Brandenburg
University of Technology
Cottbus - Senftenberg

A 2D Layered Graph Approach for Scheduling Delivery Robots

Fabian Gnegel
Stefan Schaudt
Uwe Clausen
Armin Fügenschuh

Cottbus Mathematical Preprints
COMP # 17(2021)

A 2D Layered Graph Approach for Scheduling Delivery Robots

Fabian Gnegel*†, Stefan Schaudt‡, Uwe Clausen‡, Armin Fügenschuh†

April 2021

Abstract

In recent years parcel volumes reached record highs. The logistics industry is seeking new innovative concepts to keep pace. For densely populated areas delivery robots are a promising alternative to conventional trucking. These electric robots drive autonomously on sidewalks and deliver urgent goods, such as express parcels, medicine, or meals. The limited cargo space and battery capacity of these vehicles necessitates a depot visit after each customer served. The problem can be formulated as an electric vehicle routing problem with soft time windows and a single unit capacity. The goal is to serve all customers such that the quadratic sum of delays is minimized and each vehicle operates within its battery bounds. To solve this problem, we formulate an MIQP and present an expanded formulation based on a layered graph. For this layered graph we derive two solution approaches based on relaxations, which use less nodes and arcs. The first, Iterative Refinement, always solves the current relaxation to optimality and refines the graph if the solution is not feasible for the expanded formulation. This is repeated until a proven optimal solution is found. The second, Branch and Refine, integrates the graph refinement into a branch and bound framework avoiding restarts. Computational experiments performed on modified Solomon instances demonstrate the advantage of using our solution approaches and show that Branch and Refine outperforms Iterative Refinement in all studied parameter configurations.

Keywords: delivery robots; electric vehicle routing problem; exact algorithm; graph refinement; last-mile; layered graphs; partial recharging

1 Introduction

In the last decade, e-commerce has changed consumer markets world-wide. The global e-commerce revenue reached 2,415 billion US$ in 2020, which is an increase of 25% compared to 2019 [19]. The growing e-commerce market increases the number of parcels to be delivered. In 2019, the global amount of parcels surpassed 100 billion [14]. This growth is adding to the already existing traffic challenges of congestion, noise, and air pollution. Authorities in all over Western Europe try to regulate emissions of transportation and some cities already established pedestrian-only, low-, or zero-emissions zones. These regulations effect the way people get around, how to order goods, and how logistics companies deliver. Due to these regulations and to create a better reputation, logistics companies started

*Corresponding author
†Brandenburg University of Technology Cottbus-Senftenberg, Platz der Deutschen Einheit 1, 03046 Cottbus, Germany, {gnegel,fuegenschuh}@b-tu.de
‡Institute of Transport Logistics, TU Dortmund University, Leonhard-Euler-Str. 2, 44263 Dortmund, Germany, {schaudt,clausen}@tu-dortmund.de

environmental protection programs to increase their carbon efficiency and reduce local air pollution emissions. Some of them already committed to be net zero carbon at some point in the future. To reach these goals, the logistics industry is seeking new efficient ways for transportation with a focus on the last-mile. The last-mile is the most challenging and at the same time most costly part in transportation. It is estimated that 50% of the total transportation costs are incurred in the last-mile [13]. One promising concept for good distribution in urban areas is the use of a two-tiered approach. In this structure, the goods are first transported in bundles into the city by traditional trucking and temporarily stored at decentralized micro-depots. The delivery from these micro-depots is than performed with environmental friendly solutions. One option is to use electrically-assisted freight bicycles. The distribution with these bicycles is flexible and can reduce traffic, noise, and air pollution [16]. However, it is weather dependent and the capacity is limited. Another concept is the use of small autonomous parcel robots for transportation on the last-mile. These robots are equipped with the same technology used for autonomous vehicles. The main differences to self-driving cars is in size and speed. Many startups and established companies started building delivery robots in recent years such as Starship Technologies, Nuro, Amazon, JD.com, or Alibaba. These electric robots are designed to travel on sidewalks or pedestrians zones at walking speed. Compared to conventional trucking, delivery robots over a high level of service. A customer can select a time-slot in advance at which the parcel should be delivered. Delivery robots are particularly suitable for the delivery of goods that are small in size and time-critical. The capacity and range is designed to operate in the urban environment. These robots can deliver almost everything that fits inside their compartment. Some robots are equipped with heating or refrigeration. Fields of application are the delivery of parcels, food, clothes, groceries, or medicine. To get a better understanding of delivery concepts with robots, we give a exemplary description of such a concept. Assume some customer orders a good. In a first step, this good is transported to a micro-depot, which is close to the customers location. At this micro-depot a fleet of robots is located. To offer a high level of service, the customer is able to select a delivery time-slot. To deliver the good, a robot is loaded at the micro-depot and starts its journey to the customer. After arriving at its destination, the robot is unlocked, unloaded and drives back to its micro-depot. Back at the micro-depot, the robot is either recharged or prepared for the next delivery.

In this publication we study the optimal routing, scheduling and charging strategy for a fixed number of battery powered vehicles with single unit capacity. The optimization goal is to minimize the sum of the squared differences between the start of service at the customers and their desired delivery times, i.e. the delays. Our formulation of the problem is based on the classical vehicle routing problem (VRP), but it incorporates additional variables and constraints, which are necessary to model arrival times, states of charge (SoC), and delays. Since the delays are squared in the objective, this formulation is a mixed-integer quadratic program (MIQP). As we will demonstrate in the computational approach, using state-of-the-art MIQP solvers is only a suitable solution approach for relatively few vehicles or customers. As an alternative approach we derive a formulation based on a layered graph, in which nodes not only represent customers, but a combination of customer ID, arrival time and SoC at arrival. This formulation is of theoretical nature only, since applying it in practise might be as hard as solving the problem itself, but it allows for very strong relaxations. These relaxations are relatively easy to solve and we derive an iterative approach that refines the relaxations until its solution can be converted to a solution of the original problem with equal cost, proving its optimality. For using these relaxations we propose two algorithms: Iterative Refinement in which the relaxations are refined in an outer loop and Branch and Refine in which the relaxations are refined during the exploration phase of a branch and bound algorithm.

The outline of this paper is as follows: In Section 2, we give an overview of existing literature in the field of last-mile logistics and on refinement algorithms. Section 3 gives a mathematical problem formulation and different mixed-integer formulations. Besides, the concept of a time- and battery-expanded graphs is presented. In Section 4, we discuss relaxation techniques of a completely time- and battery-expanded graph. The theoretical background is used to define a refinement algorithm in Section 5. In Section 6, the results of computational experiments are presented. Finally, Section 7 concludes and provides an outlook on future research.

2 Literature review

The delivery of parcels with innovative concepts such as drones or delivery robots is a relatively new topic. A recent overview on current and future concepts is presented by Boysen et al. [6] and an overview on trends in transportation is given by Speranza [27].

The literature concerning the optimization of a last-mile network with delivery robots is relatively small and can be divided into two categories. Firstly, literature that involves a mobile van capable of transporting robots and secondly, literature that considers stationary micro-depots. For the first category Boysen et al. [5] consider the scheduling of a delivery truck with robots on board. The truck can load robots at micro-depots and launches them at predefined locations along the route. The goal of this problem is to minimize the weighted number of late deliveries. The authors present an MILP formulation and a multi-start local search heuristic to solve the resulting scheduling problem. A similar problem is considered by Ostermeier et al. [18] with some differences in the objective function. The authors consider a multi-objective function involving the minimization of the tour length of the truck, the distances travelled by the robots, and the lateness of the deliveries. To solve this optimization problem, an MILP formulation and a heuristic that solves this problem in two stages is presented. For the second category, where micro-depot locations are fixed, Poeting et al. [21] present a simulation framework that depicts the parcel delivery of a parcel delivery company for a city with 1 Million inhabitants. Up to 3 % of the parcels are delivered with delivery robots in a two-tiered system and the rest with conventional delivery vehicles. In a follow-up publication Poeting et al. [20] investigate the number of delivery robots needed to supply a city center with parcels. Two different delivery slot selections are considered. In the first case, each customer has selected a delivery time slot in advance and delays are minimized with a simulated annealing approach. In the latter case, customers trigger their delivery on-demand and in such an event, an available robot is loaded at a micro-depot that is close to the customer location. In a contribution given by Sonneberg et al. [26] a location routing problem for delivery robots is considered. The authors try to minimize the delivery costs of urban shipments with parcel robots. The problem is modeled as an MILP and solved with an optimization software. In a case study the effect of a varying number of compartments is examined.

The optimization of unmanned aerial vehicles (UAV) or drones is a wider discussed topic in the literature. An overview on routing problems with UAVs is presented by Viloria et al. [28]. However, only a few publications consider the battery management and recharging processes. In most publications, where batteries are mentioned, they only function as a range limitation and recharging processes are assumed to be performed instantaneously. This process is represented by a battery swap at the micro-depot, where the empty battery is exchanged for a fully charged battery (see [4, 22, 30]).

A field of research, where recharging processes are modelled, is the Electric Vehicle Routing Problem (EVRP). This problem is an extension of the standard VRP to electric vehicles requiring recharging stations. A survey on the EVRP is presented by Erdelić and Carić [9]. Schneider et al. [24] extend the EVRP by time windows (EVRPTW) at the

customer locations. The authors formulate an MILP and present a Variable Neighborhood and Tabu Search heuristic to solve the problem. However, this contribution only considers full recharges. Keskin and Çatay [15] extend the EVRPTW and allow partial recharges. In their publication, an MILP formulation and an Adaptive Large Neighborhood Search heuristic is presented. An exact algorithm for the EVRPTW is presented by Desaulniers et al. [8]. The authors discuss variants of this problem with either fully or partial recharging strategies and present a Branch-Price and Cut algorithm to solve it.

The problem considered in this publication is closely related to the EVRPTW, but differs in two major aspects. Firstly, we relax the time window constraints and allow late deliveries, which are penalized. A reason for this objective function is the autonomy of the delivery vehicles, as the operational costs do not depend significantly on the operating time. The selected objective function puts a higher focus on the customer satisfaction. Another difference is the limited cargo space of autonomous delivery vehicles. This requires to model the problem with a single unit capacity and makes a depot/recharging station visit mandatory between each pair of customers.

Our modeling approach is based on layered graphs. This idea has recently been gaining traction in the literature and has been applied to a variety of problems. The nodes of a layered graph usually are derived from the set of nodes of some graph. For each original node there is a set of nodes in the layered graph, which are referred to as copies of the node. Each copy of a node represents a different state of some commodity (or some commodities) at the original node and arcs between copies model a feasible transition between those states. This allows to incorporate constraints on the commodities into the structure of the graph. Gouveia et al. [11] provide a survey on models derived from layered graphs and solution approaches for those. Maybe the most commonly used layered graph is the well-known time-expanded graph or time expanded-network, in which the nodes of a graph which usually represent locations are expanded into sets of copies which represent arrival times. Refinement algorithms have been proposed for a variety of problems in the literature, which can be modeled by time-expanded graphs. Boland and Savelsbergh [3] give a perspective on the opportunities these kinds of algorithms provide for solving time-dependent problems. The general concept here is to find partially expanded graphs, i.e., graphs which have less nodes and arcs than necessary to guarantee a correct problem formulation. Depending on whether travel times are under- or overestimated, these graphs can provide either lower or upper bounds. They are then dynamically refined to improve the bounds and close the gap between them. Boland et al. [2] successfully apply this strategy to a delivery problem, in which shipments can be consolidated if they share the same route and departure time. Other examples are Vu et al. [29] who apply it to a time-dependent variant of the traveling salesman problem, He et al. [12] who use it to solve the minimum duration shortest path problem, and Riedler et al. [23], who in their approach for solving for the traveling salesman problem with time windows also carry out refinement steps based on the linear relaxation of an MILP formulation. As an example of an application to a layered graph with nodes that do not represent arrival times, we mention here Gnegel et al. [10], who use refinement techniques in a graph whose copies represent the number of arrivals to the location. The works of Bärmann et al. [1] and Clautiaux et al. [7] show that this technique is not restricted to formulations derived from layered graphs, but can be generally applied to problems were node contractions can be used to obtain relaxed formulations. Branch and Refine is actually strongly related to one of the approaches presented by Bärmann et al. [1]. They are, to our knowledge, the first who incorporate the graph refinement into a branch and bound algorithm and thereby avoid solving MILPs in a loop to obtain better and better bounds.

3 Problem description and mathematical formulation

In the following, we present a detailed problem description of the time- and battery-constrained vehicle routing problem (TB-VRP). Let $V = \{0, \dots, n, n+1\}$ denote a set of locations, where location 0 and $n+1$ are copies of the depot and set $C = \{1, \dots, n\}$ denotes the locations of the customers. Additionally we define the following set of ordered pairs of locations:

$$A = \{(i,j) \in C \times C \mid i \neq j\} \cup \{(0,i) \mid i \in C\} \cup \{(i, n+1) \mid i \in C\}.$$

There are m identical vehicles available at the depot. Each vehicle has a single unit capacity and each customer has a single unit demand. Thus, a depot visit is required between each two consecutively visited customers.

Each location $i \in V$ has a soft time window $[\underline{t}_i, \bar{t}_i]$, where $\underline{t}_i \in \mathbb{R}^+$ denotes the lower bound and $\bar{t}_i \in \mathbb{R}^+$ the upper bound of the time window. Given a time horizon T, we set $[\underline{t}_i, \bar{t}_i] = [0, T]$ for the depot locations $i = 0, n+1$. Additionally, we assume that the service at each customer $i \in C$ takes a known duration of s_i.

An early visit at a customer location is not allowed and a tardy arrival results in a penalty, which is the squared delay. Let d_i denote the travel time from the depot to a customer $i \in C$ and vice versa. Let $d_{ij} = s_i + d_i + d_j$ denote the service time at customer i plus the traveling time between the visits at customers $i, j \in C$. For ease of notation, we additionally define $d_{0j} = d_j$ and $d_{jn+1} = d_j + s_j$ for $j \in C$.

The depot functions not only as a storage for goods, but also as a recharging station for the vehicles. We assume that the capacity of this station is sufficiently large to charge all vehicles at the same time. As the vehicles are battery powered, let B denote the battery capacity. The battery should be measured in travel time units. For simplification, we assume that the SoC increases linearly while charging and also decrease linearly while driving, although in real-world applications the charging rate decreases for the last 10% to 20% of the battery capacity (Marra et al. [17]). Moreover, we assume that other processes, such as loading, unloading, or waiting do not effect the SoC. Let α be the travel time units obtained when charging one unit of time. Let $b_{ij} = d_i + d_j$ denote the battery units consumed, when travelling between customers i and j, with $i, j \in C$. Again for ease of notation, we also introduce $b_{0j} = d_j$ and $b_{jn+1} = d_j$ for $j \in C$. Each customer $i \in C$ can be associated with a battery window $[\underline{b}_i, \bar{b}]$, with $\underline{b}_i = d_i$ and $\bar{b} = B - d_i$, in which the vehicles SoC should be upon arrival.

The optimization problem involves finding m routes that minimize the sum of squared delays, such that (i) each customer is visited exactly once by any of the vehicle, (ii) no customer is served earlier than its lower time window bound $\underline{t}_i$ and (iii) the the vehicles SoC should always be within the bounds $[0, B]$.

3.1 Exact mixed-integer formulations

In this subsection, we provide an MIQP formulation to solve the TB-VRP. Since the vehicles only have capacity of a single unit, they always have to return to the depot before making the next delivery. In our formulations we adjust the traveling times and battery consumption and model these return trips implicitly, which leads to the following definition. Given $k \in \mathbb{N}$ locations $P_1, \dots, P_k \in V$ with $2 \leq k \leq n+1$, we call the tuple $P = (P_1, \dots, P_k)$ a *tour*, if they are pairwise disjoint and if $P_1 = 0$ and $P_k = n+1$. We introduce binary decision variables x_{ij} for $(i,j) \in A$. If $i, j \in C$ the values of these variables indicate whether or not customer i precedes customer j, i.e. $x_{ij} = 1$ if and only if a vehicle returns from i to the depot and then serves customer j next. The values of x_{0j} with $j \in C$ indicate whether or not j is the first customer of a tour and the values of x_{in+1} with $i \in C$ indicate that i is the last customer of a tour. We introduce continuous

Decision variables	
x_{ij}	indicator if location i precedes location j for $(i,j) \in A$
ϑ_i	start of the service at location $i \in V$
β_i	SoC at location $i \in V$
γ_i	delay at location $i \in V$

Parameters	
m	number of vehicles
n	number of customers
s_i	service time at customer $i \in C$
d_i	travel time from the depot to customer $i \in C$
d_{ij}	traveling plus service time for $(i,j) \in A$
b_{ij}	battery consumption for $(i,j) \in A$
T	time horizon
$[\underline{t}_i, \bar{t}_i]$	time window of location $i \in V$
B	battery capacity
$[\underline{b}_i, \bar{b}_i]$	battery window of location $i \in V$
α	recharging rate

Table 1: Decision variables and parameters.

variables $\vartheta_i \in \mathbb{R}^+$ for the starting time of the service at location $i \in V$. Additionally, for each location $i \in V$ the SoC at the start of the service is captured with a continuous variable $\beta_i \in \mathbb{R}^+$ and the potential delay by a continuous variable $\gamma_i \in \mathbb{R}^+$. An overview of all decision variables and parameters is given in Table 1. Using these variables, the TB-VRP can be modeled by the MIQP:

$$\text{minimize} \quad \sum_{i \in C} \gamma_i^2 \tag{1a}$$

$$\text{subject to} \quad \sum_{i \in C} x_{0i} \leq m \tag{1b}$$

$$\sum_{j \in V:(i,j) \in A} x_{ij} = 1 \qquad i \in C \tag{1c}$$

$$\sum_{j \in V:(j,i) \in A} x_{ji} = 1 \qquad i \in C \tag{1d}$$

$$\vartheta_i + d_{ij} \leq \vartheta_j + (1 - x_{ij})T \qquad (i,j) \in A \tag{1e}$$

$$\beta_i + \alpha(\vartheta_j - \vartheta_i - d_{ij}) - b_{ij} \geq \beta_j - (1 - x_{ij})B \qquad (i,j) \in A \tag{1f}$$

$$\underline{b}_i \leq \beta_i \leq \bar{b}_i \qquad i \in V \tag{1g}$$

$$\underline{t}_i \leq \vartheta_i \leq \bar{t}_i + \gamma_i \qquad i \in V \tag{1h}$$

$$0 \leq \gamma_i \leq T - \bar{t}_i \qquad i \in V \tag{1i}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{1j}$$

In the objective (1a) the delays are squared, so that large individual delays are penalized more heavily than few short delays. Constraint (1b) guarantees that at most m vehicles are used. With the constraints (1c) and (1d) it is ensured that each customer is visited once. Given two consecutively visited locations i and j, constraints (1e) ensure that the difference between the arrival times is sufficiently large. Similarly, constraints (1f) guarantee that the difference of the SoC's accounts for battery consumption and the time spent charging. Additionally, constraints (1e) implicitly impose an ordering of the customers assigned to the same tour, which ensures that no cycles are contained in a solution. Finally, the

domains of the individual variables are imposed by (1g),(1h) and (1j). Here, the time window of a location $i \in V$ can only be violated if a positive value is assigned to γ_i. This guarantees that the delay variables are set correctly.

3.2 The time and battery-expanded formulation

For fractional values of the x-variables the right hand side of (1e) can be large and the right hand side of (1f) can be small, making them very weak constraints in the linear relaxation of (1). This can lead to a large gap between the objective of (1) and its linear relaxation, which is known to impede branch and bound algorithms for solving it. As an alternative to the MIQP formulation proposed before, we propose to model the problem by using layered graphs, which we call *partially time- and battery-expanded graphs* (TBEG), which are known to have much stronger linear relaxations (see for example Gouveia et al. [11]). To do so, for a location $i \in V$ we define a tuple (i, t, b) to be a *node representative* of i, if it is an element of $\{i\} \times [\underline{t}_i, T - d_i - s_i] \times [\underline{b}_i, \bar{b}_i]$. In this case, we also call (t, b) the *arrival state* of the node representative (i, t, b). Then, given an arc $(i, j) \in A$, we further call a tuple $(i, t_i, b_i, j, t_j, b_j)$ an *arc representative* of (i, j), if (i, t_i, b_i) is a node representative of i and (j, t_i, b_j) is a node representative of j. A TBEG is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where the set of nodes $\mathcal{V}$ consists of node representatives and the set of arcs $\mathcal{A}$ of arc representatives. Furthermore, for any location $i \in V$ there has to be at least one node representative in $\mathcal{V}$, and for each node representative (i, t, b) and each arc $(i, j) \in A$ there has to be at least one arc representative of (i, j) in $\mathcal{A}$ with (i, t, b) as its tail. For the location $i \in V$ we denote the subset of $\mathcal{V}$ which contains all the node representatives of i by $\mathcal{V}_i$, and for $(i, j) \in A$ we denote the subset of $\mathcal{A}$ containing all arc representatives of (i, j) by $\mathcal{A}_{ij}$.

For any TBEG $\mathcal{G}$ we can derive a model, which depending on $\mathcal{G}$ can be an exact formulation of the TB-VRP. We use variables x_{ij} for all $(i, j) \in A$ in the same way as they were used in (1). Instead of the other continuous variables we only need to use binary variables y_a for $a \in \mathcal{A}$, take the value 1 if and only if a vehicle transitions from the arrival state of the tail of a to the arrival state of the head of a. With these variables, we propose the following MILP for any TBEG $\mathcal{G}$:

$$\text{minimize} \quad \sum_{a=(i,t_i,b_i,j,t_j,b_j)\in\mathcal{A}} \max(0, t_j - \bar{t}_j)^2 y_a \tag{2a}$$

$$\text{subject to} \quad \sum_{j\in C} x_{0j} \leq m \tag{2b}$$

$$\sum_{j\in V:(i,j)\in A} x_{ij} = 1 \qquad i \in C \tag{2c}$$

$$\sum_{a\in\mathcal{A}_{ij}} y_a = x_{ij} \qquad (i,j) \in A \tag{2d}$$

$$\sum_{a\in\delta^+((i,t,b))} y_a = \sum_{a\in\delta^-((i,t,b))} y_a \qquad (i,t,b) \in \mathcal{V} \tag{2e}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{2f}$$

$$y_a \in \{0,1\} \qquad a \in \mathcal{A}. \tag{2g}$$

We refer to this MILP as $M_{\mathcal{G}}$ to indicate the TBEG it is derived from. Its objective function (2a) realizes the squared delay of the arrival states indicated by the heads of the arcs of the TBEG. Constraint (2b) limits the number of vehicles and the constraints (2c) guarantee that each customer is visited exactly once. The constraints (2d) guarantee that the values assigned to the x- and y-variables are consistent. Constraints (2e) are flow conservation constraints, in which we use $\delta^+((i, t, b))$ for the set of outgoing and $\delta^-((i, t, b))$ for the set of ingoing arcs of the node $(i, t, b) \in \mathcal{V}$. We note here, that, for

now, it is possible for the solutions to contain cycles, which will we will have to take into account in our algorithms later on.

Algorithm 1: transform$(\overline{x}, \overline{y})$

Input: Solutions vectors $\overline{x}$ and $\overline{y}$ of $M_{\mathcal{G}}$ for some TBEG $\mathcal{G}$;
for $(i,j) \in A$ **do**
  $x_{ij} \leftarrow \overline{x}_{ij}$;
for $a = (i, t_i, b_i, j, t_j, b_j) \in \mathcal{A}$ **do**
  if $\overline{y}_a = 1$ **then**
    $\vartheta_j \leftarrow t_j$;
    $\beta_j \leftarrow b_j$;
    $\gamma_j \leftarrow \max(0, t_j - \bar{t}_j)$;
Output: Vectors $x, \vartheta, \beta, \gamma$;

Since the nodes in $\mathcal{G}$ model arrival states at the locations, we can use Algorithm 1 with a solution of $M_{\mathcal{G}}$ as its input and check if its output fulfills the constraints of (1). For all nodes (j,t,b) for which the flow described by the y-variables of (2) is non-zero, it assigns t to the variable ϑ_j, b to the variable β_j and $\max(0, t - \bar{t}_j)$ to the variable γ_j of (1). The objective function (2a) is then exactly the squared delay given by the γ-variables in (1a). Based on this, we can then make the following observation.

Proposition 1. *Let $\mathcal{G}$ be a TBEG, with a set of arcs that guarantees that for any input to Algorithm 1 the output of Algorithm 1 fulfills the constraints (1h) and (1g). Then the solution vectors of $M_{\mathcal{G}}$ can be transformed by Algorithm 1 into vectors that fulfill all the constraints of (1). Furthermore, if additionally it is guaranteed that at least one optimal solution of (1) can be expressed as a tour in $\mathcal{G}$, the solution of $M_{\mathcal{G}}$ can be transformed by Algorithm 1 into a solution of (1).*

In the remainder of this section, we are going to derive a TBEG $\mathcal{G}^{\text{exp}} = (\mathcal{V}^{\text{exp}}, \mathcal{A}^{\text{exp}})$ that does fulfill the abstractly given conditions of Proposition 1. We call this graph the *completely time- and battery-expanded graph.*

We start by introducing the matrix

$$\mathbf{M} = \begin{pmatrix} 1 & \alpha \\ 0 & -1 \end{pmatrix}.$$

It holds $\mathbf{M}^{-1} = \mathbf{M}$ and it will be helpful to consider arrival states in a different coordinate system.

For $(i,j) \in A$, we call a node representative (j, t_j, b_j) of j *reachable* from a node representative (i, t_i, b_i) of i and an arc representative $(i, t_i, b_i, j, t_j, b_j)$ of an arc (i,j) *realizable* if there exist $\theta, \tau \geq 0$, such that

$$\begin{pmatrix} t_j \\ b_j \end{pmatrix} = \begin{pmatrix} t_i \\ b_i \end{pmatrix} + \begin{pmatrix} d_{ij} \\ -b_{ij} \end{pmatrix} + \theta \begin{pmatrix} 1 \\ \alpha \end{pmatrix} + \tau \begin{pmatrix} 0 \\ -1 \end{pmatrix} = \begin{pmatrix} t_i \\ b_i \end{pmatrix} + \begin{pmatrix} d_{ij} \\ -b_{ij} \end{pmatrix} + \mathbf{M}^{\intercal} \begin{pmatrix} \theta \\ \tau \end{pmatrix}.$$

A possible interpretation of θ is the charging time, but non-zero values of τ cannot be realized in practice. It can be interpreted as some amount of instantaneously lost charge or an overestimation of battery consumption. This is allowed in constraint (1g), although energy stored in the battery cannot simply disappear. This choice will be helpful later, when it is preferable to have the area of reachable arrival states as large as possible. Note here, that there is no downside to including the possibility to overestimate the battery consumption, when it comes to feasibility of transitions.

We can deduce that if the arcs in $\mathcal{A}^{\text{exp}}$ are realizable, all tours in $\mathcal{G}^{\text{exp}}$ are guaranteed to fulfill the time and battery constraints. So, if we could include node representatives with all arrival states of the locations and all realizable arc representatives, the conditions of Proposition 1 would be fulfilled by $\mathcal{G}^{\text{exp}}$. The set of node representatives, however, are defined as a rectangle (for each customer index). This means that we cannot include them all in a finite graph. To use the described approach, we first have to show, that it suffices to only consider a finite set of node representatives, i.e., for $\mathcal{V}^{\text{exp}}$ to contain a finite number of nodes. We do this by introducing a concept of domination based on a certain partial order $\preceq$ of the states.

A closer look at the definition of reachable arrival states shows that they are part of a shifted convex cone. Furthermore this convex cone only depends on the recharging rate α. It is therefore a natural choice to use the partial order induced by this cone. The cone is given by

$$\mathcal{C} = \left\{ \theta \begin{pmatrix} 1 \\ \alpha \end{pmatrix} + \tau \begin{pmatrix} 0 \\ -1 \end{pmatrix} \;\middle|\; \theta, \tau \geq 0 \right\}.$$

For two node representatives (i, t_1, b_1) and (i, t_2, b_2) of some location $i \in V$ then holds $(i, t_1, b_1) \preceq (i, t_2, b_2)$ if and only if $(t_2, b_2) - (t_1, b_1) \in \mathcal{C}$. For the arrival states we then similarly write $(t_1, b_1) \preceq (t_2, b_2)$. An easy way to check if two node representatives can be compared by our partial order is presented in the following lemma.

Lemma 1. *Given two node representatives (i, t_1, b_1) and (i, t_2, b_2) of some location $i \in V$ we define $(\theta_1, \tau_1) = (t_1, b_1) \cdot \mathbf{M}$ and $(\theta_2, \tau_2) = (t_2, b_2) \cdot \mathbf{M}$. It then holds $(t_1, b_1) \preceq (t_2, b_2)$, if and only if $\theta_1 \leq \theta_2$ and $\tau_1 \leq \tau_2$.*

Proof. We note, that by the definition of $\mathbf{M}$ and $\mathcal{C}$, any point $(t, b) \in \mathcal{C}$ can be written as $(\theta, \tau) \cdot \mathbf{M}$ for some $\theta, \tau \geq 0$. The result then directly follows from the definition of the partial order $\preceq$. □

With the previously outlined interpretation of θ and τ in mind, $(t_1, b_1) \preceq (t_2, b_2)$ for two arrival states of the same location, means that less time has to be spent charging and the battery consumption has to be overestimated by a smaller amount to reach (t_1, b_1) instead of (t_2, b_2). For this reason we now use the partial order to say a node representative (i, t_1, b_1) *dominates* a node representative (i, t_2, b_2) of some location $i \in V$, if $(i, t_1, b_1) \preceq (i, t_2, b_2)$, which is further justified by the following observation.

Proposition 2. *Let (i, t_i^1, b_i^1), (i, t_i^2, t_i^2) be node representatives of some location $i \in V$ such that $(i, t_i^1, b_i^1) \preceq (i, b_i^2, t_i^2)$ and let (j, t_j, b_j) be a node representative of some other location $j \in V$, such that $(i, j) \in A$. If (j, t_j, b_j) is reachable from (i, t_i^2, t_i^2), then (j, t_j, b_j) os also reachable from (i, t_i^1, b_i^1).*

Proof. This is a direct consequence of the definition of arrival states. □

The following result shows that our concept of domination is useful to reduce the number of node representatives that we have to include in $\mathcal{G}^{\text{exp}}$.

Lemma 2. *Let (i, t_i, b_i) be a node representative of some location $i \in V$ and let $j \in V$ be another location such that $(i, j) \in A$. Then there exists at most one node representative (j, t_j, b_j) of j that is reachable from (i, t_i, b_i) and is not dominated by another reachable node representative of j.*

Proof. Assume that there are two different node representatives (t_j^1, b_j^1) and (t_j^2, b_j^2) of j that are reachable from (i, t_i, b_i) and not dominated by another node representative that

is also reachable from (i, t_i, b_i). Then we can find $\theta_1, \theta_2, \tau_1, \tau_2 \geq 0$, such that

$$\begin{pmatrix} t_j^2 \\ b_j^2 \end{pmatrix} = \begin{pmatrix} t_i \\ b_i \end{pmatrix} + \begin{pmatrix} d_i + d_j \\ -d_i - d_j \end{pmatrix} + \theta_1 \begin{pmatrix} 1 \\ \alpha \end{pmatrix} + \tau_1 \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \tag{3}$$

$$\begin{pmatrix} t_j^1 \\ b_j^1 \end{pmatrix} = \begin{pmatrix} t_i \\ b_i \end{pmatrix} + \begin{pmatrix} d_i + d_j \\ -d_i - d_j \end{pmatrix} + \theta_2 \begin{pmatrix} 1 \\ \alpha \end{pmatrix} + \tau_2 \begin{pmatrix} 0 \\ -1 \end{pmatrix}. \tag{4}$$

Since they are both not dominated by another node representative, they can in particular also not dominate each other, which implies $(t_j^1, b_j^1) - (t_j^2, b_j^2) \notin \mathcal{C}$ and $(t_j^2, b_j^2) - (t_j^1, b_j^1) \notin \mathcal{C}$. For this to be true, it has to hold either $(\theta_1 - \theta_2) \geq 0$ or $(\tau_1 - \tau_2) \geq 0$, but not both of those inequalities. W.l.o.g, let $\theta_1 \geq \theta_2$ and $\tau_1 < \tau_2$. The point (t_j^3, b_j^3) given by

$$\begin{pmatrix} t_j^3 \\ b_j^3 \end{pmatrix} = \begin{pmatrix} t_i \\ b_i \end{pmatrix} + \begin{pmatrix} d_i + d_j \\ -d_i - d_j \end{pmatrix} + \theta_2 \begin{pmatrix} 1 \\ \alpha \end{pmatrix} + \tau_1 \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \tag{5}$$

then dominates both (t_j^1, b_j^1) and also (t_j^2, b_j^2). However, since $t_j^3 = t_j^2$ and $b_j^2 \leq b_j^3 \leq b_j^1$, then (j, t_j^3, b_j^3) fulfills the conditions to be a node representative of i, if (j, t_j^1, b_j^1) and (j, t_j^2, b_j^2) are node representatives. Furthermore, (j, t_j^3, b_j^3) also fulfills the condition to be reachable from (i, t_i, b_i). So, we found a node representative of j that is reachable from (i, t_i, b_i) and dominates (j, t_j^1, b_j^1) and (j, t_j^2, b_j^2), a contradiction to the initial assumption. □

Lemma 2 shows that we do not have to consider the whole set of node representatives, we just need to be able to find the reachable node representative that dominates all others. While we could for example use an LP formulation to find this point, the geometry of the reachable node representatives is simple enough to find this point by distinguishing some special cases. The steps for that are given in Algorithm 2. In this strategy the time that

Algorithm 2: recharge(i, t, b, j)

Input: A node representative (i, t, b) of some location $i \in V$ and another location j, such that $(i, j) \in A$;
$t_c \leftarrow \max(\underline{b}_j + b_{ij} - b), 0)/\alpha$; (forced charging time)
$t_w \leftarrow \max(\underline{t}_j - d_{ij} - t), 0)$; (forced wait time)
$t_d \leftarrow \max(t_{ct}, t_{wt})$; (time at depot)
$t_j \leftarrow t + t_d + d_{ij}$; (arrival time at j)
$b_j \leftarrow \min(B, b - d_i + \alpha t_d) - d_j$; (SoC at j)
Output: A node representative (j, t_j, b_j) of j;

has to be spent at the depot is calculated first. It is the maximum of the time that the vehicle has to wait at the depot in order to not arrive too early at the next customer and the time required to charge sufficient battery for the tour to the next customer and back. Based on this the arrival time and the SoC of the vehicle at the next customer can be calculated. Using this strategy guarantees that the vehicles spend as little time as possible at the depot. Therefore, using this strategy for a given tour assignment there exists no recharging strategy by which a customer is reached at an earlier time, while guaranteeing feasibility of the transition. This implies that using this strategy enforces minimal delays.

Let $(i, t_i, b_i, j, t_j, b_j) \in \mathcal{A}$ be an arc representative and let $(t, \hat{t}_j, \hat{b}_j)$ be the output of **recharge**(i, t_i, b_i, j). This implies that $\hat{t}_j \geq t_j$ holds and that $\hat{t}_j$ is a better estimation of the arrival time than t_j. So, we can replace t_j by $\hat{t}_j$ in (2a) to get a better estimation of the delays. More explicitly using $\hat{t}_j$ for the arrival time calculated in this way, we replace the objective (2a) of the MILP $M_{\mathcal{G}}$ by

$$\sum_{a=(i,t_i,b_i,j,t_j,b_j)\in\mathcal{A}} \max(0, \hat{t}_j - \bar{t}_j) - y_a. \tag{2a'}$$

This objective function provides better bounds in our algorithms, but has no effect on the

Algorithm 3: expand(P)

Input: A tour P in G ;
$(i, t, b) \leftarrow (0, 0, B)$;
for $j = 1, \ldots, k-1$ **do**
  $\mathcal{P}_j \leftarrow (i, t, b)$;
  $(i, t, b) \leftarrow$ **recharge**(i, t, b, P_{j+1});
$\mathcal{P}_k \leftarrow (i, t, b)$;
Output: A tour representative $\mathcal{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_k)$ of P;

discussions, because the two objective functions are identical for the time- and battery-expanded graph $\mathcal{G}^{\text{exp}}$.

The next step is to be able to convert tours in G into tours of node representatives. We first extend the term representatives to tours by calling a tour $\mathcal{P}$ in a TBEG $\mathcal{G}$ a *tour representative* of a tour P in G, if all of the nodes in $\mathcal{P}$ are node representatives of the nodes in P.

Given two tours $\mathcal{P}, \hat{\mathcal{P}}$ in $\mathcal{G}$, which are both tour representatives of some tour P in G, we write $\mathcal{P} \preceq \hat{\mathcal{P}}$, if the relation $\preceq$ holds true between all the node representatives in $\mathcal{P}$ and $\hat{\mathcal{P}}$. With this we can now prove the following result for Algorithm 2.

Lemma 3. *Given a tour P of length k in G, then Algorithm 3 returns a tour representative of P, for which arcs between subsequent nodes are realizable. Furthermore, it holds $\mathcal{P} \preceq \hat{\mathcal{P}}$ for all other tour representatives $\hat{\mathcal{P}}$ of P.*

Proof. Since the subsequent nodes in the output $\mathcal{P}$ of Algorithm 3 are calculated by Algorithm 2, the transition between them has to be realizable. Now, by Lemma 2 its second node $\mathcal{P}_2$ is the unique node representative of P_2 that is non-dominated and reachable from $\mathcal{P}_1$. In particular this holds for $\hat{\mathcal{P}}_2$ of some other tour representative $\hat{\mathcal{P}}$ of P. Inductively this holds true for all nodes in $\mathcal{P}$, which concludes the proof. □

Now, interpreting the tours $\mathcal{P}$ as graphs $(\mathcal{V}_\mathcal{P}, \mathcal{A}_\mathcal{P})$ with nodes $\mathcal{V}_\mathcal{P} = \mathcal{P}$ and arcs between subsequent nodes, by Lemma 3, Algorithm 3 can be used to convert tours in G into tour representatives of it, which contain only realizable arc representatives. Furthermore, there exists no other tour representative with realizable arc representatives whose nodes are non-dominated by the nodes of the output of Algorithm 3.

Based on this we can define $\mathcal{V}^{\text{exp}} = \bigcup_{P \in \mathbb{P}} \mathcal{V}_\mathcal{P}$ and $\mathcal{A}^{\text{exp}} = \bigcup_{\mathcal{P} \in \mathbb{P}} \mathcal{A}_\mathcal{P}$, where $\mathbb{P}$ denotes the set of all tours in G. By construction, any tour (including those contained in a solution of the TB-VRP) has a tour representative in the graph $\mathcal{G}^{\text{exp}} = (\mathcal{V}^{\text{exp}}, \mathcal{A}^{\text{exp}})$. Further, since all arc representatives in it are realizable, each tour representative in $\mathcal{G}^{\text{exp}}$ can be transformed into vectors that fulfill all time and battery constraints. Therefore, $\mathcal{G}^{\text{exp}}$ fulfills the conditions formulated in Proposition 1 and hence $M_{\mathcal{G}^{\text{exp}}}$ is an exact formulation of the TB-VRP. For this reason it is justified to now call $M_{\mathcal{G}^{\text{exp}}}$ the time- and battery-expanded formulation of the TB-VRP.

We note, that it should be clear from the derivation of $\mathcal{G}^{\text{exp}}$ that this formulation is not suitable for finding solutions of the TB-VRP in practise. It would require to enlist all tours in G, whose number grows exponentially with the size of the graph. However, as we will show in the following section, it is possible to use other TBEGs $\mathcal{G}$ for which $M_\mathcal{G}$ is a relaxation of the TB-VRP. For any TBEG $\mathcal{G}$, it is straightforward to test if a solution of $M_\mathcal{G}$ is also contained in $M_{\mathcal{G}^{\text{exp}}}$, simply by applying Algorithm 3 to the tours in G described by the x-variables and comparing it to the tour described by the y-variables. If they are

the same, we have found an optimal solution of $M_{\mathcal{G}^{\text{exp}}}$. If they are not the same, then we can still use the output of Algorithm 3 to obtain feasible solution candidates.

4 Relaxations of the time and battery-expanded formulation

As outlined before, the first step in deriving an algorithm making use of the time and battery-expanded formulation is to find TBEGs $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ for which $M_{\mathcal{G}}$ is a good relaxation relaxation of $M_{\mathcal{G}_{\text{exp}}}$. We start by introducing the TBEG $\mathcal{G}^{\text{init}} = (\mathcal{V}^{\text{init}}, \mathcal{A}^{\text{init}})$, where

$$\mathcal{V}^{\text{init}} = \{(i, \underline{t}_i, \overline{b}_i), i \in V\},$$
$$\mathcal{A}^{\text{init}} = \{(i, \underline{t}_i, \overline{b}_i, j, \underline{t}_j, \overline{b}_j), (i,j) \in A\}.$$

This is the TBEG with the smallest possible number of nodes and arcs. We call arc representatives $(i, t_i, b_i, j, t_j, b_j)$ of an arc $(i,j) \in A$ *underestimating* if $(j, t_j, b_j) \preceq \textbf{recharge}(i, t_i, b_i, j)$. We additionally say it is *minimally underestimating* in a TBEG $\mathcal{G}$ if there exists no node representative $(j, \hat{t}_j, \hat{b}_j) \in \mathcal{V}$, with $(j, \hat{t}_j, \hat{b}_j) \neq (j, t_j, b_j)$ such that $(j, \hat{t}_j, \hat{b}_j) \preceq \textbf{recharge}(i, t_i, b_i, j)$ and $(j, \hat{t}_j, \hat{b}_j) \preceq (j, t_j, b_j)$.

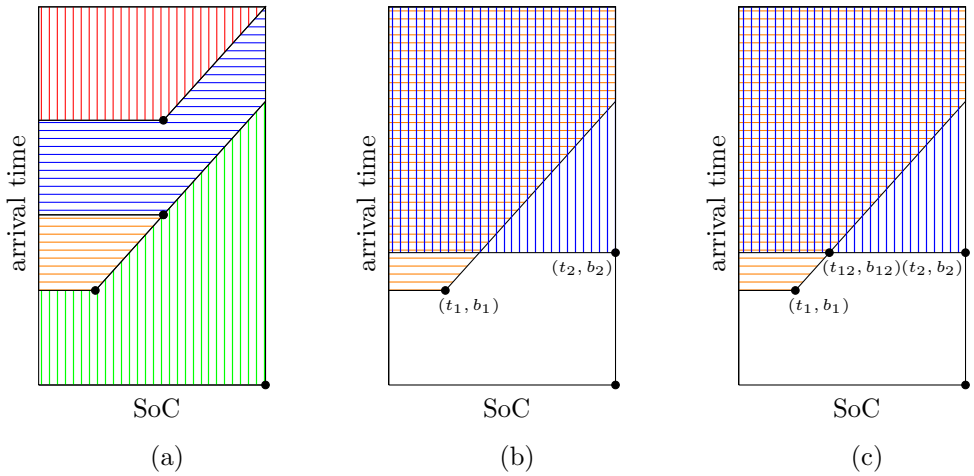
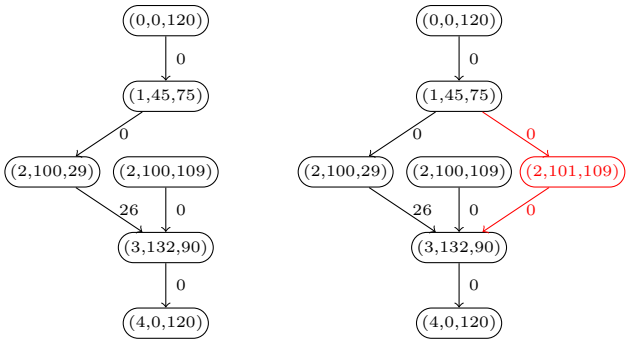


Figure 1: Arrival states at some customer j.

Let us illustrate this definition by giving a simplified example. Consider a situation, in which the set of node representatives in $\mathcal{V}_j$ for some customer $j \in C$ is given by the black dots in Figure 1a and we want to include an arc from a node (i, t_i, b_i) to a node representative of j. Even if the node representative given by the output of Algorithm 2 is not an element of $\mathcal{V}$, it is guaranteed to be in one of the colored areas. Since all points in any of the areas are dominated by its bottom right corner (the apex of a shifted $\mathcal{C}$), connecting (i, t_i, b_i) to the node representative with arrival state equal to the bottom right corner of this area, leads to a graph that underestimates the arrival states. Arc representatives chosen in this way are minimally underestimating. Constructing the whole set of arcs in this way, leads to a graph in which the nodes do not only represent a single arrival state, but a whole section of the rectangle of arrival states. Note that the graph $\mathcal{G}^{\text{exp}}$ can be interpreted in the same way, but it is designed in such a way, that the outputs of Algorithm 2 used in the definition of underestimating arcs are already present $\mathcal{V}^{\text{exp}}$.

Another reason for using TBEGs in which the arc representatives are underestimating is given by the following result.



(a) The TBEG $\mathcal{G}_1$. (b) The TBEG $\mathcal{G}_2$.

Figure 2: Illustrations of two TBEGs.

Proposition 3. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, in which all arcs are minimally underestimating, P a tour, $\mathcal{P}$ a tour representative of P in $\mathcal{G}$ and $\mathcal{P}^{\text{exp}}$ a tour representative of P in $\mathcal{G}^{\text{exp}}$, then $\mathcal{P}_i \preceq \mathcal{P}_i^{\text{exp}}$ holds for $i = 1, \dots, |P|$.*

Proof. Assume the assertion does not hold, then there exists a minimal index $2 \leq k \leq |P|$ such that $\mathcal{P}_k \not\preceq \mathcal{P}_k^{\text{exp}}$. It then has to hold $\mathcal{P}_{k-1} \preceq \mathcal{P}_{k-1}^{\text{exp}}$. Since the output of **recharge** is always reachable from its input, by Proposition 2 this implies

$$\textbf{recharge}(\mathcal{P}_{k-1}, P_k) \preceq \textbf{recharge}(\mathcal{P}_{k-1}^{\text{exp}}, P_k).$$

However, since the arcs in $\mathcal{G}$ are underestimating and by the construction of $\mathcal{G}^{\text{exp}}$, it then holds

$$\mathcal{P}_k \preceq \textbf{recharge}(\mathcal{P}_{k-1}, P_k) \preceq \textbf{recharge}(\mathcal{P}_{k-1}^{\text{exp}}, P_k) = \mathcal{P}_k^{\text{exp}},$$

which is a contradiction to the initial assumption. □

Since our concept of domination implies that the arrival times are underestimated, a direct consequence of Proposition 3 is that for any chosen tour the objective of the MILP $M_{\mathcal{G}}$ is always smaller than or equal to the objective of $M_{\mathcal{G}^{\text{exp}}}$. This implies that we can use a TBEG $M_{\mathcal{G}}$ as a relaxation of $M_{\mathcal{G}^{\text{exp}}}$, if all of its arcs are minimally underestimating. For this reason, we will from now on only consider TBEGs, in which every arc is minimally underestimating.

The situation in Figure 1a, where all nodes can be strictly ordered with respect to the partial order $\preceq$ is a special case, since this implies that there always is only one minimally underestimating arc for a given node and a fixed other location. A more complicated situation is illustrated in Figure 1b, where there are two nodes (j, t_1, b_1) and (j, t_2, b_2) and none of them dominates the other. If the output of **recharge**(i, t_i, b_i, j) for a node $(i, t_i, b_i) \in \mathcal{V}$ is within the area striped vertically and horizontally, there are two choices for minimally underestimating arcs. In this situation two problems occur, which we will demonstrate with a simple example. The TBEGs $\mathcal{G}_1$ and $\mathcal{G}_2$ depicted in Figure 2 were derived for an instance of the TB-VRP given by the parameters in Table 2 with a service time of 0 minutes for all locations, and include all minimally underestimating arcs. The numbers next to the arcs are the delays that occur when an arc is used. Starting at the node $(1, 45, 75)$, the output of **recharge**$(1, 45, 75, 2)$ is $(2, 101, 29)$. So, in $\mathcal{G}_1$ the node $(1, 45, 75)$ is only connected to $(2, 100, 29)$ and not to $(2, 100, 109)$, because $(100, 29) \preceq (100, 109)$. Now, consider $\mathcal{G}_2$ which contains the nodes of $\mathcal{G}_1$ and an extra node $(2, 101, 109)$. Because $(2, 101, 109) \not\preceq (2, 100, 29)$ and $2, 100, 29) \not\preceq ((2, 101, 109)$ both are successors of $(1, 45, 75)$. Since the arc to $(2, 101, 109)$ immensely underestimates the correct battery consumption, customer 3 can be reached without any delay. Both TBEGs could be used to obtain relaxations of the time and battery expanded formulation, but the problem is

Customer	Time window	Depot Distance	Battery Interval
1	$[45, 55]$	45	$[45, 75]$
2	$[100, 110]$	11	$[11, 109]$
3	$[132, 142]$	30	$[30, 90]$

Table 2: Parameters of a VRPTB instance.

that $\mathcal{G}_2$ can be obtained from $\mathcal{G}_1$ by adding a node, the MILP $M_{\mathcal{G}_2}$ has a smaller objective than $M_{\mathcal{G}_1}$. So, in this case the graph with less nodes is a better relaxation than the one with more nodes. In an algorithm that dynamically expands the set of nodes, however, the quality of the relaxation should only improve in order for it to provide tight bounds.

In addition to this, there is another problem that we will shortly discuss. In $\mathcal{G}_2$, there are two different tour representatives of the tour $(0, 1, 2, 3, 4)$. For $\mathcal{G}^{\text{exp}}$ and also $\mathcal{G}^{\text{init}}$, however, there is a one-to-one correspondence. One of the goals in designing our refinement strategy is to maintain this one-to-one correspondence between tours in G and their tour representatives in $\mathcal{G}$, since it makes it easy to check if a certain tour in our graph underestimates the delays or if its arrival states are represented correctly.

Our approach to prevent both of the previously mentioned problems is to expand the set of nodes more carefully, so that it is guaranteed that there is always only one minimally underestimating arc to choose. The idea we are going to use is illustrated in Figure 1c. We add the bottom right corner of the intersecting areas as nodes. As depicted there, the node representatives with arrival states in the blue and orange striped area can be connected to (j, t_{12}, b_{12}) and this is now the only minimally underestimating arc. In order to formalize this idea, we start by giving an algorithm for finding the bottom right corner of the intersecting areas, which in the illustration is denoted by (t_{12}, b_{12}). Algorithm 4 formalizes this idea. Here, a location j and two node representatives (j, t_1, b_1) and (j, t_2, b_2)

Algorithm 4: intersect(j, t_1, b_1, t_2, b_2)

Input: Node representatives (j, t_1, b_1) and (j, t_2, b_2) of some location $j \in V$;
$(\theta_1, \tau_1) = (t_1, b_1) \cdot \mathbf{M}$;
$(\theta_2, \tau_2) = (t_1, b_1) \cdot \mathbf{M}$;
$(\theta_{12}, \tau_{12}) = (\max(\theta^1, \theta^2), \max(\tau^1, \tau^2))$;
$(t_{12}, b_{12}) = (\theta_{12}, \tau_{12}) \cdot \mathbf{M}$;
Output: A node representative (j, t_{12}, b_{12}) of j;

of j are used as an input. Then we use the matrix $\mathbf{M}$ to transform their arrival states into a different coordinate system given, i.e., a coordinate system using the same vectors as the ones defining $\mathcal{C}$. In this coordinate system the coordinates of the new points can be simply calculated with the maximum function. Finally, we go back to the original coordinate system and return the calculated point.

Based on this algorithm, we call a TBEG $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ *intersection complete*, if for any node representatives $(j, t_1, b_1), (j, t_2, b_2) \in \mathcal{V}$ of some location $j \in V$ the node representative **intersect**(j, t_1, b_1, t_2, b_2) is also an element of $\mathcal{V}$.

The following two results show that intersection complete TBEGs avoid the previously illustrated problem of multiple minimally intersecting arcs.

Lemma 4. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be an intersection complete TBEG and let (j, t, b) be a node representative of a location $j \in V$. If there are two node representatives $(j, t_1, b_1) \in \mathcal{V}$ and $(j, t_2, b_2) \in \mathcal{V}$ such that $(j, t_1, b_1) \preceq (j, t, b)$ and $(j, t_2, b_2) \preceq (j, t, b)$, then there exists a node representative $(j, t_{12}, b_{12}) \in \mathcal{V}$, such that $(j, t_1, b_1) \preceq (j, t_{12}, b_{12})$, $(j, t_2, b_2) \preceq (j, t_{12}, b_{12})$ and $(j, t_{12}, b_{12}) \preceq (j, t, b)$.*

Proof. Let $(j, t_{12}, b_{12}) = \textbf{intersect}(j, t_1, b_1, t_2, b_2)$ and let

$$
\begin{aligned}
(\theta, \tau) &= (t, b) \cdot \mathbf{M}, \\
(\theta_1, \tau_1) &= (t_1, b_1) \cdot \mathbf{M}, \\
(\theta_2, \tau_2) &= (t_2, b_2) \cdot \mathbf{M}, \\
(\theta_{12}, \tau_{12}) &= (j, t_{12}, b_{12}) \cdot \mathbf{M}.
\end{aligned}
$$

By Lemma 1, it then holds $(j, t_1, b_1) \preceq (j, t_{12}, b_{12})$ and $(j, t_2, b_2) \preceq (j, t_{12}, b_{12})$. Furthermore by the same lemma it also holds $\theta \geq \theta_1, \theta_2$ and $\tau \geq \tau_1, \tau_2$, so $\theta \geq \max(\theta_1, \theta_2)$ and $\tau \geq \max(\tau_1, \tau_2)$. Then since $(\theta_{12}, \tau_{12}) = (\max(\theta_1, \theta_2), \max(\tau_1, \tau_2))$ again by Lemma 1 holds $(j, t_{12}, b_{12}) \preceq (j, t, b)$, which concludes the proof. □

Proposition 4. *Given a tour P in G and an intersection complete TBEG $\mathcal{G}$, in which all arc representatives are minimally underestimating, there exists a unique tour representative $\mathcal{P}$ of P in $\mathcal{G}$.*

Proof. Assume there are two different tour representatives $\hat{\mathcal{P}}$ and $\mathcal{P}$ of P in $\mathcal{G}$. Then there exists a minimal index k with $2 \leq k \leq |P|$, for which $\hat{\mathcal{P}}_k \neq \mathcal{P}$. Let $\mathcal{P}_{k-1} = (i, t_i, b_i)$, $\mathcal{P}_k = (j, t_1, b_1)$, and $\hat{\mathcal{P}}_k = (j, t_2, b_2)$, then holds $\hat{\mathcal{P}}_{k-1} = (i, t_i, b_i)$. Furthermore it holds $(i, t_i, b_i, j, t_1, b_1) \in \mathcal{V}$ and $(i, t_i, b_i, j, t_2, b_2) \in \mathcal{V}$, which both have to be underestimating arcs. This implies $(j, t_1, b_2), (j, t_2, b_2) \preceq \textbf{recharge}(i, t_i, b_i, j)$ and we are in the situation of Lemma 4. The existence of the state from Lemma 4 (j, t_{12}, b_{12}) now yields a contradiction to the arcs $(i, t_i, b_i, j, t_1, b_1)$ and $(i, t_i, b_i, j, t_2, b_2)$ being minimally underestimating unless $(t_1, b_1) = (t_2, b_2) = (t_{12}, b_{12})$, which is a contradiction to our initial assumptions. □

In the remainder of this section, we demonstrate how intersection complete TBEGs can be found in practise. We note that $\mathcal{G}^{\text{init}}$ is an intersection complete TBEG, because each location has only one node representative and it holds $\textbf{intersect}(j, t, b, t, b) = (j, t, b)$ for any $(j, t, b) \in \mathcal{V}^{\text{init}}$. In the following, we will show that we can use Algorithm 5 to obtain refined intersection complete graphs. Algorithm 5 takes a TBEG $\mathcal{G}$, and a node representative (i, t, b) of a location $i \in V$ as its input. The output is a larger TBEG in which (i, t, b) and some additional nodes are included in its set of nodes.

Algorithm 5: addNode$(\mathcal{G}, i, t, b)$

Input: A TBEG $\mathcal{G}$, and a node representative (i, t, b) of some location $i \in V$;
$\mathcal{V}_{\text{new}} \leftarrow \{(i, t, b)\}$;
for $(i, \hat{t}, \hat{b}) \in \mathcal{V}_i$ **do**
  $(t_{\text{new}}, b_{\text{new}}) \leftarrow \textbf{intersect}(i, t, b, \hat{t}, \hat{b})$;
  if $b_{\text{new}} \in [\underline{b}_i, \overline{b}_i]$ **then**
    $\mathcal{V}_{\text{new}} \leftarrow \mathcal{V}_{\text{new}} \cup \{(i, t_{\text{new}}, b_{\text{new}})\}$;
$\mathcal{A} \leftarrow \{a \in \mathcal{V} \times \mathcal{V} \mid a \text{ is minimally underestimating}\}$;
Output: A larger TBEG $\mathcal{G}_{\text{new}} = (\mathcal{V}_{\text{new}}, \mathcal{A}_{\text{new}})$;

For the returned TBEG we can prove the following result.

Lemma 5. *Given an intersection complete TBEG $\mathcal{G}$, a location $i \in V$ and a node representative (i, t_i, b_i) of i, the output of Algorithm 5 given by $\mathcal{G}_{\text{new}} = \textbf{addNode}(\mathcal{G}, i, t_i, b_i)$ is an intersection complete TBEG.*

Proof. Assume that the output is not intersection complete. Then there exists a $j \in C$ and two node representatives $(j, t_1, b_1) \neq (j, t_2, b_2)$, such that the $\textbf{intersect}(j, t_1, b_1, t_2, b_2) \notin$

$\mathcal{V}_{\text{new}}$. We set

$$
\begin{aligned}
(\theta_1, \tau_1) &= (t_1, b_1) \cdot \mathbf{M}, \\
(\theta_2, \tau_2) &= (t_2, b_2) \cdot \mathbf{M}, \\
(\theta_{12}, \tau_{12}) &= (\max(\theta^1, \theta^2), \max(\tau^1, \tau^2)), \\
(t_{12}, b_{12}) &= (\theta_{12}, \tau_{12}) \cdot \mathbf{M}.
\end{aligned}
$$

We note that $j = i$ has to hold because otherwise $\mathcal{G}$ is not intersection complete. For the same reason at least one of (j, t_j^1, b_j^1) and (j, t_j^2, b_j^2) is not an element of $\mathcal{V}$. Without loss of generality, let $(j, t_j^1, b_j^1) \notin \mathcal{V}$. Now, both $(\theta_{12}, \tau_{12}) = (\theta_1, \tau_1)$ and $(\theta_{12}, \tau_{12}) = (\theta_2, \tau_2)$ directly lead to a contradiction. Again without loss of generality, let $(\theta_{12}, \tau_{12}) = (\theta_2, \tau_1)$. This implies $\tau_1 \geq \tau_2$ and $\theta_1 \leq \theta_2$. Since (j, t_j^1, b_j^1) was added during the execution of Algorithm 5 there has to be a node representative $(j, t_3, b_3) \in \mathcal{V}_i \cup \{(i, t, b)\}$ such that $\theta_3 = \theta_2$ and $\tau_3 \leq \tau_2$, where $(\theta_3, \tau_3) = (t_3, b_3) \cdot \mathbf{M}$. Similarly, we can find a node representative $(j, t_4, b_4) \in \mathcal{V}_i \cup \{(i, t, b)\}$ such that $\theta_4 = \theta_1$ and $\tau_4 \leq \tau_1$, where $(\theta_4, \tau_4) = (t_4, b_4) \cdot \mathbf{M}$ (if already $(j, t_1, b_1) \in \mathcal{V}$ holds, we can choose $(t_4, b_4) = (t_1, b_1)$). Since $\mathcal{G}$ is intersection complete, it has to hold $\textbf{intersect}(j, t_3, b_3, t_4, b_4) \in \mathcal{V}_{\text{new}}$. Since $\tau_3 \leq \tau_2 \leq \tau_1 = \tau_4$ and $\theta_4 \leq \theta_1 \leq \theta_2 = \theta_3$ holds $(\max(\theta_3, \theta_4), \max(\tau_3, \tau_4)) = (\theta_2, \tau_1) = (\theta_{12}, \tau_{12})$. This implies $(j, t_{12}, b_{12}) \in \mathcal{V}_{\text{new}}$, a contradiction. □

Finally, we present Algorithm 6, which takes a TBEG and a tour as its input, and returns a TBEG, in which the tour representative of P is unique and is also contained in $\mathcal{G}^{\text{exp}}$. The input consists of a tour P in G. If some of the transitions in the tour

Algorithm 6: refine$(\mathcal{G}, P)$

```
Input: A TBEG 𝒢 and a tour P in G of length k;
𝒫 ← expand(P);
for (i, t, b) ∈ 𝒫 do
    𝒢_new ← addNode(𝒢, i, t, b);
𝒜 ← {a ∈ 𝒱 × 𝒱 | a is minimally underestimating};
Output: A larger TBEG 𝒢_new = (𝒱, 𝒜);
```

representative of P in $\mathcal{G}$ were underestimating the arrival states, then the following result shows that this algorithm guarantees that its tour representative in the output models the arrival states correctly.

Proposition 5. *Given an intersection complete TBEG $\mathcal{G}$ and a tour P in G, then the output $\mathcal{G}_{\text{new}}$ of Algorithm 6 is an intersection complete TBEG. Furthermore, there is a unique tour representative $\mathcal{P}$ of the tour P in $\mathcal{G}_{\text{new}}$ and this tour representative is the same as the tour representative of P in $\mathcal{G}_{\text{exp}}$.*

Proof. The first statement is a direct consequence of Lemma 5, since the output is created by iteratively applying Algorithm 5 to an intersection complete graph. The uniqueness of the tour representative follows from Proposition 4 and the final assertion from the fact that we use Algorithm 3 for both deriving $\mathcal{G}^{\text{exp}}$ and in Algorithm 5 to determine the added nodes. □

With this we obtained a strategy to construct practically useful TBEGs. We take $\mathcal{G}^{\text{init}}$ as the initial TBEG and keep correcting the tour representatives of tours in G, until the solution of the MILP from the current TBEG are the same as in $\mathcal{G}^{\text{exp}}$. In the following section we convert this strategy into two different algorithms for solving the TB-VRP.

5 Refinement algorithms

Based on the results of the previous section, we can derive refinement algorithms for solving the TB-VRP. For ease of notation we denote the linear relaxation of an MILP M by M^*. We further use $\text{opt}(M)$ for the optimal value of an MILP and $\text{feas}(M)$ for the set of feasible points of an MILP. Furthermore for a subtour S in G we refer to the constraint

$$\sum_{(i,j)\in A: i,j\in S} x_{ij} \leq |S| - 1$$

as the subtour elemination constraint (SEC) for S.

We start with Iterative Refinement given in Algorithm 7. In theory he initial TBEG

Algorithm 7: Iterative Refinement Algorithm

```
1  Input: An instance of the TB-VRP;
2  G ← G^init (current TBEG), 𝕊 ← {} (pool of subtours);
3  (x̄, ȳ) ← solution of M_G;
4  while (x̄, ȳ) ∉ feas(M_{G^exp}) do
5      if x̄ describes a subtour S then
6          𝕊 ← 𝕊 ∪ {S};
7      else
8          P ← a tour described by x̄;
9          G ← refine(G, P);
10     (x̄, ȳ) ← solution of M_G with additional SECs for the subtours in 𝕊;
11 Output: Vectors x̄, ȳ describing a solution of the TB-VRP instance.
```

of this algorithm could be any intersection complete TBEG, but for the sake of simplicity we choose to start with $\mathcal{G}^{\text{init}}$. This graph is then refined, until its solution is feasible for the time- and battery-expanded formulation. In addition to that the case of the solution containing subtours is taken care of, by adding subtour elimination constraints instead of refining the graph. Since upon termination of the algorithm the solution of a relaxation was found to be feasible, the returned values $(\overline{x}, \overline{y})$ are proven optimal solutions of the TB-VRP. Because after each iteration there is either one more tour that is represented correctly (due to Proposition 5) or a subtour is excluded, the algorithm has to terminate after a finite number of iterations.

An alternative approach, following the Branch and Refine strategy, is presented in Algorithm 8. Here, in line 23 **branch** represents a function that performs the usual branching step in a branch and bound algorithm. Since the integer variables are all binary variables, it returns two MILPs: one in which a variable that had a fractional value in the solution set to 0, and one in which it is set to 1. Furthermore, in line 20 **exchange** is used to express a function that takes the list of open problems, the old graph and the new graph as its input, and returns an adapted list of open problems in which the constraints and the objective derived from the old graph are exchanged with those from the new graph.

This algorithm is very similar to usual branch and bound algorithms applied to solve MILPs. The only difference is that the graph refinement is integrated into the exploration of the branch and bound search tree. Branch and bound algorithms follow the idea to divide the problem into subproblems and use relaxations to find valid bounds for reducing the number of subproblems that have to be considered. In most cases the linear relaxation of the MILP is used to do this, however it is not the only possibility. In Branch and Refine we use the linear relaxation of another MILP $M_{\mathcal{G}}$ instead and adjust this MILP during its execution. Since any of the used MILPs $M_{\mathcal{G}}$ are relaxations of $M_{\mathcal{G}^{\text{exp}}}$, we can

Algorithm 8: Branch and Refine

Input: A TB-VRP instance;
$\mathcal{G} \leftarrow \mathcal{G}^{\text{init}}$ (current TBEG), $\mathbb{S} \leftarrow \{\}$ (pool of subtours), $L_O \leftarrow \{M_{\mathcal{G}}\}$ (list of open nodes), $L_C \leftarrow \emptyset$, (list of closed nodes) $U \leftarrow \infty$ (upper bound);
while $L_O \neq \emptyset$ **do**
    Choose $M \in L_O$;
    if $\text{feas}(M^*) = \emptyset$ *or* $\text{opt}(M^*) \geq U$ **then**
        Move M from L_O to L_C;
    else
        $(\overline{x}, \overline{y}) \leftarrow$ a solution of M^* with additional SECs for the subtours in $\mathbb{S}$;
        if $(\overline{x}, \overline{y})$ *is integer* **then**
            if $(\overline{x}, \overline{y}) \in \text{feas}(M_{\mathcal{G}^{\text{exp}}})$ **then**
                $U \leftarrow \text{opt}(M^*)$;
                $(\hat{x}, \hat{y}) \leftarrow (\overline{x}, \overline{y})$;
                Move M from L_O to L_C;
            else
                if $\overline{x}$ *describes a subtour* S **then**
                    $\mathbb{S} \leftarrow \mathbb{S} \cup \{S\}$;
                else
                    $P \leftarrow$ a tour described by $\overline{x}$;
                    $\mathcal{G}_{\text{old}} \leftarrow \mathcal{G}$;
                    $\mathcal{G} \leftarrow$ **refine**$(\mathcal{G}, P)$;
                    $L_O \leftarrow$ **exchange**$(L_O, \mathcal{G}_{\text{old}}, \mathcal{G})$;
        else
            $L_O \leftarrow L_O \cup$ **branch**$(M^*, \overline{x})$;
Output: Vectors $\hat{x}, \hat{y}$ describing a solution of the TB-VRP instance.

conclude that Branch and Refine fits the branch and bound paradigm for solving $M_{G^{exp}}$ and therefore returns a solution of it.

Note that this algorithm is actually very similar to Iterative Refinement. However, in Iterative Refinement the exploration of the branch and bound search tree is hidden, because we directly use the solutions of the MILPs. So, the exploration of the branch and bound search tree is carried out in every iteration while in Branch and Refine it has to be explored only once. This does not have to be an advantage, since the branching decisions made early on during the exploration of the search tree highly impact the runtimes of branch and bound algorithms. We therefore carried out a computational study, for which we implemented these algorithms.

6 Computational experiments and implementation details

In this section, we compare the two refinement algorithms presented in this work not only to each other, but also to a direct approach of solving the MIQP (1) with a state-of-the-art solver. In order to make the comparison as objective as possible the goal was to find a framework in which all three approaches could be implemented without favoring one approach. The problem here is that not all state-of-the-art solvers are suitable for implementing Branch and Refine. To implement Branch and Refine efficiently the solver would have to not only allow for constraints and variables (i.e., rows and columns) to be added to the representation of the MILP in the storage, but also to change the coefficients of some variables in some of the constraints. This is because after adding nodes some of the arcs used before might not be minimally underestimating anymore. Instead of adding a cutting plane, which sets the associated variables to zero, an efficient implementation should remove the variable from the flow constraint of its old head and add it to the new head. However, non of the state-of-the-art solvers we tried allowed for this kind of change during the solving process, because in general removing a variable from a constraint can in general make previously found bounds invalid. So, we dropped this requirement and decided the aforementioned procedure of adding cutting planes that set these variables to zero and add new variables instead. With this workaround it was possible to implement Branch and Refine in the SCIP Optimization Suite maintained by the Zuse Institute Berlin. For a fair comparison, we also used the SCIP Optimization Suite 6.0.1 to solve the MIQP formulation given in (1) and the MILPs that have to be solved during the Iterative Refinement. Within SCIP the solver SoPlex 4.0.1. was used to solve the linear relaxations at the nodes of the search tree. All experiments were executed on a single core of a computer using a 3.30GHz CPU running Windows 10 as the operating system.

For testing the performance of the implementations, we present an extensive computational study on variety of different instances. All instances we used are based on the benchmark set introduced by Solomon [25]. This set contains instances for the capacitated VRP and is divided into six classes. These classes contain problems with a short or a long scheduling horizon for randomly distributed customers, for clustered customers and for a mixed version of those two. Important to note is that within one class the positions of the customers are fixed and only the time windows are varied. Since the vehicles in the TB-VRP always have to return to the depot, the distance between the customers is not relevant, just the distance to the depot. Therefore, distinguishing between clustered or non-clustered customers is not relevant in our case. In preliminary tests we also observed very little variety of the solutions within the classes of instances. So, we decided to derive new instances from these. We used all unique customer location and time window combinations that could be found in any of the instances and randomly chose 50 different of these to obtain one instance. These instances can be obtained via DOI 10.26127/BTUOpen-5493. The time limit for all instances was 1 hour. One of the goals of the computational

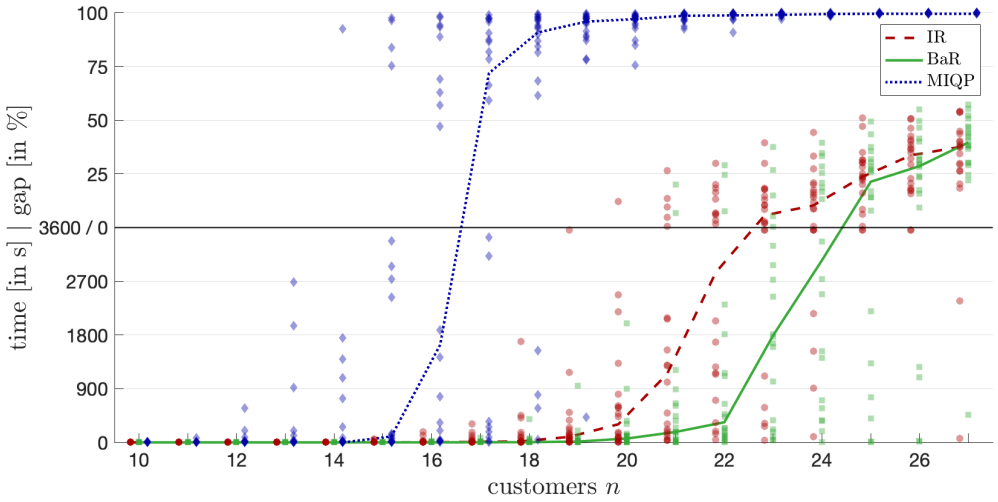


Figure 3: Computation times for different customer numbers and 3 vehicles.

experiments was to be able to evaluate the effectiveness of the three presented solution methods.

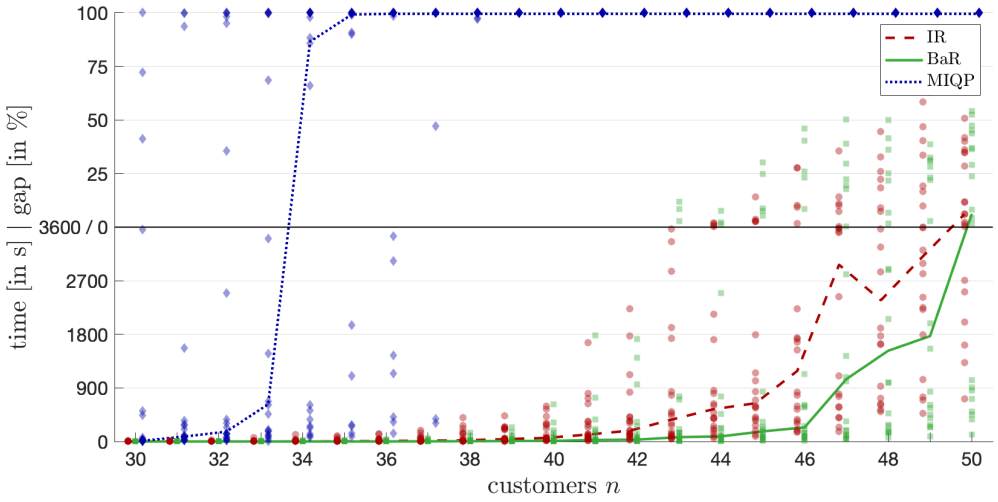


Figure 4: Computation times for different customer numbers and 10 vehicles.

In the first experiment, we study the influence of an increasing number of customers on the computation time, while keeping the number of vehicles fixed. We ran two sets of experiments, one with $m = 3$ vehicles and one with $m = 10$ vehicles. In both cases, the $B = 120$ was used for the battery capacity and the recharging rate α was set to 2.5. For the experiments with 3 vehicles, we started by using the first 10 to 27 customers of 20 different instances. The results of the experiments can be seen in Figure 3, where we introduce the abbreviations IR for Iterative Refinement and BaR for Branch and Refine. For the experiments with 10 vehicles we proceeded analogously, but the lowest amount of customers was 30 and the largest instances contained 50 customers. The results are shown in Figure 4. The x-axes of the graphs in these figures gives the number of customers. In both plots the y-axis is divided into two parts. A lower section for instances that could be solved within 3600 seconds and an upper section in which the gap of the instances that timed out are given. Here, we calculated the gap by dividing the best known lower bound by best known upper bound (if the upper bound is zero there cannot be a gap). This

kind of division of the y-axis is possible, since a non-zero gap and a method solving an instance to proven optimality are mutually exclusive. For each run of a solution method for an instance there is a marker in the plot with a transparency of 40%, whose color and shape indicate the solution method. Additionally we included a line connecting the points representing are the median of the instances for a fixed customer amount (ranked first by solution time and then by the size of the gap). In both figures it is clearly visible that the approach of directly solving an MIQP formulation is not competitive to the approaches presented in this work. In the case of 3 vehicles the MIQP timed out on more than half of the instances for 17 customers, whereas it was still possible to solve all of the instances to optimality for up to 19 customers using the other two approaches. Comparing Iterative Refinement and Branch and Refine for 19 to 24 customers, Branch and Refine is clearly performing better, but for more customers they are almost equally effective and the trend of the medians indicates that Iterative Refinement might even perform better if more customers would be added. Note that only a few instances were solved to global optimality by any of the approaches above 25 customers. Moreover, it should be mentioned that Iterative Refinement is finding smaller gaps within 3600 seconds does not allow the conclusion that an optimal solution can be found faster compared to Branch and Refine.

The results for the experiments with 10 vehicles indicates a similar behavior. From the problem instances with more than 33 customers the majority of the instances remained unsolved within the time limit when using the MIQP formulation. For the other two methods this was not the case for any of the customer amounts. For 40 to 49 customers it is visible that Branch and Refine outperforms Iterative Refinement, which again suggests that the Branch and Refine approach is more effective. The results for 50 customers again suggest, that when more than half of the instances cannot be solved to optimality anymore it is not clear, which algorithms performs better.

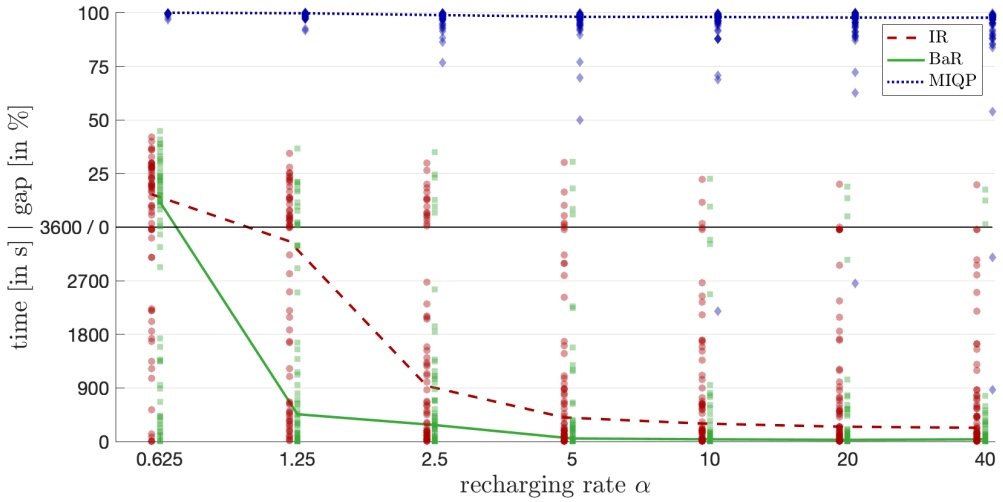


Figure 5: Computation times for different recharging rates.

In another computational experiment we want to study the impact of the recharging rate on the difficulty of the instances. The recharging rate α influences, how much impact the presence of the battery constraints have. For a very small value almost all time is spent charging at the depot, while a large value for α makes charging almost instantaneous, so that the battery constraints imposed almost no restrictions on the routes. The results for the instances with 3 vehicles and customer amounts between 20 and 22 are given in Figure 5. The layout of the plot is very similar to the previous one except that on the x-axis the different values of α used in the experiments are given and a log scale is used.

The results show that the instances become easier to solve for Iterative Refinement and Branch and Refine the larger α is. We do not expect the computation times to become much smaller for larger values than 40, since for this value the routing choices seem to have the largest impact on the delays. It is also observable that Branch and Refine performed better than Iterative Refinement and interestingly this seems to be the case even for instantaneously recharges.

The final set of computational tests was performed to study the influence of the time window width on the computation time. We used instances with 20 to 22 customers, a recharging rate of 2.5 and 3 vehicles. For the time windows we kept the lower bounds $\underline{t}_i$ and then set the due date $\bar{t}_i$ to the lower bound plus a varying constant determining the time window width. The results are given Figure 6 in the same kind of plot as before. Here, the time window width of the instances is given on the x-axis. As expected for the number of customers used in the instances the MIQP formulation was only able to solve very few instances and most had a remaining optimality gap of 100% after one hour. Again it is visible that Branch and Refine performs better than Iterative Refinement. It is worth noting that while Branch and Refine seems to benefit from a larger time window width, Iterative Refinement and also the direct approach for the MIQP formulation performed worst on the instances with the largest time window width.

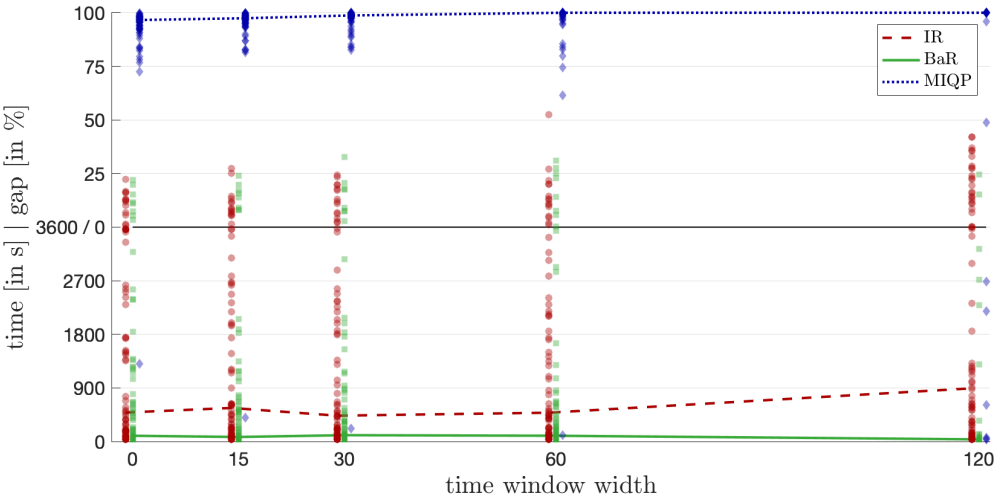


Figure 6: Computation times for different time window widths.

7 Conclusions and future work

We introduced a variant of the VRP, which imposes soft time windows together with constraints concerning the battery consumption of the vehicles. For this problem we proposed two formulations, one MIQP formulation that can be solved directly with state-of-the-art solvers and a formulation based on a two-dimensional layered graph. The latter formulation is not suitable for a direct solution approach, but allows for relaxations, which not only provide lower bounds, but can also be solved in a relatively short time. We used these relaxations in two structurally similar algorithms, Branch and Refine and Iterative Refinement. These algorithms start with a coarse layered graph in which traveling times are underestimated and battery consumption overestimated. This graph is then dynamically refined based on the solutions it provides. The results of the computational study suggest that both of the algorithms outperform the direct approach using an MIQP formulation by a large margin. Furthermore, the results suggest that Branch and Refine which

incorporates graph refinement into a branch and bound framework performs better than Iterative Refinement.

In the future, we want to investigate if similar techniques can be applied in more general settings. A natural extension is to consider multiple depots. If each vehicle is strictly assigned to one depot and can only recharge and pick-up parcels at that depot, it is possible to use a layered graph for each depot. The more difficult, but maybe also more realistic case, is a variant, where the assignment between parcels and depot is not given. In this case, the vehicles would choose a depot to pick up a parcel and decide if charging is necessary. Note, that even if the next customer is known, the obvious choice of going to the depot inducing the shortest detour might not be optimal. The current SoC can be insufficient to reach this depot, while another depot might still be in range. Also, another depot can be closer to the next customer and therefore the maximally possible SoC at the next customer can be increased by using this depot, if there is sufficient time to recharge. An even more complicated situation arises when those aforementioned situations occur simultaneously. In this case visiting two depots can be an optimal choice. In a layered graph approach all these possibilities would somehow have to be incorporated into the topology of the graph.

Another direction is to investigate whether layered graph approaches are effective for the EVRPTW. In this VRP variant, similar to a multi-depot variant of the TB-VRP, the routes between customers are not predetermined, necessitating more sophisticated modeling techniques for obtaining the correct layered graph.

We also see possibilities for improving our methodology, especially Branch and Refine. So far, we used the preimplemented node selection strategy of SCIP. However, this strategy was implemented for branch and cut or branch and price algorithms, since our implementation uses the functions of SCIP in a non-standard way, it would be interesting to investigate if a customized node selection strategy can improve Branch and Refine. Furthermore, we always enforce the graph refinement for all nodes of the search tree. The theory we presented here does not require this, a local refinement is sufficient. Whether local refinements are beneficial in practice, is another interesting topic worth a deeper analysis.

Acknowledgements

The authors Fabian Gnegel and Armin Fügenschuh acknowledge the funding by the German Research Association (DFG) grant number FU 860/1-1. The authors Stefan Schaudt and Uwe Clausen acknowledge the funding by the German Research Association (DFG) grant number CL 318/26-1.

References

[1] Andreas Bärmann, Frauke Liers, Alexander Martin, Maximilian Merkert, Christoph Thurner, and Dieter Weninger. Solving network design problems via iterative aggregation. *Mathematical Programming Computation*, 7(2):189–217, 2015. doi:10.1007/s12532-015-0079-1.

[2] Natashia Boland, Mike Hewitt, Luke Marshall, and Martin Savelsbergh. The continuous-time service network design problem. *Operations Research*, 65(5):1303–1321, 2017. doi:10.1287/opre.2017.1624.

[3] Natashia L Boland and Martin WP Savelsbergh. Perspectives on integer programming for time-dependent models. *Top*, 27(2):147–173, 2019. doi:10.1007/s11750-019-00514-4.

[4] Nils Boysen, Dirk Briskorn, Stefan Fedtke, and Stefan Schwerdfeger. Drone delivery from trucks: Drone scheduling for given truck routes. *Networks*, 72(4):506–527, 2018. doi:10.1002/net.21847.

[5] Nils Boysen, Stefan Schwerdfeger, and Felix Weidinger. Scheduling last-mile deliveries with truck-based autonomous robots. *European Journal of Operational Research*, 271 (3):1085–1099, 2018. ISSN 03772217. doi:10.1016/j.ejor.2018.05.058.

[6] Nils Boysen, Stefan Fedtke, and Stefan Schwerdfeger. Last-mile delivery concepts: a survey from an operational research perspective. *OR Spectrum*, 43(1):1–58, 2020. doi:10.1007/s00291-020-00607-8.

[7] François Clautiaux, Saïd Hanafi, Rita Macedo, Marie-Emilie Voge, and Cláudio Alves. Iterative aggregation and disaggregation algorithm for pseudo-polynomial network flow models with side constraints. *European Journal of Operational Research*, 258(2): 467–477, 2017. doi:10.1016/j.ejor.2016.09.051.

[8] Guy Desaulniers, Fausto Errico, Stefan Irnich, and Michael Schneider. Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, 64(6):1388–1405, 2016. doi:10.1287/opre.2016.1535.

[9] Tomislav Erdelić and Tonči Carić. A survey on the electric vehicle routing problem: Variants and solution approaches. *Journal of Advanced Transportation*, 2019:1–48, 2019. doi:10.1155/2019/5075671.

[10] Fabian Gnegel and Armin Fügenschuh. An iterative graph expansion approach for the scheduling and routing of airplanes. *Computers & Operations Research*, 114:104832, 2020. ISSN 0305-0548. doi:10.1016/j.cor.2019.104832.

[11] Luis Gouveia, Markus Leitner, and Mario Ruthmair. Layered graph approaches for combinatorial optimization problems. *Computers & Operations Research*, 102:22–38, 2019. ISSN 0305-0548. doi:10.1016/j.cor.2018.09.007.

[12] Edward He, Natashia Boland, George Nemhauser, and Martin Savelsbergh. A dynamic discretization discovery algorithm for the minimum duration time-dependent shortest path problem. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 289–297. Springer, 2018. doi:10.1007/978-3-319-93031-2_21.

[13] Gerrit Heinemann. *Der neue Online-Handel*. Springer Fachmedien Wiesbaden, 2018. doi:10.1007/978-3-658-20354-2.

[14] Pitney Bowes Inc. Pitney bowes parcel shipping index reports continued growth as global parcel volume exceeds 100 billion for first time ever, 2020. URL `http://news.pb.com/article_display.cfm?article_id=5958`. Accessed April 2021.

[15] Merve Keskin and Bülent Çatay. Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 65:111–127, 2016. ISSN 0968-090X. doi:10.1016/j.trc.2016.01.013.

[16] Felix Kreuz and Uwe Clausen. *Einsatzfelder von eLastenrädern im städtischen Wirtschaftsverkehr*, pages 323–333. Springer Fachmedien Wiesbaden, Wiesbaden, 2017. ISBN 978-3-658-18613-5. doi:10.1007/978-3-658-18613-5_20.

[17] F. Marra, G. Y. Yang, C. Træholt, E. Larsen, C. N. Rasmussen, and S. You. Demand profile study of battery electric vehicle under different charging options.

In *2012 IEEE Power and Energy Society General Meeting*, pages 1–7, 2012. doi:10.1109/PESGM.2012.6345063.

[18] Manuel Ostermeier, Andreas Heimfarth, and Alexander Hübner. Cost-optimal truck-and-robot routing for last-mile delivery. *Networks*, page 1–26, 2021. doi:10.1002/net.22030.

[19] Statista Digital Market Outlook. ecommerce report 2020, 2020. URL https://www.statista.com/study/42335/ecommerce-report/. Accessed April 2021.

[20] Moritz Poeting, Stefan Schaudt, and Uwe Clausen. A comprehensive case study in last-mile delivery concepts for parcel robots. In *2019 Winter Simulation Conference (WSC)*. IEEE, 2019. doi:10.1109/wsc40007.2019.9004811.

[21] Moritz Poeting, Stefan Schaudt, and Uwe Clausen. Simulation of an Optimized Last-Mile Parcel Delivery Network Involving Delivery Robots. In Uwe Clausen, Sven Langkau, and Felix Kreuz, editors, *Advances in Production, Logistics, and Traffic – Proceedings of the 4th Interdisciplinary Conference on Production Logistics and Traffic 2019*, Lecture Notes in Logistics, pages 1–19, Cham, Switzerland, 2019. Springer Nature Switzerland AG. ISBN 978-3-030-13535-5. doi:10.1007/978-3-030-13535-5_1.

[22] Stefan Poikonen, Xingyin Wang, and Bruce Golden. The vehicle routing problem with drones: Extended models and connections. *Networks*, 70(1):34–43, 2017. doi:10.1002/net.21746.

[23] Martin Riedler, Mario Ruthmair, and Günther R Raidl. Strategies for iteratively refining layered graph models. In *International Workshop on Hybrid Metaheuristics*, pages 46–62. Springer, 2019. doi:10.1007/978-3-030-05983-5_4.

[24] Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4): 500–520, 2014. doi:10.1287/trsc.2013.0490.

[25] Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987. doi:10.1287/opre.35.2.254.

[26] Marc-Oliver Sonneberg, Max Leyerer, Agathe Kleinschmidt, Florian Knigge, and Michael H. Breitner. Autonomous unmanned ground vehicles for urban logistics: Optimization of last mile delivery operations. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*. Hawaii International Conference on System Sciences, 2019. doi:10.24251/hicss.2019.186.

[27] M. Grazia Speranza. Trends in transportation and logistics. *European Journal of Operational Research*, 264(3):830–836, 2018. doi:10.1016/j.ejor.2016.08.032.

[28] Daniela Rojas Viloria, Elyn L. Solano-Charris, Andrés Muñoz-Villamizar, and Jairo R. Montoya-Torres. Unmanned aerial vehicles/drones in vehicle routing problems: a literature review. *International Transactions in Operational Research*, 2020. doi:10.1111/itor.12783.

[29] Duc Minh Vu, Mike Hewitt, Natashia Boland, and Martin Savelsbergh. Dynamic discretization discovery for solving the time-dependent traveling salesman problem with time windows. *Transportation Science*, 54(3):703–720, 2020.

[30] Zheng Wang and Jiuh-Biing Sheu. Vehicle routing problem with drones. *Transportation Research Part B: Methodological*, 122:350–364, 2019. ISSN 0191-2615. doi:10.1016/j.trb.2019.03.005.

IMPRESSUM

Brandenburgische Technische Universität Cottbus-Senftenberg
Fakultät 1 | MINT - Mathematik, Informatik, Physik, Elektro- und Informationstechnik
Institut für Mathematik
Platz der Deutschen Einheit 1
D-03046 Cottbus

Professur für Ingenieurmathematik und Numerik der Optimierung
Professor Dr. rer. nat. Armin Fügenschuh

E fuegenschuh@b-tu.de
T +49 (0)355 69 3127
F +49 (0)355 69 2307

Cottbus Mathematical Preprints (COMP), ISSN (Print) 2627-4019
Cottbus Mathematical Preprints (COMP), ISSN (Online) 2627-6100

www.b-tu.de/cottbus-mathematical-preprints
cottbus-mathematical-preprints@b-tu.de
doi.org/10.26127/btuopen-5493