


Article

Bi-Level Fleet Dispatching Strategy for Battery-Electric Trucks: A Real-World Case Study

Dongbo Peng ^{1,*} , Zhouqiao Zhao ¹ , Guoyuan Wu ²  and Kanok Boriboonsomsin ²

¹ Department of Electric and Computer Engineering, University of California Riverside, Riverside, CA 92521, USA

² Center for Environmental Research & Technology, University of California Riverside, Riverside, CA 92507, USA

* Correspondence: dpeng017@ucr.edu

Abstract: Driven by new regulations concerning greenhouse gas (GHG) emissions in the transportation sector, battery-electric trucks (BETs) are considered one of the sustainable freight transportation solutions. In this paper, a dispatching problem of the BET fleet is formulated as a capacitated electric vehicle routing problem (VRP) with pick-up and delivery. As the BET dispatching problem is NP-hard, the performance of existing approaches deteriorates in large instance problems, especially when the customers have different preferences and constraints. This article proposes a bi-level strategy that incorporates routing zone partitioning and metaheuristic-based vehicle routing to solve the large-scale BET dispatching problem, considering the delivery types, limited travel distances, and cargo payloads. We apply this strategy to a real-world fleet dispatching scenario with around 300 customer positions for pickups and drop-offs. The experimental results demonstrate that the proposed bi-level strategy can reduce total travel distance and travel time by 24–31%, compared to the baseline strategy implemented in the real world.

Keywords: sustainable urban freight transportation; meta-heuristic algorithms; truck routing problems



Citation: Peng, D.; Zhao, Z.; Wu, G.; Boriboonsomsin, K. Bi-Level Fleet Dispatching Strategy for Battery-Electric Trucks: A Real-World Case Study. *Sustainability* **2023**, *15*, 925. <https://doi.org/10.3390/su15020925>

Received: 23 November 2022

Revised: 23 December 2022

Accepted: 27 December 2022

Published: 4 January 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Heavy-duty on-road trucks account for 70% of all freight movements and 20% of transportation-sector greenhouse gas (GHG) emissions in the United States [1]. To reduce GHG emissions, green logistics [2] aim to improve the sustainability of producing and distributing goods by considering environmental and social factors. Efforts have been made over the decades to reduce vehicle miles traveled (VMT) and employ innovative technologies (e.g., transportation electrification). In real-world applications, logistics companies operate a fleet of vehicles to serve customers while maximizing their net profits. These vehicles depart from the depot, visit multiple customers for pick-up and/or delivery, and return to the depot, which can be formulated as a typical vehicle routing problem (VRP). Over the years, many varieties and extensions of VRP have been proposed to adapt to real-world constraints and applications. As the emergence of electric commercial vehicles (ECVs), the electric vehicle routing problem (EVRP) (see, e.g., [3–5]) becomes a hot topic in recent years. Especially some logistics companies are trying to find a more sustainable way during day-to-day operations in the urban area to reduce GHG emissions, for example, by substituting heavy-duty diesel (HDD) trucks for battery electric trucks (BETs) in truck fleets (e.g., [6,7]).

However, solving the VRP and varieties of VRP is NP-hard. Based on a benchmark dataset, Solomon investigated vehicle routing and scheduling problems with time window constraints, by comparing the performance of a variety of heuristic algorithms via an extensive computational study [8]. Exact metaheuristic approaches (e.g., [9,10]) were designed to solve VRP. In particular, Rizzoli et al. [10] applied the ant colony optimization (ACO) algorithm for the basic VRP, and showed that ACO can be used for real-world VRP.

Polacek et al. [11] proposed an algorithm based on the variable neighborhood search (VNS) to solve a multi-depot VRP with time windows. This study demonstrated that the approach was competitive with the tabu search algorithm.

Unlike VRP, EVRP needs to consider limited driving range (i.e., all-electric range or AER) and the recharging opportunities, which render a more complicated solution space. Lin et al. [12] conducted a network topology analysis to solve EVRP, which was used in a problem with 13 customers. This approach cannot provide a solution to an EVRP of a larger network. In [5], the green vehicle routing problem (G-VRP) was introduced for alternative fuel vehicles (including electric vehicles), and solved by two heuristic algorithms, i.e., the Modified Clark and Wright Saving, and the Density-Based Clustering. In addition, Schneider et al. [4] extended [5] by conducting the Variable Neighborhood Search/Tabu Search (VNS/TS) hybrid method to solve the electric vehicle routing problem with time window (EVRP-TW), which considered the possibility of recharging and capacity constraints on EVs. Recently, Zhao et al. [7] provided an ACO-based dispatching and scheduling algorithm for a battery-electric truck (BET) fleet, considering AER, en-route charging opportunity (including different charging rates), pick-up/delivery time window, and capacity constraints.

However, it is challenging to apply these aforementioned heuristic methods directly to a large-size vehicle routing problem (e.g., with the size of more than 250 customers) and then obtain a solution within a reasonable time frame (e.g., less than 2 h). One important issue is that the performance of heuristic algorithms deteriorates in large-scale problems [13]. To deal with this problem, Wu et al. [14] combined the centroid-based clustering approach and the set partitioning method. They developed a framework to solve combinatorial problems efficiently (e.g., ridesharing). A few recent studies implemented learning-based strategies to solve the NP-hard combinatorial problems and partition large-scale VRPs into subproblems (for instance, [15,16]). Furthermore, the application of some algorithms needs additional consideration. For example, the ACO algorithm requires knowledge of the number of routes and vehicles before solving the problem. In addition, it is computationally heavy when tackling a large-size problem, because it needs to calculate a distance matrix and check the feasibility of each individual route, especially for a real-world case study.

In this paper, we investigate a BET dispatching strategy in which we consider a home-based charging station with a significant number of pickups and deliveries. To address limitations of the large-size dispatching problem, we further extend our previous work [7] and propose a bi-level strategy to handle this challenge, by leveraging the advantages of unsupervised learning methods and metaheuristic algorithms. At the upper level, a classic centroid-based unsupervised clustering method, the K-means clustering algorithm [17] is applied to decouple the entire service region into multiple appropriate dispatching subzones, which can dramatically speed up the computation times and influence the solution configuration. At the lower level, both ACO and VNS algorithms are adopted to solve the subzone-based EVRP-TW with capacity constraints, which achieves a good solution quality. In summary, the major contributions of this research are summarized as follows:

1. We propose a bi-level strategy which incorporates zone partition and zone-based BET fleet routing to guarantee scalability.
2. We leverage the advantages of both ACO and VNS to solve the BET fleet dispatching problem considering the home-based charging opportunities.
3. We use the real-world logistics data, including orders, itineraries, and routes, and the fleet dispatching plan as a benchmark for comparison, to evaluate the performance of the proposed strategy.

The remainder of this paper is organized as follows. Section 2 formulates the mathematical problem. Section 3 describes the decomposition and the principle of both ACO and VNS to address the large-scale BET fleet dispatching problem. Section 4 presents a case study based on real-world data, and Section 5 concludes the paper with a discussion on future work.

2. Problem Formulation

Figure 1 graphically describes our BET dispatching problem, where the dispatching center receives the order information, including the address, latitude, longitude, service type (pick-up and delivery), weight, and service time for each customer. The BET fleet should visit a set of customers on each operation day, considering pick-up and drop-off as variant delivery types. Each BET has a limited cargo payload, travel distance, and operation time during each routing plan. In addition, the recharging station may be deployed in the dispatching center. The dispatching problem involves decisions in two processes: (a) partitioning the dispatching zone according to the order information to improve the dispatching efficiency and avoid the problem size being too large; and (b) determining the best routes and itineraries of the BET fleet to satisfy the constraints.

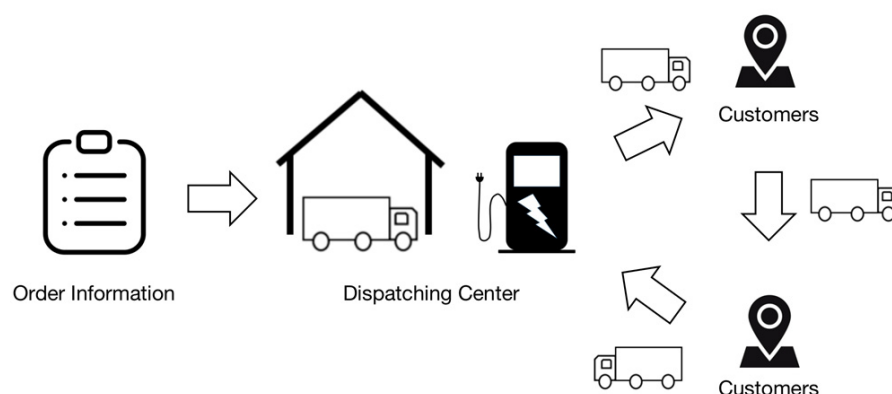


Figure 1. A simple dispatching system used for formulation.

The proposed dispatching problem can be modeled as follows: a complete directed graph can be defined as $\mathcal{G} = (\mathcal{N} \cup \mathcal{R}, \mathcal{A})$, where $\mathcal{N} = \{1, 2, \dots, N\}$ denotes the set of customers, and \mathcal{R} represents recharging station(s) that may include the depot. The set of arcs is given by $\mathcal{A} = \{(i, j) | i, j \in \mathcal{N} \cup \mathcal{R}, i \neq j\}$. Each customer $i \in \mathcal{N}$ is characterized by an assigned delivery type with demand q_i (positive if pick-up, negative if delivery), as well as a service time s_i . In addition, each arc is described by the following properties: non-negative travel distance d_{ij} and travel time t_{ij} .

A set of homogeneous BETs with a maximal capacity C is positioned at the depot. The BETs are assumed to be fully charged when departing from the depot O , while node D denotes the same depot when the BETs return, and every route starts at O and ends at D . The energy consumption is assumed to be a linear function with the travel distance.

Our objective is to minimize the total travel cost, including the travel distance and travel time to serve a set of customers. The variables and parameters used in this study are summarized in Table 1.

Table 1. Variable Definitions.

| Variable | Description |
|---------------|---|
| \mathcal{M} | Set of BETs |
| \mathcal{N} | Sets of customer vertices |
| \mathcal{R} | Recharging station(s) |
| a | Energy cost rate (\$/mile) |
| b | Labor cost rate (\$/hour) |
| g | Recharging rate |
| h | Energy consumption rate |
| d_{ij} | Distance between vertices i to j |
| t_{ij} | Travel time between vertices i to j |
| T_O | Earliest departure time from depot |
| T_D | Latest return time to depot |
| T_C | Maximum recharging time |
| C | BET capacity |
| Q | BET maximum battery capacity |

Table 1. Cont.

| Variable | Description |
|-----------|--|
| q_i | Demand at vertex (positive if pick-up and negative if drop-off) |
| s_i | Service time at vertex i |
| k_i | Node type. 0 if customer, 1 if charging station. |
| u_{im} | Decision variable specifying the remain cargo for BET m on arrival at vertex i |
| y_{im} | Current state of charge (SOC) for BET m at vertex i |
| Y_{im} | Finish charging SOC for BET m |
| x_{ijm} | Binary decision variable. 0 if the route from i to j is not visited by BET m , 1 otherwise |

Thus, the BET dispatching problem can be formulated as a mixed-integer program as follows:

$$\min_{i \in O \cup \mathcal{N} \cup \mathcal{R}, j \in D \cup \mathcal{N} \cup \mathcal{R}, i \neq j, m \in M} \sum (ad_{ij} + bt_{ij})x_{ijm} + \sum_{i \in \mathcal{R}, j \in \mathcal{N}, m \in M} (Y_{im} - y_{im})/g \cdot x_{jim}b, \quad (1)$$

Subject to:

$$\sum_{j \in D \cup \mathcal{N} \cup \mathcal{R}, i \neq j, m \in M} x_{ijm} = 1, \forall i \in (O \cup \mathcal{N}), \quad (2)$$

$$\sum_{i \in O \cup \mathcal{N} \cup \mathcal{R}, i \neq j, m \in M} x_{ijm} - x_{jim} = 0, \forall j \in (O \cup \mathcal{N} \cup \mathcal{R}), \quad (3)$$

$$\sum_{j \in (D \cup \mathcal{N} \cup \mathcal{R})} x_{ijm} \leq 1, \forall i \in \mathcal{R}, m \in M, \quad (4)$$

$$\sum_{i \in (O \cup \mathcal{N} \cup \mathcal{R})} x_{ijm} = \sum_{p \in (D \cup \mathcal{N} \cup \mathcal{R})} x_{jpm}, \forall j \in (\mathcal{N} \cup \mathcal{R}), m \in M, \quad (5)$$

$$T_O \leq \left(t_{ij} + (1 - k_i)s_i + k_i \cdot \frac{Y_{im} - y_{im}}{g} \right) x_{ijm} \leq T_D, \quad (6)$$

$$\forall i \in O \cup \mathcal{N} \cup \mathcal{R}, j \in (O \cup \mathcal{N} \cup \mathcal{R}), i \neq j, m \in M,$$

$$0 \leq u_{jm} \leq u_{im} - q_i x_{ijm} + C(1 - x_{ijm}), \quad (7)$$

$$\forall i \in O \cup \mathcal{N} \cup \mathcal{R}, j \in (D \cup \mathcal{N} \cup \mathcal{R}), i \neq j, m \in M,$$

$$0 \leq u_{im} \leq C, \forall i \in \mathcal{R}, m \in M \quad (8)$$

$$0 \leq ((1 - k_i) \cdot y_i + k_i \cdot Y_i - h \cdot d_{ij}) x_{ij} \leq Q, \quad (9)$$

$$\forall i \in O \cup \mathcal{N} \cup \mathcal{R}, j \in (O \cup \mathcal{N} \cup \mathcal{R}), i \neq j, m \in M,$$

$$Y_{im} = \text{Min}\{T_c \cdot g, (Q - y_{im})\}, m \in M, \forall i \in O \cup \mathcal{R} \quad (10)$$

$$x_{ijm} = 1 \text{ or } 0, \forall i \in O \cup \mathcal{N} \cup \mathcal{R}, j \in (O \cup \mathcal{N} \cup \mathcal{R}), i \neq j, m \in M, \quad (11)$$

The objective function of minimizing the total travel cost is defined in (1). Constraint, (2) enforces the connectivity of customer visits. Constraint (3) establishes flow conservation by ensuring that the number of incoming arcs to each vertex is equal to the number of outgoing arcs. Constraint (4) ensures the connectivity of the recharging stations. We assume that there is a recharging station at the depot, and it is capable of charging BETs at any time. Constraint (5) defines the conservation law that guarantees the number of incoming arcs to each node is equal to the number of outgoing arcs. Constraint (6) guarantees time feasibility that the sum of service time, travel time and recharging time cannot exceed the assigned total operation time. Constraints (7) and (8) ensure the demand fulfillment of all customers during BETs' visits by guaranteeing a non-negative cargo payload at any vertex, including the depot. Constraint (9) ensures that the BETs never exceed travel range limitations. Constraint (10) indicates the state of charge (SOC) when the BET m is leaving the charging station. Finally, x_{ijm} is a binary variable to indicate the selection of the routes.

3. Methodology

Figure 2 presents the overall architecture of the proposed system. We first created a node-to-node database including the travel distance and duration from one customer to home-base/charging station. Then, we define the dispatching problem as illustrated in the previous section. And we propose a bi-level hierarchical method for solving the dispatching problem. At the upper level, we decompose the large problem into multiple small dispatching zones to narrow the searching space. This may lead to a more efficient exploration of dispatching strategies. At the lower level, we apply metaheuristic algorithms to calculate near-optimal solutions.

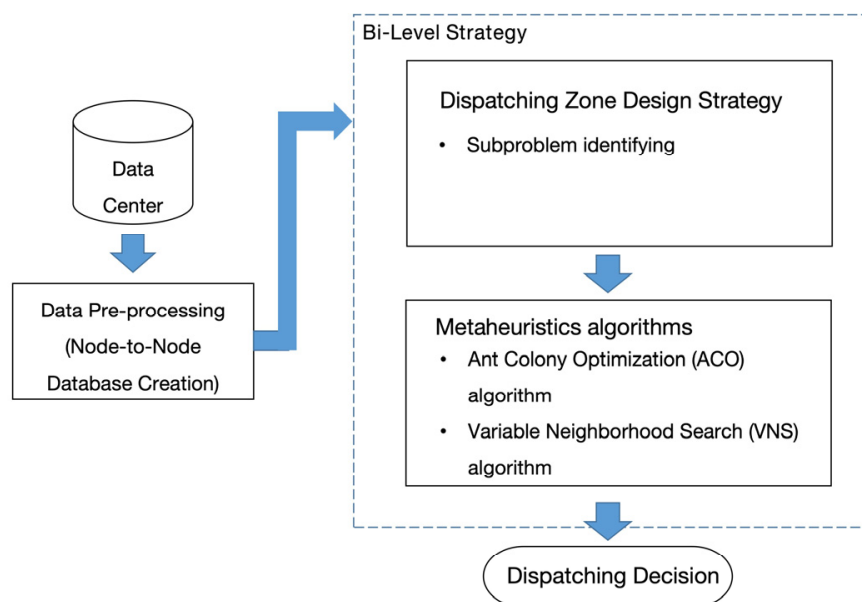


Figure 2. Block Diagram of Methodology.

As a promising approach to the BETs dispatching problem, metaheuristics algorithms can be divided into neighborhood-oriented metaheuristics (e.g., Variable Neighborhood Search (VNS)) and population metaheuristics (e.g., Ant Colony Optimization (ACO)) and a recent survey can be found in [18]. In this study, we want to evaluate the performance of two representative metaheuristic algorithms, i.e., ACO [19] and VNS [20]. The ACO algorithm implements a randomized constructive greedy heuristic that discovers good solutions based on the positive feedback of artificial pheromone trails, while the VNS algorithm solves combinatorial optimization problems based on a systematic change of neighborhood [21].

3.1. Data Pre-Processing

Data Pre-processing aims to filter key features (e.g., order information, latitude, longitude, service time, delivery type, and vehicle ID) from the dispatching center. Since the original dataset contains duplicated data, and collection errors, data preprocessing includes three steps: (1) data aggregation, (2) data cleaning, and (3) travel distance and time matrices creation. A cleaned dispatching dataset and corresponding travel distance and travel time matrices can be obtained. A more detailed description of data and pre-processing is presented in Section 4.

3.2. Dispatching Zones Partition for Large-Scale Problem

To deal with a large-scale BET fleet dispatching problem in a time-efficient and scalable manner, we first decompose the large problem into a few smaller sub-problems (sub-zones), where the goal is to find the best partition. More specifically, a clustering algorithm is applied to divide the entire service region into different dispatching zones. Based on

customers' latitude and longitude information, we implement an unsupervised learning technique, the K-means clustering, to determine the sub-zones to schedule BET dispatching before conducting the dispatching algorithms, VNS and ACO, respectively.

Existing studies show that most heuristic algorithms are capable of finding the near-optimal solutions in a reliable and efficient manner to VRPTW problems with the instance size in the range of 50–100 [22]. Therefore, in this paper, we partition the service region into sub-zones, each covering 20–70 pickup and delivery locations. This ensures the balance between the efficiency and solution quality of our dispatching algorithms. In the clustering process, a manual tuning process is needed to avoid too large clusters (e.g., more than 70 positions) and too small clusters (e.g., less than 5 positions). If a sub-problem cluster contains more than 70 pickup and delivery locations, the performance of metaheuristic algorithms is cumbersome. Then we continue to decompose this cluster. On the contrary, if a zone is too small (e.g., a small group of customers is far away from the other groups), which means the algorithms only search on a small subproblem, we need to tune the value of K manually to avoid this circumstance. In this case study, we choose K as 8 cluster centers.

3.3. Variable Neighborhood Search (VNS) Algorithm

There are two reasons for us to utilize the VNS algorithm in this study. First, the VNS algorithm can provide a control group for the ACO algorithm to ensure that the results are near-optimal. Second, unlike the ACO algorithm, the VNS algorithm does not require the number of vehicles as the user's input, and it can provide this value from its output, thus serving as the initialization parameters to the ACO algorithm for fair comparison.

Algorithm 1 shows a pseudocode for the VNS heuristic algorithm. The VNS works as follows: when initializing the VNS algorithm, a set of neighborhood structures is predefined, and all neighborhoods are around any point $x \in X$ of the solution space. An initial feasible solution x is generated by Solomon's I1 insertion heuristic [23] given as the current best solution. Then VNS randomly generates a solution x' in the shaking phase from the s th neighborhood. Next, the variable neighborhood descent approach in the local search process is applied to determine the local minimum x'' . At this point, if x'' improves the original result x , then x'' is accepted as the current best solution and restarts with neighborhood N_1 . Otherwise, if x'' fails to challenge the current best solution x , we refuse the new solution x'' and keep the current best solution x as a feasible input solution to the next neighborhood structure N_{k+1} .

Algorithm 1. Overview of VNS heuristic algorithm

Input: Dispatching zone $n = 1, 2, \dots, n_{max}$
Output: a set of solution x_{Vn}

```

1:  $N_k \leftarrow k = 1, 2, \dots, k_{max}$ 
2:  $S \leftarrow generate\_init\_solution()$ 
3:  $k \leftarrow 1$ 
4:  $x \leftarrow S$ 
5: while maximum neighborhood structure  $k_{max}$  not satisfied do
6:    $x' \leftarrow Shake(x, k)$ 
7:   Function VND( $x', k_{VNDmax}$ ):
8:      $k_{VND} \leftarrow 1$ 
9:     while maximum neighborhood structure  $k_{VND}$  not satisfied do
10:       $x \leftarrow argmin_{y \in N_k(x')} f(y)$ 
11:      //Find the best neighborhood in  $N_k(x')$ 
12:       $x'', k \leftarrow NeighborhoodChange(x, x', k_{VND})$ 
13:       $k_{VND} \leftarrow k_{VND} + 1$ 
14:    end while
15:     $x, k \leftarrow NeighborhoodChange(x, x'', k_{VND})$ 
16:     $k \leftarrow k + 1$ 
17: end while

```

The shaking phase and the local search intensification phase are equal to the standard VNS. The neighborhood structures are defined by means of a set of neighborhood operators, including 2-opt, or-opt [24], 2-opt * [25], relocate, exchange [26], cross-exchange [11], and a problem-specific operator called stationInRe [4]. The VNS is set to stop when the last neighborhood is reached.

3.4. Ant Colony Optimization (ACO) Algorithm

Inspired by the cooperative behavior of ant colonies, the ACO is used to find the optimal path (usually shortest) across a graph [19]. A group of ants read pheromones to construct solution and update the concentration of pheromone to mark their constructed routes. As the positive feedback loop contains the highest quantities of pheromone, the near-optimal solution can be built.

The ACO algorithm has three main steps: initialization, construction of ant solutions, and update of pheromones. The basic procedures of the ACO heuristic are shown in Algorithm 2. At the start of the algorithm, we need to predefine how many BETs are required in the searching step and the number of ants. Then, a set of m ants construct solutions to the problem being tackled. Each ant starts with an initially empty solution and uses a probabilistic rule to choose customer i to j as follows:

$$p(i, j) = \frac{\tau(i, j)^\alpha \eta(i, j)^\beta}{\sum \tau(i, j)^\alpha \eta(i, j)^\beta}, \quad (12)$$

where $\eta(i, j)$ is a function that assigns each feasible solution from node i to j , which is heuristic information; $\tau(i, j)$ is the current pheromone value; parameter α and β determine the relative influence between the pheromone and the heuristic information.

Algorithm 2. Overview of ACO heuristic algorithm

Input: Dispatching zone $n = 1, 2, \dots, n_{max}$

Output: a set of solution S_{An}

Initialization: Pheromone trails ($\Delta\tau$) and create a set of ants m

```

1:  $i \leftarrow 0$ 
2: while maximum iteration  $I_{max}$  not satisfied do
3:   Construct Ant Solution
    $Cost\ Q_{IterSol} \leftarrow$  Current Best Solution at iteration  $i$ 
4:   if  $Q_{IterSol} > Q_{BestSol}$  then
5:      $Q_{BestSol} \leftarrow Q_{IterSol}$ 
6:   end if
7:   Pheromone concentration ( $\Delta\tau$ ) update
8:    $i \leftarrow i + 1$ 
9: end while
```

When the solution for the current iteration is obtained, the pheromone update is intended to help the ants find a better solution for the next iteration. The pheromone trails are updated by the strategy as follows:

$$\Delta\tau = \sum_1^q Q / cost, \quad (13)$$

$$\tau(t+1) = (1 - \rho)\tau(t) + \Delta\tau, \quad (14)$$

where $\Delta\tau$ is the amount of pheromone that the best ant deposits; Q is the pheromone increasing rate; and the cost is the current best solution of the objective function. Pheromone evaporation $\rho \in (0, 1]$ is needed to avoid the searching process converging too fast and getting trapped into local optima. We keep one elite ant in the ACO during each iteration to guarantee the algorithm convergence speed.

4. Case Study: A Real-World BET Dispatching Optimization Problem in Southern California

This section presents our experiments and results using a real-world dataset to evaluate the performance of the bi-level strategy. We first introduce the details of this real-world dataset. Since the real-world data is noisy, Section 4.2 describes the pre-processing effort and presents how the travel distance and travel time matrices are obtained. Section 4.3 demonstrates computational experiments for the bi-level strategy.

4.1. Data Description

The real-world dataset contains historical movements of the fleet of conventional diesel trucks and the orders served on June 16th, 2020, covering the regions of Riverside and San Bernardino County, CA. From the vehicle perspective, each truck has a telemetry data file in a time-series format that includes timestamp, latitude, longitude, ignition status, speed, direction, and odometer reading. From the fleet management perspective, there is a summary data file containing trailer ID, the sum of total orders, the sum of total pieces, the sum of total pallets, and the sum of total weights. From the customer perspective, another summary data file is available, containing the details of pick-up and delivery information such as location address, arrival and departure times, total weight, and the total number of pieces.

4.2. Data Processing

We first aggregated the data from different sources. Although the trucks' telemetry data contain timestamps, the time interval between consecutive data points is not fixed, ranging from 1 min to 15 min. Also, the data points may not align with the arrival and departure times for delivery or pick-up. Therefore, we regenerated the itinerary of each truck with dispatching information. Specifically, the itinerary contains order ID, activity type (depart/return from/to home base, pick-up, or deliver), tractor ID (same as truck ID), arrival time, departure time, duration (at loading/unloading stops), weight, address, latitude, longitude, accumulated weight, travel distance (from the previous location to the current location), and accumulated travel distance. As the initial cargo weight on each truck is unknown, we assumed that the truck would deliver all the cargo loaded onto it at the end of the day. As a result, the accumulated weight can be calculated by backtracking through the orders. Based on the historical itineraries, a delivery order pool and a pick-up order pool were created.

In the dataset, customers are distributed in a number of cities. The total number of orders that the trucks collectively served on that day was 850. To ensure efficient execution of the proposed algorithms, we took steps to clean the data as follows. First, there were some cases where some trucks served multiple orders at the same location and at the same time. Hence, we combined these orders and summed up their weights to treat them as one bundled order. Second, we removed the orders without position information. After these two steps, we obtained a reduced dataset that includes 333 service locations served by 47 trucks.

The travel distance and travel time from one location to another in the pool are critical information for the optimization problem. However, since the telemetry data does not have enough resolution in both space and time, it was difficult to determine the actual routes taken by the trucks on that day. Therefore, we estimated the travel routes using the Direction Service Application Programming Interface (DSAPI) provided by OpenRouteService [27]. The inputs for this API include the locations of the start and end points and vehicle type, while the outputs contain the detailed route in a link-level resolution as well as route summary statistics (e.g., travel distance and travel time). We compared the travel distance of the routes estimated by the API with the travel distance of the actual routes taken inferred by the telemetry data and found that they match well with each other for most of the cases. It is noted that the travel time estimated from the API is not comparable to that from the telemetry data, as no traffic information but the speed limit of each link is used.

4.3. Performance on BETs Dispatching Instances

We first applied the K-means clustering algorithm to generate multiple smaller dispatching zones as a partition of 333 customers. According to the empirical experiments in our case study, for small-size subproblems, it is ideal to have less than 20 pick-up/drop-off locations in each dispatching zone; for large-size subproblems, the number of locations is more than 20, respectively. The parameter K represents the total number of dispatching zones. Figure 3 shows the results of the dispatching zone clustering process on 333 customer positions. Then, the ACO and VNS were applied to dispatch BETs to serve orders in each dispatching zone. Table 2 summarizes the input parameters in the case study.

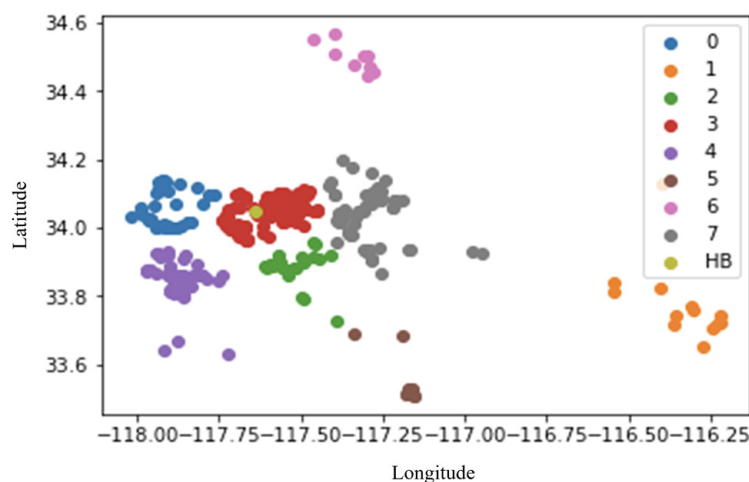


Figure 3. K-means clustering process for 333 customers positions decomposition. Numbers 0 to 7 represent the ID of each dispatching zone. All the BETs depart and return at the homebase (HB).

Table 2. Summary of Input Parameters.

| Type | Variable | Value |
|---------------------|--------------------------------|------------------|
| Scenario Assumption | Number of customers' positions | 333 |
| | Number of charging stations | 1 (at depot) |
| ACO Parameters | Loading/unloading time | [0, 2] h |
| | Cargo weight | (0, 29,600] lbs. |
| | Working hour | [8 am, 4 pm] |
| | Energy consumption rate | 2.14 kWh/mile |
| | BET battery capacity | 375 kWh |
| | BET payload capacity | 37,000 lbs. |
| | Labor cost rate | \$25/hour |
| | Energy cost rate | \$0.26/mile |
| | Recharging rate | 5 kWh/min |
| | Number of ants | 20 |
| | Number of iterations | 2000 |
| | Pheromone weight | 0.5 |
| | Heuristic information rate | 0.1 |
| | Evaporation rate | 0.05 |
| | Pheromone increasing rate | 20 |

To evaluate the feasibility of BET fleet to serve the same customer set, we restricted the all-electric range of each BET to be 175 miles (the nominal value from the manufacturer) and assumed a home-based charging at the depot (consistent with the real-world implementation). Because in the real-world dataset, some sub-zones are far away from the depot, where BETs cannot complete a round trip without en-route recharging. We performed further screening and left 278 positions to ensure the BET fleet can be recharged at the home-based depot without getting stranded on their way. The battery is assumed to be

fully charged when BET firstly departs from the depot. The maximum charging time is 60 min with 80% charging.

We used the generated BET dispatching instances to analyze the performance of our VNS and ACO heuristics. The results are shown in Table 3. For each approach, we calculated the total travel distance, total travel time, and total cost of the best solution for each of the ten dispatching zones. Furthermore, Table 3 also reports the number of BETs required to serve the orders in each zone and the computation times of both algorithms. Overall, the results show that the VNS outperforms the ACO algorithm regarding travel distance by 18.6%, travel time by 12.5%, and cost by 16.3%. However, the VNS takes a longer computation time.

Table 3. Comparison of BET Dispatching Results.

| Zone No. | No. of Orders | Total Travel Distance (miles) | | Total Travel Time (h) | | Total Cost (\$) | | Computation Time (s) | | Number of BET |
|----------|---------------|-------------------------------|------|-----------------------|------|-----------------|------|----------------------|------|---------------|
| | | ACO | VNS | ACO | VNS | ACO | VNS | ACO | VNS | |
| 1 | 42 | 234 | 165 | 7.9 | 5.9 | 329 | 238 | 106 | 1156 | 3 |
| 2 | 29 | 134 | 130 | 3.9 | 3.9 | 173 | 170 | 162 | 239 | 2 |
| 3 | 28 | 224 | 217 | 5.9 | 5.8 | 271 | 265 | 119 | 145 | 2 |
| 4 | 50 | 473 | 365 | 12.1 | 10.7 | 566 | 471 | 310 | 1009 | 5 |
| 5 | 53 | 417 | 296 | 10.5 | 8.9 | 523 | 387 | 180 | 1495 | 4 |
| 6 | 11 | 203 | 203 | 3.9 | 3.9 | 211 | 211 | 65 | 3 | 2 |
| 7 | 9 | 125 | 123 | 2.9 | 2.9 | 143 | 141 | 31 | 1 | 1 |
| 8 | 56 | 400 | 299 | 10.7 | 8.6 | 489 | 380 | 327 | 2176 | 4 |
| Sum | 279 | 2210 | 1798 | 57.8 | 50.6 | 2705 | 2264 | 1298 | 6223 | 23 |

Based on the experiment result in Table 3, compared to the VNS, when the number of orders is more than 20 (i.e., larger size problems), the ACO algorithm finds solutions more efficiently in terms of computation time, but the total costs are higher. The VNS finds solutions with much lower costs for those zones, albeit taking much more computation time. A hypothesis is that if the problem is relatively large and complicated, the ACO algorithm needs a fine-tuned parameter setting and an appropriate number of iterations and is prone to get trapped into local optima due to the randomness of its searching process. Therefore, a sensitivity analysis is conducted in Section 4.4.

4.4. Analyzing the Effect of the ACO Components

Compared with the VNS heuristic algorithm, the ACO needs a fine-tuning process to guarantee the solution quality since it has more parameters during the search [7]. This section aims to conduct a sensitivity analysis for the evaporation rate ρ and the pheromone increasing rate Q . An appropriate parameter setting significantly affects the performance or the searching process for the ACO algorithm [28]. In this way, we systematically test with different parameters as follows: $\rho = [0.01, 0.05, 0.1, 0.2, 0.4]$ and $Q = [20, 25, 30, 40, 50]$ on two instances, Zone_2 with 29 customers and Zone_4 with 50 customers. Experimentally, it should be noted that a higher evaporation rate and a lower increasing rate for the pheromone can cause a searching problem, in which the pheromone concentration is too low on each trail, so the artificial ants cannot continue to search the path. Therefore, in our experiments, we set the upper bound as 0.4 and 50 for the evaporation rate ρ and the pheromone increasing rate Q , respectively. For each parameter value, we conducted 3 runs and recorded the best test value to show the performance. Figures 4 and 5 show the experiment results for two instances, Zone_2 (with 29 customers) and Zone_4 (with 50 customers), respectively, with all the combinations of the parameters given before.

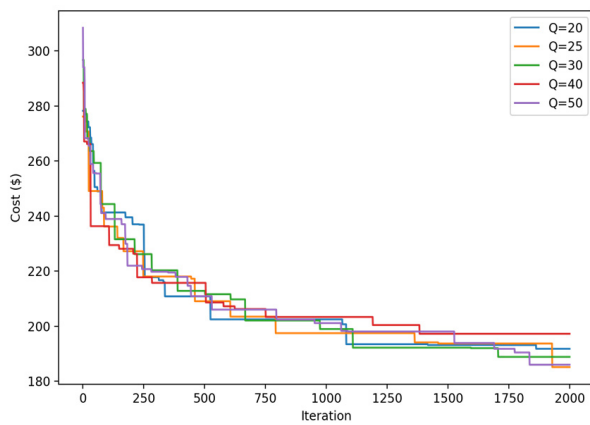
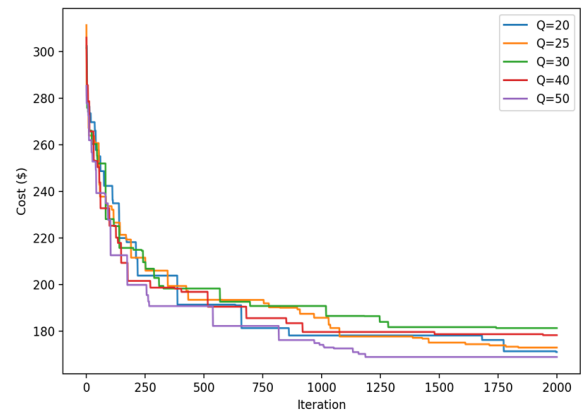
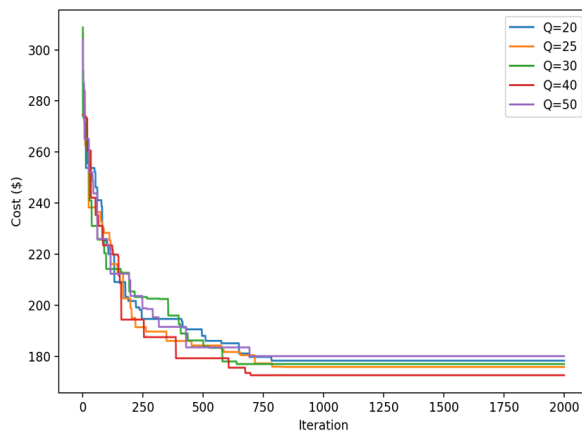
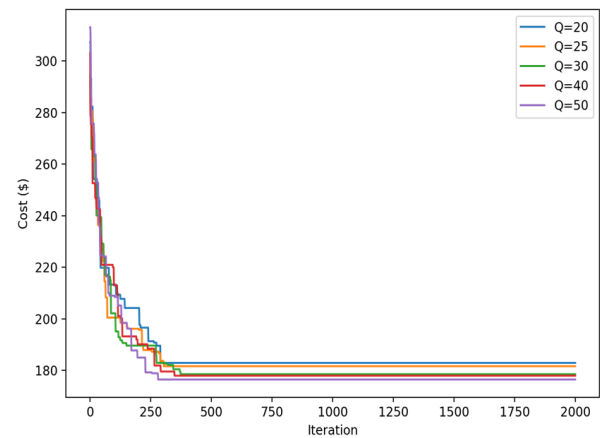
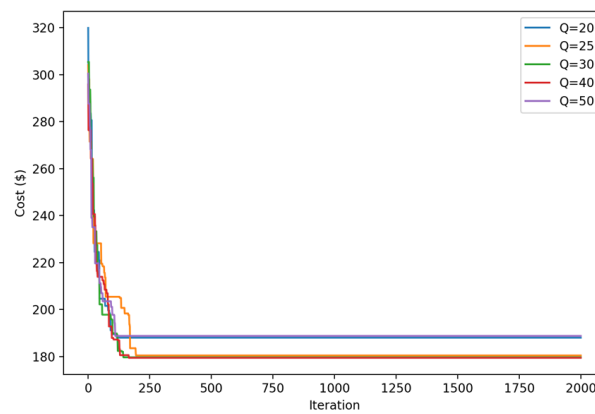
(a) Instance Zone_2, $\rho = 0.01$ (b) Instance Zone_2, $\rho = 0.05$ (c) Instance Zone_2, $\rho = 0.1$ (d) Instance Zone_2, $\rho = 0.2$ (e) Instance Zone_2, $\rho = 0.4$

Figure 4. Effect of the parameter $\rho = [0.01, 0.05, 0.1, 0.2, 0.4]$ on the solution quality of instance Zone_2 in total cost (y-axis). The x-axis shows the iterations of the ACO heuristic algorithm.

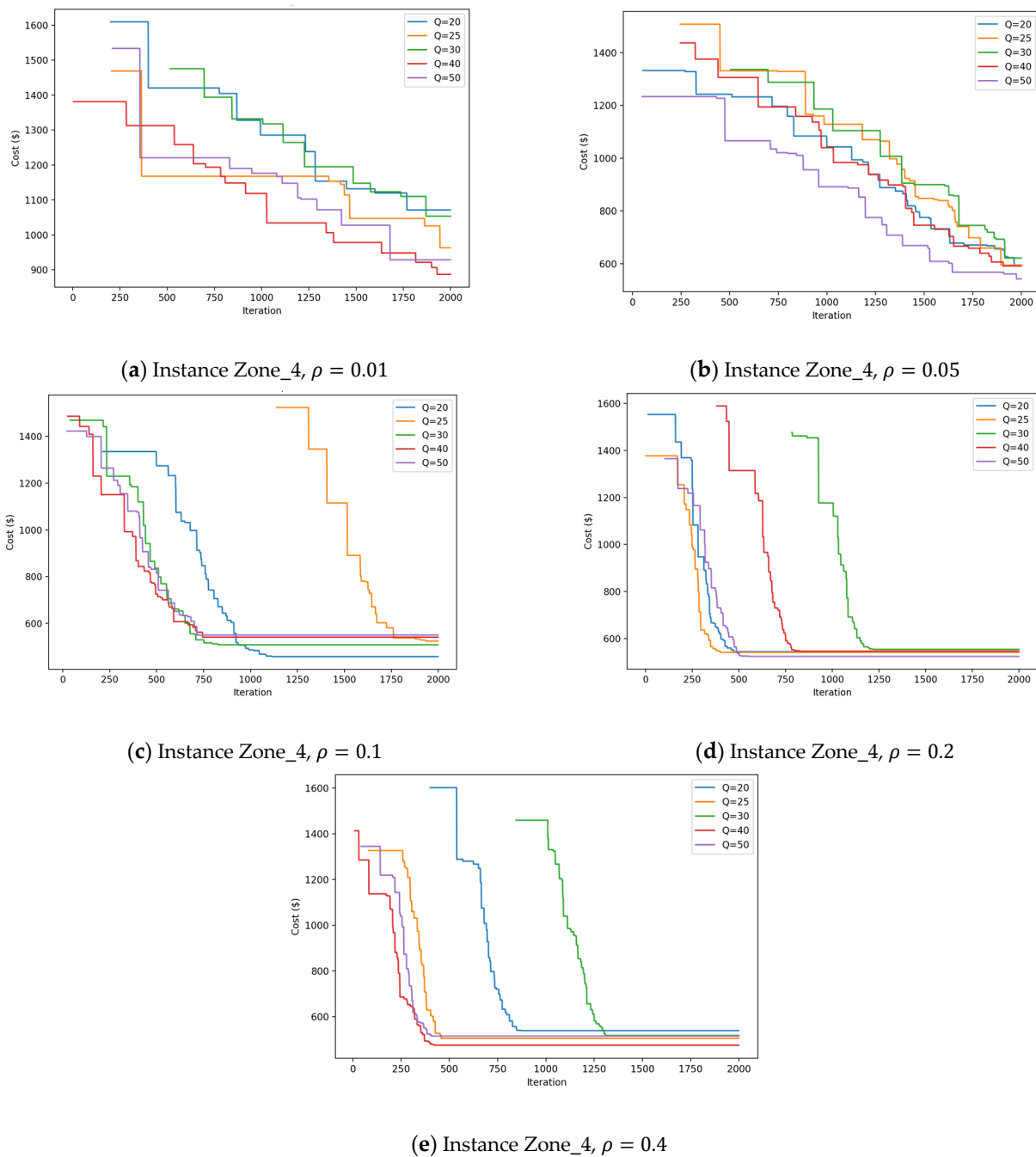


Figure 5. Effect of the parameter $\rho = [0.01, 0.05, 0.1, 0.2, 0.4]$ on the solution quality of instance Zone_4 in total cost (y -axis). The x -axis shows the iterations of the ACO heuristic algorithm.

Figure 4 shows the test results for Zone_2. As the evaporation rate ρ increases accordingly, the ACO can converge faster, but the total travel cost increases in this experiment. The reason is that a higher pheromone evaporation rate leads to the pheromone concentration decreasing quickly on each trail. As the iteration increases, only a few routes contain higher pheromone concentration, and others are much lower, so the artificial ants have a lower probability of exploring other trails. In this instance, the current best solution is generated by $\rho, Q = [0.05, 20]$.

4.5. Performance on Zone Based VRP

We further examine the performance of our bi-level framework on zone based VRP. In Table 4, we relaxed the driving range constraint due to the factors affecting the already

limited driving range of batteries and formulated a zone based VRP to test the performance of both VNS and ACO. It compared the results of our proposed strategy on the 333 positions with the historical data. Note that the travel time of the historical dispatching method in Table 4 was estimated by the same travel time matrix mentioned in Section 4.2 to be consistent with how the travel time of the other dispatching methods was estimated. By comparing our experimental results and the historical data in Table 4, the dispatching zone partition process can improve the efficiency of the heuristic strategy. The ACO algorithm saved 24.35% on the travel distance compared with the historical travel data. In addition, our bi-level approach can save approximately 30% on travel distance and time. The results show that our bi-level strategy would require 13 fewer BETs than the historical record, reducing the operating cost by 30.9%.

Table 4. Difference Between Historical Data and Bi-level Strategy.

| | Travel Distance | | Travel Time | | Cost (\$) |
|------------------------|-----------------|------------------------------|---------------|------------------------------|-----------|
| | f_d (Miles) | $\Delta_{\text{Real-World}}$ | f_t (Hours) | $\Delta_{\text{Real-World}}$ | |
| Historical data | 3967 | 0.00 | 103.2 | 0.00 | 4642.84 |
| Dispatching algorithms | | | | | |
| ACO | 3001 | 24.35% | 78.2 | 24.22% | 3510 |
| VNS | 2740 | 30.93% | 72.7 | 29.55% | 3241 |

5. Conclusions and Discussion

In this study, we have formulated an optimization problem called capacitated electric vehicle routing problem with pick-up and delivery to represent the real-world operation of a regional freight distribution with a fleet of battery electric trucks (BETs), which aims to minimize the operational cost for the BET fleet. Since the metaheuristic algorithm may fail to solve large-size instances [7,13], to address this problem, We have developed a bi-level strategy, including (a) the dispatching zone partition module and (b) metaheuristic-based vehicle routing module where two representative algorithms—*Ant Colony Optimization* (ACO) and *Variable Neighborhood Search* (VNS) are investigated. At the upper level, the zone partition strategy can efficiently decouple the dispatching problem into smaller size of instances, which improve the solution configuration for the metaheuristic algorithms. At the lower level, the metaheuristic algorithms can solve the fleet dispatching problem.

In addition, we have applied the bi-level strategy to a real-world case to validate the proposed strategy. The results demonstrate that our bi-level strategy can save approximately 30% on travel distance and time, compared to the baseline strategy. It should be noted that for small-size of instances, the ACO and VNS are able to find all optimal dispatching strategies, but for large-size of instances, our computational study shows that the solution quality of the VNS is better than the ACO. Especially, due to the neighborhood structures, the VNS may not easily fall into local optima, but it costs more searching time. Our real-world case study shows that the VNS outperforms the ACO by 16.3% in terms of total costs (both travel distance and time), with compromising in computational time. Contrary to the VNS, the ACO has more parameters requiring sensitivity analyses that fine-tune the parameters to improve the quality of solution. Furthermore, sensitivity analyses indicate the performance of the ACO heuristic algorithm varies under different parameter settings. We demonstrated how the evaporation rate and the pheromone updated rate influence the solution quality of the ACO. More specifically, we found that a higher evaporation rate will lead the ACO heuristic to converge faster, but the solution quality worsens.

There are several directions for enhancing and expanding this work in the future. A more comprehensive dispatching zone partition strategy considering the proximity in route distance rather than geographical distance should be considered for practical

implementation. Also, an ensemble strategy will be considered to enhance the system performance, which can leverage the advantages of different metaheuristic algorithms. In practice, we can consider incorporating the pick-up and delivery sequence constraints, e.g., prioritizing delivery orders to empty the trailer before pickup orders. It is also noted that the cargo payload may influence the energy consumption of BETs during operation. This dynamic feature can be incorporated into the problem formulation (e.g., [6,29]). Moreover, a safety impact on truck routing design will be considered in our future work, that may mitigate the potential risks between trucks and other vulnerable road users (e.g., cyclists [30] and pedestrians [31]). The route or road segment restriction for truck routing (e.g., truck restricted zone) will be incorporated into the optimization model.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: D.P., Z.Z., G.W., K.B.; data collection: Z.Z., D.P.; analysis and interpretation of results: D.P., Z.Z., G.W., K.B.; draft manuscript preparation: D.P., Z.Z., G.W., K.B. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was prepared as a result of work sponsored by the Volvo LIGHTS (Low Impact Green Heavy Transport Solutions) project (www.lightsproject.com). Volvo LIGHTS is part of California Climate Investments, a statewide program that puts billions of Cap-and-Trade dollars to work reducing greenhouse gas emissions, strengthening the economy, and improving public health and the environment—particularly in disadvantaged communities. Funding is also provided by South Coast Air Quality Management District’s Clean Fuels Program, which since 1988 has provided over \$320 million, leveraging \$1.2 billion, to fund projects to accelerate the demonstration and deployment of clean fuels and transportation technologies through public-private partnerships.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This paper was prepared as a result of work sponsored by the Volvo LIGHTS (Low Impact Green Heavy Transport Solutions) project (www.lightsproject.com). Volvo LIGHTS is part of California Climate Investments, a statewide program that puts billions of Cap-and-Trade dollars to work reducing greenhouse gas emissions, strengthening the economy, and improving public health and the environment—particularly in disadvantaged communities. Funding is also provided by South Coast Air Quality Management District’s Clean Fuels Program, which since 1988 has provided over \$320 million, leveraging \$1.2 billion, to fund projects to accelerate the demonstration and deployment of clean fuels and transportation technologies through public-private partnerships. The authors are thankful to Troy Musgrave of Dependable Highway Express for sharing sample fleet operations data. The opinions, findings, conclusions, and recommendations are those of the authors and do not necessarily represent the views of project sponsors or other team members.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Quiros, D.C.; Smith, J.; Thiruvengadam, A.; Huai, T.; Hu, S. Greenhouse gas emissions from heavy-duty natural gas, hybrid, and conventional diesel on-road trucks during freight transport. *Atmos. Environ.* **2017**, *168*, 36–45. [\[CrossRef\]](#)
- Sbihi, A.; Eglese, R.W. Combinatorial optimization and Green Logistics. *4OR* **2007**, *5*, 99–116. [\[CrossRef\]](#)
- Montoya, J.-A. Electric Vehicle Routing Problems: Models and Solution Approaches. Ph.D. Thesis, Université d’Angers, Angers, France, 2016.
- Schneider, M.; Stenger, A.; Goeke, D. The electric vehicle-routing problem with time windows and recharging stations. *Transp. Sci.* **2014**, *48*, 500–520. [\[CrossRef\]](#)
- Erdoğan, S.; Miller-Hooks, E. A green vehicle routing problem. *Transp. Res. Part E Logist. Transp. Rev.* **2012**, *48*, 100–114. [\[CrossRef\]](#)
- Goeke, D.; Schneider, M. Routing a mixed fleet of electric and conventional vehicles. *Eur. J. Oper. Res.* **2015**, *245*, 81–99. [\[CrossRef\]](#)
- Zhao, Z.; Wu, G.; Boriboonsomsin, K.; Kailas, A. Vehicle Dispatching and Scheduling Algorithms for Battery Electric Heavy-Duty Truck Fleets Considering En-route Opportunity Charging. In Proceedings of the 2021 IEEE Conference on Technologies for Sustainability (SusTech), Irvine, CA, USA, 22–24 April 2021; pp. 1–8. [\[CrossRef\]](#)
- Solomon, M.M. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Oper. Res.* **1987**, *35*, 254–265. [\[CrossRef\]](#)
- Baker, B.M.; Ayechew, M.A. A genetic algorithm for the vehicle routing problem. *Comput. Oper. Res.* **2003**, *30*, 787–800. [\[CrossRef\]](#)

10. Rizzoli, A.E.; Montemanni, R.; Lucibello, E.; Gambardella, L.M. Ant colony optimization for real-world vehicle routing problems. *Swarm Intell.* **2007**, *1*, 135–151. [[CrossRef](#)]
11. Polacek, M.; Hartl, R.F.; Doerner, K.; Reimann, M. A Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows. *J. Heuristics.* **2004**, *10*, 613–627. [[CrossRef](#)]
12. Lin, J.; Zhou, W.; Wolfson, O. Electric vehicle routing problem. *Transp. Res. Procedia* **2016**, *12*, 508–521. [[CrossRef](#)]
13. Li, S.; Yan, Z.; Wu, C. Learning to delegate for large-scale vehicle routing. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 26198–26211.
14. Wu, C.; Kamar, E.; Horvitz, E. Clustering for set partitioning with a case study in ridesharing. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016; pp. 1384–1388.
15. Pisinger, D.; Ropke, S. A general heuristic for vehicle routing problems. *Comput. Oper. Res.* **2007**, *34*, 2403–2435. [[CrossRef](#)]
16. Poullet, J. Leveraging Machine Learning to Solve The Vehicle Routing Problem with Time Windows. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2020.
17. Hartigan, J.A.; Wong, M.A. Algorithm AS 136: A K-Means Clustering Algorithm. *J. R. Stat. Soc. Ser. C (Appl. Stat.)* **1979**, *28*, 100–108. [[CrossRef](#)]
18. Erdelić, T.; Carić, T. A survey on the electric vehicle routing problem: Variants and solution approaches. *J. Adv. Transp.* **2019**, *2019*, 5075671. [[CrossRef](#)]
19. Dorigo, M.; di Caro, G. Ant colony optimization: A new meta-heuristic. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; Volume 2, pp. 1470–1477. [[CrossRef](#)]
20. Mladenović, N.; Hansen, P. Variable neighborhood search. *Comput. Oper. Res.* **1997**, *24*, 1097–1100. [[CrossRef](#)]
21. Gendreau, M.; Potvin, J.-Y. *Handbook of Metaheuristics*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 2.
22. Baldacci, R.; Mingozzi, A.; Roberti, R. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *Eur. J. Oper. Res.* **2012**, *218*, 1–6. [[CrossRef](#)]
23. Garcia, B.-L.; Potvin, J.-Y.; Rousseau, J.-M. A parallel implementation of the tabu search heuristic for vehicle routing problems with time window constraints. *Comput. Oper. Res.* **1994**, *21*, 1025–1033. [[CrossRef](#)]
24. Potvin, J.-Y.; Rousseau, J.-M. An exchange heuristic for routeing problems with time windows. *J. Oper. Res. Soc.* **1995**, *46*, 1433–1446. [[CrossRef](#)]
25. Lin, S. Computer solutions of the traveling salesman problem. *Bell Syst. Tech. J.* **1965**, *44*, 2245–2269. [[CrossRef](#)]
26. Savelsbergh, M.W. The vehicle routing problem with time windows: Minimizing route duration. *ORSA J. Comput.* **1992**, *4*, 146–154. [[CrossRef](#)]
27. Openrouteservice. Available online: <https://openrouteservice.org/> (accessed on 15 March 2022).
28. Ant Colony optimization for the Electric Vehicle Routing Problem | IEEE Conference Publication | IEEE Xplore. Available online: <https://ieeexplore.ieee.org/abstract/document/8628831> (accessed on 8 November 2022).
29. Macrina, G.; Pugliese, L.d.; Guerriero, F.; Laporte, G. The green mixed fleet vehicle routing problem with partial battery recharging and time windows. *Comput. Oper. Res.* **2019**, *101*, 183–199. [[CrossRef](#)]
30. Pokorný, P.; Pitera, K. Truck-bicycle safety: An overview of methods of study, risk factors and research needs. *Eur. Transp. Res. Rev.* **2019**, *11*, 1–14. [[CrossRef](#)]
31. Riccardi, M.R.; Mauriello, F.; Scarano, A.; Montella, A. Analysis of contributory factors of fatal pedestrian crashes by mixed logit model and association rules. *Int. J. Inj. Control. Saf. Promot.* **2022**, 1–15. [[CrossRef](#)] [[PubMed](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.