Accepted for Publication

Data-Driven Prediction and Optimization of Energy Use for Transit Fleets of Electric and ICE Vehicles

AFIYA AYMAN, University of Houston
AMUTHEEZAN SIVAGNANAM, University of Houston
MICHAEL WILBUR, Vanderbilt University
PHILIP PUGLIESE, Chattanooga Area Regional Transportation Authority
ABHISHEK DUBEY, Vanderbilt University
ARON LASZKA, University of Houston

Due to the high upfront cost of electric vehicles, many public transit agencies can afford only mixed fleets of internal-combustion and electric vehicles. Optimizing the operation of such mixed fleets is challenging because it requires accurate trip-level predictions of electricity and fuel use as well as efficient algorithms for assigning vehicles to transit routes. We present a novel framework for the data-driven prediction of trip-level energy use for mixed-vehicle transit fleets and for the optimization of vehicle assignments, which we evaluate using data collected from the bus fleet of CARTA, the public transit agency of Chattanooga, TN. We first introduce a data collection, storage, and processing framework for system-level and high-frequency vehicle-level transit data, including domain-specific data cleansing methods. We train and evaluate machine learning models for energy prediction, demonstrating that deep neural networks attain the highest accuracy. Based on these predictions, we formulate the problem of minimizing energy use through assigning vehicles to fixed-route transit trips. We propose an optimal integer program as well as efficient heuristic and meta-heuristic algorithms, demonstrating the scalability and performance of these algorithms numerically using the transit network of CARTA.

CCS Concepts: • **Computer systems organization** → *Embedded and cyber-physical systems*; • **Computing methodologies** → *Machine learning*; • **Information systems** → *Data management systems*; • **Applied computing** → *Transportation.*

ACM Reference Format:
Afiya Ayman, Amutheezan Sivagnanam, Michael Wilbur, Philip Pugliese, Abhishek Dubey, and Aron Laszka. 2020. Data-Driven Prediction and Optimization of Energy Use for Transit Fleets of Electric and ICE Vehicles. *ACM Trans. Internet Technol.* 0, 0 (2020), 29 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Transportation accounts for 28% of the total energy use in the U.S. [13], and as such, it is responsible for immense environmental impact, including urban air pollution and greenhouse gas emissions, and may pose a severe threat to energy security. Switching from personal vehicles to public transit systems can significantly reduce energy use and environmental impact. However, even public transit systems require substantial amounts of energy; for example, public bus transit services in the U.S. are responsible for at least 19.7 million metric tons of CO_2 emission annually [35].

Electric vehicles (EVs) can have much lower environmental impact during operation than comparable internal combustion engine vehicles (ICEVs), especially in urban areas. Unfortunately, EVs are also much more expensive than ICEVs (typically, diesel transit buses cost less than $500K, while electric ones cost more than $700K, or close to around $1M with charging infrastructure [36]). As a result, many public transit agencies can afford only mixed fleets of transit vehicles, which may consist of EVs, hybrids (HEVs), and ICEVs. Transit agencies that operate such mixed fleets of vehicles face a challenging optimization problem: these agencies need to decide *which vehicles are assigned to serving which transit trips.* Since the advantage of EVs over ICEVs varies depending on the route and time of day (e.g., the advantage of EVs is higher in slower traffic with frequent stops,

2020. 1533-5399/2020/0-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

and lower on highways), the assignment can have a significant effect on energy use and, hence, environmental impact. This optimization problem is computationally challenging as the number of possible assignments grows exponentially with the number of transit trips, and it cannot be formulated as a simple matching because of the intricate constraints due to trip schedules and vehicle travel times.

At the crux of this operational optimization is the problem of *accurately predicting the electricity and fuel consumption* of transit vehicles. Such predictions must be contextualized with a variety of factors, including the type of vehicle, traffic and weather conditions, road gradient, and type of road (e.g., highway vs. residential area) since these factors can have significant impact on energy use. Clearly, handling all of these factors using model-driven approaches, which attempt to build detailed physical models of vehicles, is very challenging.

Recent advances in sensor-based technologies, data analytics, and machine learning have enabled remedying this situation by building data-driven predictors of route-level energy use. However, to the best of our knowledge, there exists no framework that would integrate all relevant data into a route-level prediction model for public transit. Such a framework needs to address many challenges: high volume of unstructured and irregular data must be stored efficiently, allowing easy retrieval in subsequent steps; noisy data (e.g., GPS based locations) must first be cleansed (e.g., corrected or imputed based on other data sources); heterogeneous data (recorded at different rates with different precision in different formats) must be collated into samples that can be fed into training machine-learning models; etc.

A large stream of literature aims to efficiently predict energy consumption for vehicles from various perspectives. A few papers compare different predictive models for energy consumption. For example, Cauwer et al. [12] use a cascade of artificial neural network and multiple linear regression models as a data-driven energy-consumption prediction method. Their study includes data from only EVs, and their dataset does not consider traffic features. Wickramanayake and Bandara [45] compare three different methods for predicting the fuel consumption of a bus. However, their study lacks significant features, such as road information, traffic, weather, etc. Section 9.1 provides an overview of several relevant studies to give insight into the state-of-the-art research in the field of energy consumption prediction for vehicles. Unlike most prior energy consumption prediction models, our approach includes predictive models for both EVs and ICEVs. Further, it incorporates essential parameters such as weather conditions, traffic data, etc. Crucially, we do not incorporate traffic only as a proxy feature (e.g., day of week or time of day), but we incorporate it using parameters that represent actual traffic conditions as speed profile and road congestion.

Prior research efforts on energy-optimization for transit agencies mostly focus on optimally assigning buses to transit trips by altering existing transit schedules. For example, Wang et al. [43] consider a scheduling problem for transit agencies that operate only EVs, and they propose to assign buses to transit trips by modifying existing transit schedules to reduce energy consumption. Unfortunately, this is often infeasible in practice since it reduces the service level of the transit network for passengers. In our approach, we consider transit agencies that operate mixed-fleets of EVs and ICEVs based on a fixed schedule. Further, prior research efforts use a variety of data to estimate the energy consumption of EVs and ICEVs when assinging buses to transit trips. Santos et al. [39] use fixed costs, emission and consumption rates for different types of vehicles, while Paul et al. [37] assume that energy costs are fixed per unit distance, without considering any spatial or temporal features. Unlike these prior research efforts, we derive accurate energy estimates from our energy predictors using a variety of features, including high-resolution route information (i.e., location traces), traffic, elevation, and weather.

Most prior research efforts focus on minimizing the energy costs or environmental impact of transit vehicles under the assumption that energy consumption is proportional to the distance

traveled by an EV or ICEV during a trip. However, in practice, energy consumption can vary significantly due to factors such as weather, traffic, and elevation changes. Therefore, optimization based on a fixed rate of energy consumption per unit distance will yield suboptimal solutions. In contrast, we propose a complete framework for the data-driven prediction and optimization of the energy use of transit vehicles. First, we estimate trip-level energy consumption values based on machine-learning prediction models, which consider various factors such as weather, elevation, and traffic. Then, we use the energy consumption estimates as inputs to our optimization algorithms and assign buses to transit trips. Since our energy estimates are very accurate at the trip-level, the solutions found by our optimization algorithms will be feasible and near optimal in practice.

Contributions: In this paper[1], we present a novel framework for the data-driven offline prediction of route-level energy use for mixed-vehicle transit fleets and for the optimal assignment of vehicles to fixed-route transit trips, which we evaluate using data collected from the bus fleet of the Chattanooga Area Regional Transportation Authority (CARTA), the public transit authority of Chattanooga, TN.

- We collect and combine vehicle telemetry data, elevation and street-level maps, weather data, and traffic data. Our dataset is publicly available at https://smarttransit.ai/energy.html
- We present a cloud-centric data collection and storage framework for high-velocity spatio-temporal smart-city data. Our modular architecture is centered around a topic-based, distributed publish-subscribe (pub-sub) layer with easily extendable, application-specific structured views.
- We present a framework and novel algorithms for cleaning and integrating time series data from multiple sources into sets of samples with fixed-dimensional feature space, including a machine-learning based approach for accurately mapping noisy locations to road segments.
- We use this dataset to train machine-learning models for energy-use prediction (deep neural networks, linear regression, and decisions trees) and study their performance, focusing on the impact of including or omitting certain data sources.
- We formulate the problem of minimizing energy costs by assigning a mixed fleet of vehicles to fixed-route transit trips, considering constraints such as route schedules and vehicle travel times.
- We introduce an integer program for finding an optimal solution to this problem as well as randomized heuristic and meta-heuristic algorithms for finding near-optimal solutions efficiently. We demonstrate the efficiency of the proposed algorithms using numerical evaluation based on the real-world transit routes and schedules of CARTA.

Organization: The remainder of this paper is organized as follows. In Section 2, we describe our data sources, data collection methods, and data storage architecture. In Section 3, we introduce our data cleansing and integration framework. In Section 4, we propose machine-learning based prediction models. In Section 5, we evaluate the prediction models using real-world data. In Section 6, we formulate the problem of assigning vehicles to fixed-route transit trips. In Section 7, we introduce an integer program, randomized greedy heuristics, and a genetic algorithm for solving the optimization problem. In Section 8, we present numerical results on the proposed optimization algorithms. In Section 9, we discuss related work on predicting energy use, mapping noisy locations, and optimizing energy costs. Finally, in Section 10, we provide concluding remarks.

2 DATA COLLECTION AND STORAGE FRAMEWORK

We first provide an overview of the data sources that we use in our study (Section 2.1) and then describe the architecture of our data storage framework (Section 2.2). Table 1 provides an overview of our data sources.

[1]This paper is a significant extension of our previously published conference paper [8].

Table 1. Overview of Datasets

Data	Source	Frequency	Scope	Features
Diesel Vehicles (2014 Gillig Phantom diesel buses)	ViriCiti & CARTA	1 Hz	3 vehicles for 244 days (veh. IDs: 147, 149, 150; 2019-8-1 to 2020-3-31)	GPS location, fuel-level, fuel rate, odometer
Electric Vehicles (2016 BYD K9S 35-foot battery-electric buses)	ViriCiti & CARTA	1 Hz	3 vehicles for 244 days (veh. IDs: 751, 752, 753; 2019-8-1 to 2020-3-31)	GPS location, charging status, battery current, battery voltage, battery state of charge, odometer
Traffic	HERE [20]	1 Hz	TMC segments for major roads in Chattanooga	TMC ID, confidence of reading, unconstrained speed, free-flow speed, jam factor
Weather	DarkSky [11]	0.1 Hz	Chattanooga region	location, temperature, wind speed, precipitation, humidity, visibility, apparent temperature
Elevation	TN GIC [41]	static	Chattanooga region	location, elevation

2.1 Data Sources

2.1.1 Vehicle Data. To collect data from CARTA's fleet of vehicles, we partner with ViriCiti, a company that offers sensor devices and an online platform to support transit operators with real-time insight into their fleets. ViriCiti has installed sensors on CARTA's mixed-fleet of 3 electric, 41 diesel, and 6 hybrid buses, and it has been collecting data continuously at 1-second (or shorter) intervals since installation. For this study, we use data collected between August 2019 and March 2020 from 3 electric and 3 diesel vehicles as shown in Table 1. All electric buses are BYD K9S battery-electric transit vehicles, while the diesel buses are 2014 Gillig Phantom series vehicles with Cummins diesel engines.

For each vehicle, we obtain time series data from ViriCiti, which includes series of timestamps and vehicle locations based on GPS. For electric buses, we also include features such as battery current in ampere (A), battery voltage (V), battery state of charge, and charging cable status. For diesel buses, we include fuel level and the total amount of fuel used over time in gallons. In total, we have already obtained around 32.3 million data points for electric buses and 29.8 million data points for diesel buses (Table 1). Fuel data is recorded less frequently; hence, there are fewer data points for diesel buses.

2.1.2 Elevation, Weather, and Traffic Data. We collect static GIS elevation data from the Tennessee Geographic Information Council [41]. From this source, we download high-resolution digital elevation models (DEMs), derived from LIDAR elevation imaging, with a vertical accuracy of approximately 10 cm [42]. We join the DEMs for Chattanooga into a single DEM file, which we then use to determine the elevation of any location within the geographical region of our study.

We collect weather data from multiple weather stations in Chattanooga at 5-minute intervals using the DarkSky API [11]. This data includes real-time temperature, humidity, air pressure, wind speed, wind direction, and precipitation.

We collect traffic data at 1-minute intervals using the HERE API [20], which provides speed recordings for segments of major roads. Every road segment is identified by a unique Traffic Message Channel identifier (TMC ID) [1]. Each TMC ID is also associated with a list of latitude and longitude coordinates, which describe the geometry of the road segment. We use weather and traffic data collected from August 1, 2019 to March 31, 2020 to match the time range in Table 1.

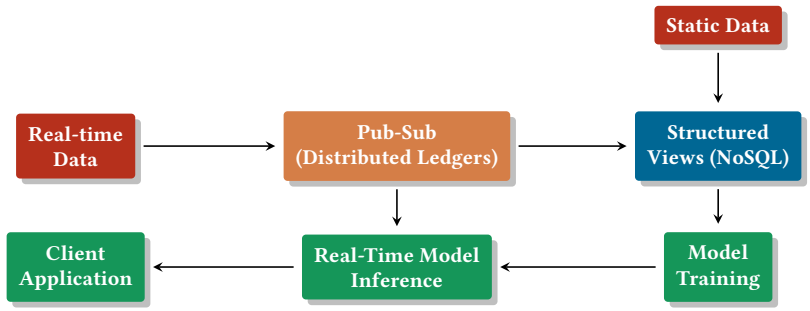


Fig. 1. Data architecture overview.

2.2 Data Architecture Framework

Next, we outline a general-purpose data architecture framework for storing the various smart-city data streams. The purpose of this framework is to store the data streams in a way that provides easy access for offline model training and updates as well as real-time access for system monitoring. Our architecture consists of a publish-subscribe cluster implemented with Apache Pulsar [3], which stores topic-labeled sensor streams, and a MongoDB database back-end. An overview of the data architecture is provided in Figure 1, while the implementation of the data storage components is shown in Figure 2.

The first challenge is the persistent storage of the high-velocity, high-volume data streams. In this study, the real-time data sources—ViriCiti, HERE, and DarkSky—produce around 100 GiB of data per month. Therefore, we choose a cloud based design to allow for fast horizontal scalability of the system.

The second concern is that the data itself is highly unstructured and irregular as the sources produce data at different rates. Therefore, we stream each data source to a topic-based publish-subscribe (pub-sub) layer that persistently stores each data stream as a separate topic. The pub-sub system consists of a single Apache Pulsar [3] cluster running on VMware [7] virtual machines hosted at Vanderbilt University. We used a three-tiered naming convention for topic labeling. The first tier represents the name of the data tenant and all authentication and access is managed at this level. The second tier is the data category, i.e., vehicle telemetry, traffic, weather, etc. The third tier is the topic name, which represents the data source or provider, such as ViriCiti, HERE, or DarkSky. For ViriCiti vehicle-telemetry data, the fleet name is appended to the topic name to separate electric, diesel, and hybrid vehicles. The tenant, category, and topic names together form a topic, which downstream applications can use to access the data streams. We persistently store all messages on each topic in an append-only ledger. Therefore, the topic can be used to read data in near real-time or to playback previous data streams to synchronize new downstream applications. All replication is handled at the ledger level, which allows downstream storage and applications to adapt and expand without concern for data resiliency.

Model-training and inference require data from various streams to be merged. Typical implementations of stream processing architectures require external processing frameworks, such as Apache Spark and Storm [22, 46]. For our system, we instead incorporate a customized stream-processing layer into the pub-sub module. In this layer, data cleansing and processing functions are applied to the raw data topics, and the processed data is then published to separate reformed topics, which can easily be accessed for prediction or model training.

As shown in Figure 2, we use the pub-sub framework Apache Pulsar for the topic-based distributed ledger module. The storage component of Apache Pulsar relies on Apache BookKeeper [2],

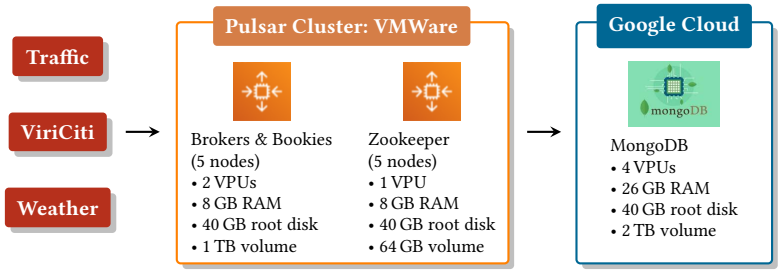


Fig. 2. Data architecture implementation.

which allows sharding of data at the topic level. As the size and velocity of data varies greatly between data sources, topic-level sharding allows data to be evenly distributed between the storage nodes and thus maximize resources in the cluster. Cluster state and coordination is managed with Apache ZooKeeper [4]. The Apache Pulsar system provides automatic failover and load balancing. Additionally, we added a custom monitoring system to detect abnormal changes in data ingestion and processing behavior.

While distributed topic-based ledgers provide fast real-time access to data and easy data replication, the complexity of working with spatiotemporal data requires a more structured representation of the data. For this, we used MongoDB [5], which provides native R-Tree geospatial indexing. As a NoSQL document data store, MongoDB provides a flexible data model and efficient query capabilities for training and batch analysis. Additionally, this model allows for easy large-scale exports in JSON format for sharing datasets between research sites. To ingest data into the MongoDB database, we developed functions that subscribe to the data-stream topics in Pulsar to synchronize MongoDB with the distributed ledgers in Pulsar. The MongoDB instance was deployed in Google Cloud [6].

3 DATA PROCESSING FRAMEWORK

Before applying machine-learning models, we have to process the time series data recorded from the vehicles by cleaning it, generating samples with a fixed-dimension feature space, and incorporating data from other sources, including traffic and weather data. For each bus, the recorded data is a series of datapoints, numbered $i = 0, 1, 2 \ldots$, where each datapoint is a tuple of a timestamp TS_i, a location L_i, etc. For electric vehicles, each datapoint i includes a battery current A_i, a battery voltage V_i, a battery state of charge SoC_i, and a charging cable status CS_i. For diesel vehicles, each datapoint i includes fuel consumption F_i.

3.1 Removing Garage Locations and Charging

Since our goal is to predict the amount of energy used for driving, we remove all datapoints that were recorded when a bus was (1) waiting in the garage or (2) charging. Thus, we remove every datapoint i whose GPS-based location L_i falls in the geographical proximity of the CARTA bus garage as follows:

$$\mathit{distance}(\text{GarageLocation}, L_i) < 400 \text{ meters} \longrightarrow \text{Drop datapoint } i \tag{1}$$

For electric buses, we also remove every datapoint i whose charging cable status CS_i indicates that the vehicle was charging:

$$CS_i = 1 \longrightarrow \text{Drop datapoint } i \tag{2}$$

3.2 Estimating Energy Use for Electric Vehicles

For diesel buses, we can compute the amount of fuel used between two consecutive datapoints as the change in the total amount fuel used. For electric buses, we could compute the amount of energy used as the change in the battery state of charge (SoC), which is the remaining battery charge as a percentage of the total capacity. However, SoC values are recorded with a low precision of only one digit after the decimal point. To obtain accurate values, we need to estimate the amount of energy used based on the recorded battery current (A) and voltage (V) values. At any time, the instantaneous power use of the vehicle (in Watt) can be computed as $A \cdot V$. We can estimate the amount of energy used (in Joule), E_i between consecutive datapoints $i-1$ and i using Equation (3):

$$E_i = A_i \cdot V_i \cdot (TS_i - TS_{i-1}), \tag{3}$$

where TS_i is the timestamp of datapoint i (in seconds). Since current and voltage values are recorded at least once every second, the above formula provides a high-accuracy estimate. We confirmed that our estimates are unbiased by comparing them to changes in SoC over large numbers of datapoints.

3.3 Mapping GPS Locations to Roads

The recorded vehicle locations are inherently noisy since they are based on GPS. For example, some of the locations fall onto streets or parking lots where a bus cannot even drive. This noise presents a significant challenge for computing accurate travel distances and for integrating the time series with other data sources. To mitigate this noise, we combine the recorded vehicle locations L_i with a street-level map of Chattanooga, which we obtain from OpenStreetMap (OSM) [16]. OSM represents each road using a disjoint set of segments, which are called OSM features. Specifically, OSM divides each road into one or more segments along its length and assigns a unique *OSM Feature ID* to each one of these road segments. These segments also have other properties, such as type of the road (primary, secondary, service, residential, etc.), geometry of the road segment, street name, etc.

To combine vehicle locations with the street-level map, we map each location L_i to a road segment. The process of mapping a sequence of noisy GPS locations to the corresponding points on the road network of a digital map has been widely used in location-based applications, such as vehicle navigation [47] and analysis of urban road networks [26]. We introduce two novel methods for mapping noisy locations to a road network: a heuristic algorithm and a machine-learning based approach. In Section 9.2, we discuss the existing alternative approaches.

3.3.1 Heuristic Algorithm for Mapping. We map each recorded location L_i to an OSM segment (i.e., road segment). For a particular location L_i, we consider as candidates the set of nearby OSM segments based on geographical distance. First, we create an R-tree spatial index for the geometry of the street-level map of Chattanooga, TN. Then, we intersect the spatial index with a bounding disk around location L_i, whose radius is the geographical distance threshold for considering a road segment. The result of this intersection is a set of road segments that may intersect with the bounding disk, from which we select the ones that actually intersect by calculating distances for them one-by-one. Finally, we filter this set based on road types since we need to consider only road segments where a bus can actually drive. So, we omit segments of several road types, such as footway and cycleway, from the set of nearby segments.

For each nearby OSM segment, we count how many of the preceding and following locations were also near this segment. Finally, we select the segment that is near the most locations. Algorithm 1 details the process of mapping locations to OSM segments.

Algorithm 1: Mapping Locations to OSM Segments

Input : $L \leftarrow$ list of locations (i.e., datapoints from vehicle)
$Map \leftarrow$ OSM street-level map
$WINDOW \leftarrow$ lookahead and lookback window
Output : $RoadSegments \rightarrow$ OSM segments traveled
Initialization :

```
NearbySegments ← [][] /* list of nearby OSM segments for each location   */
RoadSegments ← [] /* OSM segment for each L_i after mapping              */
for i ∈ {1, ..., |L|} do
    NearbySegments[i] ← Map.FindNearbySegments(L_i)
for i ∈ {1, ..., |L|} do
    if |NearbySegments[i]| > 0 then
        Frequency ← [ ]
        for Segment ∈ NearbySegments[i]) do
            Count ← 0
            /* check within WINDOW                                        */
            for j ∈ {i − WINDOW, ..., i + WINDOW} do
                for OtherSegment ∈ NearbySegments[j] do
                    if Segment = OtherSegment then
                        Count ← Count + 1
            Frequency[Segment] ← Count
        Selected ← argmax_j Frequency[j]
        RoadSegments[i] ← NearbySegments[i][Selected]
```

3.3.2 Machine Learning Approach for Mapping. In this approach, we train a regression model for mapping noisy locations to OSM segments. Since there are thousands of OSM segments in cities like Chattanooga, it would be very challenging to train a classifier that could directly output the correct segment for any location. Instead, for each location L_i, we first identify a set of nearby OSM segments as candidates as described in Section 3.3.1, use the regression model to estimate how likely each candidate segment is to be the correct one, and then output the most likely candidate.

The input variables of the regression model are based on the location and candidate OSM segment. We use a total of 25 input variables, which include distance between the location and the OSM segment, the road type of the OSM segment, the maximum, average, and minimum distance between the OSM segment and the set of 15 preceding and following vehicle locations. The model outputs a value in $[0, 1]$ with 0 indicating the lowest likelihood that the location would be correctly mapped to this OSM segment, and 1 indicating highest likelihood. For each location, we apply this regression model to each nearby candidate OSM segment and map the location to the segment with the highest likelihood.

To train the regression model, we create a training set of locations with ground truth for the correct mapping to road segments. First, we generate routes using a street-level map and select traces of locations along these routes, recording for each location the corresponding, correct road. Then, we add random noise to the locations using a two-dimensional Gaussian distribution with zero mean to simulate the noisiness of GPS-based locations. Finally, for each noisy location, we gather all the nearby road segments, and we label the correctly mapped pairs of locations and road segments with 1, and label the rest with 0. One important feature of the training is the level of

noise that is added to the locations, which should be chosen to resemble real-world noise levels. In Section 5.1, we experiment with different noise levels, the standard deviation of the noise varying between 1 meter and 140 meters in both directions.

We implement the regression model using decision tree and linear regression models. For both models, we use the implementation provided by the *scikit-learn* Python library. We set the maximum depth of the decision tree to the total number of features in order to reduce memory consumption. We also experiment with different values for the minimum number of samples for each split in the tree until we find our best-fit. This parameter controls over-fitting: low values can lead to over-fitting to a few samples, while high values may prevent learning accurate predictions from the data. We evaluate both models based on R^2 value, and find that the decision tree model performs better than linear regression at all noise levels. For example, at a noise level of 8.88 meters, decision tree model has an R^2 value of 89.42, whereas at the same noise level, linear regression has 65.38. In light of this, we opt to use decision tree regression in our framework.

3.3.3 Combination with Street-Level Information. Once we have mapped each location to an OSM segment, we add the corresponding OSM Feature ID to each datapoint, which we use to generate samples (Section 3.4) and later to calculate accurate travel distances (Section 3.5). We also add information from OpenStreetMap regarding the road, such as the type of the road, whether the road is one-way or two-way, whether it is a tunnel, etc. In our dataset, we encounter 14 different road types in total, which include primary, residential, motorway, etc. For some roads, the type is "*unknown*" on OpenStreetMap, which we treat as a distinct type.

3.4 Generating Samples

Next, we generate a set of samples, numbered $j = 1, 2, 3, \ldots$, from the time series data by dividing the time series of each bus based on the traveled road segments. Specifically, for each bus, we treat a maximal continuous travel on a particular road segment (i.e., particular OSM Feature ID) as one sample. Each sample j includes the starting location L_j^{start}, the end location L_j^{end}, the starting time TS_j^{start}, the end time TS_j^{end}, and the sum of the amount E_j of electric energy or fuel used between the starting and end points. Thus, samples may be represented using a fixed-dimension feature space, which facilitates feeding them into machine-learning models for training and prediction.

3.5 Calculating Travel Distance

Since GPS based locations are noisy, we combine them with OpenStreetMap to calculate the distance traveled, D_j for each sample accurately. First, for each sample, we obtain the geometry of the corresponding road segment from OSM as a list of contiguous line segments, $line_1, line_2, \ldots line_n$. Because the bus does not necessarily travel the complete distance of the road segment (e.g., it could turn on a different street before reaching the end of the road segment), we need to identify the first and last line segments that the bus actually traveled. We calculate the distance between each line segment and the starting location L_j^{start} and end location L_j^{end} of the sample, which we denote $Dist_S[]$ and $Dist_E[]$, respectively. Next, we identify the indices of the line segments that are closest to L_j^{start} and L_j^{end}, which we denote $index_S$ and $index_E$, respectively. Finally, we calculate the distance traveled D_j for the sample based on the partial distance on line segment $index_S$, the complete distance of all line segments in between, and the partial distance on line segment $index_E$, according to Algorithm 2.

3.6 Removing Erroneous Samples

Even though current and voltage values are almost always correctly recorded, we did find a few datapoints that have erroneous or missing values, which result in extremely low, negative energy

Algorithm 2: Calculating Travel Distance for Sample

Input : $L_j^{\text{start}} \leftarrow$ starting point of sample
$L_j^{\text{end}} \leftarrow$ end point of sample
$line_1, line_2, \ldots, line_n \leftarrow$ line segments of the OSM feature of the sample
Output : $D_j \rightarrow$ distance traveled in sample

for $i \in \{1, \ldots, n\}$ **do**
  $Dist_S[i] \leftarrow$ distance between L_j^{start} and $line_i$
  $Dist_E[i] \leftarrow$ distance between L_j^{end} and $line_i$
$index_S \leftarrow argmin_i\ Dist_S[i]$
$index_E \leftarrow argmin_i\ Dist_E[i]$
/* vehicle moving in direction $line_{index_S}, line_{index_S+1}, \ldots, line_{index_E-1}, line_{index_E}$ */
if $(index_S < index_E)$ **then**
  $l_1 \leftarrow$ distance between L_j^{start} and second endpoint of $line_{index_S}$
  $l_2 \leftarrow$ sum length of $line_{index_S+1}, \ldots, line_{index_E-1}$
  $l_3 \leftarrow$ distance between the first endpoint of $line_{index_E}$ and L_j^{end}
  $D_j \leftarrow l_1 + l_2 + l_3$
/* vehicle moving in direction $line_{index_E}, line_{index_E+1}, \ldots, line_{index_S-1}, line_{index_S}$ */
else if $(index_S > index_E)$ **then**
  $l_1 \leftarrow$ distance between L_j^{end} and second endpoint of $line_{index_E}$
  $l_2 \leftarrow$ sum length of $line_{index_E+1}, \ldots, line_{index_S-1}$
  $l_3 \leftarrow$ distance between the first endpoint of $line_{index_S}$ and L_j^{start}
  $D_j \leftarrow l_1 + l_2 + l_3$
/* $index_S = index_E$ */
else
  $D_j \leftarrow$ distance between L_j^{start} and L_j^{end}.

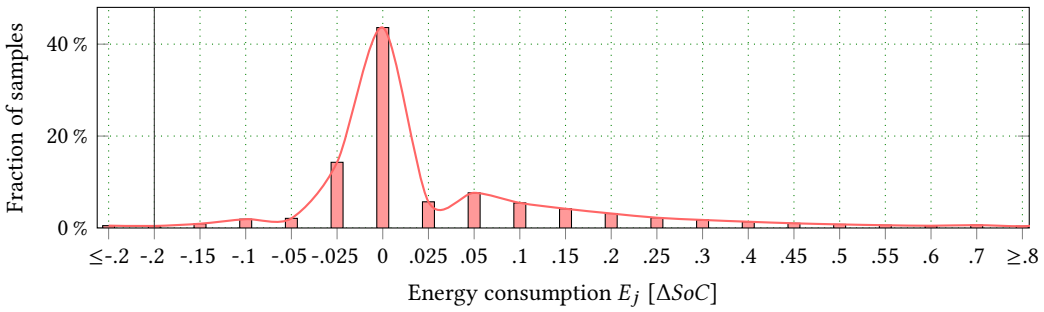


Fig. 3. Distribution of energy consumption values E_j for samples j from electric vehicles in our dataset (measured as change in battery state of charge).

consumption estimates, E_j. Note that many electric vehicles can recharge from braking; so E_j can in fact be negative for some shorter samples when the bus is slowing down or going downhill. However, erroneous values result in implausibly low values. Note that we did not observe any erroneous values in diesel fuel consumption.

Figure 3 shows the distribution of the energy consumption values E_j (measured as changes in *SoC*) for the 277,262 samples that we obtain for electric vehicles. Of these samples, 99.5% have energy use values greater than or equal to -0.2. Only 1,407 samples have values lower than -0.2, constituting 0.5% of the dataset. We remove these 1,407 samples from the dataset:

$$E_j < -0.2 \longrightarrow \text{Drop sample } j \tag{4}$$

3.7 Incorporating Elevation

To add road gradients to the samples, we calculate the difference ΔElevation between the elevation at the start and end points, L_j^{start} and L_j^{end}, of each sample:

$$\Delta\text{Elevation} = \text{Elevation of } L_j^{\text{end}} - \text{Elevation of } L_j^{\text{start}} \tag{5}$$

The change in elevation captures the net potential energy gained or lost during the sample.

3.8 Incorporating Weather

Since our goal is to provide predictions for planning transit operations in advance, we cannot rely on real-time data for weather. Instead, we compute hourly weather predictions for each station based on the recorded historical weather data. Then, for each sample, we compute the distance between the end point of the sample and each weather station, and we add the predicted weather features of the closest station to the sample. Our weather dataset has a number of features, of which we use temperature (T), humidity (H), visibility (V), wind speed (W), and precipitation (P).

3.9 Incorporating Traffic

Our traffic dataset consists of timestamped speed values recorded for segments of roads in Chattanooga, which are identified using Traffic Message Channel (TMC) identifiers [1]. Each TMC segment represents a specific, directed segment of a major road, whose geometry is stored as a list of geo-points. While the TMC format is adequate for delivering and storing traffic information, we must also be able to integrate traffic data with our samples, which reference road segments using OSM Feature IDs. To this end, we need to map OSM segments to TMC segments. This mapping presents two challenges. First, OpenStreetMap typically divides roads into significantly smaller segments than TMC segments, so matching based on similarity of geometry is difficult. Second, TMC segments cover only major roads, so most OSM segments cannot be mapped to any TMC segment.

To set up the mapping, we first generate an OpenStreetMap routing graph. This graph enables us to find the shortest driving-distance path between any two nodes, which represent real-world locations, returning a list of edges. Each edge is labeled with the ID of the corresponding OSM segment (i.e., road segment). Next, for the start and end geo-points of each TMC segment, we find the closest nodes in the OSM routing graph. Finally, for each TMC segment, we find the shortest path in the OSM routing graph between the start and end nodes, and we map each edge (i.e., OSM segment) of the path to the TMC segment.

However, in some cases, the start and end geo-points of a TMC segment are matched to OSM nodes on the opposite sides of a road, which causes errors in the mapping. Therefore, instead of finding only the nearest OSM node, we find the four nearest nodes for each start and end geo-point. Then, we find all the shortest paths between all the start and end nodes, select the path whose length matches the actual length of the TMC segment most closely, and map the OSM segments of only this path to the TMC segment. We found that this process significantly improves the OSM to TMC mapping.

Based on this mapping, we add traffic information to our samples. Similar to weather, we cannot rely on real-time traffic for energy use prediction. Instead, we compute average traffic conditions for each TMC segment for each hour of each day of the week based on the recorded data, and we use these hourly averages as traffic predictions. For each sample, we add the hourly prediction for the TMC segment to which the OSM feature of the sample is mapped. For samples that cannot be mapped, we impute special values, which we discuss below. We add two features from our traffic dataset to each sample: *speed ratio* and *jam factor*. Speed ratio is the actual traffic speed over the free-flow speed; values around 1 mean light or no traffic, while values around 0 mean very heavy traffic. Jam factor indicates the expected quality of travel, ranging from 0 (light or no traffic) to 10 (road closure) [20]. For samples that cannot be mapped to a TMC segment, we let the speed ratio and jam factor be 1 and 0, respectively, since road segments that are missing from our traffic dataset are typically minor roads, which rarely experience heavy traffic.

4 ENERGY CONSUMPTION PREDICTION MODELS

Our primary goal in this work is to enable transit agencies to minimize the energy use of their vehicles through operational optimization, including the optimization of vehicle assignments. To perform such optimization for mixed fleets of electric, diesel, and hybrid vehicles, transit agencies must be able to predict how much energy a given vehicle would use on a given route at a given time. To address this need, here we propose macroscopic energy predictors based on state-of-the-art machine learning techniques, such as artificial neural networks, which take as input the processed data that we described in the previous section. In subsequent sections, we evaluate the accuracy of the proposed predictors and the usefulness of various input features based on real-world transit data, and we formulate and solve a transit optimization problem based on the trained predictors.

We apply three different machine-learning models for predicting energy consumption: artificial neural network, linear regression, and decision tree regression. We chose neural networks for their superior prediction performance, which is confirmed by our numerical results. In contrast, linear and decision tree regression do not perform as well, but their results are easier to understand and explain. For example, linear regression shows the direct relation between input variables and target features.

The input of the energy prediction models (i.e., training or test set) is a set of samples, where each sample j is a tuple of distance travelled D_j, various road-type features, elevation change ΔElevation, various weather features, various traffic features, and energy used E_j as the target feature. Before training and testing, we map categorical variables (e.g., road type) into sets of binary features using one-hot encoding. We train all three models to minimize *mean squared error* (MSE).

4.1 Artificial Neural Network

We found that different network structures work best for diesel and electric vehicles. For electric vehicles, the best model has one input, two hidden, and one output layer. The input layer has one neuron for each predictor variable. The two hidden layers have 100 neurons and 80 neurons, respectively. For diesel, the best model has one input, five hidden, and one output layer. The five hidden layers have 400, 200, 100, 50, and 25 neurons, respectively. In all the hidden layers, we use sigmoid activation, and we use linear activation in the output layer. We optimize the models using the *Adam* optimizer [23] with learning rate 0.001. To implement the ANN models, we use Keras, which is a high-level API of TensorFlow for building and training deep learning models [10].

4.2 Linear Regression

Our second model is a standard multiple linear regression. We use the implementation provided by the *scikit-learn* Python library.

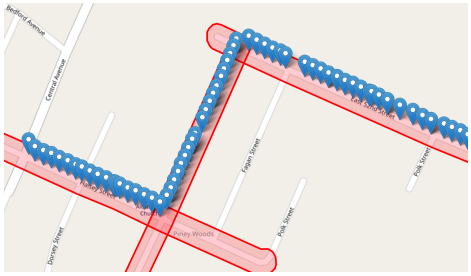


Fig. 4. Example of mapping locations to road segments without noise. Road segments that are output by Algorithm 1 are highlighted in red.

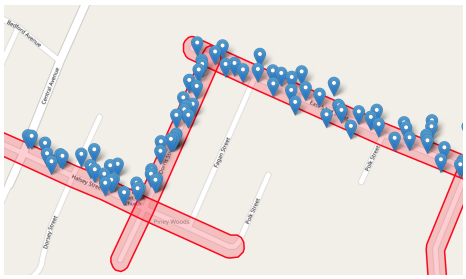


Fig. 5. Mapping with noise with 14-meter std. dev.

Fig. 6. Mapping with noise with 28-meter std. dev.

4.3 Decision Tree

Our third model is decision tree regression [40]. This model builds a tree structure based on the training samples, where each node represents a decision based on the value of a feature variable, and leaf nodes provide predictions. We use the implementation provided by the *scikit-learn* Python library.

5 NUMERICAL RESULTS ON DATA PROCESSING AND ENERGY PREDICTION

5.1 Algorithms for Mapping GPS Locations to Road Segments

We begin by evaluating the accuracy of our algorithms for mapping noisy locations to OSM features. Since we do not have ground truth for the correct mapping of the GPS-based locations of the real transit vehicles, we create a test dataset with known ground truth. First, we generate routes using a street-level map and select a set of locations along these routes, which are precisely on the roads (Figure 4). Our test dataset has 721,492 different locations.[2] Then, we add random noise to these locations, generated using a two-dimensional Gaussian distribution with zero mean. We vary the standard deviation of the noise between 1 meter and 140 meters in both directions. Figures 4 to 6 show locations with different levels of noise added, highlighting in red the road segments to which locations are mapped by Algorithm 1. Finally, we map the noisy locations to road segments using both Algorithm 1 and the machine-learning based approach with decision tree regression (Section 3.3.2) and measure accuracy as the ratio of correctly mapped locations.

Figure 7 compares the accuracy of the two approaches for various levels of noise, ranging from 1.1-meter to 140-meter standard deviation in both directions. For minimum noise level (1.1 meters), both the heuristic algorithm and the machine-learning approach attain accuracy above 98%. As

[2]Note that we use these synthetic location traces only for the evaluation of the mapping approaches, where we need ground-truth segments; for training and evaluating energy-use prediction, we use real location traces from CARTA vehicles.

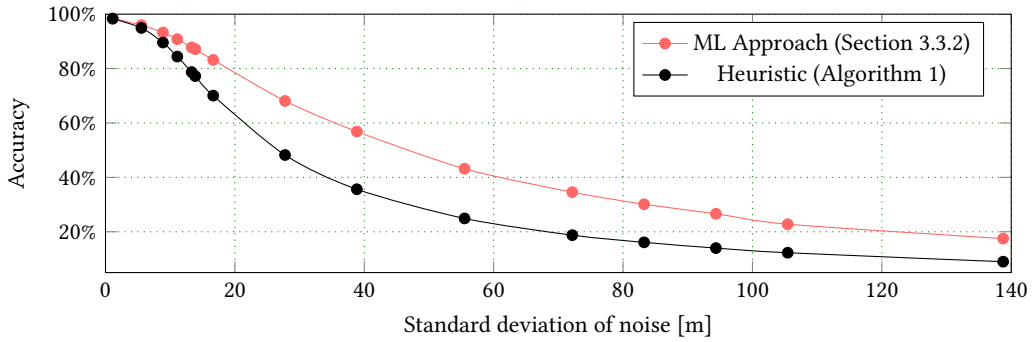


Fig. 7. Accuracy of mapping noisy locations to road segments using heuristic (Algorithm 1) and machine-learning approach (Section 3.3.2). Accuracy is measured as the fraction of locations that are mapped to the correct road segment.

expected, the accuracy of both approaches decreases as the level of noise increases. However, the machine-learning approach performs better than the heuristic algorithm for higher noise levels. For example, with 38.9-meter standard deviation, the former achieves 64.4% accuracy, while the latter attains only 56.9%. Nonetheless, for reasonable noise levels, such as around 10-meter standard deviation, the machine-learning approach can correctly map more than 90% of locations.

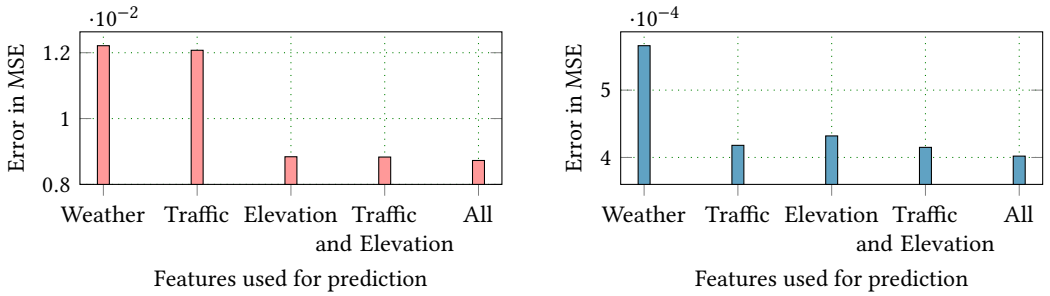
5.2 Comparison of Weather, Traffic, and Elevation Features

For both electric and diesel buses, we have a set of 26 features in each sample, besides energy use as the target feature. The input features are distance traveled, 14 road-type features (primary, secondary, residential, etc.), elevation change, day of week, time of day, 5 weather features (temperature, humidity, visibility, wind-speed, and precipitation), and 2 traffic features (speed ratio and jam factor). Now, we study which of these features are the most useful for predicting energy consumption, and which subset of features results in the lowest prediction error.

Note that there exist algorithmic approaches for feature selection (see, e.g., [28]), which can be applied to a dataset to find a subset of features that minimizes prediction error. Our objective here is different: we would like to provide general guidance on how to collect transit datasets by studying which features are the most useful for different types of vehicles. To this end, we explore various subsets of features as exhaustively as possible, and we report prediction errors for all of these subsets. These results can help transit agencies, which may have limited resources for data collection, to focus on the most useful features for their operations.

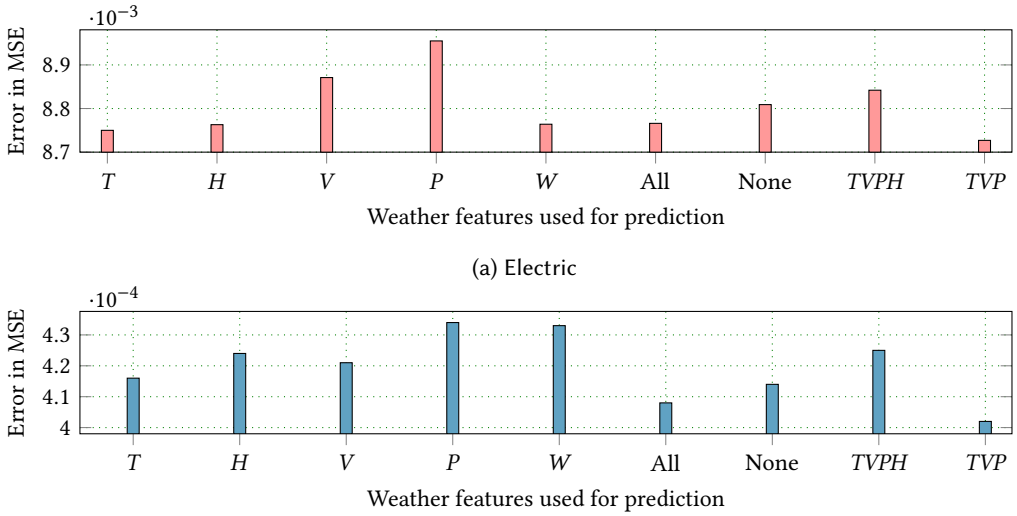
After preparing the samples for both the electric and diesel buses, we randomly split them into training (80%) and test sets (20%). We use the same split ratio in all subsequent experiments. Since artificial neural networks attain the lowest prediction error (see Section 5.4), we compare features based on this model. We include vehicle-level data in all experiments, and try different combinations of weather, elevation, and traffic data.

Figure 8a shows that elevation is by far the most significant feature for electric vehicles. Traffic data does improve prediction, but its impact is much smaller, especially if elevation is already included. This can be explained by regenerative breaking: the energy use of electric vehicles is not impacted by heavy traffic since they do not lose energy due to frequent braking. On the other hand, Figure 8b shows that for diesel vehicles, both elevation and traffic data are significant, and both need to be included for good performance. Finally, we find that weather data has the lowest impact on prediction error for both electric and diesel vehicles.



(a) Electric (b) Diesel

Fig. 8. Mean squared error for energy prediction with various sets of features. Note that electric and diesel energy are measured in different units. The predictions are based on artificial neural networks.



(a) Electric

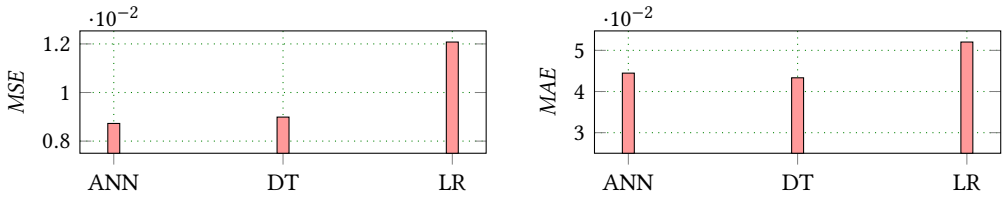
(b) Diesel

Fig. 9. Mean squared error for energy prediction with various weather features and their combinations. The predictions are based on artificial neural networks with all non-weather features included.

5.3 Comparison of Different Weather Features

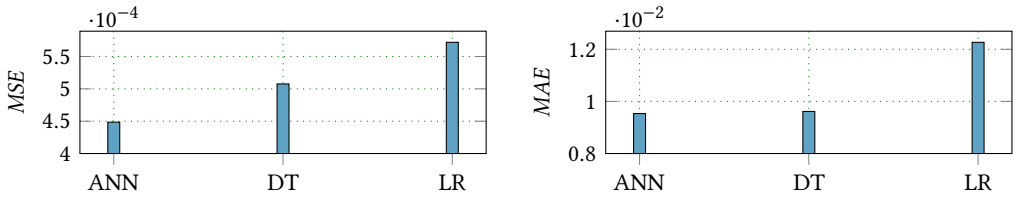
Since weather data has many features, we also present a comparison among various weather features to see which ones help with prediction the most. We consider temperature (T), humidity (H), visibility (V), wind speed (W), and precipitation (P) in this comparison.

Figure 9 shows prediction error with various combinations of weather features (with traffic and elevation always included). For both vehicles, we find temperature (T) to be the most significant weather feature. Humidity (H) and windspeed (W) also significantly reduce the prediction error for electric vehicles (Figure 9a). However, we attain the lowest prediction error for both vehicles using the combination of temperature (T), visibility (V) and precipitation (P).



(a) *Mean squared error* (MSE) (b) *Mean absolute error* (MAE)

Fig. 10. Comparison of different energy prediction models based on *mean squared error* (MSE) and *mean absolute error* (MAE) for electric vehicle samples.



(a) *Mean squared error* (MSE) (b) *Mean absolute error* (MAE)

Fig. 11. Comparison of different energy prediction models based on *mean squared error* (MSE) and *mean absolute error* (MAE) for diesel vehicle samples.

5.4 Comparison of Prediction Models for Samples

We first evaluate the three machine-learning models based on how well they predict energy use for samples. Our samples represent segments of trips that are short in both distance and duration, presenting a challenging problem for prediction.

Figures 10 and 11 show *mean squared error* (MSE) and *mean absolute error* (MAE) for the three models. Based on MSE, the artificial neural network (ANN) outperforms the other two models for both electric and diesel vehicles. However, based on MAE, ANN outperforms decision trees (DT) for diesel vehicles but not for electric vehicles. Note that we optimized all models to minimize MSE, which can explain the slightly inferior performance of ANN for MAE. We have not encountered any overfitting since our training and testing errors were consistent for each model.

5.5 Comparison of Prediction Models for Longer Trips

Finally, we study how well our models perform with respect to predicting energy use for longer trips. First, we divide our time series into longer trips, varying the length of the trips between 10 minutes and 6 hours. For each trip, we generate a set of samples (as described in Section 3), use our models to predict energy use for each sample, and then compare the sum of these predictions to the actual energy use of the trip.

Figure 12 shows the relative prediction error for trips of various lengths. For each length, we plot an average error value computed over many trips. We see that relative prediction error is generally lower for longer trips; this is expected as the individual errors of large numbers of samples cancel each other out with an unbiased prediction model. For diesel vehicles, we find that the ANN outperforms the other models significantly for all trip lengths. On the other hand, for electric vehicles, ANN and DT perform equally well for most trip lengths.

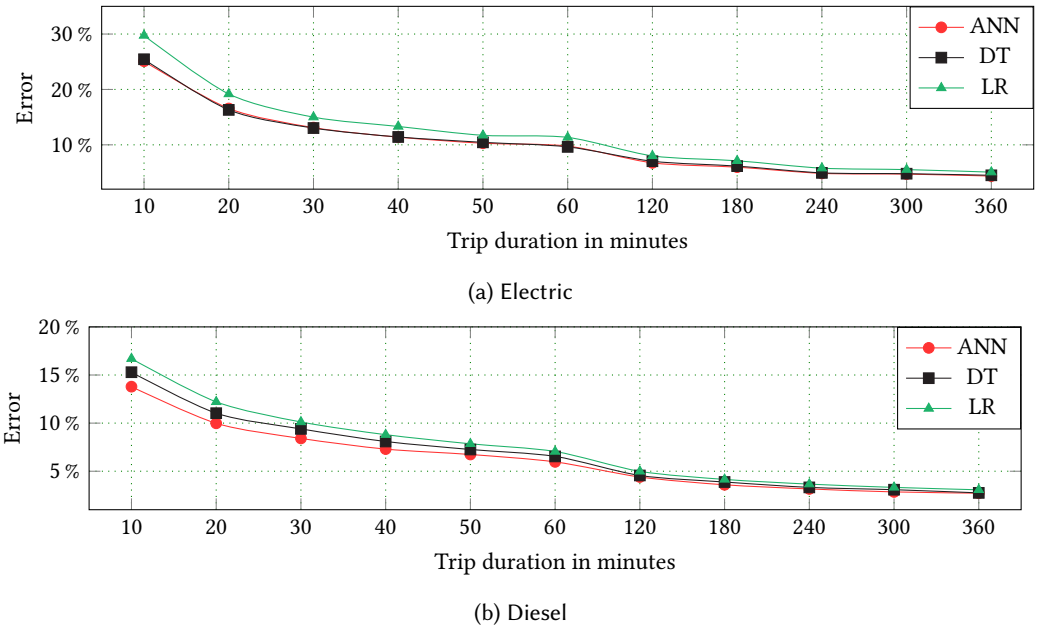


Fig. 12. Energy prediction error for longer trips, consisting of many samples, with neural network (ANN), decision tree (DT), and linear regression (LR).

6 TRANSIT OPTIMIZATION PROBLEM

In the previous sections, we have shown how to predict the energy consumption of EVs and ICEVs by incorporating a variety of processed features into machine-learning models. Based on the numerical results, we can say that our predictions are accurate, especially at the level of trips. In this section, we focus on utilizing these accurate energy consumption predictions for operational optimization. Specifically, building on our ability to predict energy consumption for any given vehicle and any given transit trip, we formulate and solve the problem of minimizing energy consumption by assigning a mixed-vehicle fleet of transit buses to fixed-route transit trips. We first introduce our model of transit vehicles and routes (Section 6.1). Our formulation considers assigning and scheduling for a single day (it may be applied to any number of consecutive days one-by-one), and permits any physically possible re-assignment during the day. Then, we formulate the problem of optimal assignment, specifying the space of feasible solution and our objective function (Section 6.2).

6.1 Transit Model

Vehicles. We consider a transit agency that operates a set of *buses* $\mathcal{V}$. Note that we will use the terms *bus* and *vehicle* interchangeably. Each bus $v \in \mathcal{V}$ belongs to a *vehicle model* $M_v \in \mathcal{M}$ (e.g., 2016 BYD K9S 35-foot battery-electric model), where $\mathcal{M}$ is the set of all vehicle models in operation. We divide the set of vehicle models $\mathcal{M}$ into two disjoint subsets: liquid-fuel models $\mathcal{M}^{\text{gas}}$ (e.g., diesel, hybrid), and electric models $\mathcal{M}^{\text{elec}}$.

Locations. Locations $\mathcal{L}$ include bus stops and garages in the transit network.

Trips. During the day, the agency has to serve a given set of *transit trips* $\mathcal{T}$ using its buses. Based on discussions with our partner agency, CARTA, we assume that all the locations and time

schedules are fixed for all the trips. A bus serving trip $t \in \mathcal{T}$ leaves from trip origin $\lambda_t^{\text{origin}} \in \mathcal{L}$ at time τ_t^{start} and arrives to destination $\lambda_t^{\text{destination}} \in \mathcal{L}$ at time τ_t^{end}. Between $\lambda_t^{\text{origin}}$ and $\lambda_t^{\text{destination}}$, the bus must pass through a series of stops at fixed times; however, since we cannot re-assign a bus during a transit trip, the locations and times of these stops are inconsequential to our model. Finally, we assume that any bus may serve any trip. Note that it would be straightforward to extend our model and algorithms to consider constraints on which buses may serve a trip (e.g., based on passenger capacity).

Non-Service Trips. Besides serving transit trips, buses may also need to drive between trips. For example, if a bus has to serve a trip that starts from a location that is different from the destination of the previous trip, the bus first needs to drive to the origin of the next trip. We will refer to these deadhead trips, which are driven outside of revenue service, as *non-service trips*. We let $T(l_1, l_2)$ denote the non-service trip from location $l_1 \in \mathcal{L}$ to $l_2 \in \mathcal{L}$, and we let $D(l_1, l_2)$ denote the time duration of this non-service trip.

6.2 Solution Space and Objective

Solution Representation. Our goal is to assign a bus to each transit trip. We represent a solution as a *set of assignments* $\mathcal{A}$. For each trip $t \in \mathcal{T}$, a solution assigns exactly one bus $v \in \mathcal{V}$ to serve t; this assignment is represented by the relation $\mathcal{A}_t \rightarrow v$.

Constraints. If a bus is assigned to serve an earlier transit trip t_1 and a later trip t_2, then the duration of the non-service trip from $\lambda_{t_1}^{\text{destination}}$ to $\lambda_{t_2}^{\text{origin}}$ must be less than or equal to the time between $\tau_{t_1}^{\text{end}}$ and $\tau_{t_2}^{\text{start}}$. Otherwise, it would not be able to serve t_2 on time. We formulate this constraint as follows:

$$\forall t_1, t_2 \in \mathcal{T}, \tau_{t_1}^{\text{start}} \leq \tau_{t_2}^{\text{start}}, \mathcal{A}_{t_1} \rightarrow v, \mathcal{A}_{t_2} \rightarrow v : \ \tau_{t_1}^{\text{end}} + D(\lambda_{t_1}^{\text{destination}}, \lambda_{t_2}^{\text{origin}}) \leq \tau_{t_2}^{\text{start}} \tag{6}$$

Note that if the constraint is satisfied by every pair of consecutive trips assigned to a bus, then it is also satisfied by every pair of non-consecutive assigned trips.

Objective. Our objective is to minimize the energy use of the vehicles. This objective can minimize both environmental impact and operating costs by imposing the appropriate cost factors on the energy use of liquid-fuel and electric vehicles.

First, we let $\mathcal{N}(\mathcal{A}, v)$ denote the set of all non-service trips that bus v needs to complete according to the set of assignments $\mathcal{A}$ (i.e., for each consecutive pair of transit trips t_1, t_2 that $\mathcal{A}$ assigns to v, there is a non-service trip $T(\lambda_{t_1}^{\text{destination}}, \lambda_{t_2}^{\text{origin}})$ in $\mathcal{N}(\mathcal{A}, v)$). We can formally express $\mathcal{N}(\mathcal{A}, v)$ as follows:

$$\begin{aligned}\mathcal{N}(\mathcal{A}, v) = \{T(\lambda_{t_1}^{destination}, \lambda_{t_2}^{origin}) | t_1, t_2 \in \mathcal{T} \wedge v \in \mathcal{V} \wedge \tau_{t_1}^{\text{start}} \leq \tau_{t_2}^{\text{start}} \wedge \mathcal{A}_{t_1} \rightarrow v \wedge \mathcal{A}_{t_2} \rightarrow v \wedge \\ (\forall t \in \mathcal{T} \wedge ((\tau_{t_1}^{\text{start}} \leq \tau_t^{\text{start}} \leq \tau_{t_2}^{\text{start}} \wedge \neg \mathcal{A}_t \rightarrow v) \vee (\tau_t^{\text{start}} < \tau_{t_1}^{\text{start}}) \vee (\tau_t^{\text{start}} > \tau_{t_2}^{\text{start}}))\}\end{aligned} \tag{7}$$

Next, we let $E(v, t)$ denote the amount of energy used by bus v to drive a transit or non-service trip t. Then, we can express the amount of energy used by bus v for all trips in assignment $\mathcal{A}$ as

$$e(\mathcal{A}, v) = \sum_{t \in \mathcal{N}(\mathcal{A}, v)} E(v, t) + \sum_{t \in \mathcal{T}: \ \mathcal{A}_t \rightarrow v} E(v, t) \tag{8}$$

Finally, let K^{gas} and K^{elec} denote the unit costs of energy use for liquid-fuel and electric vehicles, respectively. Then, we can express our objective as

$$\min_{\mathcal{A}} \sum_{v \in \mathcal{V}: \ M_v \in \mathcal{M}^{\text{gas}}} K^{\text{gas}} \cdot e(\mathcal{A}, v) + \sum_{v \in \mathcal{V}: \ M_v \in \mathcal{M}^{\text{elec}}} K^{\text{elec}} \cdot e(\mathcal{A}, v) \tag{9}$$

7 TRANSIT OPTIMIZATION ALGORITHMS

First, we present an integer program to find optimal solutions (Section 7.1), whose linear relaxation we will also use as a lower bound in our numerical evaluation. Since the integer program does not scale well, we will also introduce efficient heuristic (Section 7.2) and genetic algorithms (Section 7.3).

7.1 Integer Program

Variables. Our integer program has two sets of binary variables. First, $a_{v,t} = 1$ (or 0) indicates that trip t is assigned to bus v (or that it is not). Second, $m_{v,t_1,t_2} = 1$ (or 0) indicates that bus v takes the non-service trip between a pair t_1 and t_2 of transit trips or not.

Constraints. First, we ensure that every transit trip is served by exactly one bus:

$$\forall t \in \mathcal{T} : \sum_{v \in \mathcal{V}} a_{v,t} = 1$$

Next, we ensure that Equation (6) is satisfied. We let $F(t_1, t_2)$ be *true* if a pair t_1, t_2 of transit trips satisfy Equation (6) (in the appropriate temporal order); and let it be *false* otherwise. Then, we can express the constraint as follows:

$$\forall v \in \mathcal{V}, \forall t_1, t_2, \neg F(t_1, t_2) : a_{v,t_1} + a_{v,t_2} \leq 1$$

Finally, when a bus v is assigned to both trips t_1 and t_2, but it is not assigned to any other transit trips in between (i.e., if trips t_1 and t_2 are consecutive assignments), then bus v needs to make a non-service trip:

$$m_{v,t_1,t_2} \geq a_{v,t_1} + a_{v,t_2} - 1 - \sum_{t \in \mathcal{T}:\ \tau^{\text{start}}_{t_1} \leq \tau^{\text{start}}_{t} \leq \tau^{\text{start}}_{t_2}} a_{v,t}$$

Note that if trips t_1 and t_2 have the same location, then the non-service trip will take zero time and energy by definition.

Objective. We can express Equation (9) as minimizing

$$\sum_{v \in \mathcal{V}} K^{M_v} \left[\sum_{t \in \mathcal{T}} a_{v,t} \cdot E(v,t) + \sum_{t_1, t_2 \in \mathcal{T}} m_{v,t_1,t_2} \cdot E(v, T(t_1, t_2)) \right]$$

where K^{M_v} is K^{elec} if $M_v \in \mathcal{M}^{\text{elec}}$ and K^{gas} otherwise.

Complexity. The integer program contains both variables and constraints in the order of $O(|\mathcal{V}| \cdot |\mathcal{T}|^2)$. Note that regardless, an integer programming solver may take exponential time in the input to find an optimal solution.

7.2 Heuristic Algorithms

Next, we introduce two polynomial-time heuristic algorithms.

Feasibility. Both heuristic algorithms need to ensure that buses are assigned to trips without violating Equation (6). We let **FEASIBLE** be a subroutine that checks whether the transit trip t can be assigned to vehicle v without violating Equation (6). In other words, **FEASIBLE** checks if for every t such that $\mathcal{A}_t \rightarrow v$, trips t and t' satisfy Equation (6) (in the appropriate temporal order).

Heuristic by Location (Heuristic L):. The motivation of this approach is to minimize energy costs by reducing non-service trips. Algorithm 3 first groups together all trips that share an origin or destination location. Then, it iterates over the groups in a random order. For each group, it sorts the trips according to their start times, and then assigns vehicles to the trips one-by-one, always choosing a feasible vehicle at random. The time complexity of the algorithm is $O(|\mathcal{V}| \cdot |\mathcal{T}| \cdot \log |\mathcal{T}|)$.

Algorithm 3: HEURISTIC BY LOCATION$(\mathcal{V}, \mathcal{T})$

```
stop_pairs ← {}
for t ∈ 𝒯 do
    stop_pair ← {λ_t^origin, λ_t^destination}
    𝒯′ ← stop_pairs.get(stop_pair)
    𝒯′ ← 𝒯′ ∪ {t}
    stop_pairs ← stop_pairs ∪ {stop_pair, 𝒯′}
stop_pairs' ← shuffle(stop_pairs)
for stop_pair ∈ stop_pairs' do
    𝒯′ ← stop_pairs.get(stop_pair)
    for t ∈ sortedByTime(𝒯′) do
        𝒱′ ← shuffle(𝒱)
        for v ∈ 𝒱′ do
            feasible ← FEASIBLE(𝒜, v, t)
            if feasible then
                𝒜 ← 𝒜 ∪ ⟨v, t⟩
Result: 𝒜
```

Algorithm 4: HEURISTIC BY BUS$(\mathcal{V}, \mathcal{T})$

```
for t ∈ sortedByTime(𝒯) do
    𝒱′ ← shuffle(𝒱)
    for v ∈ 𝒱′ do
        feasible ← FEASIBLE(𝒜, v, t)
        if feasible then
            𝒜 ← 𝒜 ∪ ⟨v, t⟩
Result: 𝒜
```

Heuristic by Bus (Heuristic B):. The motivation of this approach is to optimize the utilization of every bus. Algorithm 4 first sorts all transit trips based on their start time. Then, it iterates over the buses in a random order. For each bus, it tries to assign every trip to the bus, going over the trips one-by-one. The time complexity of this approach is $O(|\mathcal{V}| \cdot |\mathcal{T}| \cdot \log |\mathcal{T}|)$.

7.3 Genetic Algorithm

Building on the two heuristic algorithms, we introduce a genetic algorithm, which uses the heuristic algorithms for its initial population $\mathcal{P}^0$, but improves upon them using iterative random search. The time complexity of each iteration is $O(|\mathcal{V}| \cdot |\mathcal{T}| \cdot |\mathcal{P}^0| \cdot \log |\mathcal{T}|)$.

Initialization. The genetic algorithm starts with a fixed-size initial population $\mathcal{P}^0$ of solutions. We generate each member of the initial population using the two heuristic algorithms.

Selection. The algorithm computes the energy cost of each solution in the current population $\mathcal{P}^i$, and then chooses the N lowest-cost solutions as the basis for the next generation of the population. To create the next generation, the algorithm performs mutation and crossover.

Mutation (Algorithm 5). Mutation first selects one solution $\mathcal{A}$ at random from the basis of the next generation. Then, it selects two buses v_1 and v_2 at random, and selects a transit trip t_1 at

Algorithm 5: MUTATION($\mathcal{P}^i$)

$\mathcal{A} \leftarrow$ **random**($\mathcal{P}^i$)
mCount $\leftarrow$ **max**(1, $|\mathcal{A}| \cdot$ mutation_prob)
for 1 $\rightarrow$ *mCount* **do**
  $v_1, v_2 \leftarrow$ **random**($\mathcal{V}$)
  $t_1 \leftarrow$ **random**($\mathcal{A}, v_1$)
  $t_2 \leftarrow$ **random**($\mathcal{A}, v_2$)
  $\mathcal{A} \leftarrow \mathcal{A} - \{v_1, t_1\} - \{v_2, t_2\}$
  feasible$_1 \leftarrow$ **FEASIBLE**($\mathcal{A}, v_1, t_2$)
  feasible$_2 \leftarrow$ **FEASIBLE**($\mathcal{A}, v_2, t_1$)
  feasible $\leftarrow$ feasible$_1 \wedge$ feasible$_2$
  if *feasible* **then**
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{v_1, t_2\} \cup \{v_2, t_1\}$
Result: $\mathcal{A}$

Algorithm 6: CROSSOVER($\mathcal{P}^i$)

$\mathcal{P}^c \leftarrow \emptyset$
$\mathcal{A}^1 \leftarrow$ **random**($\mathcal{P}^i$)
$\mathcal{A}^2 \leftarrow$ **random**($\mathcal{P}^i - \mathcal{A}^1$)
crossover_point $\leftarrow$ **random**(0, 1)
$\mathcal{A}^{1a}, \mathcal{A}^{1b} \leftarrow$ **split**($\mathcal{A}^1$, crossover_point)
$\mathcal{A}^{2a}, \mathcal{A}^{2b} \leftarrow$ **split**($\mathcal{A}^2$, crossover_point)
$\mathcal{A}^{1'} \leftarrow$ **merge**($\mathcal{A}^{1a}, \mathcal{A}^{2b}$)
$\mathcal{A}^{2'} \leftarrow$ **merge**($\mathcal{A}^{2a}, \mathcal{A}^{1b}$)
$\mathcal{P}^c \leftarrow$ **select**($\{\mathcal{A}^1, \mathcal{A}^2, \mathcal{A}^{1'}, \mathcal{A}^{2'}\}$, 2)
Result: $\mathcal{P}^c$

random from the trips assigned to v_1 by $\mathcal{A}$, and trip t_2 at random from the trips assigned to v_2. If the assignments of trips t_1 and t_2 can be switched between buses v_1 and v_2 without violating any constraints, then it switches them. The algorithm repeats from selecting two buses at random, until a desired number of mutation attempts is reached.

Crossover (Algorithm 6). Crossover first selects two solutions $\mathcal{A}_1$ and $\mathcal{A}_2$ at random from the basis of the next generation, and chooses a crossover point at random from (0, 1), which is used to divide the day into two parts at random. Then, it splits each solution $\mathcal{A}_i$ into two subsets of assignments based on the crossover point: assignments that belong to the first part of the day form the first subset, while assignment that belong to the second part form the second subset. Next, it merges the four parts by swapping the parts of the two solutions. Finally, it selects the two lowest-cost solutions out of the initial solutions and the merged solution, automatically discarding infeasible ones.

Iteration and Termination. In each iteration, the genetic algorithm generates a new generation of solutions based on selection, mutation, and crossover. The algorithm terminates when there is no decrease in the minimum energy cost over a number of new generations, which indicates that algorithm converged.

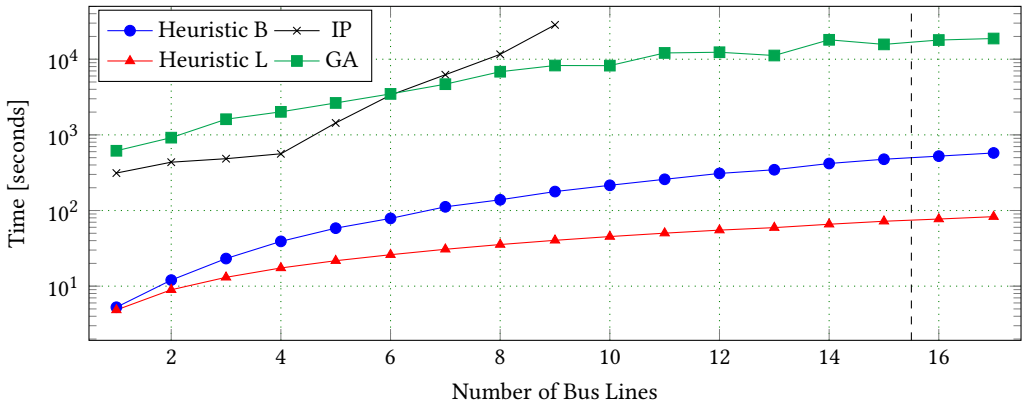


Fig. 13. Computation times for assignments using Heuristic B, Heuristic L, genetic algorithm (GA), and integer program (IP) for smaller problem instances (1 to 17 bus-lines serving 15 transit trips per bus-line). Please note the logarithmic scale on the vertical axis.

8 NUMERICAL RESULTS ON TRANSIT OPTIMIZATION

To evaluate the computational approaches that we proposed for solving the transit optimization problem (i.e., IP, heuristics, and genetic algorithm), we build on the energy consumption predictors that we introduced earlier. First, we obtain and process real-world data from CARTA, our partner transit agency in Chattanooga, TN, as described in Sections 2 and 3; and we train neural-network based energy consumption predictors as described in Sections 4 and 5. Then, we use these predictors to estimate the energy cost of each transit and non-service trip for both EVs and ICEVs. Finally, we combine these energy-cost estimates with the actual transit schedule of the agency to create instances of the transit optimization problem, which we use to evaluate the proposed computational approaches.

8.1 Settings and Data

Public Transit Schedule. We obtain the schedule of the transit agency in GTFS format, which includes all trips, time schedules, bus stop locations, etc. Trips are organized into 19 bus lines (i.e., bus routes) throughout the city. Among those 19 bus lines, 2 bus lines are dedicated for shuttle services; thus, we ignore them and consider only the remaining 17 fixed-route bus lines in our numerical analysis. For our numerical evaluation, we consider trips served during weekdays (Monday to Friday) since these are the busiest days. Each weekday, the agency must serve around 850 trips based on 17 bus lines using 3 electric buses of model BYD K9S, and 50 diesel and hybrid buses.

Non-Service Trips. Since non-service trips are not part of the transit schedule, we need to plan their routes and estimate their durations. For this, we use the Google Directions API, which we query for all 1806 possible non-service trips (i.e., for every pair of locations in the network) for each 1-hour interval of a selected weekday from 5am to 11pm. The response to each query includes an estimated duration as well as a detailed route.

Energy Costs. Finally, we take series of locations along the route of each transit and non-service trip, combine them with our other data sources as described in Section 3, and then feed them into our trained energy-use predictors to estimate the energy usage and cost of each trip.

8.2 Results

We first study how well our algorithms scale with increasing problem sizes. To this end, we measure the computation times of our algorithms with 1 to 17 bus lines (selected at random from real CARTA bus lines), and 15 trips selected at random for each line. Since two of the bus lines have less than 15 trips (10 and 14 trips, respectively), for cases with 16 to 17 bus lines, we include all trips for these lines. For cases with 11 to 17 bus lines, we assign the entire vehicle fleet of CARTA, which consists of 3 electric and 50 liquid-fuel buses. For cases with fewer bus lines, we assume that the agency has only 5 times as many liquid-fuel buses as bus lines and has 3 electric buses. We solve the integer program (IP) using IBM CPLEX. We run all algorithms on a machine with a Xeon E5-2680 CPU, which has 28 cores, and 128 GB of RAM. Figure 13 shows the average computation time for the IP, the Heuristic B algorithm, the Heuristic L algorithm, and the genetic algorithm based on 8 random instances (with different sets of bus lines and trips) for each problem size. As expected, the time to solve the IP is significantly higher than the running time of the Heuristic algorithms for all problem sizes. Further, the solution time of the IP increases rapidly with the problem size and becomes significantly higher than the running time of the genetic algorithm for 9 bus lines. In fact, solving the IP for 10 or more bus lines was infeasible on our test machine in terms of memory and running time. On the other hand, the genetic algorithm scales very well computationally despite having a fairly high constant factor in its running time.

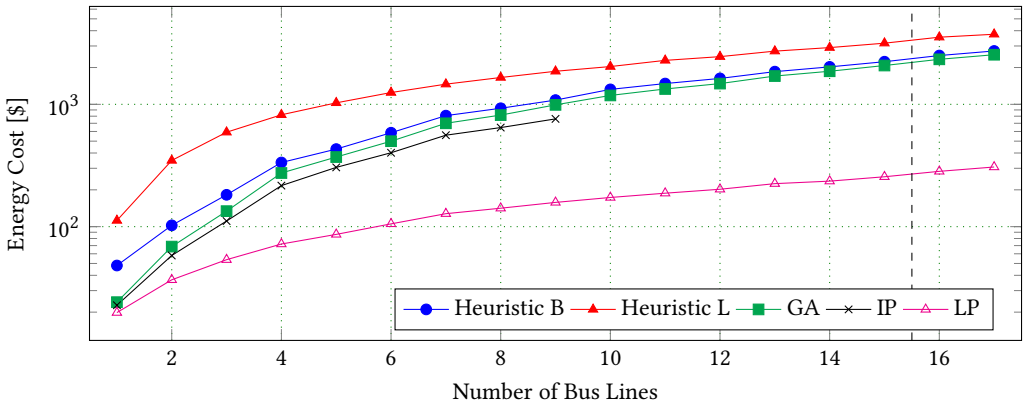


Fig. 14. Energy costs for assignments using Heuristic B, Heuristic L, genetic algorithm (GA), integer program (IP), and linear program (LP) for smaller problem instances (1 to 17 bus-lines serving 15 transit trips per bus-line). Please note the logarithmic scale on the vertical axis.

Next, we evaluate the performance of our algorithms with respect to solution quality, that is, with respect to energy cost. We use the exact same setting as in the previous experiment. Since solving the IP is infeasible for larger problem instances, we include its linear relaxation (LP) in this evaluation. While the LP does not provide a feasible solution, it can serve as a lower bound on the energy cost, to which we can compare our other algorithms. Figure 14 shows the average energy cost for the IP, its linear relaxation (LP), the Heuristic B algorithm, the Heuristic L algorithm, and the genetic algorithm based on 8 random instances (with different sets of bus lines and trips) for each problem size. In this figure, we compare the solution quality of the two heuristic algorithms and the genetic algorithm to the optimal solutions obtained from the IP. Since the IP is unable to scale for larger problem instances due to memory limitations and longer computation times, we plot the optimal solutions based on IP for instances with up to 9 bus lines. The figure shows that

our efficient algorithms perform well: the genetic algorithm performs almost as well as the IP, and the heuristic algorithms perform only slightly worse. For larger instances, the ratios between the performance of the LP and our heuristic and genetic algorithms remain stable, which suggests that our algorithms still perform close to optimal.

Finally, we compute assignments for the complete daily schedule with 3 electric and 50 liquid-fuel buses using heuristic and genetic algorithms for 5 different sample days. We were able to assign the complete daily schedule using Heuristic B algorithm in around 3 minutes, resulting in average energy cost of $8,290. Meanwhile, the genetic algorithm runs for around 1 day (around 2,000 iterations) and results in average energy cost of $8,083. Since an agency might need to find a new assignment every day (e.g., because some buses are unavailable due to maintenance), the heuristic algorithm can be a better option.

9 RELATED WORK

In this section, we provide an overview of related work on predicting the energy use of vehicles (Section 9.1), mapping noisy locations to roads (Section 9.2), and assigning and scheduling transit vehicles (Section 9.3).

9.1 Energy Use Prediction

Our study is most closely related to the work of Cauwer et al. and of Wickramanayake and Bandara. Cauwer et al. [12] use a cascade of ANN and multiple linear regression models as a data-driven energy-consumption prediction method for EVs. Their study uses vehicle monitoring data for two types of vehicles as time series of tuples with location, vehicle speed, and energy-consumption information, such as battery voltage, current, and SoC. Their dataset also includes road-network data, weather data, and an altitude map. Our approach has some similarity to this study. However, we also use traffic data in our model, which we find to be very helpful for diesel prediction. Wickramanayake and Bandara [45] assess three different techniques for predicting the fuel consumption of a long-distance public bus. Their time series of tuples include GPS location, bearing, elevation, distance travelled, speed, acceleration, ignition status, battery voltage, fuel level, and fuel consumption. The authors compare the performance among two ensemble models, random forest and gradient boosting, and one ANN model. However, their study lacks critical parameters, such as road information, traffic, weather, etc.

Perrotta et al. [38] compare the performance of SVM, RF, and ANN in modeling fuel consumption of a large fleet of trucks. Their features include gross vehicle weight, speed, acceleration, geographical position, torque percentage, revolutions of the engine, activation of cruise control, use of brakes and acceleration pedal, measurement of travelled distance, fuel consumption. The study also integrates some road characteristics. Based on comparison of the RMSE, MAE, and R^2 scores of the prediction, RF gives the best performance. Nageshrao et al. [33] model the energy consumption of electric buses based on time-dependent factors such as ambient temperature and speed, battery capacity, total mass, battery parameters, etc. They use a NARX based ANN time series predictor to predict the state of charge of the battery. Gao et al. [14] discuss an adaptive wavelet neural network (WNN) based energy prediction. Their study uses features such as day type, temperature, rainfall, the travelled distance, and clarity of the atmosphere. The study groups the trip days based on similar attributes, using Grey Relational Analysis (GRA) and then implements the Adaptive WNN.

There also exist prior efforts to utilize spatial and temporal data to model the energy consumption of bus transit networks and to estimate costs. Wang et al. [43] collect GPS records of vehicle position and vehicle status and GPS location details of bus stops and bus transaction data of passenger fares. Wang et al. [44] and Hassold et al. [19] obtain historical transaction data. Li et al. [30], Paul et al. [37] and Li et al. [29] gather the distance of each trip in the bus transit network schedule. Santos

et al. [39] and Zhou et al. [48] make use of publicly available resources, which provide details of average energy costs per unit of energy, emission rate per unit of energy consumption, and consumption rates per unit distance. In contrast to previous efforts, we collect traffic, elevation and weather data in addition to vehicle position and vehicle status data.

9.2 Mapping Locations to Roads

There are several studies on mapping GPS observations to a road network. These generally apply machine learning approaches or Hidden Markov Model (HMM) based algorithms.

Hashemi and Karimi [17] propose an ANN-based approach to reduce the horizontal error in locations obtained from GPS. Specifically, the proposed ANN modifies the GPS locations by attempting to remove the measurement error. Hashemi and Karimi [18] also introduce a weight-based map-matching algorithm, which maps these modified locations to road segments. Their map-matching algorithm attempts to identify the correct segment based on distance between the location and each road segment, direction of spatial difference, and similarity between the direction of each road segment and the heading of the vehicle at the location. Our data preparation steps for the location-mapping decision tree regression model (Section 3.3.2) are similar to their weight-based approach. However, our approach also uses the average, maximum, and minimum distances between a location and the candidate road segments of the preceding and following vehicle locations, road types, as well as the number of times a segment appears in the candidate sets of the preceding and following vehicle locations.

Newson and Krumm [34] use HMM with candidate routes chosen from within a 200-meter radius. After calculating the emission and transition probability of each route, HMM chooses the route with the highest probability. They evaluate their system based on various levels of noise and sampling rate. The algorithm is proven to be useful for noise levels as high as 50-meters. Mohamed et al. [31] implement a similar incremental HMM algorithm where a number of pre-processing modules reduce the noise and spareness in the input data. Goh et al. [15] also use HMM with candidate routes set to be within a 50-meter radius around a GPS location. In addition to the model, they use a Variable Sliding Window (VSW) algorithm that performs backtracking on the updated Markov chain.

9.3 Energy Use Optimization

Prior research efforts have explored multiple approaches for scheduling public transit and assigning vehicles. Wang et al. [43] design a real-time charge scheduling system, called bCharge, for electric bus fleets. To evaluate their system, they use a real-world streaming dataset from Shenzhen, China, which includes GPS location, bus stop, bus transaction, charging station, and electricity rate data. Their approach is based on Markov decision processes and considers both energy costs and bus fare revenues; however, it is limited to bus transit networks operating only EVs, and it modifies the existing schedule, which is not always feasible in practice. In contrast, we consider bus transit networks operating mixed fleets of EVs and ICEVs, and keep the schedule intact.

Murphey et al. [32] propose the ML_EMO_HEV framework for energy management optimization of hybrid-electric vehicles. Their framework first uses a ANN to model the road environment of a driving trip as a sequence of different roadway types and traffic congestion levels. Then, it uses an additional ANN to model the driver's instantaneous reaction to the driving environment. Finally, the framework uses an additional set of ANN to emulate the optimal energy management strategy. Li et al. [29] study scheduling electric buses considering the range constraints of EVs. They apply a branch-and-price approach for smaller instances of the problem and use a heuristic based column-generation approach with variable fixing for larger problem instances. They calculate the distance of deadheading trips by mapping a stop to the nearest intersection using point-to-curve

matching, then solve one-to-one shortest-path problems using Dijkstra's algorithm, based on the road geometry data obtained from NavTeq.

Research efforts such as Zhou et al. [48], Li et al. [30], Santos et al. [39], Paul et al. [37] and Li et al. [29] consider bus transit networks operating mixed fleets of vehicles. Zhou et al. [48] improve an existing iterative neighbour search algorithm, where they obtain the initial solution using local search strategies, such as SHIFT, 2opt* and Block, and enhance the solution based on simulated annealing. Santos et al. [39] develop evolutionary algorithms to minimize energy and emission costs, while ensuring that service level is unchanged. Paul et al. [37] implement k-greedy and genetic algorithms for minimizing energy costs by optimally assigning mixed fleets of vehicles to transit trips. Santos et al. [39] use fixed costs, emission and consumption rates for different types of vehicles, while Paul et al. [37] assume that energy costs are fixed per unit distance without considering spatial and temporal factors. Unlike the previous research efforts, we derive realistic energy estimates using our energy predictors based on vehicle locations, traffic, elevation, and weather.

Li et al. [30] apply mixed-integer linear programming and time-space bus-flow networks to schedule mixed fleets for bus transit networks, which operate EVs that have limited service range. Li et al. [30] and Kliewer et al. [24, 25] allow a bus to serve multiple lines instead of limiting it to a single line, which can reduce energy costs; we also incorporate a similar approach. Li et al.'s [30] approach groups together trips as origin-destination pairs, which also reduces the energy costs; we again explore a similar approach.

Similar to bus transit networks, operating transit services with minimum energy consumption is a challenging problem in public commuter services, such as ride-sharing, car-sharing, and car-pooling. Chen et al. [9], Korkas et al. [27], and Hulagu et al. [21] apply various approaches to schedule vehicles for commuters and to assign electric vehicles for charging. Chen et al. [9] map the road network using graph theory and schedule the vehicles using quadratic programming. Korkas et al. [27] propose a charging scheduling algorithm for dynamic transit vehicles using multi-modal approximate dynamic programming. Hulagu et al. [21] solve the electric-vehicle scheduling problem for dynamic transit vehicles using integer programming. These approaches are applicable to dynamic transit environments, where there is no fixed schedule and time constraints are flexible to a certain extent; on the other hand, we focus on the problem of assigning transit trips to a mixed-fleet of vehicles without altering the fixed transit schedule.

10 DISCUSSION AND CONCLUSION

We presented a framework for the data-driven prediction and optimization of the energy use of electric and internal-combustion vehicles, which we evaluated on real-world data collected from the transit fleet of CARTA, the public transit agency of Chattanooga, TN. Our results show that it is possible to collect, aggregate, and process heterogeneous transit data effectively. We found that generally, artificial neural networks perform best for predicting energy use. For both diesel and electric buses, we achieve best results using 21 predictor variables: travel distance, 14 road-type features, elevation change, 3 weather features, and 2 traffic features. Further, we found that relative prediction error is lower for longer trips, which facilitates the long-term planning of transit operations. We also studied the problem of minimizing energy costs through assigning vehicles to fixed-route transit trips. Although this problem is challenging computationally, we demonstrated that our heuristic and meta-heuristic algorithms scale well and provide near optimal solutions.

Acknowledgment

We thank the anonymous reviewers of the conference version of our paper [8] as well as the anonymous reviewers of our journal submission for their valuable feedback and suggestions.

This material is based upon work supported by the Department of Energy, Office of Energy Efficiency and Renewable Energy (EERE), under Award Number DE-EE0008467. Disclaimer: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

This work was completed in part with resources provided by the Research Computing Data Core at the University of Houston and in part through cloud research credits provided by Google.

REFERENCES

[1] 2018. *Traffic Message Channel.* https://wiki.openstreetmap.org/wiki/TMC
[2] 2019. *Apache BookKeeper – A scalable, fault-tolerant, and low-latency storage service optimized for real-time workloads.* https://bookkeeper.apache.org/
[3] 2019. *Apache Pulsar – An open-source distributed pub-sub messaging system.* https://pulsar.apache.org/
[4] 2019. *Apache ZooKeeper.* https://zookeeper.apache.org/
[5] 2019. *MongoDB – The database for modern applications.* https://www.mongodb.com/
[6] 2020. *Google Cloud Platform.* https://cloud.google.com/
[7] 2020. *VMware.* https://www.vmware.com/
[8] Afiya Ayman, Michael Wilbur, Amutheezan Sivagnanam, Philip Pugliese, Abhishek Dubey, and Aron Laszka. 2020. Data-Driven Prediction of Route-Level Energy Use for Mixed-Vehicle Transit Fleets. In *Proceedings of the 6th IEEE International Conference on Smart Computing (SMARTCOMP).*
[9] Tao Chen, Bowen Zhang, Hajir Pourbabak, Abdollah Kavousi-Fard, and Wencong Su. 2016. Optimal routing and charging of an electric vehicle fleet for high-efficiency dynamic transit systems. *IEEE Transactions on Smart Grid* 9, 4 (2016), 3563–3572.
[10] François Chollet et al. 2015. Keras. https://keras.io.
[11] Dark Sky. 2019. API Documentation. https://darksky.net/dev/docs, May 31st, 2020.
[12] Cedric De Cauwer, Wouter Verbeke, Thierry Coosemans, Saphir Faid, and Joeri Van Mierlo. 2017. A data-driven method for energy consumption prediction and energy-efficient routing of electric vehicles in real-world conditions. *Energies* 10, 5 (2017), 608.
[13] EIA. 2018. U.S. Energy Information Administration: Use of energy explained – Energy use for transportation (2018). https://www.eia.gov/energyexplained/use-of-energy/transportation.php, Accessed: May 31st, 2020.
[14] Yajing Gao, Shixiao Guo, Jiafeng Ren, Zheng Zhao, Ali Ehsan, and Yanan Zheng. 2018. An electric bus power consumption model and optimization of charging scheduling concerning multi-external factors. *Energies* 11, 8 (2018), 2060.
[15] Chong Yang Goh, Justin Dauwels, Nikola Mitrovic, Muhammad Tayyab Asif, Ali Oran, and Patrick Jaillet. 2012. Online map-matching based on hidden markov model for real-time traffic sensing applications. In *2012 15th International IEEE Conference on Intelligent Transportation Systems.* IEEE, 776–781.
[16] Mordechai Haklay and Patrick Weber. 2008. OpenStreetMap: User-generated street maps. *IEEE Pervasive Computing* 7, 4 (2008), 12–18.
[17] Mahdi Hashemi and Hassan A Karimi. 2016. A machine learning approach to improve the accuracy of GPS-based map-matching algorithms. In *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI).* IEEE, 77–86.
[18] Mahdi Hashemi and Hassan A Karimi. 2016. A weight-based map-matching algorithm for vehicle navigation in complex urban networks. *Journal of Intelligent Transportation Systems* 20, 6 (2016), 573–590.
[19] Stephan Hassold and Avishai Ceder. 2014. Improving Energy Efficiency of Public Transport Bus Services by Using Multiple Vehicle Types. *Transportation Research Record* 2415, 1 (2014), 65–71.
[20] HERE. 2020. HERE Maps API. https://developer.here.com/, Accessed: May 31st, 2020.
[21] Selin Hulagu and Hilmi Berk Celikoglu. 2020. An Electric Vehicle Routing Problem With Intermediate Nodes for Shuttle Fleets. *IEEE Transactions on Intelligent Transportation Systems* (2020).

[22] Muhammad Hussain Iqbal and Tariq Rahim Soomro. 2015. Big data analysis: Apache Storm perspective. *International journal of computer trends and technology* 19, 1 (2015), 9–14.
[23] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[24] Natalia Kliewer, Vitali Gintner, and Leena Suhl. 2008. Line change considerations within a time-space network based multi-depot bus scheduling model. In *Computer-aided Systems in Public Transport*. Springer, 57–70.
[25] Natalia Kliewer, Taieb Mellouli, and Leena Suhl. 2006. A time–space network based exact optimization model for multi-depot bus scheduling. *European journal of operational research* 175, 3 (2006), 1616–1627.
[26] Xiangfu Kong and Jiawen Yang. 2019. A scenario-based map-matching algorithm for complex urban road network. *Journal of Intelligent Transportation Systems* 23, 6 (2019), 617–631.
[27] Christos D Korkas, Simone Baldi, Shuai Yuan, and Elias B Kosmatopoulos. 2017. An adaptive learning-based approach for nearly optimal dynamic charging of electric vehicle fleets. *IEEE Transactions on Intelligent Transportation Systems* 19, 7 (2017), 2066–2075.
[28] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. 2017. Feature selection: A data perspective. *Comput. Surveys* 50, 6 (2017), 1–45.
[29] Jing-Quan Li. 2014. Transit bus scheduling with limited energy. *Transportation Science* 48, 4 (2014), 521–539.
[30] Lu Li, Hong K Lo, and Feng Xiao. 2019. Mixed bus fleet scheduling under range and refueling constraints. *Transportation Research Part C: Emerging Technologies* 104 (2019), 443–462.
[31] Reham Mohamed, Heba Aly, and Moustafa Youssef. 2014. Accurate and efficient map matching for challenging environments. In *Proceedings of the 22nd ACM SIGSPATIAL international conference on advances in geographic information systems*. 401–404.
[32] Yi Lu Murphey, Jungme Park, Zhihang Chen, Ming L Kuang, M Abul Masrur, and Anthony M Phillips. 2012. Intelligent hybrid vehicle power control–Part I: Machine learning of optimal vehicle power. *IEEE Transactions on Vehicular Technology* 61, 8 (2012), 3519–3530.
[33] Subramanya P Nageshrao, Jubin Jacob, and Steven Wilkins. 2017. Charging cost optimization for EV buses using neural network based energy predictor. *IFAC-PapersOnLine* 50, 1 (2017), 5947–5952.
[34] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 336–343.
[35] Office of Transportation and Air Quality. 2019. *Fast Facts: U.S. Transportation Sector Greenhouse Gas Emissions 1990–2017*. Technical Report EPA-420-F-19-047. https://nepis.epa.gov/Exe/ZyPDF.cgi?Dockey=P100WUHR.pdf
[36] Aleksandra O'Donovan, James Frith Analyst, and Colin McKerracher. 2018. Electric Buses in Cities: Driving Towards Cleaner Air and Lower CO2. Bloomberg New Energy Finance, https://data.bloomberglp.com/professional/sites/24/2018/05/Electric-Buses-in-Cities-Report-BNEF-C40-Citi.pdf.
[37] Topon Paul and Hisashi Yamada. 2014. Operation and charging scheduling of electric buses in a city bus route network. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2780–2786.
[38] Federico Perrotta, Tony Parry, and Luis C Neves. 2017. Application of machine learning for fuel consumption modelling of trucks. In *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 3810–3815.
[39] Diogo Santos, Zafeiris Kokkinogenis, Jorge Freire de Sousa, Deborah Perrotta, and Rosaldo JF Rossetti. 2016. Towards the integration of electric buses in conventional bus fleets. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 88–93.
[40] Scikit-learn Developers. [n.d.]. Sklearn.Tree.DecisionTreeRegressor. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html, Accessed: May 31st, 2020.
[41] Tennessee Department of Finance and Administration. 2019. *Elevation Data*. https://www.tn.gov/finance/sts-gis/gis/data.html
[42] USGS. 2019. *The National Map*. https://pubs.er.usgs.gov/publication/fs20193032
[43] Guang Wang, Xiaoyang Xie, Fan Zhang, Yunhuai Liu, and Desheng Zhang. 2018. bCharge: Data-Driven Real-Time Charging Scheduling for Large-Scale Electric Bus Fleets. In *2018 IEEE Real-Time Systems Symposium (RTSS)*. 45–55.
[44] Yuan Wang, Dongxiang Zhang, Lu Hu, Yang Yang, and Loo Hay Lee. 2017. A data-driven and optimal bus scheduling model with time-dependent traffic and demand. *IEEE Transactions on Intelligent Transportation Systems* 18, 9 (2017), 2443–2452.
[45] Sandareka Wickramanayake and HMN Dilum Bandara. 2016. Fuel consumption prediction of fleet vehicles using Machine Learning: A comparative study. In *2016 Moratuwa Engineering Research Conference (MERCon)*. IEEE, 90–95.
[46] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. 2016. Apache Spark: A unified engine for big data processing. *Commun. ACM* 59, 11 (2016), 56–65.
[47] Kai Zhang, Shaojun Liu, Yuhan Dong, Daoshun Wang, Yi Zhang, and Lixin Miao. 2017. Vehicle positioning system with multi-hypothesis map matching and robust feedback. *IET Intelligent Transport Systems* 11, 10 (2017), 649–658.

[48] Guang-Jing Zhou, Dong-Fan Xie, Xiao-Mei Zhao, and Chaoru Lu. 2020. Collaborative Optimization of Vehicle and Charging Scheduling for a Bus Fleet Mixed With Electric and Traditional Buses. *IEEE Access* 8 (2020), 8056–8072.