Department of Information and Computing Sciences

Integrated Vehicle and Crew Scheduling for Electric Buses

Computing Science Master Thesis
7077270

Author:
T.D. van der Plas

Supervisors:
Dr. J.A. Hoogeveen
Dr. M.E. van Kooten Niekerk
P. de Bruin, MSc
M. Stikvoort, MSc

July 2025

1 Introduction

Electric vehicles are beginning to make up large portions of the fleet for public transport providers. In The Netherlands for example, approximately 21% of all registered buses utilize an electric drivetrain, as reported by the RDW (2025). To comply with strict Dutch and European regulations (2018) on climate and sustainability, this proportion must increase substantially; from 2025 onward, the procurement of new buses powered by non-renewable energy sources will no longer be permitted, with the objective that by 2030, all public transport buses operate with zero emissions. Individual line operators such as Qbuzz (2024) are therefore rapidly replacing their remaining buses that use fossil fuels, resulting in primarily electric-powered fleets.

Effective use of these new electric vehicles is however more challenging than that of a combustion based fleet. One of the logistical steps in which this is most apparent is that of vehicle scheduling. The Vehicle Scheduling Problem (VSP) aims to assign vehicles such that a set of trips are driven at minimum cost; in the case of buses, these trips are defined by the timetables for individual lines. A typical solution to this problem comes in the form of a collection of vehicle tasks, each of which can be seen as a schedule that an individual bus will follow throughout the day. It may start at a bus storage facility (more commonly called a depot), then perform one or more trips, before finally returning to the depot. In order to string multiple trips together, a bus must make an empty travel from the ending location of one trip to the starting location of the next. These empty travels, often called deadheads, are the main focus when determining the overall costs of a collection of vehicle tasks. There is ample choice as to which deadheads can be driven, as in theory the deadhead between each pair of compatible trips can be included during the creation of a vehicle task. It is therefore necessary to determine which of these deadheads are required in order to drive all trips while minimizing costs.

In the VSP and its variants, a commonly made assumption is that a vehicle can travel an entire day without having to refuel. With electric vehicles, this assumption is often not valid: recharging throughout the day is required as the range of an electric vehicle is limited. Additionally, recharging takes a long time when compared to the refueling of a combustion vehicle: refueling can be done in a matter of minutes, whereas recharging may take multiple hours for a full charge. The electrification of fleets therefore requires the VSP to be adapted to include maximum range constraints, as well as charging possibilities within a vehicle task. This extended version of the problem, often referred to as the Electric Vehicle Scheduling Problem (E-VSP), is significantly harder than its non-electric counterpart; the VSP with a single depot can be solved in polynomial time, whereas the E-VSP is NP-Hard as shown by Bunte and Kliewer (2009) and Sassi and Oulamara (2014) respectively.

As self-driving vehicles are not in general use yet at time of writing, buses also require a driver in order to be operated. The Crew Scheduling Problem (CSP) is therefore commonly the step that follows vehicle scheduling; in this, the objective is to make an assignment of drivers such that a driver is always present when a vehicle is moving. A driver may drive multiple vehicles throughout the day, handing vehicles over to other drivers at designated relief locations. Driver shifts consisting of driving time and breaks are constructed while following local labor regulations. The CSP is also NP-Hard, as shown by Fischetti et al. (1989).

Costs in the crew scheduling step often exceed those associated with the vehicles themselves, with an estimate by Perumal et al. (2019) putting crew member costs at around 60% of overall operational costs. Optimal solutions for the CSP are directly influenced by the selected vehicle tasks; depending on which deadhead is selected, the amount of vehicle idle time before a trip may vary. It may therefore not always be feasible for a driver to perform a handover or break between two trips depending on the deadheads used in the vehicle tasks. Sequentially minimizing costs in the VSP and CSP may therefore not result in a solution with overall lowest costs, as the crew schedules in the CSP may be improved greatly by incurring higher driving costs in the VSP. This was already pointed out in the 1980s by Bodin and Golden (1983), who instead advocated for an integrated approach; here, the costs of the VSP and CSP are minimized simultaneously, resulting in the Vehicle and Crew Scheduling Problem (VCSP).

A lot of work has already been done on the VSP, CSP and VCSP. Both the sequential and integrated approach have been extensively studied since the 1980s, as shown by reviews such as Ibarra-Rojas et al. (2015) and Ge et al. (2024). However as mentioned before, the introduction of electric vehicles has introduced significant constraints on recharging times and vehicle ranges, invalidating the critical assumption in many of these works that a vehicle was able to drive an entire day without being refueled. In response, the E-VSP has also been the focus of many studies going back to around 2014. We refer the reader to a survey by Perumal et al. (2022) for a detailed overview of recent progress.

The integrated VCSP with electric vehicles (E-VCSP) has seen less attention; to the best of our knowledge, only 5 works consider this problem. In these, simplifying assumptions are made about battery

Abbreviation	Definition
ALNS	Adaptive Large Neighborhood Search
B&P	Branch-and-Price
CG	Column Generation
CP	Constraint Programming
CSP	Crew Scheduling Problem
E-...	Problem ... with electric vehicles
LNS	Large Neighborhood Search
LS	Local Search
MDVSP	Multi Depot Vehicle Scheduling Problem
MIP	Mixed Integer Program
SAA	Simulated Annealing Algorithm
SDVSP	Single Depot Vehicle Scheduling Problem
SoC	State of Charge
TCO	Total Cost of Ownership
ToU	Time of Use
TVSP	Integrated Timetabling and Vehicle Scheduling Problem
VCSP	Integrated Vehicle and Crew Scheduling Problem
VSP	Vehicle Scheduling Problem

Table 1: Nomenclature used in this work

charging behavior which might limit real world applicability or accurate modeling of costs. In this work, our aim is therefore to introduce an integrated E-VCSP model which incorporates more realistic behavior for battery charging.

The rest of this work is organized as follows. In Section 2, we will give more background information on the E-VCSP and provide a formal problem definition. In Section 3, we review literature related to the E-VCSP and identify gaps in the current research. In Section 4, we discuss the methodology used in order to solve the E-VCSP. .... An overview of nomenclature and abbreviations used throughout this work has been included in Table 1.

2 Problem definition

In this section, we give a definition of the integrated Electric Vehicle and Crew Scheduling Problem (E-VCSP). In order to do so, we will first give a global overview of the problem at hand. Afterwards, we discuss the data which is given, before finally formalizing the resulting problem.

2.1 Overview

The E-VCSP is part of the planning process used in bus public transit, as can be seen in Figure 1. It comes directly after the timetabling step, in which trips are defined for each line according to passenger demand. Each individual timetabled trip represents a bus traveling along a line at a specified time. Oftentimes, these trips are scheduled regularly throughout the day from the start to the end of a line, however irregular schedules and partial travels are also possible.

With these trips being laid out, we now need to determine how they will be driven. Individual trips often have lengths of around 0.5-2 hours, therefore making it possible to perform multiple trips with a single vehicle and driver in a day. In order to perform a trip, a vehicle must travel to its starting location before driving it; this travel is called a deadhead. Deadheads occur between trips and between a trip and a depot, and are generally defined by a driving time and distance between a pair of locations. During a deadhead, the vehicle itself is empty except for the driver. It is therefore beneficial to minimize the total amount of time that vehicles and crew members spend driving deadheads, as costs incurred during this time do not have any direct benefit to passengers.

When considering electric vehicles, the time between trips is not only used for driving deadheads. As electric vehicles generally do not have enough range to drive an entire day, this downtime is also used in order to recharge the vehicle. A variety of different recharging methods are currently in use throughout the industry, however in this work we will exclusively consider conductive charging; specifically, we consider plug-in charging (which is the same method used for commercial EVs) and pantograph charging. These

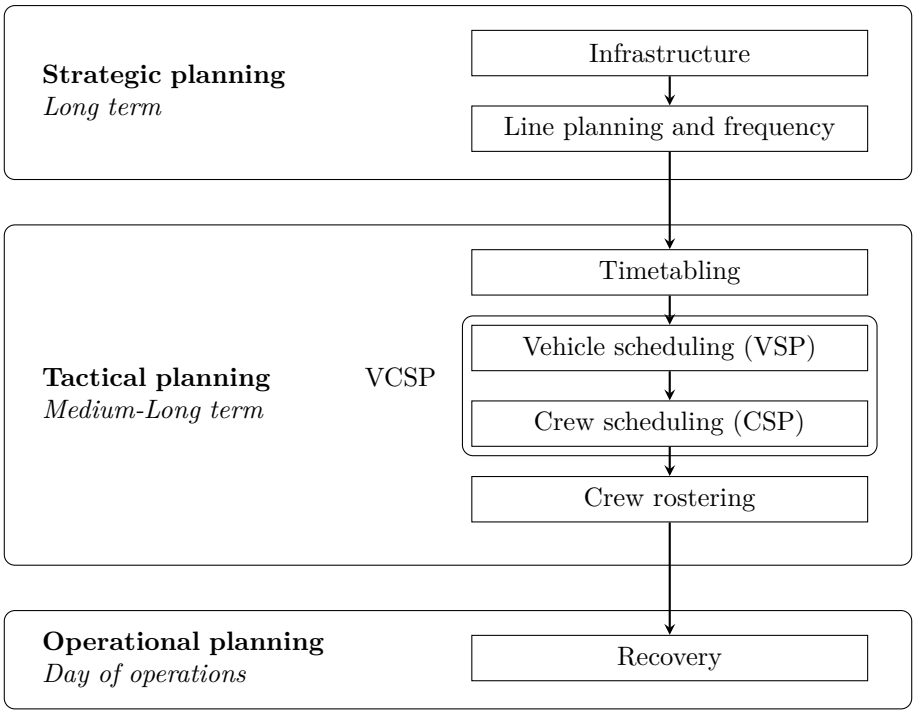

Figure 1: A general overview of the public transport planning process, based on Ceder and Wilson (1986), Ibarra-Rojas et al. (2015) and Perumal et al. (2022).

two techniques share many similarities in their behavior, with the largest difference being the physical connector that is used to connect the power source to the battery. For a detailed overview of available recharging methods and corresponding behavior, we refer the reader to a review by Zhou et al. (2024).

For conductive charging, recharging stations are often available at the depot, with additional stations sometimes being present at the starting or ending locations of trips (also called the terminal trip stops). Charging stations in the middle of a trip are generally not present. Recharging a vehicle at one of these locations follows a charging curve; this curve defines the charging rate when a vehicle is at a certain state of charge. For the most commonly used battery chemistry, lithium-ion, this curve is generally split into two phases: from 0 to around 80%, in which the charging rate is roughly constant, and from 80 to 100%, where it slows significantly. These phases often take similar amounts of time, and depending on the infrastructure available a full charge may take on the order of hours. Fast-charging infrastructure has become more prevalent in recent years with charging speeds of up to ∼350km/hour, however this still results in significant time spent charging during a vehicle task.

The resulting vehicle downtime necessitates careful planning of crew schedules. Crew members must follow local labor regulations, which often consist of maximum driving times throughout the day and required breaks. Combining these breaks with charging time for the vehicle can be beneficial, as this implies that both the crew member and vehicle have useful idle time. In order to make even better use of paid crew time, crew members may transfer between vehicles throughout the day.

These vehicle handovers are often only possible at a predetermined set of locations. These relief location have waiting areas or break rooms connected to them, providing a place for a crew member to stay before entering their next vehicle. A subset of these relief locations may also be used in order to start and end a shift; we shall call these locations crew bases.

As not all locations are suitable for handovers, a sequence of trips and deadheads which does not visit a relief point must be driven by a single driver. Such a sequence which starts and ends at a relief location is called a block, and represents the smallest unit of work that can be performed by a driver. Both vehicle and driver schedules can be seen as a collection of blocks, with possible idle time and break times to connect the blocks.

Overall, the goal is to find a set of feasible vehicle and crew schedules that minimizes costs. In this, vehicle schedules must ensure that each trip is driven while minimizing deadhead and charging costs. These schedules must also ensure that enough charging takes place such that the vehicle state of charge is within acceptable levels. Crew schedules, on the other hand, must ensure that each driven block is

covered. While doing so, local labor regulations must be taken into account.

Notation	Definition
Given	
$\mathcal{D}$	The depot
$t \in \mathcal{T}$	Set of all trips, t a single trip
$dh \in \mathcal{DH}$	Set of all feasible deadheads, dh a single deadhead
$b \in B$	Set of all blocks, b a single block
$v \in V$	Set of all vehicle tasks, v single vehicle task
$c \in C$	Set of all crew duties, c single crew duty
d_χ	Amount of charge used driving a trip/deadhead
$u_{dh,\sigma}$	Max. charge gained in deadhead dh with starting SoC σ
σ_{start}	SoC at beginning of block/task
σ_{min}	Minimum SoC of vehicle
σ_{max}	Maximum SoC of vehicle
Parameters	
$k_{i,t}$	Binary parameter, vehicle task v_i covers trip t
$l_{i,b}$	Binary parameter, vehicle task v_i uses block b
$m_{j,b}$	Binary parameter, crew duty c_j uses block b
d_j	Parameter, duration of crew duty c_j
$d_{j,long}$	Binary parameter, $d_j > 8.5$ hours
$\rho_{j,\chi}$	Binary parameter, crew duty c_j is of type χ
Decision variables	
$x_i \in \{0,1\}$	Usage of vehicle task v_i
$y_j \in \{0,1\}$	Usage of crew duty c_j
Helper functions	
$b(v) \subseteq B$	Set of blocks covered by v
$km(v)$	Kilometers traveled during v
$chg(v)$	Charge gained during v
$chl(v)$	Charge lost during v
$br(c)$	Total break time in c
$dur(\chi)$	Total duration of χ
$cost(\chi) \in \mathbb{R}^+$	Cost of task χ

Table 2: Notation used for formal problem description, where χ is used as a placeholder when multiple argument types can be applied.

2.2 Formal definition

We will now formally define the E-VCSP. In this, we consider a variant with a single depot, single vehicle type and charging stations without capacity limitations. Additionally, partial and non-linear charging are considered. For crew members, we will include a simplified version of the Dutch labor regulations. A summary of the notation used has been provided in Table 2.

Given timetables for each of the considered lines, let $\mathcal{T}$ be the set of trips that they define. Additionally, let $\mathcal{DH}$ be the set of deadheads that represent either a travel to or from the depot $\mathcal{D}$, or a travel between trips in $\mathcal{T}$. In this, we define DH as only containing deadheads which can feasibly be driven; that is, the travel time of the deadhead must be less than the time between its departure and the start of a target trip. Note that deadheads including a depot as either the origin or target are always considered to be time-feasible.

Next, let us define our battery behavior. For each trip t and deadhead dh, let d_t and d_{dh} respectively indicate the amount of charge that is used during its travel. Additionally, let $u_{dh,\sigma}$ indicate the maximum amount of charge that can be gained during a deadhead if a vehicle starts the deadhead at a State of Charge (SoC) of σ; in this, let charging at any possible given charging location be considered. Depending on available charging infrastructure, this may imply that a deadhead is not driven directly, instead making a time-feasible detour via a charging location which provides the fastest charging capabilities. SoC gained

at a location depends on given (possibly non-linear) charging curves for the vehicle, allowing for different amounts of charge to be achieved in the same time span depending on the initial SoC.

Lastly, let us define relief points and crew bases. A subset of the of all defined locations may serve as relief points; these locations are the only ones at which a handover or break for a crew member can take place. A subset of the relief points is a crew base; these locations are the only ones at which a crew member may start or end their duty.

Let a *block* be a sequence of trips and/or deadheads that must be driven by a single driver; that is, no relief points are visited during the sequence except at its start and end. Let an individual trip or deadhead in this sequence be called an *element*. In order for a block to be feasible, it must match the following requirements:

- A block consists of at least one trip or deadhead.
- Only the starting location of the first element and the ending location of the last element in the block are relief points.
- The element directly following a trip must be a deadhead and vice versa.
- For each deadhead dh driven, the vehicle may charge with some value $\alpha \cdot u_{dh,\sigma}$, where $0 \leq \alpha \leq 1$ represents a fraction of the maximal charge gained.
- Given an SoC σ_{start} at the beginning of the block, the SoC of the vehicle should remain within the range $[\sigma_{min}, \sigma_{max}]$ throughout the block.

Let the set of all feasible blocks be B. A schedule for a vehicle throughout the day can now be constructed as a sequence of blocks. Let us refer to such a schedule as a *vehicle task*. In order for a vehicle task to be feasible, the following conditions need to be met:

- The task must start and end at the depot.
- A vehicle task consists of one or more blocks.
- Sequential pairs of blocks must be compatible; that is, for each sequential pair of blocks b and b', b must either end in a deadhead whose target is the starting trip of b', or b must end in a trip which is the origin of the starting deadhead of b'.
- Given an SoC σ_{start} at the beginning of the task, the SoC of the vehicle should always remain within $[\sigma_{min}, \sigma_{max}]$ during all blocks.

Let the set of all feasible vehicle tasks be V. For a vehicle task v, let us define its cost as being $cost(v) = \gamma_1 + \gamma_2 \cdot km(v) + \gamma_3 \cdot chg(v) + \gamma_4 \cdot chl(v)$, where $km(v)$, $chg(v)$ and $chl(v)$ represent kilometers traveled, charge gained and charge lost during v respectively and γ is a vector of cost scalars.

Next, let us consider crew schedules. As with vehicle tasks, crew schedule primarily consist of blocks that need to be driven throughout the day. Crew members are however bound by labor regulations; in most cases, this means that we will need to include break times and a maximum overall duration in the schedule. Additionally, idle time at handover locations may be required in order to wait for an arriving vehicle, and crew members may travel to different locations through means other than driving a vehicle. A *crew duty* therefore consists of a sequence of blocks and idle/break times. As with vehicle tasks, we will refer to individual parts of this sequence as elements. Exact feasibility constraints for crew duties will differ per country and region based on local labor regulations, however in general the following conditions must always be met:

- The crew duty consists of one or more blocks and zero or more travels and break/idle times.
- Sequential pairs of elements must be compatible. That is, for each sequential pair of elements e and e', the location at the end of e must be the same as the start of e'. Additionally, the ending time of e must be equal to the starting time of e'.
- The crew duty must start and end at a crew base.

In addition to these universal constraints, we also incorporate a simplified version of the Dutch labor regulations into our feasibility. In order to do this we also assume that a duty has a predetermined type; this type defines additional constraints on both the duty itself as well as the eventual selection of duties that is used in order to cover the required blocks. Let us start by defining the constraints which apply regardless of the duty type:

- The crew duty may have a duty length of at most 9 hours
- A crew member may drive for at most 4 hours without a break.
- For duties between 4 and 5.5 hours, at least one break of 15 minutes must be included.
- For duties longer than 5.5 hours, at least 40 minutes of break time must be included. All breaks must be longer than 15 minutes, and at least one break must be 20 minutes or longer.
- For duties starting after 15:00, at least one break between 16:30 and 20:30 of at least 20 minutes must be included.

In addition to these, the following duty types are defined, each with their own additional constraints:

- **Early**: Must end before 16:30.
- **Day**: Must end between 16:30 and 18:15.
- **Late**: Must start after 13:00 and end before 26:30.
- **Night**: Must start before 24:00 and end after 26:30. Duty may be at most 7 hours long.
- **Between**: Must start before 13:00 and end after 18:15.
- **Broken**: Must start after 5:30 and end before 19:30. The duty is broken in two by a long unpaid idle period of at least 1.5 hours, which starts and ends at a crew base. Each of the two parts of the duty is considered individually for the universal break requirements discussed previously.

Constraints are also placed on the shifts which are eventually selected in order to cover the blocks:

- At most 15% of selected duties may have a length of more than 8.5 hours.
- The average selected duty length must be at most 8 hours.
- At most 30% of all selected duties are of type *broken*.
- At most 10% of all selected duties are of type *between*.

Let the set of all crew duties that match these conditions be C. For a crew duty c let us define its cost as being $cost(c) = \beta_1 + \beta_2 \cdot br(c) + \beta_3 \cdot dur(c)$, where $br(c)$ and $dur(c)$ represent the total break time in and duration of c respectively and β is a vector of cost scalars.

Using the definitions of the blocks B, vehicle tasks V and crew duties C, we can formulate a linear program to describe our integrated problem. In this, let x_i and y_j be binary variables indicating the usage of $v_i \in V$ and $c_j \in C$ respectively. Additionally, let $k_{i,t}$ indicate that v_i covers trip $t \in \mathcal{T}$, let $l_{i,b}$ indicate that v_i contains block $b \in B$, and let with $m_{j,b}$ indicate that c_j includes block $b \in B$. Let $\rho_{j,\text{broken}}$ and $\rho_{j,\text{between}}$ indicate whether a duty c_j is of type *broken* or *between* respectively. In this, $k_{i,t}$, $l_{i,b}$, $m_{j,b}$ and $\rho_{j,\chi}$ are all be binary parameters. Lastly, let and let d_j be a parameter indicating the duration of duty c_j, and let $d_{j,long}$ be a binary parameter indicating whether $d_j > 8.5$ hours. Our cost minimization can then be formulated as follows:

$$\min \quad \sum_{1 \leq i \leq |V|} x_i \cdot cost(v_i) + \sum_{1 \leq j \leq |C|} y_j \cdot cost(c_j) \tag{1}$$

Subject to:

$$\sum_{1 \leq i \leq |V|} x_i k_{i,t} = 1 \qquad \forall t = 1, \ldots, |T| \tag{2}$$

$$\sum_{1 \leq i \leq |V|} x_i l_{i,b} - \sum_{1 \leq j \leq |C|} y_j m_{j,b} = 0 \qquad \forall b \in B \tag{3}$$

$$\sum_{cd_j \in CD} y_j \cdot (\frac{d_j}{8\ hours} - 1) \leq 0 \tag{4}$$

$$\sum_{cd_j \in CD} y_j \cdot (0.15 - d_{j,long}) \geq 0 \tag{5}$$

$$\sum_{cd_j \in CD} y_j \cdot (0.3 - \rho_{j,broken}) \geq 0 \tag{6}$$

$$\sum_{cd_j \in CD} y_j \cdot (0.1 - \rho_{j,between}) \geq 0 \tag{7}$$

$$x_i \in \{0, 1\} \qquad \forall i = 1, \ldots, |V| \tag{8}$$

$$y_j \in \{0, 1\} \qquad \forall j = 1, \ldots, |C| \tag{9}$$

Here, constraint (2) ensures that all trips are covered by a vehicle, and constraint (3) ensures that for each block, the number of assigned drivers is equal to the number of assigned vehicles. Constraint (4) ensures a maximum average duty length, and constraints (5-7) ensure that the number of duties which are longer than 8.5 hours / of a certain type are within limits. This integrated formulation minimizes the costs for both vehicle and crew duties simultaneously. For comparison, when the E-VSP and CSP are solved sequentially, the formulation becomes the following. First, vehicle tasks are selected such that their costs are minimized:

$$\min \sum_{1 \leq i \leq |V|} x_i \cdot cost(v_i) \tag{10}$$

Subject to:

$$\sum_{1 \leq i \leq |V|} x_i k_{i,t} = 1 \qquad \forall t = 1, \ldots, |\mathcal{T}| \tag{11}$$

$$x_i \in \{0, 1\} \qquad \forall i = 1, \ldots, |V| \tag{12}$$

Here, constraint (11) once again ensures that all trips are covered by a vehicle task. Once the vehicle tasks are selected, we can use them to derive the set of blocks that are driven. Let $b(v) \subseteq B$ represent the set of blocks driven by a vehicle task v. The overall set of blocks driven is then $B' = \bigcup_{v_i \in V, x_i = 1} b(v_i)$, and let the crew duties for which all used blocks appear in B' be C'. Note that in our construction of B', each block is present at most once due to the fact that all trips are covered exactly once in our E-VSP formulation.
Using this, our formulation for crew duties becomes the following:

$$\min \sum_{1 \leq j \leq |C'|} y_j \cdot cost(c_j) \tag{13}$$

Subject to:

$$\sum_{1 \leq j \leq |C'|} y_j m_{j,b} = 1 \qquad \forall b \in B' \tag{14}$$

$$\sum_{cd_j \in CD} y_j \cdot (\frac{d_j}{8\ hours} - 1) \leq 0 \tag{15}$$

$$\sum_{cd_j \in CD} y_j \cdot (0.15 - d_{j,long}) \geq 0 \tag{16}$$

$$\sum_{cd_j \in CD} y_j \cdot (0.3 - \rho_{j,broken}) \geq 0 \tag{17}$$

$$\sum_{cd_j \in CD} y_j \cdot (0.1 - \rho_{j,between}) \geq 0 \tag{18}$$

$$y_j \in \{0, 1\} \qquad \forall j = 1, \ldots, |C'| \tag{19}$$

Here, we now consider a greatly reduced number of available blocks and crew duties, resulting in a formulation with significantly less rows and columns than the equivalent integrated formulation.

3 Related work

In this section, we discuss work related to our research into the E-VCSP. A summary of how batteries and charging behavior are modeled in the discussed works has been included in Table 3.

3.1 (E-)VSP

Before considering previous work on the E-VSP, we first cover the most basic form of vehicle scheduling: that which only considers a single depot, single vehicle type and unlimited vehicle ranges. This problem, often referred to as the Single Depot Vehicle Scheduling Problem (SDVSP), forms the underlying basis of both the multi-depot and electric vehicle extensions that we consider later. We therefore give a brief summary of two common models and solution methods used for the SDVSP, thereby having a baseline to which we can compare extensions. For a more comprehensive overview on different models used for the SDVSP and Multi-Depot VSP (MDVSP), we refer the reader to a review by Bunte and Kliewer (2009).

In order to find a solution for the SDVSP, the problem can be transformed into one of finding a min-cost flow in a graph. The graph can be constructed as follows: For all trips $\mathcal{T}$, add a pair of nodes representing the start and end of the trip respectively. Add an arc from the start of each trip to its end with capacity 1, lower bound 1 and cost equal to that of driving the trip. Next, connect the end of each trip t to the start of each trip t' for which the deadhead between the two is feasible; that is, there is enough time to drive the deadhead from t to t' before the scheduled starting time of t'. Let each of these arcs have a capacity 1 and a cost equal to that of driving the deadhead from t to t'. Lastly, let us introduce a pair of nodes representing the depot at the start and end of the day respectively. Connect the depot start node to the start of each trip, and do the same for the depot end node and the end of each trip. For each of these arcs, let the capacity once more be 1 and let the cost be equal to that of driving of the deadhead between the depot and trip. Fixed vehicle costs can be represented by adding costs to the arcs leaving the depot.

We can now find the min-cost flow in this graph; in this, let the depot start node be the source, and let the depot end node be the sink. Due to our construction, all trips must be covered by exactly 1 flow, resulting in flow paths which we can directly use as vehicle tasks due to the assumption that our vehicles have infinite range. It is therefore also shown that the SDVSP can be solved in polynomial time, as polynomial time min-cost flow algorithms exist and the graph elements are of size $|V| = O(|\mathcal{T}|)$ and $|A| = O(|\mathcal{T}|^2)$.

Alternatively, the problem can be transformed into one of finding a minimum weight matching within a bipartite graph. Here, nodes are created for each trip on either side of the of the graph, where the left side of the graph represents origins and the right side represents destinations. Additionally, for each available vehicle, a depot node is added on either side of the graph. An edge between a origin-destination pair is added when the deadhead between them is feasible, and the cost of driving the deadhead is used as its weight. A minimum weight matching in this graph can then be used to construct vehicle tasks by following origin-destination pairs from a depot node on the left side to a depot node on the right side.

	Model	ToU	SoC	Nonlinear Ch.	Partial Ch.	Ch. Location	Degradation
Li (2014)	E-VSP	No	D	No	No	D	No
Van Kooten Niekerk et al. (2017)	E-VSP	Yes	C/D	Yes	Yes	D/T	Yes
Olsen and Kliewer (2020)	E-VSP	No	C	Yes	Yes	D/T	No
Zhang et al. (2021)	E-VSP	No	C/D	Yes	Yes	D	Yes
Parmentier et al. (2023)	E-VSP	No	C	Yes	Yes	D/T	No
De Vos et al. (2024)	E-VSP	No	D	Yes	Yes	D/T	No
Perumal et al. (2021)	E-VCSP	No	C	No	No	D	No
Wang et al. (2022)	E-VCSP	Yes	C	No	Yes	D	No
Sistig and Sauer (2023)	E-VCSP	No	C	No	Yes	D/T	No
Shen and Li (2023)	E-VCSP	No	C	No	Yes	D/T	No
Cong et al. (2024)	E-VCSP	Yes	C	No	Yes	D	No
Ham and Park (2021)	E-VRPTW	Yes	C	No	Yes	D	No
Stadnichuk et al. (2024)	E-TVSP	No	C	No	Yes	D/T	No

Table 3: A brief overview of battery modeling in E-VCSP related literature. SoC modeled as (D)iscrete or (C)ontinuous variable, Charge locations at (D)epot or (T)erminal trip stops, Degradation of battery in cost function

Two common extensions to the SDVSP make it NP-Hard: the inclusion of multiple vehicle types, as well as the use of multiple depots under the assumption that vehicles must return to their depot of origin. Both of these extensions are also discussed in Bunte and Kliewer (2009). The modification to the SDVSP flow network is the same in either case: an additional source/sink pair can be added for each new depot or vehicle type, and connected to the trips in the same way as the original depot. The problem then turns into into an integral multi-commodity flow, which has been shown to be NP-Hard by Even et al. (1975).

The introduction of any resource constraints within the VSP has also been shown to be NP-Hard (see Bodin and Golden (1983)). The E-VSP specifically deals with constraints on the driving range of vehicles, thereby making it closely related to the vehicle scheduling problem with route time constraints (VSP-RTC) as described by Haghani and Banihashemi (2002). The key difference between these two problems is that the E-VSP allows for (partial) recharging of a vehicle throughout the operating period, whereas the VSP-RTC assumes a fixed maximum travel time for the vehicle within the given period. The E-VSP has been shown to be NP-Hard by Sassi and Oulamara (2014).

Li (2014) is one of the first to consider a solution method for the E-VSP. They consider a single-depot case with a single vehicle type, in which the assumption is made that full recharging (or battery swaps) can be performed in a fixed 5-minute time window. The model is based on the SDVSP flow network, with the inclusion of driving time constraints for the vehicle tasks. Additionally, time-discretized nodes are added to represent capacitated battery charging/swap stations. Connections are made from the trip nodes to the charging nodes when a travel between the two is feasible, allowing for a vehicle to perform one or more charging action during its task. By using discretized charging nodes, charging station capacity is enforced with the use of flow constraints. For smaller instances, the model can be solved to optimality using column generation and branch-and-price (B&P). For larger instances, an alternate approach using truncated column generation followed by a local search to find a local optimum is used instead. The proposed methods are tested on trips in the San Francisco Bay Area, with a maximum instance size of 242 trips. These tests resulted in optimality gaps of $< 5\%$ for buses able to drive 150km, and between 7-15% for a range of 120km depending on the instance.

Van Kooten Niekerk et al. (2017) introduce two models which aim to solve the single depot E-VSP while taking into account time dependent energy prices (ToU pricing), nonlinear charging times and battery degradation due to depth of discharge. The first model extends the traditional SDVSP flow model by discretizing the depot nodes over time, and adding a continuous variable to each trip node representing the vehicle SoC at its start. This only allows for the formulation that uses linear charging curves, and does not incorporate degradation or ToU. The second model additionally discretizes both depot and trip nodes for individual starting SoC values. Charge-feasibility of deadheads can then be considered during graph construction, only adding arcs between pairs of nodes when the SoC difference between them can be achieved through driving and charging during the deadhead. This also allows for easy integration of nonlinear charging curves, degradation, and ToU pricing, as the SoC of the vehicle at each discretized trip node is known at construction time. The second model is solved using CG and Lagrangean relaxation. Tests are performed using data provided by Belgian bus company De Lijn in the city Leuven, using a total of 543 trips. They show that the second model can be solved in a considerably shorter time frame for large instances with similar results to the first.

Olsen and Kliewer (2020) consider a multi-depot E-VSP, in which they model the nonlinear phase of charging as an exponential function. In order to solve, they implement a greedy heuristic to construct vehicle tasks. Their primary focus is comparing (piecewise) linear approximations for the second phase of charging with an exponential function based approximation. They conclude that SoC and required charging times are more comparable to real life behavior when using the exponential function.

Zhang et al. (2021) consider an E-VSP variant with a single depot, capacitated charging infrastructure, non-linear charging behavior and battery degradation. They model this by combining elements from Li (2014) and van Kooten Niekerk et al. (2017), resulting in a network which has time and charge discretized depot, trip and charging station nodes; arcs between these are again only present if the deadhead is both time and charge-feasible. Solutions are found using a combination of CG and B&P, and tests are performed on both randomly generated instances as well as 6 not yet electrified lines with up to 160 and 197 trips respectively.

Parmentier et al. (2023) consider a scalable approach to the E-VSP with non-linear charging. They introduce the concept of nondominated charging arcs, which are represented as multiple deadhead arcs between a pair of nodes within the traditional SDVSP network. Their use allows for a manageable amount of charging possibilities to be considered between trips when multiple charging points are available, as an

arc is only included if there is not another arc available with higher resulting charge and lower cost. In order to solve, a combination of CG and B&P techniques are used. Testing is done on the *large* instances introduced by Wen et al. (2016) which included up to 8 depots, 16 charging stations and 500 trips. Here, they are able to find solutions that only have an 0.06% optimality gap.

De Vos et al. (2024) consider the E-VSP with partial recharges and capacitated charging stations. Their model applies a similar discretization as the ones found in the work of van Kooten Niekerk et al. (2017) and Zhang et al. (2021). As with those models, power used during trips and deadhead arcs is rounded up to the nearest discrete value; this results in an underestimation of the actual SoC of the vehicle during its task, however ensures solutions that can be feasibly driven. This pessimistically rounded graph, which De Vos et al. refer to as the primal network, is accompanied by a dual network; in this graph, power used is rounded down, resulting in more deadheads becoming charge-feasible. The problem is then solved by applying CG with two separate approaches: branch-and-price and a diving heuristic. In this, the dual network is used in order to generate dual bounds that match those found in a non-discretized model, following ideas presented by Boland et al. (2017). Testing is performed on a bus concession south of Amsterdam with 816 trips, with subsets being used as smaller instances. Optimality gaps of 1.5-2.7% are achieved across instances. They additionally note that the framework as provided can easily be extended for nonlinear charging functions and depth-of-discharge battery degradation.

3.2 CSP

Given a solution to the (E-)VSP, the corresponding CSP is most often solved as a set partitioning (or set covering) problem. Here, the tasks described by the sequences of trips generated during vehicle scheduling must be covered by the individual schedules of crew members. This problem has been shown to be NP-Hard in general by Fischetti et al. (1989).

Research into this subject is primarily done in the context of airline crew planning; crew costs in this field are generally even higher than those found in the more general public transport sector, as shown in Barnhart et al. (2003). Additionally, strong labor unions and restrictive labor legislation due to safety concerns cause a large number of constraints to be applied to crew schedules, resulting in a non-trivial problem to solve.

Results achieved in the aviation space quite easily generalize to other sectors, and we therefore refer the reader to a review by Wen et al. (2021) for an overview of the state of the art.

3.3 (E-)VCSP

The VCSP is a widely studied problem. Following the call for integrated methods by Bodin and Golden (1983) and others in the 1980s, a large number of different methods has been applied to integrate the VSP and CSP. We refer the reader to a recent review by Ge et al. (2024) for a general overview of work done in the field in the past years.

One work that we will individually highlight is that of Huisman et al. (2005), due to its use of Lagrangean relaxation to connect the VSP and CSP . For readers unfamiliar with the technique, we recommend an introduction by Beasley (1993). Huisman et al. consider the multi-depot variant, and use a combination of CG and Lagrangean relaxation to solve both the MDVSP as well as the connection with the CSP. Of note is their assumption that crew members from each individual depot are only allowed to work on trips connected to said depot, allowing for individual depot CSPs to be solved as a subproblem. They test on instances in the Randstad metro area in the Netherlands with a maximum of 653 trips and 4 depots.

As for the electric counterpart of the VCSP, at time of writing we are aware of only five other works that discuss the integrated variant.

Perumal et al. (2021) were the first to offer a solution to the E-VCSP. They consider an instance of the problem in which only full recharges at the depot with a fixed duration of 120 minutes are possible. Solutions are found using ALNS, incorporating a B&P heuristic for constructing tasks which has been previously used to solve the VCSP, MDVSP and E-VSP by Friberg and Haase (1996), Pepin et al. (2009) and van Kooten Niekerk et al. (2017) respectively. The authors tested using real life data from lines in Denmark and Sweden with a maximum instance size of 1109 trips and multiple depots, and report an improvement of $1.17 - 4.37\%$ across different instances when compared to a sequential approach.

Wang et al. (2022) introduce a two layered model which allows for a mix of traditional combustion and electric buses using Particle Swarm Optimization and an ϵ-constraint based mechanism. The model incorporates partial depot charging, as well as measures to ensure that crew is primarily assigned to the

same vehicle throughout the day. A circular bus route with a single depot in Changchun, China with 68 daily trips is used as a basis for testing, with a focus on electric versus diesel usage and driver satisfaction.

Sistig and Sauer (2023) also offer an ALNS based approach, which aims to incorporate additional flexibility in vehicle scheduling when compared to the work presented by Perumal et al. (2021) by including partial recharges, opportunistic charging at terminal stops of trips and non-fixed ranges for the vehicles. A set of 3-step ALNS neighborhoods are used in order to solve, where each neighborhood follows the structure of first doing a number of improvement iterations to the E-VSP solution, deriving the corresponding CSP problem, before finally doing a number of improvement iterations on the CSP solution. The E-VSP and CSP iterations themselves also follow a ALNS structure, resulting in nested ALNS iterations being performed. The upper level 3-step neighborhoods only differ in the amount of iterations done in each of the two nested ALNS. This approach is tested using an instance of a city route in Germany, with a single depot and a total of 282 trips. Different scenarios based on possible crew break and relief locations were considered in order to compare diesel and electric TCO. Additionally, sensitivity analysis of the TCO was done for parameters such as costs for electricity and drivers.

Shen and Li (2023) provide a minimum-cost flow based framework for the E-VCSP. They consider the single depot variant, in which partial recharges are possible. Solutions are found by first generating a subset of all feasible blocks and corresponding crew duties using a matching-based heuristic approach. Afterwards, a MIP formulation is used in order select crew duties and create corresponding vehicle tasks that cover all trips. A city line in China with 270 daily trips and a single depot is used for testing, resulting in cost savings of up to 8.7% when compared to a sequential approach.

Cong et al. (2024) take a hybrid MIP and SAA based approach to optimize a mixed fleet of combustion and electric vehicles with ToU electricity pricing. In each SAA iteration, a collection of new E-VSP trip assignments are created using neighborhood operations, after which two MIP models are sequentially employed to solve for charging and crew schedules. The methods are tested on a collection of 3 bus routes originating from the same depot in Changchun City, China with a total of 520 trips across all routes. When compared to the sequential approach, the integrated vehicle schedule was able to reduce costs by 0.8%.

3.4 Other related fields

The VSP is closely related to the vehicle routing problem (VRP); in this problem, the aim is to find minimum cost routes for vehicles originating from a depot that need to pass multiple stops, most commonly for pickup or delivery with capacity constraints. The extension of the E-VRP which includes arrival time windows (E-VRPTW) is most closely related to the E-VSP, as the use of 0-width windows allows us to define the same precedence constraints as those naturally defined by trips in the VSP.

An example of work done on the E-VRPTW is that of Ham and Park (2021). They consider a single depot case in which they model ToU pricing and partial recharges during delivery routes. In order to model costs, a lexicographical minimization is done over the number of vehicles used, total distance traveled and energy recharged. In order to solve, a hybrid MIP and CP algorithm is used in which CP is used to model ToU related variables, and MIP is used to model the rest of the constraints.

Research has also been done into integrating the E-VSP with the step before it in the planning sequence: timetable planning. This problem, the E-TVSP, has recently been studied in the work of Stadnichuk et al. (2024). They allowed results of the E-VSP to introduce optimality cuts into the MIP used for creating timetable plans, thereby reducing overall cost. This is achieved by transforming the E-VSP problem into one of bin packing with conflicts, after which three different heuristic methods are applied and compared. They additionally prove that the bounds of the used heuristics are tight for their given instances.

3.5 Research gap

As can be seen in Table 3, research into the E-VSP has successfully incorporated many battery characteristics which can be of importance when determining overall operational costs and feasibility. The E-VCSP on the other hand has seen less progress: both non-linear charging and battery degradation have not yet been considered in any work at the time of writing. Additionally, no documented attempt at using a discretized SoC model for the E-VCSP has been made. In this work, we therefore aim to address two of these points: the use of a discretized model, as well as incorporating non-linear charging curves.

4 Methodology

In this section, we will discuss our methodology for solving the E-VCSP. In order to do so, we will first discuss our ways of solving the E-VSP and CSP sequentially; afterwards, we discuss how parts of these techniques can be reused in order to solve the integrated problem.

For both the E-VSP and CSP a column generation approach is implemented based on a set cover ILP. In order to make this approach computationally feasible, a combination of pre- and postprocessing is applied on the underlying networks and solutions. As large parts of the solution methods for these two problems are shared, a detailed overview of the any shared technique will only be provided in the E-VSP section. Where applicable, the CSP solution method will refer back to this.

4.1 E-VSP

Our approach for solving the E-VSP can roughly be broken up into 3 steps:

- *Preprocessing*, in which our initial data set is transformed into a graph which is simplified using heuristics.
- *Solving*, in which column generation using various sources is applied in order to solve the problem of finding a set of vehicle tasks which cover all trips.
- *Postprocessing*, in which duplicate trips are transformed into deadheads in order to reduce the total amount of driving hours required.

4.1.1 Preprocessing

Given are a set of trips $\mathcal{T}$, a set of locations $\mathcal{L}$, and a set of deadheads $\mathcal{DH}$ connecting pairs of locations. Of these locations, exactly one is the depot $\mathcal{D} \in \mathcal{L}$. Additionally, a subset of the locations $\mathcal{L}_c \subseteq \mathcal{L}$ has charging equipment installed. In order to keep notation compact, let us use $s(t)$ and $e(t)$ to refer to the start and end time of a trip t. Let us also use $o(t)$ and $d(t)$ to refer to the origin and destination location of t.

Let us transform this data into a graph $G_{vt} = (V_{vt}, A_{vt})$. Let $V_{vt} = \{d_s, d_e\} \cup \mathcal{T}$, where d_s and d_e represent the depot at the beginning and end of the day respectively. For each node $t \in \mathcal{T}$, add an arc to A_{vt} from d_s to t and from t to d_e if there is a deadhead in $\mathcal{DH}$ connecting the depot to $o(t)$ and $d(t)$ respectively. Next, add an arc to A_{vt} between each pair of trips t, t' if there is a deadhead in $dh \in \mathcal{DH}$ that connects the $d(t)$ to $o(t')$ and the duration of dh does not exceed $s(t') - e(t)$. This graph corresponds to the one found in the standard VSP formulation as described in Section 3.1; we will now modify it in order to be more computationally efficient and include additional charging capabilities.

Let us begin by reducing the overall size of the graph. In order to do this, let us first consider a single line within the provided timetable. Throughout the day, the number of trips traveling in one direction along the line will be roughly equal to the number of trips traveling the other direction. When this is the case, trips are often staggered such that a bus arriving at one end of the line can directly be used in order to drive to the other end, minimizing the amount of vehicle downtime. When considering the timetable as a whole, this pattern may also be found when combining multiple lines: a vehicle arriving after driving a trip on line A may be perfectly timed in order to then drive a trip on line B, assuming A and B share an endpoint. As these connections are often very good to include in a vehicle task due to short idle times, a timetable which contains these patterns can be used to substantially reduce the amount of arcs in the graph by forcing the use of these short turnarounds.

More formally: for each trip node t, let $A_t \subset A_{vt}$ be the set of all outgoing arcs from t to other trip nodes. Let $A_{t,st} \subseteq A_t$ be the collection of arcs in which for each arc from t to t' in $A_{t,st}$, $d(t) = o(t')$ and $s(t') - e(t) \leq \delta$ for some threshold δ. If $|A_{t,st}| \geq 1$, replace A_t with $A_{t,st}$ in A_{vt}. Otherwise, leave the original set of outgoing arcs the same. A δ in the order of 0 to 0.5 hours was found to be reasonable during initial testing, providing a balance between the number of arcs in the graph and the solution quality.

Once this initial reduction of arcs between trips has been performed, we will extend the remaining arcs in order to properly accommodate charging. Let us first assume that charging never occurs when driving to or from the depot; all charging operations will therefore take place in the time between pairs of trips. Additionally, let us assume that whenever we are idle at a charging station, we will charge. Given the current state of our graph G_{vt}, this implies that charging exclusively takes place if a trip starts or ends at a charging location, and a connecting trip allows for more idle time than the minimum charging time.

In order to increase the amount of charging options, let us consider the existing arcs in A_{vt} between pairs of trips t and t'; let us refer to these arcs as *base arcs*. For each charging location $l_c \in \mathcal{L}_c$ for which $l_c \neq d(t)$ and $l_c \neq o(t')$, we will attempt to add an additional charging arc. This arc will model the possibility of performing a detour between t and t' to l_c in order to charge there; let us refer to one of these arcs as a *detour arc*. Let us add a detour arc if there exists a deadhead $dh \in \mathcal{DH}$ from the ending location of t to l_c, a deadhead $dh' \in \mathcal{DH}$ from l_c to the start of t', and the combined duration of dh and dh' allows for enough time at l_c to meet the minimum charging time.

For all detour arcs, we assume that the time at the charging location is maximized. This means that we we always perform the deadhead towards the charging location at the earliest possible time, and the deadhead to the next trip at the latest possible time. For base arcs, let us also uniquely define where we will spend our idle time by maximizing time at a charging location; if the start and end location of a base arc are both non-charging or both charging locations, time at the start location is maximized as a tiebreaker. Knowing this, we can define the costs of each arc based on driving costs of the deadhead(s) and target trip, as well as SoC usage.

Any feasible vehicle task can now be described as a path in G from d_s to d_e. The costs for a task can be defined as a base cost of using a vehicle plus the sum of all used arc costs and a KWh costs. In this, we assume that the cost of energy remains constant throughout the day and at different charging locations, and that vehicles will be fully recharged overnight for the same KWh cost as during the day. Additionally note that not every path from d_s to d_e describes a feasible vehicle task; depending on the arcs used, the amount of time that a vehicle has to charge may not be sufficient in order to keep it within SoC bounds during the task. As we previously defined how much time is spent at a charging location during an arc, this information can be used to determine whether a task is feasible or not.

4.1.2 Solving

In order to now solve the E-VSP, a collection of feasible vehicle tasks that cover the trips $\mathcal{T}$ must be found. In order to do this, we use a column generation approach; this has previously been shown to be an effective solution method for the E-VSP by works such as van Kooten Niekerk et al. (2017), Zhang et al. (2021) and de Vos et al. (2024). The ILP used to model this is based on a set covering formulation. In this let VT represent the available set of vehicle tasks, let x_i be a binary decision variable indicating the use of vehicle task $vt_i \in VT$, and let $k_{i,t}$ be a parameter defining whether vehicle task vt_i includes the trip $t \in \mathcal{T}$.

$$\min \quad \sum_{vt_i \in VT} x_i \cdot cost(vt_i) \tag{20}$$

Subject to:

$$\sum_{vt_i \in VT} x_i k_{i,t} \geq 1 \qquad \forall t \in \mathcal{T} \tag{21}$$

$$x_i \in \{0, 1\} \qquad \forall vt_i \in VT \tag{22}$$

Here, constraint (21) ensures that all trips are covered by at least one vehicle task. This problem can be relaxed to an LP by replacing constraint (22) with the following:

$$x_i \geq 0 \qquad \forall vt_i \in VT \tag{23}$$

In this relaxed problem, let the dual cost of constraint (21) be π_t with $t \in \mathcal{T}$. For a new vehicle task vt_0, its reduced cost would equal:

$$\text{cost}(vt_0) - \sum_{t \in \mathcal{T}} k_{0,t} \pi_t \tag{24}$$

Or put simply: the base cost of driving the task minus the sum of the dual costs of the covered trips.

In order to find an initial solution, a local search algorithm is used; this attempts to generate a set of vehicle tasks that covers all trips exactly once with minimum cost. This local search can be run multiple times, allowing vehicle tasks from separate runs to be combined and in order to form a good base LP solution. Afterwards, two methods are proposed in order to generate new useful columns by solving the pricing problem: a labeling algorithm and another local search. We will go into more details on all 3 mentioned techniques in the next sections.

Columns are generated in rounds until no new vehicle tasks with negative reduced costs can be found, or until a predetermined time limit is reached. In each of these rounds, a column generation method is selected, and the current dual costs π are passed in order to solve the pricing problem. Once no more improvement can be found or the time limit is reached, the constraint (22) is reinstated, making the model binary again. As early termination is included, this approach can be seen as a form of truncated column generation. In order to solve our original ILP, branch-and-bound is applied in order to find an integer solution with the columns that were generated up until that point.

4.1.2.1 Initial solution - Local search

In order to find a good set of initial vehicle tasks, a local search is applied inspired by ten Bosch et al. (2021). More specifically, a simulated annealing algorithm is used in which an initial set of vehicle task is continually modified with the hope of finding a lower cost coverage for our trips. For more details on the workings of a simulated annealing algorithm we refer to Kirkpatrick et al. (1983), however a simplified outline has been provided in Algorithm 1 for readers unfamiliar with the technique.

Algorithm 1 Generic Simulated Annealing

$sol \leftarrow InitialSolution()$
while $currIt < totalIts$ **do**
  $op \leftarrow operations.GetRandom()$
  $sol' \leftarrow op(sol)$
  if $\exp((sol.Cost - sol'.Cost)/T) > uniform(0,1)$ **then** $sol \leftarrow sol'$
  $T \leftarrow T * \alpha$; $currIts \leftarrow currIts + 1$

We initialize our set of covering vehicle tasks by introducing exactly one task for each trip. Afterwards, we will continually try to improve this set by recombining (parts of) the tasks with each other, always accepting a change if it results in better costs and accepting it with a chance proportional to the increase in costs otherwise. In order to make the modification of tasks during this process easier, it can be beneficial to include the arcs previously discarded in our short turnaround preprocessing step. By doing this, an intermediate solution of the local search may contain a large vehicle idle times which can be filled by a future operation. This is also the case for the local search that we will discuss in Section 4.1.2.3.

The following set of operations is used in order to modify the set of vehicle tasks. Of these, the first two are almost directly taken from ten Bosch et al. (2021), with the addition of support for detour arcs between pairs of trips. The last operation is also added in order to handle switching between base and detour arcs, as in our instances trips may not always pass the charging locations frequently enough in order to charge throughout the day. During the algorithm, the operations are chosen at random from a weighted distribution.

2-Opt (or tail swap) Two different vehicle tasks vt and vt' are selected at random from the available set of tasks. Next, a time t between the minimum start and maximum end time of vt and vt' is randomly selected. All trips after t in vt are moved to vt', while at the same time moving all trips after t in vt' to vt. In order to connect the tails to the parts of the tasks before t, a random arc between the preceding depot / trip and the start of the tail is selected from A_{vt}. If the schedule resulting from the swap is infeasible (either due to scheduling conflicts, non-existing arcs or SoC not respecting the given bounds), the swap is reverted. If either vt or vt' becomes empty due to the entirety of its schedule being swapped to the other task, remove the empty task from the available set. A visual representation of this operation is shown in Figure 2.

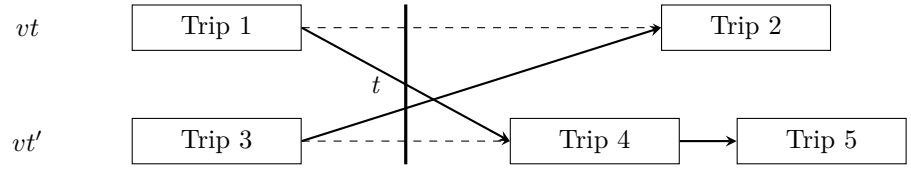

Figure 2: 2-Opt operation on two vehicle tasks vt and vt'. Trips after t (2 for vt, 4 and 5 for vt') are moved to the other task. Arrows are arcs in A_{vt}, dashed arrows between trips are replaced by solid arrows during operation.

Range move Two different vehicle tasks vt and vt' are selected at random from the available set of tasks. Next, from vt', a random continuous range of trips r is selected; r is removed from vt' and is added to vt. In order to connect r to preceding and following nodes in vt, random arcs within A_{vt} are chosen. The same is done in order to reconnect the remaining of vt'. As before, if the resulting vehicle tasks are infeasible the change is reverted. If vt' becomes empty due to the removal of r and the change is accepted, remove vt' from the available set of vehicle tasks. A visual representation of this operation is shown in Figure 3.

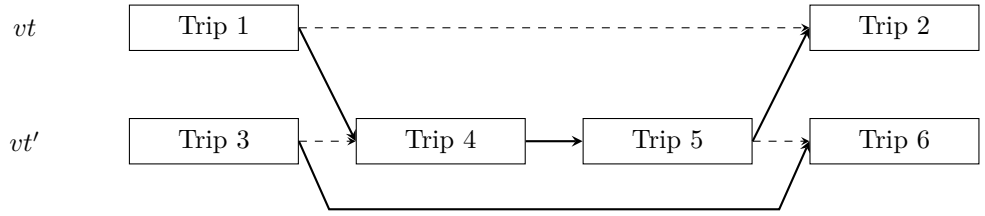

Figure 3: Range move operation on two vehicle tasks vt and vt'. A continuous range of trips (here, trip 4 and 5) is moved from vt' to vt.

Change arc A single vehicle task vt is selected at random from the available set of tasks. In this task, an arc a connecting two trips is selected for which there exist alternative connecting arcs in A_{vt}. One of these alternative arcs is selected at random in order to replace a. If this results in infeasibilities in the SoC, the change is reverted. As not all base arcs allow for charging, this operator can allow for a new charging opportunity to be added to a task; this in turn may allow for more trips to be added later. On the other hand, if a detour arc is currently selected without it being required to keep the vehicle SoC within bounds, it may be removed to save on driving costs. A visual representation of this operation is shown in Figure 4.

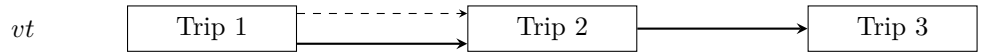

Figure 4: Change arc operation on vehicle task vt. The arc between trip 1 and trip 2 is replaced by a different arc in A_{vt}, changing the SoC of the vehicle when entering trip 2.

As the operations are randomly chosen, the resulting vehicle tasks (and corresponding quality) might differ greatly between two different runs. We therefore run the algorithm a number of times, thereafter allowing the LP relaxation to use the best available combination of columns in order to find an initial cover for the trips.

4.1.2.2 Pricing problem - Labeling

Let us consider our previously defined graph $G_{vt} = (V_{vt}, A_{vt})$. In order to solve the pricing problem, let us slightly modify the costs of the arcs within our graph such that finding a path of minimum cost from d_s to d_e corresponds with a path of minimum reduced costs. In order to do this, for each trip t, let us subtract the reduced cost π_t from the costs of each incoming arc. Now, the sum of arc costs from d_s to d_e corresponds to the reduced costs of the path, allowing us to search for a lowest cost (or shortest) feasible path from d_e to d_s.

As we have to keep our SoC within constrained bounds, this problem is a variant of a resource-constrained shortest path problem. In order to solve this, we apply a labeling algorithm. Here, starting from a source, we will propagate an achievable state (or label) through each outgoing arc; then, we select a new label and do the same, continuing until all labels have been fully propagated. A sketch of a generic labeling algorithm is outlined in Algorithm 2. In our case, for each node in $\{d_s\} \cup T$, we keep track of a set of active labels. Each of these labels contains the current costs, current SoC and the previous label. Starting in d_e with an empty label, we will propagate labels through the graph. In this, let us refer to the collection of labels that reach a certain node as being *contained* in that node.

Once all active labels have been processed, d_e will now contain a list of finalized labels. We can then find a label with lowest costs and backtrack through previous labels to d_s in order to reconstruct the entire vehicle task. If the reduced cost of this task is negative, it will improve our solution to the relaxed E-VSP when included.

Algorithm 2 Generic Labeling

```
activeLabels.Init()                                  ▷ List of node / label pairs
allLabels.Init()                                     ▷ Map from node to list of labels
while |activeLabels| > 0 do
    (label, node) ← activeLabels.Pop()
    for arc ∈ node.OutgoingArcs do
        (label', node') ← ApplyArc(label, arc)
        if Infeasible(label') then continue
        if node' ≠ endNode then
            activeLabels.Push(label', node')
        else
            allLabels[node'].Push(label')
```

As the amount of outgoing arcs may be large, propagating all labels can become computationally expensive. We therefore introduce a domination rule, allowing us to discard a label if a better one is known: for two labels l and l' on the same node v, we consider l to be dominated by l' if the costs of l' are not higher and the SoC of l' is not lower. If a label l is dominated after the introduction of a new label l', the label l is removed from the list of active labels, reducing the total number that need to be considered.

In order to make this domination rule stronger, SoC values are binned into discrete buckets. This allows for small SoC differences to be disregarded when checking for domination, allowing more labels to become dominated and further reducing computational load. During our testing, actual SoC values were rounded down to the nearest whole percentage point for domination purposes, resulting in a total of 101 buckets. Major gains can also be achieved through proper processing order of active labels. As G_{vt} is a DAG due to arcs going forward in time, a topological sorting of the nodes can be constructed. If active labels are then evaluated in this order, it is guaranteed that all domination rulings for a certain trip node will have been applied before any labels of that node are expanded. This greatly reduces the total amount of labels expanded, reducing overall runtime.

The list of finalized labels in d_e often contains more than one label with negative costs; in this case, multiple vehicle tasks with negative reduced costs can be extracted from the labeling result. In order to do this, we first sort the labels by cost in non-descending order. A maximum of $N_{VT,LB}$ labels is then selected from this sorted list. In order to encourage good coverage of all trips within a single column generation round, an attempt is first made to select labels whose trip covers are disjoint. After no more labels are available whose cover is disjoint from those already selected, we simply add labels with new trip covers until we have added $N_{VT,LB}$ in total or until all labels with unique trip covers have been added.

While this allows us to converge on a good solution for the relaxed problem quite quickly, solving the ILP with the resulting vehicle tasks might still prove challenging without significant trip overlap. In an attempt to prevent this, a secondary set of vehicle tasks is generated. In this, the top $M_{VT,LB} \leq N_{VT,LB}$ generated tasks from the initial labeling round are selected as a base. For each of these bases, we iteratively block off some of the trips from the task, starting from the front. For each set of blocked of nodes, we then rerun the labeling algorithm; this causes a new set of labels to be generated which might still use part of the base, but are forced to use alternate trips for the parts which are blocked off. This results in vehicle tasks with new covers which might not be optimal in de relaxation, however they might help during the final solve.

4.1.2.3 Pricing problem - Local search

Without binning of SoC values, the labeling algorithm provides an exact solution to the pricing problem. For larger instances however, it might not be computationally feasible to generate enough columns in order to find a good solution. As an alternative, we can approximate a solution to the pricing problem by once again using a simulated annealing algorithm. Here, we now attempt to construct a single vehicle task with minimum reduced costs. As previously discussed in section 4.1.2.1, it can be beneficial to not apply filtering of arcs in A_{vt} during preprocessing to increase the number of feasible intermediate solutions.

Starting with an empty task vt, a collection of operations is again repeatedly applied in order to modify it. After an operation is applied, the reduced cost of the new vehicle task is compared to that before the change in order to check for acceptance. The following set of operations is used in order to

construct our vehicle task:

Add trip From the set of currently unvisited trips, select a trip t at random. For both the predecessor and successor of t in vt, select a random arc between the two in A_{vt} in order to connect t with the existing parts of vt. If the resulting task is infeasible (either through scheduling conflicts, no connecting arcs existing or SoC conflicts), the change is rejected. A visual representation of this operation is shown in Figure 5.

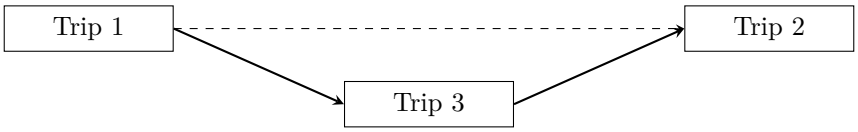

Figure 5: Add trip operation on the vehicle task. A new trip (in this example, trip 3) is added to the task.

Remove trip From the set of currently visited trips, select a trip t at random. t is then removed from vt, connecting its predecessor and successor via a random arc in A_{vt} between the two nodes. If the resulting task is infeasible, the change is rejected. A visual representation of this operation is shown in Figure 6.

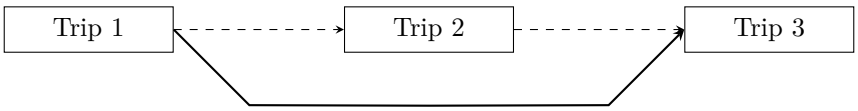

Figure 6: Remove trip operation on the vehicle task. An existing trip (in this example, trip 2) is removed from the task.

Change arc The same as the one used in the initial local search. In the vehicle task vt, an arc a is selected at random for which at least one alternative arc exist in A_{vt}. a is then replaced by a randomly chosen arc from the set of alternative arcs. If the resulting task has infeasible SoC, the change is rejected. A visual representation of this operation is shown in Figure 4.

As with the labeling algorithm, it is interesting to include a secondary set of vehicle tasks with a different covered trip set as this may help with the final solve. In order to do this, let us take $N_{VT,LS}$ samples of vt during the simulated annealing process, distributed evenly throughout the total amount of iterations performed. This results in tasks that are similar to the final result to be included in our generated set, allowing more options for inclusion in the ILP solution.

As this local search only approximates the solution to the pricing problem, there is a chance that no task with negative reduced costs is found even if one exists. In order to not terminate the search for new columns prematurely when this happens, we instead continue for a number of column generation rounds; if a new column with negative reduced cost is found within these rounds, we continue our search. If the threshold number of rounds is reached without finding a column with negative reduced cost, we terminate the search assuming that the we will not be able to improve the solution of the LP.

We also note that two different runs of this algorithm may generate different vehicle tasks, even when starting with the same dual costs. It can therefore be beneficial to run the local search multiple times in parallel during a single column generation round, allowing for a diverse set of columns to be added to the LP.

4.1.3 Postprocessing

Once an integer solution is found to the E-VSP, there may still be overlap between the selected vehicle tasks. Depending on both the columns generated and the time limit given to the final solver, this number of duplicate trips may result in significant duplication of work. In order to reduce the costs incurred in driving the selected tasks, simple postprocessing is applied in order to completely eliminate any duplicate trips.

Let us first make the assumption that for any pair of locations l and l' in $\mathcal{L}$, a corresponding deadhead $\mathcal{DH}$ exists. If this is not the case, temporarily extend $\mathcal{DH}$: a minimum duration and distance driven between l and l' using sequence of existing deadheads and/or trips can be used as a stand-in.

For each trip that is covered more than once, we can now select all tasks except the first which covers this trip. For each of these tasks, remove the duplicate trip from the task and replace it with a combination of a deadhead between the starting and ending location of the trip and idle time. If this results in multiple deadheads occurring without a previously planned charging operation or trip between them, combine the deadheads into one by going directly from the starting location of the first deadhead to the ending location of the last deadhead, padding with idle time where needed. This will ensure that no more SoC is used and distance is driven than in the original task, and in most cases less crew time is now required in order to complete the task.

4.1.4 Crew considerations

Creation of vehicle tasks as described, while minimizing vehicle related costs, may not always result in a task which can be executed in real life. Crew members, as discussed in more detail in both Section 2 and 4.2, are subject to a different set of constraints than the vehicles themselves. Most notably, crew members under most common labor regulations have a limit on continuous driving time. If a vehicle task contains a set of continuous trips without break or handover time which is longer than this limit, this immediately implies that it contains a block which no crew member is ever allowed to drive; the vehicle task itself can therefore be considered infeasible.

Another problem arises when considering crew signing on and off for the day. In most real life instances, the number of crew bases in an area is limited. Depending on the instance, a vehicle purely minimizing driving costs may not visit these places often enough for crew members to get on and off throughout the day, also resulting in blocks that can never be driven.

These two problems can be resolved in one of two ways: reevaluating vehicle tasks before entering the crew scheduling phase, or including some crew constraints into the feasibility of a vehicle task. The first option is known to have been used in commercial public transport planning software such as Hastus (as discussed by Blais et al. (1990)), where blocks are generated separately from the vehicle route throughout the day. We however selected the latter of the two options by adding constraints on the amount of continuous driving time and time away from a crew base to the vehicle task feasibility.

In the labeling algorithm, these values are simply included in the label; they are not considered for domination rules, however they are included in the feasibility check. In both local searches, the maximum times are considered a hard constraint: going over either maximum time will revert the chosen operation.

4.2 CSP

For the CSP, a similar approach is taken to that of the E-VSP: column generation in order to solve a set covering ILP. The set to be covered is now the blocks present in the selected vehicle tasks. As in Section 4.1, this process is broken up into steps:

- *Preprocessing*, in which vehicle tasks are transformed into a set of blocks, and these blocks are in turn used to generate a graph.
- *Solving*, in which truncated column generation using two sources is applied in order to solve the problem of finding a set of crew duties which cover all blocks.

As the constraints on crew duties are significantly more complex than those of a vehicle task, we now give a short summary of the most relevant ones. For a more detailed overview and definitions of terminology we refer back to Section 2.

All duties consist of blocks, idle time and break time; the maximum continuous time without a break is 4 hours. Depending on the overall duration of a duty, differing amount of break times are required. Each duty is of a certain type; this type determines acceptable start times, end times, and overall duration. A duty of type *broken* includes a single long idle, in which a crew member signs off at a crew base and then signs on again. Lastly, for a overall set of duties used to cover all blocks, constraints are in place for the percentage of long duties (> 8.5 hours), *broken* duties, *between* duties and average duty length.

4.2.1 Preprocessing

In order to assign crew members to vehicles, the previously generated vehicle tasks must first be broken up into blocks. As defined previously in Section 2, a block is part of a vehicle task which must be driven

by a single driver; there is no possibility of handover between drivers during the block. Let VT be the collection of vehicle tasks that was previously selected. For each vehicle task $vt \in VT$, let B_{vt} be the set of blocks that are contained in vt. The total collection of to be covered blocks can then be defined as the union of all B_{vt}.

Next, let us define a graph $G_{cd} = (V_{cd}, A_{cd})$ in which paths will represent crew duties. Let $V_{cd} = \{cb_s, cb_e\} \cup B$, where cb_s and cb_e represent an arbitrary crew base at the start and end of a duty respectively. Additionally, as with trips in the E-VSP, let $s(b)$ and $e(b)$ represent the start and end time of block b, and let $o(b)$ and $d(b)$ represent the origin and destination location respectively. For every block $b \in B$ for which $o(b)$ is a crew base, add a *sign-on arc* from cb_s to b. If $d(b)$ is a crew base, add a *sign-off arc* between b and cb_e. Next, between each pair of blocks b and b', attempt to add an arc if $d(b) = o(b')$ and $e(b) \leq s(b')$. The type of the arc depends on $d(b)$:

- If $d(b)$ has a place for crew members to rest and the amount of idle time is a feasible to be a break, add a *break arc*.
- Otherwise, if $d(b)$ is a crew base and $s(b') - e(b)$ is greater than some threshold, add a *long idle arc*. Here, the crew member signs off at the beginning of the idle time, before signing on to drive the next block. We note that given the labor regulations discussed in Section 2, this arc is only allowed to be used in a duty of type *broken*.
- Otherwise, add a *short idle arc*, in which the crew member remains in their vehicle.

This definition of our arcs implies that at most one arc is present between any two blocks. The number of arcs within our graph may however still be quite large; in order to reduce this, we may apply some domain knowledge in order to filter out arcs which are not part of a good solution. We will do this by enforcing minimum and maximum durations of each arc:

- Sign-on/off: Any
- Break arcs: [15min, 60min]
- Long idle arcs: [1.5h, 5h]
- Short idle arcs: [0min, 30min]

In this, the minimum duration of the break and long idle arcs is also required by the labor regulations. We note that there are gaps within the provided thresholds; as an example, an arc of 75 minutes can never be included. While this does significantly limit our amount of feasible crew duties, the given bounds are realistic; forcing a crew member to take a break which is longer than an hour will rarely occur in practice.

Any feasible crew duty can now be represented as a path between cb_s and cb_e in G_{cd}. We again note that not every path from cb_s to cb_e represents a feasible duty; based on the chosen duty type, differing combinations of duty length, break times and idle times might be feasible.

4.2.2 Solving

Similarly to our model for the E-VSP, let us define a set covering ILP in order to find a set of feasible crew duties which cover all to be driven blocks. In this, let CD be the available crew duties, let y_j be a binary decision variable indicating the usage of $cd_j \in CD$, and let $m_{j,b}$ be a parameter indicating that crew duty cd_j covers block $b \in B$. In order to model the constraints on the overall set of duties, let us introduce an additional set of parameters. Let d_j be the total working duration of duty cd_j, let $d_{j,long}$ indicate that d_j is longer than 8.5 hours, and let $\rho_{j,broken}$ and $\rho_{j,between}$ indicate that cd_j is of duty type *broken* or *between* respectively. We can then define our ILP as follows:

$$\min \quad \sum_{cd_j \in CD} y_j \cdot cost(cd_j) \tag{25}$$

Subject to:

$$\sum_{cd_j \in CD} y_j m_{j,b} \geq 1 \qquad \forall b \in B \tag{26}$$

$$\sum_{cd_j \in CD} y_j \cdot (\frac{d_j}{8\ hours} - 1) \leq 0 \tag{27}$$

$$\sum_{cd_j \in CD} y_j \cdot (0.15 - d_{j,long}) \geq 0 \tag{28}$$

$$\sum_{cd_j \in CD} y_j \cdot (0.3 - \rho_{j,broken}) \geq 0 \tag{29}$$

$$\sum_{cd_j \in CD} y_j \cdot (0.1 - \rho_{j,between}) \geq 0 \tag{30}$$

$$y_j \in \{0,1\} \qquad \forall cd_j \in CD \tag{31}$$

Here, constraint (26) ensures that all blocks are covered, constraint (27) ensures a maximum average duty length, constraint (28) ensures that a max of 15% of all selected shifts is long, and constraints (29) and (30) ensure that a maximum of 30% of selected duties are of type *broken* and 10% are of type *between* respectively. This problem can be relaxed to an LP by replacing constraint (31) with the following:

$$0 \leq y_j \leq 1 \qquad \forall cd_j \in CD \tag{32}$$

In this relaxed problem, let the dual cost of constraints (26-30) be $\tau_{cover,b}, \tau_{avgdur}, \tau_{longdur}, \tau_{broken}$ and $\tau_{between}$ respectively. Using this, we can define the reduced costs of a new crew duty cd_0 as being:

$$\begin{aligned} \text{cost}(cd_0) - \sum_{b \in B} m_{0,b}\tau_{cover,b} - \frac{d_0}{8 \text{ hours}}\tau_{avgdur} \\ - \tau_{longdur}(0.15 - d_{0,longdur}) - \tau_{broken}(0.3 - \rho_{0,broken}) - \tau_{between}(0.1 - \rho_{0,between}) \end{aligned} \tag{33}$$

An initial set of columns for the LP relaxation can once again be found by solving a number of local searches, each of which attempts to cover the all blocks. After this initial set, the pricing problem is again repeatedly solved using a labeling algorithm; this is repeated until the relaxed problem is solved to optimality or until a predefined time limit is hit. Afterwards, constraint (31) is reinstated, before finding an integer solution using branch and bound.

4.2.2.1 Initial solution - Local search

As with the E-VSP, an initial set of crew duties can be constructed using a simulated annealing algorithm. In this, let us start with a set of crew duties each containing exactly one block. As crew duties must have a duty type, a type which is feasible given the start and end time of the block is randomly selected. We note that this initial solution may not be feasible as a whole, as the start and end locations of blocks are not required to be a crew base. In order to deal with this, we apply a penalty to each duty which does not start and end at a crew base. Additionally, constraints (27-30) are each relaxed by applying a penalty if the overall set of duties does not satisfy the constraint. This penalty is based on a percentage difference between the target and value in the duty set.

Next, the following set of operations is applied in order to recombine the duties. We note that the operators used in this local search are almost identical to those described earlier in Section 4.1.2.1. The key difference is the use of blocks as the main elements within a duty as opposed to trips for a vehicle task. We will therefore refer back to the illustration provided earlier for a visual representation of the operations, substituting trips for blocks and tasks for duties where applicable. Similarly to the E-VSP local searches, it can also be beneficial to include the unfiltered arcs during the search in order to allow for longer breaks/idle time, thereby increasing the solution space.

2-Opt (or tail swap) Two different duties cd and cd' are selected at random from the available set of duties. Next, a time t between the minimum start and maximum end time of cd and cd' is selected. All blocks after t in cd are moved to cd', while at the same time moving all blocks after t in cd' to cd. If no arcs in A_{cd} exist to reconnect the swapped tails of the duties, the change is rejected. If the resulting schedules contain a different infeasibility (such as a start or end time which conflicts with the current

duty type), an additional check is done to see if changing a duty type might make the change feasible. If so, the duty type is adjusted to one which is feasible, prioritizing duty types which are not constrained. If this is not possible, the change is rejected. If one of the duties becomes empty due to the operation, it is removed from the set of available duties. A visual representation can be found in Figure 2.

Range move Two different duties cd and cd' are selected at random from the available set of duties. Next, from cd', a random continuous range of blocks r is selected; r is removed from cd' and is added to cd. If no arcs exist to connect r in cd or remove r from cd', the change is rejected. Next, as with 2-Opt, an attempt is made to make the duties feasible by changing the types; if this fails, the change is rejected. If cd' becomes empty due to the operation, it is removed from the set of available duties. A visual representation can be found in Figure 3.

Single block move Same as the range move, however we now only select a single block from cd'. This is done as blocks can be quite long, therefore moving a range might be difficult if it contains a single long block.

After termination, the generated duties may still be infeasible if the soft constraints are still violated. This can either be because the set of generated duties does not match the duty type / duration requirements, or because some of the duties do not properly start or end at a crew base. In order to resolve both of these problems, a set of dummy columns is added to the LP formulation: each covers exactly one block without any constraints, however the use of one of these columns is discouraged by giving it significant costs. This still allows for a solution to the LP relaxation to be found, allowing for dual costs to be determined. This in turn allows the column generation phase to resolve the remaining uncovered blocks.

4.2.2.2 Pricing problem - Labeling

Let us consider our previously defined graph $G_{cd} = (V_{cd}, A_{cd})$. As with the labeling algorithm described in the E-VSP section, we can once again attempt to find a path of minimum cost through our graph G_{cd} in order to generate a crew duty with minimum reduced cost. In order to do this, we once again modify the costs of the arcs in A_{cd} in order to include the dual costs, adding the dual cost for duty length and type during the evaluation of cb_e. For a more detailed overview of the algorithm, we refer to Section 4.1.2.2, however we will discuss details specific to label propagation in the crew duty variant here.

We now start with a set empty labels in cb_s, one for each available duty type. During propagation, in each label keep track of:

- The duty type of the label.
- The previous label.
- The starting time of the duty, defined by the first duty after cb_s.
- The breaks arcs taken in this duty.
- The long idle arcs taken in this duty.

Clear domination rules are not present, as the accrued reduced costs are only dominated if start time, breaks and idles are identical. As this is unlikely, removing as many labels as possible through feasibility constraints remains as the best option to ensure that computational efficiency is maintained.

Checking for feasibility during propagation can however only be done for a subset of the provided labor regulations. Properties such as overall duty duration, continuous driving time, minimum start and maximum end time, and use of long idle arcs can already be used in order to remove labels which cannot result in a feasible duty of a given type. Before reaching cb_e however, the end time and distribution of breaks in the duty are not yet finalized. This notably implies that it is not possible to check for sufficient breaks while propagation is still in progress. As an example, suppose some label in the process indicates that its corresponding duty will have a length of at least 6 hours, requiring a total of 40 minutes of break time under Dutch labor regulations. If the current amount of break time is 20 minutes, this may still become feasible before reaching cb_e by taking another break arc; we can therefore not discard this label before we know when the duty will end.

Once propagation has completed, the duties described by the labels in cb_e can be finalized. Only then can we do our full feasibility check, discarding any labels which are not valid given their state in cb_e. As

with the E-VSP, what remains is a list of labels which describe feasible duties which can be sorted on non-decreasing reduced cost. We will then take at most $N_{CD,LB}$ of these duties with negative reduced cost, prioritizing block-disjoint duties. Next, secondary columns are generated for the best $M_{CD,LB} \leq N_{CD,LB}$ of the found duties. After this, the complete set of duties is made available to the LP relaxation, before repeating the process with new values of τ.

5 Results

.

References

Barnhart, C., A. M. Cohn, E. L. Johnson, D. Klabjan, G. L. Nemhauser and P. H. Vance (2003). 'Airline Crew Scheduling'. In: *Handbook of Transportation Science*. Ed. by R. W. Hall. Boston, MA: Springer US, pp. 517–560. ISBN: 978-0-306-48058-4. DOI: https://doi.org/10.1007/0-306-48058-1_14. URL: https://doi.org/10.1007/0-306-48058-1_14.

Beasley, J. E. (1993). 'Lagrangian relaxation'. In: *Modern Heuristic Techniques for Combinatorial Problems*. USA: John Wiley & Sons, Inc., 243–303. ISBN: 0470220791.

Blais, J.-Y., J. Lamont and J.-M. Rousseau (1990). 'The HASTUS Vehicle and Manpower Scheduling System at the Société de transport de la Communauté urbaine de Montréal'. In: *Interfaces* 20.1, pp. 26–42. ISSN: 00922102, 1526551X. URL: http://www.jstor.org/stable/25061310 (visited on 10/07/2025).

Bodin, L. and B. Golden (1983). 'Routing and scheduling of vehicles and crews: The state of the art'. In: *Computers & Operations Research* https://doi.org/10.2. Routing and Scheduling of Vehicles and Crews. The State of the Art, pp. 63–211. ISSN: 0305-0548. DOI: https://doi.org/10.1016/0305-0548(83)90030-8.

Boland, N., M. Hewitt, L. Marshall and M. Savelsbergh (2017). 'The Continuous-Time Service Network Design Problem'. In: *Operations Research* 65.5, pp. 1303–1321. DOI: https://doi.org/10.1287/opre.2017.1624.

Bunte, S. and N. Kliewer (2009). 'An overview on vehicle scheduling models'. In: *Public Transport* 1.4, pp. 299–317. ISSN: 1613-7159. DOI: https://doi.org/10.1007/s12469-010-0018-5.

Ceder, A. and N. H. Wilson (1986). 'Bus network design'. In: *Transportation Research Part B: Methodological* 20.4, pp. 331–344. ISSN: 0191-2615. DOI: https://doi.org/10.1016/0191-2615(86)90047-0.

Cong, Y., Y. Bie, Z. Liu and A. Zhu (2024). 'Collaborative vehicle-crew scheduling for multiple routes with a mixed fleet of electric and fuel buses'. In: *Energy* 298, p. 131400. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2024.131400.

De Vos, M. H., R. N. van Lieshout and T. Dollevoet (2024). 'Electric Vehicle Scheduling in Public Transit with Capacitated Charging Stations'. In: *Transportation Science* 58.2, pp. 279–294. DOI: https://doi.org/10.1287/trsc.2022.0253.

European regulations (2018). *Regulation - 2018/1999 - EN - EUR-Lex — eur-lex.europa.eu*. Accessed 08-12-2024. URL: https://eur-lex.europa.eu/eli/reg/2018/1999/oj.

Even, S., A. Itai and A. Shamir (1975). 'On the complexity of time table and multi-commodity flow problems'. In: *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pp. 184–193. DOI: https://doi.org/10.1109/SFCS.1975.21.

Fischetti, M., S. Martello and P. Toth (1989). 'The Fixed Job Schedule Problem with Working-Time Constraints'. In: *Operations Research* 37.3, pp. 395–403. URL: http://www.jstor.org/stable/171059.

Friberg, C. and K. Haase (1996). 'An exact algorithm for the vehicle and crew scheduling problem'. In: 416. URL: https://ideas.repec.org/p/zbw/cauman/416.html.

Ge, L., N. Kliewer, A. Nourmohammadzadeh, S. Voß and L. Xie (2024). 'Revisiting the richness of integrated vehicle and crew scheduling'. In: *Public Transport* 16.3, pp. 775–801. ISSN: 1613-7159. DOI: https://doi.org/10.1007/s12469-022-00292-6.

Haghani, A. and M. Banihashemi (2002). 'Heuristic approaches for solving large-scale bus transit vehicle scheduling problem with route time constraints'. In: *Transportation Research Part A: Policy and Practice* 36.4, pp. 309–333. ISSN: 0965-8564. DOI: https://doi.org/10.1016/S0965-8564(01)00004-0.

Ham, A. and M.-J. Park (2021). 'Electric Vehicle Route Optimization Under Time-of-Use Electricity Pricing'. In: *IEEE Access* 9, pp. 37220–37228. DOI: https://doi.org/10.1109/ACCESS.2021.3063316.

Huisman, D., R. Freling and A. P. M. Wagelmans (2005). 'Multiple-Depot Integrated Vehicle and Crew Scheduling'. In: *Transportation Science* 39.4, pp. 491–502. DOI: https://doi.org/10.1287/trsc.1040.0104.

Ibarra-Rojas, O., F. Delgado, R. Giesen and J. Muñoz (2015). 'Planning, operation, and control of bus transport systems: A literature review'. In: *Transportation Research Part B: Methodological* 77, pp. 38–75. ISSN: 0191-2615. DOI: https://doi.org/10.1016/j.trb.2015.03.002.

Kirkpatrick, S., C. D. Gelatt Jr and M. P. Vecchi (1983). 'Optimization by simulated annealing'. In: *science* 220.4598, pp. 671–680.

Li, J.-Q. (2014). 'Transit Bus Scheduling with Limited Energy'. In: *Transportation Science* 48.4, pp. 521–539. ISSN: 00411655, 15265447. URL: http://www.jstor.org/stable/43666940 (visited on 20/12/2024).

Olsen, N. and N. Kliewer (2020). 'Scheduling electric buses in public transport: Modeling of the charging process and analysis of assumptions'. In: *Logistics Research* 13.1, pp. 1–17.

Parmentier, A., R. Martinelli and T. Vidal (2023). 'Electric Vehicle Fleets: Scalable Route and Recharge Scheduling Through Column Generation'. In: *Transportation Science* 57.3, pp. 631–646. DOI: https://doi.org/10.1287/trsc.2023.1199.

Pepin, A.-S., G. Desaulniers, A. Hertz and D. Huisman (2009). 'A comparison of five heuristics for the multiple depot vehicle scheduling problem'. In: *Journal of Scheduling* 12.1, pp. 17–30. ISSN: https://doi.org/1099-1425. DOI: https://doi.org/10.1007/s10951-008-0072-x.

Perumal, S. S., T. Dollevoet, D. Huisman, R. M. Lusby, J. Larsen and M. Riis (2021). 'Solution approaches for integrated vehicle and crew scheduling with electric buses'. In: *Computers & Operations Research* 132, https://doi.org/105268. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2021.105268.

Perumal, S. S., J. Larsen, R. M. Lusby, M. Riis and K. S. Sørensen (2019). 'A matheuristic for the driver scheduling problem with staff cars'. In: *European Journal of Operational Research* 275.1, pp. 280–294. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2018.11.011.

Perumal, S. S., R. M. Lusby and J. Larsen (2022). 'Electric bus planning & scheduling: A review of related problems and methodologies'. In: *European Journal of Operational Research* 301.2, pp. 395–413. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2021.10.058.

Qbuzz (2024). *Busvervoer 90%(!) uitstootvrij vanaf 2024*. Accessed 08-12-2024. URL: https://www.qbuzz.nl/gd/actueel/laatste-nieuws/ZBnQDREAACEAlo3q/busvervoer-90-uitstootvrij-vanaf-2024.

RDW (2025). *Opendata*. Accessed 20-01-2025. URL: https://opendata.rdw.nl/.

Sassi, O. and A. Oulamara (Mar. 2014). 'Electric Vehicle Scheduling and Optimal Charging Problem: Complexity, Exact and Heuristic Approaches'. In: *International Journal of Production Research* 55. DOI: https://doi.org/10.1080/00207543.2016.1192695.

Shen, Y. and Y. Li (Nov. 2023). 'Minimum Cost Flow-Based Integrated Model for Electric Vehicle and Crew Scheduling'. In: *Journal of Advanced Transportation* 2023, pp. 1–23. DOI: https://doi.org/10.1155/2023/6658030.

Sistig, H. M. and D. U. Sauer (2023). 'Metaheuristic for the integrated electric vehicle and crew scheduling problem'. In: *Applied Energy* 339, p. 120915. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2023.120915.

Stadnichuk, V., J. Segschneider, A. M. Koster and G. Walther (2024). *Integrated Optimization of Timetabling and Electric Vehicle Scheduling: A Case Study of Aachen, Germany*. https://optimization-online.org/2024/09/integrated-optimization-of-timetabling-and-electric-vehicle-scheduling-a-case-study-of-aachen-germany/. [Accessed 13-01-2025].

Ten Bosch, W., J. A. Hoogeveen and M. E. van Kooten Niekerk (2021). 'Scheduling electric vehicles by Simulated Annealing with recombination through ILP'. In: Pending publication.

Van Kooten Niekerk, M. E., J. M. van den Akker and J. A. Hoogeveen (2017). 'Scheduling electric vehicles'. In: *Public Transport* 9.1, pp. 155–176. ISSN: 1613-7159. DOI: https://doi.org/10.1007/s12469-017-0164-0.

Wang, J., H. Wang, A. Chang and C. Song (2022). 'Collaborative Optimization of Vehicle and Crew Scheduling for a Mixed Fleet with Electric and Conventional Buses'. In: *Sustainability* 14.6. ISSN: 2071-1050. DOI: https://doi.org/10.3390/su14063627.

Wen, M., E. Linde, S. Ropke, P. Mirchandani and A. Larsen (2016). 'An adaptive large neighborhood search heuristic for the Electric Vehicle Scheduling Problem'. In: *Computers & Operations Research* 76, pp. 73–83. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2016.06.013.

Wen, X., X. Sun, Y. Sun and X. Yue (2021). 'Airline crew scheduling: Models and algorithms'. In: *Transportation Research Part E: Logistics and Transportation Review* 149, p. 102304. ISSN: 1366-5545. DOI: https://doi.org/10.1016/j.tre.2021.102304.

Zhang, L., S. Wang and X. Qu (2021). 'Optimal electric bus fleet scheduling considering battery degradation and non-linear charging profile'. In: *Transportation Research Part E: Logistics and Transportation Review* 154, https://doi.org/102445. ISSN: 1366-5545. DOI: https://doi.org/10.1016/j.tre.2021.102445.

Zhou, Y., H. Wang, Y. Wang, B. Yu and T. Tang (2024). 'Charging facility planning and scheduling problems for battery electric bus systems: A comprehensive review'. In: *Transportation Research Part E: Logistics and Transportation Review* 183, p. 103463. ISSN: 1366-5545. DOI: https://doi.org/10.1016/j.tre.2024.103463.