# Using Public Transport in a 2-Echelon Last-Mile Delivery Network

Jeanette Schmidt[*,a], Christian Tilk[a], Stefan Irnich[a]

[a] *Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, D-55128 Mainz, Germany.*

**Abstract**

In this paper, we investigate the integration of public transport services into last-mile delivery networks. The free capacity of an already established public transport system that operates according to a given timetable on predetermined lines is used to carry goods from outside of the city into the city center. Dedicated bus or tram stations of the public transport network serve as satellites. From these satellites, city freighters pick up the goods and deliver them to the final customers. The main contribution is an extensive computational study, in which we consider various scenarios of the shared part of the public transport network and investigate different hierarchical objectives minimizing two of the three key indicators (number of city freighters, routing costs, and number of trips). We quantify the tradeoff between the key indicators for different objectives and the impact of limiting the capacity of public transport vehicles. The key instrument to conduct all these experiments are effective exact and heuristic branch-price-and-cut algorithms that we develop and evaluate in detail. In a further study, we analyse the algorithmic performance and evaluate the cost increase when customer time windows are reduced, e.g., as a reaction to increased customer dissatisfaction.

*Keywords:* routing, city logistics, shared mobility, scheduled lines, branch-price-and-cut

## 1. Introduction

The Covid-19 pandemic has led to an increase in e-commerce freight (United Nations, 2021; OECD, 2020). For example, over 4 billion deliveries had to be handled in Germany in 2020, which is an increase of more than ten percent compared to the year 2019 (BIEK, 2021). This growing e-commerce poses a major challenge for logistic service providers, especially when it comes to last-mile delivery. While minimizing logistic costs, carriers have to deal with restrictions imposed by urban development, environmental policies, and operational issues like traffic congestion and strict parking regulation (Zhou *et al.*, 2018).

One of the most common designs for handling and reducing the large number of delivery vehicles going into cities, often delivering small quantities per customer, is to consolidate this fragmented volume at *urban distribution centers* (UDCs) located in cities outskirts (Savelsbergh and van Woensel, 2016). In resulting two-echelon systems, first-echelon vehicles transport goods from the UDC to satellites, at which second-echelon vehicles pick up goods and deliver them to the customers. In this paper, we consider the special case of this system in which the first-echelon utilizes existing public services for passenger transportation, e.g., metro, tram, or bus lines. The resulting two-echelon system is multi-modal and belongs to the group of shared mobility systems (Mourad *et al.*, 2019).

Practitioners have already launched several projects utilizing scheduled services for last-mile deliveries, e.g., “GüterBim” in Vienna, “TramFret” in Saint-Étienne, “Cargo-Tram” in Zurich, and the “LastMileTram” in Frankfurt (Schocke *et al.*, 2020). While effective engineering solutions have been developed for the physical design of the system (dedicated compartments, trailers, transport boxes etc.), the operational planning for

[*]Corresponding author.

*Email addresses:* sjeanett@uni-mainz.de (Jeanette Schmidt), tilk@uni-mainz.de (Christian Tilk), irnich@uni-mainz.de (Stefan Irnich)

such systems is rarely studied. In this work, we therefore introduce the *last-mile delivery problem with scheduled lines* (LMDPSL) that models the associated operative planning problem: On the first echelon, scheduled public transport vehicles are used to carry containers with goods from outside of the city into the city center. Dedicated stations of the public transport services are used as satellites. In such a setting, the schedules of first-echelon vehicles are known and fixed, i.e., routes and arrival/departure times are known in advance. The question is here how to best utilize the provided transportation capacities of the first echelon. On the second-echelon, electric vehicles (Schneider *et al.*, 2014) and/or cargo bikes (Anderluh *et al.*, 2019), henceforth called *city freighters* (CFs), pick up containers from the stations and deliver the contained goods to the customers in a multi-trip setting. Paradiso *et al.* (2020) stress the importance of considering multiple trips for city logistics and last-mile delivery. The outstanding prices of space in many big cities (e.g., £14355/$m^2$/$year$ in London, Furmanik, 2019) justify the assumption that temporarily storing goods at stations is not possible. Consequently, the transport of goods must respect exact operation and load synchronization constraints (following the taxonomy of Drexl, 2012). Moreover, our definition of the LMDPSL features a joint limited capacity of certain subsets of stops in terms of the number of containers that can be transferred. This enables the modeling of several practically relevant constraints such as limited space in the public transport vehicle.

The main contribution of the paper at hand is an extensive computational study of the LMDPSL considering possible scenarios of the public transport system (as they occur in different cities) under various objectives (allowing to study also some tactical planning problems in connection with the LMDPSL). We assume hierarchical objectives minimizing two of the three key indicators which are the number of employed CFs, routing costs, and the number of trips. The latter key indicator, the number of trips, is rarely studied in the multi-trip literature, although, from an operational point of view, the start of a new trip can be seen as the most crucial part in the operative process: Public transport vehicle and CF must be perfectly synchronized in time and space taking also capacities into account (Drexl, 2012). Minimizing the number of trips, therefore, reduces the risk that public transport vehicles and CFs miss each other.

The key instrument to conduct experiments and meaningful analyzes is that LMDPSL instances can be solved exactly or with a relatively small optimality gap. For this purpose, we model the LMDPSL as a variant of the vehicle routing problem with time windows and intermediate replenishment. Solution methods for these types of problems are already mature and allow the solution of at least medium-sized instances. We present *branch-price-and-cut* (BPC) algorithms that allow the exact and heuristic solution of the LMDPSL, because they include the most recent and cutting-edge algorithmic components (Costa *et al.*, 2019). Our focus is however *not* the development and description of the BPC algorithms, because their algorithmic components have been well described in several recent works (e.g., Pecin *et al.*, 2017b; Pessoa *et al.*, 2020). For this reason, the description of the BPC algorithms (Section 4) has been written as compact and concise as possible.

The computational study (Section 5) consists of several parts and is not tailored to the setting of a specific city. Instead, our intention was to be rather general and to analyze different standard scenarios. To this end, we create four different scenarios for the shared part of the public transport network. In these scenarios, the scheduled public transport vehicles operate in lines following a circle and/or cross with a central station in the middle. Two pairs of scenarios are included in each other (restriction and relaxation), which allow studying the possibility to select some lines and stops out of the complete public transport network. Moreover, we investigate four different hierarchical objectives and evaluate the exact and heuristic BPC algorithm using all objectives. A scenario-wise comparison of the different objectives is possible because the same customer sets are used for all scenarios. In addition, we analyze the tradeoff between the key indicators when different objectives are employed. In a further study, the impact of limiting the capacity of public transport vehicles is evaluated. The last study concerns the performance of the exact BPC algorithm and the impact on the different objectives when customer time windows are reduced (announced delivery time windows are often large and can span half of a day), e.g., as a reaction to increased customer dissatisfaction.

The remainder of the paper is structured as follows: We review the pertinent literature in Section 2. Section 3 formally introduces the LMDPSL. In Section 4, we present the BPC algorithms including the column-generation procedure, valid inequalities, and the branching scheme. The results of an extensive computational study are presented in Section 5. The paper closes with final conclusions drawn in Section 6.

## 2. Literature Review

The LMDPSL is a special case of the *two-echelon vehicle routing problem* (Cuda *et al.*, 2015) and belongs to the class of *vehicle routing problems with intermediate stops* (see Schiffer *et al.*, 2019, for an overview). We focus on the operational planning perspective and only review the literature on integrating freight transportation and public transportation. The systematic literature review by Elbert and Rentschler (2021) covers the pertinent literature on qualitative approaches including also important quantitative works.

Masson *et al.* (2015) study the *mixed urban transportation problem* (MUTP) which is a special case of the LMDPSL. They propose a *mixed-integer programming* (MIP) formulation and an *adaptive large neighborhood search* (ALNS). The latter is tested on 15 real-world instances of the city La Rochelle (France) that contain up to 303 customers. Compared to the LMDPSL, the MUTP has the following three limitations: (1) only a single bus line is considered, (2) a restricted container capacity constraint bounds the number of containers that can be unloaded simultaneously at each station, and (3) the MUTP considers one objective only (minimizing the number of CFs first and the routing cost second).

Another problem related to the LMDPSL is the *pickup and delivery problem with time windows and scheduled lines* (PDPTW-SL, see Ghilas *et al.*, 2016b), which generalizes the pickup and delivery problem. The PDPTW-SL allows indirect shipments where a request is picked up at its origin location by a CF and brought to a public transport station. From there the request is carried to another station with a public transportation service. Ghilas *et al.* (2016b) introduced the term *scheduled line* for this service. Finally, the request is picked up at the station by another CF that delivers it to its destination location. Ghilas *et al.* propose an arc-based MIP formulation to solve the PDPTW-SL. The MIP solver CPLEX admits the solution of small-sized instances with up to 11 requests and two scheduled lines proving optimality within a time limit of 24 hours. To achieve better results more quickly, Ghilas *et al.* (2016a) developed an ALNS in a follow-up publication. The ALNS allows the consideration of larger instances and is significantly faster than the direct MIP-based approach. For almost all of the small instances, it finds an optimal solution within an average computation time of just two seconds. Larger instances with up to 100 requests are solved in less than 40 minutes. A later study by Ghilas *et al.* (2016c) extends the ALNS and embeds it into a sample average approximation framework for the PDPTW-SL with stochastic demands. Finally, Ghilas *et al.* (2018) present an exact branch-and-price algorithm which obtains good results on instances with up to 60 requests.

Mourad *et al.* (2020) employ a similar ALNS-based approach for a stochastic version of the PDPTW-SL in which autonomous pickup-and-delivery robots take the role of the CFs. The problem is modeled as a two-stage stochastic problem with the objective to minimize the overall transportation costs. The proposed sample average approximation method together with the ALNS algorithm can cope with instances with up to 60 requests.

Behiri *et al.* (2018) consider the *freight-rail-transport-scheduling problem* that captures the real-life problem studied in the *Grand Paris project*, in which freight and passenger transport are combined using the rail network of Paris. The authors present a MIP formulation and several heuristics solving instances with up to 150 freight requests shipped on a single rail line.

In all contributions mentioned above, freight and passengers share the same public transport vehicle. In contrast, Ozturk and Patrick (2018) focus on sharing the infrastructure only, i.e., dedicated freight trains and regular passengers trains share the same railway line. Similarly, Fatnassi *et al.* (2015) propose a shared freight and passenger on-demand rapid transit system. They consider passenger and freight rapid transit vehicles that offer specific on-demand transportation services from a departure station to a destination station in the shared network.

## 3. Problem Definition

The LMDPSL can be defined on a directed graph $D = (V, A)$ with vertex set $V$ and arc set $A$. The vertex set $V$ comprises $\{0, 0'\} \cup N \cup H$, where 0 and $0'$ denote copies of the CFs' depot, $N$ the set of customers, and $H$ the set of stops. We explain the latter sets in more detail now.

Let the customer set be $N$. Each customer $i \in N$ has an integer demand $d_i > 0$. Moreover, the service at customer $i$ has to start in the time window $[a_i, b_i]$, where $a_i$ ($b_i$) denotes the earliest (latest) time at which the service can start. The handling time at customer $i$ is given by $s_i \geq 0$.

We assume that a set of public transport vehicles can be used to deliver goods into the city center. These vehicles follow given public transport lines and operate as *scheduled services* a.k.a. *scheduled lines.* There can be one line or several lines, each of them visiting a sequence of several *stations* in circulation. Each of these visits is defined as a *stop*, and we denote by $H$ the set of all stops. Hence, a stop $h \in H$ is defined by a station (the physical location), the given and fixed point in time $e_h$ when the station is reached by the public transport vehicle, and a handling time $s_h$ describing how long a CF must stay when it is replenished by this stop. For the sake of simplicity, we associate the time window $[a_h, b_h] = [e_h, e_h]$ with the stop $h \in H$. For the sake of convenience, we define $d_h = 0$ for all $h \in H$.

Goods are transported in containers, each with capacity $Q$ (measured in the same unit as customers' demands). Subsets $\bar{H} \subset H$ of stops can have a joint limited capacity in terms of the number of containers that can be transferred. Such general joint capacities allow to model several practically relevant constraints: Limited space regarding the number of containers to transport in a public transport vehicle can be considered by the subset of all stops within one circulation. Moreover, limited space and time may impose that only a limited number of containers can be unloaded at a stop. Let $\mathcal{H}$ be the set of all capacity-constrained sets of stops, i.e., each $\bar{H} \in \mathcal{H}$ imposes that not more than $C_{\bar{H}}$ containers (the capacity) can be transferred at all stops $h \in \bar{H}$ together.

A homogeneous fleet of $K$ CFs is stationed at the depot, i.e., each CF starts its route from the depot 0 and performs one or several *trips.* A trip consists of picking up goods from a stop $h \in H$ and delivering them to some customer(s). The length of such a *trip* is limited by the time windows and the capacity of the CF. We assume that there are no storage possibilities at a stop $h$ so that the CFs cannot arrive later than time $b_h$ at the stop. The capacity of a CF is one container. After completing one trip, the CF can either perform another trip starting from a different stop, or return to the destination depot $0'$. The start depot 0 and destination depot $0'$ have a time window spanning the whole planning horizon.

Arcs represent possible movements of the CFs (we do not model the movement of public transport vehicles because they have a fixed line and schedule). The arc set $A$ contains the following five types of arcs $(i, j) \in A$:

- from the origin depot $i = 0$ to stop vertices $j \in H$;
- from stop vertices $i \in H$ to customer vertices $j \in N$;
- between customer vertices $i, j \in N$, $i \neq j$;
- from customer vertices to stop vertices, i.e., $i \in N$ and $j \in H$; and
- from customer vertices $i \in N$ to the destination depot $j = 0'$.

A routing cost $c_{ij}$ and a non-negative travel time $t_{ij}$ (including the handling time $s_i$ at $i$) are associated with each arc $(i, j) \in A$. Time-window infeasible arcs can be filtered out so that $A \subset \{(i, j) \in V \times V : a_i + t_{ij} + \leq b_j\}$ is assumed in the following.

The task of the LMDPSL is to determine a cost-minimal set of at most $K$ feasible multi-trip routes such that each customer is served exactly once. We consider three *key indicators* (i.e., number of CFs, routing cost, and number of trips) and get four different hierarchical *objectives*:

O1: minimize the number of CFs first and the routing costs second;
O2: minimize the routing costs first and the number of CFs second;
O3: minimize the number of trips first and the number of CFs second; and
O4: minimize the number of trips first and the routing costs second.

The four objectives can be represented in a unified manner by defining *arc costs* in the following way: Routing costs are multiplied with $\alpha$. Arcs leaving the depot include the fixed cost $\beta$ to account for the number of CFs in use. Likewise, arcs leaving a stop include the fixed cost $\gamma$ to account for the number of trips performed. Hence, the cost of an arc $(i, j) \in A$ is defined as:

$$c_{ij} = \begin{cases} \alpha \cdot d_{ij} + \beta, & \text{if } i = 0 \\ \alpha \cdot d_{ij} + \gamma, & \text{if } i \in H \\ \alpha \cdot d_{ij}, & \text{otherwise} \end{cases}$$

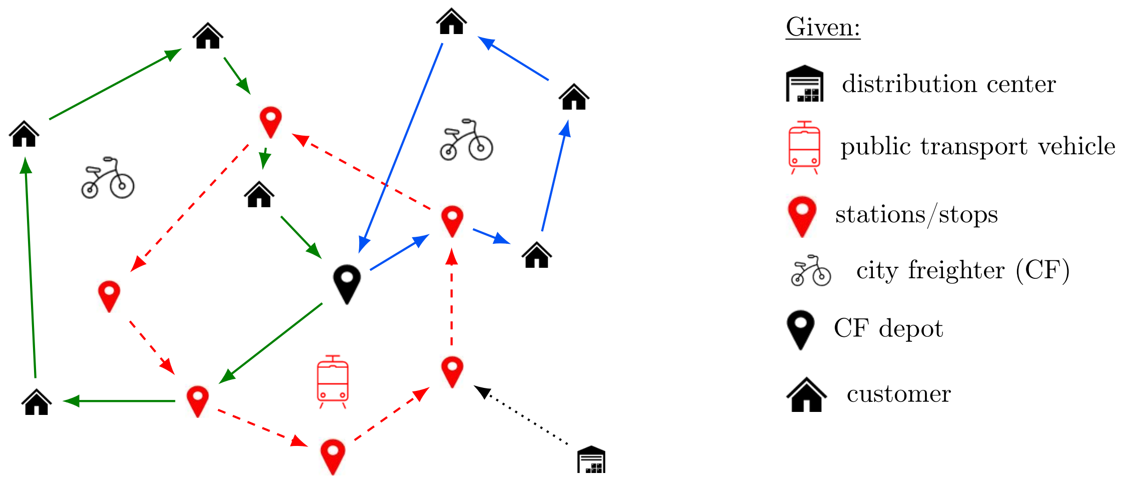


Figure 1: Example of an LMDPSL instance with solution

Depending on the objective, appropriate values for $\alpha$, $\beta$, and $\gamma$ can be chosen to obtain a proper hierarchy. For example, Objective O3 requires $\gamma \gg \beta \gg \alpha$.

Next, we formally define trips and routes. A *trip* $\tau$ is a pair of a path $P = (j_0, j_1, \ldots, j_k)$ in $D$ and a time schedule $(T_0, T_1, \ldots, T_k)$ with $k \geq 1$. A trip is feasible, if the following conditions hold:

(Trip1): $j_0 \in H$;
(Trip2): $j_i \in N$ for all $i \in \{1, \ldots, k\}$;
(Trip3): $T_i \in [a_{j_i}, b_{j_i}]$ for all $i \in \{0, 1, \ldots, k\}$;
(Trip4): $T_i + t_{j_i, j_{i+1}} \leq T_{i+1}$ for all $i \in \{0, 1, \ldots, k-1\}$; and
(Trip5): $\sum_{i=1}^{k} d_{j_i} \leq Q$.

These conditions ensure that each trip starts at a stop, visits some customer(s) afterwards, respects travel times and customer time windows as well as the capacity of the CF. The cost of a trip is defined as the sum of the cost of all arcs traversed, i.e., $c_\tau = \sum_{i=0}^{k-1} c_{j_i, j_{i+1}}$.

A *route* $r$ is a path $(0, P^1, \ldots, P^m, 0')$ with subpaths $P^v = (j_0, j_1, \ldots, j_{k_v})$ together with schedules $((T^1), \ldots, (T^m))$ such that each $\tau^v = (P^v, (T_i^v)_{i=0}^{k^v})$ is a feasible trip for $v \in \{1, 2, \ldots, m\}$. The route $r$ is feasible if the entire schedule is consistent, i.e.,

(Route1): $a_0 + t_{0, j_0^1} \leq T_0^1$;
(Route2): $T_{k^v}^v + t_{j_{k^v}^v, j_0^{v+1}} \leq T_0^{v+1}$ for all $v \in \{1, \ldots, m-1\}$; and
(Route3): $T_{k^m}^m + t_{j_{k^m}^m, 0'} \leq b_{0'}$.

The cost $c_r$ of a route is defined as the cost of all trips plus the cost of the connecting arcs, i.e.,

$$c_r = \sum_{v=1}^{m} c_{\tau_v} + \left( c_{0, j_0^1} + \sum_{v=1}^{m-1} c_{j_{k^v}^v, j_0^{v+1}} + c_{j_{k^m}^m, 0'} \right)$$

Figure 1 shows an example of an LMDPSL instance with seven customers and one public transport line. The line starts at the UDC and visits six stations in circulation (depicted with dashed red arcs). Additionally, the figure shows a solution in which two CF routes are performed to serve all customer. The first CF route (depicted in green) is composed of two trips: The first trip picks a container at a station and serves three customer and the second trip picks a container for only one customer from another stop. The second CF route (depicted in blue) serves another three customers with only a single trip that picks up a container at a third stop.

## 4. Route-based Formulation and Branch-Price-and-Cut Algorithm

We denote by $\Omega$ the set of all feasible routes as defined in the previous section. For each vertex $i \in V$ and each route $r \in \Omega$, the integer coefficient $p_{ir}$ indicates how often route $r$ visits vertex $i$. Similarly, for each set $\bar{H} \in \mathcal{H}$ of capacity-constrained stops and each route $r \in \Omega$, we define the capacity consumption of route $r$ as $p_{\bar{H}r} = \sum_{h \in \bar{H}} p_{hr}$. The following route-based formulation of the LMDPSL uses binary variables $\lambda_r$ indicating whether a route $r \in \Omega$ is selected to be part of the optimal solution ($\lambda_r = 1$) or not ($\lambda_r = 0$). The model reads as follows:

$$\begin{aligned}
\min \quad & \sum_{r \in \Omega} c_r \lambda_r && && \text{(1a)}\\
\text{subject to} \quad & \sum_{r \in \Omega} p_{ir} \lambda_r = 1, && \forall i \in N && \text{(1b)}\\
& \sum_{r \in \Omega} p_{\bar{H}r} \lambda_r \leq C_{\bar{H}}, && \forall \bar{H} \in \mathcal{H} && \text{(1c)}\\
& \sum_{r \in \Omega} \lambda_r \leq K && && \text{(1d)}\\
& \lambda_r \in \{0,1\}, && \forall r \in \Omega && \text{(1e)}
\end{aligned}$$

With the definition of the cost $c_r$ from Section 3, the objective (1a) is one of the hierarchical Objectives O1 to O4 depending on the choice of $\alpha, \beta$, and $\gamma$. The set-partitioning constraints (1b) ensure that each customer is served exactly once. The capacity constrains (1c) limit the number of containers (one for each trip) that can be handled at a particular subset of stops. The number of used CFs is bounded by the fleet size in (1d), and the variable domains are given by (1e).

For realistic instances of the LMDPSL, formulation (1) contains a huge number of feasible routes $r \in \Omega$ so that this model can only be solved with specialized techniques, e.g., a BPC algorithm (Desaulniers *et al.*, 2005). As BPC is a standard approach for solving vehicle routing problems (Costa *et al.*, 2019), we only sketch its main algorithmic components: the generation of route variables (Section 4.1), the strengthening of the linear relaxation of model (1) with non-robust subset-row inequalities (Section 4.2), and the branching scheme (Section 4.3).

### *4.1. Column Generation*

Let $(\pi_i)_{i \in N}$ be the dual prices of constraints (1b), $(\kappa_{\bar{H}})$ be the dual prices of constraints (1c), and $\mu$ be the dual price of (1d). Given these values, the pricing problem must generate at least one feasible negative reduced-cost route $r$ or prove that no such route exists. The reduced cost of an arbitrary route $r$ is

$$\tilde{c}_r = c_r - \sum_{i \in N} a_{ir} \pi_i - \sum_{\bar{H} \in \mathcal{H}} p_{\bar{H}r} \kappa_{\bar{H}} - \mu.$$

The pricing problem can be modeled as a *shortest-path problem with resource constraints* (SPPRC, Irnich and Desaulniers, 2005) and solved by means of a labeling algorithm. To this end, we define reduced costs for all arcs $(i,j) \in A$ as:

$$\tilde{c}_{ij} = \begin{cases} c_{ij} - \pi_i, & \text{if } i \in N \\ c_{ij} - \mu, & \text{if } i = 0 \\ c_{ij} - \sum\limits_{\bar{H} \in \mathcal{H}: i \in \bar{H}} \kappa_{\bar{H}}, & \text{if } i \in H \end{cases}$$

In our labeling algorithm for the LMDPSL, a forward label $\mathcal{L}_i$ represents a forward partial path $(0, \ldots, i)$ in $D$ starting at the origin depot 0 and ending at some vertex $i \in V$. A label $\mathcal{L}_i$ comprises the following attributes:

$T_i^{rdc}$: the cumulated reduced cost along the partial path;

$T_i^{load}$: the cumulated demand of the current trip including vertex $i$;
$T_i^{time}$: the earliest start time of service at vertex $i$; and
$T_i^{cust,k}$: the number of times a customer $k \in N$ is served in the path.

Starting from an initial label $\mathcal{L}_0 = (T_0^{rdc}, T_0^{load}, T_0^{time}, (T_0^{cust,k})_{k\in N}) = (0, Q, a_0, \mathbf{0})$, the labeling algorithm propagates labels towards the destination depot $0'$ with the help of *resource extension functions* (REFs, Irnich, 2008). We set the load resource $T_0^{load} = Q$ to ensure that the CF visits a stop prior to a customer. An arbitrary label $\mathcal{L}_i = (T_i^{rdc}, T_i^{load}, T_i^{time}, (T_i^{cust,k})_{k\in N})$ is extended along an arc $(i,j) \in A$ creating a new label $\mathcal{L}_j = (T_j^{rdc}, T_j^{load}, T_j^{time}, (T_j^{cust,k})_{k\in N})$ for the partial path $(0,\ldots,i,j)$ using the following REFs:

$$T_j^{rdc} = T_i^{rdc} + \tilde{c}_{ij} \tag{2a}$$

$$T_j^{load} = \begin{cases} T_i^{load} + d_j, & \text{if } j \in N \\ 0, & \text{otherwise} \end{cases} \tag{2b}$$

$$T_j^{time} = \max\{a_j, T_i + t_{ij}\} \tag{2c}$$

$$T_j^{cust,k} = \begin{cases} T_i^{cust,k} + 1, & \text{if } j = k \in N, \\ T_i^{cust,k}, & \text{otherwise} \end{cases}, \qquad \forall k \in N \tag{2d}$$

The new partial path is feasible if the label fulfills $T_j^{time} \le b_j$, $T_j^{load} \le Q$, and $T_j^{cust,k} \le 1$ for all $k \in N$ (time-window, capacity, and elementarity constraints). Infeasible labels are directly discarded. To ensure that the CF performs feasible multi-trip routes, (2b) resets the load every time a stop is visited, i.e., when a new trip starts.

To avoid enumerating all feasible paths, provable redundant labels are eliminated through a dominance procedure. Note that we can apply the following standard dominance rules, since all REFs are non-decreasing (Irnich, 2008): A label $\mathcal{L}_1 = (T_1^{rdc}, T_1^{load}, T_1^{time}, (T_1^{cust,k})_k)$ dominates another label $\mathcal{L}_2 = (T_2^{rdc}, T_2^{load}, T_2^{time}, (T_2^{cust,k})_k)$, if $T_1^{rdc} \le T_2^{rdc}$, $T_1^{load} \le T_2^{load}$, $T_1^{time} \le T_2^{time}$, and $T_1^{cust,k} \le T_2^{cust,k}$ holds for all $k \in N$. A dominated label can be discarded as long as at least one dominating label is kept.

To accelerate pricing, we implement several standard techniques such as bidirectional labeling (Righini and Salani, 2006), $ng$-path relaxation (Baldacci *et al.*, 2011), and partial pricing (Gamache *et al.*, 1999). These techniques are briefly summarized in the following paragraphs:

*Bidirectional Labeling.* Bidirectional labeling consists of propagating labels in forward and backward direction up to a half-way point. Afterwards, a merge procedure is applied to combine suitable forward and backward labels to feasible routes.

A backward label $\mathcal{L}'_j = (T_j^{\prime,rdc}, T_j^{\prime,load}, T_j^{\prime,time}, (T_j^{\prime cust,k})_{k\in N})$ represents a backward partial path $(j,\ldots,0')$ in $D$ starting at some vertex $j \in V$ and ending at the destination depot $0'$. Like in many other time-window constrained problems, the time attribute $T_j^{\prime,time}$ of a backward label represents the latest start of service at vertex $j$. The other attributes have the same meaning as for a forward label. The initial backward label for the trivial backward path is $\mathcal{L}'_{0'} = (T_{0'}^{\prime,rdc}, T_{0'}^{\prime,load}, T_{0'}^{\prime,time}, (T_{0'}^{\prime cust,k})_{k\in N}) = (0, 0, b_{0'}, \mathbf{0})$. An arbitrary backward label $\mathcal{L}'_j = (T_j^{\prime,rdc}, T_j^{\prime,load}, T_j^{\prime,time}, (T_j^{\prime cust,k})_{k\in N})$ is extended against the orientation of an arc $(i,j) \in A$ to create a new label $\mathcal{L}'_i = (T_i^{\prime,rdc}, T_i^{\prime,load}, T_i^{\prime,time}, (T_i^{\prime cust,k})_{k\in N})$ for the partial path $(i,j,\ldots,0')$ with the help of the following REFs:

$$T_i^{\prime,rdc} = T_j^{\prime,rdc} + \tilde{c}_{ij}$$

$$T_i^{\prime,load} = \begin{cases} T_j^{\prime,load} + d_i, & \text{if } i \in N \\ 0, & \text{otherwise} \end{cases}$$

$$T_i^{\prime,time} = \min\{b_i, T_j^{\prime,time} - t_{ij}\}$$

$$T_i^{\prime cust,k} = \begin{cases} T_j^{\prime cust,k} + 1, & \text{if } i = k \in N \\ T_j^{\prime cust,k}, & \text{otherwise} \end{cases}, \qquad \forall k \in N$$

The new backward partial path is feasible if the label fulfills $T_i'^{,time} \geq a_i$, $T_i'^{,load} \leq Q$, and $T_j'^{cust,k} \leq 1$ for all $k \in N$. Moreover, the following standard dominance rule can be applied: A label $\mathcal{L}_1' = (T_1'^{,rdc}, T_1'^{,load}, T_1'^{,time}, (T_1'^{cust,k})_k)$ dominates another label $\mathcal{L}_2' = (T_2'^{,rdc}, T_2'^{,load}, T_2'^{,time}, (T_2'^{cust,k})_k)$ if $T_1'^{,rdc} \leq T_2'^{,rdc}$, $T_1'^{,load} \leq T_2'^{,load}$, $T_1'^{,time} \geq T_2'^{,time}$, and $T_1'^{cust,k} \leq T_2'^{cust,k}$ holds for all $k \in N$.

Our bidirectional labeling algorithm uses the time resource as critical resource. Forward (backward) labels are only extended if their time resource $T^{time}$ does not exceed ($T'^{,time}$ is greater than) the chosen half-way point $HWP$. Bidirectional labeling is only effective if less labels are generated and processed than in the monodirectional case. Because the number of labels in forward and backward labeling can be unbalanced for an a-priori chosen half-way point, we use a dynamic half-way point $HWP$ as proposed in (Tilk *et al.*, 2017).

The merge procedure considers all vertices $i \in V$ and pairs of a forward label $\mathcal{L}_i = (T_i^{rdc}, T_i^{load}, T_i^{time}, (T_i^{cust,k})_k)$ and a backward label $\mathcal{L}_i' = (T_i'^{,rdc}, T_i'^{,load}, T_i'^{,time}, (T_i'^{cust,k})_k)$ resident at $i$. To avoid generating the same route multiple times, the forward label $\mathcal{L}_i$ must fulfill $T_i^{time} > HWP$ or $i = 0'$. The labels $\mathcal{L}_i$ and $\mathcal{L}_i'$ can be merged, if $T_i^{load} + T_i'^{,load} - d_i \leq Q$, $T_i^{time} \leq T_i'^{,time}$, and $T_i^{cust,k} + T_i'^{cust,k} \leq 1$ for all $k \in N \setminus \{i\}$. The reduced cost of the resulting route $r$ is $\tilde{c}_r = T_i^{rdc} + T_i'^{,rdc}$. Only routes with $\tilde{c}_r < 0$ are solutions to the pricing problem.

*ng-Path Relaxation.* The *ng*-path relaxation (Baldacci *et al.*, 2011) of the elementary SPPRC is a parameterized relaxation defined by neighborhoods $N_i \subset N$ for each vertex $i \in V$. A cycle $(i, i_1, \ldots, i_\ell, i)$ with $\ell \geq 1$ is feasible if $i \notin N_{i_k}$ for some $k \in \{1, 2, \ldots, \ell\}$. By varying the sizes of the neighborhoods $N_i$ one can control the trade-off between the difficulty of the resulting pricing problem and the strength of the LP-relaxation of the master program. We use neighborhoods $N_i$ that contain vertex $i$ (if $i \in N$) and the next 10 closest customer vertices.

*Partial Pricing with Reduced Networks.* Partial pricing (Gamache *et al.*, 1999) describes that (a hierarchy of) heuristics can be used to solve the pricing problem as long as they deliver some negative reduced-cost variables. Partial pricing often helps to accelerate the overall column-generation process. We solve the pricing problem heuristically over reduced networks with up to $\sigma = 5, 10$, and 15 ingoing and outgoing arcs per vertex. Only if all heuristics fail, the pricing problem is solved exactly. Arcs are chosen according to the lowest reduced cost in the current pricing iteration.

## *4.2. Valid Inequalities*

To strengthen the linear relaxation of the master program, Jepsen *et al.* (2008) introduced subset-row inequalities. To reduce the computational burden of solving the resulting pricing problems while keeping the strength comparable, *limited memory subset-row inequalities* (LmSRIs, Pecin *et al.*, 2017a) can be used instead. We restrict ourselves to subsets of three customers as proposed by Jepsen *et al.* Let $S \subset N$ be a subset of customers with $|S| = 3$. For any route $r \in \Omega$ and any non-negative integer number $h_r^S \leq \sum_{i \in S} p_{ir}$ ($h_r^S$ is a lower bound on the number of times that route $r$ serves a customer in $S$) , the corresponding LmSRI is given by

$$\sum_{r \in \Omega} \left\lfloor \frac{h_r^S}{2} \right\rfloor \lambda_r \leq 1. \tag{3}$$

For a solution of the linear relaxation of the master program, we use the same LmSRI separation algorithm as described by Pecin *et al.*, providing subsets $S$ and a violated associated LmSRI if the coefficients are chosen as $h_r^S = \sum_{i \in S} p_{ir}$. Pecin *et al.* suggest to define a subset $S$-specific vertex memory that ensures $h_r^S = \sum_{i \in S} p_{ir}$ for all routes $r$ that are part of the current solution, i.e., with $\lambda_r^* > 0$ (we distinguish between variables $\lambda_r$ and their values $\lambda_r^*$). The role of the memory is very similar to the neighborhoods in the *ng*-path relaxation (see above). With a limited memory, the difficulty of the pricing subproblem is typically drastically reduced.

LmSRIs are non-robust cuts. Thus, for each SRI with a strictly negative dual value, we add a binary attribute to each label in the labeling algorithm. The dominance rules for forward and backward labeling have to be modified to effectively cope with the additional attributes (we refer to Jepsen *et al.*, 2008; Pecin *et al.*, 2017a, for further details).

During pretest we have identified the following reasonable parameters for the cutting strategy: The total number of LmSRIs to be added is limited to a maximum of 200 inequalities. Moreover, a maximum of 10 inequalities is added per round of separation.

### *4.3. Branching*

Let $\lambda^*_r$ be a fractional solution of the *restricted master program* (RMP). We use the following two-stage branching scheme to finally obtain integer solutions: We first branch on the total number of CFs, whenever $K^* = \sum_{r\in\Omega} \lambda^*_r$ is fractional. We create two branches $\sum_{r\in\Omega} \lambda_r \leq \lfloor K^* \rfloor$ and $\sum_{r\in\Omega} \lambda_r \geq \lceil K^* \rceil$.

The second stage branches on the fractional flows on arcs $(i, j)$. To this end, we compute the value $f^*_{ij} = \sum_{r\in\Omega} f^r_{ij}\lambda^*_r$ for each arc $(i, j) \in A$, where $f^r_{ij}$ denotes the number of times that route $r$ traverses arc $(i, j) \in A$. In a feasible solution, the flow value $f^*_{ij}$ cannot exceed 1 if $i$ or $j$ is a customer vertex due to constraints (1b). The only arcs without a customer vertex are $(0, j)$ with $j \in H$, and they appear only once at the start of each route. Due to flow conservation, if all flow values $f^*_{ij}$ for $(i, j) \in A \cap ((N \times V) \cup (V \times N))$ are binary, then all arc flows are integer. As a result, the decomposition of the integer flows into routes is unique. In conclusion, the branching scheme is complete and branching on arc flows can be implemented as standard binary branching on arcs that contain at least one customer.

The binary arc flow branching decisions can be enforced directly on the underlying network by removing arcs: The zero-branch is implemented by eliminating $(i, j)$ from the arc set $A$, while for the one-branch all ingoing arcs of $j$ and all outgoing arcs of $i$ are eliminated except for the selected arc $(i, j)$. Routes that are incompatible with the modified arc sets are temporarily removed from the RMP.

As an arc selection rule, we apply strong branching with up to eight candidates (Achterberg, 2007). For each candidate arc $(i, j)$, a heuristic evaluation of the lower bounds of its child nodes is performed by using only the first pricing heuristic. We then choose the arc $(i, j)$ that maximizes the minimum of the (estimated) lower bounds of its children. Branch-and-bound nodes are processed in increasing order of the RMP solution value of the parent node (best-first search).

When considering Objectives O3 and O4, in which the number of trips are minimized first, we add an additional branching-stage at the top of the hierarchy: We branch on the number of trips first, i.e., the visits of CFs to a stop vertex. Whenever the number of trips $U^* = \sum_{r\in\Omega} \sum_{h\in H} p_{hr}\lambda^*_r$ is fractional, we branch on it by creating the two branches $\sum_{r\in\Omega} \sum_{h\in H} p_{hr}\lambda_r \leq \lfloor U^* \rfloor$ and $\sum_{r\in\Omega} \sum_{h\in H} p_{hr}\lambda_r \geq \lceil U^* \rceil$.

## 5. Computational Results

The BPC algorithm presented in Section 4 has been implemented in C++ and compiled in release mode into a 64-bit single-thread code under MS Visual Studio Enterprise 2015. The callable library CPLEX 20.1 is used for (re)optimizing the RMPs and as a primal MIP-based heuristic. The time limit for the BPC algorithm is 7200 seconds per instance. CPLEX's default values are kept for all parameters except from setting the number of threads to one. All computations were performed on a standard PC with MS Windows 10 running on an Intel® Core$^{\text{TM}}$ i7-5930K CPU clocked at 3.5 GHz and with 64 GB RAM.

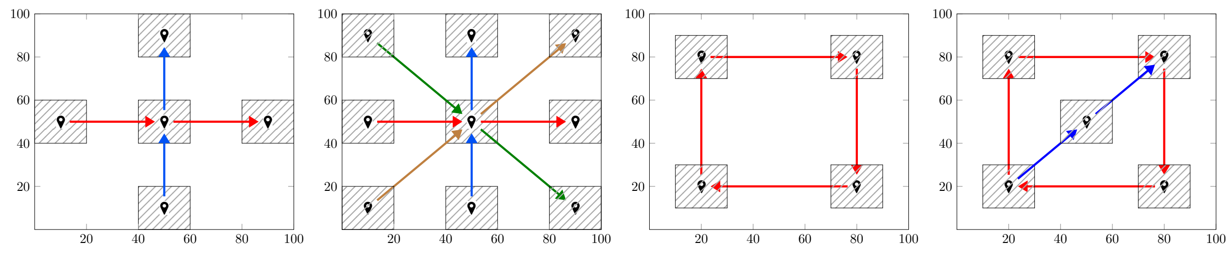
In the following, we introduce the benchmark instances (Section 5.1), present and discuss algorithmic results (Section 5.2), analyze the impact of the different conflicting Objectives O1 to O4, the capacity constraints for subsets of stops, and the increase in the objectives when time windows are reduced (Section 5.3).

### *5.1. Instances*

As no commonly used set of instances is publicly available, we generate a large set of LMDPSL instances by combining four *scenarios* for the public transport network with 60 different *sets of customers*. We consider a $100 \times 100$ grid, on which all physical locations are placed, and a planning horizon of 600 time units (minutes). The four scenarios are (see Figure 2):

S0: two scheduled lines cross the same central station stopping at three stations each;

S1: this scenario extends S0 by adding two more scheduled lines crossing the same central station and stopping at three stations each;
S2: a single scheduled line operating in a circle around the city center stopping at four stations
S3: this scenario extends S2 by adding a single line crossing the central station stopping at three stations.



(a) Scenario S0 (b) Scenario S1 (c) Scenario S2 (d) Scenario S3

Figure 2: Scenarios of the public transport network

Customer locations are randomly distributed over the $100 \times 100$ grid. We generate 20 instances with 50, 100, and 150 customers, respectively. For each customer set, the central station is placed randomly in $[40, 60] \times [40, 60]$. Similarly, the stations are placed randomly in $20 \times 20$ areas as indicated by the shaded rectangles in Figure 2. The depot 0 is located in the center $(50, 50)$ of the grid. It hosts a homogeneous fleet of CFs, each with a capacity of one container which has room for $Q = 10$ packages. The number of CFs in the fleet is assumed unconstraining. Travel times and routing costs for the CFs are computed as Euclidean distances, rounded down to one decimal place.

All customers $i \in N$ have a randomly drawn demand $d_i$ that ranges between 1 and 5 packages. The handling time $s_i$ at each customer varies between 2 and 10 minutes and is independent from the demand. Moreover, a three- or five-hour time window is assigned to each customer. This time window is drawn from the following options $[0, 300]$, $[300, 600]$, $[60, 240]$, $[240, 420]$, and $[420, 600]$ with equal probability.

Table 1 shows the average number $|H|$ of stops and number $U$ of circulation depending on the scenario and number of customers. These values result from the following assumptions: For the 50-customer instances, a single vehicle per scheduled line visits the corresponding stations in several circulations. For instances with 100 (150) customer, we increase the frequency by extending the fleet to 2 (3) vehicles. On lines that cross the central station, containers can only be dropped on the forward run, i.e., there are only three stops per circulation.

| | $\|N\| = 50$ | | $\|N\| = 100$ | | $\|N\| = 150$ | |
|---|---|---|---|---|---|---|
| Scenario | $\|H\|$ | $U$ | $\|H\|$ | $U$ | $\|H\|$ | $U$ |
| S0 | 22.1 | 7.4 | 39.6 | 13.2 | 57.8 | 19.3 |
| S1 | 37.1 | 12.4 | 66.3 | 22.1 | 98.3 | 32.8 |
| S2 | 8.2 | 2.1 | 16.0 | 4.0 | 24.0 | 6.0 |
| S3 | 18.1 | 5.4 | 30.4 | 9.5 | 51.6 | 15.2 |

Table 1: Average number $|H|$ of stops and number $U$ of circulations depending of scenario and size of the customer set.

Depending on the objective, we set the values for $\alpha, \beta$, and $\gamma$ as follows: O1: $\alpha = 1, \beta = 100000, \gamma = 0$; O2: $\alpha = 100, \beta = 1, \gamma = 0$; O3: $\alpha = 1, \beta = 100000, \gamma = 10000000$; O4: $\alpha = 1, \beta = 1, \gamma = 10000$.

The benchmark set comprises a total number of 240 instances and is available at https://logistik.bwl.uni-mainz.de/research/benchmarks.

### *5.2. Algorithmic Results*

We now present algorithmic results on all four objectives introduced in Section 3. Note that the instances do not enforce specific capacities $C_{\bar{H}}$ for subsets $\bar{H} \in \mathcal{H}$ of stops (imposed by constraints (1c)). Since a reasonable value for the stop capacity of, e.g., a circulation, is instance-specific and hard to predict beforehand, we assume infinite stop capacities in this section. In Section 5.3.2 we investigate the impact of binding stop capacities in a separate study.

#### *5.2.1. Exact Branch-Price-and-Cut Algorithm*

The exact solution with the BPC algorithm uses a best-bound-first node selection strategy. It fosters the computation of tight lower bounds as fast as possible. In order to also provide an upper bound for cases when the BPC algorithm terminates without having identified a feasible solution within the given time limit (2 hours), the final RMP is solved as an integer model with the MIP solver. All generated columns until this point are included into the integer model. This MIP-based heuristic has a maximum runtime of 60 seconds so that the total runtime can never exceed 7260 seconds.

Table 2 shows aggregated results of the BPC algorithm and all four Objectives O1 to O4. Results are grouped by number of customers ($|N|$) and scenarios (Scenario S0 to S3). Since no 150-customer instances are solved to optimality, we only present results for 50 and 100 customers. The column entries have the following meaning:

**#Opt**: the number of instances solved to proven optimality within the time limit
**#UB**: the number of instances for which an upper bound was found
**Gap**: the difference between the final upper and lower bound regarding the primary objective, i.e.,

- **Gap$_{\#\mathbf{CF}}$**: the number of times the gap in the number of CFs, comparing upper bound and the lower bound, is zero ($\pm 0$) or exactly one ($+1$), two ($+2$), etc.
- **Gap$_{\#\mathbf{trip}}$**: likewise, for the number of trips, presented in the same way as Gap$_{\#\text{CF}}$
- **Gap$_{\mathbf{costs}}$**: is calculated as $100 \cdot (UB_{costs} - LB_{costs})/LB_{cost}$, i.e., the average gap in percent

*Note:* gaps are computed only for instances for which an upper bound was found
**Time**: the average computing time in seconds
**#BB**: the average number of branch-and-bound nodes solved

The results strongly depend on the objective. We first discuss 50-customer results: For Objectives O2 and O4, all instances are solved exactly, while only 75 and 68 of 80 instances are solved for Objectives O1 and O3. Likewise, average computation times strongly differ, with approximately 3 and 5 minutes for Objectives O2 and O4, but approximately 16 and 32 minutes for Objectives O1 and O3, respectively. Comparing different scenarios, the average computation times for Objectives O1 and O2 (both minimizing the routing cost and the number of CFs) are diametrically opposed: While for Objective O1 instances of Scenarios S0 and S2 are solved faster, for Objective O2 instances of Scenarios S1 and S3 are solved faster than the others. For the Objectives O3 and O4 (both minimizing the number of trips first), Scenario S2 is solved fastest. For the 50-customer instances, all gaps regarding the primary objective are zero except for one instance with Objective O1 and three instances with Objective O3.

For instances with 100 customers, the BPC algorithm is able to solve a few instances to proven optimality, but only for Objectives O2 and O4. Moreover, the complete set of 80 upper bounds is also only computed for Objectives O2 and O4. The analysis of the gaps provides some insights into the practical difficulty of the four objectives: In the cases in which the primary objective is a natural number (the number of routes O1, the number of trips O3 and O4), the gaps tend to increase from Objective O4, over O1, to O3. In particular, rather large gaps can be observed for Objective O3 in combination with Scenarios S0 and S1. In many cases, the BPC algorithm is not able to provide an upper bound. Finally, there is a significant increase in the number of branch-and-bound nodes when comparing Objectives O1 with O2 and O4 with O3.

Summarizing, 50-customer instances can be solved easily in relatively short time, but instances with 100 and more customers still constitute a challenge. For these instances, the minimization of the number of CFs turns out to be much more difficult for the BPC algorithm than the minimization of the routing cost. This is rather counterintuitive, especially, that Objective O3 makes the solution so much harder than Objective O4 (recall that both minimize the number of trips first). Since every feasible solution with Objective O4

| | | Objective O1 | | | | | Objective O2 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\lvert N \rvert$ | Scenario | #Opt | #UB | $\text{Gap}_{\#\text{CF}}$ $(\pm 0/1/2/{>}2)$ | Time | #BB | #Opt | #UB | $\text{Gap}_{\text{cost}}$ | Time | #BB |
| 50 | S0 | 19 | 20 | 19/1/–/– | 801.5 | 94.0 | 20 | 20 | – | 172.8 | 79.7 |
| | S1 | 19 | 20 | 20/–/–/– | 1,068.8 | 91.3 | 20 | 20 | – | 28.6 | 22.7 |
| | S2 | 19 | 20 | 20/–/–/– | 670.1 | 290.1 | 20 | 20 | – | 259.9 | 283.1 |
| | S3 | 18 | 20 | 20/–/–/– | 1,279.8 | 236.6 | 20 | 20 | – | 78.8 | 59.7 |
| | Total/Avg. | **75** | **80** | **79/1/–/–** | **955.1** | **178.0** | **80** | **80** | – | **135.0** | **111.3** |
| 100 | S0 | – | 7 | –/–/ 3/4 | 7,260.0 | 234.6 | – | 20 | 8.62 | 7,260.0 | 906.4 |
| | S1 | – | 9 | 2/3/ 3/1 | 7,260.0 | 190.3 | 2 | 20 | 6.63 | 6,561.1 | 623.0 |
| | S2 | – | 20 | 1/4/13/2 | 7,260.0 | 438.1 | 2 | 20 | 6.04 | 6,761.8 | 1,483.2 |
| | S3 | – | 12 | 1/2/ 4/5 | 7,260.0 | 282.3 | 3 | 20 | 7.36 | 6,692.7 | 884.0 |
| | Total/Avg. | – | **48** | **4/9/23/12** | **7,260.0** | **286.3** | **7** | **80** | **7.16** | **6,818.9** | **974.1** |

| | | Objective O3 | | | | | Objective O4 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\lvert N \rvert$ | Scenario | #Opt | #UB | $\text{Gap}_{\#\text{trip}}$ $(\pm 0/[1;5]/{>}5)$ | Time | #BB | #Opt | #UB | $\text{Gap}_{\#\text{trip}}$ $(\pm 0/1/2/{>}2)$ | Time | #BB |
| 50 | S0 | 18 | 20 | 19/1/– | 2,153.6 | 334.9 | 20 | 20 | 20/–/–/– | 228.9 | 143.9 |
| | S1 | 15 | 20 | 18/2/– | 2,340.5 | 234.4 | 20 | 20 | 20/–/–/– | 538.5 | 225.3 |
| | S2 | 20 | 20 | 20/–/– | 575.7 | 276.2 | 20 | 20 | 20/–/–/– | 61.4 | 59.9 |
| | S3 | 15 | 20 | 20/–/– | 2,709.4 | 1,005.9 | 20 | 20 | 20/–/–/– | 270.2 | 142.7 |
| | Total/Avg. | **68** | **80** | **77/3/–** | **1,944.8** | **462.8** | **80** | **80** | **80/–/–/–** | **274.75** | **142.9** |
| 100 | S0 | – | 12 | –/4/ 8 | 7,260.0 | 207.7 | – | 20 | 2/3/6/9 | 7,260.0 | 773.2 |
| | S1 | – | 7 | –/3/ 4 | 7,260.0 | 139.5 | – | 20 | 2/6/5/7 | 7,260.0 | 549.8 |
| | S2 | – | 19 | –/7/12 | 7,260.0 | 370.6 | 2 | 20 | 4/9/4/3 | 6,617.8 | 1,087.7 |
| | S3 | – | 14 | –/2/12 | 7,260.0 | 192.1 | – | 20 | –/8/9/3 | 7,260.0 | 787.4 |
| | Total/Avg. | – | **52** | **–/16/36** | **7,260.0** | **227.4** | **2** | **80** | **8/26/24/22** | **7,099.5** | **799.5** |

Table 2: Exact branch-price-and-cut algorithm: results for all Objectives O1 to O4

constitutes a feasible solution for Objective O3, one can take the result obtained for Objective O4 in order to minimize the number of trips.

### *5.2.2. Heuristic Branch-Price-and-Cut Algorithm*

In response to the difficulty of providing good feasible solutions for instances with 100 and 150 customers and, in particular, for Objective O3, we design a heuristic version of the BPC algorithm. We omit results for 50-customer instances, since almost all average gaps were zero for the exact BPC. This heuristic version differs only slightly from the BPC algorithm presented in Section 4 in the following components.

On the larger instances, branching on arcs increases lower bounds only very moderately, leading to a large number of branch-and-bound nodes. Therefore, we skip branching on arc flows and, instead, use the MIP solver for the heuristic solution of each branch-and-bound node. The intention is to find promising feasible solutions as soon as possible. The time limit for each call of CPLEX is increased to a maximum of 1800 seconds leading to a total runtime limit of at most 9000 seconds. To emphasize finding high-quality solutions, we set CPLEX's parameter `CPX_PARAM_MIPEMPHASIS` to 5. Moreover, we increase the maximal number of LmSRIs from 200 to 300 and separate a maximum of 30 inequalities per round. Table 3 summarizes the results obtained with the heuristic BPC algorithm by presenting average gaps. As before, averages are taken over different combinations of objective and scenario.

We can directly compare the results for the 100-customer provided in Tables 2 and 3: The heuristic BPC algorithm provides a feasible solution in all cases. Moreover, the gaps decrease significantly for all four objectives. In particular, for Objective O1, the average gap reduces from two CFs to less than one. For Objective O2, the routing cost gap decreases from 7.1% to 4.6% on average, and for Objective O4, the average gap in the number of trips decreases from two to one. As expected, the heuristic BPC algorithm does not solve additional instances to optimality. Table 3 also shows that Objective O3 is still much harder to handle with the BPC algorithm than Objective O4. This is remarkable because both objectives minimize

| $\lvert N\rvert$ | Scenario | Objective O1 gap$_{\#CF}$ ($\pm0/1/2/{>}2$) | Objective O2 gap$_{cost}$ | Objective O3 gap$_{\#trip}$ ($\pm0/[1;5]/{>}5$) | Objective O4 gap$_{\#trip}$ ($\pm0/1/2/{>}2$) |
|---|---|---|---|---|---|
| 100 | S0 | 3/17/–/– | 5.65 | –/18/2 | 4/10/6/– |
| | S1 | 3/17/–/– | 4.24 | –/16/4 | 2/13/4/1 |
| | S2 | 5/15/–/– | 3.81 | –/20/– | 8/11/1/– |
| | S3 | 8/12/–/– | 4.67 | –/19/1 | 5/11/4/– |
| | Total/Avg. | **19/61/–/–** | **4.60** | **–/73/7** | **19/45/15/1** |
| 150 | S0 | –/3/16/1 | 9.49 | –/–/20 | –/–/–/20 |
| | S1 | –/5/15/– | 9.17 | –/–/20 | –/–/1/19 |
| | S2 | –/3/17/– | 7.21 | –/–/20 | –/–/3/17 |
| | S3 | –/4/16/– | 8.75 | –/–/20 | –/–/–/20 |
| | Total/Avg. | **–/15/64/1** | **8.65** | **–/–/80** | **–/–/4/76** |

Table 3: Heuristic branch-price-and-cut algorithm: gaps for Objectives O1 to O4

the number of trips first.

### 5.3. *Managerial Insights*

We now present three analyses on (i) the impact of the objective on the key indicators (number of CFs, routing cost, and number of trips), (ii) the impact of additional capacity constraints for the public transport vehicle, and (iii) the impact of reduced service time windows on the routing costs and the practical difficulty of the instances.

#### 5.3.1. *Impact of the Objectives*

All objectives that we consider minimize two of the three key indicators, i.e., number of CFs, routing cost, or number of trips, with different prioritization. Since the previous section has shown that the results for Objective O3 are less reliable due to larger gaps compared to those of the other objectives, we omit them here. We present aggregated results grouped by number of customers ($|N|$) and the different public transport scenarios (Scenario S0 to S3) in Table 4. Instance size $|N|$ ranges from 50 to 150 customers, always taking the lexicographically best solution found during the exact or heuristic experiments. The columns of Table 4 have the following meaning:

**costs**: the average routing costs
**#CFs**: the average number of CFs
**#trips**: the average number of trips

These value are adjusted disregarding the multipliers $\alpha$, $\beta$, and $\gamma$.

*Comparison of Scenarios.* Table 4 shows that across all objectives and independently from the number of customers, Scenario S1 always has the lowest routing cost. This can be attributed to the fact that Scenario S1 has the highest number of lines and stops. In addition, the stops are evenly distributed throughout the city resulting in short traveling distances for replenishing the CFs.

In contrast, Scenario S2 always has the highest routing cost and number of CFs, which has mainly two reasons: Scenario S2 has the lowest number of stops, and it is the only scenario without a stop at the central station, resulting in long distances for replenishing the CFs. Moreover, Scenario S2 consistently results in the lowest number of trips, while for all other scenarios, the numbers are almost identical.

Figure 3 further aggregates the results of Table 4 over different instance sizes, visualizing only the respective primary objective. With Objective O4 prioritizing the minimization of the number of trips, the differences are hardly visible (see Figure 3c). We conclude that minimizing the number of trips is close to a pure packing problem, and timing aspects are almost negligible for routing a minimum number of CFs.

Note that Scenario S0 is included in Scenario S1, where the latter scenario roughly doubles the number of stations and stops. In comparison, the average number of CFs (Objective O1, see Figure 3a) decreases

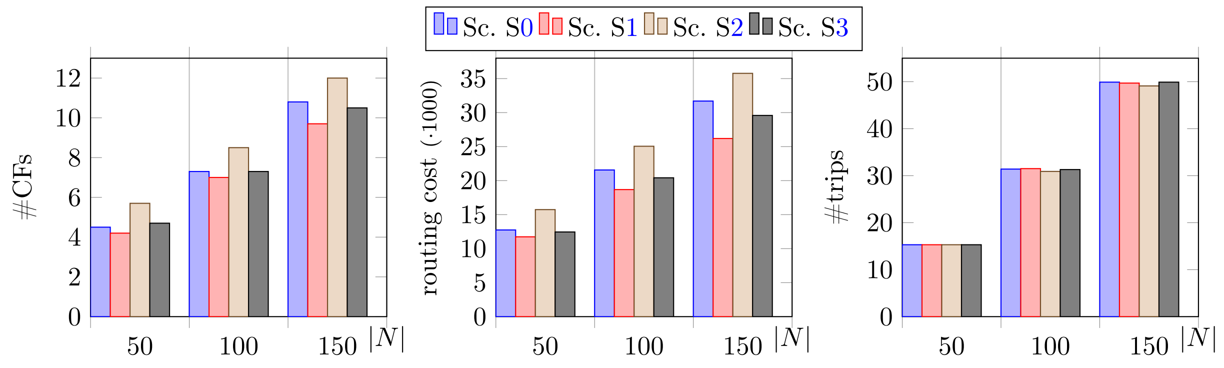
| | | Objective O1 | | | Objective O2 | | | Objective O4 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | (second.) | (prim.) | | (prim.) | (second.) | | (second.) | | (prim.) |
| $\|N\|$ | Scenario | costs | #CFs | #trips | costs | #CFs | #trips | costs | #CFs | #trips |
| 50 | S0 | 14,038 | 4.5 | 16.9 | 12,743 | 6.6 | 16.7 | 12,990 | 6.6 | 15.3 |
| | S1 | 13,051 | 4.2 | 17.1 | 11,737 | 6.4 | 17.1 | 12,200 | 6.5 | 15.3 |
| | S2 | 17,505 | 5.7 | 15.5 | 15,737 | 8.1 | 15.7 | 15,800 | 8.1 | 15.3 |
| | S3 | 13,740 | 4.7 | 16.4 | 12,448 | 7.1 | 16.7 | 12,719 | 7.1 | 15.3 |
| | Avg. | **14,583** | **4.7** | **16.5** | **13,166** | **7.0** | **16.5** | **13,427** | **7.0** | **15.3** |
| 100 | S0 | 25,485 | 7.3 | 37.4 | 21,562 | 10.3 | 34.2 | 22,533 | 11.5 | 31.4 |
| | S1 | 22,147 | 7.0 | 37.4 | 18,682 | 9.7 | 34.2 | 19,978 | 11.0 | 31.5 |
| | S2 | 29,990 | 8.5 | 35.7 | 25,051 | 11.8 | 33.4 | 26,028 | 12.6 | 30.9 |
| | S3 | 24,869 | 7.3 | 37.6 | 20,414 | 10.6 | 34.3 | 21,334 | 11.7 | 31.3 |
| | Avg. | **25,623** | **7.5** | **37.0** | **21,427** | **10.6** | **34.0** | **22,468** | **11.7** | **31.3** |
| 150 | S0 | 37,343 | 10.8 | 60.5 | 31,695 | 14.0 | 53.3 | 32,809 | 15.3 | 49.9 |
| | S1 | 32,443 | 9.7 | 61.4 | 26,192 | 12.9 | 54.3 | 27,565 | 14.5 | 49.7 |
| | S2 | 44,000 | 12.0 | 58.3 | 35,770 | 15.7 | 52.0 | 37,034 | 16.6 | 49.1 |
| | S3 | 36,175 | 10.5 | 60.3 | 29,581 | 14.1 | 53.4 | 30,614 | 15.1 | 49.9 |
| | Avg. | **37,490** | **10.7** | **60.1** | **30,809** | **14.1** | **53.3** | **32,006** | **15.4** | **49.6** |

Table 4: Impact of objectives O1, O2, and O4 on key indicators

by not more than one, the strongest for the 150-customer instances. At the same time, we notice a rather moderate average decrease in routing cost (Objective O2, see Figure 3b) of between 7% and 16% depending on the number of customers.

Likewise, Scenario S2 is included in Scenario S3, where also the latter scenario roughly doubles the number of stations and stops. On average, this leads to larger decreases in routing cost (by around 20%) and number of CFs (also around 20%), which is more significant than in the comparison of Scenarios S0 and S1.

Finally, we compare Scenarios S0 and S3, since they feature a comparable number of stops (see also Table 1). While the number of CFs and trips is nearly identical, Scenario S3 gives slightly smaller routing costs (between 2% and 3.5%) than Scenario S0. We attribute this to the better distribution of the stops in Scenario S3; the four stops in the outskirts of the map are slightly closer to the center.

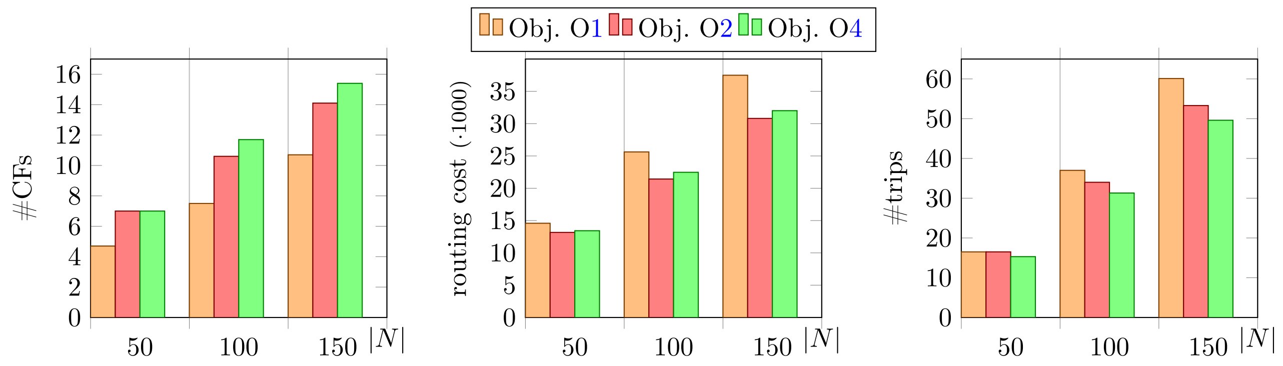


(a) Number of CFs (primary objective) using Objective O1

(b) Routing cost (primary objective) using Objective O2

(c) Number of trips (primary objective) using Objective O4

Figure 3: Comparison of Scenarios S0 to S3 with respect to the respective primary objective



(a) Number of CFs (primary objective in O1)

(b) Routing costs (primary objective in O2)

(c) Number of trips (primary objective in O4)

Figure 4: Comparison of Objectives O1, O2, and O4 with respect to all three key indicators;
*Note:* Objective O1 minimizes the number of CFs first and routing costs second; Objective O2 minimizes the routing costs first and number of CFs second; Objective O4 minimizes the number of trips first and the routing costs second

*Comparison of Objectives.* Finally, we compare the impact of the three Objectives O1, O2, and O4 on the three key indicators (number of CFs, routing cost, and the number of trips). Recall that two of them are the primary or secondary objectives, while the third is not minimized with the respective objective. Figure 4 visualizes aggregated results using the best-found solution considering results obtained with different objectives.

Comparing Objective O1 (minimizing the number of CFs first and the routing cost second) and Objective O2 (primary and secondary objective swapped), the tradeoff between the two becomes clear: Prioritizing the minimization of the number of CFs increases the average routing costs by around 10% (20%) for instances with 50 (100 and 150) customers. Conversely, prioritizing the minimization of the routing costs increases the number of CFs by around two (three) for 50-customer (100- and 150-customer) instances.

Comparing Objectives O2 and O4, with the latter objective the average number of trips can be reduced by approximately 1.3, 2.7, and 3.7 for instances with 50, 100, and 150 customers, respectively. In turn, the routing costs increase by 2% to 5%. The number of CFs takes the largest value with Objective O4, and the number of trips takes the largest value with Objective O1. In conclusion, Objectives O1 and O4 are strongly opposite.

### *5.3.2. Capacity Constraints for Circulations*

We now investigate the impact of limiting capacity for circulations on routing costs, i.e., we upper bound the number of containers that can be transported simultaneously in a scheduled line. The analysis is restricted to Objective O2, because the minimization of routing costs is the primary objective in this case only. Instances defined by all four Scenarios S0 to S3 are compared. Finding a truly constraining capacity is a non-trivial and instance-dependent task that we approach as follows: In a preparatory step, we determine, for each 50- and 100-customer instance individually, the lowest capacity value $C^{min}$ that can be added for all circulations so that this instance remains feasibly solvable. To determine $C^{min}$, we start with the minimal possible value that we determine by considering the sum of customer demands. Then, we successively increase the capacity value until we can find a feasible solution. Note that all 50-customer instances are solved with the exact BPC algorithm, hence, the minimum capacity values $C^{min}$ are exact. In contrast, for some 100-customer instances, the computed minimum capacity values $C^{min}$ might be slightly larger than the minimum, because we apply the heuristic BPC algorithm. The range of the minimum capacity values $C^{min}$ for each scenario is shown in Table 5.

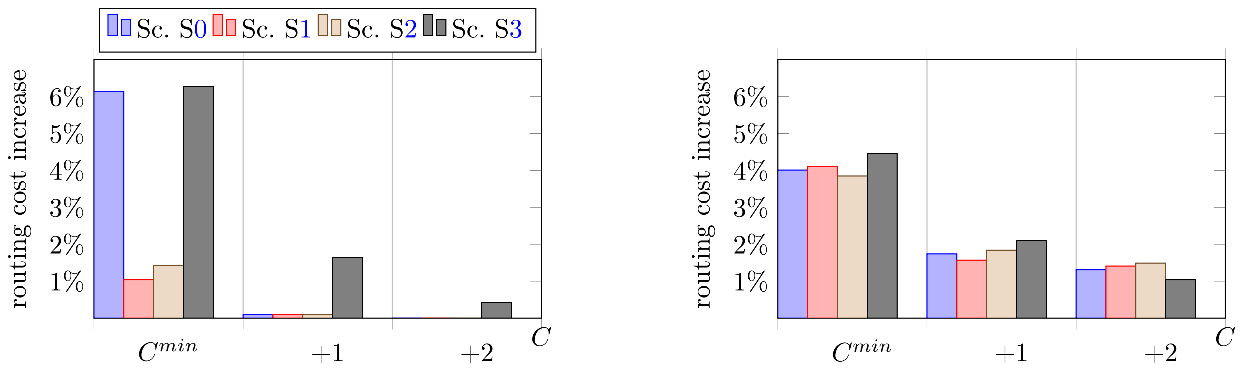
For analyzing the impact of different capacity values, we solve all instances with capacity values $C^{min}$, $C^{min}+1$, and $C^{min}+2$ (the latter two cases are abbreviated with "+1" and "+2"). Figure 5 visualizes the

| Scenario | $\lvert N\rvert = 50$ | $\lvert N\rvert = 100$ |
|---|---|---|
| S0 | $2-3$ | $3-4$ |
| S1 | $1-2$ | $2$ |
| S2 | $6-9$ | $7-10$ |
| S3 | $3-4$ | $3-5$ |

Table 5: Minimum values for circulation capacities

outcome showing the average increase in routing costs compared to the best-found solution with unlimited circulation capacity (see Section 5.2). For the 50-customer instances, Figure 5a clearly shows that the minimum circulation capacity is strongly restricting resulting in an average routing cost increase of up to 6% depending on the scenario. However, choosing a higher capacity (+1 or +2) leads to an almost negligible increase in routing costs with the exception of Scenario S3. It is noticeable that the exact solution for 50-customer instances with a capacity of $C^{min}$ takes on average four times longer than for +1 and +2 (the latter times are comparable to the unrestricted case).

For the 100-customer instances, Figure 5b shows a travel cost increase of around 4% for the minimum circulation capacity $C^{min}$, by nearly 2% for +1, and more than 1% for +2. However, the 100-customer instances are solved heuristically so that it remains unclear whether the exact solution values would give an identical result. Comparing different scenarios, differences are relatively minor, and there is no clear tendency. It seems that not the structure of the transport network but rather the ratio of the sum of customer demands and the overall circulation capacity is the main driver for increased routing costs. In summary, a realistic circulation capacity can be enforced for the analyzed instances leading to only a moderate increase in routing costs.



(a) 50-customer instances

(b) 100-customer instances

Figure 5: Routing cost increase in percent for capacitated circulations using Objective O1

### 5.3.3. *Tightening Time Windows*

For certain customers, e.g., private households, announced delivery time windows of up to 5 hours can be very unsatisfactory. They might be interested in tighter, e.g., one-hour delivery time windows. Moreover, it is generally known that the practical difficulty of many VRP variants increases with the average size of the customer or service time windows (Costa *et al.*, 2019). For mainly this reason, the BPC algorithms for the LMDPSL do often not terminate with a provably optimal solution, in particular, for instances with 100 or more customers.

In the last experiment, we study the impact that customer time windows have on the performance of the exact and heuristic BPC algorithms. For that purpose, we create additional instances derived from the instances of Section 5.1. Therein, the customer time windows are reduced to 60 time units, i.e., a tenth of the planning horizon. For all customers $i \in N$, a new time window $[a_i^{new}, b_i^{new}]$ is calculated as follows:

1. To obtain feasible instances, we first determine a lower bound $\ell_i^{new}$ for $a_i^{new}$, i.e., we estimate the earliest time that customer $i$ can be reached by a CF. Note that we determine customers with identical time windows for all four scenarios, i.e., the precise location and time of the stops is not known at this point.
2. We draw $a_i^{new}$ as a uniformly distributed integer random number from the interval $[\max\{\ell_i^{new}, a_i\}, b_i - 60]$.
3. The new latest possible time $b_i^{new}$ is then set to $a_i^{new} + 60$.

Despite of this careful re-definition, four new 100-customer instances in Scenario S0 are provably infeasible. Therefore, we report the results only for the other 16 feasible instances in this scenario, i.e., we consider 76 instead of 80 instances per objective.

| | | Objective O1 | | | | | Objective O2 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|N\|$ | Scenario | #Opt | #UB | gap$_{\#\text{CF}}$ (±0/1/2) | time | #BB | #Opt | #UB | gap$_{\text{cost}}$ | time | #BB |
| 100 | S0 | 7 | 16 | 8/6/2 | 4,657.5 | 810.8 | 11 | 16 | 0.04 | 3,501.3 | 1,808.1 |
| | S1 | 10 | 20 | 13/5/2 | 4,205.9 | 633.2 | 17 | 20 | 0.01 | 1,285.3 | 470.5 |
| | S2 | 15 | 20 | 17/3/– | 3,132.3 | 907.2 | 17 | 20 | 0.02 | 1,848.4 | 1,427.3 |
| | S3 | 9 | 20 | 13/6/1 | 5,250.7 | 904.4 | 18 | 20 | 0.14 | 1,694.8 | 862.8 |
| | Total/Avg. | **41** | **76** | **51/20/5** | **4,293.4** | **814.0** | **63** | **76** | **0.05** | **2,007.8** | **1,107.1** |

| | | Objective O3 | | | | | Objective O4 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|N\|$ | Scenario | #Opt | #UB | gap$_{\#\text{trip}}$ (±0/[1;5]/>5) | time | #BB | #Opt | #UB | gap$_{\#\text{trip}}$ (±0/1/2/>2) | time | #BB |
| 100 | S0 | 1 | 16 | 1/15/– | 7,013.5 | 1,541.6 | 6 | 16 | 15/1/–/– | 5,659.9 | 3,297.3 |
| | S1 | – | 20 | 1/19/– | 7,260.0 | 1,488.8 | 8 | 20 | 15/5/–/– | 5,215.3 | 2,245.3 |
| | S2 | 9 | 20 | 11/9/– | 5,398.9 | 1,908.0 | 12 | 20 | 20/–/–/– | 4,751.8 | 3,200.3 |
| | S3 | 1 | 20 | 4/14/2 | 6,928.7 | 1,641.4 | 7 | 20 | 15/5/–/– | 5,298.6 | 3,112.0 |
| | Total/Avg. | **11** | **76** | **17/57/2** | **6,631.2** | **1,650.4** | **33** | **76** | **65/11/–/–** | **5,208.8** | **2,946.1** |

Table 6: Exact branch-price-and-cut algorithm: results for 1-hour time windows and all Objectives O1 to O4

Table 6 summarizes the results for instances with 100 customers and all four objectives. Overall, the exact BPC performs much better than on the original instances with wide time windows: In all cases, a valid upper bound is computed. In the best case of Objective O2, 63 out of 76 instances can now be solved to proven optimality, depending on the objective. In cases where no optimal solution was computed, the gap decreases significantly compared to the large time windows:

- For Objective O1, out of the 35 instances not solved to optimality, there are ten cases in which the primary objective is met and the remaining gap refers to the secondary objective, i.e., the travel costs. For the remaining instances, the average gap is one CF, compared to an average gap of two CFs for instances with large time windows.
- For Objective O2, the gap becomes negligible, it improves from 7.16% to 0.05%.
- Similarly, the gap measured in number of trips can be significantly reduced for Objectives O3 and O4. Especially for Objective O4, the gap now differs by at most one trip compared to an average gap of more than two trips on the original instance set.

As a side note: we also tested our exact BPC algorithm on instances with 50 customers. Here, only 50 out of 80 instances are feasible, since there is only one public transport vehicle operating on every line. Especially for Scenario S2, only three instances are feasible. We omit the presentation of these results but mention that all feasible instances were solved to proven optimality within a few seconds on average.

Finally, the reduction of the time windows produces restrictions of the original instances. Hence, the objective values possibly increase when comparing original and new instances. We now quantify this increase restricting ourselves to the 100-customer instances only, because optimality gaps for these instances are acceptably small. For the original instances with wide time windows, we take the best solution value found during the exact and heuristic experiments (Section 5.2). For the new instances, we only apply the exact BPC algorithm. Figure 6 visualizes the average difference (in percent) grouped by scenario and objective:

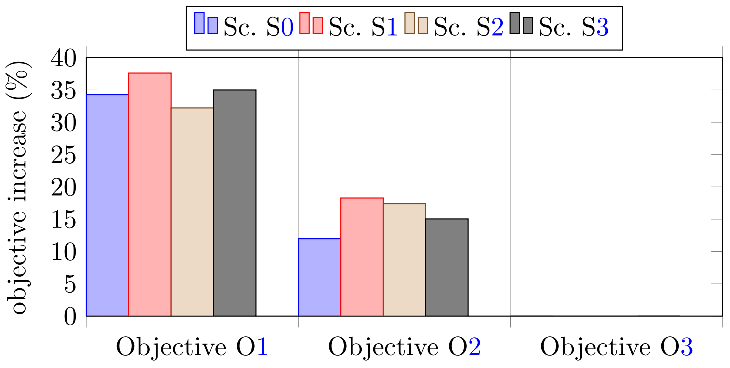


Figure 6: Objective increase in percent when the width of all customer time windows is decreased to one hour

- For Objective O1, the number of CFs increases by at least one for all instances with a maximum increase of five CFs. The average increase is around 2.5 CFs for each scenario.
- For Objective O2, we observe an average increase of nearly 16% in routing costs (minimum is around 8%, maximum is more than 26%).
- Surprisingly, reducing the time windows has no impact on the primary objective of Objective O4, i.e., the minimization of trips.

In total, all instances with tight time windows are either solved to optimality or only a small gap remains, while only seven instances with wide time windows are solved to optimality. As a result, the presented increase of the objective values is rather an underestimate than an overestimate of the true increase. It has also become clear that reducing the time windows (to one hour) leads to a noticeable increase in routing costs and in the number of CFs that need to be employed.

## 6. Conclusions

The paper has introduced the LMDPSL, a prototypical 2-echelon last-mile delivery problem that utilizes existing public transport services, and presented exact and heuristic BPC algorithms to tackle instances of LMDPSL with up to 150 customers. Scenarios reflecting the possible settings found in real-world public transport networks with different hierarchical objectives can be solved exactly or with a relatively small optimality gap with the proposed algorithms. The strong performance of the BPC algorithms did not only allow us to solve the operational planning problems (vehicle routing and scheduling) but also address various questions arising when looking at the system from a more tactical perspective (analyzing tradeoffs and the impact of different assumptions).

The studies reveal that scenarios with a high number of lines and stops (like Scenario S1) almost always have the lowest routing costs and require a smaller number of CFs regardless of the number of customers. Moreover, scenarios with scheduled lines operating in a circle around the city center (like Scenario S2) always lead to the highest routing costs and number of CFs, because a small number of stops or a missing stop at a central station results in longer distances for replenishing the CFs. Our results also show that minimizing the number of trips is close to a pure packing problem, i.e., timing aspects are almost negligible. Under realistic assumptions, the capacity of public transport vehicles is limited. We, therefore, investigated the impact of capacity constraints on circulations and found that they result in only a moderate increase of the total routing costs for the analyzed instances. Finally, reduced time windows typically lead to a noticeable increase in routing costs and the number of CFs employed.

The proposed shared last-mile delivery system offers several benefits such as reduced inner-city traffic, fast and reliable deliveries, and cost savings. For a real-world implementation of the system, operators must clarify several legal and liability issues that arise when people and goods share the same public transport vehicle (Schocke *et al.*, 2020). Any negative effects on travelers must be avoided when goods and people are transported together during their journey (Mourad *et al.*, 2019), e.g., by selecting only lines and stations with a low passenger traffic for the goods transfer.

## Acknowledgement

This research was supported by Deutsche Forschungsgemeinschaft (DFG) under grant IR 122/8-1. This support is gratefully acknowledged.

## References

Achterberg, T. (2007). *Constraint Integer Programming*. Ph.D. thesis, Technische Universität Berlin, Fakultät II – Mathematik und Naturwissenschaften, Berlin, Germany.

Anderluh, A., Hemmelmayr, V. C., and Nolz, P. C. (2019). Sustainable logistics with cargo bikes—methods and applications. In J. Faulin, S. E. Grasman, A. A. Juan, and P. Hirsch, editors, *Sustainable Transportation and Smart Logistics*, chapter 8, pages 207–232. Elsevier.

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

Behiri, W., Belmokhtar-Berraf, S., and Chu, C. (2018). Urban freight transport using passenger rail network: Scientific issues and quantitative analysis. *Transportation Research Part E: Logistics and Transportation Review*, **115**, 227–245.

BIEK (2021). KEP-Studie 2021: Analyse des Marktes in Deutschland: Möglichmacher in bewegten Zeiten. Bundesverband Paket und Expresslogistik e. V. (BIEK) https://www.biek.de/files/biek/downloads/papiere/BIEK_KEP-Studie_2021.pdf, Berlin, Germany (in German).

Costa, L., Contardo, C., and Desaulniers, G. (2019). Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, **53**, 946–985.

Cuda, R., Guastaroba, G., and Speranza, M. G. (2015). A survey on two-echelon routing problems. *Computers and Operations Research*, **55**, 185–199.

Desaulniers, G., Desrosiers, J., and Solomon, M., editors (2005). *Column Generation*. Springer, New York, NY.

Drexl, M. (2012). Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. *Transportation Science*, **46**(3), 297–316.

Elbert, R. and Rentschler, J. (2021). Freight on urban public transportation: A systematic literature review. *Research in Transportation Business & Management*, page 100679.

Fatnassi, E., Chaouachi, J., and Klibi, W. (2015). Planning and operating a shared goods and passengers on-demand rapid transit system for sustainable city-logistics. *Transportation Research Part B: Methodological*, **81**, 440–460.

Furmanik, G. (2019). How much does retail space cost? *realla*. https://blog.realla.co.uk/how-much-does-retail-space-cost.

Gamache, M., Soumis, F., Marquis, G., and Desrosiers, J. (1999). A column generation approach for large-scale aircrew rostering problems. *Operations Research*, **47**(2), 247–263.

Ghilas, V., Demir, E., and Woensel, T. V. (2016a). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. *Computers and Operations Research*, **72**, 12–30.

Ghilas, V., Demir, E., and Woensel, T. V. (2016b). The pickup and delivery problem with time windows and scheduled lines. *INFOR: Information Systems and Operational Research*, **54**(2), 147–167.

Ghilas, V., Demir, E., and Woensel, T. V. (2016c). A scenario-based planning for the pickup and delivery problem with time windows, scheduled lines and stochastic demands. *Transportation Research Part B: Methodological*, **91**, 34–51.

Ghilas, V., Cordeau, J.-F., Demir, E., and Woensel, T. V. (2018). Branch-and-price for the pickup and delivery problem with time windows and scheduled lines. *Transportation Science*, **52**(5), 1191–1210.

Irnich, S. (2008). Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, **30**(1), 113–148.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, pages 33–65. Springer.

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

Masson, R., Trentini, A., Lehuédé, F., Malhéné, N., Péton, O., and Tlahig, H. (2015). Optimization of a city logistics transportation system with mixed passengers and goods. *EURO Journal on Transportation and Logistics*, **6**(1), 81–109.

Mourad, A., Puchinger, J., and Chu, C. (2019). A survey of models and algorithms for optimizing shared mobility. *Transportation Research Part B: Methodological*, **123**, 323–346.

Mourad, A., Puchinger, J., and Woensel, T. V. (2020). Integrating autonomous delivery service into a passenger transportation system. *International Journal of Production Research*, **59**(7), 2116–2139.

OECD (2020). E-commerce in the time of COVID-19. https://www.oecd.org/coronavirus/policy-responses/e-commerce-in-the-time-of-covid-19-3a2b78e8/.

Ozturk, O. and Patrick, J. (2018). An optimization model for freight transport using urban rail transit. *European Journal of Operational Research*, **267**(3), 1110–1121.

Paradiso, R., Roberti, R., Laganá, D., and Dullaert, W. (2020). An exact solution framework for multitrip vehicle-routing problems with time windows. *Operations Research*, **68**(1), 180–198.

Pecin, D., Pessoa, A., Poggi, M., and Uchoa, E. (2017a). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, **9**, 61–100.

Pecin, D., Contardo, C., Desaulniers, G., and Uchoa, E. (2017b). New enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS Journal Computing*, **29**(3), 489–502.

Pessoa, A., Sadykov, R., Uchoa, E., and Vanderbeck, F. (2020). A generic exact solver for vehicle routing and related problems. *Mathematical Programming*, **183**(1-2), 483–523.

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

Savelsbergh, M. and van Woensel, T. (2016). 50th Anniversary invited article—City logistics: Challenges and opportunities. *Transportation Science*, **50**(2), 579–590.

Schiffer, M., Schneider, M., Walther, G., and Laporte, G. (2019). Vehicle routing and location routing with intermediate stops: A review. *Transportation Science*, **53**(2), 319–343.

Schneider, M., Stenger, A., and Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, **48**(4), 500–520.

Schocke, K.-O., Schäfer, P. K., Höhl, S., and Gilbert, A. (2020). *Last mile tram: Empirische Forschung zum Einsatz einer Güterstraßenbahn am Beispiel Frankfurt am Main*. Frankfurt University of Applied Sciences, Frankfurt am Main. (in German).

Tilk, C., Rothenbächer, A.-K., Gschwind, T., and Irnich, S. (2017). Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. *European Journal of Operational Research*, **261**(2), 530–539.

United Nations (2021). COVID-19 and e-commerce: A global review. UNCTAD/DTL/STICT/2020/13 https://unctad.org/system/files/official-document/dtlstict2020d13_en.pdf.

Zhou, L., Baldacci, R., Vigo, D., and Wang, X. (2018). A multi-depot two-echelon vehicle routing problem with delivery options arising in the last mile distribution. *European Journal of Operational Research*, **265**(2), 765–778.