frvcpy: An Open-Source Solver for the Fixed Route Vehicle Charging Problem

Nicholas D. Kullman, Aurelien Froger, Jorge E. Mendoza, and Justin C. Goodson

Abstract

Electric vehicles offer a pathway to more sustainable transportation, but their adoption entails new challenges not faced by their petroleum-based counterparts. One of the most challenging tasks in vehicle routing problems addressing these challenges is determining how to make good charging decisions for an electric vehicle traveling a given route. This is known as the fixed route vehicle charging problem. An exact and efficient algorithm for this task exists, but its implementation is sufficiently complex to deter researchers from adopting it. In this work we introduce `frvcpy`, an open-source Python package implementing this algorithm. Our aim with the package is to make it easier for researchers to solve electric vehicle routing problems, facilitating the development of optimization tools that may ultimately enable the mass adoption of electric vehicles.

1 Introduction

Governmental regulations as well as a growing population of environmentally conscious consumers have led to increased pressure for firms to act sustainably. This pressure is particularly high in the logistics domain, which accounts for about one third of emissions in the United States (Office of Transportation and Air Quality 2019). Electric vehicles (EVs) offer a means to more sustainable transportation; however, they present technical challenges to which their conventional (i.e., internal-combustion engine) vehicle (CV) counterparts are immune. For example, because the distance EVs can travel on a single charge is often less than the distance an equivalent CV can travel on a tank of gas, EVs may demand more frequent recharging operations. This difficulty is compounded by the sparseness of EV recharging infrastructure relative to the network of refueling stations available to CVs, potentially forcing EVs to perform longer detours to recharge their batteries. Further, despite recent improvements to battery and charging station (CS) technology, recharging an EV still requires orders of magnitude more time than refueling a CV. The time required to recharge is also nonlinear with respect to the EV's *state of charge* (SoC), the relative amount of energy left in its battery, posing yet another challenge not applicable to CVs (Uhrig et al. 2015). Companies choosing to adopt EVs require optimization tools capable of handling these additional challenges.

The development of such optimization tools for conventional vehicles has commanded significant attention from the operations research (OR) community in the study of vehicle routing problems (VRPs). The incorporation of additional constraints that address the challenges posed by EVs has marked a new family of problems within VRPs known as electric vehicle routing problems (E-VRPs). One of the primary tasks in solving E-VRPs is making good charging decisions – namely, where to recharge and how long to do so. This is the crux of the fixed route vehicle charging problem (FRVCP), in which charging operations must be inserted into a fixed sequence of customers being visited by an EV so as to

minimize the time for the EV to reach the end of the sequence in an energy-feasible manner. The FRVCP naturally arises as a subproblem in many E-VRPs, since its solution is required in order to determine the true duration or cost of a given route. Having a capable solution method for the FRVCP is thus crucial to the advancement of E-VRP research.

While FRVCP solution methods exist, they tend to suffer from one or more of the following issues: inexactness (e.g., heuristic methods that provide suboptimal solutions), inefficiency (e.g., mixed-integer programs (MIPs) whose solvers require significant run time), or a lack of robustness (e.g., exact algorithms that are limited to simplified versions of the FRVCP). With this work, we offer an implementation of a solution method that suffers from none of these issues. Our implementation provides optimal solutions in low runtime for FRVCPs with rich, realistic problem features. It is based on the labeling algorithm proposed in Froger et al. (2019), which, though capable, is notoriously difficult to implement.

In an attempt to remove the burden of implementation for future E-VRP researchers, we offer our implementation in an open-source Python package, `frvcpy`. `frvcpy` is designed to be easily embedded in more complex solution schemes for E-VRPs (such as in a (meta)heuristic or Benders decomposition): it requires minimal dependencies and inputs, can be accessed either via the command line or a Python API, and includes a translator to generate the required inputs from a common instance format in the VRP community (VRP-REP (Mendoza et al. 2014)). Our aim with `frvcpy` is to make it easier to solve E-VRPs, thereby stimulating additional research in this field which promises to bring about more sustainable practices in logistics.

The remainder of the paper is organized as follows. We first define the FRVCP in §2, then discuss some of the previous work on FRVCPs in §3. In §4 we give an overview of the algorithm implemented in `frvcpy`, then describe the package itself in §5. We conclude with brief comments in §6.

2 Defining the FRVCP

We consider an EV with a fixed route $\Pi = (\pi_1, \ldots, \pi_R)$ that begins at some node π_1 (usually the depot), has a sequence of stops at other nodes $(\pi_i)_{i=2}^{R-1}$ (customers to visit), and terminates at some node π_R (also usually the depot). The vehicle begins at π_1 with its battery at some initial energy level q_0, often taken to be equal to its maximum battery capacity Q. Let the set of nodes in the route be $I = \{\pi_i |\, i \in 1..R\}$. We also consider a set of charging stations C at which the EV may recharge between stops in Π. Each CS $c \in C$ has some charging type (e.g., fast, slow) associated with a piecewise linear concave *charging function* $\Phi_c(t)$ specifying, for an empty battery, the resulting energy after charging for time t at CS c (see Figure 3). Additionally, define the related function $\bar{u}_c(q_1, q_2) = \Phi^{-1}(q_2) - \Phi^{-1}(q_1)$ to be the time required to charge from q_1 to q_2 at CS c. Let the set of breakpoints defining the charging function of CS c be B_c, where a breakpoint $b_i \in B_c$ is a (time, charge) pair: (b_i^t, b_i^q). When the EV travels between nodes $i, j \in I \cup C$, it incurs some known travel time t_{ij} and energy consumption e_{ij} (we assume the triangle inequality holds for both). At stops in Π, the EV may also incur some processing time (e.g., waiting). The objective of the FRVCP is to determine charging decisions – how much to recharge, at which CSs, between which stops in Π – that minimize the total time for the EV to traverse the route in an energy-feasible manner.

3 Related Literature

FRVCPs fall under the category of EV routing problems, which are themselves part of a larger body of research on VRPs. We focus our review here solely on FRVCPs. For an overview of E-VRPs we refer the reader to Pelletier et al. (2016), and similarly to Braekers et al. (2016) for an overview of VRPs.

Montoya et al. (2016) encounter an FRVCP in their work on a green vehicle routing problem. The FRVCP they consider assumes that vehicles may only visit one CS between stops, that they always fully restore their energy when recharging (that is, they follow a "full recharging strategy"), and that doing so requires constant time. To solve this FRVCP, they offer an exact algorithm. Roberti and Wen (2016) address an FRVCP in their work on the E-VRP with time windows (E-VRP-TW) and also offer an algorithm that solves this FRVCP exactly. Their solution accommodates a partial recharging policy, assuming that the time required to recharge is linear with the amount of energy. However, unlike in Montoya et al. (2016), they assume that the network of CSs is homogeneous; that is, that all CSs have the same charging technology. The FRVCP again arises in related works by Hiermann et al. (2016), Schiffer and Walther (2017), and Hiermann et al. (2019). These studies offer exact algorithms for the FRVCP under the assumption that at most one CS may be visited between stops, that the CSs are homogeneous, and that recharging requires linear time. Hiermann et al. (2016) additionally assume a full recharging strategy while Schiffer and Walther (2017) and Hiermann et al. (2019) allow partial recharging.

Montoya et al. (2017) then consider the first FRVCP that accommodates realistic (nonlinear) recharging times. In the study, they also demonstrate that the assumption of linear recharging times can lead to infeasible or suboptimal solutions. Their FRVCP allows for partial recharging and heterogeneous CSs but assumes that at most one CS may be inserted between stops. To solve their FRVCP, Montoya et al. offer both a heuristic and a MIP formulation. Koç et al. (2019) adopt the heuristic and MIP formulations from Montoya et al. (2017) to solve a similar FRVCP that arises in their work on the E-VRP with shared CSs and nonlinear charging. Baum et al. (2019) then offer a labeling algorithm to solve an FRVCP on real road networks that also accommodates realistic recharging times and allows for multiple CS insertions, although it is restricted to the special case where the route length is two (an origin-destination (OD) pair).

Finally, Froger et al. (2019) propose an exact labeling algorithm to solve the FRVCP from Montoya et al. (2017). Their algorithm is not restricted to OD pairs, and it additionally allows the EV to visit multiple CSs between stops in the route, making theirs the richest of the aforementioned FRVCP variants. Over a testbed of nearly 30,000 instances, they compare their labeling algorithm against a heuristic and a commercial solver for a MIP formulation. They find that the labeling algorithm matches the optimality of the MIP with a runtime comparable to the heuristic. The algorithm is thus state of the art for solving FRVCPs. However, the authors note that its performance is not without cost. They state that E-VRP researchers may ultimately prefer to adopt the heuristic solution, despite its inferior performance, given the complexity of implementing the labeling algorithm. Here, we offer an implementation of the algorithm in `frvcpy` in an attempt to ensure that its complexity does not prevent its adoption.

The algorithm from Froger et al. (2019) has also been adapted to accommodate FRVCPs with additional constraints. For example, in work on a stochastic E-VRP with public CSs, Kullman et al. (2019) adapt the algorithm to accommodate an FRVCP with discrete charging decisions and time-dependent waiting times at CSs. Similarly, Kullman et al. (2020a) adapt an early version of `frvcpy` to accommodate an FRVCP with customer time windows. In both cases, the algorithm's speed and exactness were required as it was called repeatedly to solve a subproblem in a Benders-based branch-and-cut procedure.

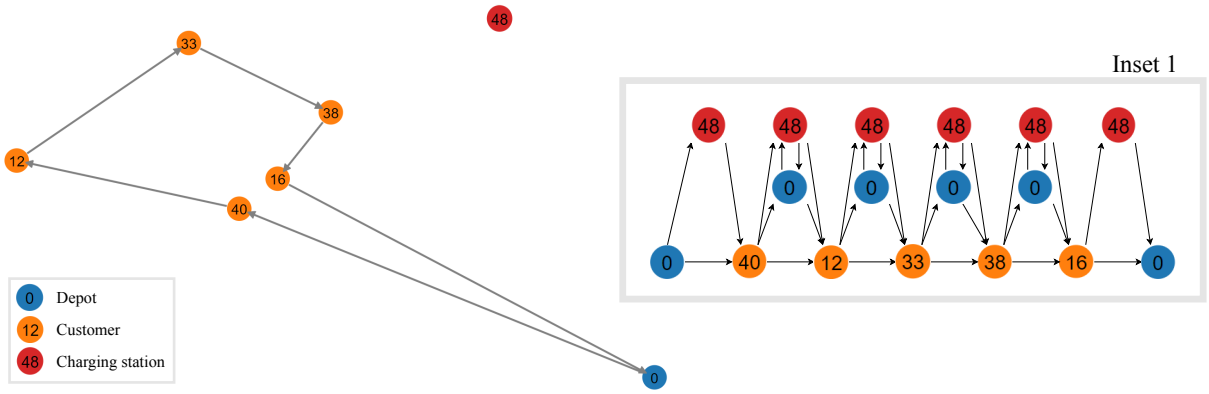

Figure 1: Instance excerpt depicting a fixed route in the original problem graph (left) and the modified problem graph for the FRVCP (Inset 1), which includes dummy nodes for CS insertions.

4 Overview of Labeling Algorithm from Froger et al. (2019)

Given the algorithm's complexity, we provide here a cursory overview and refer the reader to Froger et al. (2019) for additional details (see, in particular, their discussion of Algorithm 3 in §5.3 and Appendix E).

To find the optimal charging decisions for a given route Π, the FRVCP is reformulated as a resource-constrained shortest path problem. The algorithm then works by setting labels at nodes on a modified graph reflecting the vehicle's possible movements along Π (Figure 1, Inset 1). Labels are defined by *SoC functions* — piecewise-linear functions comprised of *supporting points* $z = (z^t, z^q)$ that describe a state of departure from a node in terms of time z^t and charge (SoC) z^q.

During the algorithm's execution, labels are extended along nodes in the graph. When a label is extended to a CS node c, we create new supporting points for each breakpoint in B_c to which we could charge (that is, breakpoints with a higher energy than that with which we arrived). Consider Figure 2, which depicts this process when extending a label along the edge from customer 33 to CS 48. When it arrives to CS 48, its SoC function has only one supporting point z_1 (assuming the EV has not yet stopped to recharge) depicted by the black square in the right graph of Figure 2. Then for each breakpoint in the CS's charging function to which the EV could recharge (b_2, b_3, b_4), we add a supporting point to the label's SoC function (z_2, z_3, z_4) whose time and charge reflect the decision to charge to that breakpoint. Figure 2 shows this explicitly for the new supporting point z_4, corresponding to the decision to recharge to the breakpoint b_4 (more specifically, to b_4^q).

We continue to extend labels along nodes in the graph until the destination node π^R is reached, whereat the algorithm returns the time of the first supporting point in the label's SoC function. Bounds on energy and time are established in pre-processing and are used alongside dominance rules during the algorithm's execution to improve its efficiency.

5 The `frvcpy` Package

`frvcpy` is an open-source Python-based implementation of the labeling algorithm from Froger et al. (2019) for solving the FRVCP. In this section we give an overview of its structure (§5.1), demonstrate its usage (§5.2), and briefly comment on its performance (§5.3).

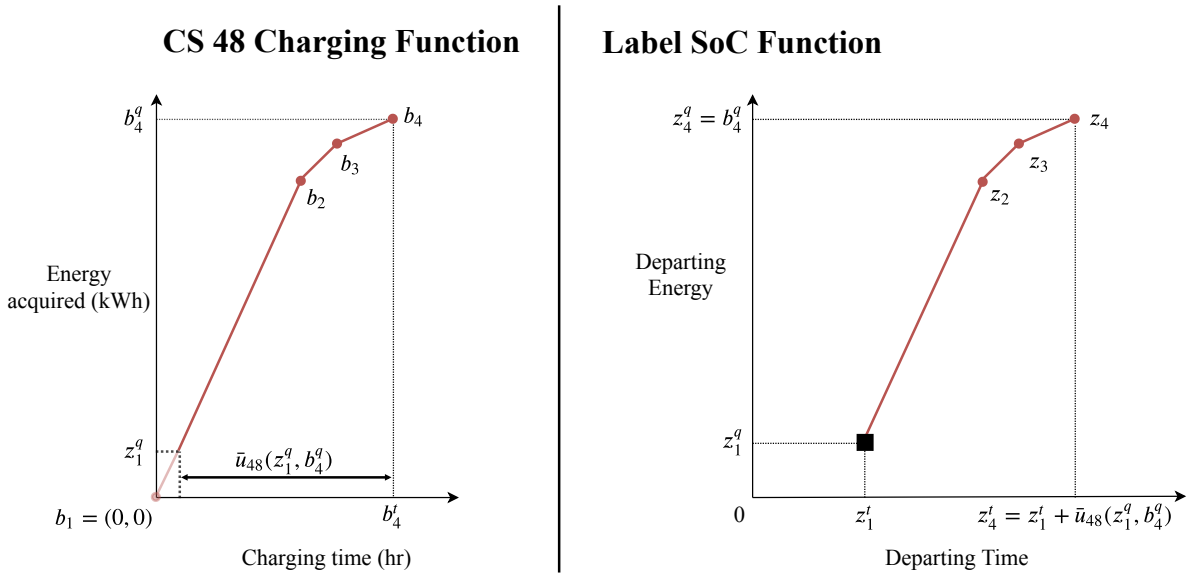

Figure 2: Depicting the creation of new supporting points at CS nodes for the case of CS 48 between customers 33 and 38 in Figure 1. Right shows the SoC function of the label extended to CS 48, with a black square for the initial supporting point (z_1^t, z_1^q). We create additional supporting points (z_2, z_3, z_4) for each breakpoint to which we could charge, shown by circles b_2, b_3, and b_4 in the CS's charging function (left). Axis labels for new supporting point z_4 (right) detail its creation from the decision to recharge to breakpoint b_4.

5.1 Structure

`frvcpy` is a small package (approximately 1000 lines of code) built in the Python programming language; it is available on the Python Package Index and can be installed via "pip install frvcpy." It is comprised of three primary modules: core.py, solver.py, and algorithm.py. core.py consists of class definitions for ancillary objects required in the algorithm's execution such as nodes, labels, and the FRVCP problem instance. solver.py defines the user-facing `Solver` class which is responsible for pre-processing, calling the algorithm, and writing solutions to file. The algorithm itself and its accompanying functions are contained in algorithm.py. Additionally, the package contains the module translator.py. This module provides the ability to generate instance files compatible with `frvcpy` from instances formatted according to the VRP-REP specification. VRP-REP is a community-driven repository for vehicle routing problem data files; see Mendoza et al. (2014) for more details.

Input/output. Users can interact with `frvcpy` using a Python API or via the command-line interface (CLI). As input, `frvcpy` requires a compatible instance, the fixed route for the EV to travel, and the EV's initial energy. Compatible instances are JSON files (or equivalent Python dictionaries) following the schema available on `frvcpy`'s homepage (Kullman et al. 2020b). After execution, the algorithm returns the energy-feasible route and its duration. The returned route is a list of tuples indicating stops' node IDs and the amount of energy to be recharged there (the latter given by the keyword 'None' for non-CS nodes; see Listing 1, Line 24).

Testing the installation. `frvcpy` provides simple testing to determine if its installation was successful. From the command line, users can execute the command `frvcpy-test` to run a suite of tests that

performs an instance translation and solves 134 FRVCPs from the Froger et al. (2019) testbed. The same test suite can also be run in Python via

```
import frvcpy.test
frvcpy.test.runAll()
```

5.2 Example Usage

We provide an example demonstrating the use of `frvcpy` through the Python API.[1] Consider a user with the VRP-REP-compliant instance "vrprep-instance.xml," depicted in Figure 4. The instance contains "fast," "normal," and "slow" CSs whose charging functions are shown in Figure 3. An EV, which begins at the depot with full battery, has been assigned the fixed route $\Pi = (0, 40, 12, 33, 38, 16, 0)$, depicted by the gray arrows in Figure 4. Because the EV does not have sufficient energy to traverse Π without recharging, we solve an FRVCP to determine the optimal insertion of charging operations. We can do this using `frvcpy` as follows:

Listing 1: Example `frvcpy` usage with Python API

```
from frvcpy.translator import translate
from frvcpy.solver import Solver

# translate the VRP-REP instance
frvcp_instance = translate("instances/vrprep-instance.xml")

route = [0,40,12,33,38,16,0]     # route to make energy feasible
q_init = frvcp_instance["max_q"] # EV begins with max battery capacity

# initialize solver with the instance, route, and initial charge
frvcp_solver = Solver(frvcp_instance, route, q_init)

# run the algorithm
duration, feas_route = frvcp_solver.solve()

# write a VRP-REP compliant solution file
frvcp_solver.write_solution("my-solution.xml", instance_name="frvcpy-instance")

print(f"Duration: {duration:.4}")
# Duration: 7.339

print(f"Energy-feasible route:\n{feas_route}")
# Energy-feasible route:
# [(0, None), (40, None), (12, None), (33, None), (48, 6673.379615520617), (38, None), (16,
    None), (0, None)]
```

[1] Readers interested in recreating this example are encouraged to clone the repository from `frvcpy`'s homepage which contains the data discussed here (`https://github.com/e-VRO/frvcpy`; (Kullman et al. 2020b)). The example follows route "route_tc0c40s8cf0_23" for instance "tc0c40s8cf0" from Froger et al. (2019). The full testbed of instances from Froger et al. (2019) is available at `https://www.math.u-bordeaux.fr/~afroger001/documents/data-Improved_formulations_and_algorithmic_components.zip`

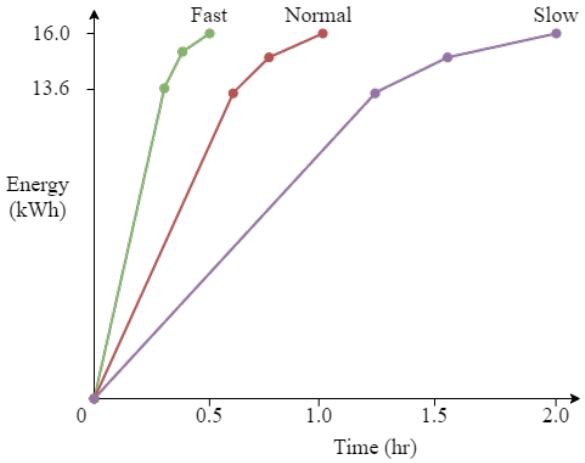

Figure 3: Piecewise linear charging functions for example instance "vrprep-instance.xml."

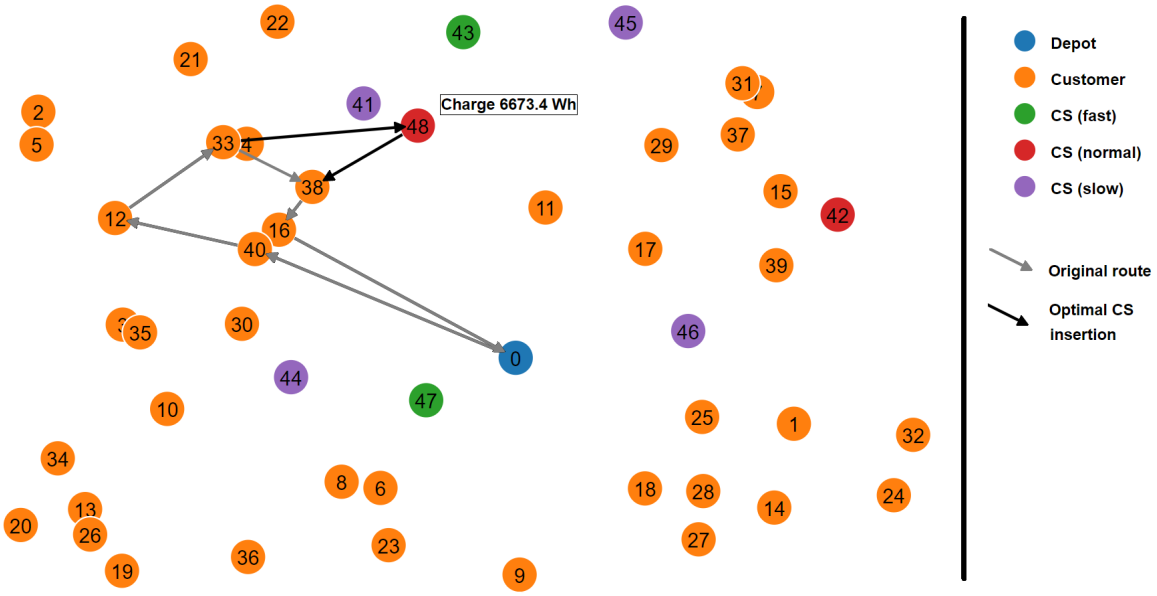

Figure 4: Depiction of example instance "vrprep-instance.xml." We consider an EV given the route shown by the gray arrows. The solution to the FRVCP for this route instructs the EV to recharge at CS 48 between customers 33 and 38 (black arrows).

The solution to the FRVCP instructs the EV to recharge at CS 48 between customers 33 and 38, as depicted by the black arrows in Figure 4 and the printed output on line 24 in Listing 1. This results in a total route duration of about 7.34 hours. We note that detouring to CS 48 actually requires more travel time than detouring to CS 41; however, given that CS 48 offers a faster charging speed, it is ultimately preferred over CS 41 (recharging at CS 41 instead of 48 results in an objective of 7.44 hrs).

The above example would be accomplished with the CLI via

Listing 2: Example `frvcpy` usage with CLI

```
# translate existing VRP-REP instance, write it to file
frvcpy-translate ./instances/vrprep-instance.xml new-frvcp-instance.json

frvcpy --instance=new-frvcp-instance.json --route=0,40,12,33,38,16,0 --qinit=16000
    --output=my-solution.xml
# Duration: 7.339
# Energy-feasible route:
# [(0, None), (40, None), (12, None), (33, None), (48, 6673.379615520617), (38, None), (16,
    None), (0, None)]
```

5.3 Performance

We test the performance of `frvcpy` over the nearly 30,000 instances comprising the testbed from Froger et al. (2019). These instances have a median route length of 10 stops and a median of 18 CSs that may be inserted. On average over the tests, the algorithm has an average run time of 5.6 ms. In addition, in the same tests we find that our translator can translate instances from VRP-REP format in an average of 0.1 s. These results suggest that `frvcpy` requires sufficiently low runtime so as to be included in larger solution schemes for E-VRPs.

6 Conclusion

We introduced `frvcpy`, a Python-based open-source implementation of the labeling algorithm from Froger et al. (2019) for the fixed route vehicle charging problem. The algorithm and our implementation are flexible, able to accommodate realistic problem features such as non-linear recharging times, partial charging decisions, and heterogeneous charging station technologies. Because FRVCPs are often encountered as subproblems of more general EV routing problems, we designed `frvcpy` to be easily embedded in larger solution schemes. To that end, the package offers two modes of interaction, has minimal requirements, and is computationally efficient. Our hope is that `frvcpy` facilitates the solution of E-VRPs, lowering the barrier to entry in this field, and ultimately helping bring about a faster transition to more sustainable transportation practices.

Acknowledgements

Nicholas Kullman is grateful for support from the Société des Professeurs Français et Francophones d'Amérique. This research was also partly funded by the French Agence Nationale de la Recherche through project e-VRO (ANR-15-CE22-0005-01). Justin Goodson wishes to express appreciation for the support from the Center for Supply Chain Excellence at the Richard A. Chaifetz School of Business.

References

Moritz Baum, Julian Dibbelt, Andreas Gemsa, Dorothea Wagner, and Tobias Zündorf. Shortest feasible paths with charging stops for battery electric vehicles. *Transportation Science*, 53(6):1627–1655, 2019.

Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313, 2016.

Aurélien Froger, Jorge E Mendoza, Ola Jabali, and Gilbert Laporte. Improved formulations and algorithmic components for the electric vehicle routing problem with nonlinear charging functions. *Computers & Operations Research*, 104:256–294, 2019.

Gerhard Hiermann, Jakob Puchinger, Stefan Ropke, and Richard F Hartl. The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, 252(3):995–1018, 2016.

Gerhard Hiermann, Richard F Hartl, Jakob Puchinger, and Thibaut Vidal. Routing a mix of conventional, plug-in hybrid, and electric vehicles. *European Journal of Operational Research*, 272(1):235–248, 2019.

Çağrı Koç, Ola Jabali, Jorge E Mendoza, and Gilbert Laporte. The electric vehicle routing problem with shared charging stations. *International Transactions in Operational Research*, 26(4):1211–1243, 2019.

Nicholas D Kullman, Justin C Goodson, and Jorge E Mendoza. Electric vehicle routing with public charging stations. 2019.

Nicholas D Kullman, Martin Cousineau, Justin Goodson, and Jorge E. Mendoza. Dynamic Ridehailing with Electric Vehicles. working paper or preprint, January 2020a. URL `https://hal.archives-ouvertes.fr/hal-02463422`.

Nicholas D Kullman, Aurelien Froger, Jorge E Mendoza, and Justin C Goodson. e-vro/frvcpy v0.1.0, February 2020b. URL `https://doi.org/10.5281/zenodo.3677583`.

Jorge E Mendoza, C Guéret, M Hoskins, H Lobit, V Pillac, T Vidal, and D Vigo. VRP-REP: the vehicle routing community repository. In *Third Meeting of the EURO Working Group on Vehicle Routing and Logistics Optimization (VeRoLog). Oslo, Norway*, 2014.

Alejandro Montoya, Christelle Guéret, Jorge E Mendoza, and Juan G Villegas. A multi-space sampling heuristic for the green vehicle routing problem. *Transportation Research Part C: Emerging Technologies*, 70:113–128, 2016.

Alejandro Montoya, Christelle Guéret, Jorge E Mendoza, and Juan G Villegas. The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B: Methodological*, 103:87–110, 2017.

Office of Transportation and Air Quality. U.S. Transportation Sector Greenhouse Gas Emissions 1990-2017, June 2019. URL `https://nepis.epa.gov/Exe/ZyPDF.cgi?Dockey=P100WUHR.pdf`. EPA-420-F-19-047.

Samuel Pelletier, Ola Jabali, and Gilbert Laporte. 50th anniversary invited article—goods distribution with electric vehicles: review and research perspectives. *Transportation Science*, 50(1):3–22, 2016.

Roberto Roberti and Min Wen. The electric traveling salesman problem with time windows. *Transportation Research Part E: Logistics and Transportation Review*, 89:32–52, 2016.

Maximilian Schiffer and Grit Walther. An adaptive large neighborhood search for the location-routing problem with intra-route facilities. *Transportation Science*, 52(2):331–352, 2017.

Martin Uhrig, Lennart Weiß, Michael Suriyah, and Thomas Leibfried. E-mobility in car parks–guidelines for charging infrastructure expansion planning and operation based on stochastic simulations. In *EVS28 International Electric Vehicle Symposium and Exhibition*, pages 1–12, 2015.