A Matheuristic for a 2-Echelon Vehicle Routing Problem with Capacitated Satellites and Reverse Flows

Dorian Dumez, Christian Tilk, Stefan Irnich, Fabien Lehuédé, Katharina Olkis, Olivier Péton

A Matheuristic for a 2-Echelon Vehicle Routing Problem with Capacitated Satellites and Reverse Flows

Dorian Dumez[2], Christian Tilk*[1], Stefan Irnich[1], Fabien Lehuédé[2], Katharina Olkis[1], and Olivier Péton[2]

[1]Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, 55099 Mainz, Germany.
[2]IMT Atlantique, Laboratoire des Sciences du Numérique de Nantes (LS2N, UMR CNRS 6004), Nantes, France.

October 18, 2021

Abstract

Last-mile collection and delivery services often rely on multi-echelon logistic systems with many types of practical spatial, temporal, and resource constraints. We consider three extensions of the basic 2-echelon vehicle routing problem that are of practical interest: First, second-echelon vehicles need to simultaneously deliver and collect goods at customers within their specified time window. Second, first-echelon vehicles are allowed to perform multiple trips during the planning horizon. Third, the intermediate facilities, called satellites, allow temporary storage of goods, but the quantity that can be stored at a time is limited. This paper integrates these complicating features in a single mathematical model. To solve the problem, we design a decomposition-based matheuristic. It employs a reduced and refined mixed-integer programming formulation and two echelon-specific large neighborhood searches (LNS) to produce improving routes for the respective echelon. The most important algorithmic component is the feasibility check of LNS that relies on a sequence of constant-time and low-complexity tests. The final test allows re-scheduling of the operations taking place at a satellite. It adapts the double-justification algorithm known from the scheduling literature. Extensive computational experiments systematically evaluate the new components of the matheuristic and benchmark it against a recent exact method for a related problem. Moreover, the impact of the main problem features such as the number of satellites and their capacity as well as the integration of forward and reverse flows is analyzed.

1 Introduction

In more and more cities, governmental regulations restrict the number and type of vehicles allowed accessing city centers because of the intense inner-city traffic, narrow streets, and the lack of adequate parking spaces (Muñuzuri *et al.*, 2013). Coordinated urban deliveries can be part of the solution to reduce traffic. We consider modern inner-city distributions systems that rely on a two-echelon structure to reduce the nuisances associated with freight transportation in urban areas while supporting their economic and social development (Crainic *et al.*, 2009).

*Corresponding author.

One of the most common designs for handling and reducing the large number of freight vehicles going into cities, often delivering small quantities per stop, is to consolidate this fragmented volume at *urban distribution centers* (UDCs) located in cities outskirts (Savelsbergh and van Woensel, 2016). In the *two-echelon vehicle routing problem* (2E-VRP), customers are supplied from the UDC through intermediary transfer and storage locations called *satellites*. First-echelon vehicles transport goods from the UDC to satellites, at which second-echelon vehicles collect the goods and deliver them to the customers. The flow of goods must respect operation and load synchronization constraints (following the taxonomy of Drexl, 2012), i.e., one must decide on the assignment of each customer's demand to a satellite and ensure that incoming and outgoing quantities at each satellite coincide.

We have identified three extensions of the basic 2E-VRP as highly relevant. These are: (i) the integration of reverse flows, (ii) the consideration of several timing aspects (including time windows and the possibility to perform multiple trips with the same vehicle), and (iii) satellite capacities. We elaborate their importance in the following.

To reduce traffic, pollution, and noise in city centers, a holistic view onto the logistics system is required including the consideration of goods transportation into as well as out of the city. Bektaş *et al.* (2017) point out that handling forward, reverse, and transiting flows is a key activity within urban areas. In some fields of application, the consideration of forward and reverse flows is therefore almost imperative. Examples include beverage-delivery services (Bruck and Iori, 2017) and package and urban courier services (Wong, 2008), where reverse quantities of some customers can easily exceed their forward quantities.

Typical fleets on the second echelon comprise small trucks (vans and light commercial vehicles) as well as locally emission-free vehicles such as (battery) electric vehicles (Schneider *et al.*, 2014; Desaulniers *et al.*, 2016) and cargo bikes (Elbert and Friedrich, 2020). These vehicles typically have a much smaller capacity than the trucks used on the first echelon. As a result, second-echelon routes are relatively short so that a vehicle can be cleared and replenished several times over a day. With a limited fleet, the associated problem is therefore a *multi-trip VRP* (Paradiso *et al.*, 2020, stress its relevance for city logistics and last-mile delivery). Moreover, when distances between UDC and satellites are short, multiple trips of the first-echelon vehicles are also possible (see, e.g., Nolz *et al.*, 2020).

In their review, Cuda *et al.* (2015) highlight the steadily increasing number of publications on the 2E-VRP. They conclude that mostly, basic versions of the 2E-VRP have been studied and that many practical issues, in particular temporal interdependencies are still to be further investigated both in heuristic and exact approaches. Temporal interdependencies may refer to the requirement that

(1) either first-echelon and second-echelon vehicles have to meet at the satellite in order to directly transfer loads (no storage, *exact operation synchronization* as defined by Drexl, 2012)
(2) or satellite visits of first-echelon vehicles can precede those of second-echelon vehicles (storage at satellites, *operation synchronization with precedences*, see also Drexl, 2012).

Furthermore, satellite capacities impose *resource synchronization* constraints (Drexl, 2012).

For case (1), Grangier *et al.* (2016) introduce the *two-echelon multiple-trip vehicle routing problem with satellite synchronization* (2E-MTVRP-SS) which extends the basic variant by customer time windows and multiple trips on the second echelon. In their variant, there are no storage possibilities at the satellites. For case (2), it is obvious that modeling and solving problems with satellite capacities is much more involved. Only very few works cover this aspect (see literature review in Section 2).

In response to this research gap, we introduce a variant of the 2E-VRP with time windows with a specific focus on incorporating the processes at the satellites in combination with the

requirement to handle both forward and reverse flows (goods and returns). Overall, we extend the 2E-MTVRP-SS of Grangier *et al.* (2016) by three aspects:

- First, customers can have a pickup demand introducing reverse flows into the two-echelon system. As a consequence, second-echelon vehicles need to deliver goods and pick up goods at customers within their specified time window. We assume that the delivery and pickup are performed simultaneously. The pickup demand must be transported to a satellite and from this satellite to the UDC by a first-echelon vehicle.
- Second, also first-echelon vehicles are allowed to perform multiple trips during the planning horizon.
- Third, satellites allow a temporary storage of goods, but they feature a limited capacity in terms of goods that can be stored at a time. The most prevalent situation is probably that goods to be delivered (forward flow) and picked up (reverse flow) are stored together, thus, they share and compete for the satellite capacity.

We introduce the *two-echelon multiple-trip vehicle routing problem with capacitated satellites and reverse flows* (2E-MTVRP-CSRF) that addresses all three extensions together. The 2E-MTVRP-CSRF models the time and quantity dependencies at capacitated satellites in a more precise manner than previous works. It is, therefore, obvious that for 2E-MTVRP-CSRF instances of realistic size, the only viable solution approach is heuristic.

To solve the 2E-MTVRP-CSRF, we propose an algorithm that partially decomposes the problem into three different subproblems. For two of them, we present two *large neighborhood search* (LNS) algorithms, each of them optimizing one echelon while parts of the other echelon are fixed. For the third subproblem, trips found in both LNS algorithms are recombined by solving a restricted and refined version of a trip-based formulation of the 2E-MTVRP-CSRF with a MIP-solver (MIP= mixed integer programming). Overall, the matheuristic controls the interaction between the three subproblem solution methods, i.e., how much computational effort is put into which subproblem and how (possibly infeasible or/and incomplete) solutions are communicated between subproblems. The matheuristic benefits from the MIP-solver-based component that has a global view on satellite-capacity constraints and on the spatial and time synchronization between the first and second echelon.

Our computational studies reveal that the matheuristic can be helpful to quantify the marginal value of satellite capacity, which is worth considering in regard to the high prices per square meter in large cities. Moreover, the matheuristic can be used to estimate the value of integrating forward and reverse flow, which reduces the routing costs by up to 40% in our computational experiments.

The remainder of the paper is structured as follows. Section 2 reviews the 2E-VRP literature focusing on those variants with reverse flows, timing aspects including multiple trips, and satellite capacities. In Section 3, we formally introduce the 2E-MTVRP-CSRF, detail feasibility conditions, and present the new trip-based formulation. Section 4 presents the matheuristic algorithm. Computational results and managerial insights are presented in Section 5, before final conclusions are drawn in Section 6.

2 Literature

The basic 2E-VRP assumes that all transfers between the first and second echelon are always operable, e.g., justified when all transfers are performed during a fixed cut-off period so that all first (second)-echelon vehicles must arrive (depart) before (after) that period. Hence, operation synchronization is *pure spatial* in the basic version (see Drexl, 2012). The first study on the 2E-VRP was conducted by Gonzalez-Feliu *et al.* (2008) who introduced the problem into the

literature. They proposed a commodity-flow model and a *branch-and-cut* algorithm that was tested on instances with up to 50 customers and four satellites. The 2E-VRP was surveyed by Cuda *et al.* (2015). More recent successful solution approaches are exact (Perboli *et al.*, 2018; Marques *et al.*, 2020b) and heuristic (Breunig *et al.*, 2016; Wang *et al.*, 2017; Amarouche *et al.*, 2018).

The standard assumption for the 2E-VRP is that customers are served by a single visit so that demands are not split on the second echelon. However, splitting a customer's demand on the first echelon is an option.

2.1 Reverse flows

Crainic *et al.* (2012) present the integration of forward and reverse flows in a two-echelon distribution system as an important challenge. Belgin *et al.* (2018) were the first to solve a 2E-VRP variant with forward and reverse flows so that a customer can have both a delivery and a pickup demand. The vehicle routing problem at the second echelon is one with *simultaneous deliveries and pickups* (Subramanian *et al.*, 2013). Additionally, Belgin *et al.* model the satellite capacity as the maximum total demand that can be served by the satellite during the planning horizon (we also use the term *total demand* in the following to refer to the total delivery plus pickup demand transferred at a satellite). The problem is solved heuristically by a *variable neighborhood descent* (Hansen *et al.*, 2009). Additionally, they present a two-index model and computationally evaluate the effect of additional valid inequalities on lower bounds. Tests were conducted on an instance set comprising up to 50 customers and five satellites.

2.2 Time windows and multiple trips

Crainic *et al.* (2009) were the first to consider timing aspects in 2E-VRP. The authors discuss customer time windows and multiple trips on both echelons among other features of a general two-tier city logistic systems. The work also presents several modeling ideas and describes possible heuristics without computational testing.

Grangier *et al.* (2016) introduced the 2E-MTVRP-SS which features customer time windows, multiple trips in the second echelon and exact synchronization between first and second echelon vehicles at satellites. They present an LNS algorithm with a fast and sophisticated feasibility check. Computational experiments were conducted on instances with up to eight satellites and 100 customers derived from the VRPTW benchmark instances of Solomon (1987).

Anderluh *et al.* (2017) extend the 2E-MTVRP-SS by allowing the first-echelon vehicle to serve some customers. A two-phase greedy randomized adaptive search procedure with path relinking was applied to solve the extended 2E-MTVRP-SS. Computational tests were conducted on instances with up to 125 customers and 18 satellites.

Dellaert *et al.* (2019) suggest a two-path formulation for the single-trip multi-depot 2E-VRP with time windows and exact synchronization (2E-VRPTW). They assume that the entire demand delivered by a second-echelon vehicle must be provided by a single first-echelon trip. This latter assumption is exploited in a *branch-and-price* (BP) algorithm that enumerates configurations of first-echelon trips. The BP algorithm is capable of exactly solving some instances with up to 100 customers and five satellites within the 3-hour time limit. Mhamedi *et al.* (2020) developed a sophisticated *branch-price-and-cut* (BPC) algorithm for the 2E-VRPTW. The algorithm performs well on the instances generated by Dellaert *et al.* (2019), computing 41 new optimal solutions in the 3-hour time limit. Independently, Marques *et al.* (2020a) present BPC algorithms for solving the 2E-VRPTW and the 2E-MTVRPTW. The latter is a relaxed variant of the 2E-MTVRP-SS in which synchronization with precedences is required (instead of exact

synchronization). They report optimal solutions for nine 2E-MTVRPTW instances and 54 of the 2E-VRPTW instances.

The practical relevance of timing aspects has led to the introduction of several other 2E-VRP variants covering new technologies and real-world features such as electric vehicles (Jie *et al.*, 2019), cargo bikes with swap containers (Mühlbauer and Fontaine, 2021), and multiple commodities provided by different UDCs (Dellaert *et al.*, 2021). Anderluh *et al.* (2020) analyzed and discussed the impact of stochastic travel times on costs.

2.3 Satellite capacity

Two approaches for modeling the satellite capacities are common in the literature. In the first approach, an upper bound on the number of second-echelon vehicles leaving the satellite is imposed (first suggested by Crainic *et al.* (2009) and very recently employed by Mühlbauer and Fontaine (2021)). The second approach limits the total demand that can be served by a satellite (in the unit-demand case, this means limiting the number of served customers, see, e.g., Breunig *et al.*, 2016; Belgin *et al.*, 2018). Both approaches do not consider the dynamics over the planning horizon, i.e., available capacities depending on the goods entering and leaving the satellite over time.

To the best of our knowledge, only three works use a detailed time-dependent capacity model. Li *et al.* (2018) study a 2E-VRP with customer time windows, a given assignment of customers to satellites, and real-time transshipment capacities. The real-time transshipment capacities define the currently available capacity of the satellite as the difference between the maximum capacity and the quantity of goods currently waiting at a satellite to be collected by a second-echelon vehicle. Additionally, the problem features split deliveries on the first echelon and multiple trips on the second echelon. A MIP formulation is provided and a two-stage heuristic based on variable neighborhood search (Hansen *et al.*, 2009) is proposed. Small-scale instances with up to three satellites and five customers per satellite were solved to optimality with a MIP solver within a 4-hour time limit. The two-stage heuristic has been tested on 99 larger instances with up to 30 satellites and 30 customers per satellite. Li *et al.* (2020) slightly extend the problem by varying the maximum transshipment capacity for each time period.

Recently, Nolz *et al.* (2020) considered a two-echelon distribution with a single capacitated satellite, customer time windows, and multiple trips on both echelons. They propose a new MIP model and a three-phase heuristic method which uses population-based metaheuristics and integer programs. Nolz *et al.* evaluate their methods on instances with up to 81 customers generated from real-world data from the city of Vienna.

More generally, time-dependent satellite capacity involves synchronizing routes on shared renewable capacitated resources. To our knowledge, this type of synchronization problem has received little attention in the literature. Most resource constrained routing and scheduling problems stem from sharing vehicles or personnel that need to be routed and scheduled to perform different tasks (Paraskevopoulos *et al.*, 2017; Castillo-Salazar *et al.*, 2016; Fikar and Hirsch, 2017). Several cases are related to loading or unloading unary resources in applications such as forestry (El Hachemi *et al.*, 2011, 2013), construction (Schmid *et al.*, 2009), and public work (Grimault *et al.*, 2017). Grangier *et al.* (2019) consider a VRP with cross-docking with a limited number of docks that can be used simultaneously. Froger *et al.* (2017) integrate a limited number of chargers at charging stations in electric VRPs. In the two latter contributions, each vehicle consumes one unit of the resource per time unit. We are not aware of works that investigate the synchronization of a renewable resource that depends on a transferred quantity, in particular, in a problem with forward and reverse flows.

3 Problem statement

We define the 2E-MTVRP-CSRF by formally describing customers, satellites, fleet, trips, routes, and satellite capacities. The three latter aspects require a detailed explanation provided in Sections 3.1, 3.2, and 3.3. An integer (linear) programming model for the 2E-MTVRP-CSRF is then presented in Section 3.4.

Let N denote the set of customers, where each customer $i \in N$ has a delivery (=forward a.k.a. linehaul) demand q_i^{fw}, or a pickup (=reverse a.k.a. backhaul) demand q_i^{rv}, or both. Both demands have to be fulfilled by a single visit of a second-echelon vehicle. For the visit of customer $i \in N$, a vehicle can wait at the customer before actually providing service, which must start within the time window $[a_i, b_i]$.

Let S denote the set of satellites. We assume that all satellites $s \in S$ are available over the entire planning horizon $\mathbb{T} = \{0, \ldots, \bar{t}\}$, i.e., $[a_s, b_s] = [0, \bar{t}]$. Moreover, each satellite $s \in S$ has a limited capacity C_s^{sat} and a constant transfer time p_s required for transferring goods from one echelon to the other. The interpretation of the satellite capacity and transfer time is further discussed in Section 3.3.

A homogeneous fleet F^1 of first-echelon vehicles is stationed at the UDC o^1. Likewise, a homogeneous fleet F^2 of second-echelon vehicles is housed at the second-echelon depot o^2. All vehicles must start and end their routes at their depot o^1 and o^2, respectively. They are allowed to perform multiple trips. Each first(second)-echelon vehicle has a capacity of Q^1 (Q^2) that has to be shared by forward and reverse demands when transported together.

The objective of the 2E-MTVRP-CSRF is to find cost-minimal sets of feasible first-echelon and second-echelon trips (formally defined in Section 3.1) such that the following constraints hold:

(F1) All customers are visited exactly once by exactly one second-echelon trip.

(F2) The set of first(second)-echelon trips can be combined to at most $|F^1|$ ($|F^2|$) feasible first(second)-echelon routes.

(F3) The forward flow of goods is conserved at each satellite $s \in S$ and each point in time $t \in \mathbb{T}$, i.e., the forward flow that has reached satellite s by first-echelon trips until time $t - p_s$ is at least as large as the forward demands leaving satellite s on second-echelon trips until time t.

(F4) Likewise, the reverse flow of goods is conserved at each satellite $s \in S$ and each point in time $t \in \mathbb{T}$, i.e., the collected demands reaching satellite s on second-echelon trips arriving until time t is at least as large as the reverse flow that has left from the satellite s by first-echelon trips starting until time $t + p_s$.

(F5) The capacities of all satellites are never exceeded, i.e., at any point in time, the quantity of goods stored or processed at a satellite is not larger than the satellite capacity.

Note that flow conservation via (F3) and (F4) allows a customer's demand (either forward or reverse demand) to be split on the first echelon using two or more trips with possibly different vehicles, while this demand cannot be split on the second echelon.

3.1 Trips

Next, we define feasible trips and routes in both echelons with the help of two directed graphs. For the first echelon, let $G^1 = (V^1, A^1)$ be the complete digraph with vertex set $V^1 = S \cup \{o^1\}$ and arc set A^1. Each arc $(i, j) \in A^1$ is associated with a non-negative travel time t_{ij}^1 and a non-negative travel cost c_{ij}^1.

A *first-echelon trip* $h = (P, T, L)$ consists of a closed directed walk $P = (i_0, i_1, \ldots, i_n, i_{n+1})$ in G^1 with $i_0 = i_{n+1} = o^1$ and $i_1, \ldots, i_n \in S$, a *time schedule* $T = (T_0, T_1, \ldots, T_n, T_{n+1}) \in \mathbb{T}^{n+2}$,

and a *loading plan* $((L_1^{fw}, L_1^{rv}), \ldots, (L_n^{fw}, L_n^{rv})) \in ([0, Q^1] \cap \mathbb{Z})^{2\times n}$. The attribute $T_k \in \mathbb{T}$ models the start of the service/operation at vertex i_k for $k \in \{0, 1, \ldots, n+1\}$. Moreover, the attributes L_k^{fw} and L_k^{rv} are the quantities dropped off and collected, respectively, at satellite i_k for $k \in \{1, 2, \ldots, n\}$.

Each pair (i_k, T_k) of vertex and time is denoted as *first-echelon satellite visit*. Note that consecutive visits to the same satellite are allowed within a walk.

A first-echelon trip $h = (P, T, L)$ is *feasible* if the following three conditions hold:

(Tr1) all vertices are visited within the planning horizon:

$$T_k \in [0, \bar{t}] \qquad \text{for all } k \in \{0, 1, \ldots, n+1\},$$

(Tr2) the time schedule is consistent with respect to travel and transfer times:

$$T_k + t^1_{i_k, i_{k+1}} + p_{i_k} \leq T_{k+1} \qquad \text{for all } k \in \{0, 1, \ldots, n\},$$

where we assume a transfer time $p_{o^1} = 0$ for the UDC,

(Tr3) and the load at each vertex does not exceed the capacity of a first-echelon truck:

$$\sum_{k=l+1}^{n} L_k^{\text{fw}} + \sum_{k=1}^{l} L_k^{\text{rv}} \leq Q^1 \qquad \text{for all } l \in \{0, 1, \ldots, n\}.$$

The *cost* c_h^1 of the first-echelon trip h is given by $\sum_{k=0}^{n} c^1_{i_k, i_{k+1}}$.

Similarly, for the second echelon, let $G^2 = (V^2, A^2)$ be the simple and complete digraph with vertex set $V^2 = S \cup N \cup \{o^2\}$ and arc set A^2. Each arc $(i, j) \in A^2$ is associated with a non-negative travel time t^2_{ij} and a non-negative travel cost c^2_{ij}. Service times at customers are integrated into travel times.

A *second-echelon trip* $h = (P, T)$ comprises a (closed or open) walk $P = (i_0, i_1, \ldots, i_n, i_{n+1})$ in G^2 with $i_0, i_{n+1} \in S \cup \{o^2\}$, $i_1, \ldots, i_n \in N$, and a time schedule $(T_0, \ldots, T_{n+1}) \in \mathbb{T}^{n+2}$.

The second-echelon trip $h = (P, T)$ is *feasible* if the following five conditions hold:

(Tr4) the vertices are served within their time windows:

$$T_k \in [a_{i_k}, b_{i_k}] \qquad \text{for all } k \in \{0, 1, \ldots n+1\},$$

(Tr5) the time schedule respects travel times:

$$T_k + t^2_{i_k, i_{k+1}} \leq T_{k+1} \qquad \text{for all } k \in \{0, 1, \ldots, n\},$$

(Tr6) the vehicle capacity is respected at each vertex on the trip:

$$\sum_{k=l+1}^{n} q_{i_k}^{\text{fw}} + \sum_{k=1}^{l} q_{i_k}^{\text{rv}} \leq Q^2 \qquad \text{for all } l \in \{0, 1, \ldots n\},$$

(Tr7) if $i_0 = o^2$, then $q_{i_k}^{\text{fw}} = 0$ must hold for all $k \in \{1, 2, \ldots, n\}$, and

(Tr8) if $i_{n+1} = o^2$, then $q_{i_k}^{\text{rv}} = 0$ must hold for all $k \in \{1, 2, \ldots, n\}$.

The latter two conditions (Tr7) and (Tr8) impose that trips that start (end) at the depot o^2 cannot serve customers that have a positive forward (reverse) demand. The *cost* c_h^2 of the second-echelon trip h is given by $c_h^2 = \sum_{k=0}^{n} c^2_{i_k, i_{k+1}}$.

Note that second-echelon vehicles and trips can start and end at different satellites. In particular, second-echelon trips that do not visit any customers are allowed (i.e., $n = 0$). These *transfer trips* model that a vehicle can change the starting satellite for its next trip.

3.2 Routes

A *first(second)-echelon route* R is a sequence $(h_1, \ldots, h_m)$ of $m \geq 1$ feasible first(second)-echelon trips. For $j \in \{1, 2, \ldots, m\}$, we denote the associated walks by $P_j = (i_{j0}, i_{j1}, \ldots, i_{j,n_j}, i_{j,n_j+1})$ and the time schedules by $T_j = (T_{j0}, T_{j1}, \ldots, T_{j,n_j}, T_{j,n_j+1})$.

A first-echelon route is *feasible* if

(R1) two consecutive trips h_j and h_{j+1} do not overlap in time:

$$T_{j,n_j+1} \leq T_{j+1,0} \qquad \text{for all } j \in \{1, \ldots, m-1\}.$$

A second-echelon route is *feasible* if the following three conditions hold:

(R2) it starts and ends at the second-echelon depot:

$$o^2 = i_{1,0} = i_{m,n_m+1},$$

(R3) consecutive trips have an identical end and start satellite:

$$i_{j,n_j+1} = i_{j+1,0} \in S \qquad \text{for all } j \in \{1, 2, \ldots, m-1\},$$

(R4) consecutive trips do not overlap in time:

$$T_{j,n_j+1} + p_{i_{j,n_j+1}} \leq T_{j+1,0} \qquad \text{for all } j \in \{1, 2, \ldots, m-1\}.$$

In particular, the transfer time $p_{i_{j,n_j+1}}$ in (R4) for $i_{j,n_j+1} \in S \cup \{o^2\}$ can be used to model a mandatory time period needed to clear and load a second-echelon vehicle.

3.3 Flow conservation and satellite capacity

In this section, we show how to check whether a given set of feasible first-echelon and second-echelon routes, including the quantities dropped off and collected by them over time, respect the forward and reverse flows and whether the quantities stored at a satellite exceed the satellite capacity. This shows how the feasibility conditions (F3)–(F5) can be tested.

For that purpose, we denote by Ω^1 the set of all feasible first-echelon trips and by Ω^2 the set of all feasible second-echelon trips. We introduce additional attributes for a first-echelon trip $h = (P, T, L) \in \Omega^1$:

- the quantity dropped off by the trip h at satellite $s \in S$ at time $t \in \mathbb{T}$:

$$\gamma^{\text{fw}}_{h,s,t} = \begin{cases} L^{\text{fw}}_k, & \text{there exists an index } k \in \{1, 2, \ldots, n\} \text{ with } i_k = s \text{ and } T_k = t \\ 0, & \text{otherwise,} \end{cases}$$

- the quantity that trip h collects at satellite $s \in S$ at time $t \in \mathbb{T}$:

$$\gamma^{\text{rv}}_{h,s,t} = \begin{cases} L^{\text{rv}}_k, & \text{there exists an index } k \in \{1, 2, \ldots, n\} \text{ with } i_k = s \text{ and } T_k = t \\ 0, & \text{otherwise.} \end{cases}$$

Similarly, we define for a second-echelon trip $h = (P, T) \in \Omega^2$:

- the quantity that trip h collects from satellite $s \in S$ at time $t \in \mathbb{T}$:

$$\gamma^{\text{fw}}_{h,s,t} = \begin{cases} \sum_{k=1}^{n} q^{\text{fw}}_{i_k}, & i_0 = s \text{ and } T_0 = t \\ 0, & \text{otherwise,} \end{cases}$$

- the quantity that trip h drops off at satellite $s \in S$ at time $t \in \mathbb{T}$:

$$\gamma^{\text{rv}}_{h,s,t} = \begin{cases} \sum_{k=1}^{n} q^{\text{rv}}_{i_k}, & i_{n+1} = s \text{ and } T_{n+1} = t \\ 0, & \text{otherwise.} \end{cases}$$

We also define the *load profile* Λ_t^s of a satellite $s \in S$ as the quantity processed and stored at satellite s at time $t \in \mathbb{T}$.

Before we formalize the feasibility conditions (F3)–(F5), we explain them with the help of an example.

Example 1 *Figure 1 shows a solution for an instance of the 2E-MTVRP-CSRF with two satellites, eight customers, one first-echelon vehicle, and two second-echelon vehicles. Forward and/or reverse demands of each customer are depicted next to the corresponding vertex. We assume that the customers have non-restricting time windows and that travel times and travel costs coincide (depicted on the arcs). Moreover, the capacity of both satellites is* $C_s^{sat} = 10$ *and the transfer time is* $p_s = 1$.

The first-echelon route performs two trips (in blue), while the two second-echelon routes perform four (in red) and three trips (in green). Table 1 shows a feasible time schedule and loading plan for the three routes.

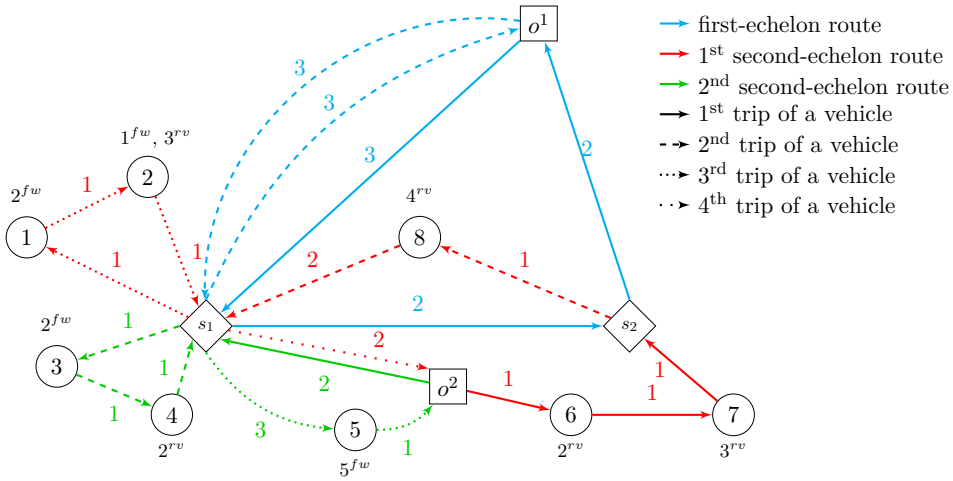


Figure 1: 2E-MTVRP-CSRF solution used in Example 1

Note that consecutive second-echelon trips must respect the transfer time $p_s = 1$ *so that departure and arrival must differ by at least one time unit. On a first-echelon satellite visit, a first-echelon vehicle must stay at a satellite for at least* $p_s = 1$ *time unit, however, there is no dedicated minimum timespan between consecutive first-echelon trips at the UDC* o^1*. To fulfill the feasibility conditions (F3)–(F5), waiting is mandatory at three vertices due to interdependencies: Trip* h_2' *has to wait for 1 time unit before starting at satellite* s_1*, because there is no forward demand ready before time 4, i.e., when the dropped off forward demand of trip* h_1 *has been processed. The corresponding entries are marked in blue. Similarly, trip* h_2^f *has to wait for 2 time units at* s_1*, because it has to collect the reverse demand dropped off by trip* h_3^2 *at time 13, which is ready for collection at time 14 (red entries). Last, trip* h_2'' *must wait for 2 time units before unloading at* s_1*, because there is no free capacity to store its reverse demand. Indeed, the trip has to wait for trip* h_3' *to free up capacity at* s_1 *(green entries).*

Figure 2a shows that these waiting times are necessary to ensure feasibility regarding flows and capacities. The figure visualizes the load profile of satellite s_1*. Forward quantities are depicted in red and reverse quantities in blue. Striped areas indicate that the quantities are transferred to the satellite but not processed so that they are not ready to be collected yet. Processed goods are shown in solid. All quantities resulting from first-echelon operations are depicted with arrows on the top of the diagram and all second-echelon operations at the bottom. Regarding the three waiting times, we can see from the figure that:*

	trip index j	1				2		
first-echelon route (h_1, h_2)	vertex i_k	o^1	s_1	s_2	o^1	o^1	s_1	o^1
	schedule T_k	0	3	6	9	9	14	18
	quantity $\gamma^{\text{fw}}_{h_j,i_k,T_k}$	–	10	0	–	–	0	–
	$\gamma^{\text{rv}}_{h_j,i_k,T_k}$	–	0	5	–	–	9	–

	j	1		2				3			
second-echelon route (h'_1, h'_2, h'_3)	vertex i_k	o^2	s_1	s_1	3	4	s_1	s_1	5	o^2
	schedule T_k	0	2	4	6	7	8	9	12	14
	quantity $\gamma^{\text{fw}}_{h'_j,i_k,T_k}$	0	–	2	–	–	–	5	–	–
	$\gamma^{\text{rv}}_{h'_j,i_k,T_k}$	–	0	–	–	–	2	–	–	0

	j	1				2			3				4	
second-echelon route $(h''_1, h''_2, h''_3, h''_4)$	vertex i_k	o^2	6	7	s_2	s_2	8	s_1	s_1	1	2	s_1	s_1	o^2
	schedule T_k	0	1	2	3	4	5	9	10	11	12	13	14	17
	quantity $\gamma^{\text{fw}}_{h''_j,i_k,T_k}$	0	–	–	–	0	–	–	3	–	–	–	0	–
	$\gamma^{\text{rv}}_{h''_j,i_k,T_k}$	–	–	–	5	–	–	4	–	–	–	3	–	0

Table 1: Time schedule and loading plan of the routes used in Example 1

- *There is no forward quantity ready to collect at the satellite before time 4.*
- *The 9 units of reverse demand collected at time 14 cannot be collected earlier, since the last 3 units dropped off are not ready at an earlier point in time.*
- *The 4 units of reverse demand dropped off at time 9 cannot be dropped off earlier due to the satellite capacity.*

Conditions (F3)–(F5) are fulfilled, because for each point in time the stored quantity is not greater than the satellites capacity $C^{sat} = 10$ and because both the available forward quantities and available reverse quantities are non-negative (see Figure 2b).

□

3.4 Trip-based formulation

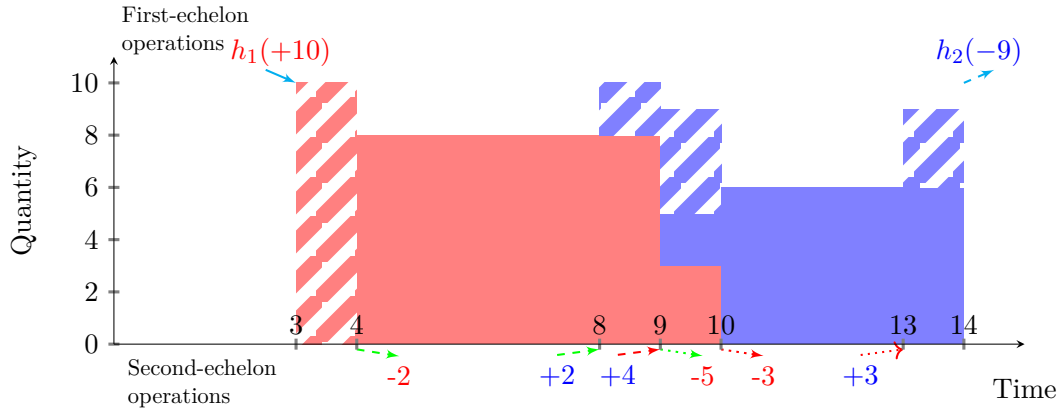
The *integer programming* (IP) model of the 2E-MTVRP-CSRF presented next can be characterized as a trip-based formulation, since there are variables for all feasible trips. Clearly, both sets Ω^1 and Ω^2 are finite but typically extremely large, because trips are defined as combinations of a walk, time schedule, and loading plan (the latter only in case of first-echelon trips). Even if not directly solvable, the model serves two purposes: First, it precisely defines the 2E-MTVRP-CSRF. Second, one component of the matheuristic explained in Section 4 uses a MIP solver with a restricted and refined version of the model.

For the IP model, we describe a first-echelon trip $h = (P, T, L) \in \Omega^1$ with the additional attribute $\beta_{h,t} \in \{0, 1\}$, which indicates whether trip h takes place at time $t \in \mathbb{T}$:

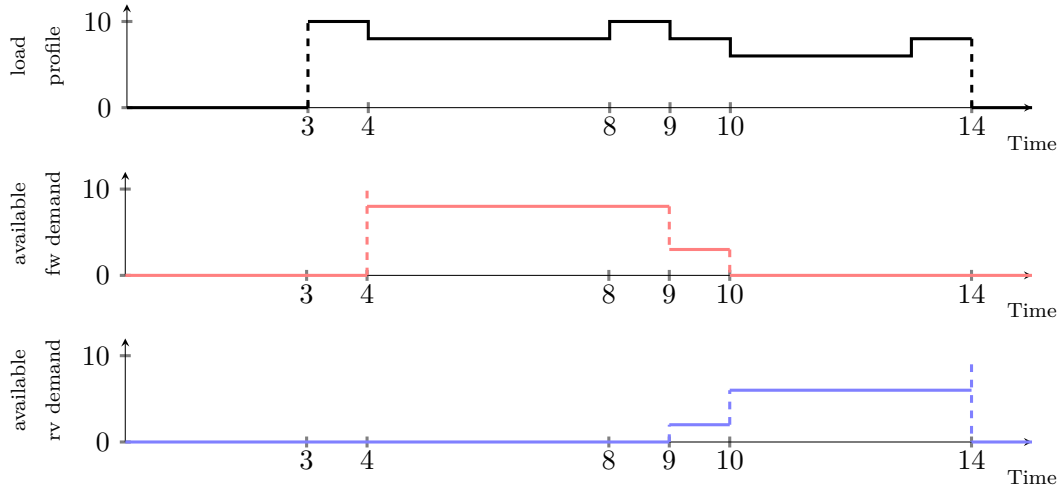
$$\beta_{h,t} = \begin{cases} 1, & \text{if } T_0 \leq t \leq T_{n+1} \\ 0, & \text{otherwise.} \end{cases}$$

In a similar manner, a second-echelon trip $h = (P, T)$ has the additional attribute $\alpha_{h,i} \in \{0, 1\}$, which indicates whether trip h visits customer $i \in N$:

$$\alpha_{h,i} = \begin{cases} 1, & \text{there exists an index } k \in \{1, 2, \ldots, n\} \text{ with } i_k = i \\ 0, & \text{otherwise.} \end{cases}$$



(a) Quantities unloaded, loaded, processed, not yet processed



(b) Satellite load profile Λ_{s_1}, profile of forward demands, and profile of reverse demands

Figure 2: Quantities and load profiles of satellite s_1 in Example 1

Additionally, it is convenient to define two subsets of second-echelon trips for satellites $s \in S$ and times $t \in \mathbb{T}$:

$$\Omega^{2+}_{s,t} = \{h \in \Omega^2 : T_0 = t \text{ and } i_0 = s\} \qquad \text{and} \qquad \Omega^{2-}_{s,t} = \{h \in \Omega^2 : T_{n+1} = t \text{ and } i_{n+1} = s\}$$

The sets $\Omega^{2+}_{s,t}$ ($\Omega^{2-}_{s,t}$) comprise those second-echelon trips that leave from (arrive at) satellite s at time t.

The model for the 2E-MTVRP-CSRF uses two types of variables: integer variables x^1_h for $h \in \Omega^1$ count the number of times that trip h is performed (note that two or more first-echelon vehicles can perform an identical trip), and binary variables x^2_h indicate whether trip $h \in \Omega^2$ is performed. The model reads as follows:

$$z(x) = \min \sum_{h \in \Omega^1} c^1_h x^1_h + \sum_{h \in \Omega^2} c^2_h x^2_h \tag{1a}$$

$$\text{subject to} \sum_{h \in \Omega^2} \alpha_{h,i} x^2_h = 1 \qquad \forall i \in N \tag{1b}$$

$$\sum_{h \in \Omega^1} \beta_{h,t} x^1_h \leq |F^1| \qquad \forall t \in \mathbb{T} \tag{1c}$$

$$\sum_{t\in\mathbb{T}}\sum_{h\in\Omega^{2+}_{o^2,t}} x^2_h = \sum_{t\in\mathbb{T}}\sum_{h\in\Omega^{2-}_{o^2,t}} x^2_h = |F^2| \tag{1d}$$

$$\sum_{t'\le t-p_s}\sum_{h\in\Omega^{2-}_{s,t'}} x^2_h - \sum_{t'\le t}\sum_{h\in\Omega^{2+}_{s,t'}} x^2_h \ge 0 \qquad \forall s\in S, t\in\mathbb{T} \tag{1e}$$

$$\sum_{t'\le t-p_s}\sum_{h\in\Omega^1} \gamma^{\mathrm{fw}}_{h,s,t'} x^1_h - \sum_{t'\le t}\sum_{h\in\Omega^2} \gamma^{\mathrm{fw}}_{h,s,t'} x^2_h \ge 0 \qquad \forall s\in S, t\in\mathbb{T} \tag{1f}$$

$$\sum_{t'\ge t+p_s}\sum_{h\in\Omega^1} \gamma^{\mathrm{rv}}_{h,s,t'} x^1_h - \sum_{t'\ge t}\sum_{h\in\Omega^2} \gamma^{\mathrm{rv}}_{h,s,t'} x^2_h \ge 0 \qquad \forall s\in S, t\in\mathbb{T} \tag{1g}$$

$$\sum_{t'\le t}\sum_{h\in\Omega^1} \gamma^{\mathrm{fw}}_{h,s,t'} x^1_h - \sum_{t'\le t}\sum_{h\in\Omega^2} \gamma^{\mathrm{fw}}_{h,s,t'} x^2_h + \sum_{t'>t}\sum_{h\in\Omega^1} \gamma^{\mathrm{rv}}_{h,s,t'} x^1_h - \sum_{t'>t}\sum_{h\in\Omega^2} \gamma^{\mathrm{rv}}_{h,s,t'} x^2_h \le C^{sat}_s \qquad \forall s\in S, t\in\mathbb{T} \tag{1h}$$

$$x^1_h \in \mathbb{N}_0 \qquad \forall h\in\Omega^1 \tag{1i}$$

$$x^2_h \in \{0,1\} \qquad \forall h\in\Omega^2 \tag{1j}$$

The objective (1a) minimizes the routing costs of the chosen first-echelon and second-echelon trips. Constraints (1b) enforce that every customer is served by a second-echelon trip, i.e., condition (F1). Constraints (1c) and (1d) are the fleet-size constraints, i.e., condition (F2). Constraints (1e) guarantee feasible combinations of second-echelon trips, i.e., they ensure for every satellite and point in time that the number of second-echelon trips that have already left a satellite can never exceed the number of second-echelon trips that have arrived and are ready to leave again. The coupling between the first and second echelon is enforced via constraints (1f)–(1h). Forward flow conservation, i.e., condition (F3), is guaranteed by constraints (1f), and reverse flow conservation, i.e., condition (F4), by constraints (1g). The satellite capacity constraints (1h) limit the quantity of goods that can simultaneously be processed and stored at each satellite, see condition (F5). Note that the third term counts quantities that will be collected by first-echelon vehicles and the fourth term quantities that are dropped off by second-echelon vehicles after time t at the satellite. The domains of the trip variables are defined by constraints (1i) and (1j).

For the flow conservation, i.e., constraints (1f) and (1g), it is always feasible to increase the quantities that first-echelon trips drop off and collect at satellites. By doing so, more flexibility is granted to schedule second-echelon trips. However, increasing first-echelon quantities may violate the satellite-capacity constraints (1h). We will exploit these observations in Section 4.3 when solving the model with a proper subset of all feasible trips. A restricted and refined model will allow the increase of first-echelon quantities to grant more flexibility to the second echelon, while the resulting surplus at the first echelon does not burden the satellite-capacity constraints.

4 Solution method

This section presents the matheuristic for the 2E-MTVRP-CSRF. It follows the *fix-and-optimize* paradigm (Helber and Sahling, 2010) with the fundamental idea that the 2E-MTVRP-CSRF can be decomposed in different ways, where parts of the solution are fixed while the remainder is exactly or heuristically optimized.

4.1 Overview of the decomposition

An overview of the decomposition that we use is provided in Table 2. The first subproblem, denoted by *SP1*, optimizes the first echelon while the second is fixed. Contrarily, the second subproblem, denoted by *SP2*, optimizes the second echelon while the first is partly fixed. We refer to the two LNS algorithms that solve the two subproblems as LNS^1 and LNS^2. In addition, a MIP solver is used with a refined version of model (1) restricted to trips taken from trip pools generated by the two LNS algorithms.

	SP1	**SP2**	**MIP**
Input:	solution (x^1, x^2)	solution (x^1, x^2)	$\bar{\Omega}^1$, $\bar{\Omega}^2$
first-echelon trips	[$\star$]	[+]	[$\star$, set of]
· walks	[$\star$]	[fixed]	[fixed]
· schedules	[$\star$]	[fixed]	[fixed]
· loading plans	[$\star$]	[$\star$]	[fixed]
first-echelon routes	[$\star$]	[+]	[impl. result]
second-echelon trips	[fixed]	[$\star$]	[$\star$, set of]
· walks	[fixed]	[$\star$]	[fixed]
· schedules	[fixed]	[$\star$]	[fixed]
second-echelon routes	[fixed]	[$\star$]	[impl. result]
Solution method:	LNS^1	LNS^2	MIP solver
Output:	solution (x'^1, x'^2) trips H^2	solution (x'^1, x'^2) trips H^1	solution (x'^1, x'^2)

[$\star$]: modifies; [+]: partially modifies; [fixed]: does not modify; [$\star$, set of]: selects a solution from sets of trips; [impl. result]: implicit result, i.e., routes are not constructed, but it is guaranteed that trips can be combined into routes respecting the fleet-size constraint

Table 2: Overview of the decomposition used in the matheuristic (Algorithm 1)

Algorithm 1 shows how the three subproblems and corresponding solution methods interact in the matheuristic that we characterize as an *iterative two-stage heuristic* (ITSH). ITSH starts with the construction of an initial solution $x = (x^1, x^2)$ and empty trip pools $\bar{\Omega}^1$ and $\bar{\Omega}^2$ (Steps 1 and 2). The current solution is improved by repeating the main loop (Steps 4–12) until the given time limit Γ_{tot} is reached. In each iteration, SP2 is optimized first with LNS^2 (Step 4), and all generated second-echelon trips are added to $\bar{\Omega}^2$ (Step 5). Next, SP1 is optimized with LNS^1 (Step 6) and all generated first-echelons trips are added to $\bar{\Omega}^1$ (Step 7). If LNS^1 and LNS^2 fail to improve the best-found solution and the MIP execution condition is satisfied, the refined version of formulation (1) is solved using only (modified) trips from the trip pools $\bar{\Omega}^1$ and $\bar{\Omega}^2$ (Step 11). To keep the size of the MIP reasonable, the trip pools are re-initialized (Step 9) every time a new best solution is found and after each call to the MIP solver.

Details on the two LNS algorithms follow in Section 4.2. As the feasibility check for insertion steps used in the two LNS algorithms is of high importance for the efficiency of the overall matheuristic, we present its details starting with the precedence-graph-based solution representation in Section 4.2.1, a higher-level description of LNS^1 and LNS^2 is given in Sections 4.2.2 and 4.2.4, and in-depth discussions of the sequence of tests performed when checking feasibility in Sections 4.2.3 and 4.2.5, respectively. The MIP component, its parameters, execution conditions, how it communicates with the LNS, and the update of the trip pools is described in Section 4.3.

Algorithm 1: Iterative Two-Stage Heuristic (ITSH)

$(x^1, x^2) \leftarrow$ initialization
$\bar{\Omega}^1 \leftarrow \emptyset$, $\bar{\Omega}^2 \leftarrow \emptyset$
repeat
  $(x^1, x^2, H^2) \leftarrow \text{LNS}^2(x^1, x^2)$
  $\bar{\Omega}^2 \leftarrow \bar{\Omega}^2 \cup H^2$
  $(x^1, x^2, H^1) \leftarrow \text{LNS}^1(x^1, x^2)$
  $\bar{\Omega}^1 \leftarrow \bar{\Omega}^1 \cup H^1$
  if *the best solution has been improved* **then**
    $(\bar{\Omega}^1, \bar{\Omega}^2) \leftarrow$ re-initialize$(\bar{\Omega}^1, \bar{\Omega}^2)$
  else if *the MIP execution conditions are satisfied* **then**
    $(x^1, x^2) \leftarrow \text{MIP}(\bar{\Omega}^1, \bar{\Omega}^2)$
    $(\bar{\Omega}^1, \bar{\Omega}^2) \leftarrow$ re-initialize$(\bar{\Omega}^1, \bar{\Omega}^2)$
until *time limit* Γ_{tot} *exceeded*
return $(x^1, x^2) \leftarrow$ the best-found feasible solution

4.2 LNS algorithms

In the LNS metaheuristic, first proposed by Shaw (1998) in a constraint programming context, the current solution is iteratively destroyed and subsequently repaired until a stopping criterion is reached. LNS algorithms have been shown to be successful in optimizing routes and trips, especially for VRP variants that only comprise standard resource constraints such as route length, capacity, and time-window constraints (Pisinger and Ropke, 2019). Even more complicated inter-route constraints (an overview is provided in Irnich *et al.*, 2014, Section 1.3.5) can be integrated easily, as long as the removal of a customer is always feasible and an efficient feasibility test for the insertion of customers is available (see, e.g., Grangier *et al.*, 2016).

Our LNS algorithms switch between large destruction steps and small destruction steps. Small destruction steps are performed in most iterations to locally and quickly improve solutions (Christiaens and Vanden Berghe, 2020). If the best solution has not been improved for several iterations, a large destruction step is performed for diversification. This small-and-large strategy has been proved highly successful on variants of the generalized vehicle routing problem with time windows (Dumez *et al.*, 2021). A short synopsis of the types of operators used in our LNS algorithms is provided in Table 3 indicating whether the operator is used in LNS^1 and LNS^2 and who first introduced it. For the sake of brevity, a pseudocode of the LNS and more details of the destroy and repair operators as well as the acceptance criterion are provided in Appendix A.1 and the construction of a starting solution in Appendix A.2. Note that so-called *incomplete solutions* are explored during LNS. For that purpose, a solution features a *request bank* of non-served customers which are penalized in the objective function (Ropke and Pisinger, 2006a).

4.2.1 Solution representation

Ordinary LNS algorithms for standard VRPs remove customers and re-insert them trying to obtain a feasible and improving solution. The situation for the 2E-MTVRP-CSRF is much more involved because there are no customers on the first echelon and already the removal of customers can make a feasible solution infeasible w.r.t. the satellite capacity if the corresponding quantities are removed arbitrarily from first-echelon satellite visits. Moreover, re-inserting a customer modifies at least the schedule and quantities of the affected second-echelon trip. Besides, it may

Type		Operator name	LNS[1]	LNS[2]	Source
Destroy operators	small	route split string removal	✓	✓	Christiaens and Vanden Berghe (2020)
		satellite split string removal	✓	✓	*introduced in this paper*
		distance related removal		✓	Ropke and Pisinger (2006b)
		bundle removal		✓	*introduced in this paper*
		cluster removal		✓	Pisinger and Ropke (2007)
	large	random customer removal		✓	Ropke and Pisinger (2006b)
		random bundle removal	✓		*introduced in this paper*
		worst bundle removal	✓		*introduced in this paper*
		historical knowledge node removal		✓	Demir *et al.* (2012)
		trip and route removal	✓	✓	Nagata and Bräysy (2009)
Repair operators		random order best insertion	✓	✓	Christiaens and Vanden Berghe (2020)
		largest first best insertion	✓	✓	Christiaens and Vanden Berghe (2020)
		farthest first best insertion		✓	Christiaens and Vanden Berghe (2020)
		closest first best insertion		✓	Christiaens and Vanden Berghe (2020)
		earliest first best insertion	✓	✓	Christiaens and Vanden Berghe (2020)
		latest first best insertion	✓	✓	Christiaens and Vanden Berghe (2020)
		narrow first best insertion	✓	✓	Christiaens and Vanden Berghe (2020)

Table 3: Destroy and Repair Operators.

trigger additional necessary modifications on the schedule and loading plan of other trips and routes. Because of these complications, we introduce an additional representation of possibly incomplete solutions. Our new solution representation is based on a *precedence graph* and it helps to properly describe interdependencies between objects of both echelons that can be fixed, removed, re-inserted, or modified. In particular, it supports fast feasibility tests and allows us to perform modifications on the schedule and loading plan of the solution's trips and routes.

A first fundamental simplification in our LNS algorithms is that customer demands are never split over several first-echelon satellite visits. However, the problem definition and model (1) allow demand splits on the first echelon. Given this simplification, we can introduce, for each customer $i \in N$, a forward *first-echelon customer* (FEC) i^{fw} if the forward demand q_i^{fw} is positive and a reverse FEC i^{rv} if the reverse demand q_i^{rv} is positive. First-echelon customers are *not* real customers, but they constitute the transportation tasks for the first-echelon given that the real customer is served at the second echelon. FECs are introduced to link the first echelon with the second echelon, they are removed, re-inserted, and re-assigned in the course of both LNS algorithms thereby ensuring that removals preserve the feasibility of a solution.

Moreover, to link first-echelon satellite visits and FECs, we define a *bundle* as a pair (s, I) of a satellite $s \in S$ and a subset I of FECs. Each first-echelon satellite visit to s constitutes a bundle (s, I) and exactly the demand of FECs in I are transferred at satellite s during this first-echelon satellite visit. In the following, inserting (removing) a FEC into (from) a bundle (s, I) means that the FEC is added to (removed from) the set I. Moreover, if a FEC is re-assigned, it means that the FEC is removed from one bundle and inserted into another bundle.

The following precedence graph $G_P^x = (V_P^x, A_P^x)$ allows us to represent any possibly incomplete solution x. The vertices V_P^x are:

1. copies of the UDC representing departure and arrival of each first-echelon trip,
2. bundles indicating that the subset of FECs is jointly transferred at the associated first-echelon satellite visit,
3. copies of satellites and of the second-echelon depot representing departure and arrival of second-echelon trips, and

4. customers.

Accordingly, we group these vertices into *levels* 1 to 4. The arcs A_P^x result from
(a) first-echelon routes (connections between vertices of levels 1 and 2),
(b) second-echelon routes (connections between vertices of levels 3 and 4), and
(c) linking FECs (included in a bundle) with copies of satellites, i.e., departures and arrivals of second-echelon trips (connections between vertices of levels 2 and 3).

Example 2 *Figure 3 shows an incomplete solution for a small instance with only one satellite s with capacity $C^{sat} = 3$ and five customers $1, 2, \ldots, 5$. The capacity of first- and second-echelon vehicles is $Q^1 = 4$ and $Q^2 = 2$, respectively. For the sake of simplicity, we assume that the second-echelon depot o^2 is identical to the satellite s, that customers have non-restricting time windows, that the travel times between all pairs of locations as well as transfer times at the satellite are equal to one time unit, and that the customer service duration is zero. The length of the planning horizon is $\bar{t} = 9$ time units.*

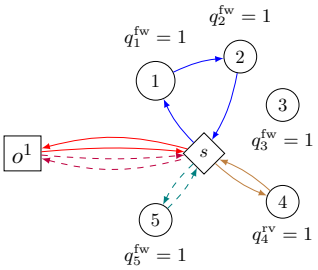
The first-echelon vehicle (colored red) performs two back-and-forth trips between the depot and the satellite. The demands of customers 1, 2, and 5 are dropped during the first visit at the satellite and the demand of customer 4 is collected at the second visit. The blue-colored second-echelon vehicle serves customers 1 and 2 during its first trip and customer 5 during its second trip. The brown-colored second-echelon vehicle serves customer 4 during its only trip. Customer 3 is not served.

In the graphical representation in Figure 3a, solid lines represent the first trip and dashed lines the second trip of a vehicle. The time schedule of all routes is depicted in Figure 3b. The last row of the table shows the load profile Λ_s of the satellite s. Recall that the load profile describes the quantity processed and stored at satellite s over time t, i.e., each visit of a vehicle changes the value by the quantities collected or dropped-off. Finally, the precedence graph of the incomplete solution is shown in Figure 3c. □

Note first that there is a natural one-to-one correspondence between the vertices of the routes of a solution x and the vertices V_P^x. The structural feasibility conditions (F1), (F2), (Tr7), (Tr8), (R2), and (R3) are ensured by the structure of the precedence graph G_P^x. For example, (F1) is fulfilled if all customers have ingoing and outgoing arcs in the precedence graph, and if all FECs are present in exactly one bundle. For the fleet-size condition (F2), not more than $|F^1|$ ($|F^2|$) chains of arcs are allowed at levels 1 and 2 (levels 3 and 4). *Time windows* and *processing times* can be added to the vertices and arcs of the precedence graph G_P^x as shown in Tables 4a and 4b. We interpret these attributes in the same way as the time resource in resource constrained project scheduling (see, e.g., Brucker and Knust, 2012).

Proposition 1 *Let x be a solution without splitting of customer demands on the first echelon. The schedule $(T_v)_{v \in V_P^x}$ of the solution x fulfills all temporal constraints, i.e., (F3), (F4), (Tr1), (Tr2), (Tr4), (Tr5), (R1), and (R4), if and only if the same schedule is feasible for the associated precedence graph G_P^x with respect to time windows and processing times as defined in Table 4.*

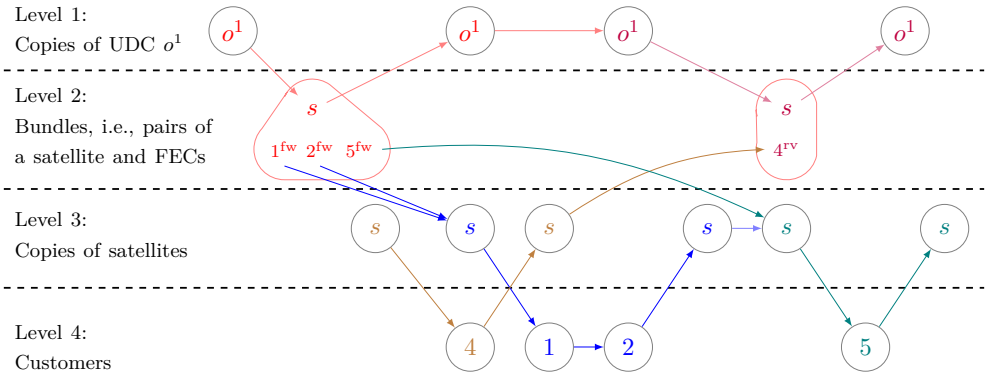
Feasibility checking for a precedence graph is a known problem in deterministic project scheduling that can be efficiently solved, e.g., with the same computations as in the critical path method or PERT (Miller, 1963). To make the situation perfectly fit, the reader may add two artificial vertices for the start and the end of the project to the precedence graph. These two vertices are connected by an arc having processing time $\bar{t}$. For each vertex v with time window $[a_v, b_v]$, the time-window constraint translates to two additional arcs: From the start vertex to vertex v with processing time a_v and from vertex v to the end vertex with processing time b_v. Using forward (backward) propagation of times, the earliest (latest) time



(a) Graphical representation

Time	0	1	2	3	4	5	6	7	8	9
1st-echelon route	o^1	s	o^1			o^1	s	o^1		
2nd-echelon routes			s	1	2	s	s	5	s	
		s	4	s						
Satellite load profile Λ_s	0	3	1	2	–	–	0	–	–	–

(b) Representation with routes, trips, and satellite load profile



(c) Representation with precedence graph

Figure 3: Three representations of the same incomplete solution of Example 2, customer 3 is not served

can be computed for each vertex $v \in V_P^x$. If the difference between latest and earliest time is non-negative at each vertex, the precedence graph G_P^x is feasible and x fulfills all temporal constraints by Proposition 1.

For a given topological ordering of the vertices V_P^x (note that the precedence graph is acyclic), the computational effort of the temporal feasibility check is $\mathcal{O}(|A_P^x|) = \mathcal{O}(|N|)$.

The concept of earliest and latest service times has been introduced in the VRP context by Savelsbergh (1992) using the name *forward time slack* to describe the maximum possible delay w.r.t. the as-early-as-possible schedule of a route (computed in forward direction; explaining the name). For all vertices in a route of the 2E-MTVRP-CSRF, the earliest feasible service time *route-EST* and the latest feasible service time *route-LST* are computed disregarding potential interdependencies between routes.

To take into account such interdependencies, Masson *et al.* (2013) generalize the work of Savelsbergh to develop constant-time feasibility checks for the pickup-and-delivery problem with transfers. They introduce a *generalized forward time slack* for precedence graphs. The concept has been extended to the 2E-VRP by Grangier *et al.* (2016). The calculated times are called *solution earliest service time* (*solution-EST*) and *solution latest service time* (*solution-LST*).

The capacity constraints of the vehicles, i.e., conditions (Tr3) and (Tr6), can be tested directly per trip (and not per route). Recall that on the first (second) echelon, these trips can (must) simultaneously drop off (deliver) and collect (pick up) in a single visit. For the insertion of customers and FECs, the feasibility check of Dell'Amico *et al.* (2006) is linear in the length of the trip (when computed from scratch), and constant-time feasibility checks rely on pre-computed auxiliary information (Irnich, 2007).

Therefore, we assume that checking feasibility w.r.t. the capacity of the vehicles does not

Level	Vertex	Time window
1	o^1	$[0, \bar{t}]$
2	(o^2, I)	$[0, \bar{t}]$
	(s, I)	$[a_s, b_s]$
3	o^2	$[0, \bar{t}]$
	s	$[a_s, b_s]$
4	i	$[a_i, b_i]$

(a) Time windows; $s \in S \cup \{o^2\}$ is a satellite, $i \in N$ a customer, and I is a subset of FECs

Connecting Levels	Arc	Processing Time
1 and 1	(o^1, o^1)	0
1 and 2	$(o^1, (s, I))$	$t_{o^1 s} + p_s$
	$((s, I), o^1)$	$t_{s o^1} + p_s$
2 and 2	(s_1, s_2)	$t_{s_1 s_2} + p_{s_1}$
2 and 3	$((s, I), s)$	p_s
	$(s, (s, I))$	p_s
3 and 4	(i, s)	t_{is}
	(s, i)	t_{si}
4 and 4	(i_1, i_2)	$t_{i_1 i_2}$

(b) Processing times; $s, s_1, s_2 \in S \cup \{o^2\}$ are satellites or the second-echelon depot, and $i, i_1, i_2 \in N$ are customers

Table 4: Time windows and processing times of the precedence graph

need to be discussed in the following.

The crucial point that remains is checking feasibility regarding the satellite capacities, i.e., condition (F5). Recall that if the load profile Λ_s of a satellite $s \in S$ is always not greater than the satellite capacity C_s^{sat}, then condition (F5) is fulfilled for satellite s. As a consequence, the feasibility of the load profile depends on the schedule of the routes. Considering the complexity of the resulting integrated scheduling and routing problem, we apply a heuristic feasibility evaluation procedure. This type of approach was successfully used by Masson *et al.* (2014) to test feasibility in the dial-a-ride problem with transfers and by Grangier *et al.* (2019) in the VRP with cross-docking and resource constraints. In the latter case, it was shown that a heuristic feasibility test allows many more LNS iterations in the same timespan, leading to a better overall performance than an LNS equipped with an exact feasibility test using constraint programming.

To employ such a fast heuristic feasibility check, we should *not* alter the sequence of operations (i.e., arrivals and departures of vehicles) at each satellite s if the current solution x fulfills condition (F5), because this conserves the same increases and decreases in the profile Λ_s. As a result, all schedules maintaining the sequence of operations also maintain $\Lambda_s \leq C_s^{sat}$. Moreover, the evaluation of the satellite load profile for an insertion is straightforward in this case. This idea is exploited in both LNS algorithms. In addition, LNS^2 embeds another heuristic feasibility check that allows to alter the sequence of operations when an insertion is infeasible with the current sequence.

4.2.2 LNS for the first echelon

LNS^1 operates on the first echelon routes of the solution while the second echelon is fixed (see Table 2), i.e., second-echelon routes remain unchanged. In particular, the levels 3 and 4 of the precedence graph remain fixed including their current schedule times. To this end, we reduce the time windows of the vertices of G_P^x belonging to levels 3 and 4 to the current schedule times of the given solution x, while the remaining vertices keep their time windows as indicated in Table 4a.

The destroy operators in LNS^1 first select and remove FECs from the current solution x and put them into the request bank. Additionally, bundles without any FEC are removed. If this removal results in an empty first-echelon trip, the trip is removed, too (i.e., the two copies of the UDC are removed). To compute the load profile of an incomplete solution, we only take

into account the demands of second-echelon customers if their FECs are served. Thereby, the incomplete solution obtained after removing FECs is always feasible. However, these customers remain in their second-echelon trips until the end of the insertion process. If finally FECs cannot be feasibly inserted, the corresponding customers are removed from their second-echelon trips.

The repair operators are all list operators, i.e., they consider one FEC in the request bank at a time and insert it into a bundle. The insertion of a forward FEC i^{fw} into a bundle (s, I) creates an additional arc in the precedence graph from the bundle (s, I) to the start of the second-echelon trip that serves customer i. Likewise, the insertion of a reverse FEC i^{rv} into a bundle (s, I) creates an additional arc in the precedence graph from the end of the second-echelon trip that serves customer i to the bundle (s, I). Additionally, depending on the insertion type, arcs of first-echelon routes are modified in the precedence graph. We consider three types of insertions of a FEC into a bundle:

- *Insertion into an existing bundle:* In the precedence graph, the FEC is inserted into an existing bundle.
- *Insertion by creation of a new bundle:* A new bundle is added to an existing trip and route arcs are changed appropriately. The FEC is inserted into the new bundle.
- *Insertion by creation of a new trip with a single bundle:* A new first-echelon trip with a single bundle is added to an existing route. In the precedence graph, arcs are changed to insert the trip into the route and the FEC is inserted into the new bundle.

For each insertion type, all possible insertion positions are evaluated in LNS^1 according to the evaluation process described next.

4.2.3 Insertions and feasibility tests in the LNS for the first echelon

The repair operators consider all insertion positions for each unassigned FEC in each LNS iteration. Consequently, a fast and efficient feasibility evaluation is required for the numerous potential insertions. Figure 4 provides an overview of the feasibility evaluation procedure of LNS^1. The heuristic nature of the evaluation lies in rejecting solutions of unknown feasibility status because of their costs or because of an otherwise too time-consuming re-scheduling and re-evaluation.

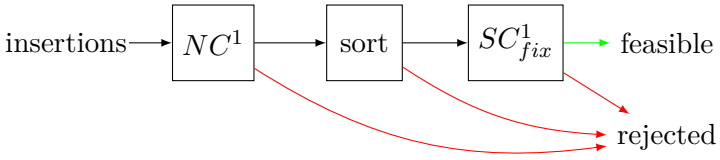


Figure 4: Sequence of necessary and sufficient conditions in the feasibility test of LNS^1

The necessary condition NC^1 tests for all insertion types and positions of a FEC the vehicle capacity of the affected trip and identifies infeasible insertion with respect to route-EST and route-LST. All potential insertions passing NC^1 are evaluated in terms of cost and sorted in non-decreasing order of their cost, before the more elaborate feasibility tests are performed. The entire feasibility evaluation terminates as soon as one insertion is proven feasible. By avoiding tests of costly solutions, the sorting procedure implicitly rejects solutions that are found but have not been checked yet because of their higher cost compared to a cheaper feasible solution considered before.

When considering the insertion of a FEC, the time schedule of the first-echelon route and satellite capacity of the satellite s serving the customer must be tested. Let x_- be the current incomplete solution, and let $(T_v)_{v \in V_P^{x_-}}$ denote the time schedule. The sufficient condition SC^1_{fix} checks whether the considered insertion into x_- can be performed

- without changing the sequence of operations at any satellite s as given in x_- and
- without modifying the bundles for currently assigned FECs.

Under these assumptions, SC^1_{fix} checks the temporal feasibility and satellite capacity. We discuss these two checks now:

Schedule checking: First, SC^1_{fix} evaluates whether performing the considered insertion respects the solution-LST when starting from the solution-EST (for all pairs of predecessors and successors in $G^{x_-}_{PF}$). If it is not the case, then SC^1_{fix} rejects the considered insertion. The temporal feasibility check uses generalized forward time slacks in the *precedence graph with fixed sequence* $G^{x_-}_{PF}$. In this augmented precedence graph $G^{x_-}_{PF}$, *additional* arcs guarantee that the current sequences of operations taking place at all satellites $s \in S$ are maintained. (For a visualization, the later Example 3 shows such an augmented precedence graph $G^{x_-}_{PF}$ for LNS^2 in Figure 6a.) Hence, the additional arcs (v, w) connect every two vertices v and $w \in V^{x^-}_P$ with $T_v < T_w$ referring to the same satellite s (vertices of levels 2 and 3). (Note that the transitive reduction of the pairs/relation (v, w) suffices.) In case that $T_v = T_w$ holds for two vertices v and w, we add a arc only in one direction: second-level arrivals are connected to the bundle, and bundles are connected to second-level departures (other ties are broken arbitrarily). All these additional arcs (v, w) have a processing time of 0. Note that neither the precedence graph nor the precedence graph with fixed sequence can be decomposed by satellite because trips can connect any pair of satellites.

Capacity checking: Second, the satellite-capacity constraint of the affected satellite is tested. This check can be performed efficiently due to the fixed sequence of operations at the satellite. Recall that the sequence of operations was feasible w.r.t. (F5) before the insertion.

We distinguish two cases:

- When a forward FEC is inserted, we need to compute the resulting satellite load profile for all relevant points in time between the bundle that contains the inserted FEC and the start of the second-echelon trip that serves the corresponding customer.
- Likewise, when a reverse FEC is inserted, we need to compute the resulting satellite load profile for all relevant points in time between the end of the second-echelon trip that serves the corresponding customer and the bundle that contains the inserted FEC.

If the satellite capacity is violated, the considered insertion is rejected (although it might be feasible for another sequence of operations). Otherwise, the insertion is feasible w.r.t. the satellite capacity. The insertion is conducted in this case. The times of newly created vertices (bundles and visits to the UDC) are set to the solution-EST of the precedence graph with fixed sequence, thereby, updating solution-EST and solution-LST of all other vertices.

If LNS^1 terminates and some forward (reverse) FEC cannot be feasibly inserted, then these FECs, the corresponding reverse (forward) FECs, and the corresponding second-echelon customers are removed from the solution. They are all put into the request bank for further consideration in LNS^2. Note that these removals maintain the feasibility of the solution w.r.t. condition (F5).

4.2.4 LNS for the second echelon

LNS^2 works on the second echelon and fixes parts of the first-echelon solution (see Table 2), i.e., the arcs and current schedule times at levels 1 and 2 of the precedence graph remain fixed. To this end, time windows of the vertices V^x_P belonging to levels 1 and 2 are reduced to the current schedule times of the given solution x, while the remaining vertices keep their time windows as indicated in Table 4a. However, bundles can be modified at level 2 of the precedence graph,

i.e., FECs can be re-assigned. In all LNS^2 destroy operators, when a customer is removed from a second-echelon route, its FECs are also removed from their corresponding bundles and they are put together into the request bank. This combined removal ensures that a feasible solution remains feasible. Conversely, in repair operators, an insertion of a customer i into a second-echelon trip $(s, \ldots, s')$ includes inserting i^{fw} into a suitable bundle of satellite s and inserting i^{rv} into a suitable bundle of satellite s'. As in LNS^1, all repair operators are list operators, hence, they consider one customer and his FECs at a time.

LNS^2 employs three types of insertions of a customer i into a second-echelon route: For all three insertion types, all possible insertion positions are evaluated according to the feasibility test procedure explained afterwards. The feasibility test process also determines how FECs are assigned to bundles and adds the corresponding arcs to the precedence graph. Accordingly, we explain here only the changes on the precedence graph regarding the second echelon routes (levels 3 and 4).

- *Insertion into an existing trip:* This inserts the customer vertex i between two vertices of an existing second-echelon trip in the precedence graph.
- *Insertion by creation of a new trip:* This insertion creates a new trip in an existing second-echelon route. The trip comprises an origin satellite, the customer i, and a destination satellite. Arcs in the precedence graph are changed appropriately to connect the new trip with the route.
- *Insertion by trip split (Grangier* et al.*, 2016):* This splits an existing trip in two by adding a destination satellite vertex for the end of the first part of the trip and (the same) satellite vertex as a start of the second half of the trip between any two consecutive vertices in the trip. All insertion positions for customer i in any of the two new trips are considered. All FECs associated with customers in the trip might need to be re-assigned to a bundle at the inserted satellite (together with the arcs connecting levels 2 and 3).

4.2.5 Insertions and feasibility tests in the LNS for the second echelon

Figure 5 provides an overview of the feasibility evaluation procedure of LNS^2. For a given customer, the condition NC^2 checks necessary conditions for all insertion types and positions. All insertions that pass NC^2 are sorted in non-decreasing order of their cost. Their feasibility is tested in this order and the procedure stops as soon as one insertion is proven feasible by one of the two sufficient conditions SC^2_{fix} and SC^2_{DJ}. SC^2_{fix} tests if the insertion can be performed without changing the order of operations at the satellites similar to SC^1_{fix}. If SC^2_{fix} fails, the solution is re-scheduled with the *double-justification* algorithm (Wiest, 1964) in SC^2_{DJ}.

Double justification is a two-phase algorithm shown to be effective in improving solutions of resource-constrained project scheduling problems (Valls *et al.*, 2005). If SC^2_{DJ} also fails the insertion is rejected. Note that the complete procedure is heuristic, i.e., the feasibility status of rejected insertions is unknown.

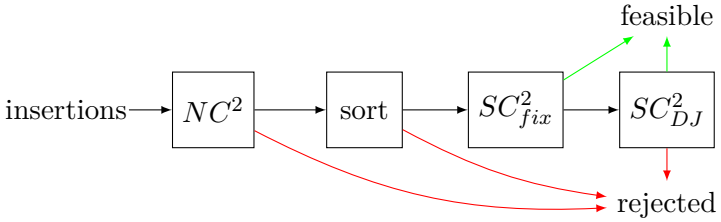


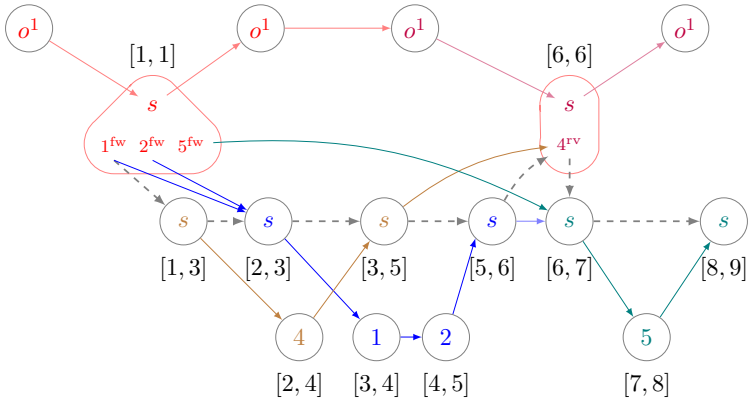
Figure 5: Sequence of necessary and sufficient conditions in the feasibility tests of LNS^2

Let x_- be the current incomplete solution, let $(T_v)_{v \in V_P^{x_-}}$ denote its schedule, and let $i \in N$ be the customer chosen by the repair operator to get inserted next. The necessary condition NC^2 first tests, for all insertion types and positions of customer i, the vehicle capacity of the affected second-echelon trip and uses standard forward time slacks (Savelsbergh, 1992) for the affected second-echelon route to identify insertions that violates route-EST and route-LST. The FECs of customer i are not inserted yet and, therefore, the capacity is not checked for first-echelon trips. Overall, NC^2 can be evaluated in constant time, after an $O(|V|)$ pre-processing. As in LNS[1], all potential insertions passing NC^2 are evaluated in terms of cost, sorted in non-decreasing order, and the entire feasibility evaluation terminates as soon as one insertion is proven feasible.

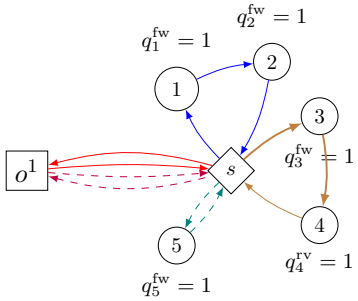
The sufficient condition SC^2_{fix} follows the same idea as its counterpart SC^1_{fix}, i.e., to maintain the sequence of operations at the satellites. However, the situation is more complicated here, because we need to insert the customer into the second-echelon trip and insert the corresponding FECs into a bundle. First, the insertion of customer i into second-echelon trip h is checked for temporal feasibility with the help of the precedence graph. Infeasible insertions are directly handed over to SC^2_{DJ}. If the insertion of customer i is feasible w.r.t. the temporal constraints, the next step is to insert the FECs of i. Let h be the second-echelon trip that customer i is inserted in, let T_1 be its starting time at satellite s_1, and let T_2 be its ending time at satellite s_2:

- The forward FEC i^{fw} is added to the latest bundle (s_1, I) with $T_{(s_1,I)} + p_{s_1} \leq T_1$ such that the capacity of the corresponding first-echelon vehicle is respected.
- Likewise, the reverse FEC i^{rv} is added to the earliest bundle (s_2, I') with $T_{(s_2,I')} \geq T_2 + p_{s_2}$ such that the first-echelon vehicle capacity is respected.

In both cases, first-echelon vehicle capacity is checked in constant time. If no suitable bundle can be found in at least one of the two cases, SC^2_{fix} fails and the insertion is handed over to SC^2_{DJ}.



(a) Precedence graph with fixed sequence $G^{x_-}_{PF}$; augmented arcs are depicted as dashed arcs; recall that they have a processing time of 0.



(b) Graphical representation of the solution after the insertion of customer 3.

Figure 6: Insertion of customer 3 into the second-echelon trip $(s, 4, s)$, continued from Example 2; the new solution is temporal feasible

Example 3 *(continued from Example 2) In this example, we illustrate the feasibility checks that are performed when testing whether customer* 3 *can be inserted into the trip* $(s, 4, s)$ *of the incomplete solution* x_- *that was discussed in Example 2 and visualized in Figure 3.*

Figure 6a depicts the precedence graph with fixed sequence $G^{x_-}_{PF}$ *associated with the incomplete solution* x_-*. Next to each vertex, the resulting solution-EST and solution-LST are shown as a*

time interval. Recall that:

- *First-echelon routes are fixed in time, i.e., solution-EST equals solution-LST for the vertices at level 2 (values are set to those of the time schedule shown in Table 3b).*
- *Augmented arcs connect subsequent operations at the satellite s according to their time schedule in x_-. Note that the bundle $(s, 4^{rv})$ (level 2) and the satellite departure of the second-echelon trip $(s, 5, s)$ (level 3) both take place at time 6. Hence, they are connected from the bundle $(s, 4^{rv})$ to the departure from s (depicted as a vertical, dashed, downward directed arc).*
- *Recall that processing times of solid (=route) arcs are all 1, while those of dashed (=augmented) arcs are all 0.*

At this point, NC^2 has already verified that the new second-echelon trip $(s, 3, 4, s)$ is feasible regarding the vehicle capacity and the route forward time slack. Next, SC^2_{fix} compares the solution-EST 1 of the departure of trip $(s, 4, s)$ and the solution-LST 4 of customer 4 and we see that a 1-time-unit detour to visit customer 3 does not violate temporal constraints in the precedence graph. The tentative insertion of customer 3 produces the new second-echelon trip $h = (s, 3, 4, s)$ (depicted in Figure 6b) with new solution-EST and solution-LST values $[1, 2]$, $[2, 3]$, $[3, 4]$, and $[4, 5]$ for the respective vertices. The first test of SC^2_{fix} is passed.

Next, the associated FEC 3^{fw} must be integrated into the solution, too. It is only possible to insert the FEC 3^{fw} into the bundle $(s, \{1^{fw}, 2^{fw}, 5^{fw}\})$ and connect it with the departure of the second-echelon trip h. The solution-EST of this bundle is 1 and the (new) solution-LST of the start of this trip is 2, allowing a processing time of 1 unit in-between. Moreover, the additional quantity q_3^{fw}=1 does not violate the first-echelon vehicle capacity of 4 units. The only other bundle $(s, \{4^{rv}\})$ has a solution-EST of 6 so that the insertion would violate the temporal constraint. As a result, the departure of the new trip h must be delayed to time 2 in the as-early-as-possible schedule of the new (tentative) solution x' shown in Table 5a. At this point, we have shown that the insertion results in a new solution x' that respects all temporal constraints. □

The next aspect to be tested is the feasibility of the satellite load profile. For the feasibility test, the satellite capacity is evaluated for all operations occurring between times $T_{(s,I)}$ and T_1 and/or between times T_2 and $T_{(s',I')}$. If SC^2_{fix} fails, the next test SC^2_{DJ} allows to change the sequence of operations at the satellite to restore feasibility. In that process, FECs can also be re-assigned.

Example 4 *(continued from Example 3) We reconsider the tentative solution x' (see Figure 6b) with the inserted customer 3 and the as-early-as-possible schedule computed over the precedence graph with fixed sequence $G^{x'}_{PF}$. This graph is not identical to $G^{x_-}_{PF}$ depicted in Figure 6a; it has three additional arcs, namely arc $(s, 3)$ where s is the copy of the satellite representing the start of the new trip $(s, 3, 4, s)$, arc $(3, 4)$, and arc $((s, \{1^{fw}, 2^{fw}, 3^{fw}, 5^{fw}\}), s)$ where the head vertex is the same copy of the satellite. The computed time schedule is presented in Table 5a. The table also includes the bundles and the resulting satellite load profile. As all four forward FECs are simultaneously dropped at satellite s at time 3, the tentative solution x' is infeasible w.r.t. the satellite capacity $C^{sat} = 3$.* □

Now we describe the procedure SC^2_{DJ}, which is the most elaborate but potent insertion test: To allow changes in the sequence of operations, SC^2_{DJ} works on the precedence graph G^x_P. First, customer i is inserted into G^x_P without checking temporal feasibility. Its FECs are inserted as in SC^2_{fix}, but tolerating capacity violations of first-echelon vehicles. Note that these insertions may cause violations of flow-conservation and satellite-capacity constraints.

Time	0	1	2	3	4	5	6	7	8	9
1st-echelon route	o^1	s	o^1			o^1	s	o^1		
		$\{1^{\text{fw}}, 2^{\text{fw}}, 3^{\text{fw}}, 5^{\text{fw}}\}$					$\{4^{\text{rv}}\}$			
2nd-echelon routes			s	1	2	s	s	5	s	
			s	3	4	s				
Satellite load profile Λ_s	0	4	1	–	–	2	0	0	–	–

(a) Time schedule directly after the tentative insertion of customer 3

Time	o^1	1	2	3	4	5	6	7	8	9
1st-echelon route	o^1	s	o^1			o^1	s	o^1		
		$\{1^{\text{fw}}, 2^{\text{fw}}, 3^{\text{fw}}\}$					$\{4^{\text{rv}}, 5^{\text{fw}}\}$			
2nd-echelon routes				s	1	2	s	s	5	s
			s	3	4	s				
Satellite load profile Λ_s	0	3	2	0	–	1	1	0	–	–

(b) As-late-as-possible time schedule, not violating the satellite-capacity constraint

Time	0	1	2	3	4	5	6	7	8	9
1st-echelon route	o^1	s	o^1			o^1	s	o^1		
		$\{1^{\text{fw}}, 2^{\text{fw}}, 3^{\text{fw}}\}$					$\{4^{\text{rv}}, 5^{\text{fw}}\}$			
2nd-echelon routes			s	1	2	s		s	5	s
			s	3	4	s				
Satellite load profile Λ_s	0	3	0	–	–	1	1	0	–	–

(c) As-early-as-possible time schedule, not violating the satellite-capacity constraint

Table 5: Time schedules used in Examples 4 and 5

Second, a possibly infeasible time schedule $(T_v^{init})_{v \in V_P^x}$ has to be determined to be repaired afterwards. The time of the inserted customer i is scheduled as early as possible after its predecessor in the route. This time is propagated to all its successors in the route. For all other vertices v, the time is kept, i.e., $T_v^{init} = T_v$.

Third, to restore feasibility, we adapt the double-justification algorithm as follows: Starting from the constructed schedule $(T_v^{init})_{v \in V_P^x}$ the first phase of double justification iteratively re-schedules FECs and all vertices on levels 3 and 4 as late as possible to obtain a new schedule $(T_v^{late})_{v \in V_P^x}$. This is accomplished by considering the vertices v iteratively in non-increasing order of their current time T_v^{init} (times of bundles in level 2 are fully fixed). At the end of the first phase, a new schedule $(T_v^{late})_{v \in V_P^x}$ has been constructed. In the second phase, all vertices v are iteratively re-scheduled as early as possible in non-decreasing order of their new times T_v^{late}. In both phases, all constraints are checked when re-scheduling a vertex and violations are not allowed to increase.

SC^2_{DJ} succeeds if the final schedule is temporal feasible and respects the capacities of satellites and first-echelon vehicles. The run-time complexity of SC^2_{DJ} is $O(|A_P^x|)$ (same as for SC^2_{fix}), but it is considerably slower than SC^2_{fix} in practice.

While re-scheduling vertices of levels 3 and 4 is straightforward, next we outline how FECs are re-assigned. In the following, let h denote the second-echelon trip serving customer j, starting at satellite s_1 at time T_1, and ending at satellite s_2 at time T_2. FECs in the same bundle are re-assigned in an arbitrary order, but forward FECs are re-assigned before reverse FECs in the first phase. Contrary, in the second phase, reverse FECs are re-assigned before forward FECs.

- In the first phase, a forward FEC j^{fw} is assigned to the latest bundle (s_1, I) with $T_{(s_1,I)} + p_s \leq T_1$ such that the first-echelon vehicle capacity is respected.
- To re-assign a reverse FEC j^{rv} in the first phase, the bundles at the satellite s_2 are considered in non-decreasing order of their time T_v, starting from the time of the current bundle of j^{rv}. The procedure stops as soon as the remaining satellite capacity at the current bundle is lower than q_j^{rv}. Then, j^{rv} is assigned to the latest evaluated bundle that respects the first-echelon vehicle capacity.
- To re-assign a forward FEC j^{fw} in the second phase, the bundles at the satellite of s_1 are considered in non-increasing order of their time T_v, starting from the time of the current bundle of j^{fw}. The procedure stops as soon as the remaining satellite capacity at the current bundle is lower than q_j^{fw}. Then, j^{fw} is assigned to the earliest evaluated bundle that respects the first-echelon vehicle capacity.
- In the second phase, a reverse FEC j^{rv} is assigned to the earliest bundle (s_2, I) with $T_{(s_2,I)} \geq T_2 + p_s$ such that the first-echelon vehicle capacity is respected.

Example 5 *Table 5b presents the time schedule of solution x' in Figure 6b after the first phase of the double justification algorithm. During the as-late-as-possible scheduling, the second-echelon trip $(s, 5, s)$ is shifted first by one time unit. Next, second-echelon trip $(s, 1, 2, s)$ is also postponed from 2 to 3. Last, the forward FEC 5^{fw} is re-assigned from the first bundle to the second which is possible due to the postponement of trip $(s, 5, s)$. Thereby, the load profile becomes feasible.*

Table 5c presents the time schedule after the second phase (left-alignment) of the double-justification. Here, only the second-echelon trip $(s, 1, 2, s)$ is shifted back to start time 2 keeping a feasible satellite load profile. □

4.3 Mixed integer program (MIP)

The MIP component of ITSH is called in Step 11 of Algorithm 1 to assemble trips generated in several iterations of LNS[1] and LNS[2]. It is able to provide a potentially cheaper solution in which

some customer demands are split in the first echelon (we analyze the benefit from split solutions in Section 5.2). In this section, we explain how the MIP interacts with the LNS components: First, we describe the MIP execution conditions that determine when and how often the MIP is solved. Second, we explain why the MIP that is solved is a restricted and refined version of model (1). Third, we explain how the trip pools are filled and filtered. Finally, we describe how the MIP solution is fed-back into the ITSH framework as a current solution for the next LNS iterations.

MIP execution strategy The trip pool management is important for the overall effectiveness of the ITSH algorithm. With trip pools $\bar{\Omega}^1$ and $\bar{\Omega}^2$ containing too many trips, the MIP solver may consume too much computation time. Note that we warm-start the solver with the best-known solution, however, improving it and proving optimality is often computationally costly. Conversely, if the pools are too small, the solver may fail to recombine trips into an improving or even overall feasible solution. To controll the pool size and the computation time granted to the MIP component, ITSH uses a MIP time limit, an adaptive MIP execution condition, and a pool re-initialization policy.

MIP time limit: For every MIP (re-)optimization, an instance-size-dependent time budget Γ_{MIP} bounds the MIP computation time. Preliminary experiments have shown that solving the MIP tends to be much harder when no solution serving all customers has been found yet. Thus, the value of Γ_{MIP} is increased by 10 seconds for each unserved customer (second echelon) and by 5 seconds for each unserved FEC (first echelon) in the current best-known solution. The additional time budget helps to find feasible solutions.

MIP execution conditions: The MIP is executed every η ITSH iterations, when the LNS^2-LNS^1-sequence has failed to improve the best-known feasible solution. Initially, we set $\eta = 1$ and adjust it after each MIP execution depending on the solver result:

- η is increased by 1, if the MIP solution is proven optimal within the time limit, or if the best-known feasible solution is improved in two successive MIP executions.
- η is decreased by 1 (if $\eta \geq 2$), if the MIP solver fails to solve the root node within the time budget or fails to improve the best-known feasible solution twice in a row.

Pool re-initialization policy: A pool re-initialization consists in clearing the trips pools $\bar{\Omega}^1$ and $\bar{\Omega}^2$ keeping only the trips of the best-known solution. In ITSH, this is done at the end of the LNS^2-LNS^1-sequence if the best-known feasible solution has been improved, and every time the MIP is solved.

Restricted and refined MIP The sets of all possible trips Ω^1 and Ω^2 in model (1) are replaced by smaller subsets $\bar{\Omega}^1$ and $\bar{\Omega}^2$, denoted as *trip pools*. This restriction has several consequences:

First, the restricted model can be infeasible. To ensure the computation of an incomplete solution, we introduce auxiliary variables σ_i acting as a penalty for not serving customers $i \in N$. In the refined model's objective function, these variables are penalized with a high penalty cost M.

Second, in a first-echelon trip, the quantities dropped off or collected at each first-echelon satellite visit are given and cannot be altered in model (1). Hence, a first-echelon trip that drops off or collects more than the forward/reverse demand of the second echelon cannot be used in a solution if its loading plan produces some violation of the satellite-capacity constraint. To enable the use of such trips, we introduce additional slack and surplus variables for a potential surplus at the first echelon. More precisely, the variables $u^{\text{fw}}_{s,t}$ ($u^{\text{rv}}_{s,t}$) decreases the quantity of

goods dropped off (collected) by first-echelon vehicles visiting the satellite s at time t in the reformulated flow-conservation (1f)–(1g) and satellite-capacity constraints (1h).

Third, in model (1), fleet-size, flow-conservation, and satellite-capacity constraints (1c), (1e)–(1h) are defined for each point in time $t \in \mathbb{T}$. In the refined MIP, we reduce the number of these constraints (without sacrificing correctness) by identifying times t at which a particular constraint is dominated.

Fourth, preliminary experiments have revealed that prioritizing decisions on whether a satellite is in use can accelerate the MIP solution. For this purpose, we introduce binary indicator variables ζ_s for all $s \in S$. Additional constraints enforce that no first-echelon and no second-echelon trip visiting satellite s can be selected if $\zeta_s = 0$. The solver is configured to prioritize branching on the ζ-variables.

Appendix A.3 provides the complete restricted and refined formulation including precise definitions of its index sets, additional variables, and non-redundant constraints.

Pool accumulation and update All trips generated by the LNS algorithms are added to the trip pools $\bar{\Omega}^1$ and $\bar{\Omega}^2$ (Step 17 of Algorithm 2 in Appendix A.1). To foster a better synchronization between echelons, we create additional trip variables from existing ones having identical walks but different time schedules. Recall that, for each generated trip, four time schedules can be computed: route-EST, route-LST, solution-EST, and solution-LST. We always generate a trip variable with the solution-EST schedule. If η was not decreased after the last call to the MIP-solver, we create for each second-echelon trip additionally a trip variable with either the route-EST time schedule or the route-LST time schedule, selected at random with probability 50%.

We also modify the loading plan of all generated first-echelon trips to grant more flexibility by building new trips with loading plans that try to fully utilize the vehicle capacity, i.e., with $\sum_{k=1}^{n} L_k^{\mathrm{fw}} = \sum_{k=1}^{n} L_k^{\mathrm{rv}} = Q^1$. To meet this condition, the missing forward and reverse quantities of each first-echelon trip are computed and distributed among all first-echelon satellite visits such that the vehicle capacity is never exceeded: For a trip h, the average increase in forward and reverse demand is calculated as $z^{\mathrm{fw}} = (Q^1 - \sum_{k=1}^{n} L_k^{\mathrm{fw}})/n$ and $z^{\mathrm{rv}} = (Q^1 - \sum_{k=1}^{n} L_k^{\mathrm{rv}})/n$, respectively. We then iteratively increase L_k^{fw} by z^{fw} and L_k^{rv} by z^{rv} for $k = 1, \ldots, n$. If these increases violates the vehicle capacity, we decrease until feasibility is restored. Afterward, L_1^{fw} is increased until condition $\sum_{k=1}^{n} L_k^{\mathrm{fw}} = Q^1$ is met. Similarly, L_n^{rv} is increased until condition $\sum_{k=1}^{n} L_k^{\mathrm{rv}} = Q^1$ is met. Note that the loading plan of a first-echelon trip always remains feasible if we increase the dropped off amount at the first visited satellite and the collected amount at the last visited satellite to meet the condition $\sum_{k=1}^{n} L_k^{\mathrm{fw}} = \sum_{k=1}^{n} L_k^{\mathrm{rv}} = Q^1$.

Communication of a MIP solution to the LNS Communicating the solution of the refined MIP to both LNS algorithms consists of the following tasks:

(1) *Assemble the chosen trips into routes:* This is done by solving a simple assignment problem (like in vehicle scheduling).

(2) *Correct the quantities dropped off and collected at each first-echelon satellite visit:* The value of the auxiliary u-variable is subtracted from the quantities dropped off or collected by first-echelon trips at their corresponding first-echelon satellite visit. In case a u-variable can affect several of the chosen first-level trips, the surplus quantity is substracted from the corresponding first-echelon satellite visits such that all quantities remain non-negative.

(3) *Assign FECs to first-echelon satellite visits:* A generalized assignment problem is solved with the MIP solver to assign the FECs to first-echelon satellite visits that provide the corrected quantities dropped off and collected. If no complete feasible assignment exists, we assign as many FECs as possible. Customers whose FEC(s) are unassigned are removed and

placed in the request bank. If such a possibly incomplete assignment violates the satellite-capacity constraints, we iteratively remove customers and their FECs until feasibility is restored. Removed customers are stored together with their FECs in the request bank.

5 Computational results

In this section, we report the results of computational experiments. The algorithm is coded in C++ and compiled with g++ 5.4.0 and option `O3`. We use the MIP solver IBM Ilog CPLEX 20.1.0 (IBM, 2020). Options 'branch up first' and 'emphasize finding high quality feasible solutions earlier' are activated. The experiments were performed on a PC running Linux, Ubuntu 20.04.2 LTS, equipped with an Intel Xeon Gold 6230 clocked at 2.10GHz using a single thread per instance.

Section 5.1 describes the new instances that we used in the experiments. The LNS and MIP components are evaluated in Section 5.2. In Section 5.3, we compare our algorithm on 2E-VRPTW instances of Marques *et al.* (2020a). Finally, the sensitivity analyses of Section 5.4 provide some managerial insights.

5.1 Instances

Since there are no 2E-MTVRP-CSRF instances publicly available (with reverse flow, satellite capacities, and multiple first-echelon trips), we have generated new ones with the following properties:

- The instances have a Euclidean representation over a $[0, 100] \times [0, 100]$ grid. Customers are placed randomly with a uniform distribution on the grid, while the UDC is located at $(50, 100)$. The second-echelon depot and all satellites are randomly placed in $[20, 80] \times [20, 80]$ with a pairwise minimum distance of 20.
- The planning horizon is $\mathbb{T} = \{0, 1, \ldots, 600\}$. Customer time windows are assigned with equal probability: early $[0, 300]$, late $[300, 600]$, or two hours during the day as $[60 + 120\psi, 180 + 120\psi]$ with $\psi \in \{0, 1, 2, 3\}$. The customer service duration is 5 minutes, and all satellites have a transfer time of 15 minutes.
- Customers have either only forward demand, only reverse demand, or both, with probability 50%, 25%, and 25%, respectively. This demand is uniformly drawn from $\{1, 2, 3, 4\}$. The first(second)-echelon vehicle capacity is set to 75 (10).

We systematically vary the following parameters to obtain 36 *instance groups*:

- The number of customers is either 50 or 100. The 50-customer instances have a fleet of a single first-echelon vehicle and 5 second-echelon vehicles, and 100-customer instances have a fleet of 2 first-echelon vehicles and 10 second-echelon vehicles.
- The number of satellites is 2, 4, or 8.
- For all satellites, the satellite capacity is set to the same value $C^{sat} \in \{20, 25, 35, 50, 70, 500\}$. Since 500 is an upper bound for the total demand, this value represents uncapacitated satellites, i.e., $C^{sat} = \infty$.

We have generated 10 instances per group resulting in 360 2E-MTVRP-CSRF instances that are available at `https://logistik.bwl.uni-mainz.de/research/benchmarks/`.

For the 2E-VRPTW benchmark, distances are rounded up to two digits as done by Grangier *et al.* (2016). For the 2E-MTVRP-CSRF benchmark, travel times (in minutes) and distances (=routing costs) are computed as Euclidean distances rounded up to the next integer value (zero digits) as done for the instances described by Marques *et al.* (2020a).

5.2 Evaluation of algorithmic components

First, we evaluate different configurations of the MIP component of the matheuristic. We limit the test instance set to the 60 instances with a tight satellite capacity of $C^{sat} = 25$ (recall that 25 is one third of the capacity of a first-echelon and 2.5 times the capacity of a second-echelon vehicle). Most of these instances are feasible, but the determination of a feasible solution is non-trivial. The overall time limit is 700(2000) seconds for 50(100)-customer instances and the MIP solver has a time budget $\Gamma_{\text{MIP}} = 60(150)$ seconds in each iteration. We compare the performance of the complete configuration (*ITSH*; as presented in Section 4.3) with the following restricted configurations:

- *noMIP*: the MIP is not used.
- *noAdaptTime*: the time budget of the MIP is not dynamically adapted, instead it is fixed to the default value Γ_{MIP}.
- *noVAR-u*: the additional u-variables are not used. Nonetheless, the penalty variables σ are necessary during early stages and, thus, kept in the MIP model.
- *noSplit*: the MIP decomposition allows the split of a customer's demand over different first-echelon satellite visits. Here, solutions of the MIP that split a customer's demand on the first echelon are considered infeasible.

Table 6 presents the results, where for each instance, the best solution found out of 5 runs is considered. Each line corresponds to a group of 10 instances defined by the number $|N|$ of customers and the number $|S|$ of satellites. For each configuration, the entry #f is the number of instances for which a feasible solution has been computed with the configuration. For each of these feasible solutions, we compare their cost to the cost of the respective best solution found by *ITSH*. While *ITSH* provides the reference solutions (average costs are presented in column 'cost'), the average percentage gap to the *ITSH* solution, computed as $100 \cdot (z_{conf} - z_{ITSH})/z_{ITSH}$, is presented in columns 'Gap' for all four configurations.

			ITSH		*noMIP*		*noAdaptTime*		*noVAR-u*		*noSplit*	
$\|N\|$	$\|S\|$	#inst	#f	cost	#f	Gap	#f	Gap	#f	Gap	#f	Gap
50	2	10	10	1 965.0	8	2.01	10	0.65	9	−1.03	10	0.71
	4	10	10	1 725.7	9	2.36	10	0.65	10	2.36	10	0.83
	8	10	10	1 663.1	10	2.54	10	0.35	10	1.24	10	0.09
Total		30	30	1 784.6	27	2.33	30	0.55	29	0.92	30	0.55
100	2	10	9	3 402.1	6	3.23	8	3.66	6	2.54	9	0.14
	4	10	10	3 021.8	9	2.22	9	0.63	10	0.91	10	0.33
	8	10	10	2 572.0	10	3.94	10	−0.21	10	1.66	10	0.03
Total		30	29	2 984.7	25	3.15	27	1.30	26	1.61	29	0.17

Table 6: Comparison of different MIP configurations

The complete configuration *ITSH* computes feasible solutions for 59 of the 60 instances, which is better than any restricted configuration. The solutions computed with configuration *ITSH* are on average better than the solutions computed with any other configuration. The MIP is an essential component of the matheuristic, in particular, when equipped with the adaptive time budget (average gap of approximately 2.7% over all instances). Moreover, using the additional u-variables improve the quality of the solutions. Last, the split of customer demands seems to have only a small impact on the solution quality. This was expected, since the largest gain from the split delivery VRP compared to the VRP occurs when the customer demands are close to half of the vehicle capacity (Archetti and Speranza, 2008), while, in our

case, customer demands are much smaller than the first-echelon vehicle capacity.

Second, we evaluate the impact of the multi-level feasibility checking strategy for the second-echelon LNS presented in Section 4.2.5. Recall that only LNS^2 uses two sufficient feasibility checks. Figure 7 visualizes the number of potential insertions filtered by the insertion feasibility check. We consider an instance with 50 customers, two satellites, and satellite capacity 25. The run of *ITSH* includes more than 17M insertion attempts. Most infeasible insertions (>16M) are detected by NC^2 and directly rejected. The subsequent sort of the insertions further reduces the number of calls to SC^2_{fix} from 1.42M to 654k. Then, SC^2_{fix} identifies approximately half (29k) of the feasible insertions performed. Finally, the very time-consuming re-scheduling with SC^2_{DJ} identifies the other half (27k).

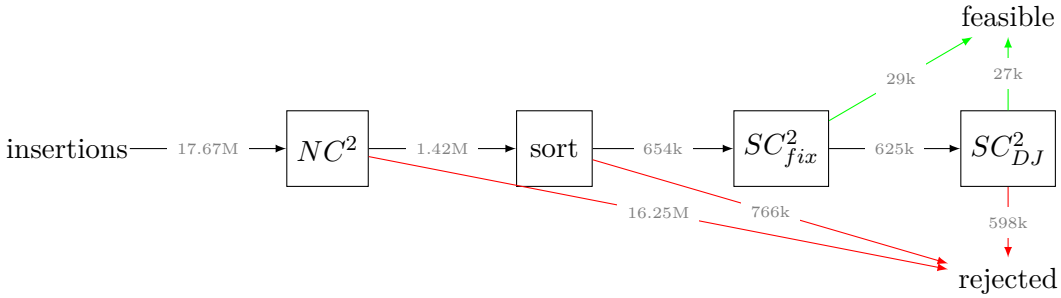


Figure 7: Example of the usage of the feasibility tests

We now analyze the impact of SC^2_{fix} and SC^2_{DJ}. For this experiment, we use a restricted test set consisting of the 30 instances with 100 customers and a satellite capacity of 25:

- When SC^2_{fix} is deactivated, the average cost of a solution increases by 1.8%. The impact on the cost is even more striking given that approximately 90% of the computation time is consumed by the MIP solver. A single evaluation of SC^2_{DJ} takes approximately 450 times more time than the evaluation of SC^2_{fix}. Hence, SC^2_{fix} is a very powerful acceleration procedure to detect feasible insertions fast.
- When SC^2_{DJ} is deactivated in LNS^2, many feasible insertions remain undetected because all undecided insertions after SC^2_{fix} are rejected. As a result, not a single feasible solution has been determined with *ITSH*. The solution process is stuck producing only new solutions similar to the initial infeasible solution. Recall that SC^2_{fix} assumes that the order of operations taking place at each satellite cannot change. Without SC^2_{DJ}, the order of second-echelon arrivals and departures is fixed relative to the current first-echelon satellite visits, which strongly constrains the insertion of vertices into alternative positions.

5.3 Comparison on 2E-MTVRPTW benchmark

We compare the matheuristic (the configuration *ITSH*) with state-of-the-art algorithms for the 2E-VRPTW. For the 2E-MTVRP-SS, Grangier *et al.* (2016) modified and extended instances from the well-known benchmark set of Solomon by placing the UDC to the coordinate (50,150) and adding eight satellites. Customer time windows are shifted to account for first-echelon and satellite processes. We cannot compare our results with those of Grangier *et al.* (2016), who considered exact synchronization at satellites between first-echelon and second-echelon vehicles. Recently, Marques *et al.* (2020a) extended this benchmark set by creating smaller instances with 25, 50, and 75 customers. They consider only the first customers in the respective 100-customer instance. The smaller instances have four to six satellites.

We can directly compare our method with the results of the *branch-price-and-cut* (BPC)

algorithm of Marques *et al.* (2020a) for the 2E-MTVRPTW. To do so, the capacity of satellites is assumed infinite and we minimize the vehicle-related cost that includes a fixed cost of 50 (25) per first(second)-echelon vehicle. For a fair comparison, we only consider the instances for which Marques *et al.* report a feasible solution value, because we use their number of vehicles as our fleet-size limit.

Table 7 summarizes the comparison with the exact BPC algorithm proposed by Marques *et al.* (2020a) that was run with a 10-hour time limit.

	$\|N\| = 25$ $\Gamma_{\text{tot}} = 30$			$\|N\| = 50$ $\Gamma_{\text{tot}} = 120$			$\|N\| = 75$ $\Gamma_{\text{tot}} = 500$			$\|N\| = 100$ $\Gamma_{\text{tot}} = 800$		
		BPC			BPC			BPC			BPC	
Group	#	time	Gap	#	time	Gap	#	time	Gap	#	time	Gap
`c100`	0/9	23	0.50	0/9	161	0.29	3/8	9 621	1.01	0/7	16 503	1.91
`r100`	0/12	62	0.15	0/12	882	1.13	2/10	27 049	0.97	0/3	19 970	0.41
`rc100`	0/8	10	0.55	0/8	89	1.10	0/7	23 168	1.07	3/4	36 092	−0.75
`c200`	0/8	585	0.39	1/8	8 294	2.41	0/4	7 809	2.35	0/1	15 935	0.44
`r200`	0/8	5 169	3.34	0/1	1 714	0.07						
`rc200`	0/8	356	5.16	0/6	7 847	2.12						
All	0/53	942	1.54	1/44	2 907	1.30	5/29	18 932	1.19	3/15	22 382	0.80

Table 7: Performance of the matheuristic *ITSH* on the 2E-VRPTW instances compared to the BPC algorithm of Marques *et al.* (2020a)

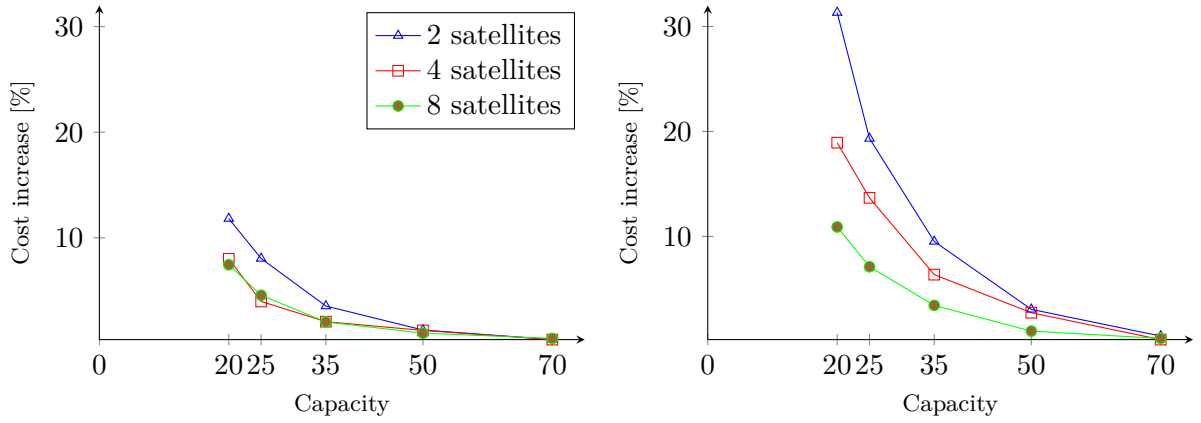
The two numbers x/y in columns headed '#' are the number y of instances considered and the number x of new best solutions computed with *ITSH*. If no feasible solution was provided by the BPC algorithm, the cells are left blank. Our algorithm has a fixed time budget Γ_{tot} (in seconds) controlled by the number $|N|$ of customers (see entries Γ_{tot} in the second line of the table's header; associated CPLEX time limits Γ_{MIP} are set to 2, 10, 40, and 60 seconds, respectively). In contrast, run times of the BPC algorithm vary, average times (in seconds) are presented in columns 'BPC time'. Moreover, the table shows the average gaps ('Gaps'; in percent) between the best solutions found by our algorithm (five independent runs per instance) and the best solutions found by Marques *et al.* (2020a). Detailed instance-by-instance results can be found in Appendix A.4.

The matheuristic *ITSH* delivers feasible solutions to all instances for which the BPC algorithm also computes feasible solutions. Moreover, *ITSH* finds nine new best solutions for the benchmark set (leading to a negative gap in the group `rc100` with $|N| = 100$). We interpret the results in the following way: Instances with more customers, wider time windows and longer routes (series 2) are more difficult for the BPC algorithm. As a heuristic, *ITSH* can provide good feasible solutions with average gaps that may even decrease for instances with more customers. However, some larger gaps (up to 6.41% for a 25-customer instance in group `rc200`) show that *ITSH* can still fail to reach some close-to-optimal solution values, while proven optima are computed by the BPC algorithm in several hours of computation time.

5.4 Sensitivity analyses

We now provide sensitivity analyses that show the impact of capacitated satellites and reverse flows on costs. All results are based on the best solution computed with the complete *ITSH* configuration in five independent runs for each instance.

Cost of satellites capacity Figure 8 depicts the impact of satellites capacity for 50- and 100-customer instances. In both subfigures, the average increase in overall routing cost (in percent) relative to the respective uncapacitated instance (with $C^{sat} = \infty$) is displayed for groups of instances differing in the capacity of the satellites. Three plots distinguish results for 2, 4, and 8 satellites. To put the provided capacities in perspective, the average total forward demand is 94 (188) and the average total reverse demand is 64 (124) on the instances with 50 (100) customers.



(a) Instances with 50 customers

(b) Instances with 100 customers

Figure 8: Average cost increase (in percent) compared to the respective instance with infinite satellite capacity

As expected, the average cost increase diminishes with larger satellite capacities. Moreover, the more satellites available, the smaller the cost increase (with the exception that 4 and 8 satellites make nearly no difference for the 50-customer instances). The cost increase is substantially smaller for the 50-customer than for the 100-customer instances which can be explained by the fact that the capacity of a satellite is identical in contrast to the approximately doubled total demand. No feasible solutions could be found for some instances with two satellites and a capacity of 20 and 25 due to the (overly) restrictive capacity. We disregard these instances.

The convex shape of the curves indicates that the value of additional capacity units is decreasing with the absolute capacity: For the 100-customer instances, five additional units for extending the capacity from 20 to 25 have a much higher impact (8.08%) than the 20-unit increase for extending the capacity from 50 to 70 (1.96%). As an extreme example, the difference between the satellite capacity of 70 and uncapacitated satellites becomes negligible (0.32%).

It is interesting to see that doubling the satellite capacity leads to a substantially larger cost decrease than doubling the number of satellites. Nevertheless, a small number and size of satellites can be a reasonable design given the outstanding prices of space in some cities (e.g., $£14355/m^2/year$ in London, Furmanik, 2019).

Gain of simultaneously handling forward and reverse flows To quantify the gain that results from the integration of forward and reverse flows, we create twin instances, i.e., for each original instance (with both forward and reverse demands), the first twin has only forward demands and the second twin has only reverse demands. We denote by c^i the cost of a best solution found by *ITSH* for the original instances and by c^{fw} (c^{rv}) the cost found for the new instances with only forward (reverse) demands. In Table 8, the average gain of the integrated approach, computed as $(c^{fw} + c^{rv} - c^i)/c^i$, is shown grouped by the number of customers and satellites as well as satellite capacity. Note that the comparison underestimates the cost of the

non-integrated problems, since both new twin instances (only forward or only reverse flows) can use the complete satellite capacity, while forward and reverse flows of the original instance compete for satellite capacity. As a result, the non-integrated solution with cost $c^{fw} + c^{rv}$ would be even more expensive when the same capacity were shared and allocated. However, any other *a priori* assignment of satellite capacities (e.g., half of the satellite capacity for each direction) would also be questionable.

		Satellite capacity C^{sat}					
$\|N\|$	$\|S\|$	20	25	35	50	70	∞
50	2	23.55	27.99	33.08	35.74	36.89	37.33
	4	30.80	34.07	35.85	36.50	37.45	37.83
	8	31.50	33.12	35.09	36.08	36.21	36.59
Total		29.55	31.73	34.67	36.11	36.85	37.25
100	2	11.27	22.75	31.04	37.55	39.74	40.38
	4	22.47	27.02	32.98	36.05	39.00	39.13
	8	33.74	37.16	39.53	42.19	42.95	43.24
Total		25.30	29.19	34.52	38.60	40.56	40.92

Table 8: Average gain resulting from simultaneously handling forward and reverse flows

Some general statements can be made: The gain from the integration tends to increase with the total space provided by all satellites and the total quantity of goods to transport. It increases from 11% for instances with two small satellites and 100 customers, to 43% for instances with eight uncapacitated satellites and 100 customers. On the one hand, the gain consistently increases with the satellite capacity. On the other hand, the gain tends to increase with the number of satellites, but this trend is not consistent for the 50-customer instances and $|S| = 4$ or 8. We attribute this behavior to the fact that instances with eight satellites are much more difficult for the matheuristic compared to those with only four satellites, since in the former not all satellites have to be used.

6 Conclusion

Two-echelon distribution systems are a cornerstone of city logistics. In this paper, we study important functionalities of two-echelon systems that are rarely addressed in route planning algorithms: forward and reverse flows, multiple vehicle trips, and storage capacities at satellites. We introduce the 2-echelon multi-trip vehicle routing problem with capacitated satellites and reverse flows (2E-MTVRP-CSRF) to integrate these new features into one new problem, for which we provided a trip-based integer programming formulation.

To solve the 2E-MTVRP-CSRF, we design a decomposition-based matheuristic denoted as iterative two-stage heuristic (ITSH). ITSH uses two echelon-tailored LNS algorithms, each of them optimizing one echelon while (a major part of) the other echelon is fixed. To efficiently test the feasibility of insertions, both LNS algorithms rely on a sequence of constant-time and low-complexity tests. A first sufficient test fixes the order of operations at satellites to allow a fast evaluation, while the final test employs the double-justification algorithm to allow a re-scheduling of the order. Additionally, a MIP solver is used with a restricted and refined version of a trip-based formulation to recombine trips generated in different LNS iterations. Thanks to its more global vision of the synchronization constraints between both echelons, the MIP

component is able to improve the solutions proposed by the LNS^1 and LNS^2 algorithms. ITSH uses adaptive mechanisms to efficiently share the computation time between the search for new trips and the combination of existing trips.

The computational study evaluates the ITSH matheuristic and analyze the impact of the main problem features. We find that:

- ITSH provides competitive results on instances of the 2E-MTVRPTW in comparison to the exact algorithm of Marques *et al.* (2020a).
- The combination of heuristic and exact components is a key success factor of the algorithm.
- Routing costs are sensitive to the satellite capacity. For example, on the 100-customer instances with two satellites, reducing a large satellite capacity of 70 units to a tight capacity of 20 units results in a 30% cost increase on average.
- Doubling the capacity of well-located satellites leads to a much larger decrease in the routing costs than doubling the number of satellites.
- Integrating forward and backward flows can generate considerable savings (approximately 40% on average for 100-customer instances and uncapacitated satellites). These savings decrease with tighter satellite capacities but remain substantial (always more than 10%).

Given the complexity of the 2E-MTVRP-CSRF, there are many avenues for future research trying to improve and accelerate the solution, whether exact or heuristic. In particular, the feasibility problem related to the satellite capacity with forward and reverse flows is a challenging scheduling problem. For future studies, different backhauling policies (mixed deliveries, strict backhauling etc.) could be compared at the second echelon. Furthermore, customer visits by first-echelon vehicles could be integrated to better reflect practices evolving in the field (Nolz *et al.*, 2020).

Acknowledgement

This research was supported by the Agence Nationale de la Recherche (ANR) under grant ANR-17-CE22-0015 and Deutsche Forschungsgemeinschaft (DFG) under grant IR 122/8-1. This support is gratefully acknowledged.

References

Amarouche, Y., Guibadj, R. N., and Moukrim, A. (2018). A neighborhood search and set cover hybrid heuristic for the two-echelon vehicle routing problem. In R. Borndörfer and S. Storandt, editors, *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, (ATMOS 2018)*, volume 65, pages 11:1–11:15, Helsinki, Finland.

Anderluh, A., Hemmelmayr, V. C., and Nolz, P. C. (2017). Synchronizing vans and cargo bikes in a city distribution network. *Central European Journal of Operations Research*, **25**(2), 345–376.

Anderluh, A., Larsen, R., Hemmelmayr, V. C., and Nolz, P. C. (2020). Impact of travel time uncertainties on the solution cost of a two-echelon vehicle routing problem with synchronization. *Flexible Services and Manufacturing Journal*, **32**(4), 806–828.

Archetti, C. and Speranza, M. G. (2008). The Split Delivery Vehicle Routing Problem: A Survey. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43, pages 103–122. Springer US, Boston, MA.

Bektaş, T., Crainic, T. G., and Woensel, T. V. (2017). From managing urban freight to smart city logistics networks. In *Network Design and Optimization for Smart Cities*, Series on Computers and Operations Research, pages 143–188. World Scientific.

Belgin, O., Karaoglan, I., and Altiparmak, F. (2018). Two-echelon vehicle routing problem with simultaneous pickup and delivery: Mathematical model and heuristic approach. *Computers & Industrial Engineering*, **115**, 1–16.

Breunig, U., Schmid, V., Hartl, R. F., and Vidal, T. (2016). A large neighbourhood based heuristic for two-echelon routing problems. *Computers & Operations Research*, **76**, 208–225.

Bruck, B. P. and Iori, M. (2017). Non-elementary formulations for single vehicle routing problems with pickups and deliveries. *Operations Research*, **65**(6), 1597–1614.

Brucker, P. and Knust, S. (2012). *Complex scheduling*. Springer-Verlag Berlin Heidelberg, Heidelberg New York.

Castillo-Salazar, J. A., Landa-Silva, D., and Qu, R. (2016). Workforce scheduling and routing problems: literature survey and computational study. *Annals of Operations Research*, **239**(1), 39–67.

Christiaens, J. and Vanden Berghe, G. (2020). Slack induction by string removals for vehicle routing problems. *Transportation Science*, **54**, 417–433.

Crainic, T. G., Ricciardi, N., and Storchi, G. (2009). Models for evaluating and planning city logistics systems. *Transportation Science*, **43**(4), 432–454.

Crainic, T. G., Errico, F., Rei, W., and Ricciardi, N. (2012). Integrating c2e and c2c traffic into city logistics planning. *Procedia - Social and Behavioral Sciences*, **39**, 47–60.

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

Dellaert, N., Dashty Saridarq, F., van Woensel, T., and Crainic, T. G. (2019). Branch-and-price–based algorithms for the two-echelon vehicle routing problem with time windows. *Transportation Science*, **53**(2), 463–479.

Dellaert, N., van Woensel, T., Crainic, T. G., and Dashty Saridarq, F. (2021). A multi-commodity two-echelon capacitated vehicle routing problem with time windows: Model formulations and solution approach. *Computers & Operations Research*, **127**, 105154.

Dell'Amico, M., Righini, G., and Salani, M. (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, **40**(2), 235–247.

Demir, E., Bektaş, T., and Laporte, G. (2012). An adaptive large neighborhood search heuristic for the Pollution-Routing Problem. *European Journal of Operational Research*, **223**(2), 346–359.

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

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

Dumez, D., Tilk, C., Irnich, S., Lehuédé, F., and Péton, O. (2021). Hybridizing large neighborhood search and exact methods for generalized vehicle routing problems with time windows. *EURO Journal on Transportation and Logistics*, **10**, 100040.

El Hachemi, N., Gendreau, M., and Rousseau, L.-M. (2011). A hybrid constraint programming approach to the log-truck scheduling problem. *Annals of Operations Research*, **184**(1), 163–178.

El Hachemi, N., Gendreau, M., and Rousseau, L.-M. (2013). A heuristic to solve the synchronized log-truck scheduling problem. *Computers & Operations Research*, **40**(3), 666–673.

Elbert, R. and Friedrich, C. (2020). Urban consolidation and cargo bikes: a simulation study. *Transportation Research Procedia*, **48**, 439–451.

Fikar, C. and Hirsch, P. (2017). Home health care routing and scheduling: A review. *Computers & Operations Research*, **77**, 86–95.

Froger, A., Mendoza, J. E., Jabali, O., and Laporte, G. (2017). A matheuristic for the electric vehicle routing problem with capacitated charging stations. Technical Report CIRRELT-2017-31, CIRRELT, Montréal, Canada.

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

Gonzalez-Feliu, J., Perboli, G., Tadei, R., and Vigo, D. (2008). The two-echelon capacitated vehicle routing problem. Archive ouverte HAL 00879447, Centre pour la Communication Scientifique Directe. `https://halshs.archives-ouvertes.fr/halshs-00879447`.

Grangier, P., Gendreau, M., Lehuédé, F., and Rousseau, L.-M. (2016). An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. *European Journal of Operational Research*, **254**(1), 80–91.

Grangier, P., Gendreau, M., Lehuédé, F., and Rousseau, L.-M. (2019). The vehicle routing problem with cross-docking and resource constraints. *Journal of Heuristics*, **27**, 31–61.

Grimault, A., Bostel, N., and Lehuédé, F. (2017). An adaptive large neighborhood search for the full truckload pickup and delivery problem with resource synchronization. *Computers & Operations Research*, **88**, 1–14.

Hansen, P., Mladenović, N., and Moreno Pérez, J. A. (2009). Variable neighbourhood search: methods and applications. *Annals of Operations Research*, **175**(1), 367–407.

Helber, S. and Sahling, F. (2010). A fix-and-optimize approach for the multi-level capacitated lot sizing problem. *International Journal of Production Economics*, **123**(2), 247–256.

IBM (2020). CPLEX. `ibm.com/analytics/cplex-optimizer`.

Irnich, S. (2007). Resource extension functions: properties, inversion, and generalization to segments. *OR Spectrum*, **30**(1), 113–148.

Irnich, S., Toth, P., and Vigo, D. (2014). The family of vehicle routing problems. In *Vehicle Routing*, chapter 1, pages 1–33. Society for Industrial & Applied Mathematics (SIAM).

Jie, W., Yang, J., Zhang, M., and Huang, Y. (2019). The two-echelon capacitated electric vehicle routing problem with battery swapping stations: Formulation and efficient methodology. *European Journal of Operational Research*, **272**(3), 879–904.

Li, H., Liu, Y., Jian, X., and Lu, Y. (2018). The two-echelon distribution system considering the real-time transshipment capacity varying. *Transportation Research Part B: Methodological*, **110**(2), 239–260.

Li, H., Liu, Y., Chen, K., and Lin, Q. (2020). The two-echelon city logistics system with on-street satellites. *Computers & Industrial Engineering*, **139**, 105577.

Marques, G., Sadykov, R., Deschamps, J.-C., and Dupas, R. (2020a). A branch-cut-and-price approach for the single-trip and multi-trip two-echelon vehicle routing problem with time windows. Archive ouverte HAL 03139799, Centre pour la Communication Scientifique Directe. `https://hal.inria.fr/hal-03139799`.

Marques, G., Sadykov, R., Deschamps, J.-C., and Dupas, R. (2020b). An improved branch-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem. *Computers & Operations Research*, **114**, 104833.

Masson, R., Lehuédé, F., and Péton, O. (2013). Efficient feasibility testing for request insertion in the pickup and delivery problem with transfers. *Operations Research Letters*, **41**(3), 211–215.

Masson, R., Lehuédé, F., and Péton, O. (2014). The dial-a-ride problem with transfers. *Computers & Operations Research*, **41**, 12–23.

Mhamedi, T., Andersson, H., Cherkesly, M., and Desaulniers, G. (2020). A branch-price-and-cut algorithm for the two-echelon vehicle routing problem with time windows. Cahiers du GERAD G-2020-63, GERAD, Montréal, Canada. `https://www.gerad.ca/fr/papers/G-2020-63`.

Miller, R. W. (1963). *Schedule, cost, and profit control with PERT: A comprehensive guide for program management*. McGraw-Hill.

Mühlbauer, F. and Fontaine, P. (2021). A parallelised large neighbourhood search heuristic for the asymmetric two-echelon vehicle routing problem with swap containers for cargo-bicycles. *European Journal of Operational Research*, **289**(2), 742–757.

Muñuzuri, J., Grosso, R., Cortés, P., and Guadix, J. (2013). Estimating the extra costs imposed on delivery vehicles using access time windows in a city. *Computers, Environment and Urban Systems*, **41**, 262–275.

Nagata, Y. and Bräysy, O. (2009). A powerful route minimization heuristic for the vehicle routing problem with time windows. *Operations Research Letters*, **37**(5), 333–338.

Nolz, P. C., Absi, N., Cattaruzza, D., and Feillet, D. (2020). Two-echelon distribution with a single capacitated city hub. *EURO Journal on Transportation and Logistics*, **9**(3), 100015.

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.

Paraskevopoulos, D. C., Laporte, G., Repoussis, P. P., and Tarantilis, C. D. (2017). Resource constrained routing and scheduling: Review and research prospects. *European Journal of Operational Research*, **263**(3), 737–754.

Perboli, G., Tadei, R., and Fadda, E. (2018). New valid inequalities for the two-echelon capacitated vehicle routing problem. *Electronic Notes in Discrete Mathematics*, **64**, 75–84.

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, **34**(8), 2403–2435.

Pisinger, D. and Ropke, S. (2019). Large neighborhood search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 272, pages 99–127. Springer International Publishing, Cham.

Ropke, S. and Pisinger, D. (2006a). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, **40**(4), 455–472.

Ropke, S. and Pisinger, D. (2006b). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, **171**(3), 750–775.

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

Savelsbergh, M. W. P. (1992). The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing*, **4**(2), 146–154.

Schmid, V., Doerner, K. F., Hartl, R. F., Savelsbergh, M. W., and Stoecher, W. (2009). A hybrid solution approach for ready-mixed concrete delivery. *Transportation Science*, **43**(1), 70–85.

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.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In G. Goos, J. Hartmanis, J. van Leeuwen, M. Maher, and J.-F. Puget, editors, *Principles and Practice of Constraint Programming — CP98*, volume 1520, pages 417–431. Springer Berlin Heidelberg, Berlin, Heidelberg.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, **35**(2), 254–265.

Subramanian, A., Uchoa, E., Pessoa, A. A., and Ochi, L. S. (2013). Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery. *Optimization Letters*, **7**(7), 1569–1581.

Valls, V., Ballestín, F., and Quintanilla, S. (2005). Justification and RCPSP: A technique that pays. *European Journal of Operational Research*, **165**(2), 375–386.

Wang, K., Shao, Y., and Zhou, W. (2017). Matheuristic for a two-echelon capacitated vehicle routing problem with environmental considerations in city logistics service. *Transportation Research Part D: Transport and Environment*, **57**, 262–276.

Wiest, J. D. (1964). Some properties of schedules for large projects with limited resources. *Operations Research*, **12**(3), 395–418.

Wong, R. T. (2008). Vehicle routing for small package delivery and pickup services. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 475–485. Springer US.

A Appendix

A.1 Large Neighborhood Search (LNS)

In this section, we detail LNS[1] and LNS[2] algorithms. They are both based on Algorithm 2 but differ in the LNS operators and parameters used.

Acceptance criterion Both LNS algorithms use a record-to-record acceptance criterion with *modified cost* as proposed in (Pisinger and Ropke, 2007):

$$f'(x) = z(x) \cdot \left(1 + \beta \cdot \frac{B(x)}{|N|}\right), \tag{2}$$

where $z(x)$ is the objective function as defined in (1a), $B(x)$ is the number of unserved customers and FECs (request bank), and β a penalty factor. In LNS^1, we replace additionally $z(x)$ by $z'(x)$ to mitigate undesired solutions in which a first-echelon vehicle successively visits the same satellite, i.e., utilizing it as a waiting station. The modified objective function $z'(x)$ is equal to $z(x)$ plus a penalty for successive visits of first-echelon vehicles to the same satellite.

LNS Algorithm The algorithm starts from an initial solution x, obtained as described in Section A.2. x^* denotes the best solution found that is returned at the end of the algorithm. The pool $\bar{\Omega}$ stores all trips constructed in the course of the algorithm. The index i counts the number of iterations performed since the best-found solution x^* was improved or since the last large destruction happened. A large destruction step is performed if i is equal to the parameter ω. Similarly, the index i' counts the number of large destruction steps performed since the last feasible solution was found and index j counts the overall number of large destruction steps conducted. We denote by Σ^- the set of destroy operators and by Σ^+ the set of repair operators. $\Sigma^-|_{\text{local}}$ denotes the set of local destroy operators, i.e., operators used for small destruction steps.

LNS stops after a predefined number ζ of large destruction steps or if no feasible solution (serving all customers/FECs) has been found during the last ζ' large destruction steps (Step 4). In order to accelerate the interactions between the two LNS, we chose $\zeta' \ll \zeta$ to find a feasible solution faster during the early stages of ITSH.

In each iteration, a repair operator is randomly selected in Σ^+ (Step 5). Depending on the value of i, either a small destruction step (Steps 7–9) or a large destruction step (Steps 11–15) is conducted. In the former case, a copy x' of the current solution is created, a local destroy operator is selected in $\Sigma^-|_{\text{local}}$, and the destruction size is randomly selected in $[\delta_{\text{small}}, \Delta_{\text{small}}]$. If a large destruction step is conducted, a copy x' of the best-found solution x^* is created, the destroy operator is randomly selected in Σ^-, and the destruction size in $[\delta_{\text{large}}, \Delta_{\text{large}}]$. Either way, the selected operators, σ^- and σ^+, are then applied to the copied solution x' at Step 16.

In Step 17, the trips of the newly generated solution are added to the trip pool $\bar{\Omega}$. After a large destruction step, solution x' is always accepted as the new current solution x on Step 20. If a small destruction step was conducted, the new solution x' is accepted as the new current solution x only if its modified cost is less than ϵ percent worse than the modified cost of the best-found solution x^* (Step 23), calculated with equation (2). In Steps 26–28, the best-found solution is updated.

Operators Below, we briefly describe all LNS operators used. Symbols [1], [2] and [1,2] identify the operators used only in LNS[1], or LNS[2], or both. All operators are chosen with the same probability in LNS[1] and LNS[2]. During a large destruction, there is 20% chance of taking a local destroy operator. The *local destroy operators* are:

Algorithm 2: LNS with large and small destruction

x, $x^* \leftarrow$ initial solution
pool of trips $\bar{\Omega} \leftarrow \emptyset$
$i, i', j \leftarrow 0$
while $j < \zeta$ ***and*** $i' < \zeta'$ **do**
    randomly select a repair operator $\sigma^+ \in \Sigma^+$
    if $i < \omega$ **then**
        $x' \leftarrow x$
        randomly select a destroy operator $\sigma^- \in \Sigma^-|_{\text{local}}$
        randomly select a destruction size $\Phi \in [\delta_{\text{small}}, \Delta_{\text{small}}]$
    else
        $x' \leftarrow x^*$
        randomly select a destroy operator $\sigma^- \in \Sigma^-$
        randomly select a destruction size $\Phi \in [\delta_{\text{large}}, \Delta_{\text{large}}]$
        $j \leftarrow j + 1$
        $i' \leftarrow i' + 1$
    $x' \leftarrow \sigma^+(\sigma^-(x', \Phi))$
    store the trips of x' into the pool $\bar{\Omega}$
    if x' *is feasible* **then**
        $i' \leftarrow 0$
    if $i = \omega$ **then**
        $x \leftarrow x'$
        $i \leftarrow 0$
    else if $f'(x') < (1 + \epsilon) \cdot f'(x^*)$ **then**
        $x \leftarrow x'$
        $i \leftarrow i + 1$
    if $f'(x') < f'(x^*)$ **then**
        $x^* \leftarrow x'$
        $i \leftarrow 0$
return x^*

[1,2] *Route (split) string removal* (Christiaens and Vanden Berghe, 2020): Removes sequences of FECs/customers in different routes of the considered solution. We also implement the *split* version of this operator, which conserve a sub-string of vertices in the middle of the string to be removed.

[1,2] *Satellite (split) string removal*: This operator is equivalent to the route (split) string removal on consecutive operations at a satellite. In LNS^2, if a FEC is removed, the associate customer is removed from its second-echelon trip.

[2] *Distance related removal* (Ropke and Pisinger, 2006b): Removes customers that are close to each other with respect to the Euclidean distance.

[2] *Bundle removal*: Randomly selects a bundle and removes it together with the corresponding customers in the second-echelon.

[2] *Cluster removal* (Pisinger and Ropke, 2007): Removes customers that are served by the same trip. A trip is randomly selected and Kruskal's algorithm is run on the arcs of this trip until two clusters remain. One cluster is randomly chosen and all its customers are removed from the solution.

The **large destroy operators** are:

[2] *Random customer removal* (Ropke and Pisinger, 2006b): Randomly removes customers.

[1] *Random bundle removal*: Randomly selects bundles and removes them. It is equivalent to random removal applied to bundles.

[1] *Worst bundle removal* (derived from worst removal, Ropke and Pisinger (2006a)): Iteratively removes the bundle whose removal decreases the objective function the most.

[2] *Historical knowledge node removal* (Demir *et al.*, 2012): The minimum cost of each customer is recorded over all past LNS iterations. The cost of a customer i in a solution x is the difference $f(x) - f(x')$, where x' is x without the visit of customer i. The operator iteratively removes the customers with the largest difference between their current cost and their lowest cost. It can be seen as a history-biased worst removal.

[1,2] *Trip removal and Route removal* (Nagata and Bräysy, 2009): Removes a randomly selected trip/route.

Following Christiaens and Vanden Berghe (2020), the used *repair operators* are all list heuristics. In these operators, unserved FECs/customers are first sorted according to one of the following simple rules and then inserted one by one with the procedure described in Sections 4.2.2 and 4.2.4. For each iteration, one rule is selected at random with identical selection probabilities.

[1,2] *Random order*: FECs/customers are ordered at random.

[1,2] *Largest first*: FECs/customers are sorted in non-increasing order of their total demand quantity.

[2] *Farthest first*: Customers are sorted in non-increasing order of their distance to the nearest satellite.

[2] *Closest first*: Customers are sorted in non-decreasing order of their distance to the nearest satellite.

[1,2] *Earliest first*: FECs/customers are sorted in non-decreasing order of their ready date/start of their time window. The ready date of all forward FECs is 0 and the ready date of a reverse FECs is the end of the second-echelon trip that serves their customer.

[1,2] *Latest first*: FECs/customers are sorted in non-increasing order of their ready-date/start of their time window.

[1,2] *Narrow first*: FECs/customers are sorted in non-decreasing order of time window width.

LNS parameters The numeric parameters for both LNS^1 and LNS^2 are summarized in Table 9. The values of these parameters were obtained from preliminary experiments on 2E-MTVRP-CSRF and 2E-MTVRPTW instances.

Parameter	Notation	LNS^1	LNS^2
Large destruction frequency	ω	$9\|N\|^{0.75}$	$9\|N\|^{1.0}$
Number of large destructions	ζ	30	5
Number of large destructions without improvement	ζ'	5	1
Minimal size of small destructions	δ_{small}	1%	1%
Maximal size of small destructions	Δ_{small}	15%	10%
Minimal size of large destructions	δ_{large}	20%	10%
Maximal size of large destructions	Δ_{large}	100%	30%
Penalty for unserved FEC /customer	β	10	25
Range of acceptance	ϵ	2%	2%
Penalty for successive visits to the same satellite	-	1	1

Table 9: LNS parameters

A.2 Initial solution

During the first iteration of ITSH (Algorithm 1), LNS^2 is called first in Step 4. In order to get initial (fixed) first-echelon routes, we create four single-trip routes per satellite s. These routes just contain a single visit to the satellite s and the satellite visits of different routes are evenly spread over the time horizon. This procedure is repeated F^1 times for each satellite. Clearly these are far too many routes but in this way, the second-echelon routes can rely on nearly unlimited supply at each satellite but still needs to synchronize with the given visit times.

To initialize the second-echelon routes, each customer demand is assigned to its nearest satellite. The available second-echelon vehicles are then assigned to satellites such that the proportion of vehicles assigned to a satellite is proportional to the proportion of customer demands assigned to it. Each second-echelon vehicle route is initialized with an empty trip from the second-echelon depot to its assigned satellite, an empty trip starting and ending at this satellite and finally, an empty trip from its satellite back to the depot. Initially, all customers and their FECs are put into the request banks.

When LNS^1 is called for the first time in Step 6, the first-echelon routes are initialized with an empty trip starting and ending at the UDC. Initially, all FECs are put into the request bank.

A.3 Restricted and refined MIP

Let $\bar{\Omega}^1$ be the set of first-echelon trips and let $\bar{\Omega}^2$ be the set of second-echelon trips generated in the course of the LNS algorithms, modified according to Section 4.3. The restricted and refined model uses the following additional variables:

$\sigma_i \geq 0$: Penalty variable for not serving customer $i \in N$.

$u^{\text{fw}}_{s,t} \geq 0$: Counts the surplus amount of forward demand that is not dropped off at satellite $s \in S$ at time t by first-echelon trips.

$u^{\text{rv}}_{s,t} \geq 0$: Counts the surplus amount of reverse demand that is not collected at satellite $s \in S$ at time t by first-echelon trips.

$\zeta_s \in \{0,1\}$: Indicates whether satellite $s \in S$ is utilized.

Moreover, we remove constraints (1c)–(1h) for those time periods in which the constraints are dominated.

- A $\leq$-constraint for a time period t is deemed dominated if the $\leq$-constraint for time period $t+1$ is identical or tighter, i.e., the left-hand side contains the same or more positive and the same or less negative terms.
- A $\geq$-constraint is identical or tighter if the left-hand side contains the same or less positive and the same or more negative terms.

For example, for constraints (1f) for satellite $s \in S$, it suffices to consider only those points in time at which a second-echelon trip starts.Therefore, we introduce the following sets of time periods:

- $\mathbb{T}^{1,\text{start}} := \{t \in \mathbb{T} : \exists h = (P,T,L) \in \bar{\Omega}^1 \text{ with } s_0 = s \text{ and } T_0 = t\}$, the time periods at which a first-echelon trips starts from the UDC o^1.
- $\mathbb{T}^{1,\text{visit}}_s := \{t \in \mathbb{T} : \exists h = (P,T,L) \in \bar{\Omega}^1 \text{ with } s = i_k \in P \text{ and } T_{i_k} = t\}$, the time periods at which a first-echelon trip visits a satellite $s \in S$.
- $\mathbb{T}^{2,\text{start}}_s := \{t \in \mathbb{T} : H^{\bar{\Omega}^{2}+}_{s,t} \neq \emptyset\}$, the time periods at which a second-echelon trip starts at a satellite $s \in S$.
- $\mathbb{T}^{2,\text{end}}_s := \{t \in \mathbb{T} : H^{\bar{\Omega}^{2}-}_{s,t} \neq \emptyset\}$, the time periods at which a second-echelon trip ends at a satellite $s \in S$.

The restricted and refined model reads as follows:

$$\min \quad \sum_{h\in\bar{\Omega}^1} c^1_h x^1_h + \sum_{h\in\bar{\Omega}^2} c^2_h x^2_h + \sum_{i\in N} M\sigma_i \tag{3a}$$

$$\text{s.t.} \quad \sum_{h\in\bar{\Omega}^2} \alpha_{h,i} x^2_h + \sigma_i = 1 \qquad \forall i \in N \tag{3b}$$

$$\sum_{h\in\bar{\Omega}^1} \beta_{h,t} x^1_h \leq |F^1| \qquad \forall t \in \mathbb{T}^{1,\text{start}} \tag{3c}$$

$$\sum_{t\in\mathbb{T}} \sum_{h\in H^{\bar{\Omega}^{2}+}_{o^2,t}} x^2_h = \sum_{t\in\mathbb{T}} \sum_{h\in H^{\bar{\Omega}^{2}-}_{o^2,t}} x^2_h = |F^2| \tag{3d}$$

$$\sum_{t'\leq t-p_s} \sum_{h\in H^{\bar{\Omega}^{2}-}_{s,t'}} x^2_h - \sum_{t'\leq t} \sum_{h\in H^{\bar{\Omega}^{2}+}_{s,t'}} x^2_h \leq 0 \qquad \forall s \in S, t \in \mathbb{T}^{2,\text{start}}_s \tag{3e}$$

$$\sum_{t'\leq t-p_s} \Big(\sum_{h\in\bar{\Omega}^1} \gamma^{\text{fw}}_{h,s,t'} x^1_h - u^{\text{fw}}_{s,t'}\Big) - \sum_{t'\leq t} \sum_{h\in\bar{\Omega}^2} \gamma^{\text{fw}}_{h,s,t'} x^2_h \geq 0 \qquad \forall s \in S, t \in \mathbb{T}^{2,\text{start}}_s \tag{3f}$$

$$\sum_{t'\geq t+p_s} \Big(\sum_{h\in\bar{\Omega}^1} \gamma^{\text{rv}}_{h,s,t'} x^1_h - u^{\text{rv}}_{s,t'}\Big) - \sum_{t'\geq t} \sum_{h\in\bar{\Omega}^2} \gamma^{\text{rv}}_{h,s,t'} x^2_h \geq 0 \qquad \forall s \in S, t \in \mathbb{T}^{2,\text{end}}_s \tag{3g}$$

$$\begin{aligned}&\sum_{t'\leq t} \Big(\sum_{h\in\bar{\Omega}^1} \gamma^{\text{fw}}_{h,s,t'} x^1_h - u^{\text{fw}}_{s,t'}\Big) - \sum_{t'\leq t} \sum_{h\in\bar{\Omega}^2} \gamma^{\text{fw}}_{h,s,t'} x^2_h \\ &+ \sum_{t'>t} \Big(\sum_{h\in\bar{\Omega}^1} \gamma^{\text{rv}}_{h,s,t'} x^1_h - u^{\text{rv}}_{s,t'}\Big) - \sum_{t'>t} \sum_{h\in\bar{\Omega}^2} \gamma^{\text{rv}}_{h,s,t'} x^2_h \leq C^{sat}_s \qquad \forall s \in S, t \in \mathbb{T}^{1,\text{visit}}_s\end{aligned} \tag{3h}$$

$$u^{\text{fw}}_{s,t} \leq \sum_{h \in \bar{\Omega}^1} \gamma^{\text{fw}}_{h,s,t} x^1_h \qquad \forall s \in S, t \in \mathbb{T}^{1,\text{visit}}_s \qquad (3\text{i})$$

$$u^{\text{rv}}_{s,t} \leq \sum_{h \in \bar{\Omega}^1} \gamma^{\text{rv}}_{h,s,t} x^1_h \qquad \forall s \in S, t \in \mathbb{T}^{1,\text{visit}}_s \qquad (3\text{j})$$

$$\sum_{t \in \mathbb{T}} \sum_{h \in \Omega^{2+}_{s,t} \cup \Omega^{2-}_{s,t}} x^2_h \leq M \cdot \zeta_s \qquad \forall s \in S \qquad (3\text{k})$$

$$x^1_h \in \mathbb{N}_0 \qquad \forall h \in \bar{\Omega}^1 \qquad (3\text{l})$$

$$x^2_h \in \{0,1\} \qquad \forall h \in \bar{\Omega}^2 \qquad (3\text{m})$$

$$u^{\text{fw}}_{s,t}, u^{\text{rv}}_{s,t} \geq 0 \qquad \forall s \in S, t \in \mathbb{T}^{1,\text{visit}}_s \qquad (3\text{n})$$

$$\sigma_i \geq 0 \qquad \forall i \in N \qquad (3\text{o})$$

$$\zeta_s \in \{0,1\} \qquad \forall s \in S \qquad (3\text{p})$$

The objective (3a) adds the new penalty variables for unserved customer with big-M coefficients to the routing costs. Constraints (3b) state that each customer $i \in N$ is either served by a second-echelon trip or the associated penalty variable σ_i is set to 1. The fleet-size constraints (3c)–(3e) are identical to those in formulation (1) using sets $\bar{\Omega}^1$ and $\bar{\Omega}^2$. Constraints (3f)–(3h) are the flow-conservation and satellite-capacity constraints now subtracting the new u-variables to manage surplus quantities. Constraints (3i) and (3j) limit the variables u to the quantity dropped off and collected at each satellite $s \in S$ at time t. In addition, constraints (3k) allow trips to visit a satellite only if the satellite is utilized according to the new ζ-variables. Finally, the variable domains are given by (3l)–(3p).

A.4 Detailed results for the 2E-MTVRPTW instances

Instance	Cost
r101	872.37
r102	800.18
r103	703.95
r104	666.01
r105	801.67
r106	714.36
r107	673.67
r108	652.33
r109	701.08
r110	695.84
r111	677.73
r112	648.10
r201	717.33
r202	661.68
r203	641.77
r205	648.51
r206	659.74
r207	642.39
r208	574.47
r210	651.19

Instance	Cost
c101	471.03
c102	459.75
c103	443.16
c104	435.11
c105	471.03
c106	471.03
c107	465.75
c108	461.93
c109	440.13
c201	455.81
c202	445.23
c203	441.62
c204	438.10
c205	453.27
c206	453.27
c207	449.68
c208	445.70

Instance	Cost
rc101	802.84
rc102	757.23
rc103	743.34
rc104	709.95
rc105	828.21
rc106	759.81
rc107	718.17
rc108	700.96
rc201	619.93
rc202	592.38
rc203	607.06
rc204	595.39
rc205	564.54
rc206	579.80
rc207	576.01
rc208	524.35

Table 10: Results of the matheuristic *ITSH* on the 2E-MTVRPTW instances with 25 customers

Instance	Cost
r101	1 316.38
r102	1 185.01
r103	1 086.47
r104	963.87
r105	1 198.87
r106	1 080.09
r107	1 033.93
r108	937.21
r109	1 065.14
r110	1 036.03
r111	993.24
r112	1 002.22
r201	1 064.08

Instance	Cost
c101	1 008.91
c102	978.24
c103	947.95
c104	916.21
c105	995.86
c106	1 011.95
c107	975.37
c108	968.11
c109	934.44
c201	744.20
c202	740.53
c203	759.51
c204	706.63
c205	741.87
c206	741.32
c207	754.60
c208	727.20

Instance	Cost
rc101	1 505.31
rc102	1 429.51
rc103	1 383.80
rc104	1 265.18
rc105	1 487.50
rc106	1 453.06
rc107	1 431.03
rc108	1 315.12
rc201	968.99
rc202	916.46
rc203	866.15
rc205	934.23
rc206	911.71
rc207	890.79

Table 11: Results of the matheuristic *ITSH* on the 2E-MTVRPTW instances with 50 customers

Instance	Cost
r101	2 033.36
r102	1 863.84
r103	1 604.68
r104	1 428.00
r105	1 792.73
r106	1 707.32
r107	1 537.08
r110	1 608.41
r111	1 522.28
r112	1 539.07

Instance	Cost
c101	1 480.58
c102	1 434.96
c103	1 443.70
c104	1 309.81
c105	1 431.34
c106	1 455.36
c107	1 403.53
c108	1 393.20
c201	985.71
c202	1 007.61
c205	946.99
c206	944.77

Instance	Cost
rc101	2 019.64
rc102	1 915.65
rc103	1 790.81
rc105	1 959.16
rc106	1 879.94
rc107	1 795.84
rc108	1 731.54

Table 12: Results of the matheuristic *ITSH* on the 2E-MTVRPTW instances with 75 customers

Instance	Cost
r101	2 302.51
r102	2 131.33
r105	2 064.17

Instance	Cost
c101	2 007.48
c102	1 933.68
c105	1 911.79
c106	1 920.90
c107	1 864.91
c108	1 867.74
c109	1 939.84
c201	1 283.63

Instance	Cost
rc101	2 548.11
rc102	2 403.70
rc105	2 549.57
rc106	2 375.75
rc106	2 375.75

Table 13: Results of the matheuristic *ITSH* on the 2E-MTVRPTW instances with 100 customers

A.5 Detailed results for the 2E-MTVRP-CSRF instances

	Satellite capacity C^{sat}					
Instance	20	25	35	50	70	∞
A-50-1	1 876	1 759	1 642	1 598	1 598	1 598
A-50-2	–	1 956	1 907	1 847	1 847	1 842
A-50-3	1 976	1 870	1 798	1 753	1 753	1 753
A-50-4	–	2 042	1 914	1 862	1 862	1 862
A-50-5	–	2 260	2 116	2 081	2 066	2 066
A-50-6	1 993	1 988	1 973	1 973	1 973	1 973
A-50-7	1 988	1 905	1 819	1 818	1 798	1 738
A-50-8	–	1 854	1 854	1 791	1 727	1 727
A-50-9	–	2 170	2 030	1 986	1 936	1 936
A-50-10	1 931	1 846	1 776	1 714	1 700	1 700
B-50-1	1 960	1 881	1 835	1 835	1 802	1 802
B-50-2	1 857	1 857	1 787	1 749	1 749	1 749
B-50-3	1 888	1 733	1 713	1 713	1 713	1 699
B-50-4	1 765	1 647	1 641	1 566	1 566	1 566
B-50-5	1 768	1 721	1 713	1 712	1 660	1 641
B-50-6	1 762	1 684	1 664	1 664	1 616	1 616
B-50-7	1 625	1 573	1 533	1 533	1 521	1 521
B-50-8	1 875	1 822	1 752	1 752	1 752	1 746
B-50-9	1 760	1 738	1 710	1 710	1 710	1 710
B-50-10	1 661	1 601	1 585	1 566	1 566	1 547
C-50-1	1 746	1 666	1 666	1 666	1 666	1 666
C-50-2	1 772	1 762	1 733	1 677	1 631	1 631
C-50-3	1 637	1 571	1 546	1 546	1 513	1 513
C-50-4	1 845	1 732	1 654	1 654	1 654	1 654
C-50-5	1 800	1 800	1 711	1 689	1 689	1 624
C-50-6	1 783	1 781	1 703	1 703	1 703	1 700
C-50-7	1 553	1 449	1 449	1 449	1 449	1 449
C-50-8	1 815	1 770	1 692	1 636	1 636	1 636
C-50-9	1 578	1 578	1 578	1 542	1 542	1 534
C-50-10	1 555	1 522	1 487	1 487	1 487	1 487

Table 14: Results on the 2E-MTVRP-CSRF instances with 50 customers

	Satellite capacity C^{sat}					
Instance	20	25	35	50	70	∞
A-50-1	–	3 384	3 110	2 992	2 821	2 782
A-50-2	–	3 756	3 525	3 273	3 180	3 180
A-50-3	–	3 167	2 840	2 744	2 671	2 660
A-50-4	3 921	3 685	3 319	3 086	3 086	3 086
A-50-5	3 789	3 338	2 999	2 822	2 783	2 783
A-50-6	–	-	3 380	3 187	3 093	3 036
A-50-7	–	3 588	3 214	3 027	2 956	2 935
A-50-8	3 835	3 354	3 128	2 870	2 813	2 813
A-50-9	3 682	3 380	3 196	2 921	2 869	2 848
A-50-10	3 279	2 967	2 727	2 637	2 566	2 566
B-50-1	3 313	3 157	2 932	2 769	2 669	2 669
B-50-2	3 490	3 358	3 112	3 041	2 865	2 865
B-50-3	3 268	3 096	2 934	2 831	2 767	2 767
B-50-4	3 064	2 952	2 816	2 605	2 605	2 605
B-50-5	3 164	3 124	2 923	2 846	2 728	2 728
B-50-6	3 073	2 837	2 547	2 547	2 516	2 516
B-50-7	3 660	3 464	3 235	3 083	2 961	2 933
B-50-8	2 541	2 526	2 412	2 391	2 391	2 373
B-50-9	3 344	3 166	2 933	2 745	2 667	2 667
B-50-10	2 702	2 538	2 423	2 423	2 406	2 406
C-50-1	2 948	2 831	2 631	2 531	2 524	2 514
C-50-2	2 686	2 546	2 483	2 410	2 363	2 326
C-50-3	2 811	2 565	2 485	2 485	2 473	2 456
C-50-4	2 704	2 508	2 429	2 408	2 370	2 370
C-50-5	2 476	2 476	2 422	2 367	2 367	2 367
C-50-6	2 554	2 513	2 462	2 354	2 354	2 354
C-50-7	2 449	2 449	2 340	2 340	2 279	2 279
C-50-8	2 732	2 707	2 629	2 479	2 479	2 477
C-50-9	2 383	2 326	2 287	2 265	2 265	2 265
C-50-10	2 895	2 799	2 659	2 593	2 593	2 593

Table 15: Results on the 2E-MTVRP-CSRF instances with 100 customers