepts the parameters N, R, r_1, r_2, K, E, and δ as explained in Algorithm 2 and consists of *three stages* that run every δ time units to update the global model.

In the *first stage* (Algorithm 2, lines 2 through 11), the value of α^* (discussed in Section 4.6) is computed based on the value of r_1 and r_2 using equation (4.7). The first α^* candidate clients that are available are then tested to determine the best test accuracy. However, none of those candidate clients are accepted. Whenever a candidate client becomes available, the server initializes the global model and communicates with the candidate client to evaluate its test accuracy. Testing is performed by sending the initialized global model's parameters from the server to the candidate client for one communication round so that the candidate client trains the local model with these parameters using the local dataset. Then, the candidate client sends back the updated parameters to the server. The server evaluates the received parameters (i.e., no averaging is applied since only one candidate client is involved) using the test dataset to determine the test accuracy of the candidate client. After testing α^* candidate clients, the server selects the best test accuracy to be used as a threshold in the second section.

In the *second stage* (Algorithm 2, lines 12 through 27), whenever a candidate client becomes available, it gets tested in the same way explained in the first section. Next, the server accepts (i.e., selects) the candidate client only if its test accuracy is greater than

Algorithm 3 Proposed heuristic

Input: N (expected number of clients), R (number of selected candidate clients), r_1, r_2 (to compute α^*), K (number of communication rounds), E (number of epochs per client), and δ
Output: Trained global model

for every δ time units **do**
  // Find best test accuracy for first m ***candidate clients,*** $m : 1..\alpha^*$
  Initialize the global model
  Compute α^* based on r_1, r_2, and N using equation (4.7)
  Set A_b, best test accuracy = 0
  for $m = 1$ to α^* **do**
    Test client $\mathcal{C}_{\mathcal{M}}$ and record A_m, its test accuracy
    if $A_m > A_b$ **then**
      Set $A_b = A_m$
    end if
    Reject candidate client C_m
  end for
  // Find R ***best candidate clients***
  Set S_b, set of best candidate clients = []
  Set N_b, number of best candidate clients found = 0
  for $m = \alpha^* + 1$ to N **do**
    if $N_b = R$ **then**
      Reject candidate client C_m
    else if $(N - m) \leq (R - N_b)$ **then**
      Accept candidate client C_m and add it to S_b
      Increment N_b by 1
    else
      Test client C_m and record A_m, its test accuracy
      if $A_m > A_b$ **then**
        Accept candidate client C_m and add it to S_b
        Increment N_b by 1
      end if
    end if
  end for
  // Start training
  for $k = 1$ to K **do**
    Send global model to all candidate clients in S_b
    Candidate clients train global model on local dataset for E epochs
    Server average aggregated model parameters from candidate clients in S_b
  end for
end for

the best test accuracy found in the first section. Nonetheless, if the number of available candidate clients is less than the number of required candidate clients (i.e., R) then the server has no choice but to select those remaining candidate clients. In the worst-case

scenario, the candidate client with the best test accuracy is met during the first section. Consequently, all candidate clients met early in the second stage are rejected for having a test accuracy less than that of the best test accuracy found in the first section. As a result, the server is forced to accept all candidate clients that are met at the end of the second section. In fact, in the worst-case scenario, the proposed heuristic behaves similarly to the random algorithm explained in Section 4.8.

In the *third stage* (Algorithm 2, lines 28 through 32), once the best candidate clients are identified, the global model is trained using the selected candidate clients for K communication rounds as described in Section 4.1.

4.5.2. Toy Illustrative Example

To understand the proposed algorithm in more depth, we present an example where we observe one run cycle (when δ is 1) of the proposed algorithm overtime (see Figure 4.1). Assume a total of 10 candidate clients becoming available over time during the observed period (i.e. $N = 10$). We refer to candidate i as $\mathcal{C}_i$. Additionally, we set the budget to 2 candidate clients (i.e. $R = 2$), which means that we want to select the best 2 candidate clients for training the global model. Moreover, we set E, the number of epochs, to 3 and set K, the number of communication rounds between the server, and selected candidate clients for training the global model to 20. Also, the value of α^* is computed based on equation (4.7) in section 4.6.1 (assuming r_1 is 1 and r_2 is 2) and its value is 2.

When a candidate client becomes available then (1) the proposed algorithm initiates the global model's parameters then sends them to the candidate client, (2) The candidate client trains the local model for E epochs using the received parameters from the server on the local dataset, (3) the candidate client sends the updated parameters to the server, (4) the server evaluate the accuracy of the candidate client by testing the global model using the updated

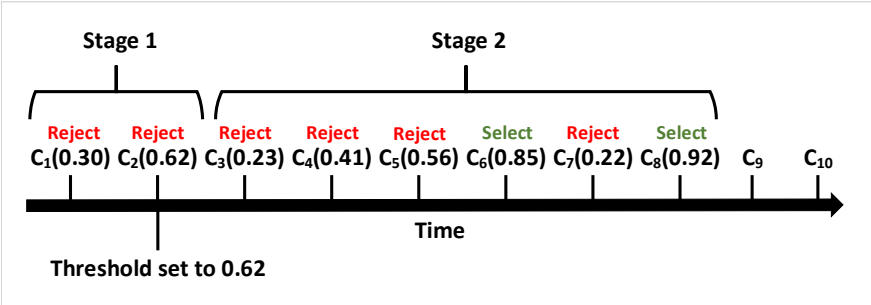


Figure 4.1: An illustrative example of the proposed algorithm.

parameters on the test dataset. Then, the proposed algorithm must make an irrevocable decision on whether to use this client or not based on its evaluated test accuracy.

The proposed algorithm runs in three stages. In the first stage, the proposed algorithm communicates with the first α^* (i.e. 2) candidate clients and evaluate their test accuracy to determine the best test accuracy, which is used as a selection threshold with the rest of candidate clients that become available later. Thus, when $\mathcal{C}_1$ becomes available, the proposed algorithm communicates with $\mathcal{C}_1$ then evaluates its test accuracy and finds it 0.30. The proposed algorithm sets its selection threshold to 0.30 and rejects $\mathcal{C}_1$. Next, $\mathcal{C}_2$ becomes available and the proposed algorithm communicates with $\mathcal{C}_2$ then evaluates its test accuracy and finds it 0.62. The proposed algorithm updates its selection threshold to 0.62 as illustrated in Fig. 4.1 where $\mathcal{C}_i(x)$ represents candidate client i with evaluated test accuracy x (test accuracy is a number between 0 and 1, where 0 means the trained model fails to identify all test samples while 1 means the trained model identifies all test samples successfully).

In the second stage, the proposed algorithm will continue to communicate with any candidate client that becomes available and evaluate its test accuracy to decide on the selection of this candidate client. This process continues as shown in Fig. 4.1 until the proposed algorithm selects 2 candidate clients and as follows:

- $\mathcal{C}_3$ becomes available and its test accuracy is 0.23 and thus gets rejected.

- $\mathcal{C}_4$ becomes available and its test accuracy is 0.41 and thus gets rejected.
- $\mathcal{C}_5$ becomes available and its test accuracy is 0.56 and thus gets rejected.
- $\mathcal{C}_6$ becomes available and its test accuracy is 0.85 and thus gets selected.
- $\mathcal{C}_7$ becomes available and its test accuracy is 0.2 and thus gets rejected.
- $\mathcal{C}_8$ becomes available and its test accuracy is 0.92 and thus gets selected.
- The server is not going to communicate with $\mathcal{C}_9$ and $\mathcal{C}_{10}$ once they are available since the proposed algorithm has already selected two candidate clients.

In the third stage, the proposed algorithm trains the global model using $\mathcal{C}_6$ and $\mathcal{C}_8$ with K communication rounds but without initiating the global model in every round. A best-case scenario is presented in this example, but a worst-case scenario can occur if the test accuracy of $\mathcal{C}_2$ is evaluated and found as 9. In this case, the proposed algorithm rejects both $\mathcal{C}_6$ and $\mathcal{C}_8$. Eventually, the proposed algorithm will have to communicate with the last two clients ($\mathcal{C}_7$ and $\mathcal{C}_{10}$) and selects both.

4.6. Performance Analysis

The performance of the proposed algorithm explained in the previous section depends vitally on the optimal value of α, which is α^*. In this section, we derive an equation for computing the value of α^* and prove its validity. This equation is plugged in the first stage of the proposed algorithm as mentioned in section 4.5.1. Finally, we analytically analyze the performance of the proposed algorithm in worst-case scenario.

4.6.1. Optimal Value for α

By assuming (i) M and i_m positions are not known in advance, (ii) the candidates can arrive in any order, and (iii) $N, \alpha >> R$, we aim to find the optimum value α^*, depending on both, allowing to maximize $\mathcal{P}^{(r_1,r_2)}$.

Lemma 1. *The following summation*

$$\mathcal{K}(R,\alpha) = \sum_{i_R=\alpha+1}^{N-R+1} \frac{1}{i_R-1} \sum_{i_{R-1}=i_R+1}^{N-R+2} \frac{1}{i_{R-1}-1} .. \sum_{i_1=i_2+1}^{N} \frac{1}{i_1-1}, \tag{4.1}$$

can be tightly approximated by

$$\mathcal{K}(R,\alpha) \approx \frac{\left(\log \frac{N}{\alpha}\right)^R}{R!}. \tag{4.2}$$

Proof. Let us proceed by induction. one can ascertain that for $R = 1$, the summation $\sum_{i_1=\alpha+1}^{N} \frac{1}{i_1-1}$ can be approximated by the $\int_\alpha^N \frac{dt}{t} = \log \frac{N}{\alpha}$, confirming (4.2).

Let us assume that (4.2) holds for $R-1$. One obtains

$$\begin{aligned}
\mathcal{K}(R,\alpha) &= \sum_{i_R=\alpha+1}^{N-R+1} \frac{1}{i_R-1} \mathcal{K}(R-1, i_R) \\
&\approx \frac{1}{(R-1)!} \sum_{i_R=\alpha+1}^{N-R+1} \frac{\left(\log \frac{N}{i_R}\right)^{R-1}}{i_R-1} \\
&\approx \frac{1}{(R-1)!} \int_\alpha^{N-R+1} \frac{\left(\log \frac{N}{t}\right)^{R-1}}{t} dt.
\end{aligned} \tag{4.3}$$

Finally, taking into account that $R << N$ (i.e., $N-R+1 \approx N$), it follows that

$$\mathcal{K}(R,\alpha) \approx \frac{1}{R!} \left[-\left(\log \frac{N}{t}\right)^R \right]_\alpha^N \tag{4.4}$$

$$\approx \frac{1}{R!} \left(\log \frac{N}{\alpha}\right)^R, \tag{4.5}$$

which concludes the proof. □

Proposition 1. *For all positive numbers $r_1, r_2 << \alpha, N$, the approximation*

$$\mathcal{P}^{(r_1,r_2)} \approx \frac{\alpha}{N} \sum_{R=r_1}^{r_2} \frac{1}{R!} \left(\log \frac{N}{\alpha}\right)^R, \tag{4.6}$$

holds, and the optimum value maximizing such probability is

$$\alpha^* = N \exp\left(-\left(\frac{r_2!}{(r_1-1)!}\right)^{\frac{1}{r_2-r_1+1}}\right). \tag{4.7}$$

Proof. Given that the indices of the selected candidates are sorted in increasing order of candidates' accuracies, $\mathcal{E}_R$ can be broken into R exclusives events as follows

- Candidate client M is the best one in $[1, i_R - 1]$ **and**
- Candidate clients i_m are the best ones in $[1, i_{m-1} - 1]$, $2 \leq m \leq R$ **and**
- Candidate client i_1 is best one in $[1, N]$.

Consequently,

$$\Pr\left(\mathcal{E}_R\right) = \sum_{i_R=\alpha+1}^{N-R+1} \sum_{i_{R-1}=i_R+1}^{N-R+2} .. \sum_{i_1=i_2+1}^{N} \Pr\left(\underbrace{\bigcap_{\ell=1}^{R} \left\{\mathcal{A}^{(i_\ell)} \cap \mathcal{B}_R^{(\ell,i_\ell)}\right\}}_{\mathcal{D}_R}\right). \tag{4.8}$$

With the aid of the Bayes's rule, $\mathcal{D}_R$ can be rewritten as

$$\Pr\left(\mathcal{D}_R\right) = \Pr\left(\mathcal{A}^{(i_R)} \middle| \mathcal{B}_R^{(R,i_R)} \cap \mathcal{D}_{R-1}\right) \Pr\left(\mathcal{B}_R^{(R,i_R)} \middle| \mathcal{D}_{R-1}\right). \tag{4.9}$$

The probability to select the Rth best one among $[1..N] \setminus \{i_1, i_2, .., i_{R-1}\}$ is

$$\Pr\left(\mathcal{B}_R^{(R,i_R)} \middle| \mathcal{D}_{R-1}\right) = \frac{1}{N-R+1}, \tag{4.10}$$

with the conditional probability in (4.9) can be evaluated as

$$\Pr\left(\mathcal{A}^{(i_R)}\middle|\,\mathcal{B}_R^{(R,i_R)}\cap\mathcal{D}_{R-1}\right)=\frac{\alpha}{i_R-1}\prod_{\ell=1}^{R-1}\frac{1}{i_\ell-1}. \tag{4.11}$$

Substituting (4.11), (4.10), and (4.9) into (4.8), one obtains

$$\Pr\left(\mathcal{E}_R\right)=\frac{\alpha}{N-R+1}\mathcal{K}(R,\alpha). \tag{4.12}$$

Leveraging **Lemma 1** and noting that $N-R+1\approx N$, (4.6) is obtained. Now, defining $x=\alpha/N$ (i.e., $0\leq x\leq 1$), the two first derivatives of $\mathcal{P}^{(r_1,r_2)}$ with respect to x can be expressed as

$$\frac{\partial\mathcal{P}^{(r_1,r_2)}}{\partial x}=\frac{(-\log x)^{r_2}}{r_2!}-\frac{(-\log x)^{r_1-1}}{(r_1-1)!}, \tag{4.13}$$

$$\frac{\partial^2 P^{(r_1,r_2)}}{\partial x^2}=-\frac{1}{x}\left[\frac{(-\log x)^{r_2-1}}{(r_2-1)!}-\frac{(-\log x)^{r_1-2}}{(r_1-2)!}\right]. \tag{4.14}$$

Thus, by solving $\frac{\partial\mathcal{P}^{(r_1,r_2)}}{\partial x}=0$ and setting $\alpha^*=Nx^*$, we get (4.7). Moreover, it can be easily checked that the second derivative evaluated at x^*

$$\begin{aligned}\frac{\partial^2 P^{(r_1,r_2)}}{\partial x^2}&=\frac{(-\log x)^{r_1-2}}{x^*\,(r_1-2)!}\left[\frac{(r_1-2)!}{(r_2-1)!}\underbrace{(-\log x^*)^{r_2-r_1+1}}_{=\frac{r_2!}{(r_1-1)!}}-1\right]\\&=-\left(r_2-r_1+1\right)\frac{(-\log x)^{r_1-1}}{x^*\,(r_1-1)!},\end{aligned} \tag{4.15}$$

is negative as $r_2>r_1$ and $x^*\leq 1$, which completes the proof. □

Table 4.2 summarizes some values of the optimal number α^* along with the aforementioned maximum probability for various values of r_1 and r_2, when $N=1000$. Note that the probability (4.6) is an increasing function on r_2, while its maximum value is not monotone as it depends also on r_1 as summarized in Table I. It can be seen also that:

- The smaller r_1 is, the greater the optimal value ($\alpha^* = Nx^*$).
- For a fixed r_1, the larger r_2 is, the smaller α^*.

Fig. 4.2 shows that the probability of selecting the best R clients is higher when the value of α is small.

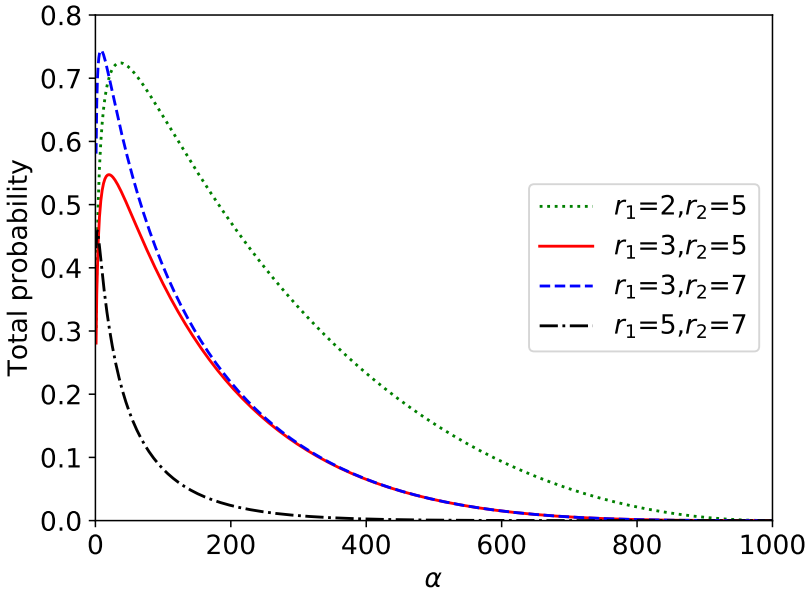


Figure 4.2: Effects of α on the probability of selecting the best clients.

Table 4.2: Choosing α^* that maximizes the probability to select R best candidates such that $r_1 \leq R \leq r_2$ and $N = 1000$

r_1	r_2	α^*	Percentage (%) $x^* = \frac{\alpha^*}{N}$	$\mathcal{P}_{\max}^{(r_1,r_2)}$
2	2	135.3353	13.53	0.2707
2	3	49.7871	4.97	0.4481
2	4	0.3355	0.03	0.0966
3	3	49.7871	4.97	0.2240
3	4	2.4788	0.24	0.2231

4.6.2. Worst-Case Analysis (Competitive Ratio Analysis)

The worst-case scenario is encountered when the proposed heuristic does not find candidates that exceed $\mathcal{C}_{\mathcal{M}}$ from index α^* until N. The competitive ratio in the worst-case scenario is computed over all possible input sequences as the maximum ratio of the gain of the online algorithm and the optimal offline algorithm [103].

Proposition 2. *The heuristic's worst-case performance has a competitive ratio of* $O(1)$ *when* R *is proportional to* N*.*

Proof. Let ALG be the proposed heuristic and OPT be the optimal algorithm.

The worst-case happens when the highest element appears before index α^*. In that case, the proposed algorithm randomly selects candidate clients from index $\alpha^* + 1$ until N. A candidate client within this range of indices is selected with probability $\frac{R}{N-\alpha^*}$. Consequently, the following proof is concluded as follows:

$$Com_r = \frac{ALG}{OPT} = \frac{R}{N - \alpha^*}$$

Thus, Com_r, the competitive ratio, becomes $\mathcal{O}(1)$ when R is proportional to N. □

4.7. Experimental Settings

In this section, we describe the application proposed in this paper in detail first. Next, we describe the dataset used in the simulation and describe the dataset preparation phases used to transform the raw dataset into N candidate clients' datasets. Finally, we discuss conduced experiments.

4.7.1. Use Case: IoT Device Classification

IoT devices perform specific tasks, which makes their network behavior predictable [104]. There are plenty of studies on IoT device classification or fingerprinting in the literature [75], [104]–[112]. Those studies concentrate on identifying IoT devices for different reasons including security, access control, provisioning, resource allocation, and management [105]. Actually, most of those studies concentrate on security in response to recent incidents. In one incident, thousands of IoT devices including surveillance cameras are used for Distributed Denial of Service (DDoS) attack [75]. Therefore, we propose a client alarm application based on IoT device type classification in FL settings to identify unauthorized IoT devices. The IoT device type classification is inspired by the work in [112]. We aim to use the proposed application as a use-case to test the performance of the proposed heuristic.

The proposed application consists of N candidate clients, a main server in the cloud, and an alarm mechanism. Each client's environment has several IoT devices, a local machine (i.e., the local server), and an alarm device as shown in Fig. 4.3. The alarm can be a physical device or software that delivers email, text messages, or any other form of notification to the client. The local server monitors the traffic generated by IoT devices, extract features, and build a local dataset. Then, train the local model using the local dataset. However, training on local dataset is not sufficient to identify unknown IoT devices in the environment. As a result, the clients subscribe to the alarm service provided by the server through the use of FL. The server is responsible for running the proposed FL algorithm. Also, the server and clients cooperate to build a global model capable of classifying devices used by participating clients. In other words, clients can use the global model to identify unknown devices from the knowledge of other clients.

The proposed algorithm in the server trains a global model by sharing only the model's

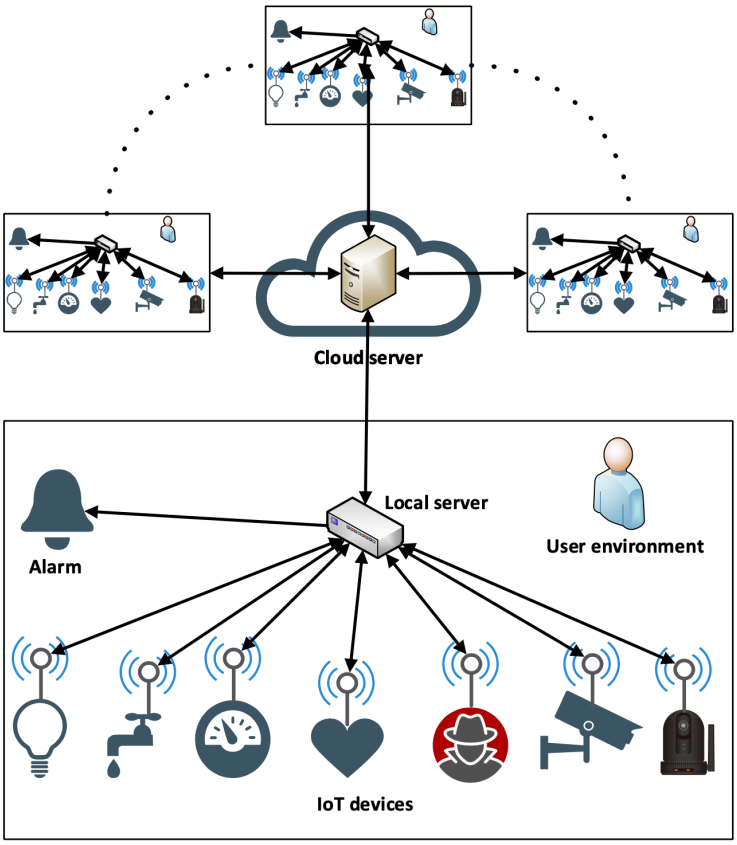


Figure 4.3: An illustration of the clients alarm application. The cloud server running the proposed algorithm communicates with the local servers of the best subscribed clients to train the global model.

parameters with clients and thus preserving the security and privacy of clients. The process of training the global model is repeated every δ time units to make sure that the new clients, and clients with the new installed IoT devices, are considered and included.

Employing all clients in the training process produce high traffic, which overloads the network. Additionally, this might be infeasible since some clients are not available all the time. However, selecting clients with high accuracy contribution to the global model training enhance the classification accuracy, which is done by the proposed heuristic.

4.7.2. Dataset Details and Preprocessing Phases

To test the performance of the proposed heuristic, we use a real dataset collected by researchers from the University of New South Wales (UNSW), Sydney, Australia [112]. The dataset is created using 28 IoT devices and also some non-IoT devices installed in a lab on the campus of the university. Trace data are captured over 6 months between October 1, 2016 and April 13, 2017. However, only 20 days of trace data are available for the public. Raw data consisting of packet headers and payload information are captured using the `tcpdump` tool installed on the gateway. The dataset is available as a set of pcap (packet capture) files and also as a set of CSV (comma-separated values) files. The dataset consists of 20 pcap files, one file per day.

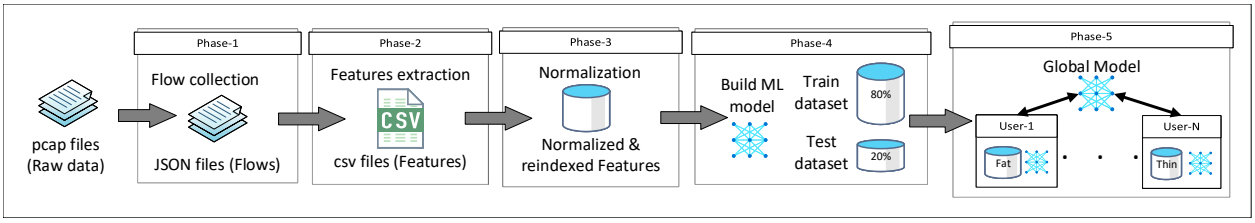


Figure 4.4: Dataset preprocessing phases (through which the raw dataset is transformed to the N candidate clients' datasets).

The raw dataset is processed in five phases (illustrated in Fig 4.4) in order to create N candidate client's datasets to simulate FL settings as described next.

In the ***flows collection phase*** (**Phase-1**), we collect flows from raw data in pcap files using the `joy` tool developed by Cisco Systems [113]. `Joy` is a data collection tool that reads the data from raw traffic (or from pcap files) and produces a JavaScript Object Notation (JSON) file with a summary of the traffic data in the form of flows. We create a bash script that uses the `joy` tool to process the pcap files and produce JSON files. Each JSON file contains flows related to a specific IoT device based on the MAC address listed in Table 4.3, which includes names of devices and their MAC addresses as indicated in

Table 4.3: Names and MAC addresses of the used IoT devices

IoT device name	***MAC address***
Amazon Echo	44:65:0d:56:cc:d3
August Doorbell Cam	e0:76:d0:3f:00:ae
Awair air quality monitor	70:88:6b:10:0f:c6
Belkin Camera	b4:75:0e:ec:e5:a9
Belkin Motion Sensor	ec:1a:59:83:28:11
Belkin Switch	ec:1a:59:79:f4:89
Blipcare BP Meter	74:6a:89:00:2e:25
Canary Camera	7c:70:bc:5d:5e:dc
Dropcam	30:8c:fb:2f:e4:b2
Google Chromecast	6c:ad:f8:5e:e4:61
Hello Barbie	28:c2:dd:ff:a5:2d
HP Printer	70:5a:0f:e4:9b:c0
iHome PowerPlug	74:c6:3b:29:d7:1d
LiFX Bulb	d0:73:d5:01:83:08
NEST Smoke Sensor	18:b4:30:25:be:e4
Netatmo Camera	70:ee:50:18:34:43
Netatmo Weather station	70:ee:50:03:b8:ac
Phillip Hue Lightbulb	00:17:88:2b:9a:25
Pixstart photo frame	e0:76:d0:33:bb:85
Ring Door Bell	88:4a:ea:31:66:9d
Samsung Smart Cam	00:16:6c:ab:6b:88
Smart Things	d0:52:a8:00:67:5e
TP-Link Camera	f4:f2:6d:93:51:f1
TP-Link Plug	50:c7:bf:00:56:39
Triby Speaker	18:b7:9e:02:20:44
Withings Baby Monitor	00:24:e4:10:ee:4c
Withings Scale	00:24:e4:1b:6f:96
Withings Sleep Sensor	00:24:e4:20:28:c6

the dataset's website [112]. To filter by MAC address, we use the Berkeley/BSD Packet Filter syntax supported by the `joy` tool through the data feature options. Each flow in the resultant JSON file has a flow key that includes the source and destination addresses, and the source and destination port and protocol numbers. Each flow also contains number of bytes, number of packets, start time, and end time. Additionally, `joy` can be configured to save more information per flow. Algorithm 4 describes the flow collection process. Also, the script is available on GitHub [114].

In the ***features extraction phase*** (**Phase-2**), we extract features from the flows stored in JSON files. Inspired by a previous study [112], we analyze the flows and extract features as listed in Table 4.4. Features are saved in a CSV file with the first 10 columns for features and the last column for the labels, which are the IoT device IDs. Algorithm 5 shows the

Algorithm 4 Flows collection algorithm

Input: dataset pcap files.
Output: JSON files.
1: **for** each pcap file as $pFileName$ **do**
2: Open $pFileName$ for reading
3: Set $deviceCo = 1$
4: Set $json = pFileName + deviceCo$
5: **for** each MAC address in Table 4.3 as mac **do**
6: Run `joy` with $pFileName$ as input, $json$ as output, and mac as the host MAC address
7: Set $deviceCo = deviceCo + 1$
8: **end for**
9: Close $pFileName$
10: **end for**

Table 4.4: IoT device features

Feature	***Description***
$totalSleepTime$	Total time of no activity
$totalActiveTime$	Total time of activity
$totalFlowVolume$	Number of bytes (sent/received) by the IoT device
$flowRate$	Total flow volume divided by total active time
$avgPacketSize$	Number of bytes sent or received divided by no. of packets sent or received
$numberOfServers$	Number of servers Excluding DNS (53) and NTP (123)
$numberOfProtocols$	Number of protocols based on destination port number
$numberOfUniqueDNS$	Number of unique DNS requests
$DNSinterval$	Total time for using DNS
$NTPinterval$	Total time for using NTP

steps used in the extraction process. In addition, the Python code for extracting the features is made available on GitHub [114].

In the ***normalization*** *phase* (**Phase-3**), we first normalize all features by transforming features' values to be between 0 and 1 using the `MinMaxScaler` function from the `scikit-learn` library [115]. Second, to ensure that samples are distributed randomly, we

Algorithm 5 Features extraction algorithm

Input: JSON files.
Output: features csv file.

Open features file for writing
Set $maxPeriod = 10$ minutes
for each JSON file **do**
  Open JSON file for reading
  Read $deviceID$ from JSON file
  Set $totalSleepTime = 0$; $totalActiveTime = 0$
  Set $totalFlowVolume = 0$; $totalPackets = 0$
  Set $numberOfServers = 0$; $numberOfProtocols = 0$
  Set $numberOfUniqueDNS = 0$; $DNSinterval = 0$
  Set $NTPinterval = 0$; $lastFlowEndTime = 0$
  for each flow **do**
    Set #flow = flow number
    Set $flowTime = flowEndTime - flowStartTime$
    Set $totalActiveTime = totalActiveTime + flowTime$
    Set $totalFlowVolume = totalFlowVolume$ + number of bytes in the flow
    Set $totalPackets = totalPackets$ + number of packets in the flow
    if port in flow is not recorded before **then**
      Set $numberOfProtocols = numberOfProtocols + 1$
      Record port
    end if
    if port in flow = 53 **then**
      Set $DNSinterval = DNSinterval + flowTime$
      if DNS query in flow is not recorded before **then**
        Set $numberOfUniqueDNS = numberOfUniqueDNS + 1$
        Record DNS query
      end if
    else if port in flow = 123 **then**
      Set $NTPinterval = NTPinterval + flowTime$
    else
      if destination address in flow is not recorded before **then**
        Set $numberOfServers = numberOfServers + 1$
        Record destination address
      end if
    end if
    if #flow = 1 **then**
      Set $startTime = flowStartTime$
    else
      Set $totalSleepTime = totalSleepTime +$
        $(flowStartTime - lastFlowEndTime)$
      if $flowEndTime - startTime \geq maxPeriod$ **then**
        Set $flowRate = 0$
        if $totalActiveTime \geq 0$ **then**
          Set $flowRate = totalFlowVolume/totalActiveTime$
        end if

Set $avgPacketSize = 0$
if $totalPackets \geq 0$ **then**
Set $avgPacketSize$=$totalFlowVolume/totalPackets$
end if
Add a record to features file with features and $deviceID$
Reinitialize all features variables
end if
end if
Set $lastFlowEndTime = flowEndTime$
end for
Close JSON file
end for
Close features file

randomly re-index all normalized features in the dataset.

In the ***ML model design phase*** (**Phase-4**), we split the dataset into two parts: the training dataset (80% of the original) and the test dataset (20% of the original). To ensure a fair comparison between the proposed algorithm and other algorithms, the test dataset is stored on the server. We then design a Deep Neural Network (DNN)-based ML model with three layers, each having 25 neurons. The first two layers use the ReLu activation function, while the last layer uses a softmax activation function. Adam optimizer is utilized for optimization.

Finally, in the ***dataset splitting phase*** (**Phase-5**), we create N datasets to represent local datasets for the N candidate clients. Each of the N datasets is created randomly from the training dataset. To reflect a real scenario, we ensure that those datasets do not have the same size. The majority of candidate clients possess a small amount of the dataset while the minority of candidate clients possess large portions of the dataset. Consequently, the fat clients constitute 20% of candidate clients and each fat client has about 10% of the training dataset selected randomly. On the other hand, the thin clients constitute 80% of candidate clients and each thin client has about 1% of the training dataset randomly selected. All

these parameters along with other FL parameters are listed in Table 4.5.

Table 4.5: FL parameters

Parameter	***Value(s)***
Batch size	3
E (Epochs)	8
K (Communication rounds)	20
Test dataset	20% of the dataset
Train dataset	80% of the dataset
Number of fat clients	20% of N
Number of thin clients	80% of N
Fat client dataset	10% of the train dataset
Thin client dataset	1% of train dataset

Table 4.6: Simulation parameters

Sym.	***Parameter***	***Value(s)***
N	No. of candidate clients	(100, 200, 400, 800, and 1600)
R	No. of best candidate clients	(10, 20, 30, 40, and 50)
r_1	Minimum no. of best candidate clients	1
r_2	Maximum no. of best candidate clients	(1, 2, 3, 4, and 5)

4.7.3. Experiments

After Phase-5, N candidate clients' datasets are formed to simulate FL settings, which are utilized in the conducted experiments. To measure the performance of the proposed heuristic, we compare the results of two algorithms against the proposed heuristic. The two algorithms are the online random algorithm and the offline best algorithm. The online random algorithm selects and rejects candidate clients randomly. On the other hand, the offline best algorithm is an offline algorithm that can work with all candidate clients at the same time. In other words, the offline best algorithm does not have to wait for clients to be available over time and instead have the advantage of working with all candidate clients at the same time. The offline best algorithm creates a sorted list of all (i.e., N) candidate clients

based on accuracy and selects the top R candidate clients, which are mostly fat candidate clients.

We conduct 125 experiments to test the performance of the proposed heuristic. In all these experiments, we fixed the number of communication rounds to 20; the number of epochs per client to 8; and the batch size to 3 (as shown in Table 4.5). We are not interested in optimizing the aforementioned parameters since the goal of this paper is not to achieve the highest accuracy possible, but to investigate the ability of the proposed heuristic compared against the state-of-the-art algorithms. Thus, we vary the number of clients N, number of selected candidate clients R, and r_2 (used to compute the value of α^*) each with five different values as indicated in Table 4.6. Experimenting with different values of N, R, and r_2 is vital to truly test the abilities of the proposed heuristic.

The values in Table 4.6 are not selected arbitrarily. We test with values of N that vary from hundreds to thousands by doubling the numbers to see how this increase affects the performance. As for R, we test with different values in tens and noticed that raising R more is not interested since the performance of all algorithms converges as explained later. Setting r_1 to one is a must since we need to select the first best candidate client. Additionally, we noticed that raising the value of r_2 to more than 5 will results in a very low α^* especially when N is 100. In other words, setting r_2 to higher numbers will reduce the search size to zero candidate clients since α^* will be close to zero.

4.8. Result Discussion

Since we cannot present all values and figures of the 125 experiments. In each case, we fixed 2 of the variables (i.e., N, R, and r_2) and show the results for changing the third variable. Also, for each case, we present 3 figures: the first represents the general accuracy of the system; the second for the average accuracy of selected candidate clients, which is

used to indicate the contribution of individual candidate clients toward the general accuracy of the system; and finally, the percentage of the accepted Fat clients, which is useful for investigating if more Fat clients lead to higher accuracy.

4.8.1. Experimenting with Different Values of r_2

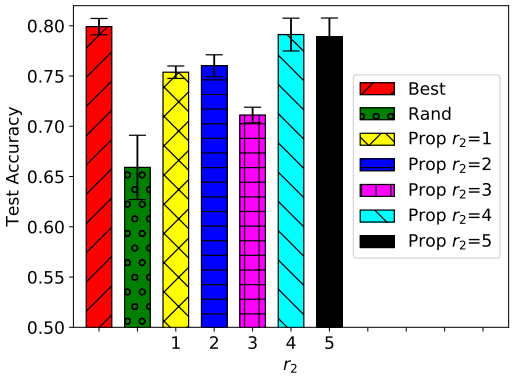
The results illustrated in Fig. 4.5 support the discussion in Section 4.6. Increasing the value of r_2 while fixing r_1 to 1 reduces the value of α^* and thus increases the probability of finding the best candidate clients as shown in Fig. 4.5a. Furthermore, Fig. 4.5b and Fig. 4.5b confirms the fact that increasing the value of r_2 leads to accepting more fat candidate clients with higher accuracy.

4.8.2. Experimenting with Different Values of R

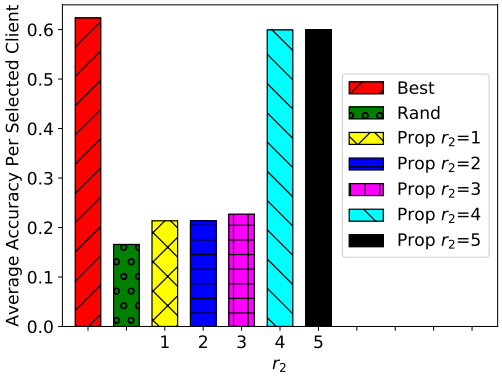
Fig. 4.6 shows that the proposed heuristic is more competitive when the number of selected candidate clients (i.e., R) is low. However, as R goes up, the accuracy of all algorithms converges as indicated in Fig. 4.6a. As the number of selected candidate clients increases, all algorithms will have a good portion of the dataset and will be able to converge to high accuracy in less time. As a result, there is no problem to solve for high values of R. Besides, sometimes it is not feasible to contact many candidate clients since some of them are not available. Figures 4.6b and 4.6c show that the accuracy of the system increases regardless of the accuracy of individual candidate clients and the number of fat nodes.

4.8.3. Experimenting with Different Values of N

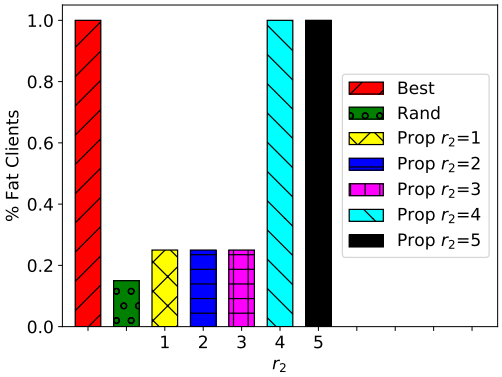
The performance of the proposed heuristic is almost stable when the total number of candidate clients is increased while fixing R to 30 as illustrated in Fig. 4.7. The accuracy



(a) Test Accuracy.

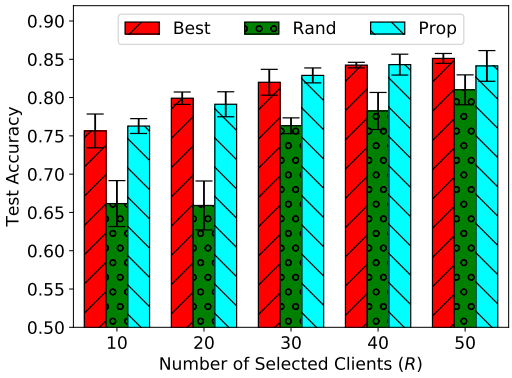


(b) Average Accuracy Per Selected Client.

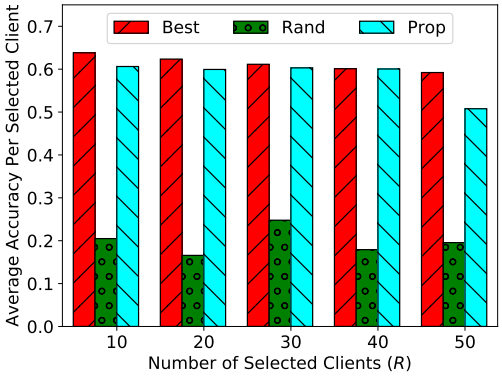


(c) % Fat Clients.

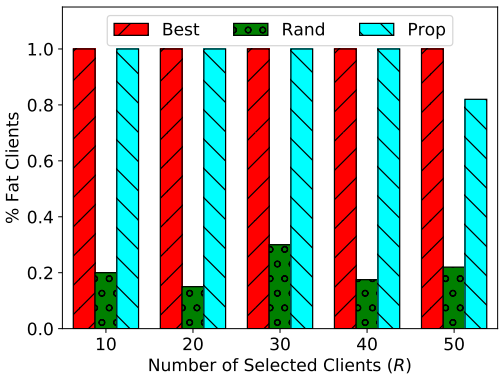
Figure 4.5: Performance of algorithms for different r_2 values (1, 2, 3, 4, 5) while fixing N, number of clients, to 400 and R, number of selected clients, to 20. *Our proposed algorithm performs better than the random algorithm approaching the performance of the best algorithm as r_2 is increased.*



(a) Test Accuracy.

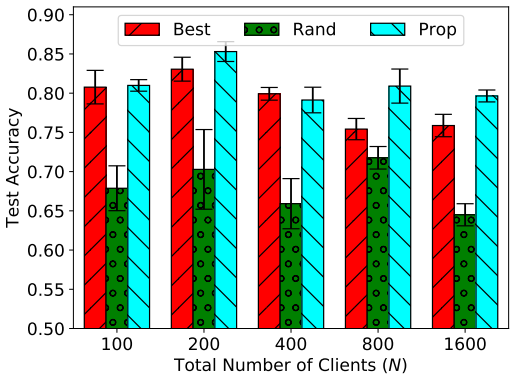


(b) Average Accuracy Per Selected Client.

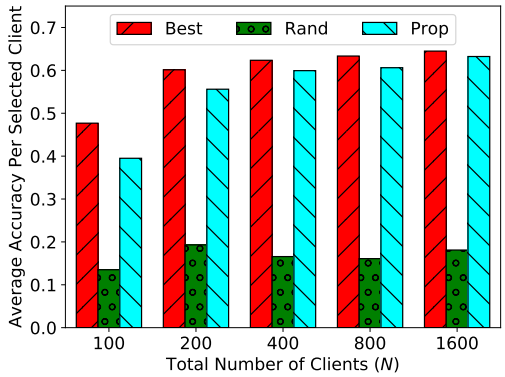


(c) % Fat Clients.

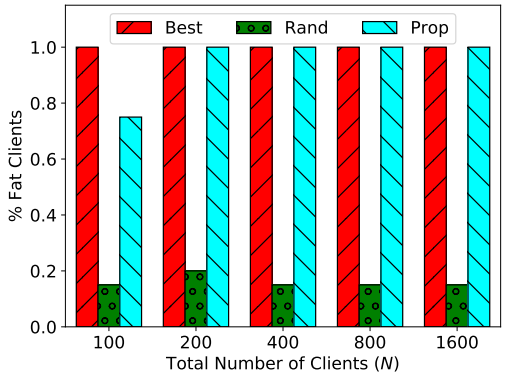
Figure 4.6: Performance of algorithms for different R, number of selected clients, values (10, 20, 30, 40, 50) while fixing N, number of clients, to 400 and r_2 to 4 (α^* is 43). *Our proposed algorithm is more competitive for smaller values of R and as R is increased, the performance of algorithms converges.*



(a) Test Accuracy.



(b) Average Accuracy Per Selected Client.



(c) % Fat Clients.

Figure 4.7: Performance of algorithms for different N, number of clients, values (100, 200, 400, 800, and 1600) while fixing R, number of selected clients, to 30 and r_2 to 4 (different α per N value). *Our proposed algorithm performs better than the random algorithm approaching the performance of the best algorithm regardless of the value of N.*

of the proposed heuristic is almost 80% for different values of N as shown in Fig. 4.7a. Additionally, Fig. 4.7b and Fig. 4.7c support this argument.

4.8.4. Lessons Learned

We can conclude the lessons learned in this paper based on the presented results as follows:

- The performance of the proposed online heuristic is stable regardless of the number of total clients N, as illustrated in Fig. 4.7.
- The accuracy of the proposed heuristic increases as the number of selected candidate clients R increases. However, as R goes up, the performance of the proposed online heuristic, the online random algorithm, and the offline best algorithm tends to converge as indicated in Fig. 4.6. This is because, with more candidate clients, algorithms have access to a larger portion of the overall dataset.
- As the number of best candidate clients (r_2) is increased, the performance of the proposed online heuristic is enhanced since better candidate clients are used as shown in Fig. 4.5.

4.9. Conclusions and Future Work

In this paper, the problem of optimizing accuracy in stateful federated learning by selecting the best candidate clients based on test accuracy is considered. Then, the problem of maximizing the probability of selecting the best candidate clients based on accuracy is formulated as a secretary problem and performance analysis is presented along with proofs. Based on the formulation, an online stateful federated learning heuristic is proposed to find the best

candidate clients. In addition, an IoT client alarm application is proposed that utilizes the proposed heuristic along with IoT device classification to identify unauthorized IoT devices and alert clients. To test the efficiency of the proposed heuristic, we run many experiments using a real IoT dataset and the performance of the online random algorithm and the offline best algorithm are compared against the performance of the proposed heuristic. Results show that the proposed heuristic performs better than the two state-of-the-art algorithms. Additionally, we notice the stability in the performance of the proposed heuristic compared against the performance of the other two algorithms regardless of the number of participating candidate clients. We also notice that when increasing the number of best selected candidate clients, the proposed heuristic becomes less competitive. This is because with more clients comes more data and thus the performance of algorithms converges regardless of how bad an algorithm in selecting candidate clients.

In the future, we plan to devise different variations of the secretary problem and provide performance analysis along with proofs for each. We also intend to run several experiments using a real dataset to evaluate those variations and compare their performance with the performance of the proposed heuristic.

CHAPTER 5

CONCLUSION AND FUTURE WORK

This chapter concludes the presented work and list ideas for future research.

5.1. Conclusion

In this work, we present three solutions in terms of real-time and online algorithms to optimize the use of resources in IoT applications. The main concluded points are listed as follows:

- In response to the strict quality of service (QoS) requirements in vehicular networks (VANETs), we propose a generic real-time heuristic that provides differentiated services based on a given set of flows and their corresponding severity metric. We show that our proposed heuristic can be used to offer differentiated services and improve QoS in VANETs. Additionally, we find out that the proposed prioritized processing heuristic is superior and provides better performance only when the system is under higher loads.

- We proposed to use vehicles as data ferries to transport data in smart communities as an alternative solution for the substantial infrastructural cost of setting up smart cities. Also, we proposed an online algorithm based on an ensemble of four online algorithms that strives to select the best vehicles in terms of waiting and delivery delays. Moreover, we note that the proposed algorithm outperforms other baseline algorithms by either performing better on the waiting time or on the delivery time, but not both.

- We optimize the accuracy of stateful federated learning by proposing an online algorithm inspired by the solution of the secretary problem to select the best candidate clients in terms of test accuracy to participate in training the global model. The performance of the proposed algorithm is stable regardless of the number of participating clients. Also, we notice that as the number of best clients increases, the proposed algorithm becomes less competitive compared against state-of-the-art algorithms.

5.2. Future Work

Resource management in IoT applications and especially in smart cities and smart vehicles is a very important research topic with more room for improvement. I'm planning to keep searching in this area to cover many gaps and come up with intelligent algorithms that lead to better results in terms of computation and communication. The following is a list of ideas inspired by the current research:

- Utilize the algorithm we develop for improving QoS in VANETs in a realistic scenario with a quality of service less than 10 ms.
- Develop an online algorithm to organize search and rescue operations using unmanned aerial vehicles such as drones during natural disasters. Actually, designing better algorithms leads to saving more lives.
- Study the pros and cons of variations of the online secretary problem when used to optimize the selection of resources in different IoT applications.

BIBLIOGRAPHY

[1] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, "Internet of things (iot) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5g-iot scenarios", *IEEE Access*, vol. 8, pp. 23 022–23 040, Jan. 2020.

[2] A. López-Vargas, M. Fuentes, and M. Vivar, "Challenges and opportunities of the internet of things for global development to achieve the united nations sustainable development goals", *IEEE Access*, vol. 8, pp. 37 202–37 213, Feb. 2020.

[3] K. L. Ang and J. K. P. Seng, "Application specific internet of things (asiots): Taxonomy, applications, use case and future directions", *IEEE Access*, vol. 7, pp. 56 577–56 590, May 2019.

[4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications", *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, Jun. 2015.

[5] Y. L. Morgan, "Notes on dsrc & wave standards suite: Its architecture, design, and characteristics", *IEEE Communications Surveys & Tutorials*, vol. 12, no. 4, pp. 504–518, 2010.

[6] "Ieee standard for wireless access in vehicular environments (wave) – multi-channel operation - redline", *IEEE Std 1609.4-2016 (Revision of IEEE Std 1609.4-2010) - Redline*, pp. 1–206, Mar. 2016.

[7] Y. Liu, *802.11 enhanced distributed channel access*, US Patent 9,155,027, Oct. 2015.

[8] N. Gupta, A. Prakash, and R. Tripathi, "Medium access control protocols for safety applications in vehicular ad-hoc network: A classification and comprehensive survey", *Vehicular Communications*, vol. 2, no. 4, pp. 223 –237, 2015, ISSN: 2214-2096. DOI: http://dx.doi.org/10.1016/j.vehcom.2015.10.001. [Online]. Available: //www.sciencedirect.com/science/article/pii/S2214209615000546.

[9] Y. J. Li, "An overview of the dsrc/wave technology", in *Quality, Reliability, Security and Robustness in Heterogeneous Networks: 7th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, QShine 2010, and Dedicated Short Range Communications Workshop, DSRC 2010, Houston, TX, USA, November 17-19, 2010, Revised Selected Papers*, X. Zhang and D. Qiao, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 544–558, ISBN: 978-3-642-29222-4. DOI: 10.1007/978-3-642-29222-4_38. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29222-4_38.

[10] "Ieee guide for wireless access in vehicular environments (wave) - architecture", *IEEE Std 1609.0-2013*, pp. 1–78, Mar. 2014. DOI: 10.1109/IEEESTD.2014.6755433.

[11] M. B. Brahim, E. B. Hamida, F. Filali, and N. Hamdi, "Performance impact of security on cooperative awareness in dense urban vehicular networks", in *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2015, pp. 268–274. DOI: 10.1109/WiMOB.2015.7347971.

[12] E. B. Hamida and M. A. Javed, "Channel-aware ecdsa signature verification of basic safety messages with k-means clustering in vanets", in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, Mar. 2016, pp. 603–610. DOI: 10.1109/AINA.2016.51.

[13] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, "A comprehensive survey on vehicular ad hoc network", *Journal of Network and Computer Applications*, vol. 37, pp. 380 –392, 2014, ISSN: 1084-8045. DOI: http://dx.doi.org/10.1016/j.jnca.2013.02.036. [Online]. Available: //www.sciencedirect.com/science/article/pii/S108480451300074X.

[14] M. A. Javed, E. Ben Hamida, and W. Znaidi, "Security in intelligent transport systems for smart cities: From theory to practice", *Sensors*, vol. 16, no. 6, p. 879, 2016, ISSN: 1424-8220. DOI: 10.3390/s16060879. [Online]. Available: http://www.mdpi.com/1424-8220/16/6/879.

[15] M. A. Javed and E. B. Hamida, "Adaptive security mechanisms for safety applications in internet of vehicles", in *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2016, pp. 1–6. DOI: 10.1109/WiMOB.2016.7763268.

[16] M. Timmers, S. Pollin, A. Dejonghe, L. V. der Perre, and F. Catthoor, "A distributed multichannel mac protocol for multihop cognitive radio networks", *IEEE Transactions on Vehicular Technology*, vol. 59, no. 1, pp. 446–459, Jan. 2010, ISSN: 0018-9545. DOI: 10.1109/TVT.2009.2029552.

[17] H. El Ajaltouni, A. Boukerche, and A. Mammeri, "A multichannel qos mac with dynamic transmit opportunity for vanets", *Mobile Networks and Applications*, vol. 18, no. 6, pp. 814–830, 2013. DOI: 10.1007/s11036-013-0475-6. [Online]. Available: http://dx.doi.org/10.1007/s11036-013-0475-6.

[18] J. M.-Y. Lim, Y. C. Chang, M. Y. Alias, and J. Loo, "Cognitive vanet with enhanced priority scheme", in *2014 International Conference on Telecommunications and Multimedia (TEMU)*, Jul. 2014, pp. 116–121. DOI: 10.1109/TEMU.2014.6917746.

[19] D. Lee, S. H. Ahmed, D. Kim, J. Copeland, and Y. Chang, "Distributed sch selection for concurrent transmissions in ieee 1609.4 multi-channel vanets", in *Communications (ICC), 2017 IEEE International Conference on*, IEEE, 2017, pp. 1–6.

[20] ——, "An efficient sch utilization scheme for ieee 1609.4 multi-channel environments in vanets", in *Communications (ICC), 2016 IEEE International Conference on*, IEEE, 2016, pp. 1–6.

[21] D. B. Rawat, D. C. Popescu, G. Yan, and S. Olariu, "Enhancing vanet performance by joint adaptation of transmission power and contention window size", *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 9, pp. 1528–1535, 2011.

[22] R. Baldessari, D. Scanferla, L. Le, W. Zhang, and A. Festag, "Joining forces for vanets: A combined transmit power and rate control algorithm", in *6th international workshop on intelligent transportation (WIT)*, 2010.

[23] M. Amadeo, C. Campolo, and A. Molinaro, "Enhancing {ieee} 802.11p/wave to provide infotainment applications in {vanets}", *Ad Hoc Networks*, vol. 10, no. 2, pp. 253 –269, 2012, Recent Advances in Analysis and Deployment of {IEEE} 802.11e and {IEEE} 802.11p Protocol Families, ISSN: 1570-8705. DOI: http://dx.doi.org/10.1016/j.adhoc.2010.09.013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870510001538.

[24] L. Urquiza-Aguiar, A. Vázquez-Rodas, C. Tripp-Barba, M. A. Igartua, L. J. de la Cruz Llopis, and E. S. Gargallo, "Max-min based buffer allocation for vanets", in *2014 IEEE 6th International Symposium on Wireless Vehicular Communications (WiVeC 2014)*, Sep. 2014, pp. 1–5. DOI: 10.1109/WIVEC.2014.6953231.

[25] C. Y. Chang, H. C. Yen, and D. J. Deng, "V2v qos guaranteed channel access in ieee 802.11p vanets", *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 1, pp. 5–17, Jan. 2016, ISSN: 1545-5971. DOI: 10.1109/TDSC.2015.2399912.

[26] Q. Wang, S. Leng, Y. Zhang, and H. Fu, "A qos supported multi-channel mac for vehicular ad hoc networks", in *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, IEEE, 2011, pp. 1–5.

[27] H. T. Cheng, H. Shan, and W. Zhuang, "Infotainment and road safety service support in vehicular networking: From a communication perspective", *Mechanical Systems and Signal Processing*, vol. 25, no. 6, pp. 2020 –2038, 2011, Interdisciplinary Aspects of Vehicle Dynamics, ISSN: 0888-3270. DOI: http://dx.doi.org/10.1016/j.ymssp.2010.11.009. [Online]. Available: //www.sciencedirect.com/science/article/pii/S0888327010004127.

[28] D. T. Tuan, S. Sakata, and N. Komuro, "Priority and admission control for assuring quality of i2v emergency services in vanets integrated with wireless lan mesh networks", in *2012 Fourth International Conference on Communications and Electronics (ICCE)*, Aug. 2012, pp. 91–96. DOI: 10.1109/CCE.2012.6315877.

[29] C. Chrysostomou, C. Djouvas, and L. Lambrinos, "Dynamically adjusting the min-max contention window for providing quality of service in vehicular networks", in *2012 The 11th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, Jun. 2012, pp. 16–23. DOI: 10.1109/MedHocNet.2012.6257117.

[30] W. Alasmary and W. Zhuang, "Mobility impact in {ieee} 802.11p infrastructureless vehicular networks", *Ad Hoc Networks*, vol. 10, no. 2, pp. 222 –230, 2012, Recent Advances in Analysis and Deployment of {IEEE} 802.11e and {IEEE} 802.11p Protocol Families, ISSN: 1570-8705. DOI: http://dx.doi.org/10.1016/j.adhoc.2010.06.006. [Online]. Available: //www.sciencedirect.com/science/article/pii/S1570870510000703.

[31] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani, "Exploiting context severity to achieve opportunistic service differentiation in vehicular ad hoc networks", *IEEE Transactions on Vehicular Technology*, vol. 63, no. 6, pp. 2901–2915, 2014.

[32] M. A. Salahuddin, A. Al-Fuqaha, F. Jacquelin, and Y. Shim, "Context severity based opportunistic service reprioritization for ieee 802.11 p vanets", in *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, IEEE, 2013, pp. 1623–1628.

[33] S. Banani and S. Gordon, "Selecting basic safety messages to verify in vanets using zone priority", in *The 20th Asia-Pacific Conference on Communication (APCC2014)*, Oct. 2014, pp. 423–428. DOI: 10.1109/APCC.2014.7092849.

[34] S. Biswas and J. Mišić, "Relevance-based verification of vanet safety messages", in *2012 IEEE International Conference on Communications (ICC)*, Jun. 2012, pp. 5124–5128. DOI: 10.1109/ICC.2012.6364399.

[35] J. Y. L. Boudec and P. Thiran, "A short tutorial on network calculus. i. fundamental bounds in communication networks", in *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, vol. 4, 2000, 93–96 vol.4. DOI: 10.1109/ISCAS.2000.858696.

[36] A. Ceselli and G. Righini, "An optimization algorithm for a penalized knapsack problem", en, *Operations Research Letters*, vol. 34, no. 4, pp. 394–404, Jul. 2006, ISSN: 01676377. DOI: 10.1016/j.orl.2005.06.001. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0167637705000751 (visited on 09/16/2016).

[37] L. Bedogni, M. Gramaglia, A. Vesco, M. Fiore, J. Härri, and F. Ferrero, "The bologna ringway dataset: Improving road network conversion in sumo and validating urban mobility via navigation services", *IEEE Transactions on Vehicular Technology*, vol. 64, no. 12, pp. 5464–5476, 2015.

[38] H. Rakouth, P. Alexander, A. J. Brown, W. Kosiak, M. Fukushima, L. Ghosh, C. Hedges, H. Kong, S. Kopetzki, R. Siripurapu, and J. Shen, "V2x communication technology: Field experience and comparative analysis", in *Proceedings of the FISITA 2012 World Automotive Congress: Volume 12: Intelligent Transport SystemITS & Internet of Vehicles*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 113–129, ISBN: 978-3-642-33838-0. DOI: 10.1007/978-3-642-33838-0_10. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33838-0_10.

[39] *VLC media player*. [Online]. Available: http://www.videolan.org/.

[40] *Wireshark - network protocol analyzer*. [Online]. Available: https://www.wireshark.org/.

[41] United Nations Population Division, *Data booklet - the world's cities in 2016*, 2016. [Online]. Available: http://tinyurl.com/WorldCities2016.

[42] S. Tabatabai, I. Mohammed, A. Al-Fuqaha, and M. A. Salahuddin, "Managing a cluster of IoT brokers in support of smart city applications", in *IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Oct. 2017, pp. 1–6. DOI: 10.1109/PIMRC.2017.8292620.

[43] I. Jawhar, N. Mohamed, and J. Al-Jaroodi, "Networking and communication for smart city systems", in *IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation*, Aug. 2017, pp. 1–7. DOI: 10.1109/UIC-ATC.2017.8397563.

[44] A. Gharaibeh, M. A. Salahuddin, S. J. Hussini, A. Khreishah, I. Khalil, M. Guizani, and A. Al-Fuqaha, "Smart cities: A survey on data management, security, and enabling technologies", *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2456–2501, 2017, ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2736886.

[45] Z. Su, Y. Hui, and Q. Yang, "The next generation vehicular networks: A content-centric framework", *IEEE Wireless Communications*, vol. 24, no. 1, pp. 60–66, Feb. 2017, ISSN: 1536-1284. DOI: 10.1109/MWC.2017.1600195WC.

[46] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks", in *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, ACM, 2004, pp. 187–198.

[47] J. Paradells, C. Gomez, I. Demirkol, J. Oller, and M. Catalan, "Infrastructureless smart cities. use cases and performance", in *International Conference on Smart Communications in Network Technologies (SaCoNeT)*, Jun. 2014, pp. 1–6. DOI: 10.1109/SaCoNeT.2014.6867772.

[48] A. Bouroumine, M. Zekraoui, and M. Abdelilah, "The influence of the opportunistic vehicular networks on smart cities management study case on Agdal district in Rabat city", in *4th IEEE International Colloquium on Information Science and Technology (CiSt)*, Oct. 2016, pp. 830–834. DOI: 10.1109/CIST.2016.7805002.

[49] M. Aloqaily, I. A. Ridhawi, B. Kantraci, and H. T. Mouftah, "Vehicle as a resource for continuous service availability in smart cities", in *IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Oct. 2017, pp. 1–6. DOI: 10.1109/PIMRC.2017.8292752.

[50] F. Hagenauer, C. Sommer, R. Onishi, M. Wilhelm, F. Dressler, and O. Altintas, "Interconnecting smart cities by vehicles: How feasible is it?", in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2016, pp. 788–793. DOI: 10.1109/INFCOMW.2016.7562184.

[51] M. A. Khan, S. Sargento, and M. Luis, "Data collection from smart-city sensors through large-scale urban vehicular networks", in *IEEE 86th Vehicular Technology Conference (VTC-Fall)*, Sep. 2017, pp. 1–6. DOI: 10.1109/VTCFall.2017.8288308.

[52] C. Giannini, P. Calegari, C. Buratti, and R. Verdone, "Delay tolerant network for smart city: Exploiting bus mobility", in *AEIT International Annual Conference (AEIT)*, Oct. 2016, pp. 1–6. DOI: 10.23919/AEIT.2016.7892779.

[53] M. Otomo, K. Hashimoto, N. Uchida, and Y. Shibata, "Mobile cloud computing usage for onboard vehicle servers in collecting disaster data information", in *IEEE 8th International Conference on Awareness Science and Technology (iCAST)*, Nov. 2017, pp. 475–480. DOI: 10.1109/ICAwST.2017.8256504.

[54] M. Bonola, L. Bracciale, P. Loreti, R. Amici, A. Rabuffi, and G. Bianchi, "Opportunistic communication in smart city: Experimental insight with small-scale taxi fleets as data carriers", *Ad Hoc Networks*, vol. 43, pp. 43 –55, 2016, ISSN: 1570-8705. DOI: https://doi.org/10.1016/j.adhoc.2016.02.002.

[55] L. Yao, A. Chen, J. Deng, J. Wang, and G. Wu, "A cooperative caching scheme based on mobility prediction in vehicular content centric networks", *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 5435–5444, Jun. 2018, ISSN: 0018-9545. DOI: 10.1109/TVT.2017.2784562.

[56] P. C. Besse, B. Guillouet, J. Loubes, and F. Royer, "Destination prediction by trajectory distribution-based model", *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, Aug. 2018, ISSN: 1524-9050. DOI: 10.1109/TITS.2017.2749413.

[57] X. Wang, W. Wu, and D. Qi, "Mobility-aware participant recruitment for vehicle-based mobile crowdsensing", *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4415–4426, May 2018, ISSN: 0018-9545. DOI: 10.1109/TVT.2017.2787750.

[58] A. Narayanan, N. Mitrovic, M. T. Asif, J. Dauwels, and P. Jaillet, "Travel time estimation using speed predictions", in *IEEE 18th International Conference on Intelligent Transportation Systems*, Sep. 2015, pp. 2256–2261. DOI: 10.1109/ITSC.2015.364.

[59] A. Broder, A. Kirsch, R. Kumar, M. Mitzenmacher, E. Upfal, and S. Vassilvitskii, "The hiring problem and lake wobegon strategies", *SIAM Journal on Computing*, vol. 39, no. 4, pp. 1233–1255, 2010. DOI: 10.1137/07070629X. eprint: https://doi.org/10.1137/07070629X. [Online]. Available: https://doi.org/10.1137/07070629X.

[60] S. Vassilvitskii, A. Broder, A. Kirsch, R. Kumar, M. Mitzenmacher, and E. Upfal, *The hiring problem: Going beyond secretaries*, 2007. [Online]. Available: http://theory.stanford.edu/~sergei/slides/hiring-dagstuhl.pdf.

[61] H. Zhu and M. Li, *Studies on Urban Vehicular Ad-hoc Networks*, ser. SpringerBriefs in Computer Science. New York, NY: Springer New York, 2013, ISBN: 978-1-4614-8047-1. DOI: 10.1007/978-1-4614-8048-8. [Online]. Available: http://link.springer.com/10.1007/978-1-4614-8048-8.

[62] R. Moussalli, M. Srivatsa, and S. Asaad, "Fast and flexible conversion of geohash codes to and from latitude/longitude coordinates", in *2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines*, May 2015, pp. 179–186. DOI: 10.1109/FCCM.2015.18.

[63] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence", *IEEE Transactions on Industrial Informatics*, pp. 1–1, Oct. 2019, ISSN: 1941-0050. DOI: 10.1109/TII.2019.2945367.

[64] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks", *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4641–4654, May 2020, ISSN: 2372-2541. DOI: 10.1109/JIOT.2020.2964162.

[65] J. Jeon, J. Kim, J. Huh, H. Kim, and S. Cho, "Overview of distributed federated learning: Research issues, challenges, and biomedical applications", in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju Island, Korea (South), Oct. 2019, pp. 1426–1427. DOI: 10.1109/ICTC46691.2019.8939954.

[66] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang, "Towards efficient and privacy-preserving federated deep learning", in *2019 IEEE International Conference on Communications (ICC)*, Shanghai, China, May 2019, pp. 1–6. DOI: 10.1109/ICC.2019.8761267.

[67] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications", *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, Jan. 2019, ISSN: 2157-6904. DOI: 10.1145/3298981. [Online]. Available: https://doi.org/10.1145/3298981.

[68] Q. Li, Z. Wen, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection", *ArXiv*, vol. abs/1907.09693, 2019. [Online]. Available: http://arxiv.org/abs/1907.09693.

[69] D. Liu, T. Miller, R. Sayeed, and K. Mandl, "FADL: federated-autonomous deep learning for distributed electronic health record", *ArXiv*, vol. abs/1811.11400, 2018. [Online]. Available: http://arxiv.org/abs/1811.11400.

[70] D. Conway-Jones, T. Tuor, S. Wang, and K. Leung, "Demonstration of federated learning in a resource-constrained networked environment", in *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, Washington, DC, USA, Jun. 2019, pp. 484–486. DOI: 10.1109/SMARTCOMP.2019.00095.

[71] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach", *IEEE Access*, vol. 8, pp. 23 920–23 935, Jan. 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2968399.

[72] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data", in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, A. Singh and J. Zhu, Eds., ser. Proceedings of Machine Learning Research, vol. 54, Fort Lauderdale, FL, USA: PMLR, Apr. 2017, pp. 1273–1282. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a.html.

[73] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions", *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[74] P. Kairouz, H. McMahan, B. Avent, A. Bellet, M. Bennis, A. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R.D'Oliveira, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. Stich, Z. Sun, A. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning", *ArXiv*, vol. abs/1912.04977, 2019. [Online]. Available: http://arxiv.org/abs/1912.04977.

[75] V. Selis and A. Marshall, "A classification-based algorithm to detect forged embedded machines in IoT environments", *IEEE Systems Journal*, vol. 13, no. 1, pp. 389–399, Mar. 2019.

[76] W. Lim, N. Luong, D. Hoang, Y. Jiao, Y. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey", *ArXiv*, vol. abs/1909.11875, 2019. [Online]. Available: http://arxiv.org/abs/1909.11875.

[77] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecný, S. Mazzocchi, H. McMahan, T. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design", *ArXiv*, vol. abs/1902.01046, 2019. [Online]. Available: http://arxiv.org/abs/1902.01046.

[78] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems", *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019, ISSN: 1558-0008. DOI: 10.1109/JSAC.2019.2904348.

[79] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT", *IEEE Internet of Things Journal*, pp. 1–1, Nov. 2019.

[80] H. Zhu and Y. Jin, "Multi-objective evolutionary federated learning", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 4, pp. 1310–1322, Apr. 2020.

[81] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent", *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1754–1766, Feb. 2020.

[82] Y. Liu, Y. Kang, X. Zhang, L. Li, Y. Cheng, T. Chen, M. Hong, and Q. Yang, "A communication efficient collaborative learning framework for distributed features", *ArXiv*, vol. abs/1912.11187, 2020. [Online]. Available: http://arxiv.org/abs/1912.11187.

[83] X. Yao, C. Huang, and L. Sun, "Two-stream federated learning: Reduce the communication costs", in *2018 IEEE Visual Communications and Image Processing (VCIP)*, Dec. 2018, pp. 1–4.

[84] L. Liu, J. Zhang, S. Song, and K. Letaief, "Client-edge-cloud hierarchical federated learning", *ArXiv*, vol. abs/1905.06641, 2019. [Online]. Available: http://arxiv.org/abs/1905.06641.

[85] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Federated learning-based computation offloading optimization in edge computing-supported internet of things", *IEEE Access*, vol. 7, pp. 69 194–69 201, Jun. 2019.

[86] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "Performance optimization of federated learning over wireless networks", in *2019 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2019, pp. 1–6.

[87] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge", in *2019 IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–7. DOI: 10.1109/ICC.2019.8761315.

[88] L. WANG, W. WANG, and B. LI, "Cmfl: Mitigating communication overhead for federated learning", in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, Jul. 2019, pp. 954–964. DOI: 10.1109/ICDCS.2019.00099.

[89] T. T. Anh, N. C. Luong, D. Niyato, D. I. Kim, and L. Wang, "Efficient training management for mobile crowd-machine learning: A deep reinforcement learning approach", *IEEE Wireless Communications Letters*, vol. 8, no. 5, pp. 1345–1348, May 2019.

[90] H. Nguyen, N. Luong, J. Zhao, C. Yuen, and D. Niyato, "Resource allocation in mobility-aware federated learning networks: A deep reinforcement learning approach", *ArXiv*, vol. abs/1910.09172, 2019. [Online]. Available: http://arxiv.org/abs/1910.09172.

[91] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, and R. Yonetani, "Hybrid-fl for wireless networks: Cooperative learning mechanism using non-iid data", *arXiv*, vol. abs/1905.07210, 2019. [Online]. Available: http://arxiv.org/abs/1905.07210.

[92] F. Sattler, S. Wiedemann, K. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data", *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, Nov. 2019.

[93] J. Konecný, H. McMahan, F. Yu, P. Richtárik, A. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency", *ArXiv*, vol. abs/1610.05492, 2017. [Online]. Available: http://arxiv.org/abs/1610.05492.

[94] S. Caldas, J. Konecný, H. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements", *ArXiv*, vol. abs/1812.07210, 2018. [Online]. Available: http://arxiv.org/abs/1812.07210.

[95] T. Ferguson, "Who solved the secretary problem", *Statistical Science*, vol. 4, no. 2, pp. 282–289, 1989.

[96] L. Bayón, P. Fortuny, J. Grau, A. Oller-Marcén, and M. Ruiz, "The best-or-worst and the postdoc problems with random number of candidates", *Journal of Combinatorial Optimization*, vol. 38, pp. 86–110, Jul. 2019.

[97] P. Freeman, "The secretary problem and its extensions: A review", *International Statistical Review*, vol. 51, pp. 189–206, Aug. 1983.

[98] J. Gilbert and F. Mosteller, "Recognizing the maximum of a sequence", *Journal of the American Statistical Association*, vol. 61, no. 313, pp. 35–73, 1966. DOI: 10.1080/01621459.1966.10502008. [Online]. Available: https://amstat.tandfonline.com/doi/abs/10.1080/01621459.1966.10502008.

[99] R. Kleinberg, "A multiple-choice secretary algorithm with applications to online auctions", in *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '05, Vancouver, British Columbia, Canada: Society for Industrial and Applied Mathematics, Jan. 2005, 630–631, ISBN: 0898715857.

[100] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg, "Online auctions and generalized secretary problems", *SIGecom Exch.*, vol. 7, no. 2, Jun. 2008. DOI: 10.1145/1399589.1399596. [Online]. Available: https://doi.org/10.1145/1399589.1399596.

[101] L. Guo and I. Matta, "The war between mice and elephants", in *Proceedings Ninth International Conference on Network Protocols. ICNP 2001*, Riverside, CA, USA, Nov. 2001, pp. 180–188. DOI: 10.1109/ICNP.2001.992898.

[102] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg, "Online auctions and generalized secretary problems", *SIGecom Exch.*, vol. 7, no. 2, Jun. 2008. DOI: 10.1145/1399589.1399596. [Online]. Available: https://doi.org/10.1145/1399589.1399596.

[103] R. Vaze, "Competitive ratio analysis of online algorithms to minimize packet transmission time in energy harvesting communication system", in *2013 Proceedings IEEE INFOCOM*, Turin, Italy, Apr. 2013, pp. 115–1123.

[104] M. Shahid, G. Blanc, Z. Zhang, and H. Debar, "IoT devices recognition through network traffic analysis", in *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, 2018, pp. 5187–5192.

[105] M. Santos, R. Andrade, D. Gomes, and A. Callado, "An efficient approach for device identification and traffic classification in IoT ecosystems", in *2018 IEEE Symposium on Computers and Communications (ISCC)*, Natal, Brazil, Jun. 2018, pp. 304–309.

[106] J. Bugeja, P. Davidsson, and A. Jacobsson, "Functional classification and quantitative analysis of smart connected home devices", in *2018 Global Internet of Things Summit (GIoTS)*, Bilbao, Spain, Jun. 2018, pp. 1–6.

[107] B. Desai, D. Divakaran, I. Nevat, G. Peter, and M. Gurusamy, "A feature-ranking framework for IoT device classification", in *2019 11th International Conference on Communication Systems Networks (COMSNETS)*, Bengaluru, India, India, Jan. 2019, pp. 64–71.

[108] F. Shaikh, E. Bou-Harb, J. Crichigno, and N. Ghani, "A machine learning model for classifying unsolicited IoT devices by observing network telescopes", in *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*, Limassol, Cyprus, Jun. 2018, pp. 938–943.

[109] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. Sadeghi, and S. Tarkoma, "IoT SENTINEL: Automated device-type identification for security enforcement in IoT", in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, Atlanta, GA, USA, Jun. 2017, pp. 2177–2184.

[110] A. Sivanathan, D. Sherratt, H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Characterizing and classifying IoT traffic in smart cities and campuses", in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Atlanta, GA, USA, May 2017, pp. 559–564.

[111] Y. Meidan, M. Bohadana, A. Shabtai, J. Guarnizo, M. Ochoa, N. Tippenhauer, and Y. Elovici, "Profiliot: A machine learning approach for IoT device identification based on network traffic analysis", in *Proceedings of the Symposium on Applied Computing*, ser. SAC '17, Marrakech, Morocco: Association for Computing Machinery, Apr. 2017, 506–509, ISBN: 9781450344869. DOI: 10.1145/3019612.3019878. [Online]. Available: https://doi.org/10.1145/3019612.3019878.

[112] A. Sivanathan, H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying IoT devices in smart environments using network traffic characteristics", *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, Aug. 2019.

[113] P. Perricone, B. Hudson, B. Anderson, B. Long, and D. McGrew, *Joy a package for capturing and analyzing network data features*, 2nd ed., Cisco Systems, Jan. 2018.

[114] I. Mohammed, *Federated IoT classification*, https://github.com/IhabMoha/Federated-IoT-Classification, 2020.

[115] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, Oct. 2011.