Article

A Mixed Vehicle Routing Problem with Electric and Internal Combustion Vehicles and Common Carriers

Yibo Dang--- , Theodore Tetreault Allen [*] , Manjeet Singh

Posted Date: 19 July 2023

doi: 10.20944/preprints202307.1240.v1

Keywords: vehicle routing problem; electric vehicles; internal combustion vehicles; common carriers; branch and cut and pricing; labelling algorithms.

Article

A Mixed Vehicle Routing Problem with Electric and Internal Combustion Vehicles and Common Carriers

Yibo Dang [1,†], Theodore Allen[2,*] and Manjeet Singh [3]

1 Amazon.com Inc.; dangyibo@amazon.com
2 The Ohio State University - Integrated Systems Engineering; allen.515@osu.edu
3 DHL Supply Chain; manjeet.singh3@dhl.com
* Correspondence: allen.515@osu.edu; Tel.: +01-1614-292-1793 (T.T.A.)
† Current address: 440 Terry Ave N, Seattle, WA 98109, USA

Abstract: With the continuous improvement and observable benefits of electric vehicles (EVs), major logistic companies are introducing more EVs into their conventional fleets. This gives rise to a new type of vehicle routing problem with mixed vehicles, where heterogeneous internal combustion vehicles (ICVs) and electric vehicles are considered in route planning. In addition, certain deliveries that are not efficient on any type of vehicles, are outsourced to third-party common carriers. In this paper, we define this problem as a mixed vehicle routing problem with common carriers (MVRPC). The objective of such problems is to minimize the transportation costs by considering routes with ICVs and EVs, the possibility of visiting recharging stations, outsourcing options, and drivers' layover regulations. This variant of the vehicle routing problems has many practical applications, particularly in the design of long-haul transportation and last-mile delivery services. Effective MVRPC solutions play a key role in promoting the going Green image and optimally allocating resources. The problem has received limited attention in the literature likely because addressing all the needed aspects is especially challenging. To solve the large-scale problem, we develop a branch-and-cut pricing framework that relies on strong cuts and customized labeling algorithms. Numerical experiments highlight the effectiveness of our algorithm. This success can be attributed to tailored critical resources, dynamically bounded bidirectional labeling procedures, strong dominance criteria, and implementation strategies.

Keywords: vehicle routing problem; electric vehicles; internal combustion vehicles; common carriers; branch and cut and pricing; labelling algorithms

1. Introduction

In recent years, we have witnessed a growing interest in solving environmental problems such as greenhouse gas emissions, air pollution, noise, and energy wastes in the logistic industry, especially in the transportation sector. In fact, major logistic companies and smart-city designers have launched different green energy plans, see DHL GoGreen Program (2020). Sustainability has long been a competitive factor: Consumers increasingly consider environmental aspects in their purchasing decisions. The same applies to investors who consult sustainability rankings when looking for viable investment options. In this context, the entire industry is developing ways to achieve a reduction of greenhouse gas emissions and improve their overall environmental performance in transportation optimization. Meanwhile, more and more governmental laws have been passed to regulate the emission of greenhouse gases in the transportation sector, forcing major transfers from reliance on the traditional internal combustion vehicles (ICV) to electric vehicles (EV). In addition, governments and electric car makers have intensified their coverage of required infrastructures to further boost this go-green trend, see Tesla Motors Inc. (2020).

These external factors together with mounting environmental and social awareness have triggered increasing needs for high quality solutions in this field. EV, as a serious alternative for ICV, has been utilized for city-wide deliveries and public transportation, see Asghari and Alehashem (2020).

Nevertheless, EV is not as competitive as ICV from a cost perspective (Goeke et al. (2015)) and from the perspective of driving ranges. The acquisition cost of electric trucks is relatively high today even with tax remedies from local governments. All the long-range trailers in usage, e.g., BYD 8TT, are still of limited diving range within 200 miles (Semi-trailer, Wikipedia (2020)), which requires visits to rechargers en route. Other heavy-duty EVs, e.g. Rivian and Tesla Semi, are still under test production, although they claim comparable long-range capability as ICVs. To overcome the limitations, efficient management of the EV fleet for delivery planning and recharge scheduling is important for companies who are considering incorporating EVs to operate their transportation businesses. The underlying management problem is known as the vehicle routing problem (VRP), whose objective is to minimize route costs by delivering goods from a depot to a given set of customers, see, e.g, Toth and Vigo (2014).

Currently, many researchers started studying the classical VRPs for EVs, which we refer as EVRPs. To our best knowledge, all of the existing EV-related literature are focused on last-mile deliveries or short-distance routing problems without considering required constraints for long-haul transportation business, such as hours of service regulations and outsourcing options. Moreover, except for two recent heuristic works, Goeke et al. (2015); Macrina et al. (2019), no research currently addresses the size and complexity of operating a large mixed dedicated fleet.No logistic company currently relies purely on EV fleets for their business but some of them are gradually introducing EVs into their existing ICV dedicated fleet. Clearly, there is a gap between the simplistic EVRP solutions and a more comprehensive model that covers the relevant real-life constraints.

In our work, we consider the compound vehicle routing problem with mixed fleet (heterogeneous ICVs and EVs) and an outsourcing option to common carriers. We call this problem MVRPC. Compared to ICVs, energy costs for operating EVs are generally lower while the fixed costs per day of using the trucks are relatively high because of the initial acquisition costs. The driver costs for both truck types are assumed to be equal. At times, the total customer demand may exceed the capacity of the dedicated fleet. In cases where customers are located far away from the distribution center or isolated from others, it can be more cost-efficient to subcontract the shipments to a third-party carrier (*common carrier*) by fixed tariffs. Therefore, optimal route planning decisions for MVRPC have to find the cost trade-off among the three transportation modes. Furthermore, many governments worldwide have imposed hours of service regulations for truck drivers to avoid fatigue-related accidents, see Federal Motor Carrier Safety Administration (2011). These regulations ensure that rest periods, which called *layover*, are regularly taken by the truck drivers. This type of problem has many practical applications, particularly at the transportation planning stage for long-haul less-than-truckload transports and last-mile distribution services where subcontracting unprofitable shipments are standard options, see, e.g., Dang et al. (2020). Restricting real-life constraints can be considered to make the planned solutions more concrete and provide informative guidance for daily operations.

In this paper, we consider a real-world transportation planning problem MVRPC that has challenged the logistic industry for many years. In MVRPC, heterogeneous-sized ICVs and a single-sized EV are available at a single depot. The objective is to determine a set of vehicle routes, each starting and ending at the depot, so that each customer is visited exactly once. If certain shipments can be fulfilled more cheaply by a common carrier, then they are outsourced. By making the routing and outsourcing decisions optimally, the overall transportation costs are minimized. Truck capacities and EV battery capacities are observed. Delivery time windows are specified for each shipment in which the drivers must arrive the customer site within the period. Along the trips, drivers must respect the U.S. hours of service regulations to take mandatory layovers once their continuous working periods hit the time limit, which is 14 hours. The rest period is flexible within an interval long enough for all the windows to be fitted. In addition, EV recharges and power consumption are present in this problem.

Unlike in most previous publications, the vehicle usage costs depend on the length of the routes, instead of a fixed number. All the parameters and constraints discussed are deterministic. Despite its practical importance, MVRPC has received limited attention in the literature. No former work has considered the comprehensive features as we do in MVRPC.

We present here an exact algorithm for MVRPC. The algorithm is a branch-and-cut-and price (BCP) procedure based on the column generation framework. Some new features are introduced compared to the BCP algorithms in literature. The methodologies provide significant improvements in solving realistic large-scale vehicle routing problems. The contributions can be summarized as: 1) the first mixed mode optimization problem involving electric vehicles and the first work that involves multiple modes including different vehicle sizes, electric vehicles, and subcontracting option; 2) the demonstration that mixed modes are critical to profitability as well as the enabling of electric vehicles; 3) the first mixed integer linear programming arc-flow formulation for such VRP variants; 4) a new labeling algorithm that deals with layovers and recharging requirements along the route, which has a broad extension to renewable resource problems that could only be solved by meta-heuristics before; 5) three sets of cutting planes that can be introduced to other column generation framework.

The paper is organized as follows. In Section 2, we provide a short discussion of the related literature on each feature of MVRPC. We also review the existing branch-and-cut-and-price algorithms. In Section 3, we provide a problem description, parameter settings, and an arc-flow-based mixed integer linear programming formulation. Sections 4 and 5 describe details about the proposed column generation framework, valid inequalities, and the branch-and-cut-and-price algorithm. The results of our computational experiments are presented in Section 6. Finally, Section 7 presents conclusions and opportunities for future research.

2. Literature Review

Many researchers have studied vehicle routing problems. For more comprehensive reviews, see Laporte (2007); Toth and Vigo (2014). For specific reviews on the vehicle routing problems with time windows (VRPTW), readers are referred to Kallehauge (2008). For reviews on heterogeneous vehicle routing problems, surveys are available in Toth and Vigo (2014), and an important structural study was conducted by Yaman (2006). A thorough survey provided by Pelletier et al. (2016) covers the topics and progress on electric vehicle routing problems in the past five decades. Unfortunately, there is no publication in the literature that covers the exact same features as ours. Therefore, we split the literature review into different segments. In this section, we review the specific parts of the literature that relate to our contributions. First, we review the related literature about green vehicle and mixed fleet routing, VRP with common carriers, and VRP with layover or driver's time regulations. Second, we explore the current state of the art for the branch-and-cut-and-price algorithms in the VRP domain.

2.1. Related VRP Variants

We start by reviewing the green vehicle and mixed fleet routing problems. Conrad and Figliozzi (2011) conducted a bounding study for an EVRP with time windows and fixed recharging time. Fukasawa et al. (2016) proposed a branch-and-cut-and-price algorithm for an EVRP with time windows. In particular, Fukasawa et al. (2016) studied the strengths of different reformulations and developed the 2-cycle-free q-routes column generation framework. Schneider et al. (2014) considered an EVRP with time windows and truck capacities. Like Schneider et al. (2014), we also allow EVs to visit any available rechargers before running out of electricity. We assume that after each visit the battery is fully charged. Schneider et al. (2014) provided a non-linear mixed integer formulation and a hybrid meta-heuristic algorithm. We seek to show that a linear formulation can capture all the elements permitting greater computational efficiency.

It should be noted some authors consider richer options for recharging, while only focusing on a single type of vehicle. Desaulniers et al. (2016) studied the EVRP under four different recharging policies. The authors tailored their BCP algorithms under different scenarios and ran their models on several benchmark problems. Andelmin and Bartolini (2017) considered an alternative fuel charge vehicle with delayed recharging policy. They introduced tighter cycle elimination techniques which allow exact solutions for more than 100 customers in the previously unsolvable instances.

Although many EVRP works focus on constant or linearized electricity consumption, research that involves accurate energy estimations is of importance in providing guidance for the optimization models. For example, Arasu et al. (2019); Demir et al. (2012) both provided parameter estimation models, e.g. for powertrain recharging rates, nonlinear speed and electric consumption, and heuristic algorithms for solving energy-oriented VRPs. As mentioned before, the only two existing publications on similar mixed vehicle routing variants (Goeke et al. (2015); Macrina et al. (2019)) are focused on heuristic approaches. Apparently, none of the previous research provides a clear optimization formulation that can be solved by off-the-shelf solvers.

In terms of the VRP variants with common carriers, most of the research still relies on the field of meta-heuristics. The first work to address outsourcing options for VRPs is Chu C-W (2005). The author considered a single-depot routing problem with a less-than-truckload carrier option. The objective is to minimize the total network cost. Chu C-W (2005) solved the problem by a simple "modified savings" math-heuristic for up to 30 customers. The first work for outsourcing options with a time window is Moon et al. (2012). The authors assign deliveries to vehicles of different carriers in a greedy manner. A heuristic based on a genetic algorithm is proposed in Vidal et al. (2016). Gahm et al. (2017) introduced a new variant in which a heterogeneous dedicated fleet and multiple cost options are considered. This work involved only variable travel costs and three options on the common carrier costs. The problem is solved by a neighborhood search heuristic. More recently, Alcaraz et al. (2019) combined heterogeneous fleet types with last-mile outsourcing options and hours of service regulation (layovers). The authors employed three heuristics: simulated annealing, tabu search, and variable neighborhood search - in solving their problem. They examined the results on up to 100 shipments random instances and studied the stability of the three heuristics. Dang et al. (2020) considered the layover regulations and the outsourcing options in their work and solved the new problem by an ant colony based meta-heuristic approach. The resulting problem is called vehicle routing problem with common carriers and time regulations (VRPCCTR). A recent publication by Dabia et al. (2019,a) is the only one that focuses on exact methods. This work considered complex tariff structures for the outsourcing options. The authors provided two different set-partitioning reformulations, strengthened cuts and modified dominance criteria for the BCP algorithm. Moreover, they conducted relatively structural studies on the impacts of the outsourcing decisions to the routing decisions. Since VRP with common carriers is not thoroughly studied yet, their work yields many opportunities for further study and analysis.

The publications regarding hours-of-service regulations contain more diversified solution methodologies. Alcaraz et al. (2019) reviewed works that considered both layovers and outsourcing options. From their review, this hybrid setting is still mainly solved by meta-heuristic approaches. Goel and Irnich (2014) presented the first exact method for the VRP variants considering layover regulations. The authors proposed a branch-and-price algorithm with a forward labeling search procedure. New dominance rules involving the layover feature were introduced. Their numerical results on Solomon instances suggested the effectiveness of the algorithm. Later, Tilk (2016) improved the work of Goel and Irnich (2014) by adding cutting planes to the master problem and a bidirectional labeling search. The acceleration strategies discussed in this work are also useful in speeding up the pricing procedures.

2.2. Column Generation Algorithms for VRPs

We solve the MVRP to optimality using BCP algorithms. With BCP, an extensive formulation is linearly relaxed. The relaxation works as a master problem which is solved using column generation. Additional valid inequalities are added to strengthen the lower bounds. Integer solutions are finally reached by branch and bound, see Lübbecke and Desrosiers (2005). The column generation process iterates between the linear reoptimization of the master problem and the solution of the pricing subproblem. In general, column generation is one of the most successful large-scale exact methods in solving combinatorial optimization problems. Desrochers et al. (1992) is the first to apply column generation in the context of the VRPTW, resulting in a branch-and-pricing algorithm. Later, Kohl et al. (1999) improved the branch-and-price algorithm by introducing subtour elimination constraints and two-path inequalities which resulted in the branch-and-cut-and-price. An additional line of research focuses on valid inequalities for column generation algorithms based on the variables of the master problem. For example, Jepsen et al. (2008) proposed a set of valid inequalities called subset-row inequalities with the variables in the Dantzig-Wolfe master problem. SR inequalities have been observed to effectively improve the lower bounds. Petersen et al. (2008) discovered how to apply generalized Chavátal-Gomory cuts to speed up the column generation solutions in the context of VRPTW. More recently, Dabia et al. (2019,b) studied the strengths of robust and non-robust cover inequalities applied on BCP algorithms. Some column generation methods often show very slow convergence partly due to heavy degeneracy problems (Rousseau et al. (2007)). Such problems arise when multiple dual solutions are associated with each primal solution. Rousseau et al. (2007) showed that the interior point method helps in stabilizing the column generation algorithms for VRPTWs.

While the master problem for most VRP variants is standard, the pricing algorithms for the subproblems are differentiated. Back in the 1990's and earlier, researchers treated subproblems as a shortest path problem with resource constraints. Exact dynamic programming and 2-cycle eliminations were developed to price the columns with negative reduced cost, see Desrochers et al. (1992). Irnich and Villeneuve (2006) improved the elimination capability from 2-cycles to k-cycles, which increases the lower bounds of the relaxation solutions. However, the procedure is NP-Hard. Nowadays researchers mostly regard the pricing subproblem as an elementary shortest path problem with resource constraints. This consensus is attributed to the contributions of Feillet et al. (2004); Righini and Salani (2006 2008). Their works are cornerstones for the so-called labeling algorithm. The authors also introduced various accelerating techniques such as bounding, state-relaxation, bidirectional labeling search, and decremental state space relaxation, etc. Tilk et al. (2017) further improved the bidirectional labeling algorithm by dynamically adjusting the midpoint so that the number of labels generated by each direction can be controlled to a low level. Another group of researchers are focused on developing heuristic pricing algorithms to speed up the solutions for subproblems. For example, Desrochers et al. (1992) proposed a tabu search heuristic to quickly price the elementary paths with negative reduced cost. By far the most widely used route relaxation method is ng-routes, which was introduced by Baldacci et al. (2011). The method was proved by many researchers to be effective in solving difficult VRPTW instances.

3. Problem Description and Formulation

The MVRPC is an updated version of the transportation mode decision problem discussed in Dang et al. (2020). This strategic planning problem is mostly valued by logistic practitioners. Currently, major transportation companies are quantitatively evaluating the benefits of introducing electric vehicles into their fleets. They are interested in how EVs can change their current mode decisions between the dedicated fleet and common carriers. They also want to know how many more resources (e.g., rechargers, labors,etc.) should be put in order to accomplish the transitions. In this section, we describe the problem in detail and provide a formal mathematical model.

3.1. Problem Settings

We state the assumptions and policies that support the modeling of MVRPC. All dedicated fleet routes (ICVs and EVs) must start and end at the same depot. The average speed of every truck type is assumed to be constant and the same. While we regard the ICV fleet as containing multiple truck types differentiated by heterogeneous sizes, we assume there is only one type of EVs because they are single-sized. The demands for each shipment are inseparable, which means they can either be consolidated on a route or be entirely outsourced to a common carrier. Each shipment that is not outsourced must be visited exactly once by exactly one route traveled by one type of truck. A vehicle must visit the nodes within its corresponding time window such that no waiting time or late delivery is permitted. Furthermore, we assume the time windows for the rechargers and the depot are relaxed, which means they can be visited anytime that they are needed. This setting gives decision makers the data needed to optimally schedule the limited rechargers. Whenever a recharger is visited, the battery is fully recharged and the time will be counted. Yet, such time will not be regarded as working time subjected to the hours of service regulation. To make our problem more general, we do not consider the subtle break period as regulated in Federal Motor Carrier Safety Administration (2011). Instead, we only request that drivers take a layover period between 10 and 14 hours once their uninterrupted working time reaches the 14-hour limit. We realize that some previous literature considered curb weight, road infrastructures, traffic conditions in computing power consumption on a given trip, Arasu et al. (2019); Goeke et al. (2015). In our work, however, we assume that the EV fleet has a constant unit electric consumption per mile. Then the consumption on the arc is linear to the distance.

3.2. Mixed Integer Linear Formulation

The MVRPC is defined on a directed complete graph $G(\bar{I}_0, A)$, where $\bar{I}_0 = \{0, 1, ..., n, e_1, ..., e_m\}$ denotes the set of nodes and $A = \{(i,j) \in \bar{I}_0 \times \bar{I}_0 : i \neq j\}$ is the set of arcs. Node 0 represents the depot at which all the vehicle routes start and end. The nodes in $I = \{1, ..., n\}$ represent the customers, while $E = \{e_1, ..., e_m\}$ is the set of recharging stations. The recharging stations may share the same locations because it is extremely expensive to establish and operate high-voltage rechargers in separate locations. The combination of customers and recharging stations are denoted as $\bar{I}$. Inclusion of the depot is indicated by subscripting the respective set, e.g., I_0, $\bar{I}_0$, E_0. Each node $i \in \bar{I}$ is associated with a demand ω_i, a time window $[a_i, b_i]$, and a service time λ_i. Note that the demand and the service time are positive for $\forall i \in I$ and 0 for $\forall i \in E$. Each arc $(i,j) \in A$ is associated with a non-negative travel distance d_{ij}, and an estimated energy consumption δ_{ij} determined by models discussed in Arasu et al. (2019). We assume constant energy consumption in analyzing applications in the logistic industry in which vehicles usually carry full truckloads. When payload distribution is considered, the energy assumption term δ_{ij} becomes a function of total vehicle weight en arc. Due to operational requirements, a maximum-allowed intra-node distance d^{max} is enforced to rule out infeasible arcs. A mixed dedicated fleet consisting of electric vehicles (EVs) and internal combustion vehicles (ICVs) is available at the depot, where ICVs are usually diesel-powered trucks. Let $V = V_D \cup \{e\}$ be the set of truck types, where V_D denotes the set of heterogeneous diesel trucks and $\{e\}$ is the available homogeneous electric trucks. Each truck of type $k \in V$ has a limited capacity Q_k, a fixed cost f_k, a stop cost p for each delivery, and a finite number of trucks n_k available. For all fleet types $k \in V$, we use the same averaged speed $g = 55\ mph$. Furthermore, for all arcs $(i,j) \in A$ and fleet types $k \in V$, let c_{ijk} be the travel cost from node i to node j, where c_{ijk} is an average of drivers' wages and fuel cost for ICVs and an average of driver's wages and energy cost for electric trucks. Since there is no reason to operate an ICV route from customer sites to recharging stations, or vice versa, we define an infeasible set that represent these arcs. Let A^- represent the infeasible arc set, where $A^- = \{(i,j,k), (j,i,k) : \forall i \in \bar{I}_0, \forall j \in E, k \in V_D\}$.

On each visit to a recharger, electric trucks are recharged to their maximum battery capacity φ. The recharging time depends on the recharging rate α and the remaining electrics on arrival at the recharger. Common carriers are available for which demand can be outsourced. If customer $i \in I$ is outsourced to a common carrier, the corresponding cost is retrieved from the tariff as m_i. According to U.S. governmental hours of service regulations, the truck drivers need to take a layover when their working time reaches the upper threshold $\xi = 14$ hours, where the layover period is specified between $\beta^{lb} = 10$ hours and $\beta^{ub} = 14$ hours (according to industry conventions).

Thereafter, MVRPC can be formulated as a mixed integer linear program (MILP). For every arc $(i,j) \in A$ and $k \in V$, let x_{ijk} be a binary variable that takes value 1 only if fleet type k travels arc (i,j) and 0 otherwise with $x_{ijk} = 0$ is fixed if $(i,j,k) \in A^-$. Let y_i be a binary variable which takes the value 1 only if customer i is subcontracted to a common carrier and 0 otherwise. Continuous variables s_{ik} define the arrival time at node i by truck type k and u_i is the accumulated demands from the depot up to node i. Continuous variables z_i trace the remaining battery level on arrival at node i. t_i is defined as the remaining working time before a layover on arrival at node i. r_{ik} is used to determine the hours

doi:10.20944/preprints202307.1240.v1

taken by the driver for layovers immediately before node i and l_{ik} is the number of layovers needed precedent i. To keep the notations consistent in the formulation, we define $l_{0k} = \{l_{10k}, l_{20k}, ..., l_{e_m 0k}\}$ as a $(n+m) \times 1$ vector for each $k \in V$, where l_{i0k} represents the number of layovers needed on the way back to depot. Note that $r_{ik} = l_{ik} = 0$ for all $k \in V_D$ and $i \in E$.

For clarity, we summarize the parameters and variables of the MILP model in Table 1.

Table 1. Parameters and variables used by the MILP model.

		Descriptions
Parameters	I	$\{1, ..., n\}$ Set of customers in the dataset.
	I_0	Nodes from a particular depot, which includes the customer set I and the depot 0.
	E	$\{e_1, ..., e_i\}$ Set of available electric rechargers, the chargers may share the same locations.
	E_0	$E_0 = E \cup \{0\}$, which is the set of electric rechargers and the depot 0.
	$\bar{I}$	Set of nodes that combines the customer set I and the recharger set E.
	$\bar{I}_0$	Set of nodes that includes the customers, electric rechargers and the depot.
	A	Set of arcs on the directed graph, which includes the arcs that connect customers, depot and electric rechargers.
	A^-	Set of infeasible arcs caused by moving diesel trucks to rechargers.
	V	Set of available fleet types, where V_D denotes the set of heterogeneous diesel trucks and e represents the electric truck.
	n_k	Numbers of available vehicles of fleet type $k \in V$.
	Q_k	Weight capacity of fleet type k.
	ω_i	A positive demand associated to node $i \in \bar{I}$, where $\omega_i = 0 \; \forall i \in E$.
	$[a_i, b_i]$	Node i needs to be fulfilled between the earliest delivery time a_i and the latest delivery time b_i.
	λ_i	Service time at node i, where $\lambda_i = 0 \; \forall i \in E_0$.
	d_{ij}	Travel distance from node i to node j.
	δ_{ij}	Energy consumption amount on arc (i, j).
	d^{max}	Maximum-allowed intra-node distance for an arc (i, j).
	α	Recharging rate.
	φ	Battery capacity for each electric truck.
	M, M'	Big constants associated with the constraints.
	f_k	Fixed cost per layover for using fleet type k.
	c_{ijk}	Travel cost from node i to node j by fleet type k.
	g	Average speed of the dedicated fleets.
	m_i	Rated common carrier cost for shipment i.
	p	Basic stop cost conducted by each delivery.
	$[\beta^{lb}, \beta^{ub}]$	Layover time required for the drivers at least β^{lb} hours and at most β^{ub} hours.
	ξ	Maximum-allowed working time for the truck driver before a layover.
Variables	x_{ijk}	Binary decision variable indicating if fleet type k takes arc (i, j).
	y_i	Binary decision variable indicating if shipment i is outsourced to a common carrier.
	s_{ik}	Continuous decision variable that traces the completion time for $i \in \bar{I}_0$ by fleet type k.
	u_i	Continuous decision variable that specifies the accumulated truck load from the depot up to node $i \in \bar{I}$.
	z_i	Continuous decision variable specifying the remaining battery capacity on arrival at vertex i.
	t_i	Continuous decision variable specifying the remaining working time per layover on arrival at vertex i.
	r_{ik}	Continuous decision variable specifying the layover time precedent node i.
	l_{ik}	Integer decision variable specifying the number of layovers immediately precedent node i. $l_{0k} = \{l_{10k}, l_{20k}, ..., l_{e_m 0k}\}$

This problem $(\mathcal{P})$ can be represented by the following arc flow formulation:

$$\min \quad \sum_{k \in V} \{ f_k (\sum_{i \in \bar{I}_0} l_{ik} + \sum_{i \in \bar{I}} x_{i0k}) + \sum_{(i,j) \in A} [(c_{ijk} + p) x_{ijk}] - \sum_{i \in \bar{I}} p \cdot x_{i0k} \} + \sum_{i \in I} m_i \cdot y_i \tag{1}$$

$$\text{s.t.} \quad \sum_{k \in V_D} \sum_{j \in I_0} x_{ijk} + \sum_{j \in \bar{I}_0} x_{ije} + y_i = 1, \quad \forall i \in I, i \neq j \tag{2}$$

$$\sum_{i \in I} x_{0ik} \leqslant n_k, \quad \forall k \in V_D \tag{3}$$

$$\sum_{i \in \bar{I}} x_{0ie} \leqslant n_e \tag{4}$$

$$\sum_{j \in \bar{I}_0} x_{ije} \leqslant 1, \quad \forall i \in E, i \neq j \tag{5}$$

$$\sum_{i \in I_0, i \neq j} x_{ijk} = \sum_{h \in I_0 h \neq j} x_{jhk}, \quad \forall k \in V_D, \forall j \in I \tag{6}$$

$$\sum_{i \in \bar{I}_0, i \neq j} x_{ije} = \sum_{h \in \bar{I}_0 h \neq j} x_{jhe}, \quad \forall j \in \bar{I} \tag{7}$$

$$b_i \geqslant s_{ik} \geqslant a_i \quad \forall i \in \bar{I}, \forall k \in V \tag{8}$$

$$s_{je} \geqslant s_{ie} + (\frac{d_{ij}}{g} + \lambda_i) + r_{je} + \alpha \cdot (\varphi - z_i) - (1 - x_{ije}) \cdot M, \qquad \forall i \in E, \forall j \in \bar{I}_0, i \neq j \tag{9}$$

$$s_{jk} \geqslant s_{ik} + (\frac{d_{ij}}{g} + \lambda_i) \cdot x_{ijk} + r_{jk} - (1 - x_{ijk}) \cdot M', \qquad \forall k \in V, \forall i \in I_0, \forall j \in \bar{I}_0, i \neq j \tag{10}$$

$$z_i - \delta_{ij} \cdot x_{ije} + \varphi \cdot (1 - x_{ije}) \geqslant z_j, \quad \forall i \in I, \forall j \in \bar{I}_0 \tag{11}$$

$$\varphi - \delta_{ij} \cdot x_{ije} \geqslant z_j, \quad \forall i \in E_0, \forall j \in \bar{I}_0, i \neq j \tag{12}$$

$$\xi \cdot l_{jk} + t_i - (\frac{d_{ij}}{g} + \lambda_i) \cdot x_{ijk} + \xi \cdot (1 - x_{ijk}) \geqslant t_j, \quad \forall k \in V, \forall i \in \bar{I}, \forall j \in \bar{I}_0 \tag{13}$$

$$\xi \cdot (l_{jk} + 1) - \frac{d_{0j}}{g} \cdot x_{0jk} \geqslant t_j, \quad \forall k \in V, \forall j \in \bar{I} \tag{14}$$

$$\beta^{ub} \cdot l_{ik} \geqslant r_{ik} \geqslant \beta^{lb} \cdot l_{ik}, \quad \forall k \in V, \forall i \in \bar{I}_0 \tag{15}$$

$$u_i - u_j + \sum_{k \in V} Q_k \cdot x_{ijk} \leqslant \omega_i \cdot y_i - \omega_j + \sum_{k \in V} \sum_{j \in \bar{I}_0} x_{ijk} \cdot Q_k, \quad \forall i, j \in \bar{I} \tag{16}$$

$$\sum_{k \in V} Q_k \sum_{j \in \bar{I}_0} x_{ijk} + \omega_i \cdot y_i \geqslant u_i \geqslant \omega_i, \quad \forall i \in \bar{I} \tag{17}$$

$$d_{ij} \cdot x_{ijk} \leqslant d^{max}, \quad \forall (i,j) \in A, i \neq 0, j \neq 0, \forall k \in V \tag{18}$$

$$x_{ijk}, y_i \in \{0,1\} (\text{where } x_{ijk} = 0, \forall (i,j,k) \in A^-) \quad \forall (i,j) \in A, \forall i \in \bar{I}, \forall k \in V \tag{19}$$

$$l_{jk} \in Z^*; t_j, z_j, u_i, r_{jk}, s_{jk} \in R^*, \quad \forall k \in V, \forall i \in \bar{I}, \forall j \in \bar{I}_0. \tag{20}$$

The objective function (1) minimizes the total fixed costs, travel costs, and outsourcing costs. Constraint (2) ensures that a customer node that is not outsourced is visited exactly once by a type-k truck. Constraints (3) and (4) ensures that the number of trucks in use is less than the available trucks for each type k. Constraint (5) guarantee that if a recharger i is visited, it can be used at most once. Constraints (6) and (7) are the flow conservation constraint of the trucks. Constraints (8)-(10) guarantee the customer time windows are respected, with variable layovers and recharging time added if necessary. It is important to set M and M' as small as possible so that constraints (9) and (10) are tight enough but still valid. Here, we let $M = \frac{d_{ij}}{g} + \lambda_i + \alpha \cdot \varphi + T \cdot \beta^{ub} + b_i$, where T is a constant parameter that equals 1 for all $i, j \neq 0$ and equals an integer that refers to the largest possible layover numbers between depot and any node. Furthermore, we let $M' = T \cdot \beta^{ub} + b_i$. Constraints (11) and (12) specify the electric consumption on each arc, which follow the non-decreasing rule. Constraints (13) and (14) trace the moves of truck type k along each arc to add necessary layovers. Again, the layover period and work time follow the non-decreasing rule node by node. Next, constraint (15) ensures the dynamic layover time added to truck type k obeys the hours of service regulation. The subtour elimination constraints (16), (17) are extended from Dang et al. (2020) to the heterogeneous case. Constraint (18) ensures the intra-node distances are feasible. Finally, we define the domains for each decision variable in constraints (19) and (20).

4. The Set Partitioning Formulation

The set partitioning (SP) formulation has been successfully applied to solve large scale CVRP instances and crew scheduling problems because of its tight linear programming (LP) which motivates our choice of an SP reformulation for MVRPC.In addition to the parameters and variables shown in Table 1, the following additional notations are needed. We define Ω_k as the set of feasible dedicated fleet routes that can be fulfilled by vehicle type $k \in V$. A dedicated fleet route is feasible for vehicle type k if and only if the delivery windows, capacity constraints, layover requirements, and the intra-node distance are all satisfied. Then, we define $\Omega = \bigcup_{k \in V} \Omega_k = \bigcup_{k \in V_D} \Omega_k \cup \Omega_e$ to be the set of all columns (feasible routes). For each route $q \in \Omega$, let c_q denote the cost, which includes the fixed cost and the variable travel cost. Let η_{ijq} be the number of times arc (i, j) is visited by route q and x_q be a binary decision variable that takes value 1 if and only if route q is included in the current solution. Again, $\eta_{ijq} = 0$ such that $(i, j, q) \in A^-$. The MVRPC is formulated as the following SP problem $(\mathcal{R})$:

$$\min \quad \sum_{q\in\Omega} c_q \cdot x_q + \sum_{i\in I} m_i \cdot y_i \tag{21}$$

$$\text{s.t.} \quad \sum_{q\in\Omega_k:k\in V_D} \sum_{j\in I_0} \eta_{ijq} \cdot x_q + \sum_{q\in\Omega_e} \sum_{j\in \bar{I}_0} \eta_{ije} \cdot x_q + y_i = 1, \quad \forall i \in I \tag{22}$$

$$\sum_{q\in\Omega_k} x_q \leqslant n_k, \quad \forall k \in V \tag{23}$$

$$x_q \in \{0,1\}, y_i \geqslant 0, \quad \forall q \in \Omega, \forall i \in I. \tag{24}$$

The objective function (21) minimizes the cost of the selected dedicated fleet routes (columns) and the cost charged for outsourcing shipments to the common carriers. Constraint (22) is the degree constraint that is projected to the space of (i, j, q), which ensures that each shipment is either served by a type-k truck or is outsourced to a common carrier. Constraint (23) guarantees that the number of occupied trucks does not exceed the available number n_k of each fleet type. Next, constraint (24) sets the domain of decision variables.

4.1. *Cutting Planes on Restricted Master Problem*

To further tighten the LP bounds obtained from the SP formulation, we can add additional constraints satisfied by all feasible solutions into $(\mathcal{R})$. One way to obtain additional constraints in the (x_q, y_i)-space is through the transformation of valid inequalities in the space of other variables. For examples, some well-known valid inequalities in the (x_{ijk})-space derived for HVRP are also valid for MVRPC. The reason is that their three-index flow formulations share the same feasible set. However, those valid inequalities are no longer strong because of the existence of y_i. When the (x_{ijk})-variables and the (x_q)-variables both refer to the same feasible route, the coupling constraints between them can be defined as in constraints (25). Then, any valid inequality in the (x_{ijk})-space can be lifted and transformed to a valid inequality in the (x_q, y_i)-space.

$$x_{ijk} - \sum_{q\in\Omega_k} \eta_{ijq} \cdot x_q = 0, \quad \forall (i, j, k) \notin A^- \tag{25}$$

Constraints (23) are the obvious projection of constraints (3) and (4) from the (x_{ijk})-space to the (x_q)-space. Considering that constraints (3) and (4) hold and $\sum_{i\in I} x_{i0k} = \sum_{q\in\Omega_k} x_q \sum_{i\in I} \eta_{i0q} = \sum_{q\in\Omega_k} x_q$ holds, we thus have $\sum_{q\in\Omega_k} x_q \leqslant n_k$, which is exactly constraints 23.

4.1.1. Chvátal-Gomory Rank-1 Cut

Leveraging this fact and constraints (25), we first study the Chvátal-Gomory (CG) rank-1 cuts that can be applied to $(\mathcal{R})$ and strengthen the LP relaxation.

Consider a general IP and its polyhedron.

$$\min\{cx : Ax \leqslant b, x \geqslant 0, x \in \mathbb{Z}^n\} \tag{26}$$

$$P_{IP} = conv\{x \in \mathbb{Z}^n : Ax \leqslant b, x \geqslant 0\} = conv(P_{LP} \cap \mathbb{Z}^n) \tag{27}$$

A Chavátal-Gomory (CG) cut is a valid inequality for P_{IP}:

$$\lfloor uA \rfloor x \leqslant \lfloor ub \rfloor \tag{28}$$

where A is a $m \times n$ integral matrix and $b \in Z^m$. The continuous parameter $u \in R^m_+$ arises from the valid CG cuts (Conforti et al. (2014). Note that CG cuts depend on P_{LP} and not directly on P_{IP}, i.e., a different LP formulation of the same IP problem can produce different CG cuts (Fischetti et al. (2007)). Inequality (28) is said to have rank-1 with respect to the polyhedron P_{LP}. Then the separation optimization problem is defined as follows. Given a point $x^* \in P_{LP}$, the CG separation problem consists of finding a CG cut that is violated by x^*, i.e., finding $u \geqslant 0$ such that $\lfloor uA \rfloor x^* > \lfloor ub \rfloor$, or proving that no such u exist. Given the fractional input x^* to be separated, the maximally violated CG cut for some non-negative u can be found by solving the following MIP (Conforti et al. (2014)):

$$\begin{aligned}
\max \quad & \lfloor uA \rfloor x' - \lfloor ub \rfloor && (29)\\
\text{s.t.} \quad & \lfloor uA_j \rfloor \leqslant uA_j, \quad \forall j \in N && (30)\\
& \lfloor ub \rfloor > ub - 1 && (31)\\
& u_i \geqslant 0, \quad \forall i \in M && (32)\\
& \lfloor uA_j \rfloor, \lfloor ub \rfloor \in \mathbb{Z}, \quad \forall j \in N. && (33)
\end{aligned}$$

It was shown by Gomory that every fractional vertex x^* of P_{LP} associated with a certain basis B of (A, I) can be cut off by the CG cut in which u is chosen as the i^{th} row of B^{-1}, see Gomory (1958). Moreover, since only basis variables with non-zero values can contribute to the violation of the CG rank-1 cut, all zero-valued variables can be left out of the formulation which is how CG cuts are added to the simplex algorithm. As for MVRPC, the CG rank-1 cuts are based on constraints (22). With $u \geqslant \mathbf{0}$, the CG rank-1 cut with respect to the restricted master problem (RMP) (21) and (22) is given as:

$$\sum_{q \in \Omega_k : k \in V_D} \lfloor \sum_{i \in I} u_i \sum_{j \in I_0} \eta_{ijq} \rfloor x_q + \sum_{q \in \Omega_e} \lfloor \sum_{i \in I} u_i \sum_{j \in \bar{I}_0} \eta_{ijq} \rfloor x_q + \sum_{i \in I} \lfloor u_i \rfloor y_i \leqslant \lfloor \sum_{i \in I} u_i \rfloor \tag{34}$$

Proposition 4.1. *CG rank-1 cuts (34) are valid for the SP problem $(\mathcal{R})$.*

Proof. The proof follows the definition of CG rank-1 cuts, see Chvátal (1973). Consider constraints (22) with a vector $u \geqslant \mathbf{0}$, then we have $uAx \leqslant \sum_{i \in I} u_i \Rightarrow \lfloor uAx \rfloor \leqslant \sum_{i \in I} u_i$. This step actually weakens the inequality. To avoid badness, we can designate u, s.t. uA is an integer, so $\lfloor uA \rfloor = uA$. Also, since $x_q, y_i \in \{0, 1\}$, we have $\lfloor uA \rfloor x \leqslant \lfloor \sum_{i \in I} u_i \rfloor$, where A stands for the matrix formed by η_{ijq}'s and 1's. This is a standard format of inequality (34). □

Therefore, given a fractional solution x' from the RMP, the separation problem of finding the most violated CG rank-1 cut is again the MIP problem (29). Without loss of generality, we can choose the $u_i's$ equal to ω_i, so that the weight of each node is captured in (34). Since the CG cut (34) is valid for the entire customer set I, it should also hold for $S \subset I$. However, the opposite direction does not necessarily hold, as we will see in *Example 1*. This results is a so-called sub-row (SR) inequality, see Jepsen et al. (2008). Since SR inequality has been successfully implemented to related BCP algorithms by many previous researchers, e.g., Jepsen et al. (2008) and Dabia et al. (2019,a), we can also extend the inequality to MVRPC as a special case of CG rank-1 cuts.

$$\sum_{q \in \Omega} \lfloor \frac{1}{\phi} \sum_{i \in S} \omega_i \sigma_{iq} \rfloor x_q + \sum_{i \in S} \lfloor \frac{\omega_i}{\phi} \rfloor y_i \leqslant \lfloor \frac{\sum_{i \in S} \omega_i}{\phi} \rfloor \tag{35}$$

where $S \subseteq I$ and $0 < \phi \leqslant \omega(S)$. σ_{iq} is a constant that counts the number of times node i is visited by route q. This is equivalent to the set of CG cuts (34), where $|S|$ of the ω_i's are divided by ϕ and the rest are set to 0, i.e., u is designed as a very sparse vector.

Example 1. *Let's consider a simple numerical example with u_i's as in constraint (35). The route decision variables x_q's and the subcontracting decision variables y_i's are all fractional. Let $u_i = \frac{\omega_i}{\phi}$, where $0 < \phi \leqslant |I| - 1$. Let $k \in V$. Assume in the current iteration, we have 3 columns generated $(q = 3)$ and 5 shipments in. Table 2* *reports the fractional solutions and associated parameters obtained by solving the RMP. Let's consider a CG rank-1 cut of the form (34) and a SR cut of the form (35) with $S = \{2,3\}$ and $\phi = 5$. By substituting the variables and parameters with the real values, we have the CG cut:*

$$2x_{R1} + 2x_{R2} + x_{R3} + y_1 + y_5 \leqslant 3 \tag{36}$$

While the corresponding SR inequality is:

$$x_{R1} + x_{R2} \leqslant 1 \tag{37}$$

With the fractional values added, we have the CG cut be $3.1 \leqslant 3$, but the SR inequality be $0.7 \leqslant 1$. Therefore, the derived CG cut is violated whereas the SR inequality is not. However, if we choose $s = \{1,5\}$, then the SR cut is $x_{R1} + x_{R2} + x_{R3} + y_1 + y_5 \leqslant 2$, which indicates $2.4 \leqslant 2$. The SR cut is more violated than the general CG rank-1 cut. This example indicates that SR cut is not always robust and the separation optimization are important to identify efficient CG rank-1 cuts at each branching node. In fact, Fischetti et al. (2007) suggested strong cuts correspond to "minimal" CG multiplier vectors with as few nonzero entries as possible because of the cumbersome separation time and the cut strength. In addition, the experimental results in Petersen et al. (2008) showed that there is barely no difference between the LP bounds of CG rank-1 and SR cuts on VRPTW instances. Therefore, to speed up the cut generations, we use SR inequalities in the BCP algorithms.

Table 2. Fractional decisions for the outsourcing options (left panel) and the dedicated fleet routes (right panel).

i	ω_i	y_i
1	5	0.7
2	3	0.3
3	2	0.1
4	2	0.1
5	5	0.8

q	x_q	k	Truck loads	Route
R1	0.3	2	12	1,2,3,4
R2	0.4	1	12	2,3,4,5
R3	0.2	e	9	3,4,5

As in Jepsen et al. (2008); Petersen et al. (2008), we consider a subset $S \subset I$ with three shipments. The subset row inequalities are separated by enumerating all subsets of three shipments and checking for each subset whether the corresponding inequality is violated. Violated inequalities are added to the RMP. And the separation problem of SR inequalities is defined as follows: given the current LP solution x where $x_q < 1$ for all $q \in \Omega$, find the most violated inequality (35), such that:

$$\max \sum_{q\in\Omega} \lfloor \frac{1}{\phi} \sum_{i\in S} \omega_i \sigma_{iq} \rfloor x_q + \sum_{i\in S} \lfloor \frac{\omega_i}{\phi} \rfloor y_i - \lfloor \frac{\sum_{i\in S} \omega_i}{\phi} \rfloor \tag{38}$$

Jepsen et al. (2008) has proved that optimizing the problem (38) is NP-Complete. Therefore, we introduce simple separation heuristic ideas to identify the subset S that leads to the most violated SR inequality in each column generation iteration. Firstly, we check any three shipment combinations from every priced column (route). Then, we add the top three violated SR inequalities to the RMP. Meanwhile, we store the ten nearest shipments in an adjacent list for each shipment $i \in I$, so that each possible subset of three is checked for violation of the SR inequality. In this way, the algorithm will not be slowed down by optimizing the separation problem in a single iteration.

4.1.2. 2-path Inequalities

2-path inequalities were introduced by Kohl et al. (1999) for VRPTW and have been widely applied in multiple BCP algorithms in solving different variants of VRPs, e.g., Desaulniers et al. (2016). Similar to Desaulniers et al. (2016), we define $S \subseteq I$ to be a subset of vertices in $\bar{I}_0$ that includes at least one shipment, and assume it is not possible to serve all shipments in S with a single truck. Here, we only consider the shipments to simplify the problem because we assume that no feasible ICV route will pass through a recharger. The corresponding 2-path inequality is given by:

$$\sum_{q\in\Omega}\sum_{(i,j)\in(I\backslash S,S)} \eta_{ijq}\cdot x_q \geqslant 2 \tag{39}$$

Following the separation heuristic proposed by Kohl et al. (1999), we generate the candidate sets S with the cardinality $|S| \leqslant 6$ that satisfies $\sum_{q\in\Omega}\sum_{(i,j)\in(I\backslash S,S)}\eta_{ijq}\cdot x_q < 2$. In other words, we generate S such that the flow entering S is the current solution of the RMP. Next, each subset S is tested to determine whether or not more than one truck is needed. If $\sum_{i\in S}\omega_i > Q_{max}$, then a violated inequality (39) is identified, where Q_{max} is the capacity of the maximal fleet size. If none of the violations can be found, we replace Ω with Ω_k and Q_{max} with Q_k ($k \in V$) to incorporate the checks for each fleet type. In addition, if the capacity fits, we solve an elementary shortest path problem with time windows over $S \cup \{0, 0'\}$, where $0'$ is introduced here as a duplicate of the depot that represents the destination of the route. The reduced cost of each arc (i, j) is set as -1 and the truck capacity is Q_{max}. When the solution cost is less than $-|W| = -6$, it indicates this subset of shipments can be served by a single truck. Since solving this shortest path problem is extremely time-consuming, we only apply the solution procedure for the largest truck to identify violated inequalities (39).

4.1.3. Generalized Large Multistar Inequalities

Multistar inequalities are exponential VIs for Capacitated VRP problems and have been successfully implemented in different branch-and-cut and branch-and-cut-and-price algorithms, see Fukasawa et al. (2006 2016); Letchford et al. (2002); Yaman (2006). This type of VIs cut the graph $G\backslash\{0\}$ into two sets, i.e., selected subset $S \subseteq \bar{I}$ and its complementary set $\bar{I}\backslash S$. So, instead of considering the flow into a single node, these VIs imply that the remaining capacity on the trucks entering S should be at least the demand flow of S and the flows that satisfy the demand in the node sets outside S. Multistar inequalities can be adapted to the RMP by considering the set partitioning variables and introducing the demand outsourcing variables y_i. Therefore, we have:

$$\sum_{q\in\Omega}\sum_{(i,j)\in(\bar{I}\backslash S,S)}(Q_q-\omega_i)\cdot\eta_{ijq}x_q \geqslant \omega(S) - \sum_{i\in I}\omega_i\cdot y_i + \sum_{q\in\Omega}\sum_{(i,j)\in(S,\bar{I}\backslash S)}\omega_j\cdot\eta_{ijq}x_q \tag{40}$$

To separate these inequalities, we follow the heuristic that is similar to a separation procedure for *2-path inequalities* (Kohl et al. (1999)). We first enumerate all subsets $S \subseteq \bar{I}$ such that $|S|$ is less than or equal to nine, where the flow entering S is less than two, and constraints (22) are satisfied for all $i \in S$. If the inequality (40) is violated, then we store it and repeat the procedure until we reach a specified number of iterations set to 200. If not, we check whether the energy consumption on path $S \cup \{0, 0'\}$ is more than the battery capacity times the possible rechargers on the path plus one. If the consumption exceeds the threshold, then this flow is not feasible for an EV.

Note that both the 2-path inequalities and the generalized multistar inequalities are robust, therefore incorporating them requires no adjustment in the structure of the pricing problem. The violated 2-path inequalities and generalized multistar inequalities are added to the RMP. The shadow prices associated with the two types of cuts for subset S must be subtracted from the reduced cost of all arcs $(i, j) \in (I\backslash S, S)$.

As researched by Jepsen et al. (2008), CG rank-1 cuts (SR inequalities) are nonrobust inequalities, meaning that adding them to the RMP destroys the structure of the pricing problems. The reason is that the indicator parameters η_{ijq} are summed over the rows and the reduced cost of the violated path can only be observed when the path is generated. Thus, the dual variables cannot be subtracted directly by the related arc $(i, j) \in A$. Details of how to implement the cuts in the pricing problem will be discussed in Section 5.

5. Column Generation and Pricing Problems

Given a primal LP optimal solution for the restricted RMP, the pricing problem tries to find a dedicated fleet route (a column) with a negative reduced cost if there exists one. We use the column generation framework (Lübbecke and Desrosiers (2005)) to solve the LP relaxation of (21)-(24), i.e., we consider a restricted master problem containing a subset of the columns p and generate additional columns as needed. Starting with a state where all shipments are outsourced ($y_i = 1, \forall i \in I$)), we generate additional columns for the master problem. This framework requires solving k similar subproblems for each fleet type, while the remaining unrouted shipments are handled by the y_i's in the master problem.

5.1. Dedicated Fleet Pricing Problem

In this section, we describe the pricing problem in detail. Let $\mu_i \in \mathbb{R}$ ($\mu_i \geqslant 0$) for $i \in I$ be the dual variables associated with constraints (22) and $\gamma_k \in \mathbb{R}$ ($\gamma_k \leqslant 0$), $k \in V$ be the dual variables associated with constraints (23). Let $\bar{c}_q$, $q \in \Omega$ be the reduced cost of a complete path q that is operated by a type k truck, i.e., a closed loop route that starts and ends at the depot. The pricing problem without strengthened cuts for the problem $(\mathcal{R})$ can be formulated as follows.

$$\min_{q \in \Omega_k : k \in V_D} \bar{c}_q = c - A^T x = c_q - \sum_{(i,j) \in A} \eta_{ijq} \cdot \mu_i + \gamma_k \tag{41}$$

We first write down the calculation of c_q in (41) explicitly, $c_q = f_k + f_k \cdot l_q + \sum_{(i,j) \in A} (c_{ijk} + \frac{1}{2} p) \cdot \eta_{ijq}$. Here, we use l_q to denote the number of layovers spent on path q. A feasible route in Ω is an elementary path starting and ending at the depot in which every node is visited exactly once, i.e., no loops or subtours exists. However, not all elementary paths are feasible as they may violate the time windows, truck capacities, battery capacity, or the intra-node distance constraints. The recharging time and layover time should be applied on the elementary paths if necessary to check if time windows of the shipments are respected. These resource constraints are required to be defined and traced while extending the paths. The functions that record the constraint-related resources are called *resource extension functions* (REFs , see Desaulniers et al. (1998)).

Problem (41) aims at finding a feasible dedicated fleet route with vehicle type k with the minimum reduced cost. The arc cost c_{ijk} for $(i,j,k) \in A \backslash A^-$ is replaced by a defined edge cost $\bar{c}_{ijk} = c_{ijk} + \frac{1}{2} p - \mu_i$, where $\mu_0 = \mu_{0'} = 0$. In order to include the three types of cuts, the reduced cost of the pricing problem needs to be adjusted accordingly. Firstly, let π be the dual value associated with the 2-path cuts (39) and τ be the dual value associated with the generalized multistar cuts (40). Then, for a given shipment candidate set S, π and τ are subtracted from the reduced cost $\bar{c}_{ijk}$ of all arcs $(i,j) \in (I \backslash S, S)$, such that $(i,j,k) \notin A^-$. Therefore, we have the modified reduced cost for columns of type-k vehicle in the RMP as follows.

$$\hat{c}_q = \bar{c}_q - \sum_{(i,j) \in (I \backslash S, S)} \eta_{ijq} \cdot \tau - \sum_{(i,j) \in (I \backslash S', S')} \eta_{ijq} \cdot \pi \tag{42}$$

Similarly, consider a violated SR inequality of the form (35), defined by a subset of three shipments $S_r \subset I$, $\phi \leqslant |S_r|$, and $\chi_r < 0$ as its associated dual value. Since χ is negative, it will act as a penalty. Because the terms of SR inequalities are summed over the "rows", the penalties cannot be added directly to each arc. The value χ must be subtracted from the reduced cost $\hat{c}_q$ each time a new column q is passed to the master problem but the column revisits shipments in S_r that define the violated SR cuts. Therefore, an additional resource is required for each violated SR cut to indicate the regeneration times of the shipments. Note that we know such a column is regenerated only if completed, because the total demands fulfilled and other shipments on route could possibly change the left-hand side of the cuts. We have the updated reduced cost for the type-k column as below.

$$\tilde{c}_q = \hat{c}_q - \lfloor \frac{1}{\phi} \sum_{i \in S} \omega_i \sigma_{iq} \rfloor \cdot \chi_r \tag{43}$$

In this setting, the subproblem corresponds to an elementary shortest path problem with resource constraints (ESPPRC) that can be solved by dynamic programming using a labeling algorithm, see Feillet et al. (2004). This well-known method has been successfully applied to many problems in routing and scheduling. Compared with labeling algorithms for other VRP variants in the literature, solving the pricing problem for MVRPC is differentiated by the following factors: 1) multiple REFs are extended separately by truck type k; 2) additional resource constraints are extended to satisfy the working time regulations, recharging needs, and recharger information; 3) necessary layovers and recharging add difficulties to the dominance tests. Moreover, we must be careful about the complexity arisen by these additional constraints. In fact, Dror (1994) has proved that the ESPPRC is strongly NP-hard for VRPTW.

Theorem. The ESPPRC for MVRPC is strongly NP Hard, whereas the SPPRC without elementarity can be solved by route-relaxed dynamic programming in pseudo-polynomial time. *Proof.* The idea of NP-hardness proof is attributable to Dror (1994). The dynamic programming backward recurrence for a type-k truck of ESPPRC for MVRPC can be stated as:

$$F_j^q(S,t) = \min_{(i,j,k) \in A \setminus A^-} \{F_i^q(S - \{j\}, t') + f_k \cdot l_{jk} + c_{ijk} | t' + t_{ij} + r_{jk} + h \leqslant t; b_i \geqslant t \geqslant a_i;$$
$$\sum_{i \in S} \omega_i \leqslant Q_k; h \geqslant \alpha(\phi - z_j), \forall j \in E_0; z_j \geqslant \phi, \forall k \in V_D; z_j \leqslant \phi - \delta_{ij}\} \tag{44}$$

where $F_j^q(S,t)$ denotes the minimal reduced cost of the partial route going from the depot up to node i, S is the node set selected for route q, and t is the time ready to leave node i while t_{ij} is the time spent on the arcs to travel and service. From the constructions of the recurrent function and the graph G, if there exists a feasible shortest path solution from 0 to $0'$ for VRPTW on graph G, then there is also a feasible ESPPRC solution satisfying the constraints in (44) for MVRPC. Since the shortest path problem is NP-hard, and we found a polynomial transformation from Dror (1994) to our elementary path problem, our problem is NP-hard as well. Note that, the exact dynamic programming approach in solving ESPPRC yields an exponential number of possible states (S,t).

However, if we shrink the state space by retaining only the m closest shipments to the current end point of the partial path q, which can be visited before i, and relax the elementary path requirement, then the dynamic programming algorithm only explores a pseudo-polynomial number of states between 0 and $\mathcal{O}(|N| \cdot |t|)$. This method is called ng-relaxation. Another known relaxed method is state relaxation where only the solutions with minimal cost are stored and extended at each state. Details refer to Baldacci et al. (2011); Righini and Salani (2008). Therefore, SPPRC is solvable by a route-relaxed method, which gives the lower bound to the solution. This will lead to the effective labeling algorithms we discuss in the following sections. □

We first review the basic concept of labeling algorithm (Irnich and Desaulniers (2005)). In this method, labels are generated and updated to represent partial paths that start at the depot 0. Starting from an initial label associated with node 0, paths are constructed iteratively by extending its label and its descendants forward/backward on graph G. These extensions of a label relay the updates of REFs along an arc. In each iteration, a new extended label is checked for feasibility with respect to the defined *critical resource* (Feillet et al. (2004)) constraints. If the label breaks any *critical resource*, it is discarded. To avoid enumerating all feasible partial paths, different dominance criteria are developed and applied to speed up the labeling algorithms. This concept is similar to pruning a tree while eliminating partial paths that cannot generate solutions better than the current bounded minimal reduced cost and pruning for further fathom. When labels are generated in one direction from the origin to the destination, a so-called mono-directional labeling search is performed.

Probably inspired by the method of *Divide and Conquer*, Righini and Salani (2008) designed a new bidirectional labeling algorithm, in which forward and backward labeling are performed simultaneously often yielding considerable computational time reduction. In this case, the *critical resources* have to be non-decreasing along the path. In other words, the triangle inequality must be satisfied. For MVRPC, this is indeed the fact, because time windows, truck capacities, layover time accumulations, and total required electricity are all suitable monotone resources. However, the intra-node distance constraint (18) does not satisfy the triangle inequality, i.e., even if $d_{ij} > d^{max}$ for the current partial path q with end node i, j could also be feasible for later extensions of the path. Moreover, a midpoint $M \in [M_0, M_1]$ is required to be specified. For instance, if we choose time windows as the resource, any $M \in [a_0, b_{0'}]$ can be a candidate midpoint. Bidirectional labeling algorithm works in three steps. First, labels are extended forward from the sink to the midpoint M; second, labels are extended from destination backward to M; third, two labels are merged and checked for feasibility.

We will discuss how to develop efficient mono-directional and bidirectional labeling algorithms for MVRPC in the following sections.

5.2. Mono-directional Labeling Algorithm

In this section, we describe the mono-directional labeling algorithm for the ESPPRC, where the reduced cost of the subproblem is defined by (43) with the three strengthened cuts. The labeling algorithm seeks to find a negative reduced-cost route that is feasible with respect to truck capacities, time windows and the current U.S. hours of service regulations.

The forward label search concept discussed here involves a relaxation technique, in which the state space explored by the exact dynamic programming method is projected onto a lower dimensional space, so that certain "non-critical" nodes are allowed to be visited several times. Different from normal VRPTW or EVRPTW, the pricing problem of MVRPC requires additional resources to keep track of the accumulated work time since the last layover and the remaining time to reach the next customer j. Goel and Irnich (2014) has illustrated that it is effective to use the minimal layover time when updating the time REFs and add necessary hours to each layover later on to avoid waiting time. We adopt the same idea and incorporate the possible recharging time for EV routes. In a forward mono-directional labeling algorithm for MVRPC, each label L corresponds to a partial path q starting at the depot 0 up to a node $i \in \bar{I}_0$. In the following descriptions of labeling algorithms, we will keep using $0'$ as a duplicate of the depot, which is the destination of each route. For a type-k truck, a label L_i is defined by the following attributes:

$L_i(cost)$	Reduced cost of partial path q.
$L_i(demand)$	Accumulated demand fulfilled by partial path q.
$L_i(recharge)$	Number of recharges performed along path q.
$L_i(time)$	Service start time at node $i \in \bar{I}_0$ when reached through partial path q.
$L_i(electric)$	Accumulated electricity consumption (in hours) from depot or a recharger up to node i along the partial path q. Note this resource is renewable.
$L_i(visit, n)$	A dummy resource variable that records the number of times any critical shipment $n \in S$ is visited along path q. The path is elementary only if $\forall L_i(visit, n) \leqslant 1$.
$L_i(elapsed)$	Accumulated elapsed time (hours) since the last layover (or depot if no layover).
$L_i(work)$	Accumulated working time (hours) since the last layover (or depot if no layover).
$L_i(layover)$	Number of layovers the driver has taken up to node i.

$L_i(latest)$	The latest possible time point to which the end of the last layover period can be extended to i without violating any resource constraint.
$L_i(nodes)$	A vector that stores the visited nodes along partial path q.

To further shrink the searching space, we introduce the idea of *unreachable nodes*, see Feillet et al. (2004). In the pricing problem of MVPRC, our considered critical resources obey the triangle inequality. Then, we can identify the nodes that cannot be visited by any extension of L_i because the time window and the capacity constraints are violated. These nodes are called *unreachable* by the label. If a label is assigned by an ICV, all of the available rechargers will not be considered in the search algorithm, and the corresponding REFs are set as 0. When a node is set as unreachable by a label, the dummy resource variable $L_i(visit, n) = 1$, i.e., as if the node n had been visited. Feillet et al. (2004) observed that setting $L_i(visit, n)$ to 1 will not change the structure of the label search but will reduce the computational time. Therefore, we define another attribute $\bar{L}_i(nodes)$ as the infeasible set that cannot be extended from the current state L_i. That is, $\bar{L}_i(nodes) = L_i(nodes) \cup \{j \in \bar{I}_0 \backslash L_i(nodes) : L_i(time) + \frac{d_{ij}}{g} + \lambda_i > b_j \vee L_i(demand) + \omega_j > Q_k \vee \forall j \in E, k \in V_D\}$. In addition, we introduce $Tabu_i$ as a candidate set of arcs that are feasible to be explored. Its definition will be introduced later in the section on implementation strategies.

In the initial state at node 0, all attributes of the label L are set to 0 except for $L_0(latest)$, which is equal to ∞. The extension of a label $L_i = (L_i(cost), L_i(demand), L_i(recharge), L_i(time), L_i(electric), L_i(visit, n), L_i(elapsed), L_i(work), L_i(layover), L_i(latest), L_i(nodes), \bar{L}_i(nodes))$ along an arc $(i, j) \in A$ is performed for each vehicle type k using the following REFs:

$$L_j(cost) = \begin{cases} L_i(cost) + \bar{c}_{ijk} - \tau - \pi, & \text{if } (i,j) \in (I\backslash S, S) \cap (I\backslash S', S'), \forall S, S' \subset I \\ L_i(cost) + \bar{c}_{ijk} - \tau, & \text{else if } (i,j) \in (I\backslash S, S), \forall S \subset I \\ L_i(cost) + \bar{c}_{ijk} - \pi, & \text{else if } (i,j) \in (I\backslash S', S') \\ L_i(cost) + \bar{c}_{ijk}, & \text{otherwise.} \end{cases} \tag{45}$$

$$L_j(demand) = L_i(demand) + \omega_j \tag{46}$$

$$L_j(recharge) = L_i(recharge) + \begin{cases} 1, & \text{if } j \in E \\ 0, & \text{otherwise} \end{cases} \tag{47}$$

While updating the time resource $L_i(time)$, we need to consider the possible recharging time and the viable layover time. We first define an alert indicator Δ_{ij} passing through the arc (i, j) to check whether or not the service hours regulation is broken, where $\Delta_{ij} = \min\{0, \xi - L_i(work) - (\frac{d_{ij}}{g} + \lambda_i)\}$. A direct extension from i to j can be performed only if $\Delta_{ij} = 0$, otherwise, a layover must be added since the service hours regulation is about to be violated. Considering the two situations, we define the REF for $L_i(time)$ as:

$$L_j(time) = \begin{cases} L_i^{adjust}(time) + L_i(electric) + \frac{d_{ij}}{g}, & \text{if } \Delta_{ij} = 0 \text{ and } i \in E, \\ L_i^{adjust}(time) + \frac{d_{ij}}{g} + \lambda_i, & \text{if } \Delta_{ij} = 0 \text{ and} \\ & \quad i \in I_0 \cup \{0'\}, \\ L_i^{adjust}(time) + L_i(electric) + \lceil \frac{|\Delta_{ij}|}{\xi} \rceil \cdot \beta^{lb} + \frac{d_{ij}}{g}, & \text{if } \Delta_{ij} < 0 \text{ and } i \in E, \\ L_i^{adjust}(time) + \lceil \frac{|\Delta_{ij}|}{\xi} \rceil \cdot \beta^{lb} + \frac{d_{ij}}{g} + \lambda_i, & \text{otherwise.} \end{cases} \tag{48}$$

$$L_j(electric) = \begin{cases} \alpha \cdot \delta_{ij}, & \text{if } i \in E_0, \\ L_i(electric) + \alpha \cdot \delta_{ij}, & \text{otherwise.} \end{cases} \tag{49}$$

Note that in REF (49), $L_i(electric) = \delta_{ij} = 0$ for all partial path q that travels by ICV. Next, we define a function named $unreachable_n(i)$ to measure the feasibility status of $\forall n \in S$ for the current label L_i. If the time window and capacity resources are not violated, then this function returns 1, otherwise, it returns 0.

$$L_j(visit, n) = \begin{cases} L_i(visit, n) + 1, & \text{if } j = n, \\ \max\{L_i(visit, n), unreachable_n(j)\}, & \text{otherwise.} \end{cases} \tag{50}$$

$$L_j(elapsed) = \begin{cases} L_i(elapsed) + L_j(electric) + \frac{d_{ij}}{g} + \lambda_i, & \text{if } \Delta_{ij} = 0 \text{ and } j \in E, \\ L_i(elapsed) + \frac{d_{ij}}{g} + \lambda_i, & \text{if } \Delta_{ij} = 0 \text{ and } j \in I_0 \cup \{0'\}, \\ L_i(work) + L_j(electric) + \frac{d_{ij}}{g} + \lambda_i & \\ -\lceil\frac{|\Delta_{ij}|}{\xi}\rceil \cdot \xi, & \text{if } \Delta_{ij} < 0 \text{ and } j \in E, \\ L_i(work) + \frac{d_{ij}}{g} + \lambda_i - \lceil\frac{|\Delta_{ij}|}{\xi}\rceil \cdot \xi, & \text{otherwise.} \end{cases} \tag{51}$$

$$L_j(work) = \begin{cases} L_i(work) + \frac{d_{ij}}{g} + \lambda_i, & \text{if } \Delta_{ij} = 0, \\ L_i(work) + \frac{d_{ij}}{g} + \lambda_i - \lceil\frac{|\Delta_{ij}|}{\xi}\rceil \cdot \xi, & \text{otherwise.} \end{cases} \tag{52}$$

$$L_j(layover) = \begin{cases} L_i(layover), & \text{if } \Delta_{ij} = 0, \\ L_i(layover) + \lceil\frac{|\Delta_{ij}|}{\xi}\rceil, & \text{otherwise.} \end{cases} \tag{53}$$

$$L_j(latest) = \begin{cases} \min\{L_i(latest), b_j - L_j(elapsed)\}, & \text{if } \Delta_{ij} = 0, \\ \min\{\infty, b_j - L_j(elapsed), L_j(time) - L_i(time) & \\ +L_i(latest) + L_i(elapsed) - L_j(elapsed) & \\ +(\beta^{ub} - \beta^{lb}) \cdot \lceil\frac{|\Delta_{ij}|}{\xi}\rceil\}, & \text{otherwise.} \end{cases} \tag{54}$$

$$L_j(nodes) = L_i(nodes) \cup \{j\}. \tag{55}$$

REF (53) explains how prolonged layover period(s) can be added to a partial path. For every non-depot start node on a partial path, we assume it is delivered at its beginning time window a_i. The layover periods are specified whenever needed with the minimum hours β^{lb} as shown in REFs (48) and (52). By this scheduling policy, we avoid times during which the driver is not productive. However, this may also cause early arrivals due to time window constraints at subsequent shipments and thus miss some valid routes. By lengthening the duration of the layover periods between $[\beta^{lb}, \beta^{ub}]$, we allow the possibility of measuring the before-missed routes. However, the arrival time at already routed subsequent shipments may be pushed out of their respective time windows. We therefore must calculate the resource values of $L_i(latest)$ indicating the maximum amount by which the duration of a layover can be extended without violating any resource constraints. REF (54) says that the latest end time $L_j(latest)$ of the last layover can be updated by minimizing values between the current latest time $L_i(latest)$ and the latest possible end time of the last layover without violating node j's time window. In some cases, $L_j(time)$ is allowed to be deferred, such that we use the term $[L_i(latest) + L_i(elapsed)]$ to represent the arrival time at i. Let the term $[L_j(time) - L_i(time)]$ be the work and potential layover period between i and j. As shown in (54), $L_j(elapsed)$ is the truncated working time left after an immediate layover. Then, the duration of the latest layover can be lengthened by any value less than or equal to $NewTime = L_j(latest) - [L_j(time) - L_i(time) + L_i(latest) + L_i(elapsed) - L_j(elapsed)]$. In other words, at each state, only if the following condition (56) is satisfied can we continue extending the label.

$$NewTime + \beta^{lb} \cdot \lceil\frac{|\Delta_{ij}|}{\xi}\rceil \leqslant \beta^{ub} \cdot \lceil\frac{|\Delta_{ij}|}{\xi}\rceil. \tag{56}$$

Because, if a layover is applied on the corresponding arc (i, j), the layover period can only be adjusted between $[\beta^{lb}, \beta^{ub}]$. For arcs (i, j) that do not take any layover either because the work time limit is never hit or because layovers have been taken in previous states, based on our policy, the current time can still be adjusted to satisfy the time window constraints for subsequent nodes. Therefore, whether or not there is a layover added on the arc, we need to check the following condition to ensure adjusting the time resource $L_j(time)$ is feasible.

$$L_j(latest) + L_j(elapsed) \geqslant a_j. \tag{57}$$

When checking whether or not a new label extension from i to j satisfies the time window resource constraint, we initially need to check if $L_j(time) \leqslant b_j$ is feasible. Next, suppose (56) and (57) are not violated under the defined situations, we should update the current route time $L_j(time)$ to ensure the earliest acceptable time a_j is respected.

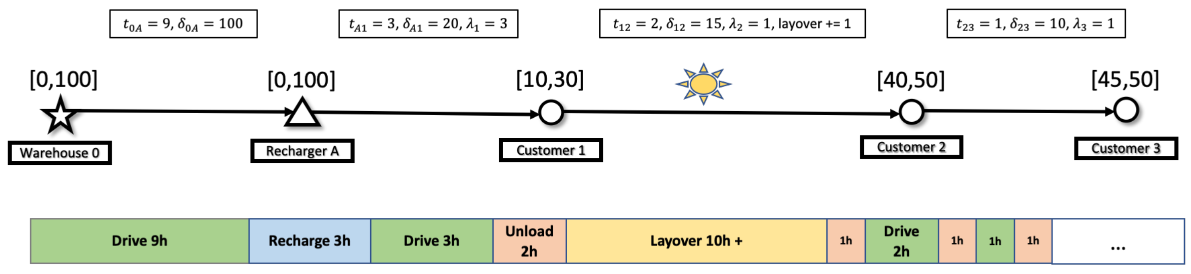
$$L_j^{adjust}(time) = \max\{L_j(time), a_j\}. \tag{58}$$

In each label extension, the REFs defined in (45)-(58) are updated by the following sequence.

$$\begin{aligned}
&L_j(demand) \Rightarrow L_j(time) \Rightarrow L_j(electric) \Rightarrow L_j(elapsed), L_j(work) \Rightarrow L_j(latest) \\
&\Rightarrow Feasible? \Rightarrow Yes! \\
&\Rightarrow L^{adjust}(time) \Rightarrow L_j(cost), L_j(recharge), L_j(visit, n), L_j(layover), L_j(nodes).
\end{aligned}$$

The label $L_j = (L_j(cost),\ L_j(demand),\ L_j(recharge),\ L_j^{adjust}(time),\ L_j(electric),\ L_j(visit, n),$ $L_j(elapsed),\ L_j(work),\ L_j(layover),\ L_j(latest),\ L_j(nodes),\ \bar{L}_j(nodes), k)$ resulting from this extension is deemed feasible if $L_j(demand) \leqslant Q_k$, $L_j(electric) \leqslant \alpha \cdot \varphi$, $a_j \leqslant L_j^{adjust}(time) \leqslant b_j$, and $L_j(visit, n) \leqslant 1$ for all $n \in S$. If L_j failed to pass any feasibility check, it is dropped.

Example 2. *Figure 1 illustrates an example of extending a partial path following the above defined REFs. For a fully rested driver at depot, we set $L_0(latest) = \infty$ and assume all the EVs are fully charged to a battery level of $\varphi = 200$. We also assume the accumulated demands of customer 1,2,3 fit the EV capacity Q_e. The time window for each node is given, e.g. $[a_1, b_1] = [10, 30]$. The travel time, electric consumption, and service time are stored in parameters t_{ij}, δ_{ij}, and λ_j, respectively. The "sun" on the arc from customer 1 to customer 2 means there is a layover required. Given the charging rate $\alpha = 0.03$, maximum-allowed working time before a layover $\xi = 14$, and layover period $[10, 14]$, the forward path generated by updating the corresponding time-related REFs in the network is as follows:*



$t_{0A} = 9, \delta_{0A} = 100$
$t_{A1} = 3, \delta_{A1} = 20, \lambda_1 = 3$
$t_{12} = 2, \delta_{12} = 15, \lambda_2 = 1$, layover += 1
$t_{23} = 1, \delta_{23} = 10, \lambda_3 = 1$
[0,100]
[0,100]
[10,30]
[40,50]
[45,50]
Warehouse 0
Recharger A
Customer 1
Customer 2
Customer 3
Drive 9h
Recharge 3h
Drive 3h
Unload 2h
Layover 10h +
1h
Drive 2h
1h
1h
1h
...

Figure 1. A feasible EV schedule in which additional layover period is required corresponding to a partial path $0 - A - 1 - 2 - 3$.

$$☆\ Start \Rightarrow \begin{bmatrix} L_0(time) = 0 \\ L_0(electric) = 0 \\ L_0(elapsed) = 0 \\ L_0(work) = 0 \\ L_0(latest) = \infty \\ L_0^{adjust}(time) = 0 \end{bmatrix} \Rightarrow \begin{bmatrix} L_A(time) = 9 \\ L_A(electric) = 3 \\ L_A(elapsed) = 9 \\ L_A(work) = 9 \\ L_A(latest) = 91 \\ L_A^{adjust}(time) = 9 \end{bmatrix} \Rightarrow \begin{bmatrix} L_1(time) = 15 \\ L_1(electric) = 0.6 \\ L_1(elapsed) = 15 \\ L_1(work) = 12 \\ L_1(latest) = 15 \\ L_1^{adjust}(time) = 15 \end{bmatrix} \Rightarrow \begin{bmatrix} L_2(time) = 30 \\ L_2(electric) = 1.05 \\ L_2(elapsed) = 3 \\ L_2(work) = 3 \\ L_2(latest) = 46 \\ L_2^{adjust}(time) = 40 \end{bmatrix} \Rightarrow \begin{bmatrix} L_3(time) = 42 \\ L_3(electric) = 1.35 \\ L_3(elapsed) = 5 \\ L_3(work) = 5 \\ L_3(latest) = 45 \\ L_3^{adjust}(time) = 45 \end{bmatrix}$$

The forward propagation starts from the depot 0*, then through recharger* A *and customer* 1*. No time adjustment is performed on these two states due to* $\Delta_{ij} = 0$ *and both time windows are fitted. When reaching customer* 2*, a layover is enforced because the accumulated work time* $15 > \xi = 14$*. In addition, the current time* $L_2(time)$ *needs to be adjusted since otherwise the path would have arrived before* a_2 *otherwise. Similarly,* $L_3(time)$ *is adjusted. The numbers are calculated recursively by updating* $L_j(latest)$ *and* $L_j^{adjust}(time)$*.*

Note that all REFs are non-decreasing functions. Hence, a standard dominance rule can be applied here, see Desaulniers et al. (1998). Another useful dominance rule arising by introduction of the layover constraint can be derived from the results of Tilk (2016).

Proposition 5.1 (Rule 1.)**.** *Define two labels* L^1 *and* L^2 *associated with the same node* i*. Then, label* L_i^2 *is dominated by label* L_i^1 *only if all the following conditions are satisfied and at least one of the inequalities is strict. Therefore,* L_i^2 *can be discarded.*

$$\begin{aligned}
\forall i \in \bar{I}_0 \cup \{0'\} \qquad & L_i^1(nodes) \subseteq L_i^2(nodes), L_i^1(cost) \leqslant L_i^2(cost), \\
& L_i^1(demand) \leqslant L_i^2(demand), L_i^{1,adjust}(time) \leqslant L_i^{2,adjust}(time), \\
& L_i^1(work) \leqslant L_i^2(work), L_i^1(electric) \leqslant L_i^2(electric), \\
& L_i^1(recharge) \leqslant L_i^2(recharge), L_i^1(visit,n) \leqslant L_i^2(visit,n), \\
& L_i^1(layover) \leqslant L_i^2(layover), L_i^1(latest) \geqslant L_i^2(latest).
\end{aligned}$$

Proposition 5.2 (Rule 2.)**.** *Define two labels* L^1 *and* L^2 *associated with the same node* i*. Then, label* L_i^2 *is dominated by label* L_i^1 *only if all the following conditions are satisfied and at least one of the inequalities is strict. Therefore,* L_i^2 *can be discarded.*

$$\begin{aligned}
\forall i \in \bar{I}_0 \cup \{0'\} \qquad & L_i^1(nodes) \subseteq L_i^2(nodes), L_i^1(cost) \leqslant L_i^2(cost), \\
& L_i^1(demand) \leqslant L_i^2(demand), L_i^{1,adjust}(time) + \xi \leqslant L_i^{2,adjust}(time), \\
& L_i^1(electric) \leqslant L_i^2(electric), L_i^1(recharge) \leqslant L_i^2(recharge), \\
& L_i^1(visit,n) \leqslant L_i^2(visit,n), L_i^1(layover) \leqslant L_i^2(layover).
\end{aligned}$$

Recall that the resource $L_i(visit, n)$ only stores the visiting information for “critical" shipments, i.e., the shipments visited more than once are marked as critical and the labeling algorithm repeats until all the optimal routes are elementary. The so-called decremental state-space relaxation (DSSR) was initially proposed by Righini and Salani (2008). Let Θ be the set of critical shipments found in the current iteration. Let Γ be the set of critical shipments in the optimal solutions discovered by the DSSR labeling algorithm, where non-elementarity is allowed. In each iteration, we update the set Θ by $\Theta = \Gamma \cup \Theta$. This iteration procedure is repeated until the sets Γ becomes empty, and we reach an optimal solution for the ESPPRC. Algorithm 1 shows a generic version of this forward labeling algorithm for the pricing problem of MVRPC. $List(\mathbf{L_i})$ is the list of forward states associated with node i. $Forward$ is the set of nodes to be checked and extended. And S_Θ is the set of critical shipments found up to the current iteration plus the unreachable nodes for each. The function $Duplicate()$ returns the set of nodes that cause the cycles in the current n optimal paths. The parameter n is set to 3 in our study.

Algorithm 1: Bounded forward search with decremental state-space relaxation

```
Initialization: $\Gamma = \Theta = \emptyset$; truck type $k$;
while $\Gamma \neq \emptyset$ do
    $\Theta = \Theta \cup \Gamma$;
    $List(\mathbf{L_0}) \leftarrow \{(L_0, 0)\}$; $Forward \leftarrow \{0\}$;
    for all $i \in \bar{I}$ do $List(\mathbf{L_i}) \leftarrow \emptyset$;
    while $Forward \neq \emptyset$ do
        pick $i \in Forward$;
        for $L_i \in List(\mathbf{L_i})$, s.t. $L_i(visit, n) = L_i(visit, n)_{n \in S_\Theta}$ do
            for $j \notin Tabu_i \cup \{0'\}$ and $(L_i(visit, j)_{j \in S_\Theta} = 0$ or $j \notin \Theta)$ do
                $L'_j \leftarrow$ extend REFs for $i$ to $(L'_j(cost), L'_j(demand), L'_j(recharge), L'^{adjust}_j(time),$
                  $L'_j(electric), L'_j(visit, n), L'_j(work), L'_j(layover), L'_j(latest), L'_j(nodes)),$
                  $\bar{L}'_j(nodes), k)$;
                check Rule 1. and Rule 2.;
                if $L'_j$ is not dominated by any label in $List(\mathbf{L_j})$ then $List(\mathbf{L_j}) \leftarrow List(\mathbf{L_j}) \cup \{L'_j\}$;
                $Forward \leftarrow Forward \cup \{j\}$;
            end
        end
        $Forward \leftarrow Forward \backslash \{i\}$;
    end
    /* extend the labels back to the depot */
    for all $j \in \bar{I}$ for all $L_j \in List(\mathbf{L_j})$ do extend REFs from $j$ to $0'$;
    $\Gamma \leftarrow Duplicate(\mathbf{L^{opt}_{0'}})$.
end
```

5.3. Bidirectional Labeling Algorithm

Bidirectional labeling algorithms are widely used in solving the pricing problems of VRP variants. They are especially beneficial in mitigating the explosion of labels, see Tilk et al. (2017). It has been observed in Righini and Salani (2006) and many more recent works that the bidirectional labeling algorithms are usually superior to their mono-directional counterparts and lead to substantially improved computational times for solving related SPPRC. When running a bidirectional algorithm, both forward partial paths and backward partial paths are created and extended alternatively. Each direction is processed up to a so-called “halfway point" (Righini and Salani (2006)). The halfway point is the midpoint of the domain of a monotone critical resource, e.g. $L_i(time)$ for our case. Recall that along a partial path the time-related and load-related resources are all monotone. Along a forward path, the arrival times and the accumulated demands are non-decreasing. This standard bidirectional search mechanism with fixed halfway point has been adopted as almost a quasi-standard for solving ESPPRC. However, when it comes to renewable resources, such as EV recharging, Tilk et al. (2017) observed the number of forward and backward labels can differ significantly due to the fixed halfway point setting. Since MVRPC involves two renewable resources, i.e., EV recharging and replenishment layovers, we decide to adopt a dynamic halfway point scheme as in Tilk et al. (2017) to combat the potential unbalanced extensions. In our work, we choose to use time windows as the critical resource because it considers the layover resources and is more restrictive than the demand resource. In fact, the more restrictive the monotone resource is, the more effective is bidirectional labeling.

It is worth noting that the relaxed bounded bidirectional search with forward and backward propagations may result in either feasible or infeasible complete paths because of cycles. As can be seen from Figure 2, both forward and backward extensions can result in cyclic partial paths (top); even if the two partial paths are elementary, because certain shipments may occur in both paths, a cycle can be produced by joining them (e.g., shipment k in the bottom panel). This issue can be addressed by a modified DSSR feasible check, which will be discussed later.

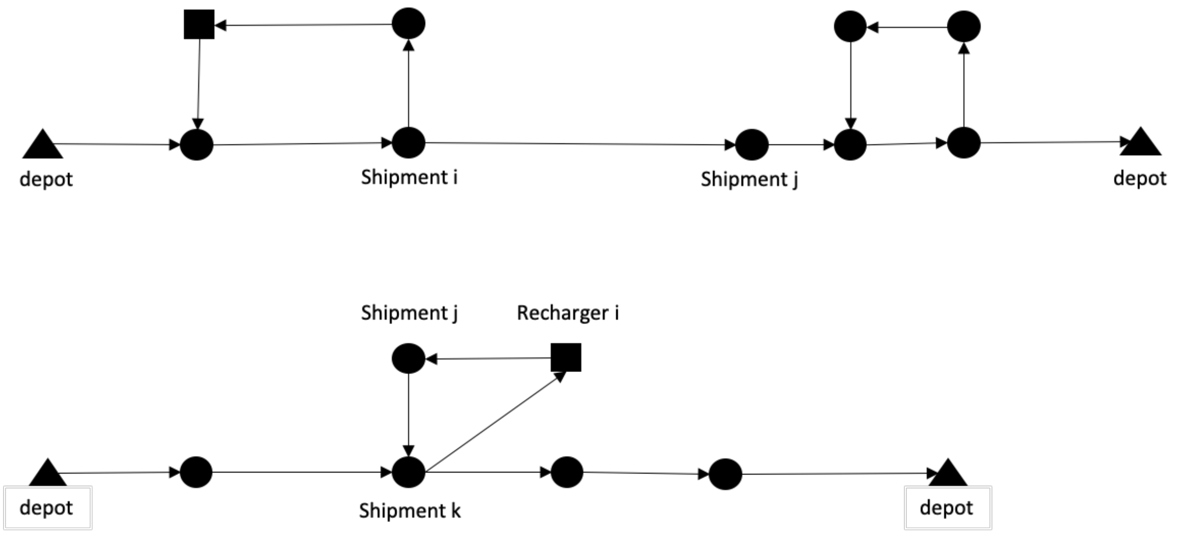


Figure 2. Top: path from 0 to $0'$ merged by two non-elementary paths $0-i$ and $j-0'$. Bottom: two elementary paths $0-i$ and $j-0'$ are joined to a complete path with a cycle.

5.3.1. Dynamic Halfway Points

We start with the definitions of necessary components for the bidirectional search. The upper bound of the halfway point H_F of the time resource is initialized as $(\max_{i\in I}\{b_i\} + \max_{i\in \bar{I}}\{\frac{d_{i0}}{g}\} + \max_{i\in I}\{\lambda_i\})$. H_F is the halfway point for the forward search, which indicates half of the latest possible return time to the depot. Then, the forward search proceeds as in Algorithm 1 only if its $L_i^{adjust}(time)$ does not pass the forward halfway point H_F. As for the backward extension, the search direction is against the travel direction. The extension of the critical resource will be proceeded towards the lower bound of the halfway point H_B, which is initialized as $(\min_{i\in I}\{a_i\} + \min_{i\in \bar{I}}\{\frac{d_{i0}}{g}\} + \min_{i\in I}\{\lambda_i\})$. Let $B_i^{adjust}(time)$ be the adjusted backward time resource. Every time a forward label is extended, the backward halfway point H_B is updated by the following rule to $H_B = \min\{L_i^{adjust}(time), H_F\}$. Alternatively, when a backward label is extended, the forward halfway point is updated to $H_F = \max\{B_i^{adjust}(time), H_B\}$. Note that the forward halfway point remains non-decreasing whereas the backward halfway point remains non-increasing. The two alternative updating steps ensure that $H_F \geqslant H_B$, which guarantees the optimality. In addition, the initial values of H_F and H_B restrict the evolution of the dynamic halfway point to lie in the interval $[H_F, H_B]$. If we initialize $H_F = H_B$, then this becomes the standard fixed bidirectional labeling algorithm.

5.3.2. Backward Resource Extension Functions

Although the REFs for backward search are defined similar to the forward search, some adaptions need to be made. Consider a backward path traveled by an EV, $q = (i, ..., e_q..., 0')$, that requires a visit to the recharger e_q. Then, e_q is named the “next recharger" of q, which implies there is no additional recharging visit between node i and e_q. The amount of electricity consumed from i to e_q varies while we extend the label backwards. We need an additional resource $B_i(slack)$ for the sake of merging forward and backward labels. This resource is the maximum available recharging time for e_q, assuming that all time windows along path $(j, 0')$ are respected. Moreover, whether the working hours hit the limit is unknown as well because it depends on the schedules from 0 to i, which are performed by the forward search. For a type-k backward path $q = (i, ..., 0')$, the label B_i is defined by the following attributes: $B_i = (B_i(cost), B_i(demand), B_i(recharge), B_i^{adjust}(time), B_i(electric), B_i(visit, n), B_i(elapsed),$

$B_i(work)$, $B_i(layover)$, $B_i(latest)$, $B_i(slack)$, $B_i(nodes))$, $\bar{B}_i(nodes), k)$. In order to use the similar REFs definitions as in forward labeling procedure, we need the following adaptions.

$B_i(cost)$	Reduced cost of partial path q.
$B_i(demand)$	Residual space on the truck, which equals to the accumulated demand from $0'$ except at the last node i.
$B_i(recharge)$	Negative number of recharges along path q except at node i.
$B_i(time)$	Latest arrival time at $i \in \bar{I}_0$ without breaking any time window.
$B_i(electric)$	Accumulated electricity consumption from $0'$ or a recharger back to node i along the path q. Note this resource is renewable.
$B_i(visit, n)$	A dummy resource variable that records the number of times any critical shipment $n \in S$ is visited along path q except node i. Let the value be 1 if unreachable, which indicates $B_i(demand) + \omega_i + \omega_n > Q_k$ or $B_i(time) - \frac{d_{ni}}{g} - \lambda_n$.
$B_i(elapsed)$	Accumulated elapsed time (hours) from the next layover ($0'$ if no layover) to i.
$B_i(work)$	Accumulated working time (hours) from the next layover ($0'$ if no layover) to i.
$B_i(layover)$	Negative number of layovers the driver has taken from $0'$ to i.
$B_i(latest)$	The latest possible time to which the end of last layover period can be extended to j without violating any resource constraint.
$B_i(slack)$	Maximum flexible period that can be used for the next recharge that satisfies the time windows while running EV on the path.
$B_i(nodes)$	A vector that stores the visited nodes along backward path q.

The backward labeling procedure starts from node $0'$ with the initial values assigned to each resource of $B_{0'}$, where $B_{0'}(time) = B_{0'}^{adjust}(time) = b_{0'}$, $B_{0'}(latest) = a_{0'}$, $B_{0'}(slack) = \alpha \cdot \varphi$ (and $B_{0'}(slack) = 0$ if $k \in V_D$), and the other resources are all set to 0. REFs for $B_i(cost)$, $B_i(demand)$, $B_i(visit, n)$, $B_i(electric)$, $B_i(nodes)$ are defined the same as $L_i(cost)$, $L_i(demand)$, $L_i(visit, n)$, $L_i(electric)$, $L_i(nodes)$. If the backward path is assigned by an ICV, then $B_i(electric) = \delta_{ij} = 0, \forall (i, j) \in A$. Again, nodes are *unreachable* to the current label if and only if the time window and/or the truck capacity are violated. Δ_{ij} is also the same as before except that we use $B_i(work)$ to substitute $L_i(work)$ in the forward labeling procedure. Other resources of a label B_i along a backward arc $(i, j) \in A$ are extended towards H_B by the following REFs. Moreover, we introduce the feasible arcs list $Tabu_i^{-1}$ on the inverted network, follow the same logic as forward search. Details are elaborated in section 5.4.1. Note that the extension will be proceeded from $0'$ backwards before $(B_i^{adjust}(time) - \frac{d_{ij}}{g})$ hitting the lower bound of the halfway point H_B.

$$B_j(time) = \begin{cases} B_i^{adjust}(time) - \frac{d_{ij}}{g} - \alpha \cdot \delta_{ij}, & \text{if } \Delta_{ij} = 0 \text{ and } i \in E, \\ B_i^{adjust}(time) - \frac{d_{ij}}{g} - \lambda_i, & \text{if } \Delta_{ij} = 0 \text{ and } i \in I \cup \{0'\}, \\ B_i^{adjust}(time) - \alpha\delta_{ij} - \lceil \frac{|\Delta_{ij}|}{\xi} \rceil \cdot \beta^{lb} - \frac{d_{ij}}{g}, & \text{if } \Delta_{ij} < 0 \text{ and } i \in E, \\ B_i^{adjust}(time) - \lceil \frac{|\Delta_{ij}|}{\xi} \rceil \cdot \beta^{lb} - \frac{d_{ij}}{g} - \lambda_i, & \text{otherwise.} \end{cases} \tag{59}$$

$$B_j(recharge) = B_i(recharge) - \begin{cases} 1, & \text{if } i \in E, \\ 0, & \text{otherwise.} \end{cases} \tag{60}$$

$$B_j(slack) = \begin{cases} B_{0'}(slack), & \text{if } B_i(recharge) = 0 \text{ and } i \notin E, \\ \min\{B_{0'}(slack) - \alpha \cdot \delta_{ij}, \\ B_j^{adjust}(time) - a_j - \lambda_j\}, & \text{if } i \in E, \\ \min\{B_i(slack) - \alpha \cdot \delta_{ij}, \\ B_j^{adjust}(time) - a_j - \lambda_j\}, & \text{otherwise.} \end{cases} \tag{61}$$

$$B_j(layover) = B_i(layover) - \begin{cases} 0, & \text{if } \Delta_{ij} = 0, \\ \lceil \frac{|\Delta_{ij}|}{\xi} \rceil, & \text{otherwise.} \end{cases} \tag{62}$$

$$B_j(elapsed) = \begin{cases} B_i(elapsed) + \alpha \cdot \delta_{ij} + \frac{d_{ij}}{g}, & \text{if } \Delta_{ij} = 0 \text{ and } i \in E, \\ B_i(elapsed) + \frac{d_{ij}}{g} + \lambda_i, & \text{if } \Delta_{ij} = 0 \text{ and } i \in I \cup \{0'\}, \\ B_i(work) + \frac{d_{ij}}{g} - \lceil \frac{|\Delta_{ij}|}{\xi} \rceil \cdot \xi, & \text{if } \Delta_{ij} < 0 \text{ and } i \in E, \\ B_i(work) + \frac{d_{ij}}{g} + \lambda_i - \lceil \frac{|\Delta_{ij}|}{\xi} \rceil \cdot \xi, & \text{otherwise.} \end{cases} \tag{63}$$

$$B_j(work) = \begin{cases} B_i(work) + \frac{d_{ij}}{g} + \lambda_i, & \text{if } \Delta_{ij} = 0, \\ B_i(work) + \frac{d_{ij}}{g} + \lambda_i - \lceil \frac{|\Delta_{ij}|}{\xi} \rceil \cdot \xi, & \text{otherwise.} \end{cases} \tag{64}$$

$$B_j(latest) = \begin{cases} \max\{B_i(latest), B_j(elapsed) + a_j + \lambda_j\}, & \text{if } \Delta_{ij} = 0, \\ \max\{B_j(elapsed) + a_j + \lambda_j, B_i(latest) - B_i(elapsed) \\ -[B_i(time) - B_j(time)] + B_j(elapsed) \\ -(\beta^{ub} - \beta^{lb}) \cdot \lceil \frac{|\Delta_{ij}|}{\xi} \rceil\}, & \text{otherwise.} \end{cases} \tag{65}$$

REFs (59), (63)-(65) are similar to their forward counterparts, yet are propagated in an inverted direction. The unloading times at the current j are not taken into account in computing the resources. Furthermore, when the recharging activity happens right before a layover (in the reversed chronological order), the recharging time $\alpha \cdot \delta_{ij}$ at recharger i should not be considered in the calculations of $B_j(elapsed)$. The reason is that this activity happened before the layover period and the truncated elapsed time should only track the driving and unloading time afterwards. When checking the time resource feasibility, each time window at shipment i will be shifted by λ_i to $[a_i + \lambda_i, b_i + \lambda_i]$. Since the time windows for the rechargers are relaxed, a labeling extension to rechargers always satisfies the time window constraint. Moreover, even without any layover associated with a backward arc (i, j), the current time $B_j(time)$, which may be greater than b_j, can be adjusted to fit into the time window if and only if $B_j(latest) - B_j(elapsed) \leqslant b_j + \lambda_j$. Therefore, the updated current time for the backward path is:

$$B_j^{adjust}(time) = \min\{B_j(time), b_j + \lambda_j\}. \tag{66}$$

The sequence of extending and checking the backward REFs are the same as discussed for the forward labeling procedure. The resulting label $B_i = (B_i(cost), B_i(demand), B_i(recharge), B_i^{adjust}(time),$ $B_i(electric), B_i(visit, n), B_i(elapsed),$
$B_i(work), B_i(layover), B_i(latest), B_i(slack), B_i(nodes)), \bar{B}_i(nodes), k)$ is deemed feasible if $B_i(demand) \leqslant Q_k$, $a_i \leqslant B_i^{adjust}(time) - \lambda_i \leqslant b_i$, $B_i(slack) \geqslant 0$, $B_i(electric) \leqslant \alpha \cdot \varphi$, and $B_i(visit, n) \leqslant 1$ for all $n \in S$. If B_i failed to pass any feasibility check, it is dropped. Now let us consider another straightforward example which shows the consistency and difference between forward and backward labeling.

Example 3. *Figure 3 shows another complete EV route with a single visit to rechargers. The route is built by the backward extension. Other configurations are the same as Example 2 except for the parameter values. In this example, the route generated from the backward propagation can be exactly reproduced by the forward extension. However, the detailed schedules (arrival time and layovers) may or may not be different. This allows generation* *of diverse columns while keeping the number of labels lower than mono-directional searches. The aroused issues such as infeasible routes after merging the partial paths can be offset by the framework of dynamic halfway points.*

The backward extension starts from the copy of depot $0'$*, then the extension follows the above-defined REFs and the depicted time windows, travel time, power consumption, and unloading time. After the first two extensions, this label arrives at node 1. The resource* $B_1(time)$ *needs to be adjusted to fit in the time window. Also, the resource* $B_1(slack)$ *is updated reflecting a minimum recharging of the power consumed on arc* $(1, A)$*.* $B_1(slack)$ *indicates how much additional electricity can be assumed on the paths ahead, which have not yet been determined. The extension along arc* $(1, A)$ *yields* $B_1(time) = 76 - 3 - 0.9 - 10 = 62.1$, $B_1(electric) = 0.03 \times 30 = 0.9$, $B_1(elapsed) = 12 + 3 - 14 = 1 = B_1(work)$, $B_1(slack) = 6 - 0.9 = 5.1$, $B_1(latest) = \max\{30 + 1 + 4, 58 - 12 - 13.9 + 1 - 4\} = 35$*. This example illustrates how recharging time should be eliminated from the calculation* $B_i(elapsed)$*. In fact, recharging is not taken into account for the hours of service regulation and is only used for tracking the current time. Indeed, even if the recharge at A was additionally required for up to* $B_1(slack) = 5.1$ *units of time due to other possible extensions,* $B_1(time)$ *would be set at* $62.1 - 5.1 = 57$*. The adjusted time would still fit into the time window, which is set at* $B^{adjust} = \min\{57, 54\} = 54$.

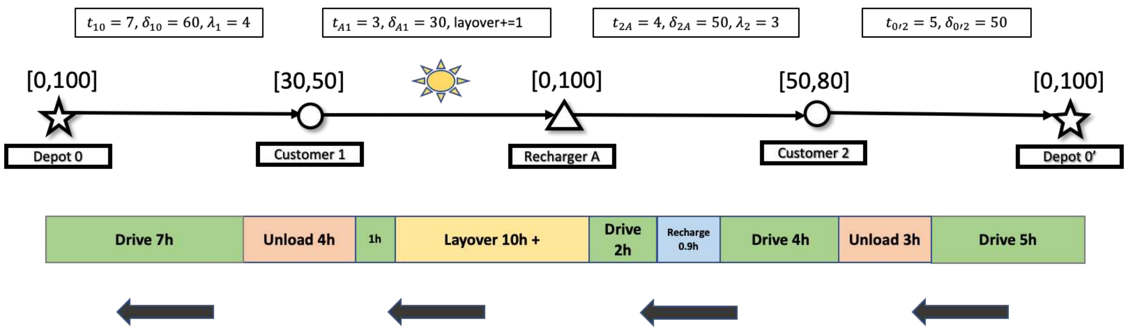

$t_{10} = 7, \delta_{10} = 60, \lambda_1 = 4$
$t_{A1} = 3, \delta_{A1} = 30$, layover+=1
$t_{2A} = 4, \delta_{2A} = 50, \lambda_2 = 3$
$t_{0'2} = 5, \delta_{0'2} = 50$
[0,100]
[30,50]
[0,100]
[50,80]
[0,100]
Depot 0
Customer 1
Recharger A
Customer 2
Depot 0'
Drive 7h
Unload 4h
1h
Layover 10h +
Drive 2h
Recharge 0.9h
Drive 4h
Unload 3h
Drive 5h

Figure 3. A complete backward label extended on an EV path $0' - 2 - A - 1 - 0$.

$$\begin{bmatrix} B_0(time) = 43 \\ B_0(electric) = 2.7 \\ B_0(elapsed) = 12 \\ B_0(work) = 12 \\ B_0(slack) = 3.3 \\ B_0(latest) = 35 \\ B_0^{adjust}(time) = 43 \end{bmatrix} \Leftarrow \begin{bmatrix} B_1(time) = 62.1 \\ B_1(electric) = 0.9 \\ B_1(elapsed) = 1 \\ B_1(work) = 1 \\ B_1(slack) = 5.1 \\ B_1(latest) = 35 \\ B_1^{adjust}(time) = 54 \end{bmatrix} \Leftarrow \begin{bmatrix} B_A(time) = 76 \\ B_A(electric) = 3 \\ B_A(elapsed) = 12 \\ B_A(work) = 12 \\ B_A(slack) = 6 \\ B_A(latest) = 58 \\ B_A^{adjust}(time) = 76 \end{bmatrix} \Leftarrow \begin{bmatrix} B_2(time) = 95 \\ B_2(electric) = 1.5 \\ B_2(elapsed) = 5 \\ B_2(work) = 5 \\ B_2(slack) = 6 \\ B_2(latest) = 58 \\ B_2^{adjust}(time) = 83 \end{bmatrix} \Leftarrow \begin{bmatrix} B_{0'}(time) = 100 \\ B_{0'}(electric) = 0 \\ B_{0'}(elapsed) = 0 \\ B_{0'}(work) = 0 \\ B_{0'}(slack) = 6 \\ B_{0'}(latest) = 0 \\ B_{0'}^{adjust}(time) = 100 \end{bmatrix} \Leftarrow \star\ \textit{Start}: 0'$$

To avoid explosion of backward labels, the two dominance rules introduced in Rule 1. and Rule 2. are still valid for the backward counterparts. The only difference is that certain non-decreasing monotone resources become non-increasing. To apply the two dominance rules, apart from changing all resources from $L_i's$ to $B_i's$, we need the following adaptions. First, the signs of the time resource ($B_i^{adjust}(time)$), the number of recharges resource ($B_i(recharge)$), and the layover resource ($B_i(layover)$) are switched from $\leqslant$ to $\geqslant$. Also, the sign of $B_i(latest)$ is reversed from $\geqslant$ in forward search to $\leqslant$. Second, according to Desaulniers et al. (2016), all strictly negative values of $B_i(recharge)$ are equally good. Therefore, REF (60) can be strengthened to $B_j(recharge) = \max\{-1, B_i(recharge) - 1\}$, if $i \in E$. Similarly, REF (62) for the layover resource can be strengthened in the same way to $B_j(layover) = \max\{-\lceil\frac{|\Delta_{ij}|}{\xi}\rceil, B_i(layover) - \lceil\frac{|\Delta_{ij}|}{\xi}\rceil\}$, if $\Delta_{ij} < 0$. Finally, the dominance criteria for the backward resource $B_i(slack)$ is added, i.e., $B_i^1(slack) \geqslant B_i^2(slack)$ for label B_i^2 dominated by label B_i^1.

5.3.3. Merge Forward and Backward Paths

In bidirectional labeling algorithms, labels are propagated to the halfway point. If either bound of the dynamic halfway point stays at the initial position, then a mono-directional labeling search is performed. Sometimes, it is also beneficial to make an adjustment such that more columns with negative reduced cost can be priced, although in the most cases halfway points are updated towards

doi:10.20944/preprints202307.1240.v1

the middle of the critical resource. Once the forward labeling and the backward labeling extensions are completed, forward and backward labels are joined together at the halfway point to form complete paths. Suppose there are two labels L_i and B_i representing a forward and a backward path, respectively, ending at node i. Merging these two labels yields a complete origin to destination path with reduced cost $L_i(cost) + B_i(cost)$. The path is feasible if and only if the following conditions are fulfilled:

1. $L_i(visit, n) + B_i(visit, n) \leqslant 1, \forall n \in S_\Theta$, where S_Θ is the set of critical shipments that maintain elementarity.
2. The path respects the vehicle capacity: $L_i(demand) + B_i(demand) \leqslant Q_k + \omega_i, \forall k \in V$.
3. The path is feasible with respect to the time resource. The feasibility should be considered by two distinct cases.

   Case 1. The backward path from $i+1$ to $0'$ visits at least 1 recharger, i.e., $B_i(recharge) < 0$,
   $L_i^{adjust}(time) + L_i(electric) + \lambda_i \leqslant B_i^{adjust}(time)$,
   $L_i(electric) + B_i(electric) \leqslant \alpha \cdot \varphi$, and $L_i(electric) \leqslant B_i(slack), L_i(latest) + L_i(elapsed) \geqslant B_i^{adjust}(time) - \lambda_i$.
   Case 2. The backward path from $i+1$ to $0'$ visits no recharger, i.e.,
   $B_i(recharge) = 0$, $L_i^{adjust}(time) \leqslant B_i^{adjust}(time) - \lambda_i$,
   $L_i(electric) + B_i(electric) \leqslant \alpha \cdot \varphi$, and
   $L_i(latest) + L_i(elapsed) \geqslant B_i^{adjust}(time) - \lambda_i$.
4. The accumulated working time does not break the hours of service regulation: $L_i(work) + B_i(work) \leqslant \xi$.

After the merge, we perform a final dominance test with *Rule 1.* and *Rule 2.* for all merged routes. The non-dominated routes with negative reduced cost are candidate columns for the RMP.

5.3.4. Generic Structure of the Algorithm

The bidirectional labeling algorithm updates the two bounds of the halfway points towards each other. Apart from the extension mechanism introduced in the previous sections, it is important to determine criteria for how to alternate between forward and backward labeling so that the overall performance of the algorithm can be guaranteed. An ideal strategy is to choose the direction which may minimize the overall computational time. Tilk et al. (2017) suggest that a heuristic be added by choosing the direction with smaller number of already generated labels. This heuristic is also added in a function of our algorithm, which is named $Direction()$. $Direction() = 1$ if it chooses to do a forward extension. Otherwise, $Direction() = 0$. However, purely relying on this simple heuristic may cause too many infeasible merges. To overcome the issue, we collect L_i and B_i that are produced in the current iteration, such that no merge is possible for those labels, and store them into a list Ξ. After finishing the procedures of merge and feasibility check, the algorithm will extend the labels from Ξ in a monotone direction until they approach the depot (0 for backward and $0'$ for forward). In particular, if infeasibility has arisen due to the violation of service hour regulation, we continue forward extension from this state in the following iterations. For simplicity, we denote the operations for the forward REFs (45)-(58) as $f_j(L_i)$ and the corresponding operations for the backward REFs as $g_j(B_i)$. Let $feasible(L_i, B_i)$ equal 1 if the merge of a forward and a backward label at node i passes all the conditions, and equals 0 otherwise. Once a complete route is merged, we store the feasible route q in a set F. The selected routes in F with negative reduced cost will finally be passed to Ω in the RMP. With these additional notations, the bounded bidirectional labeling framework is given in Algorithm 2. The forward and backward functions are shown in Algorithm 3.

Algorithm 2: Bounded bidirectional labeling algorithm

```
Initialization: $\Gamma = \Theta = \varnothing$; truck type $k$;
while $\Gamma \neq \varnothing$ do
    $\Theta = \Theta \cup \Gamma$; $\Xi = \varnothing$; $F = \varnothing$;
    $List(\mathbf{L_0}) \leftarrow \{(L_0, S_\Theta = \mathbf{0})\}$; $List(\mathbf{B_{0'}}) \leftarrow \{(B_{0'}, S_\Theta = \mathbf{0})\}$;
    $Forward \leftarrow \{0\}$; for all $i \in \bar{I}$ do $List(\mathbf{L_i}) \leftarrow \varnothing$;
    $Backward \leftarrow \{0'\}$; for all $i \in \bar{I}$ do $List(\mathbf{B_i}) \leftarrow \varnothing$;
    while $Forward \cup Backward \neq \varnothing$ do
        if $Direction() = 1$ then
            pick $i \in Forward$;                    /* perform forward extension */
            for $L_i \in List(\mathbf{L_i})$ do
                if $L_i^{adjust}(time) \leqslant H_F$ then
                    do forward search as in Algorithm 3;
            $Forward \leftarrow Forward \backslash \{i\}$.
        else
            pick $i \in Backward$;                   /* perform backward extension */
            for $B_i \in List(\mathbf{B_i})$ do
                if $B_i^{adjust}(time) > \frac{d_{ij}}{g} + H_B$ then
                    do backward search as in Algorithm 3
    /* merge forward and backward paths */
    for $\forall i \in \bar{I}$ do
        for $L_i \in List(\mathbf{L_i})$ do
            for $B_i \in List(\mathbf{B_i})$ do
                if $feasible(L_i, B_i)$ then
                    save route $q$ to $F$;
    /* find nodes that cause cycles in the $n$ optimal routes */
    $\Gamma \leftarrow Duplicate(F)$.
```

Algorithm 3: Forward and backward search steps

```
Forward search:
for $j \notin Tabu_i$ and $(L_i(visit, j)_{j \in S_\Theta} = 0$ or $j \notin \Theta)$ do
    $L'_j \leftarrow f_j(L_i)$; check Rule 1. and Rule 2.;
    if $L'_j$ is not dominated by any label in $List(\mathbf{L_j})$ then
        $List(\mathbf{L_j}) \leftarrow List(\mathbf{L_j}) \cup \{L'_j\}$;
        $Forward \leftarrow Forward \cup \{j\}$; $H_B = \min\{L'^{adjust}_j(time), H_F\}$;
    end
end
Backward search:
for $j \notin Tabu_i^{-1}$ and $(B_i(visit, j)_{j \in S_\Theta} = 0$ or $j \notin \Theta)$ do
    $B'_j \leftarrow g_j(B_i)$; check modified backward Rule 1. and Rule 2.;
    if $B'_j$ is not dominated by any label in $List(\mathbf{B_j})$ then
        $List(\mathbf{B_j}) \leftarrow List(\mathbf{B_j}) \cup \{B'_j\}$;
        $Backward \leftarrow Backward \cup \{j\}$; $H_F = \max\{B'^{adjust}_j(time), H_B\}$;
    end
end
```

5.4. Implementation Strategies

5.4.1. Preprocessing and Candidate List

The search space of MVPRC is extremely large. Therefore, preprocessing controlling the feasible arcs is an effective way to improve the algorithmic performance. First, as in section 3.2, any visit to electric rechargers through an ICV is defined infeasible. The associated variables are set to 0. Also, we allow no consecutive visits to rechargers, i.e., $(i, j, e) = 0, \forall i, j \in E$. Second, due to the truck capacity constraint, shipments that cannot be fit on the same truck k are pivoted. Third, the intra-node distance constraint (18) detects the infeasible arc combinations. Next, we preprocess the nodes that cannot be fit into the same time windows. For example, suppose shipment 1 is restricted to be delivered within time interval $[0, 10]$ and shipment 2 is within $[35, 50]$. Since the difference is more than the longest layover period, there is no feasible arc (i, j, k) on the network. Furthermore, shipments with a lower common carrier cost m_i than the stop cost p are deemed to be outsourced. Based on the above conditions, we preprocess the graph before the column generation framework. For each node i, we create a candidate list which stores all the nodes that can be directly linked to i without violating any of the above conditions. Conversely, for nodes that cannot be visited by i that are specified by the conditions, we store them in a forbidden list called $Tabu_i$. Similarly, we define the forbidden list $Tabu_i^{-1}$ for backward search by subtracting and checking the resources.

5.4.2. Heuristic Pricing Procedure

To speed up the generation of dedicated fleet paths with negative reduced cost, we first employ a simple heuristic procedure. The two developed exact labeling algorithms are implemented only if the heuristic procedure stops generating paths with negative reduced cost. This procedure has two steps. Firstly, for each node $i \in \bar{I}$, we reduce the search space by allowing only the extension from i to ρ nodes with the smallest arc reduced cost $\bar{c}_{ijk}$. In this way, at most $\rho \cdot (n + m)$ REF operations can be performed in each iteration. Whenever the first step fails, the number of candidate nodes are enlarged to 2ρ and then to 3ρ. Secondly, we set a limit to the number of labels kept along the label extensions. For example, we can start with allowing only 500 labels to be stored in $List()$ and then increase the number to 1000 if 500 failed to generate paths with negative reduced costs, and so on.

5.4.3. 2-cycle Elimination

As our relaxed labeling algorithm allows cycles to exist in the extension steps, non-elementary paths contain a cycle of the form (i, j, i), e.g. $(0, 1, 2, 1, 3, 0')$. These cycles are known as 2-cycles and are the most frequently occurring type of cycle caused by DP-based approaches (Desrochers et al. (1992)). When extending labels as shown in Algorithm 1 and Algorithm 2, we apply the 2-cycle elimination procedure introduced by Desrochers et al. (1992). The basic idea is to keep the best and the second-best partial paths for each state i and re-evaluate the new cost of extending i to j complying with the resources. The value of j is determined by two conditions, one is set to i's predecessor,the other is not. We add the predecessor to S_Θ and updated this value on a rolling-basis.

Recall that we use the relaxed state space Θ in the backward and forward labeling algorithm to keep track of the critical shipments. This technique drastically reduces the number of states required to be explored. However, this manipulation introduces cycles in the generated solutions. Goel and Irnich (2014) have computed the minimum mandatory layover and break periods between two customers. Here, we discover another way to expand set Θ based on an approximate lower bound of the layovers. More specifically, a 2-cycle of the form (i, j, i) may be possible if $\max\{a_i + \frac{(\frac{d_{0i}}{g} + \lambda_i)}{\xi} \cdot \beta^{lb}, a_j\} + \frac{(\frac{d_{0i}}{g} + \lambda_i)}{\xi} \cdot \beta^{lb} \leqslant b_i$. We check each pair $i, j \in \bar{I}$ with this inequality before calling the algorithm, each time an arc (i, j) is valid, we add node i to Θ to ensure elementarity along the path.

5.5. Modified Dominance Criteria

Consider the dedicated fleet pricing problem of the format (41). Recall that the 2-path inequality and the multistar inequality are both robust, The reduced cost changed by adding them to the RMP can be decomposed by arcs on the subproblem network, as shown in (42). However, applying the SR inequality interrupts the search space and the feasibility of the generated columns, see Jepsen et al. (2008). The dominance criteria developed in Rule 1. and Rule 2. are no longer valid. The two rules need to be modified to capture the dual values invoked by the SR inequality (35). During the execution of the modified algorithm, we need only to consider non-dominated paths.

By solving the separation problem (38), we get a SR inequality (35) with the modified reduced cost shown in (43). Set $S \in I$ is of cardinality three and $0 < \phi \leqslant \omega(S)$. Using the notations of labels, we can express the coefficient of the dual value χ_r applied on the current partial path by $\lfloor \frac{\omega(S \cap L_i(nodes))}{\phi} \rfloor$. Then, the first improvement on the two dominance criteria is produced by substituting the conditions $L_i^1(nodes) \subseteq L_i^2(nodes)$ and $L_i^1(visit, n) \leqslant L_i^2(visit, n)$ by $\bar{L}_i^1(nodes) \subseteq \bar{L}_i^2(nodes)$. In fact, this means the possible space of extensions for label L^1 is larger than that for label L^2. As further cuts are identified and added, Rule 1. and Rule 2. will not be effective as in the initial stage because penalties had already been imposed. Therefore, rare conditions in the two rules are satisfied. In order to overcome this problem, we note the following property of (43).

$$\tilde{c}_q = \hat{c}_q - \lfloor \frac{\omega(S \cap L_i(nodes))}{\phi} \rfloor \cdot \chi_r = \tilde{c}_q - \lfloor \omega(S \cap L_i(nodes)) \mod \phi \rfloor \cdot \chi_r. \tag{67}$$

Therefore, we define $m(L_i) = \omega(S \cap L_i(nodes)) \mod \phi$ after the last dual penalty has been applied on the reduced cost by delivering ϕ demands through set S. In the case that L_i^1 dominates L_i^2, $L_i^1(cost) \leqslant L_i^2(cost)$, and $\omega(S \cap L_i^1(nodes) \leqslant \omega(S \cap L_i^2(nodes)$. Thus, $L_i^1(\tilde{c}) \leqslant L_i^2(\tilde{c})$, given $\chi_r < 0$. Only demands after the last penalty towards the target node will be affected, see Jepsen et al. (2008). Considering two labels L^1 and L^2, then the larger the $m(L_i)$, the sooner will the corresponding label incur another penalty to the reduced cost. We define the following set:

$$M(L_i^1, L_i^2) = \{S \in SEP : m(L_i^1) > m(L_i^2)\}. \tag{68}$$

where SEP denotes combinations of the set S found by the separation optimization that induce active SR inequality (35). The dominance criterion Rule 1. can be strengthened to:

Proposition 5.3 (Rule 3.). *Define two labels L^1 and L^2 associated with the same node i. Then, label L_i^2 is dominated by label L_i^1 only if all the following conditions are satisfied and at least one of the inequalities is strict. Therefore, L_i^2 can be discarded. $L_i^q(\tilde{c})$ denotes the modified reduced cost of label q imposed with SR inequalities.*

$$\begin{aligned}
\forall i \in \bar{I}_0 \cup \{0'\} \qquad & \bar{L}_i^1(nodes) \subseteq \bar{L}_i^2(nodes), L_i^1(demand) \leqslant L_i^2(demand) \\
& L_i^{1,adjust}(time) \leqslant L_i^{2,adjust}(time), L_i^1(work) \leqslant L_i^2(work), \\
& L_i^1(electric) \leqslant L_i^2(electric), L_i^1(recharge) \leqslant L_i^2(recharge), \\
& L_i^1(layover) \leqslant L_i^2(layover), L_i^1(latest) \geqslant L_i^2(latest) \\
& L_i^1(\tilde{c}) - \sum_{M(L_i^1, L_i^2)} \chi_r \leqslant L_i^2(\tilde{c}).
\end{aligned}$$

Proof. Consider any possible extension $L_j^2 \notin \bar{L}_i^2$. Because for any $S \in M(L_i^1, L_i^2)$ we have $m(L_i^1) > m(L_i^2)$, the future coefficients for the dual variables maintain the following relationship.

$$\lfloor \frac{\omega(S \cap L_j^2(nodes)) + m(L_i^1)}{\phi} \rfloor \geqslant \lfloor \frac{\omega(S \cap L_j^2(nodes)) + m(L_i^2)}{\phi} \rfloor. \tag{69}$$

Since $0 \leqslant m(L_i^2) < m(L_i^1) \leqslant \phi$, the left-hand side of 69 is at most one unit larger than the right-hand side, which means label L^1 can conduct at most one more χ_r penalty to the reduced cost than L^2. This lead to:

$$\lfloor \frac{\omega(S \cap L_j^2(nodes)) + m(L_i^1)}{\phi} \rfloor - 1 \leqslant \lfloor \frac{\omega(S \cap L_j^2(nodes)) + m(L_i^2)}{\phi} \rfloor. \tag{70}$$

Therefore, we have:

$$\begin{aligned} L_j^1(\tilde{c}) &= L_i^1(\tilde{c}) + \hat{c}_{ijk} - \lfloor \frac{\omega(S \cap L_j^2(nodes)) + m(L_i^1)}{\phi} \rfloor \cdot \chi_r \\ &= L_i^1(\tilde{c}) - \chi_r + \hat{c}_{ijk} - (\lfloor \frac{\omega(S \cap L_j^2(nodes)) + m(L_i^1)}{\phi} \rfloor - 1) \cdot \chi_r \\ &\leqslant L_i^2(\tilde{c}) + \hat{c}_{ijk} - \lfloor \frac{\omega(S \cap L_j^2(nodes)) + m(L_i^2)}{\phi} \rfloor \cdot \chi_r \\ &= L_j^2(\tilde{c}). \end{aligned}$$

Because we allow multiple SR inequalities to be added in a single iteration, this indicates $L_i^1(\tilde{c}) - \sum_{M(L_i^1, L_i^2)} \chi_r \leqslant L_i^2(\tilde{c})$ is satisfied. The other conditions are ensured by Rule 1. Hence, label L^2 is dominated by label L^1. □

By similar adjustment of Rule 2. we can reach a modified dominance criteria Rule 4.

5.6. Branching Rules

When the subproblem does not pass any columns with negative reduced cost to the RMP, the LP solver provides us the optimal solution for the linear relaxation of the set partitioning reformulation. If the solution is integer and all the constraints are respected, this solution is also optimal for MVRPC. Otherwise, the solution is fractional and a branch-and-bound tree must be explored. Additional columns could be generated at each branch. In our implementation, we refer to the best-bound strategy used by Desrochers et al. (1992) and Dabia et al. (2019,a). Note that all the branching decisions are made based on the arc-flow formulation ($\mathcal{P}$). The variables are projected by equation (25). The branching strategy can be separated into multiple levels: (i) we start with branching on the outsourcing decision variable y_i. The algorithm finds the y_i that is closest to 0.5 in each iteration and creates two branches $y_i \leqslant 0$ and $y_i \geqslant 1$. When the variables y_i are integers for all $i \in I$, we proceed to the next level of branching. (ii) The algorithm branches on the number of vehicles $\sum_{k \in V} \sum_{i \in \bar{I}} x_{0ik}$ over all vehicle types and sizes. Two branches are created, $\sum_{k \in V} \sum_{i \in \bar{I}} x_{0ik} \leqslant \lfloor \sum_{k \in V} \sum_{i \in \bar{I}} x_{0ik}^{LP} \rfloor$ and $\sum_{k \in V} \sum_{i \in \bar{I}} x_{0ik} \geqslant \lceil \sum_{k \in V} \sum_{i \in \bar{I}} x_{0ik}^{LP} \rceil$, where x^{LP} is the optimal fractional solution at the current branch. (iii) If the number of total vehicles is an integer, then we decide to branch on the number of routes per truck type k. We choose the truck type k for which the fractional part of the number is closest to 0.5. Thereafter, a branch is created, $\sum_{i \in \bar{I}} x_{0ik} \leqslant \lfloor \sum_{i \in \bar{I}} x_{0ik}^{LP} \rfloor$ and $\sum_{i \in \bar{I}} x_{0ik} \geqslant \lceil \sum_{i \in \bar{I}} x_{0ik}^{LP} \rceil$. (iv) If the previous two numbers are both integers, we start branching on the number of recharges. First, we choose to branch on the fractional portion of the total number of visits to $i \in E$ that closest to 0.5. Then, a branch on each location of the recharger is created follow the same logic. For every branch created up to this stage, the decision is imposed by adding the associated inequality to the RMP. The corresponding dual values of the added inequality need to be transferred to the reduced cost of the related columns. Each time the past four branching strategies are utilized, the current solution is no longer feasible, which means a reoptimization is required on the linear relaxation.

Next, branching decisions are taken on the arcs of the subproblem network. (v) we check the pair of $(i, j, k) \notin A^-$ such that $x_{ijk}^{LP} + x_{jik}^{LP}$ is closest to 0.5, and imposes two branches $x_{ijk} + x_{jik} \leqslant 0$ and $x_{ijk} + x_{jik} \geqslant 1$. (v) Finally, among all arcs (i, j, k) with a fractional value in the current solution, we choose several x_{ijk}^{LP} close to 0.5. Ties are broken by preferring arcs with higher cost $\bar{c}_{ijk}$. Incident arcs to j on vehicle k are removed temporarily from the graph G if x_{ijk}^{LP} is set to 1. The branch-and-bound tree is enumerated according to the impact of the branching decisions on the two child nodes of the candidates. In other words, each time we reoptimize the RMP, the increase of the lower bounds of the two child nodes are measured, and the maximum is chosen as the next move. The number of branch candidates is set as 20 in our study. We adopt the hybrid search technique used by Desaulniers et al. (2016), in which subtrees will be explored only if their father nodes' lower bound is within a gap of ε of the current best lower bound.

This process is repeated until the entire search tree is explored.

6. Computational Results

In this section, we present computational experiments to analyze the effectiveness of the proposed branch-and-cut-and-price algorithm and to assess the impacts of allowing mixed type vehicle fleets and outsourcing options. Section 6.1 describes the instance and parameter settings used in our study. All algorithms are implemented in Python 3.8 on an Intel Core i9 CPU, 2.3 GHz 8-Core, 16 GB of RAM. For all experiments, we use a time limit of 2 hours. The LP relaxation of the master problem is solved using commercial solver Gurobi 9.0.

6.1. Parameter Settings

The benchmark set for MVRPC is the same as introduced in Dang et al. (2020), where the 11 instances are provided by DHL Supply Chain. We apply the following modifications to obtain MVRPC instances: 1) 40 selected rechargers are deployed to several warehouse locations; 2) the battery capacity of the electric vehicles, energy consumption, and other parameters are set by the medium configurations of Tesla's Semi truck, see Tesla Motors Inc. (2020); 3) Instances $S1$, $S2$ and $S5$ shown in Table 1 are directly tested, while we randomly generate another three instances of 80, 120, and 150 customers from instance $M0$ and $M1$; 4) Other route-related parameters and parameters for the trucks are shown in Tables 3 and 4, respectively. Note that the fixed cost per day of each truck type is estimated by the purchase cost divided by 7-year asset depreciation with 260 yearly working days. This estimation is an industry convention provided by experienced transportation analysts. For the cost of outsourcing, we again use the tariff sheets depending on the geographical location of the customers. For simplicity, we access the tariff in advance and assign the common carrier cost for each shipment. For all instances, we set the number of available vehicles to be five for the four truck types. At any point during the search, we limit the number of active valid inequalities to be 100. An initial upper bound of the problem is determined by the cost of outsourcing all shipments.

Table 3. Default settings on parameters.

Route related parameters	
Average speed	55 mph
Stop cost	$30
Minimum duration of a layover period	10 hours
Maximum duration of a layover period	14 hours
Maximum working time per day	14 hours
Maximum intra-node distance	120 miles
EV battery capacity	720 kwh
EV power consumption	1.8 kw/mile
Recharging rate	0.015 hour/kw

Table 4. Available vehicle type information.

	Sizes	Weight capacity (lbs)	Volume capacity	\$ Cost per mile	\$ Fixed cost per day
Vehicle types	Size 18	20,000	1000	1	100
	Size 26	30,000	2000	1.5	150
	Size 53	45,000	3000	2	200
	Tesla Semi-truck	40,000	3000	0.1	300

6.2. Algorithmic Performance

First, we study the performances of the MILP formulation and our branch-and-cut-and-price algorithms. Impacts of the three sets of valid inequalities are compared between robust and non-robust categories. A valid inequality is generated when it is violated by more than 0.2. The heuristic pricing procedures introduced in section 5.4 are called to speed up the procedure of finding the routes with negative reduced cost. The number of labels stored in the $List()$ is updated by increasing the number of labels by 500 each time we call the pricing problem. At most three of the routes for each vehicle type with the negative reduced costs are added to the RMP in the same pricing iteration.

Table 5 shows the results over the six instances by the branch-and-cut solver using our MILP formulation and the BCP algorithms with different valid inequalities. The first setting of the BCP is to disable the cutting planes so that a branch-and-price method is utilized. Based on the analytical results in section 4.1, we decide to combine the 2-path inequalities and the generalized large multistar inequalities together because they are both developed based on the arc-flow formulation and are thus robust inequalities. Next, SR inequalities are applied to the current BCP algorithm to test the effectiveness of the CG rank-1 cuts. In Table 5, The sizes of the instances are given in the column named *Instances*, (e.g., S1-N25 indicates instance S1 has 25 shipments). The time in seconds indicates when the instances are solved to proven optimality within the time limit 7200 seconds. In the following columns, the number of branch-and-bound nodes explored in the master problem (*Nodes*) and the relative integrality gap between upper and lower bounds (*Gap (%)*) are reported. The relative gap is computed as $\frac{z^{IP}-z^{LP}}{z^{IP}}$, where z^{IP} and z^{LP} are the optimal values of the integer solution and the linear relaxation. For instances that are not solved to optimality within the two-hour limit, we report the final gaps.

We see that MILP formulation is not competitive even for the modest-size instances, which were also observed in previous works for other VRP variants. The main reason is that the LP relaxation bound of $(\mathcal{P})$ is so weak that a large number of nodes needs to be explored to prove optimality. The branch-and-price method without applying any cutting plane is able to solve four out of the six instances and yields a small optimality gap for the two large instances that are not solved within the time limit. The BCP method with the two robust valid inequalities solves the same number of instances as the method without the cuts, however, the run time is decreased and the final gaps are reduced for the two large-sized, which means the two sets of cuts are effective. When the SR inequalities are separated and applied to the BCP algorithm, the run time reduction is more obvious. One more instance M1-N120 with 120 shipments can be solved to optimality within the two-hour limit. Compared with the two robust inequalities, SR inequality outperforms the other two and has the biggest influence on improving the lower bounds. Finally, we observe that the number of nodes fathomed is very small compared with the MILP solver, and the root optimality gap is close to 0 for those solved instances.

Table 5. Computational performance of MILP solver and BCP algorithm with different set of valid inequalities.

Instances	MILP Solver			B&P			B&P+2-path+Multistar			BCP		
	Time (sec)	Nodes	Gap (%)	Time (sec)	Nodes	Gap (%)	Time (sec)	Nodes	Gap (%)	Time (sec)	Nodes	Gap (%)
S1-N25	1879.90	4,495,682	0.0	649.50	1920	0.0	682.18	1109	0.0	630.73	940	0.0
S2-N50	7200	1,239,368	32.7	4280.40	5732	0.4	4095.28	3271	0.4	3476.10	2413	0.3
S5-102	7200	956,124	28.8	2187.85	1875	1.0	2099.12	1880	1.0	1793.72	1325	0.9
M0-N80	7200	311,589	58.5	7066.43	3930	1.4	6815.20	3651	1.4	5223.84	2719	1.2
M1-N120	7200	105,271	76.4	7200	4187	4.3	7200	3918	3.6	7130.57	3145	1.5
M1-N150	7200	67,593	83.9	7200	4352	15.5	7200	4217	14.9	7200	4108	13.1

Second, we investigate the performance of the branch-and-cut-and-price algorithms with forward and bidirectional labeling procedures for MVRPC. We compare the performance of the forward labeling and the bidirectional labeling algorithms. The heuristic pricing procedures are used in this experiment. Table 6 gives results for the four small-sized instances. For each labeling algorithm, the table contains the time spent on the pricing problem in seconds, the number of labels generated during the iterations, and the number of pricing problem iterations needed. A single pricing problem iteration is completed when the optimal routes are passed to the RMP to get the new LP solution.

The results of this test show that the bidirectional labeling algorithm is superior to the mono-directional version. The solution by bidirectional search of a single instance takes on average over one third the time forward search needs. The number of labels generated in total deceased by nearly half by utilizing the bidirectional method. However, we notice that the average number of pricing problem iterations needed to solve a single instance is nearly identical for both versions. By subtracting the solution time shown in Table 5 by the times spent in the pricing problems shown in Table 6, we have an interesting finding. Most of the solution time is spent on strong branching, but in most cases, strong branching does help to keep the size of the branch-and-bound tree small.

Figure 4 shows an example of the accumulated labels generated by the three labeling directions, i.e., forward, backward, and bi-directional. This example indicates that the mono-directional labeling algorithms generate about twice the labels of the bidirectional method. In fact, at the dynamic halfway point, the bidirectional labeling algorithm merges the forward and backward labels. Recall that, for labels that cannot be merged from either direction, we extend the labels crossing the halfway point until the depot (0 or $0'$).

Table 6. Comparisons between forward and bidirectional labeling algorithms.

Instances	Forward labeling algorithm			Bidirectional labeling algorithm		
	Time (sec)	No. labels	No. pricing iter.	Time (sec)	No. labels	No. pricing iter.
S1-N25	295.55	7718	35	108.40	4223	34
S2-N50	1585.79	20,135	115	974.71	13,570	111
S5-102	887.32	52,146	260	560.84	23,210	275
M0-N80	2959.45	96,479	310	1880.05	34,158	302

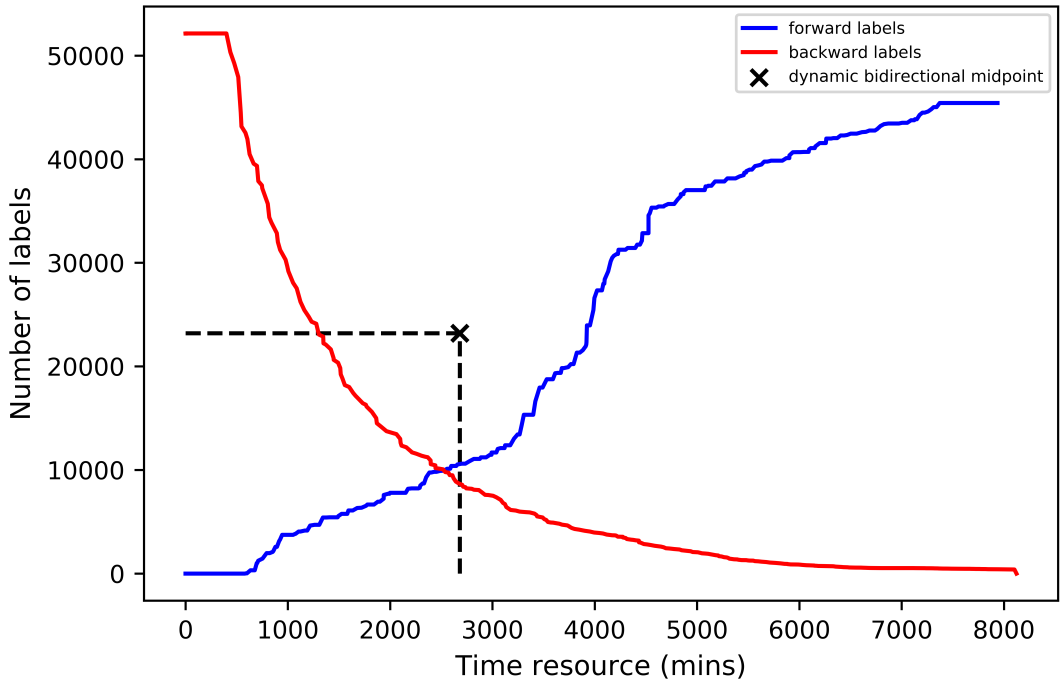
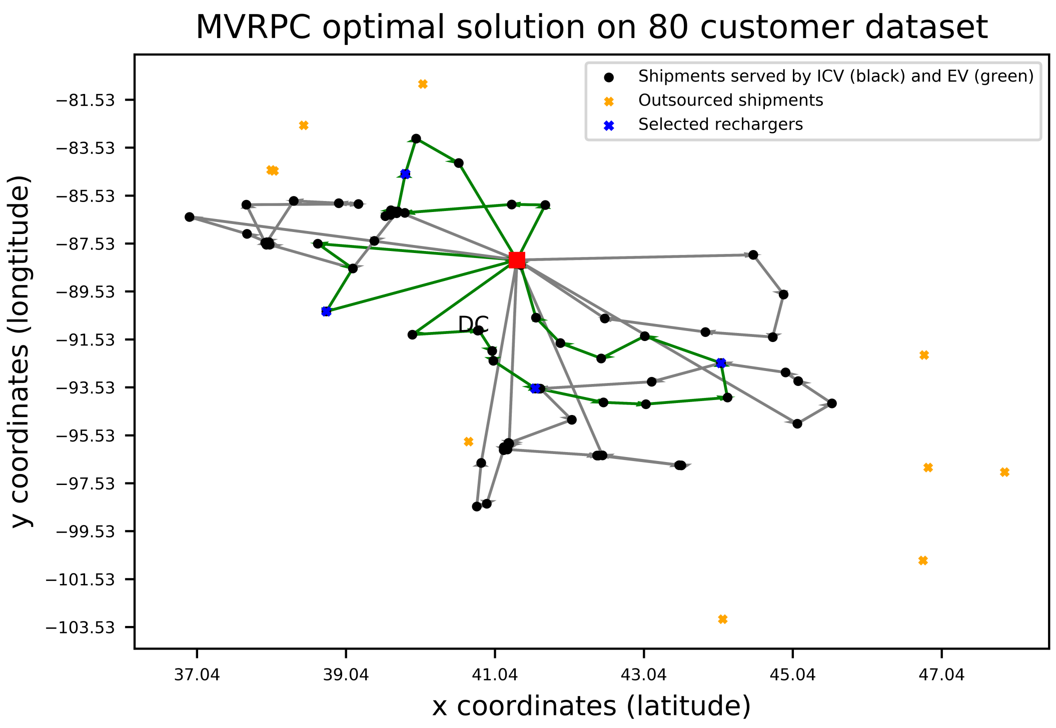


Figure 4. Forward and backward labels generated for instance M0-N80, where the "x" denotes the number of labels $= 23,210$ generated by the bidirectional labeling method at the halfway point $Time = 2681$.

6.3. Sensitivity Analyses on Recharging and Outsourcing

By closely looking into the structure of the optimal solutions visualized in the figures, we can derive some insights by conducting a few sensitivity analyses.

To start, we investigate the impact of the vehicle fixed cost and the battery capacity. Recall that different from the previous works, MVRPC considers an accumulated daily usage cost of the trucks, which means the fixed vehicle cost may affect the solution structures because more layovers imply higher truck usage cost. While Tesla's Semi truck is still a protocol, BYD's 8TT has already been implemented in the market. The parameters of an electric tractor are: 40,000 lbs weight capacity, 409 kwh battery capacity, 2.6 kw/mile of EV power consumption, and fixed cost per day of \$209. The energy cost per mile is still set as the electric cost divided by mileage, which is $\$0.1/mile$. Figure 5 shows the optimal solutions of instance M0-N80 where the left panel (**??**) with the EV settings is shown in Table 4 and the right panel (**??**) with the EV settings as BYD's 8TT. Clearly, customers located in more remote areas tend to be outsourced. Moreover, these customers either have relatively smaller demands or lower common carrier costs than shipping with the dedicated fleet. When the EV cost increases, it is expected that more shipments are served by either ICVs or common carriers. However, due to the battery capacity deduction, we observed in Figure 5 ICV routes and common carriers covered all the shipments while the BYD's 8TT is introduced to the mixed fleet, which has cheaper daily usage cost. Moreover, the solution time is reduced by more than 35% together with a 5% cost increase. This is due to the decrease of battery power caused by less feasible EV routes. In other words, the algorithm does not need to consider selecting and scheduling over the 40 rechargers and thus reduces the solution time. But in general, we observed most instances are sensitive to the change of EV daily usage cost.



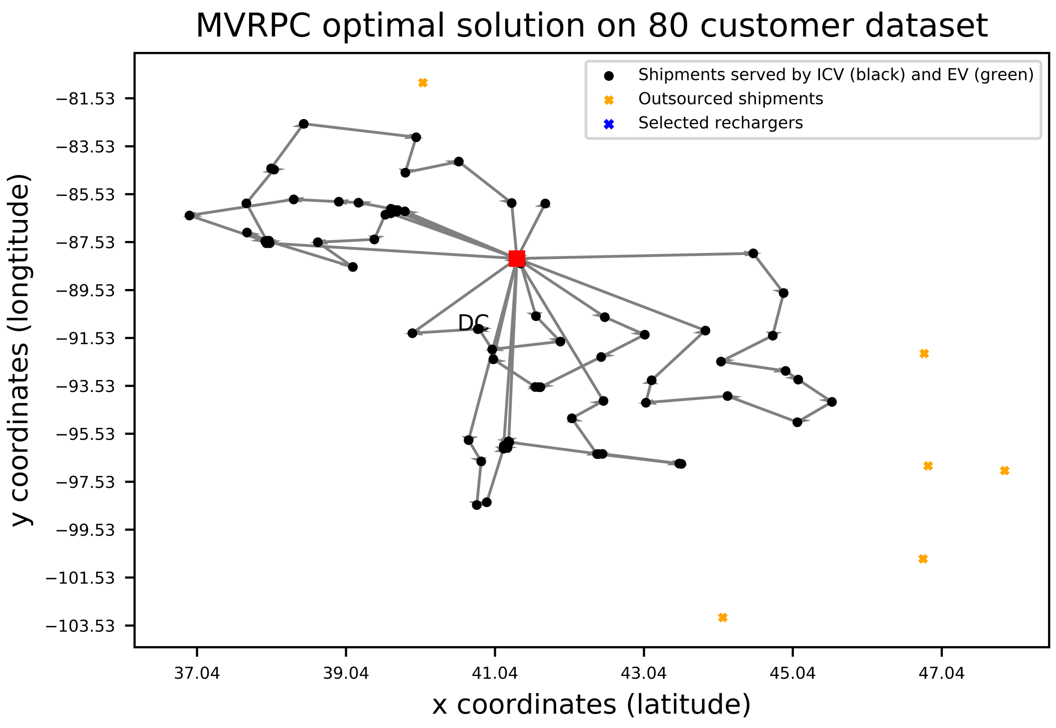
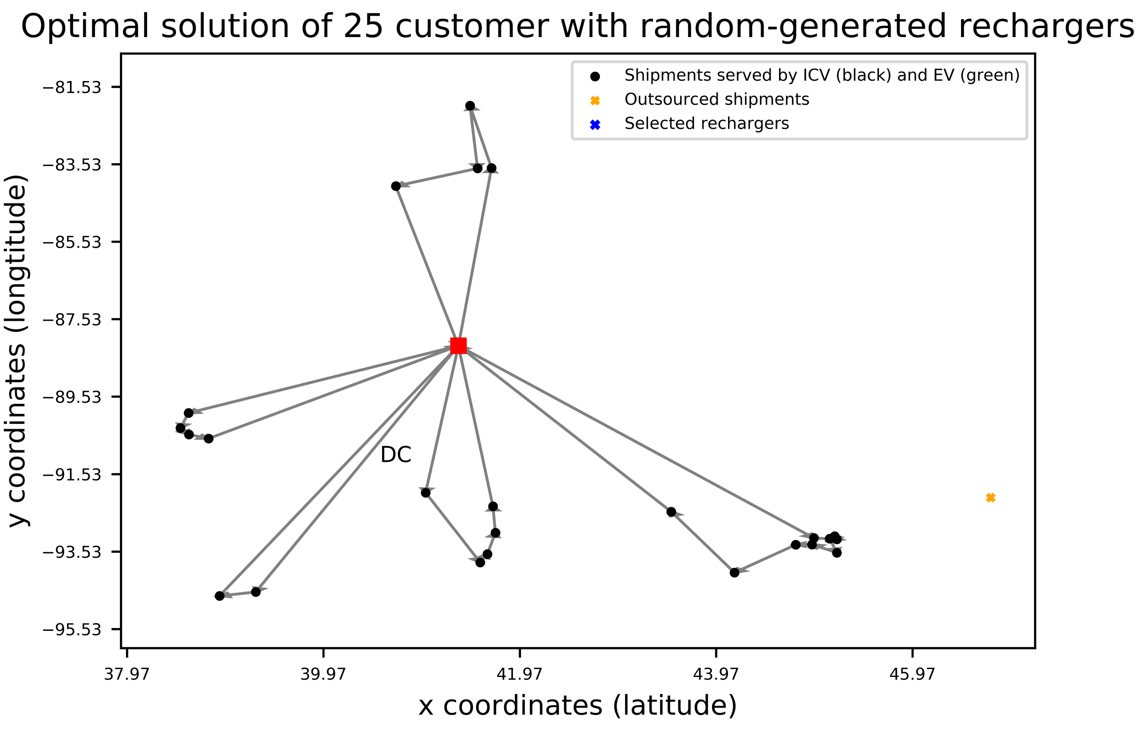


Figure 5. Example solutions for instance M0-N80, optimal solution with Tesla Semi truck in the mixed fleet, and optimal solution with BYD 8TT in the mixed fleet.

Figure 6 shows a comparison between deploying random-generated rechargers and the cherry-picked locations. Clearly, since the recharger locations are not well-designed, the optimal solution shown in Figure 6 does not utilize any EV route. In fact, the longer distance caused by visiting a recharger result in inefficient EV routes or infeasible EV paths due to tight time windows and possible layover periods. However, when we deploy the rechargers at or near to the customers' warehouses, three ICV routes are substituted by EV routes to optimize the network costs.



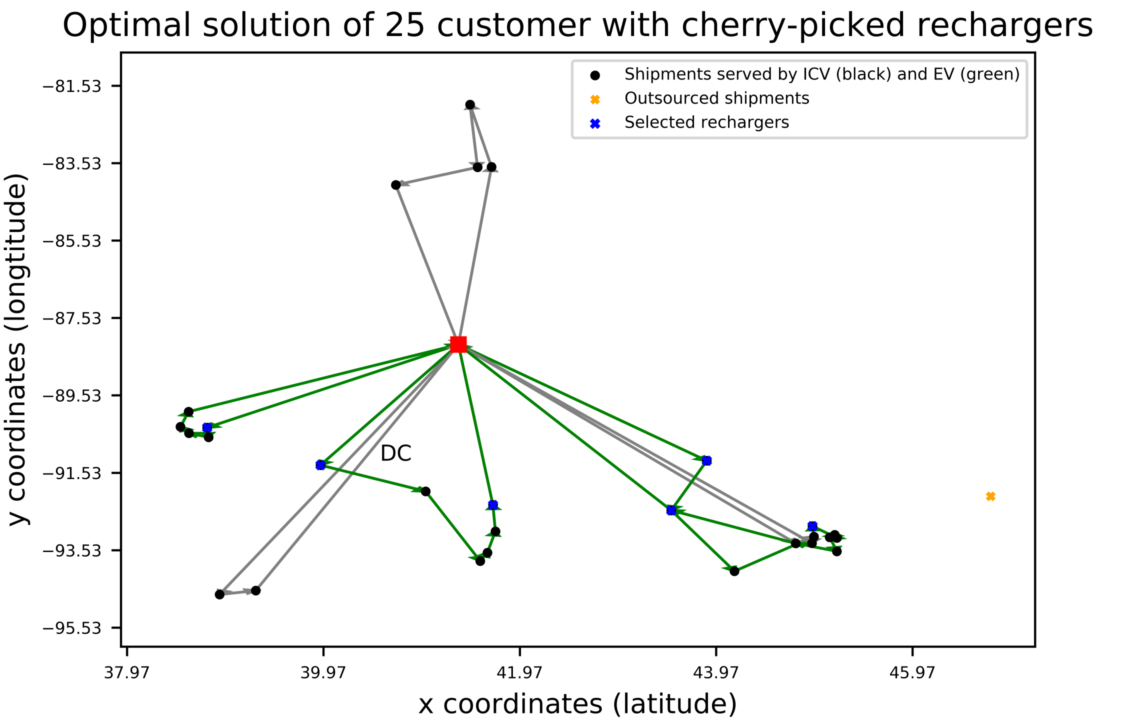


Figure 6. Impact of recharger locations to the optimal solution of a 25-shipments instance.

Next, we investigate the impact of the outsourcing option. We discuss the impact of outsourcing on the structure of the optimal routes. We look at the solution by disabling the outsourcing option for instance M0-N80. Other parameters are set to be the same as in Figure **??**. In this case, all shipments must be delivered by the mixed fleet. Compared with the optimal solution with an outsourcing option, the solution provided in Figure 7 results in 30% higher objective values. Furthermore, by outsourcing some customers, better routes with fewer trucks carrying a single or two shipments can be established. This might be due to the exclusion of uneconomical deliveries such that their time windows are tight or their locations are far away from the depot.

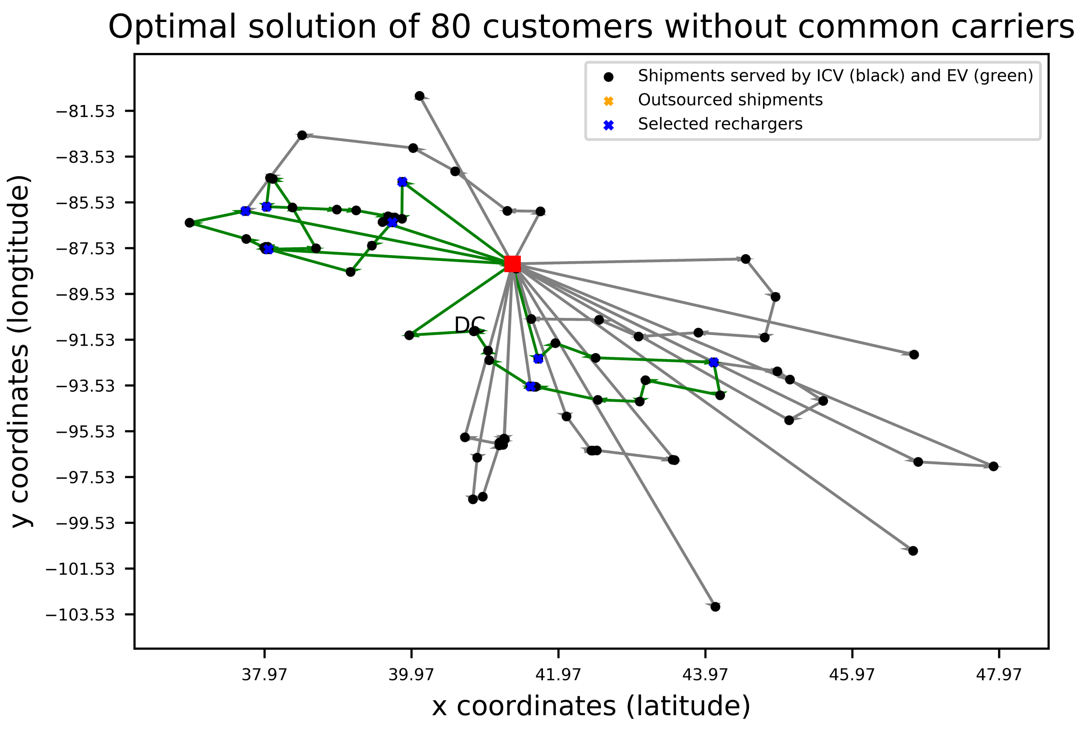


Figure 7. Solution of instance M0-N80 without the outsourcing option.

7. Conclusions and Future Works

In this paper, we presented the first exact approach for the MVRPC, which is a new variant of the vehicle routing problem with common carriers researched in the literature. New features are included to solve the question with heterogeneous fleet sizes, flexible layover periods, and the electric vehicle recharges. This is the first exact approach designed for the mixed fleet routing and mode decision problem.

The exact algorithm is based on the branch-and-cut-and-price framework. We provide a MILP formulation for the problem. Since the arc-flow MILP is usually not efficient in solving large-scale cases, we define an extensive set-partitioning reformulation for column generation. In this reformulation, variables for the fleet routes are considered to be further established in the subproblem pricing procedures, whereas the outsourcing decisions are made in the master problem. The performance of the algorithm is enhanced by using three sets of valid inequalities. To efficiently solve the subproblem, we designed new bounded labeling algorithms that successfully combine the idea of decremental state space relaxation and the dynamic halfway points. Mono-directional and bidirectional labeling algorithms for generating feasible routes are presented. Their efficiency results from tailored REFs that allow for time adjustments, renewable resources tracking, and strong dominance rules. Efficient implementation strategies were introduced to speed up the solution process.

In numerical studies, we demonstrate that our algorithms are capable of solving real-world instances with up to 120 customers and 40 rechargers. In addition, we show that the special case of CG rank-1 cuts – subrow inequality is more effective in cutting off fractional solutions than the other two developed valid inequalities based on arc-flow variables. Moreover, we show that, especially for large instances, the bidirectional labeling is superior compared to the mono-directional one. The time savings are mainly due to the reduction in the number of labels.

As this is the first exact algorithm for this variant of problems, there is apparently room for some improvement and future research. First, we notice that the main run time of the BCP algorithms are in the strong branching. Applying new branching techniques, e.g. see Alvarez et al. (2017), would be strongly beneficial to improve the performance of the algorithm. Second, different recharging policies, like partial recharging, non-linear power consumption, and different recharger costs/schedules, can be applied to the problem, leading to more complex models and solutions.

References

Alcaraz J, Caballero-Arnaldos, Vales-Alonso (2019) Rich vehicle routing problem with last-mile outsourcing decisions. *Transportation Research Part E* 129:263-286.

Alvarez AM, Louveaux Q, Wehenkel L (2017) A machine learning-based approximation of strong branching. *INFORMS Journal on Computing* 2017 Feb;29(1):185-95.

Andelmin J, Bartolini E (2017) An Exact Algorithm for the Green Vehicle Routing Problem. *Transportation Science* 51(4):1288-1303.

Arasu M. T., Anwar H, Ahmed Q, Rizzoni G (2019) Energy Optimal Routing of a Delivery Vehicle Fleet With Diverse Powertrains. *Proceedings of the ASME 2019 Dynamic Systems and Control Conference* 1:Park City, Utah, USA. October 8–11, 2019.

Asghari M, Alehashem M (2020) Green vehicle routing problem: A state-of-the-art review. *International Journal of Production Economics* doi: https://doi.org/10.1016/j.ijpe.2020.107899.

Baldacci R, Mingozzi A, Roberti R (2011) New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research* 59(5):1269-1283.

Chu C-W (2005) A heuristic algorithm for the truckload and less-than-truck-load problem*Euro. Jo. Oper. Res.* 165(3):657-667.

Chvátal V (1973) Edmonds polytopes and hierarchy of combinatorial problems. *Discrete Mathematics* 4(4):305-337.

Conforti M, Cornuejols G, Zambelli G (2014) *Integer Programming* Chapter 5.1-5.3: 195-222, 1st Edition, Springer.

Conrad R.G. and Figliozzi M.A. (2011). The recharging vehicle routing problem. In Proceedings of the 2011 industrial engineering research conference (p. 8). IISE Norcross, GA.

Dabia S, Lai D, Vigo D (2019) An exact alogirhm for a rich vehicle routing problem with private fleet and common carrier. *Transportation Science* 53(4):986-1000.

Dabia S, Ropke S, Woensel T (2019) Cover Inequalities for a Vehicle Routing Problem with Time Windows and Shifts. *Transportation Science* 53(5):1354-1371.

Dang Y, Singh M, Allen T (2020) Network mode optimization for the DHL Supply Chain. *INFORMS Journal On Applied Analytics*. doi: 10.1287/inte.2020.1046.

Demir E, Bektas T, Laporte G (2012). An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research* 223(2), 346–359.

Desaulniers G, Errico F, Irnich S, Schneider M (2016) Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research* 64(6): 1388–1405.

Desaulniers G, Desrosiers J, Ioachim I, Solomon M. M., Soumis F, Villeneuve D (1998) A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. *T. Crainic, & G. Laporte (Eds.), Fleet management and logistics* 57–93. Boston, Dordrecht, London: Kluwer Academic Publisher.

Desaulniers G, Lessard F, Hadjar A (2008) Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science* 42(3):387-404.

Desrochers M, Desrosiers J, Solomon M (1992) A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research* 40(2):342-354.

Dror M (1994) Improvements and extensions to the Miller-Tucker-zemlin subtour elimination constraints. *Operations Research* 42:977-978.

DHL GoGreen Program (2020) Mission 2050: zero emissions. URL: https://www.dpdhl.com/en/sustainability /environment-and-solutions/gogreen-program. Accessed: 09/23/2020.

Desrochers M, Laporte G (1991) Note on the complexity of the shortest path mod- els for column generation in VRPTW. *Operations Research Letters* 10:27-36.

Federal Motor Carrier Safety Administration (2011) Hours of service of drivers. Federal Register. Vol. 76, No. 248, page 81134-81188.

Feillet D, Dejax P, Gendreau M, Gueguen C (2004) An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks* 44(3):216-229.

Fischetti M, Lodi A (2007) Optimizing over the first Chavátal closure *Math. Program.* Ser. B 110:3-20.

Fukasawa R, Longo H, Lysgaard J, Poggi de Aragao M, Reis M, Uchoa E, and Werneck RF (2006) Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Math. Program.* 106 (3):491-511.

Fukasawa R, He Q, Song Y (2016) A branch-cut-and-price algorithm for the energy minimization vehicle routing problem. *Transportation Science* 50(1):23-34.

Gahm C, Brabander C, Tuma A (2017) Vehicle routing problem with private fleet, multiple common carriers offering volume discounts, and rental options. *Transportation Res. Part E.* 97(C):192-216.

Goeke D, Schneider M (2015) Routing a mixed fleet of electric and conventional vehicles. *European Journal of Operational Research* 245:81-99.

Goel A and Irnich S (2014) An exact method for vehicle routing and truck driver scheduling problems. Technical Report No. 33, Jacobs University, School of Engineering and Science, Bremen, Germany.

Gomory RE (1958) Outline of an algorithm for integer solutions to linear programs *Bull AMS* 64:275-258.

Lübbecke M, Desrosiers J (2005) Selected topics in column generation. *Operations Research* 53(6):1007–1023.

Irnich S, Desaulniers G (2005) Shortest path problems with resource constraints. Desaulniers G, Desrosiers J, Solomon M, eds. *Column Generation (Springer, New York)*: 33–65.

Irnich S, Villeneuve D (2006) The shortest path problem with resource constraints and k-cycle elimination for $k \geqslant 3$. *INFORMS Journal on Computing* 18(3):391-406.

Jepsen M, Petersen B, Spoorendonk S, Pisinger D (2008) Subset-row inequalities applied to the vehicle routing problem with time windows. *Operations Research* 56(2):497-511.

Kallehauge B (2008) Formulations and exact algorithms for the vehicle routing problem with time windows. it Comput. Oper. Res. 35(7):2307–2330.

Kohl N, Desrosiers J, Madsen OBG, Solomon MM, Soumis F (1999) 2-path cuts for the vehicle routing problem with time window. *Transportation Science* 33(1):101-116.

Laporte G (2007) What you should know about the vehicle routing problem. *Naval Res. Logist.* 54(8):811-819.

Letchford AN, Eglese RW, Lysgaard J (2002) Multistars partial multistars and the capacitated vehicle routing problem. *Math. Program.* 94:21-40.

Lübbecke M.E., Desrosiers J (2005) Selected Topics in Column Generation. *Operations Research* 53(6):1007-1023.

Macrina G, Pugliese L.D.P., Guerriero, F, Laporte, G (2019) The green mixed fleet vehicle routing problem with partial battery recharging and time windows. *Computers and Operations Research* 101:183-199.

Moon I, Lee JH, Seong J (2012) Vehicle routing problem with time windows considering overtime and outsourcing vehicles. *Expert Syst. Appl.* 39(18):13202–13213.

Pelletier S, Jabali O, Laporte G (2016) 50th anniversary invited article— Goods distribution with electric vehicles: Review and research per- spectives. *Transportation Sience* 50(1): 3-22.

Petersen B., Pisinger D., Spoorendonk S. (2008) Chvátal-Gomory Rank-1 Cuts Used in a Dantzig-Wolfe Decomposition of the Vehicle Routing Problem with Time Windows. In: Golden B., Raghavan S., Wasil E. (eds) *The Vehicle Routing Problem: Latest Advances and New Challenges.* Operations Research/Computer Science Interfaces, vol 43. Springer, Boston, MA.

Righini G, Salani M (2006) Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optim.* 3(3):255–273.

Righini G, Salani M (2008) New dynamic programming algorithms for the resource constrained shortest path problem. *Networks* 51(3):155-170.

Rousseaua LM, Gendreaub M, Feilletc D (2007) Interior point stabilization for column generation. *Operations Research Letters* 35:660-668.

Schneider M, Stenger C, Goeke D (2014) The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations. *Transportation Science* 48(4): 500-520.

Semi-trailer, Wikipedia (2020) Electric Truck. URL: https://en.wikipedia.org/wiki/Electric truck. Accessed: 09/15/2020.

Tesla Motors Inc. (2020) Supercharger. URL: https://www.tesla.com/supercharger. Accessed: 09/23/2020.

Tilk C (2016) Branch-and-Price-and-Cut for the Vehicle Routing and Truck Driver Scheduling Problem. Working Papers 1616, Gutenberg School of Management and Economics, Johannes Gutenberg-Universität Mainz.

Tilk C, Rothenbächer A, Gschwind T, Irnich S (2017) Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. *European Journal of Operational Research* 261: 530-539.

Toth P, Vigo D (2014) *Vehicle routing: problems, methods, and applications, 2nd ed.* MOS-SIAM Series on Optimization, vol. 18 (SIAM, Philadelphia).

Vidal T, Maculan N, Satoru OL, Vaz Penna (2016) Large neighborhoods with implicit customer selection for vehicle routing problem with profits. *Transportation Science* 50(2):720-734.

Yaman H (2006) Formulations and valid inequalities for the heterogeneous vehicle routing problem. *Math. Program.* Ser. A 106:365-390.