Management of Data Brokers in Support of Smart Community Applications

Shadha Tabatabai

*Western Michigan University*, shatha_alhassani@yahoo.com

Follow this and additional works at: https://scholarworks.wmich.edu/dissertations

Part of the Artificial Intelligence and Robotics Commons

**Recommended Citation**
https://scholarworks.wmich.edu/dissertations/3832
The widespread use of smart devices has led to the Internet of Things (IoT) revolution. Big data generated by billions of devices must be analyzed to make better decisions. However, this introduces security, communication, and processing problems. To solve these problems, we develop algorithms to enhance the work of brokers. We focus our efforts on three problems.

In the first problem, brokers are used in the cloud along with Software Defined Network (SDN) switches. We formulate minimizing brokers’ load difference within a reconfiguration budget with the constraint of indivisible topics as an Integer Linear Programming (ILP) problem. We show that the problem is NP-Hard and propose a heuristic driven by long-term statistics of topics. The proposed heuristic is evaluated with realistic simulation traffic traces and compared against a threshold-based baseline heuristic driven by instantaneous statistics of topics. Results show that the proposed heuristic performs up to 2000% better load distribution than the baseline heuristic and at least 27% less topic switching.

In the second problem, we used vehicles as data brokers for exchanging data between smart devices and service providers in the cloud. We propose an opportunistic algorithm that strives to select vehicles in order to maximize Local Community Broker’s (LCB) service time. The proposed opportunistic algorithm utilizes an ensemble of online selection algorithms by running all of them together in passive mode and selecting the one that has performed the best in recent history. The data set used in the proposed algorithm is evaluated using real taxi traces from the city of Shanghai in China and compared against a baseline of 9 Threshold Based Online (TBO) algorithms. A number of experiments are conducted and
results indicate that the proposed algorithm achieves up to 87% more service time with up to 10% fewer vehicle selections compared to the best performing TBO algorithm.

In the third problem, we used a broker (server) to implement a proposed Federated Learning (FL) algorithm that tackle security, communication, and accuracy problems. The proposed algorithm groups clients randomly in many clusters, each with a model selected randomly to explore the performance of different models. The clusters are then trained in a repetitive process where the worst performing cluster is removed in each iteration until one cluster remains. In each iteration, some clients are expelled from clusters either due to using poisoned data or low performance. The surviving clients are exploited in the next iteration. The remaining cluster with surviving clients is then used for training the best FL model (i.e., remaining FL model). Communication cost is reduced since fewer clients are used in the final training of the FL model. To evaluate the performance of the proposed algorithm, we conduct a number of experiments using FMNIST dataset and compare the result against the random FL algorithm. The experimental results show that the proposed algorithm outperforms the baseline algorithm in terms of accuracy, communication cost, and security.
Management of Data Brokers in Support of Smart Community Applications

by

Shadha Muhi Noor Tabatabai

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
Spring 2022

Doctoral Committee:

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

I dedicate this work to the souls of my mother and father, who always loved, trusted, and supported me till the last moment of their lives, and I’m sure that their spirits are happy for me and proud of me.
ACKNOWLEDGEMENTS

I'm grateful to God for blessing me and helping me during this long journey. Thanks to my brother Adel, my brother in law Mohammed, and my sisters Faten and Maha, who supported me and loved me endlessly. Thanks to my husband Ihab, my soul mate, for being there for me in every hard moment during my study journey. Thanks to my kids Mohammed, Anas, and Yosef for loving me and being patient with me. Thanks to my friends and colleagues for being always there for me, especially my best friend Asma, who treated me like her sister. I also want to thank my supervisor Dr. Ala, who always supported me in all my difficult situations, was always understanding, and without his help, kindness, guidance, and support, this work would not be possible. 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.
* The Iraqi Ministry of Higher Education and Scientific Research.

Shadha Muhi Noor Tabatabai
TABLE OF CONTENTS

DEDICATION ........................................................................... i

ACKNOWLEDGEMENTS .......................................................... ii

LIST OF TABLES .................................................................... vi

LIST OF FIGURES ................................................................... vii

LIST OF ABBREVIATIONS ......................................................... viii

1. INTRODUCTION ................................................................. 1
   1.1. Introduction ................................................................. 1
   1.2. Statement of the Problem ............................................... 1
   1.3. Purpose of the Research ............................................... 2
   1.4. Significance of the Study ............................................... 2
   1.5. Contributions .............................................................. 3
   1.6. Structure of the Dissertation ......................................... 4

2. Managing a Cluster of IoT Brokers in Support of Smart City Applications .... 5
   2.1. Introduction ................................................................. 5
   2.2. Related Works ............................................................ 7
   2.3. System Model ............................................................. 8
   2.4. Problem Formulation .................................................. 11
   2.5. Dataset ................................................................... 11
   2.6. Heuristic Solution ....................................................... 13
# Table of Contents—Continued

2.7. Results and Discussions ................................................................. 16  
2.8. Conclusions and Future Work .................................................... 17  

3. Opportunistic Selection of Vehicular Data Brokers as Relay Nodes to the Cloud  ......................... 25  
   3.1. Introduction ........................................................................... 25  
   3.2. Motivation ........................................................................... 28  
   3.3. Related Work ...................................................................... 29  
   3.4. System Model ...................................................................... 32  
   3.5. Online Hiring Algorithms .................................................... 34  
   3.6. Proposed Heuristic Solution ................................................ 35  
   3.7. Illustrative Example ............................................................... 37  
   3.8. Experimental Results ............................................................ 38  
      3.8.1. Dataset and Experimental Settings .................................. 39  
      3.8.2. Results Discussion ....................................................... 40  
   3.9. Conclusions and Future Work ................................................ 43  

4. Exploration and Exploitation in Federated Learning to Exclude Clients with Poisoned Data ................................................................. 44  
   4.1. Introduction ........................................................................... 45  
   4.2. Related Work ...................................................................... 46  
      4.2.1. Algorithm Optimization ................................................. 48  
      4.2.2. Poisoning Attacks Against Federated Learning ............... 49  
   4.3. System Model ...................................................................... 50  
   4.4. Proposed Algorithm ............................................................... 51  
   4.5. Experimental Settings ............................................................ 54  
      4.5.1. Dataset ......................................................................... 54
# Table of Contents—Continued

4.5.2. Experiments ..................................................... 55  
4.6. Results Discussion ................................................. 56  
  4.6.1. Performance of Clustering ..................................... 59  
  4.6.2. Performance with Poisoned Dataset .......................... 60  
  4.6.3. Performance with Expelling Clients ......................... 61  
  4.6.4. Lessons Learned ................................................ 62  
4.7. Conclusions and Future Work .............................. 62  
5. CONCLUSION AND FUTURE WORK ............................ 64  
  5.1. Conclusion ........................................................... 64  
  5.2. Future Work ........................................................... 65  
BIBLIOGRAPHY ............................................................ 66
LIST OF TABLES

3.1 Performance of the proposed algorithm compared with two TBO algorithms. 38
3.2 Performance of the Proposed Algorithm Compared to TBO-10 Online Algorithm In Areas of High Traffic Volume 41

4.1 An overview of the related work. 47
4.2 Simulation Parameters. 57
4.3 Efficiency of expelling clients with $N = 400$, $P = 40\%$, $X = 20\%$, and $C = 8$. 60
LIST OF FIGURES

2.1 System model showing the communication between the smart city and the cloud via SDN and publish/subscribe protocols. .................................................. 18
2.2 Dataset characteristics. ................................................................. 19
2.3 Load difference and topic switching of the baseline heuristic for different threshold values. ............................................................. 20
2.4 Load difference of the baseline heuristic over time for two thresholds with both having one reconfiguration but different average load differences. .......... 21
2.5 Average load difference and total topic switching of the proposed and the baseline heuristics. .............................................................. 22
2.6 Performance of the proposed heuristic using three reconfigurations limit and the baseline heuristics using two thresholds. ......................... 23
2.7 Brokers’ load difference over time of the proposed and the baseline heuristics with zero reconfigurations. ..................................................... 24
3.1 Observations on the performance of algorithms on single and multiple zones. 30
3.2 Publish-subscribe model between the smart devices and the SCMC through the LCBs. ................................................................. 33
3.3 Proposed algorithm switches to the best performing algorithm in terms of service time over time. ......................................................... 37
3.4 Performance of algorithms for different traffic volumes for five days results. The proposed algorithm delivers more service time than other algorithms in a high-traffic area. .............................................................. 42
4.1 An illustration of the system model. The cloud server running the proposed algorithm communicates with the local servers in different clusters to train a number of global models in stage 2, then the best model is trained in stage 3. 52
4.2 Accuracy of the proposed v.s. the baseline algorithms with $P = 0\%$ and $X = 0\%$. ................................................................. 57
4.3 Accuracy of models (i.e clusters) used in the proposed algorithm with $N = 400$, $P = 40\%$, and $X = 20\%$. ......................................................... 58
4.4 Accuracy of the proposed v.s. the baseline algorithms with $C = 32$ and $N = 900$. 58
4.5 Communication cost of the proposed v.s. the baseline algorithms with $N = 400$, $P = 0\%$, and $C = 8$. ......................................................... 59
4.6 Meaning of Negative, Positive, True, and False in Table 4.3 .................. 60
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSN</td>
<td>Body Sensor Networks</td>
</tr>
<tr>
<td>CSB</td>
<td>Cloud Service Broker</td>
</tr>
<tr>
<td>CSC</td>
<td>Cloud Service Customer</td>
</tr>
<tr>
<td>CSP</td>
<td>Cloud Service Provider</td>
</tr>
<tr>
<td>DPI</td>
<td>Deep Packet Inspection</td>
</tr>
<tr>
<td>FD</td>
<td>Federated Learning</td>
</tr>
<tr>
<td>FGSM</td>
<td>Fast Gradient Sign Method</td>
</tr>
<tr>
<td>ILP</td>
<td>Integer Linear Programming</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>IoV</td>
<td>Internet of Vehicles</td>
</tr>
<tr>
<td>LCB</td>
<td>Local Community Broker</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>MQTT</td>
<td>Message Queuing Telemetry Transport</td>
</tr>
<tr>
<td>OBU</td>
<td>On-Board Unit</td>
</tr>
<tr>
<td>PSO</td>
<td>Particle Swarm Optimization</td>
</tr>
<tr>
<td>QoE</td>
<td>Quality of Experience</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RSU</td>
<td>Road Side Unit</td>
</tr>
<tr>
<td>SCMC</td>
<td>Smart Community Management Center</td>
</tr>
<tr>
<td>SVaaS</td>
<td>Smart Vehicle as a Service</td>
</tr>
<tr>
<td>TBO</td>
<td>Threshold Based Online algorithm</td>
</tr>
<tr>
<td>TOS</td>
<td>Type of Service</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>VANETs</td>
<td>Vehicular Ad-hoc Networks</td>
</tr>
<tr>
<td>VFC</td>
<td>Vehicular Fog Computing</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1. Introduction

Humans were the main users of the Internet until recently. However, this is changing fast as billions of devices, such as sensors and actuators, are connected to the Internet, which results in what is known as the Internet of Things (IoT). There are many applications of IoT, such as smart cities, smart energy, and smart agriculture, to name a few.

IoT applications are facing many issues related to management, reliability, scalability, mobility, availability, performance, interoperability, and security and privacy. As a result, brokers or servers that control the traffic, process data, and provide load balance while maintaining the security of data in IoT applications need intelligent and efficient offline and online algorithms. In this work, we tackle some of these problems by developing algorithms to enhance performance and reduce communication costs while maintaining security and privacy.

1.2. Statement of the Problem

The huge amount of data generated by billions of devices in IoT applications introduces communication, processing, and security problems. Offline and online algorithms used in brokers (servers) are not optimized to tackle these problems.
1.3. Purpose of the Research

In this work, we develop algorithms to efficiently manage brokers (servers) to reduce the cost of processing and communications between brokers and clients while enhancing security.

1.4. Significance of the Study

In one study, we show that using the proposed algorithm with Software-Defined Networking (SDN) devices to minimize the load difference between brokers in support of data and decision fusion applications in smart cities reduces the network overhead. IoT devices (publishers) generate and publish their data through channels known as topics such as temperature in smart cities. Subscribers subscribe for services based on topics. To support data and decision fusion applications, all sensory data messages that share the same topic are routed to the same broker so that the same broker can fuse all those messages. We proposed a data flow management algorithm that runs in the Software-Defined Networking (SDN) controller to reconfigure the flow tables in SDN switches to reroute messages of the same topic to a specific broker. This reconfiguration is run every period to minimize the load difference between brokers.

In a second study, we used an infrastructure-less system that avoid expensive infrastructure for communication. We developed an online algorithm that selects vehicles to serve as Local Community Brokers (LCB), substituting Road Side Units (RSUs). The proposed algorithm select vehicles that stay longer in the area to manage communication between smart devices and the cloud. The proposed algorithm utilizes an ensemble of online selection algorithms, runs all of them together in passive mode, and selects the one that has performed the best in recent history.
In the third study, we develop a Federated Learning (FL) algorithm inspired by evolutionary techniques to reduce communication costs, enhance accuracy, and enhance security in FL applications. The proposed algorithm distributes clients in a number of clusters and uses different Machine Learning (ML) models for every cluster. Clients are trained for several communication rounds in an iterative process, and clients with poisoned or weak datasets are expelled. Additionally, the algorithm runs a number of phases, and in each phase, the worst-performing cluster is removed, and its clients are moved to the best-performing cluster. Eventually, the best performing cluster with the best clients remains.

1.5. Contributions

This dissertation is based on the following publications:


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


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

Managing a Cluster of IoT Brokers in Support of Smart City Applications

Publish/subscribe brokers enable the efficient dissemination of events to a large number of subscribers in support of smart city applications. These events convey data gathered from devices and published to named logical channels called topics. Software-Defined Networking (SDN) can provide the advantage of balancing the load between brokers by switching topics between brokers. However, this switching results in network overhead. Besides, supporting data and decision fusion applications is a challenging task since sensory data has to be fused before being forwarded to subscribers. Therefore, we propose an algorithm utilized by the SDN controller to minimize the load difference between brokers while respecting a reconfiguration limit in support of data and decision fusion applications. We formulate minimizing brokers’ load difference within a reconfiguration budget with the constraint of indivisible topics as an Integer Linear Programing (ILP) problem. We show that the problem is NP-Hard and propose a heuristic driven by long-term statistics of topics. The proposed heuristic is evaluated with realistic simulation traffic traces and compared against a threshold-based baseline heuristic driven by instantaneous statistics of topics. Results show that the proposed heuristic performs up to 2000% better load distribution than the baseline heuristic and at least 27% less topic switching.

2.1. Introduction

In 2016, 54.4% of the world’s population lived in cities. This percentage is expected to increase to 60% by 2030[1]. In addition, 8.4 billion "things" will be connected to the Internet in 2017, increasing to 20.4 billion by 2020[2]. The trend of moving to urban centers and
the increasing number of connected things in the cities require infrastructure and services. These services and infrastructure must meet the needs of citizens and visitors, which opens the door for the development of smart cities. To gather data in smart cities, topic-based publish/subscribe protocols let Internet of Things (IoT) devices publish their data as messages to named logical channels, known as topics. Subscribers register for services based on these topics. To forward messages between publishers and subscribers, a smart city requires a cluster of brokers in the cloud.

Software-Defined Networking (SDN) is the technology of choice for managing the network in the context of a smart city [3]. The significant benefits of using SDN with IoT are presented in [4]. SDN switches have a flow table with rules that dictate the assignment of topics to brokers. To balance the load between brokers, the SDN controller reconfigures flow table rules in SDN switches (i.e., change the assignment of topics to brokers). This reconfiguration routes topic-based messages to the appropriate broker to minimize the load between brokers. Consequently, topics are switched between brokers (i.e., topics shift from one broker to the other). Nevertheless, this switching introduces overhead on the network.

In this work, we focus on minimizing the load difference between brokers in the cluster given an SDN reconfiguration budget (i.e., topic switching budget) in support of data and decision fusion applications. In data and decision fusion applications, multiple sensory data messages are fused or processed by a broker, and results are forwarded to the subscribers in one message. For example, in Body Sensor Networks (BSN), a topic can be the fusion of physiological and psychological sensory data sent to subscribers (e.g., caregivers).

In the case of divisible topics, where subscribers of a topic are distributed across brokers, balancing the load between brokers is a simple problem. However, sensory data messages are forwarded from the SDN switch to all brokers, which introduces overhead on the network. Besides, the same sensory data processing or fusing will be repeated in all brokers.
To provide efficient performance while supporting data and decision fusion applications, all sensory data messages that share the same topic are routed to the same broker. Therefore, subscribers per topic are not distributed (i.e., the topic is indivisible) across multiple brokers. Consequently, network overhead is reduced but balancing the load between brokers is a challenge.

To address this problem, we propose a system that utilizes (i) SDN network with topic-based publish/subscribe protocol, such as Message Queuing Telemetry Transport (MQTT) or Apache Kafka, and (ii) a long-term topic balancing heuristic on the SDN controller, which is the contribution of this paper. We formulate the problem as an Integer Linear Programming (ILP) problem and show that it is NP-Hard. To evaluate the performance of the proposed heuristic, a threshold-based topic balancing baseline heuristic is implemented. Both heuristics are evaluated using a realistic simulation of traffic traces. Results show that the proposed heuristic performs up to 2000% better than the baseline in load difference and at least 27% less topic switching.

2.2. Related Works

Conventional load balancing algorithms try to strike a balance between brokers using greedy strategies. They do so by distributing the load between brokers to the extent possible every time step; thus, resulting in high topic switching. The literature is rich with research on routing optimization of topic-based publish/subscribe systems[5]. However, using SDN with these systems is first studied in[6].

Wang et al.[5] propose a system named SDNPS, which utilizes SDN central control with a topic-based publish/subscribe system for better and non-redundant topic-based events dissemination. SDNPS works by abstracting and aggregating the link state of the network to capture a general overview of the topology. Next, the system predicts traffic distribution
for the future. Finally, a per-topic minimum overlay network is calculated, and the shortest path algorithm is used to extract multiple routing paths. The authors of the SDNPS system claim that their system achieves a better trade-off between links’ global load balancing and per-topic events’ minimum cost forwarding.

Bhowmik et al. [7] propose a scalable publish/subscribe SDN-based middleware named PLEROMA that achieves an effective forwarding at line-rate. They state that publish/subscribe middleware can access SDN switches to increase line-rate performance, bandwidth efficiency, and decrease latency when forwarding messages between publishers and subscribers. However, it is challenging to maintain a line-rate performance since publishers and subscribers change their interests dynamically.

Xu et al.[8] use SDN-based fog computing and implement an MQTT broker at the edge switches instead of the cloud due to its proximity to the data source. They show their technique to improve message delivery performance. Nonetheless, edge switches support simple analysis and can not support complicated computations as offered in the cloud. Whereas Xia et al.[9] propose a method called community-based load balancing, which utilizes interest similarity in the community to cluster brokers in a way that enhances the network performance.

None of the mentioned studies deal with the problem of minimizing the load difference between brokers within a reconfiguration limit. Additionally, data and decision fusion applications in the smart city context were not considered in these studies.

### 2.3. System Model

We assume a smart city with devices capable of publishing data to the cloud and devices/users willing to subscribe for one or more services in the cloud. We also assume a cloud that consists of an SDN controller with a data flow management algorithm, SDN switches,
and $N$ brokers, as shown in Figure 2.1. The data flow management algorithm dictates the performance of the system. In this paper, this algorithm is either the proposed or the baseline heuristic. Further, we assume $K$ indivisible topics in the system, each topic $i$ has $p_i$ subscribers.

First, the system run-time period is divided equally into $R$ configuration periods. Additionally, we define the system configuration $C$ during a configuration period as the assignment of topics to brokers as follows:

$$C = S_1, S_2, ..., S_N$$

(2.1)

Where $S_j$ is the set of topics served by broker $j$. The system configuration is fixed during a configuration period, which results in zero topic switching. Nonetheless, reconfiguration, which is the transition from one system configuration to another, introduces topic switching.

Second, we define the system load during a configuration period as the load difference between the overloaded and the underloaded brokers in the following equation:

$$L = \max(\{\sum_{i \in S_j} p_i : j = 1 \ldots N\}) - \min(\{\sum_{i \in S_j} p_i : j = 1 \ldots N\})$$

(2.2)

Where $L$ is the system load in number of subscribers, $\min$ and $\max$ are functions that return the minimum and maximum number of subscribers served by any of the brokers respectively.

Publish/subscriber messages sent from the smart city are intercepted by the SDN switches, which in turn uses Deep Packet Inspection (DPI) as a service to extract the topic and the type of message.
For publish messages, the extracted topic is hashed to a number then stored in a predefined field in the incoming message that flow rules can check. For example, this predefined field can be the Type of Service (TOS) field in the Internet Protocol (IP) header. Overwriting the TOS field does not conflict with the Quality of Service (QoS) since the message is at the penultimate hop before its delivery to the designated broker. For subscribe request messages, a value of zero is injected in a predefined field.

For publish messages, if a flow rule with a matching topic is found, the messages are routed using the broker IP address found in the action field of the flow rule. Otherwise, a packet-in message is sent from the SDN switch to the SDN controller. As a result, the heuristic (cf. Section VI) in the SDN controller sends a packet-out message for the new topic to the SDN switch with the appropriate broker IP address in the action field.

Subscribe request messages always have a zero value in their predefined field, which is injected by the DPI module. Therefore, the SDN switch sends a packet-in message to the SDN controller. The heuristic (cf. section VI) in the SDN controller notices a new subscribe request and extracts the topic number to update the number of subscribers per extracted topic. Next, the heuristic copies the topic number to the predefined field of the message and sends a packet-out message to the SDN switch. This packet-out message has an action list with the IP addresses of all brokers. In other words, for every new subscribe request, the heuristic updates its counters of subscribers per topic and sends a copy of this subscribe request to all brokers. This is important because when the heuristic decides to reassign a topic to a new broker, this new broker knows of all subscribers.

In the next section, we formulate the problem of minimizing the load difference between brokers within a reconfiguration limit as an ILP problem.
2.4. Problem Formulation

Finding the best system configuration that minimizes the load difference between brokers for each of the $R$ configuration periods is an NP-Hard problem. To prove it, we show that the partition problem, which is known to be NP-Hard [10], can be reduced to our problem.

Let $K = \{p_1, p_2, ..., p_k\}$ be a multiset (set with repeated items) of all topics’ sizes. Assuming a cluster of two brokers, we want to partition multiset $K$ into two subsets, $K_1$ and $K_2$. In addition, each subset is assigned to a broker, such that the sum of elements in $K_1$ is equal to the sum of elements in $K_2$. In other words, the load difference between the two brokers is zero, which is the definition of our problem. However, instead of having two subsets, there are $N$ subsets in our problem.

We define $a_{ij}$ as a selector with a value of one if topic $i$ is assigned to broker $j$ and zero otherwise. The optimization formulation of the problem is as follows:

$$\begin{align*}
\text{minimize} & \quad \sum_{j=1}^{N} \sum_{q=1}^{N} \left( \sum_{i=1}^{K} a_{ij} \cdot p_i - \sum_{i=1}^{K} a_{jq} \cdot p_i \right) \\
\text{s.t.} & \quad \sum_{j=1}^{N} a_{ij} = 1 \quad \forall i \in 1 \ldots K \\
& \quad a_{ij} \in \{0, 1\} \quad \forall i \in 1 \ldots K, \forall j \in 1 \ldots N
\end{align*}$$

Constraint (2.4) indicates that a topic can not exist in more than one subset, while constraint (2.5) emphasizes that the problem is an ILP problem (i.e., no fractional assignment of topics to subsets).

2.5. Dataset

Actual IoT brokers are available and used in the industry. However, their traffic traces are not publicly available. Therefore, we use vehicular traffic with the number of vehicles as
a proxy metric for the number of subscribers for smart city applications. In this work, we use the German city of Cologne’s realistic simulation scenario that describes the traffic in the city during a 23 hour period. This traffic is based on the traveling habits of the inhabitants [11].

The city of Cologne spans a $W \times H$ km$^2$ area. We divide the city into $K$ zones, and each zone is $W_z \times H_z$ km$^2$. Every zone represents topic $i$, and the number of vehicles moving within its boundaries is the number of subscribers $p_i$. The event of a vehicle entering the boundaries of another zone is interpreted as a new subscribe request for a topic. Therefore, by recording the number of vehicles per zone per time unit, we capture the number of subscribers per topic per time unit.

In our simulation experiments, vehicles’ details (e.g., location coordinates) are recorded every second. However, only a few cars can cross the borders of a zone within this period. Consequently, the variance in the number of subscribers per topic is very low, which does not simulate a publish/subscribe system for a smart city. Thus, we recorded the number of vehicles (i.e., subscribers) per zone (i.e., topic) every 60 seconds and noticed that the number of subscribers per topic is around hundreds of thousands.

While analyzing the dataset available in [12], we studied the evolution of topics over time and observed that topics change their size frequently and unpredictably, as shown in Figure 2.2(a). Additionally, we noticed that topics are much like elephant and mice flows in the Internet traffic [13]. The majority of topics are small in size (mice), while the minority of topics are large in size (elephants), as shown in Figure 2.2(b). In fact, we fit the probability distribution of topics based on the size as a Pareto distribution. This implies that small-sized topics have a power-law functional relationship with large-sized topics, as illustrated in Figure 2.2(b).

In the next section, we develop a long-term algorithm that exploits this observation.
2.6. Heuristic Solution

The current problem can be solved using a greedy heuristic that considers instantaneous statistics of topics. It reconfigures SDN switches every time step striving for optimal load distribution between brokers. Nevertheless, the greedy heuristic suffers from very high topic switching.

To provide a balance between the load difference between brokers and the number of reconfigurations, we implement a threshold-based topic balancing heuristic as a baseline heuristic. For every time step, the baseline heuristic computes the current brokers’ load difference $L_1$ using the assignment of topics to brokers used in the last step. Additionally, it sorts topics based on the size in descending order then iteratively assigns the next topic from the sorted list to the broker with the minimum number of subscribers (i.e., load). Using the new assignment of topic to brokers, $L_2$ is computed. Finally, the gain in load difference $L_1 - L_2$ is computed. If this gain is less than the threshold, then no topic switching is performed. Otherwise, if the gain is greater than the threshold, then the system is reconfigured using the new assignment of topics to brokers.

A low threshold value leads to a better load difference at the expense of more reconfiguration (i.e., high topic switching). While a high threshold value results in a high load difference but less reconfiguration (i.e., less topic switching), as illustrated in Figure 2.3. Finding the best threshold is difficult since it is an optimization problem. Furthermore, some threshold values are misleading. For example, Figure 2.4 shows that threshold values 200,000 and 800,000 have the same number of reconfigurations. However, the high threshold of 800,000 results in more than two times the load difference obtained using a low threshold of 200,000.

Instead of looking for the best threshold, the proposed heuristic divides the time interval into one or more configuration periods depending on the reconfiguration limit. Next, the
heuristic analyzes each configuration period to come up with a configuration that minimizes the load difference. A limit of one configuration period results in an optimal (i.e., zero) topic switching with a good load difference. By increasing the number of configuration periods, topic switching is increased while load difference is enhanced or decreased.

Algorithm 1 Long-Term Topic Balancing

1: Input: Number of subscribers per topic over time, $p_t^i$ for time range $(t_s \ldots t_e)$
2: Output: System configuration consisting of topics assignment to brokers
3: $\forall t \in \{t_s, \ldots, t_e\}$, compute $O_t$
4: $\forall i \in \{1, \ldots, K\}$, compute $P_i$
5: Sort set $P$, set of $P_i$’s, in descending order
6: Set $M = \min(K, N)$
7: $\forall j \in \{1, \ldots, M\}$, initialize broker $j$ with topic $j$ from set $P$ and compute $D_j$
8: $\forall j \in \{M + 1, \ldots, K\}$, assign topic $i$ to the broker with max($D_j$) then recompute $D_j$

Algorithm 1 presents the proposed heuristic. It performs the following steps for every configuration period in the time range $(t_s \ldots t_e)$:

- **Line 3**: Find the ideal broker load by distributing the number of subscribers evenly on all brokers to have a perfect load balance using the following:

  $$O_t = \frac{\sum_{i=1}^{K} p_t^i}{N} \quad \forall t = t_s \ldots t_e \quad (2.6)$$

- **Line 4**: For every topic, find the number of subscribers in a given sub-interval as shown below:

  $$P_i = \sum_{t=t_s}^{t_e} p_t^i \quad \forall i = 1 \ldots K \quad (2.7)$$

- **Line 5**: Sort the set $P$, set of $P_i$’s, in descending order so that large-size topics are fitted first.
• **Lines 6 – 7**: Distribute the first $N$ topics (or less) based on the order of $P$ over brokers (i.e., first topic according to the order of $P$ is assigned to the first broker, second topic to the second broker, and so on).

• **Line 8**: For the remaining topics, do the following:

  – Find the total number of subscribers per time step over the configuration period for every broker based on $S_j$, which is the set of topics assigned so far for broker $j$ as shown below:

  $\begin{align*}
  B^t_j &= \sum_{i \in S_j} p^t_i \quad \forall j = 1 \ldots N, \forall t = t_s \ldots t_e \\
  \end{align*}$

  $\quad\quad(2.8)$

  – Find the difference in the number of subscribers between the ideal broker load $(O_t)$ and broker $j$ for every time step of the configuration period as shown below:

  $\begin{align*}
  D^t_j &= O_t - B^t_j \quad \forall j = 1 \ldots N, \forall t = t_s \ldots t_e \\
  \end{align*}$

  $\quad\quad(2.9)$

  – Find the difference in the number of subscribers between the ideal broker load $(O_t)$ and broker $j$ over the overall configuration period as shown below:

  $\begin{align*}
  D_j &= \sum_{t = t_s}^{t_e} D^t_j \quad \forall j = 1 \ldots N \\
  \end{align*}$

  $\quad\quad(2.10)$

  – Select the broker with the maximum $D_j$ to assign the next topic in $P$. 

15
2.7. Results and Discussions

We conducted simulation experiments to evaluate the performance of the proposed heuristic against a baseline heuristic that utilizes a fixed threshold. Performance results indicate that, unlike the baseline heuristic, the proposed heuristic provides predictable and improved performance, as shown in Figure 2.5. The proposed heuristic guarantees an inversely proportional relationship between the number of reconfigurations (i.e., topic switching) and the load difference between brokers. On the other hand, a higher threshold for the baseline heuristic does not guarantee a lower number of reconfiguration, as shown in Figure 2.4, which leads to unpredictable results.

Moreover, increasing the length of configuration periods (i.e., reducing the number of reconfigurations) widens the performance gap between the two heuristics. For example, when using zero reconfigurations, the proposed heuristic outperforms the baseline heuristic with about 2000% fewer brokers’ load difference, which is clearly seen in Figure 2.5(a) and Figure 2.7. Also, Figure 2.5(b) shows the optimal (i.e., zero) topic switching of the proposed heuristic compared to hundreds of topic switching for the baseline heuristic.

While the proposed heuristic is not optimal, our simulation experiments indicate that it provides a better tradeoff between load difference and topic switching compared to the fixed threshold heuristic. That is, the baseline heuristic can obtain better load difference or topic switching but not both, as shown in Figure 2.6. Finding the best threshold for the baseline heuristic can cause it to outperform the proposed heuristic with respect to load difference but underperform with respect to topic switching and vice versa.
2.8. Conclusions and Future Work

In this paper, we address the problem of minimizing the load difference between brokers given a reconfiguration budget in support of data and decision fusion applications in the context of smart cities. We formulate this problem as an ILP problem and show it to be NP-Hard. Due to the complexity of the problem, a heuristic based on the long-term statistics of topics is proposed. The proposed heuristic and a threshold-based baseline heuristic are evaluated using real traffic traces. Results show that the proposed heuristic outperforms the baseline heuristic with more than 2000% better load distribution and at least 27% less topic-switching.

In the future, we plan to evaluate the proposed heuristic via large-scale experiments on a cluster of computers using actual IoT data. We also plan to explore the potential to design an online algorithm that provides worst-case performance guarantees.
Figure 2.1: System model showing the communication between the smart city and the cloud via SDN and publish/subscribe protocols.
Figure 2.2: Dataset characteristics.

(a) Number of subscribers per topic over time.

(b) Probability distribution of topics’ sizes.
Figure 2.3: Load difference and topic switching of the baseline heuristic for different threshold values.
Figure 2.4: Load difference of the baseline heuristic over time for two thresholds with both having one reconfiguration but different average load differences.
Figure 2.5: Average load difference and total topic switching of the proposed and the baseline heuristics.
Figure 2.6: Performance of the proposed heuristic using three reconfigurations limit and the baseline heuristics using two thresholds.
Figure 2.7: Brokers’ load difference over time of the proposed and the baseline heuristics with zero reconfigurations.
CHAPTER 3

Opportunistic Selection of Vehicular Data Brokers as Relay Nodes to the Cloud

The Internet of Things (IoT) revolution and the development of smart communities have resulted in increased demand for bandwidth due to the rise in network traffic. Instead of investing in expensive communications infrastructure, some researchers have proposed leveraging Vehicular Ad-Hoc Networks (VANETs) as the data communications infrastructure. However, VANETs are not cheap since they require the deployment of expensive Road Side Units (RSUs) across smart communities. In this research, we propose an infrastructure-less system that opportunistically utilizes vehicles to serve as Local Community Brokers (LCBs) that effectively substitute RSUs for managing communications between smart devices and the cloud in support of smart community applications. We propose an opportunistic algorithm that strives to select vehicles in order to maximize the LCBs’ service time. The proposed opportunistic algorithm utilizes an ensemble of online selection algorithms by running all of them together in passive mode and selecting the one that has performed the best in recent history. We evaluate our proposed algorithm using a dataset comprising real taxi traces from the city of Shanghai in China and compare our algorithm against a baseline of 9 Threshold Based Online (TBO) algorithms. A number of experiments are conducted and our results indicate that the proposed algorithm achieves up to 87% more service time with up to 10% fewer vehicles selected compared to the best-performing TBO algorithm.

3.1. Introduction

The widespread use of smart devices has led to the revolution of the Internet of Things (IoT). According to Gartner, 20 billion devices (things) will be connected by 2020 [14].
Many applications in the cloud use data generated by billions of devices for data analysis and decision making in different business sectors such as transportation, aviation, healthcare, and social networking [15]. Nevertheless, those billions of devices generate a huge amount of data that is increasing at an exponential rate\(^1\), which introduces communication and processing challenges [17].

To respond to these challenges, some researchers propose using Vehicular Ad-Hoc Networks (VANETs) to reduce the burden on the cloud by locally using the vehicles for communication as well as computation. VANETs utilize vehicles as computation and communication units to provide services to communities since vehicles travel across communities. Besides, vehicles are equipped with an On Board Unit (OBU), which is a device highly capable of processing and communication. In fact, it is expected that nearly 90 percent of vehicles by the year 2020 [18] will be equipped with an OBU.

Additionally, VANETs utilize a number of Road Side Units (RSU), which are road-side devices with computation and communication capabilities. An RSU can use 5G communication technologies to increase data rates. In spite of that, 5G technologies require dense deployment in small cell formation and suffer from great loss in penetrating buildings and obstacle blockages [16]. The deployment of RSUs also incurs additional costs and requires maintenance. As a result, this solution is not appropriate for communities with limited communications infrastructure. Furthermore, 5G technologies and their backhaul transport technologies are expensive [19].

To solve this problem, we envision a system where a community is divided into a number of zones. In each zone, we propose to use one vehicle to serve as a Local Community Broker (LCB). LCB uses the publish-subscribe paradigm [20] to manage the communications between the smart devices locally in the zone and those between smart devices across different zones.

\(^1\)As per one projection, data traffic per subscriber is increasing roughly at a rate of 50 percent annually [16].
zones through the cloud. A smart device can subscribe to a service provided by another smart device located in the same zone or in a different zone through the LCB. At any given time, one vehicle in each zone (if any exists) is selected to work as an LCB. The selection is made by the Smart Community Management Center (SCMC), which is hosted on the cloud.

In our proposed system, the SCMC represents a Cloud Service Provider (CSP), the LCB is a Cloud Service Broker (CSB), and smart devices are the Cloud Service Customers (CSC). A CSB plays the role of an intermediary between the CSP and the CSC, which has possible benefits from economic and security perspectives [21].

Selecting the best LCB is a challenge. When a new vehicle enters a zone, this vehicle is either selected or rejected by the SCMC to serve as an LCB. When a vehicle is rejected, it can not be selected later in the same zone. As a result, the SCMC may regret its decision of rejecting a vehicle, especially when new coming vehicles have shorter service times (i.e., time spent by a vehicle serving as an LCB). This problem is of an online nature and is similar to the online hiring problem were once an employee is hired or rejected for a job, the decision can not be reversed in the future (see Section 3.5).

While maximizing the total number of hired vehicles to serve as LCBs during the running period increases the service time in a zone (which is desirable since it provides the longest possible coverage possible), it also results in greater vehicle switching (which is costly since some incentive has to be used to convince vehicles to serve as LCBs while in the zone). Besides, subscribers need to be moved from the old LCB to the new one, which results in a delay. Consequently, the more the LCBs used, the more the cost. However, reducing the number of LCBs in a bid to reduce the cost minimizes the service time in the zone. Therefore, we propose an algorithm that utilizes an ensemble of Threshold Based Online (TBO) algorithms for LCB selection with the goal of maximizing service time while maintaining a reasonable number of vehicle switching.
The proposed algorithm opportunistically selects the best performing TBO algorithm from its ensemble based on the observed performance in terms of service time in recent history. The proposed algorithm chooses one algorithm from its ensemble to be active and sets all other algorithms to be passive. By doing so, the active algorithm alone selects the vehicle to be used as an LCB, while the performance of passive algorithms is used to choose the best performing algorithm in the ensemble by choosing the one that demonstrated the highest average service time in recent history (i.e., greedy approach). Consequently, different TBO algorithms might be utilized over time.

To the best of our knowledge, this is the first research effort that utilizes TBO algorithms for the selection of vehicles to serve as LCBs in support of smart community applications. Even though this work utilizes the proposed algorithm presented in our previous paper [22], the application domain is different in which the previous work focuses on minimizing the total delivery delay of messages using vehicles between the two zones while this work is focused on maximizing the vehicles’ service time in a given zone. Furthermore, this work has significant results and insights in this particular vertical application domain of using vehicles as data brokers when contrasted to our previous work that focuses on vehicular data ferrying.

3.2. Motivation

We studied the online hiring algorithms and found that some online hiring algorithms can be replaced by a TBO algorithm (see section 3.5), with a specific value. Therefore, we analyze the performance of the TBO algorithm in terms of service time using a variety of threshold values that range from low to high values. We found that the TBO algorithm performs better using some of these threshold values in one zone and worse in other zones. To understand this behavior in more depth, nine TBO algorithms are run (explained later in Section 3.8) with each having a different threshold value. For each of these algorithms,
we count the number of zones where the algorithm performance is the best compared with the other algorithms. We found that algorithms perform the best in some but not all of the zones in the area of high traffic volume, as indicated in Fig. 3.1a. Also, this figure shows that more than one algorithm performed the best in the same zone—in other words, in some zones, a number of algorithms may perform equally well. Moreover, focusing on one zone, we observe that for every period of time, one algorithm performs the best, as indicated in Fig. 3.1b. For that reason, we were inspired to develop an intelligent algorithm that studies the history of these nine TBO algorithms and switches to the algorithm that performs best in recent history. By doing so, the intelligent algorithm performs the best over time and across all zones.

3.3. Related Work

In recent years, there has been an increasing amount of literature on the use of vehicles as a communications infrastructure for smart communities. In this section, We will initially outline the contributions of these existing works, and then we will describe how these existing approaches related to our work and problem setting.

Alahmadi et al. [23] created a vehicular cloud network model in which a number of vehicles close to a traffic light form a non-permanent vehicular cloud by gathering the clustered computational resources. The researchers aim to reduce the processing and network power utilized in the data center.

Aloqaily et al. [24] introduced a method in which vehicular services in smart cities are provided continuously by using the concept of smart vehicle as a service (SVaaS). Such services include: sensing, storing, computing, infotainment, and/or mobilizing. To provide continuous services, the researchers predicted the location of vehicles in order to prepare the demanded services ahead of time.
(a) Performance of algorithms across zones in terms of service time. *No single algorithm performs the best in all zones in the area of high traffic volume.*

(b) Performance of algorithms in one zone over time in terms of service time. *No single algorithm performs the best over time.*

Figure 3.1: Observations on the performance of algorithms on single and multiple zones.
Hou et al. [17] state that solutions to computation and communication challenges in vehicular applications like RSU, cellular network, and mobile cloud computing are not applicable due to dependency on existing infrastructure and high cost. The researchers proposed an architecture named Vehicular Fog Computing (VFC) that utilizes a number of collaborating vehicles to perform computation and communication tasks. In another related work [15], the authors indicate that in some cases, such as peak hours, the computation capacity of fog is overloaded by an increasing number of requests. The authors propose to use VFC for supporting fog computing with more storage and computation power and focuses on the use case of parked cars.

Want et al. [25] propose using the Internet of Vehicles (IoV) as a fog consisting of a number of vehicles close to RSU for executing real-time traffic management in order to reduce the average response time per vehicle’s report. The authors formulate and solve the offloading operation as an optimization problem.

Cao et al. [26] claim that cloud-based solutions have many issues such as network bandwidth bottlenecks, high delay, and low Quality of Experience (QoE). They investigated the use of IoV in edge computing and proposed a strategy for users to pick vehicles to achieve maximum QoE.

In [27], the authors claim that some services such as those related to information about local weather and traffic depend on time and geographical location and are thus readily available through the Internet. To make such information available, the authors proposed using a VANET without the dependency on existing infrastructure for retaining this kind of information. Also, they developed a method in which data transmission probability is determined by vehicles depending on the density of data retention in near vehicles.

All the aforementioned works have one or more of the following failings (which makes them ill-suited for our problem setting):
1. they depend on clusters of vehicles near a traffic light;

2. they depend on parked vehicles;

3. they provide specific services to vehicles and not to the cloud in general;

4. they depend on users for picking service vehicle. Other infrastructure-based projects are also not appealing due to the high cost of commissioning the infrastructure.

In addition, most of the existing works do not approach the problem from an online perspective. Our work is distinctive in that our proposal assumes no infrastructural support and works in an online fashion to hire a vehicle as a local community broker (LCB) regardless of geographical constraints.

3.4. System Model

In this paper, we assume that for every vehicle, the time spent in a zone is estimated by the SCMC. Our assumption is based on many research efforts that appeared in the recent literature [28]–[30] to predict this parameter.

In our proposed system, to use vehicles as LCBs in a given area, we divide the area of interest (i.e., city) into $Z$ zones. A number of smart devices (i.e., sensors and actuators) exist in every zone. Smart devices subscribe to services provided by other smart devices in the same zone or in a different zone through the LCB. The LCB manages publish/subscribe requests between the smart devices through the SCMC in the cloud, as illustrated in Fig. 3.2. In addition, the SCMC controls the selection of LCBs in all zones. Once an LCB is selected in a zone, subscribers (i.e., smart devices) are moved from the previous LCB to the newly selected one.

The SCMC is responsible for the following tasks:
Figure 3.2: Publish-subscribe model between the smart devices and the SCMC through the LCBs.

- Detection of zone changes of vehicles;
- Computation of the estimated service time of vehicles;
- Running of the proposed algorithm in the cloud.

To detect a zone change in a vehicle, the SCMC computes the current zone number at time $t$ and compares it with the zone number at time $t - \delta$. Consequently, if the two
values are different, then the vehicle has entered a different zone at time \( t \). The method for computing the zone number is later detailed in Section 3.8.1.

When vehicle \( v_i \) enters a zone, the SCMC computes the estimated service time \( s_i \) then uses the proposed algorithm to decide on the selection of \( v_i \) as the LCB of the zone.

Now, if vehicle \( v_j \) is working as an LCB in the zone and the proposed algorithm decides to select \( v_i \) as the new LCB, then the SCMC stops \( v_j \) from working as LCB and moves all subscribers to \( v_i \). Switching between the LCBs results in a delay as subscribers are moved from the old LCB to the new one. Also, an incentive is used with every new LCB, as explained earlier.

The proposed algorithm utilizes an ensemble of TBO algorithms, as explained in Section 3.6. Once a vehicle is selected as an LCB, all smart devices in the zone either publish their data or subscribe for services through the LCB.

### 3.5. Online Hiring Algorithms

The problem of finding the best candidate is studied first in the classical online secretary problem, where \( M \) candidates are interviewed in random order with the goal of maximizing the probability of finding the best candidate. Later different variations of the secretary problem were proposed.

Recently, the problem has been revisited and studied using hiring algorithms, which have been shown to be effective in selecting the best candidates by many companies. In fact, instead of selecting employees manually, many companies like Google prefer to use hiring algorithms for automating this task in order to save time and efforts [31], [32]. There are different strategies of the hiring algorithm such as TBO, ‘hire above minimum or maximum’,...
and ‘Lake Wobegon’\textsuperscript{2} (‘hire above mean or median’) [33].

In this paper, we investigate the use of a TBO algorithm using different threshold values. This algorithm is based on a fixed threshold $\tau$. In this algorithm, vehicle $i$ is selected only if its estimated service time $s_i$ is greater than $\tau$.

### 3.6. Proposed Heuristic Solution

In this paper, we propose an algorithm that strives to maximize the total service time of a zone using vehicles as LCBs. The idea is simply to run an ensemble of $N$ TBO 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 broker, 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 is capable of analyzing the history of all TBO algorithms in its ensemble in terms of service time. Additionally, to secure the best performance, the proposed algorithm may switch to another algorithm in its ensemble every $D$, where $D$ is a constant number of time units. When the SCMC detects a zone change in a vehicle (i.e., the vehicle enters a different zone), it runs the proposed algorithm. The status of the proposed algorithm that is maintained by the SCMC, comprises:

1. Active algorithm;

2. Cumulative service time for each of the $N$ algorithms;

3. The vehicle selected as the data broker (if any);

\textsuperscript{2}Lake Wobegon refers to a fictional town conceived by the author Garrison Keillor, where “all the women are strong, all the men are good looking, and all the children are above average.”
4. Service time of the selected vehicle (if any);

5. The remaining time until changing the active algorithm is explored.

**Algorithm 2** Algorithm for selecting the best LCB (in terms of the service time)

**Input:** vehicle service time $S$.

**Initialization:**
1: Set all algorithms as passive
2: Set active = select an algorithm number randomly.

**Executed when zone change is detected:**
3: for each of the nine algorithms do
4: Run the algorithm
5: if Decision is accept then
6: Add $S$ to the cumulative service time.
7: if this algorithm number = active then
8: Accept the vehicle
9: end if
10: else if this algorithm number = active then
11: Reject the vehicle
12: end if
13: end for

**Executed every $D$ minutes:**
14: Set best = algorithm number that has the maximum cumulative service time
15: if active $\neq$ best then
16: Set active = best
17: end if
18: Set cumulative service time for every algorithm to zero.

**Output:** decision (accept or reject)

Algorithm (2) shows the three parts of the proposed algorithm. In the first part, one of the algorithms in the ensemble is selected randomly to serve as the active algorithm, and all other algorithms in the ensemble are set to passive. In the second part, all the algorithms in the ensemble are executed, and the cumulative serve time is updated accordingly. Moreover, the decision made by the active algorithm is committed, while decisions of other algorithms are ignored. The last part is only executed after $D$ time units have passed. Furthermore,
the algorithm with the maximum cumulative service time is set as the active algorithm while setting the other algorithms as passive. In other words, every $D$ time units, the proposed algorithm compares all the algorithms in terms of cumulative service time and switches to another algorithm in case the currently active algorithm is not the best. The switching behavior of the proposed algorithm is shown in Fig. 3.3.

3.7. Illustrative Example

In this section, we discuss the proposed algorithm by utilizing two TBO algorithms (TBO-10 and TBO-20). We assume that at $t_0$, TBO-20 is randomly selected by the proposed algorithm (i.e., TBO-20 is active) while TBO-10 is initialized in passive mode. The performance in terms of average service time for the two algorithms is recorded every $D$ minutes (see Algorithm 2), as shown in Table 3.1. The proposed algorithm starts with a weak
performance initially at time $t_1$, but then the performance enhances over time, and by time
$t_4$, the performance of the proposed algorithm surpasses that of the two TBO algorithms.
In $t_1$, the proposed algorithm detects that TBO-10 is performing better and thus switches
to TBO-10 (set TBO-10 as active and TBO-20 as passive). However, in $t_3$, the proposed
algorithm switches back to TBO-20 since it starts performing better than TBO-10.

Table 3.1: Performance of the proposed algorithm compared with two TBO
algorithms.

<table>
<thead>
<tr>
<th>Time</th>
<th>TBO-10</th>
<th>TBO-20</th>
<th>Proposed</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average Service Time</td>
<td>Average Service Time</td>
<td>Average Service Time</td>
<td></td>
</tr>
<tr>
<td>$t_0$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>TBO-20</td>
</tr>
<tr>
<td>$t_1$</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>TBO-10</td>
</tr>
<tr>
<td>$t_2$</td>
<td>20</td>
<td>12</td>
<td>10</td>
<td>TBO-10</td>
</tr>
<tr>
<td>$t_3$</td>
<td>28</td>
<td>22</td>
<td>26</td>
<td>TBO-20</td>
</tr>
<tr>
<td>$t_4$</td>
<td>32</td>
<td>32</td>
<td>36</td>
<td>TBO-20</td>
</tr>
</tbody>
</table>

3.8. Experimental Results

In this section, we describe the dataset used in our experiment, explain the experiments’
settings, and evaluate the performance of the proposed algorithm by comparing it with
nine baseline ‘hire above a threshold’ online algorithms using real vehicular traces of the
Shanghai dataset. Finally, we discuss the results and present the insights learned from our
experiments.
3.8.1. Dataset and Experimental Settings

In this paper, we conducted our experiments using the Shanghai dataset, which consists of taxi traces observed in the city of Shanghai in China. To trace their positions, every taxi vehicle is equipped with a GPS unit. Additionally, each taxi has a GPRS wireless communication modem, which is used for sending GPS location along with other information to a data center. This dataset was collected in 2007 by monitoring 2,109 taxis. Moreover, the information sent by the taxis to the data center includes the taxi ID, the timestamp, the longitude and latitude, the speed, and the heading direction [34].

To divide the city into zones, we encoded the longitude and latitude, which represents the geographical location, into a string of seven characters using the GeoHashing method [35]. Every string produced using the GeoHashing method represents a zone in the city. Additionally, a string of seven characters encoding divides the globe into a number of zones, each of 153 by 153 meters, which is within the communication coverage of vehicles (i.e., RSU). Next, we convert the zone string to a number using hashing. Also, we filter the dataset by removing zones that have no traffic activity.

The used dataset is only one day long, while the proposed algorithm needs more time to start working efficiently. Consequently, to have datasets with longer periods, we replicate the one-day dataset to create 5, 10, 15, 20, and 25 days datasets.

To study the effect of different traffic scenarios on the performance of the proposed algorithm, we divide the city based on the traffic volume into three areas: low, medium, and high traffic areas. To compute the traffic volume per zone, the average number of vehicles per zone along with the standard deviation is computed. We noticed that the standard deviation is greater than the average. Therefore, based on $N_z$, the number of vehicles in zone $z$, each zone is categorized as follows:
• **Light traffic zone**: \( N_z < \text{average} \)

• **Medium traffic zone**: \( \text{average} \leq N_z \leq \text{standard deviation} \)

• **High traffic zone**: \( N_z > \text{standard deviation} \)

To test the capabilities of the proposed algorithm, we set the proposed algorithm to utilize nine TBO algorithms. We use the terminology \( TBO-x \)—where \( x \) is the threshold value as a percentile of the service times in a zone. For example, TBO-10 refers to the TBO algorithm with a threshold value of 10 percentile; the threshold value of TBO-20 is set to the value of the 20\(^{th}\) percentile of \( L \), and so on for the other seven TBO algorithms. First, list \( L \) is constructed from estimated service times of all coming vehicles in a zone. Then \( L \) is sorted, and threshold values of every TBO-x online algorithm is set as the \( x \)-th percentile of \( L \).

It can be noticed that threshold values of 0\(^{th}\) and 100\(^{th}\) percentiles are never used. We compare the TBO-0 online algorithm with a threshold value of 0\(^{th}\) percentile and TBO-10. TBO-0 has less average service time per zone and three times more average vehicle selections per zone than TBO-10. Thus, the 0\(^{th}\) percentile threshold value is not worth to be considered. As for using the 100\(^{th}\) percentile threshold value, the results in terms of the average service time per zone are too small to be considered. Finally, we set \( D \) to 6 hours in all of the experiments to be consistent.

### 3.8.2. Results Discussion

To evaluate the performance of the proposed algorithm compared with the nine TBO algorithms, we run the algorithms in different experiments. Also, we observed that the TBO-10 online algorithm is performing better than the other eight TBO algorithms in terms of service time. Consequently, the proposed algorithm is compared with the TBO-10 online
algorithm in terms of the average service time per zone and the average number of selections per zone, as indicated in Table 3.2.

Table 3.2: Performance of the Proposed Algorithm Compared to TBO-10 Online Algorithm In Areas of High Traffic Volume

<table>
<thead>
<tr>
<th>Selection Budget</th>
<th>Improvement Percentage in Service Time</th>
<th>Improvement Percentage in Number of Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5 days</td>
<td>10 days</td>
</tr>
<tr>
<td>8 per day</td>
<td>+62.77</td>
<td>+78.05</td>
</tr>
<tr>
<td>17 per day</td>
<td>+29.86</td>
<td>+38.07</td>
</tr>
<tr>
<td>35 per day</td>
<td>+12.59</td>
<td>+17.97</td>
</tr>
<tr>
<td>70 per day</td>
<td>+5.36</td>
<td>+9.45</td>
</tr>
<tr>
<td>Open</td>
<td>+1.67</td>
<td>+5.18</td>
</tr>
</tbody>
</table>

3.8.2.1. Measuring the performance of the proposed algorithms for different traffic volumes

The city is divided based on traffic volume into three areas: low traffic area, medium traffic area, and high traffic area. Results are recorded in terms of the average service time per zone and the average number of selections per zone.

We found that the proposed algorithm is not as competitive in low and medium traffic areas as in high traffic areas, as shown in Fig. 3.4. This is because the areas of low and medium traffic have fewer vehicles and longer periods with no vehicles. Therefore, we focus our discussion on high traffic area results.

3.8.2.2. Measuring the performance of the proposed algorithms for different selection budgets and time intervals

We test the algorithms for a different number of days using different selection budgets. A selection budget represents the maximum number of vehicles hired by an algorithm. Selection budgets are set based on different fractions of the average number of
Figure 3.4: Performance of algorithms for different traffic volumes for five days results. The proposed algorithm delivers more service time than other algorithms in a high-traffic area.
vehicles per zone, which is 35 for the used dataset. In each experiment, results are recorded per city area.

Table 3.2 shows that the proposed algorithm performs better in terms of average service time in all selection budget settings. Also, the proposed algorithm collects more profit in terms of service time and reduces the cost in terms of the number of selections in scenarios with more selection budget. Moreover, the proposed algorithm collects more profit in scenarios with a limited selection budget at the cost of enhanced switching between the vehicular data brokers. As more selection budget is available, the proposed algorithm becomes more inclined towards optimizing cost, which provides a balance between cost and profit.

3.9. Conclusions and Future Work

In this paper, the problem of selecting vehicles to serve a data broker in support of smart community applications is considered. The selection process strives to achieve the maximum service time. An algorithm is proposed that utilizes an ensemble of TBO algorithms by running them altogether in passive mode and selecting the one that performs 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 nine TBO algorithms. Experiments with these traces demonstrate that the proposed algorithm outperforms TBO algorithms presented in the literature in high traffic volume regardless of the selection budget by performing better on the service time. Also, the proposed algorithm reduces the number of selections with high selection budgets. 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

Exploration and Exploitation in Federated Learning to Exclude Clients with Poisoned Data

Federated Learning (FL) is one of the hot research topics, and it utilizes Machine Learning (ML) in a distributed manner without directly accessing private data on clients. However, FL faces many challenges, including the difficulty to obtain high accuracy, high communication cost between clients and the server, and security attacks related to adversarial ML. To tackle these three challenges, we propose an FL algorithm inspired by evolutionary techniques. The proposed algorithm groups clients randomly in many clusters, each with a model selected randomly to explore the performance of different models. The clusters are then trained in a repetitive process where the worst performing cluster is removed in each iteration until one cluster remains. In each iteration, some clients are expelled from clusters either due to using poisoned data or low performance. The surviving clients are exploited in the next iteration. The remaining cluster with surviving clients is then used for training the best FL model (i.e., remaining FL model). Communication cost is reduced since fewer clients are used in the final training of the FL model. To evaluate the performance of the proposed algorithm, we conduct a number of experiments using FEMNIST dataset and compare the result against the random FL algorithm. The experimental results show that the proposed algorithm outperforms the baseline algorithm in terms of accuracy, communication cost, and security.
4.1. Introduction

Recently Federated learning (FL) has been proposed as an emerging approach to build Machine Learning (ML) models across multiple decentralized edge devices [36]. This helps to overcome the challenge of privacy preservation by keeping all the training data on the device, decoupling the ability to do ML from the need to store the data in the cloud. However, several challenges need to be considered for the implementation of FL, including communication cost between the servers and clients, the accuracy of the model, and security.

FL is vulnerable to security attacks whereby a group of malicious clients could harm the performance of the model by carrying out a poisoning attack [37]. These attacks may cause the model to fail and converge to biased models that do not accurately represent the data. Applying anti-poisoning techniques might lead to the discrimination of minority groups whose data are significantly and legitimately different from those of the majority of clients [38]. In addition, detection and identification of unauthorized IoT devices are very important, especially with the increase in the number of attacks on IoT devices.

In this paper, to cope with the FL challenges, we propose an FL framework inspired by evolutionary techniques consisting of three stages. In the first stage, participating clients are grouped randomly into a number of clusters. Subsequently, a random model is selected for each cluster. In the second stage, models are explored, and the best performing cluster, in terms of classification accuracy, is selected in a repetitive process. In each iteration, all the clusters are trained in parallel, and the worst performing cluster is removed from the process till one cluster remains. Additionally, in each iteration, several clients may be expelled from each cluster either due to their low performance compared with other clients in the cluster or their data being poisoned. The remaining clients of removed clusters are exploited by joining the best-performing cluster. In the third stage, the best performing cluster is utilized
such that FL is trained with the cluster’s model using the remaining clients in that cluster.

The salient contributions of this paper are:

- Optimize the performance of FL in terms of accuracy by exploring a number of clusters, each with a different model, to select the best performing model (i.e., cluster).

- Optimize the security of FL by identifying clients with poisoned data and expel them from every cluster using cosine similarity while surviving clients are exploited in every iteration.

- Optimize the communication cost during the training process by expelling clients with either poisoned or weak data. Thus fewer clients participate in the training process leading to less communication.

The organization of the remainder of the paper is as follows. Related literature is reviewed in Section 4.2. The system model is described in Section 4.3. Section 4.4 discusses the proposed algorithm. The used dataset and conducted experiments are explained in Section 4.5. A discussion of the results and the salient lessons learned are provided in Section 4.6. Finally, the paper is concluded in Section 4.7 by summarizing the work and identifying future research directions.

4.2. Related Work

Recently, a significant amount of work has been done in the area of FL. This section reviews recent related works on the different aspects of the work, including algorithm optimization and poisoning attacks against FL. The reviewed papers focus on optimizing the algorithm used in FL to gain more accuracy, reduce communication between the clients and the server, or enhance security. To the best of our knowledge, this work is the first
Table 4.1: An overview of the related work.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Year</th>
<th>Target/Focus</th>
<th>Description of the Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Accuracy</strong></td>
</tr>
<tr>
<td>[39]</td>
<td>2020</td>
<td>✓</td>
<td>Proposes a heuristic approach to optimize the accuracy in stateful FL with a budgeted number of candidate clients by selecting the best candidate clients in terms of test accuracy to participate in the training process.</td>
</tr>
<tr>
<td>[40]</td>
<td>2020</td>
<td>✓</td>
<td>Relies on an active learning scheme to make use of unlabelled data available at each client for improving the FL model’s accuracy.</td>
</tr>
<tr>
<td>[41]</td>
<td>2020</td>
<td>✓</td>
<td>Proposes a PSO-based technique to optimize the hyperparameter settings for the local ML models in an FL environment.</td>
</tr>
<tr>
<td>[42]</td>
<td>2019</td>
<td>✓</td>
<td>proposes a fusion scheme by aggregating the features from both the local and global models to address the problem of heavy communication round cost when the local data is distributed in a Non-IID way.</td>
</tr>
<tr>
<td>[43]</td>
<td>2020</td>
<td>✓</td>
<td>Relies on a clustering mechanism to exploit geometric properties of the FL loss surface to group the client population into clusters with jointly trainable data distributions.</td>
</tr>
<tr>
<td>[38]</td>
<td>2020</td>
<td>✓</td>
<td>Proposes two different approaches. The first one is based on micro aggregation where clients who identify themselves as belonging to a minority group announce some relevant attributes to their peers. The second approach is based on Gaussian mixture models to characterize the distribution of the client-provided updates.</td>
</tr>
<tr>
<td>[44]</td>
<td>2021</td>
<td>✓</td>
<td>Relies on an SVM model for data vetting process to mitigate data poisoning attacks in an FL setting.</td>
</tr>
<tr>
<td>[45]</td>
<td>2021</td>
<td>✓</td>
<td>Relies on a poison data generation method to eliminate the conventional attacking assumption that the attacker already owns a proportion of other participants’ training data.</td>
</tr>
<tr>
<td>[46]</td>
<td>2020</td>
<td>✓</td>
<td>Proposed a scheme that identifies benign local models by solving a maximum clique problem, and poisoned local models are ignored during global model updating.</td>
</tr>
<tr>
<td><strong>This Work</strong></td>
<td>2021</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
attempt that optimizes FL model accuracy, reduces the number of client-server communication rounds, and enhances the security of FL models. We describe and compare the most relevant previous works with our work in Table 4.1.

4.2.1. Algorithm Optimization

Several interesting optimization techniques have been proposed to deal with the challenges associated with FL, including learning an ML model in an FL environment, unbalanced distribution of local data, and reducing the generated traffic in the network.

Mohammed et al. [39] proposed a stateful FL heuristic algorithm to solve the problem of optimizing accuracy in stateful FL with a budgeted number of candidate clients by selecting the best candidate clients in terms of test accuracy to participate in the training process. Ahmed et al. [40] tried to improve the accuracy of the FL model by employing unlabeled data available at each client through an active learning scheme. Qolomany et al. [41] proposed a Particle Swarm Optimization (PSO)-based technique to optimize the hyperparameter settings for the local ML models in an FL environment. They evaluated and compared the proposed approach with the grid search technique. They found that the number of communication rounds used by their proposed approach is two orders of magnitude less than the grid search method. To address the issue of local clients’ data distributions diverge, Sattler et al. [43] proposed clustered multitask FL framework, which exploits geometric properties of the FL loss surface to group the client population into clusters with jointly trainable data distributions. They found that the cosine similarity between the weight-updates of different clients is highly indicative of the similarity of their data distributions. Yao et al. [42] proposed a feature fusion method by aggregating the features from both the local and global models to address the problem of high communication round cost when the local data is distributed in a Non-IID way. Yao and Sun [47] proposed a local continual training strategy
to address the problem of weight divergence of ML model in FL environment by evaluating the important weight matrix on a small proxy dataset on the central server and then used to constrain the local training.

4.2.2. Poisoning Attacks Against Federated Learning

The communication protocol amongst different nodes in the FL environment could be exploited by attackers to launch data poisoning attacks, which has been demonstrated as a big threat to most ML models. To improve the robustness of real-world ML systems, it is critical to study how well these models perform under poisoning attacks.

To this aim, Singh et al. [38] proposed two approaches to distinguish malicious behaviors of a node from legitimate ones in FL. The first approach is based on micro aggregation; with this approach, clients who identify themselves as belonging to a minority group announce some relevant attributes to their peers, such as gender, sexual orientation, or ethnicity. In contrast, the second approach is based on Gaussian mixture models to characterize the distribution of the client-provided updates. Doku and Rawat [44] proposed an approach based on an SVM model for the data vetting process to mitigate data poisoning attacks in an FL setting. They introduced the concept of a facilitator that gets assigned to an end device. The facilitator’s job is to ensure the data that an end device owns has not been compromised. Zhang et al. [45] proposed a poisoning attack model based on generative adversarial networks to explore an active and powerful attack model, poisoning attacks, in FL-aided IoT systems. They designed a poison data generation method to eliminate the conventional attacking assumption that the attacker already owns a proportion of other participants’ training data. Cao et al. [48] proposed a scheme, Sniper, to eliminate poisoned local models from malicious participants during training. Sniper identifies benign local models by solving a maximum clique problem, and poisoned local models will be ignored during global model updating.
They analyzed how the number of poisoned samples and the number of attackers as variables affecting the performance of distributed poisoning attacks. They observed that the attack success rate increases linearly with the number of poisoned samples. The attack success rate increases with the number of attackers when the number of poisoned samples is unchanged, and the increasing speed becomes faster when more attackers are involved. Liu et al. [46] proposed a blockchain-based secure FL framework to address data privacy leakage issues related to ensuring secure FL in 5G networks. They used smart contracts in blockchain to validate the model updates against poisoning attacks automatically. They also introduced the local differential privacy technique in smart contracts to prevent membership inference attacks.

4.3. System Model

We assume one server, $N$ clients, and $C$ clusters such that there are $N/C$ clients per cluster except for the last cluster, which may have fewer clients, as shown in Fig. 4.1. We also assume a training budget of $R$ rounds. There are three stages for the system to find and train the best model. In the first stage, the server randomly assigns clients to $C$ clusters and selects a random model for each cluster. In other words, the server assigns the same model to all clients of the same cluster. Additionally, the server has a small unlabeled dataset used for testing models during the training process to determine the performance of models and thus determine the best and worst-performing models. The use of an unlabeled dataset rather than a labeled one ensures the privacy of data on the server. In the second stage, the system runs $C - 1$ iterations to explore models. In each iteration, clients in each cluster are engaged with the server for a number of communication rounds to train the global model of that specific cluster. By the end of each iteration, two actions take place. First, some clients (depending on the value of $X$ as explained later) are expelled from each cluster due
to a poisoned or poor dataset. Second, the best and worst-performing models (i.e., clusters) are determined, the worst cluster is removed, and its clients are exploited and assigned to the best performing cluster. By the end of the last iteration, only one cluster remains with $M \leq N$ clients, as shown in Fig. 4.1. In the third stage, clients in the remaining cluster engage with the server for a number of iterations to train the remaining global model.

### 4.4. Proposed Algorithm

The proposed algorithm consists of three stages. In the first stage (Algorithm 3, lines 1 through 7), $C$ clusters are formed, and each cluster hosts about $N/C$ clients and uses a model selected randomly. We created a pool of random models to select from. Each model in the pool is created with three parameters: a number of convolutional layers (1 or 2), filters (64, 128, 196, or 256), and kernel size (3x3, 3x5, 5x3, or 5x5). There is an input layer with 28x28 size. Also, a max-pool layer is used after every convolutional layer. The last three layers are one flatten layer and two dense layers.

In the second stage (Algorithm 3, lines 8 through 27), models are explored, and the best performing model (i.e., cluster) is selected in a repetitive process with $C - 1$ iterations. In each iteration, all clients in all clusters are trained in parallel for $R_c$ communication rounds, then $X\%$ clients are expelled from each cluster due to low performance or poisoned data. Next, the cluster with the most deficient performance is deleted, and its clients are exploited by joining the highest performing cluster. This process continues until only one cluster is left. In the third stage (Algorithm 3, lines 28 through 34), clients in the remaining cluster are trained for $R/2$ communication rounds, and the trained model is returned.

We can summarize the proposed algorithm (Algorithm 3) as follows:

- Lines 1 through 5: initialize a random model and select $N/C$ clients randomly for each of the $C$ clusters.
Figure 4.1: An illustration of the system model. The cloud server running the proposed algorithm communicates with the local servers in different clusters to train a number of global models in stage 2, then the best model is trained in stage 3.
**Algorithm 3 Proposed heuristic**

**Input:** \( N \): number of clients, \( C \): number of clusters, \( R \): number of communication rounds, \( E \): number of epochs, \( R_c \): number of communication rounds per cluster, \( X \): percentage of expelled clients per cluster per phase

**Output:** Trained global model

// Server initialization

1: for \( c = 1 \) to \( C \) do
2:   Pick random model \( M_c \) for cluster \( c \)
3:   Initialize the global model \( M_c \)
4:   Pick \( N_c \leq N/C \) clients for cluster \( c \)
5: end for
6: Set \( E_{\text{stage2}} = (R/2) \times E \)
7: Set \( E_c = E_{\text{stage2}} / (C - 1) / R_c \)

// Find the best model in terms of predicted labels

8: while \( C > 1 \) do
9:   for \( c = 1 \) to \( C \) do
10:      for \( r = 1 \) to \( R_c \) do
11:         Server sends global model \( M_c \) to all clients in \( c \)
12:         Clients train the global model on the local dataset with \( E_c \) epochs and return updated model
13:         Server average aggregated model parameters from clients
14:      end for
15:      Predict labels of the unlabeled dataset using the global model on the server
16:      Predict labels of the unlabeled dataset for every returned clients’ model on the server
17:      Compute cosine similarity for every returned clients’ models
18:      Exclude \( X\% \) of clients with the lowest cosine similarity from cluster \( c \)
19:   end for
20: Compute the average of predicted labels using global models of all clusters
21: Compute cosine similarity for predicted labels of every cluster against the average predicted labels of all clusters
22: Find \( c_h \) the cluster with the highest cosine similarity
23: Find \( c_l \) the cluster with the lowest cosine similarity
24: Move all clients from \( c_l \) to \( c_h \)
25: Delete cluster \( c_l \)
26: Set \( C = C - 1 \)
27: end while

// Train the last remaining cluster

28: Set \( c = \) last remaining cluster number
29: for \( r = 1 \) to \( R/2 \) do
30:   Server sends global model \( M_c \) to all clients in \( c \)
31:   Clients train the global model on the local dataset with \( E \) epochs and return updated model
32:   Server average aggregated model parameters from clients
33: end for
34: Return the trained global model
• Lines 6 through 7: set training parameters per cluster

• Lines 8 through 27: find the best model by running $C - 1$ phases. In each phase, clients of all available clusters are trained in parallel. Clients are evaluated in every cluster based on the cosine similarity of predicted labels of the global model against predicted labels of every client’s model using an unlabeled dataset in the server. Then, $X\%$ clients with the lowest cosine similarity are expelled from every cluster. The low performance of expelled clients is either related to poor data or poisoned data. Finally, the clusters are evaluated based on their average predicted labels, and the cluster with the lowest cosine similarity is deleted with its clients moved to the cluster with the highest cosine similarity.

• Lines 28 through 34: train the remaining clusters and return the trained model.

4.5. Experimental Settings

4.5.1. Dataset

We use the Federated Extended MNIST, FEMNIST [49], for classifications tasks. The FEMNIST is used for both letters and digits (A-Z, a-z, and 0-9), and it has 244,154 images for training and 61500 for testing. We use this dataset under non-i.i.d data distribution, where the FEMNIST dataset is first split into 62 partitions (number of labels). Then each of the 900 users is assigned batches of two classes only.

For adversarial versions: we applied the Fast Gradient Sign Method (FGSM) proposed by Goodfellow [50]. FGSM is widely used to produce adversarial examples. The original input is manipulated by adding or subtracting a small error of $\epsilon$ to each data sample. $\epsilon$ is a small number controlling the size of the adversarial attack to be effective. Any addition or
subtraction of the $\epsilon$ depends on the gradient sign for any given input that is either positive or negative. Adding errors in the gradient direction means that classification is intentionally altered so that the model classification fails.

We divided the training dataset over $N$ clients in all experiments and used a random unlabeled number of records from the test dataset on the server, which is used for evaluating clients’ local models and global cluster models.

4.5.2. Experiments

To evaluate the performance of the proposed algorithm, we compared the results of the proposed algorithm against the random FL algorithm (baseline algorithm) [39]. To ensure a fair comparison, we use the same number of epochs for both algorithms. We set $R$, the number of communication rounds to 32, and the number of epochs $E$ to 8 for all experiments of the baseline algorithm, so every client uses a total of 256 (8 x 32) epochs. We used less than or equal number of epochs ($\leq 256$) with all experiments of the proposed algorithm.

For example, assume that we have 8 clusters. Since there are 256 total epochs available for every client in the baseline algorithm, we use half of it or less in the second stage of the proposed algorithm and use the other half in the third stage of the proposed algorithm. To do that, we first compute the total number of epochs available for the second stage of the proposed algorithm as shown in equation (4.1), which is 128.

$$E_{stage2} = \frac{R}{2}E$$  \hspace{1cm} (4.1)

Since we have $C - 1$ iterations, we need to compute the number of epochs per client in each iteration, and the total number of epochs per client for the $C - 1$ iterations must be $\leq 128$. Assuming that $R_c$, the number of communication rounds per cluster is 4, we compute the total number of epochs used by every client per iteration in the second stage of the
proposed algorithm as shown in equation (4.2). This number is 4.57, so we round down the number to 4 epochs. Consequently, the second stage of the proposed algorithm uses only 112 epochs (four epochs times four rounds times seven iterations) and not 128.

\[ E_c = \frac{E_{\text{stage2}}}{(C - 1)R_c} \]  

(4.2)

In the third stage of the proposed algorithm, we use \( R/2 \) times \( E \) epochs, which is 128. Thus, the proposed algorithm is actually using less number of epochs, 240 in this example, compared to the baseline algorithm. In other words, the proposed algorithm consumes fewer computing resources compared to the baseline algorithm. In general, the proposed algorithm uses \( \leq \) epochs compared to the baseline algorithm.

We conducted 208 total experiments using the parameters shown in Table 4.2. We run 16 experiments with the baseline algorithm using all combinations of \( N \) and \( P_{\text{perc}} \). Then we run 192 experiments with the proposed algorithm using all combinations of \( N, C, P_{\text{perc}}, \) and \( X_{\text{perc}} \).

Running those experiments on a single computer takes months. Thus, we utilized tens of nodes in the Holland Computing Center at the University of Nebraska [51]. We run all of the 16 experiments of the baseline algorithm in parallel as a single batch, then divided the 192 experiments of the proposed algorithm into eight batches, each having 24 experiments that run in parallel.

4.6. Results Discussion

Since we cannot present all 208 experiments, we fixed some parameters and show the results for changing the other parameters.
Table 4.2: Simulation Parameters.

<table>
<thead>
<tr>
<th>Sym.</th>
<th>Parameter</th>
<th>Value(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>No. of clients</td>
<td>(100, 200, 400, 900)</td>
</tr>
<tr>
<td>$C$</td>
<td>No. of clusters</td>
<td>(4, 8, 16, and 32)</td>
</tr>
<tr>
<td>$R$</td>
<td>Communication rounds</td>
<td>32</td>
</tr>
<tr>
<td>$E$</td>
<td>Epochs</td>
<td>8</td>
</tr>
<tr>
<td>$B$</td>
<td>Batches</td>
<td>32</td>
</tr>
<tr>
<td>$R_c$</td>
<td>Rounds per cluster</td>
<td>4</td>
</tr>
<tr>
<td>$P$</td>
<td>Percentage of poisoned dataset</td>
<td>(0, 10, 20, and 40)</td>
</tr>
<tr>
<td>$X$</td>
<td>Percentage of expelled clients per cluster</td>
<td>(0, 10, and 20)</td>
</tr>
</tbody>
</table>

Figure 4.2: Accuracy of the proposed v.s. the baseline algorithms with $P = 0\%$ and $X = 0\%$. 

57
Figure 4.3: Accuracy of models (i.e clusters) used in the proposed algorithm with \( N = 400 \), \( P = 40\% \), and \( X = 20\% \).

Figure 4.4: Accuracy of the proposed v.s. the baseline algorithms with \( C = 32 \) and \( N = 900 \).
4.6.1. Performance of Clustering

We measured the performance of both algorithms in terms of accuracy using a non-poisoned dataset ($P = 0\%$) and disabled expelling in the proposed algorithm ($X = 0\%$). Results are illustrated in Fig. 4.2 (summary of 20 experiments), which shows the superior performance of the proposed algorithm over the baseline algorithm using different numbers of clusters and clients. These results support the claim that the second stage of the proposed algorithm selects a better model than the baseline algorithm and thus results in better (i.e., higher) accuracy while using the same number of epochs. However, this gain in performance comes at the expense of communication cost, which can be reduced when clients are expelled (i.e., $X \geq 0$), as discussed in subsection 4.6.3. Fig. 4.3 shows the second stage of the proposed algorithm in action. It shows the accuracy of each model (i.e., cluster) over iterations, starting with eight models at iteration one and ending with one model (best performing
model in cluster five) at iteration seven.

4.6.2. Performance with Poisoned Dataset

To study the effect of the poisoned dataset on the performance of the two algorithms, we run both algorithms using different percentages of the poisoned dataset, as shown in Fig. 4.4. In this figure, we can see that when the poisoned percentage of the dataset is higher, especially when $P = 40\%$, the performance of the baseline algorithm in terms of accuracy is severely impacted. On the other hand, the proposed algorithm is barely impacted due to clustering (i.e., better-performing model) and expelling of clients with the poisoned dataset.

Table 4.3: Efficiency of expelling clients with $N = 400$, $P = 40\%$, $X = 20\%$, and $C = 8$.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>True Positive</th>
<th>False Positive</th>
<th>True Negative</th>
<th>False Negative</th>
<th>Total Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td>56</td>
<td>184</td>
<td>136</td>
<td>400</td>
</tr>
<tr>
<td>2</td>
<td>31</td>
<td>33</td>
<td>151</td>
<td>105</td>
<td>320</td>
</tr>
<tr>
<td>3</td>
<td>37</td>
<td>11</td>
<td>140</td>
<td>68</td>
<td>256</td>
</tr>
<tr>
<td>4</td>
<td>31</td>
<td>9</td>
<td>131</td>
<td>37</td>
<td>208</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>12</td>
<td>119</td>
<td>17</td>
<td>168</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td>10</td>
<td>109</td>
<td>1</td>
<td>136</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>20</td>
<td>89</td>
<td>0</td>
<td>110</td>
</tr>
</tbody>
</table>

Figure 4.6: Meaning of Negative, Positive, True, and False in Table 4.3
4.6.3. Performance with Expelling Clients

Expelling clients with poisoned or poor dataset prevent the deterioration of the performance of the proposed algorithm as indicated in Fig. 4.4 compared to the baseline algorithm. However, expelling a client with a good dataset can reduce the performance of the proposed algorithm. To study the expelling process in more depth, we track the number and status of expelled clients after each iteration and create a confusion matrix, as shown in Table 4.3. Fig. 4.6 explains the meaning of the four main columns in Table 4.3. For example, a True Positive number represents the number of clients being expelled by the proposed algorithm that has a poisoned dataset. A False Positive is not always a bad indication because the expelled clients may have a poor dataset. On the other hand, a False Negative always negatively impact the performance of the proposed algorithm. We start with 400 nodes in the experiment in Table 4.3 with a total of 160 poisoned nodes (40%). In the first iteration, the number of False Positive is higher than the number of True Positive, which is expected since the proposed algorithm is just starting and need more training. Also, the number of False Negative is high because the proposed algorithm is defined to expel only 20% after each iteration and thus cannot eliminate all poisoned clients in one iteration. In the second iteration and on, the number of True Positive becomes close or higher than the number of False Positive, which proves that the proposed algorithm is working as expected and detects poisoned clients more accurately. After the last iteration, we have only 110 clients left out of the 400, 0 False Negative, and all of the 160 clients with poisoned datasets are expelled successfully (i.e., 100%). Out of the 290 expelled clients, there are 130 clients with a clean dataset, and those have poor datasets compared with the remaining clients.

The proposed algorithm uses only 110 clients out of 400 in the third stage and still gets better results compared with the baseline algorithm in terms of accuracy. This big reduction
in the number of utilized clients reduces the communication cost tremendously, as illustrated in Fig. 4.5.

4.6.4. Lessons Learned

The key lessons learned from the experiments conducted in this work can be summarized as follows.

- The proposed algorithm can select a better model compared to the baseline algorithm due to using exploration and exploitation and thus results in better accuracy while using the same number of epochs.

- The proposed algorithm is better suited to cope with poisoned data, compared to the conventional FL algorithm, by expelling clients with the poisoned dataset.

- The proposed algorithm can accurately identify the clients with poisoned or poor data without affecting the overall performance of the final model.

- The expelling process not only expels clients with poisoned and poor dataset but also reduce communication cost.

4.7. Conclusions and Future Work

In this paper, we presented an algorithm that clusters clients in the first stage into a number of clusters, each with a random model. In the second stage, the proposed algorithm explores those models (i.e., clusters) by training them in a number of iterations and reduces the number of clusters by one after each iteration to find the best performing model (i.e., cluster). Also, in each iteration, the proposed algorithm expels some clients that have poisoned or poor datasets while surviving clients are exploited in the next iterations. Then, in
the third stage, the proposed algorithm continues the training process with the one remaining cluster, representing the selected model (best performing model) and returning the trained selected global model. The proposed algorithm is compared with a baseline algorithm, which is the random FL. Results show that the proposed algorithm is performing better in terms of accuracy and number of communication rounds when configured to expel clients compared with the baseline algorithm.

In the future, we plan to solve the selection of clients as an optimization problem to maximize the accuracy and minimize communication rounds given a fixed percentage of the poisoned dataset.
CHAPTER 5

CONCLUSION AND FUTURE WORK

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

5.1. Conclusion

In this work, we present solutions to three different problems in IoT applications by developing offline and online algorithms that run in brokers. The main concluded points are summarized as follows:

- We proposed an algorithm utilized by SDN controllers to minimize the load difference between brokers while respecting a reconfiguration limit in support of data and decision fusion applications in smart cities. The algorithm balance the load between brokers by switching topics between brokers based on long-term statistics of topics. Results of experiments show that the proposed algorithm outperformed the baseline algorithm by up to 2000% better load distribution and 70% less topic switching.

- We proposed an infrastructure-less system that opportunistically utilizes vehicles to serve as Local Community Brokers (LCBs) that effectively substitute Road-Side Units (RSUs) for managing communications between smart devices and the cloud in support of smart community applications. We proposed an algorithm that runs an ensemble of online selection algorithms in passive mode and selects the one that has performed the best in recent history. The proposed algorithm achieves up to 87% more service time with up to 10% fewer vehicles selected compared to the best-performing Threshold-Based Online Algorithm (TBO) algorithm.
• We propose a Federated Learning (FL) algorithm inspired by evolutionary techniques that groups clients randomly in many clusters, each with a model selected randomly to explore the performance of different models. The clusters are then trained in a repetitive process where the worst performing cluster is removed in each iteration until one cluster remains. In each iteration, some clients are expelled from clusters either due to using poisoned data or low performance. The surviving clients are exploited in the next iteration. The remaining cluster with surviving clients is then used for training the best FL model. Conducted experiments show that the proposed algorithm outperforms the baseline algorithm in terms of accuracy, communication cost, and security.

5.2. Future Work

Brokers play a vital role in IoT applications since they manage resources in terms of processing, communication, and security. IoT applications face many challenges, and research doors are widely open for enhancements and novel ideas. My goal is to continue doing research on developing algorithms to tackle those challenges. I plan to work on two paths:

• Develop algorithms inspired by nature, such as bees and bats, and study their behavior in solving processing, communication, and security problems.

• Investigate the use of machine learning, deep learning, and federated learning in enhancing the performance of algorithms in support of IoT applications.
BIBLIOGRAPHY


