Dynamic Discretization Discovery for Solving the Continuous Time Inventory Routing Problem with Out-and-Back Routes

Felipe Lagos[1] Natashia Boland[2] Martin Savelsbergh[2]

[1]Faculty of Engineering and Sciences
Universidad Adolfo Ibáñez
Santiago, Chile 7941169

[2]H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

Abstract

In time dependent models, the objective is to find the optimal times (continuous) at which activities occur and resources are utilized. These models arise whenever a schedule of activities needs to be constructed. A common approach consists of discretizing the planning time and then restricting the decisions to those time points. However, this approach leads to very large formulations that are intractable in practice. In this work, we study the Continuous Time Inventory Routing Problem with Out-and-Back Routes (CIRP-OB). In this problem, a company manages the inventory of its customers, resupplying a single product from a single facility during a finite time horizon. The product is consumed at a constant rate (product per unit of time) by each customer. The customers have local storage capacity. The goal is to find the minimum cost delivery plan with out-and-back routes only that ensures that none of the customers run out of product during the planning period. We study the Dynamic Discovery Discretization algorithm (DDD) to solve the CIRP-OB by using partially constructed time-expanded networks. This method iteratively discovers time points needed in the network to find optimal continuous time solutions. We test this method by using randomly generated instances in which we find provable optimal solutions.

Funding: This material is based upon work supported by the National Science Foundation [Grant 1662848]

1 Introduction

There are important classes of integer linear programming models which cannot be solved satisfactorily in practice. One of these classes is characterized by time dependent decisions that have to be made about the times at which activities occur and/or resources are utilized. These models are called *time-dependent models* (Boland and Savelsbergh 2019), and they arise whenever a schedule of activities needs to be constructed.

In practice, two types of formulations are used when solving time dependent models: compact or extended formulations. Compact formulations have a polynomial number of variables and constraints and do not depend on a discretization of time. These formulations usually require "big-M" constraints and have weak linear programming (LP) relaxations. As a result, only small instances can be solved with current integer programming technology. Extended formulations involve a discretization of time and have (integer) variables associated with each time point in the discretization. Extended formulations have much stronger LP relaxations, but (tend to) have a huge number of variables. Discretization of time introduces an approximation (as not all possible times are explicitly represented). A new iterative algorithm has been proposed in Boland et al. (2017) which uses extended formulations but which guarantees that an optimal continuous-time solution is found. The proposed algorithm, the Dynamic Discretization Discovery (DDD) algorithm, is used to solve instances of a continuous-time service network design problem.

Several continuous time problems have been studied using the DDD (Boland and Savelsbergh 2019), among them stands out the Inventory Routing Problem (IRP). The IRP integrates inventory management, vehicle routing, and delivery scheduling decisions. The IRP is commonly found in the context of Vendor Managed Inventory (VMI), in which a supplier makes the replenishment decisions for products delivered to its customers. An industry in which this problem frequently arises is the liquid gas industry, with companies like Air Liquide (`www.airliquide.com`), Air Products (`www.airproducts.com`), and Praxair (`www.praxair.com`). These companies produce liquefied gases, e.g., liquid oxygen, liquid nitrogen, or liquid argon, and they guarantee minimum product availability at any time for their customers by installing tanks on their customers' premises and refilling the tanks whenever is needed during a planning horizon. In practice, the company contracts typically specify that the customers own/purchase the product upon delivery, which means that the companies do not have to consider product holding costs at the customer's location. Customers use (consume) product at a certain rate, often 24 hours per day, and the companies continuously monitor product usage and tank inventory levels in order to guarantee product availability. The rate can differ at different times of the planning horizon, so the amount of product that can be delivered to the tank changes at the same rate. As consequence, some customers might require multiple deliveries per day, while others require as few as one or two deliveries per week. The companies seek to find cost-effective delivery schedules that meet their service commitments (i.e., the guaranteed minimum product availabilities). As is argued in Lagos et al. (2020), the use of a continuous time variant of the IRP is most appropriate in these settings because it provides the most accurate representation of the system. This challenge, that the liquid gas industry faces, motivates the IRP variant studied

in this work, the Continuous Time Inventory Routing Problem (CIRP). As we find in this study, modeling inventory in continuous time and accurately modeling vehicle travel times make for a particularly challenging IRP.

In a direct delivery or out-and-back routing strategy, vehicles depart from the depot, visit only one customer, and then return. It is natural that this strategy is considered first for solving the IRP, because it is easy-to-implement and is frequently used in industrial distribution systems (Li et al. 2010). In certain settings, for example when storage capacity at (most) customers is large relative to the vehicle capacity, a direct delivery strategy is highly effective and there is no need to consider more complex distribution strategies. However, the primary reason for restricting ourselves to the direct delivery strategy is that it reduces the computational complexity (although by no means resulting in a simply, easy-to-solve optimization problem). Given the complexity of the ideas we study in this work, we consider this simpler setting for the continuous time IRP. We call this variant the CIRP with Out-and-Back routes only (CIRP-OB), which we propose as a starting point for further work.

Both the IRP and the time-dependent models literature are very active. The IRP is an important problem in several industries (besides liquid gas industry), such as maritime logistics, transportation of perishable items and chemical components. For these applications, several variants have been considered, including differences with respect to: time horizon (finite, infinite); distribution structure (one-to-one, one-to-many, many-to-many); routing (multiple, out-and-back routes); fleet composition (homogeneous, heterogeneous); demand (stochastic, deterministic); etc (Coelho et al. 2014). Literature reviews are provided in Bertazzi et al. (2008) and Coelho et al. (2014). The DDD is also an active area of research. The DDD was introduced in Boland et al. (2017), and during these few years, it has been applied to several problems, including, the Service Network Design Problem, the Traveling Salesman Problem with Time Windows and the Shortest Path Problem. In Boland et al. (2019), the authors present perspectives on various aspect of time-dependent models and also, introduce the primary concepts underlying DDD.

In this paper, we fully study the DDD for the CIRP with out-and-back routes (CIRP-OB). Our contributions are both theoretical and algorithmic. In particular, we

- successfully develop and implement the DDD algorithm; adapt a mixed integer program to yield lower bounds on the optimal CIRP-OB value, develop a mixed integer program to repair solutions obtained from the lower bound model, and put forward routines to improve the time-expanded network when the lower bound solution cannot be converted;

- show that the DDD algorithm is an exact algorithm for the CIRP-OB, proving that for a feasible CIRP-OB instance, the algorithm finishes with an optimal solution in a finite number of iterations;

- derive conditions on the waiting times and arrival visit times that a subset of CIPR-OB optimal solutions hold (so we restrict our search to that subset). Then we show how to incorporate these conditions into the lower bound formulation (strengthening the model); and,

- generate a new set of instances for the CIRP-OB to carry out a computational study, and show that our DDD algorithm is able to produce provable optimal solutions for instances with up to 30 customers.

The remainder of the paper is organized as follows. In Section 2, we present a brief literature review. In Section 3, we formally introduce the Continuous Time Inventory Routing Problem with Out-and-Back routes only (CIRP-OB), and we show that this problem is *strongly NP-Hard*. In Section 4, we provide a mixed integer programming formulation for the CIRP-OB over a time discretization. We present some conditions that a subset of CIRP-OB optimal solutions hold in Section 5, while in Section 6, we show how the mixed integer programming formulation can be modified to produce a lower bound on the cost of an optimal delivery plan. In Section 7, we outline two approaches for constructing feasible delivery plans. In Section 8, we describe the DDD algorithm and show that it is an exact algorithm for the CIRP-OB, and in Section 9, we present and discuss the results of an extensive computational study. Finally, in Section 10, we discuss relevant aspects of the algorithm and models presented and outline future work.

2 Literature Review

The variant of interest in this paper was introduced in the seminal paper by Bell et al. (1983) and two critical characteristics made this variant different to the majority of the variants of the IRP in the literature: only transportation costs are considered in the objective function to minimize and that the system evolves in continuous time. That is, the amount of product that can be delivered to a customer at a particular point in time depends on the storage capacity and inventory at that point in time. The inventory level at a given time depends on the initial inventory, the product usage rate, and the time elapsed since the start of the planning period, and on the amount of product delivered since the start of the planning period. In this time dependent system, delivery times have to be scheduled carefully and vehicle travel times have to be accounted for accurately. In contrast, in the majority of the variants of the IRP considered in the literature, the inventory levels, deliveries and route executions happen within a period instantaneously; the planning horizon is partitioned into periods and the system evolves restricted to those periods. In addition, delivery routes take place at the start of the period and product consumption occurs at the end of the period.

Solving instances of any variant of the IRP is challenging (Coelho and Laporte 2013). Integer programming techniques have been used for the "period" variants, e.g., branch-and-cut (Coelho and Laporte 2014, Avella et al. 2017, Archetti et al. 2007) and branch-and-cut-and-price (Coelho and Laporte 2014, Desaulniers et al. 2015). Archetti et al. (2007) also propose some valid inequalities to strengthen the model. The most advanced and successful of these can now solve instances (to optimality) with up to 50 customers, no more than 6 time periods and up to 5 vehicles.

The "continuous time" variant of the IRP has attracted attention in the past, e.g., Campbell et al. (1998), Campbell et al. (2002), and Campbell and Savelsbergh (2004a,b), and, was considered interesting and challenging enough to form the ROADEF/EURO 2016 Challenge

(for more information, see `www.roadef.org/challenge/2016/en/sujet.php`) with real-life data provided by Air Liquide. For this variant, lower bounding techniques have been developed in Song and Savelsbergh (2007).

The IRP with out-and-back routes only (or direct deliveries) has been studied in the past (Kleywegt et al. 2002, Bertazzi 2008). Studies show direct deliveries are effective compared to any other feasible distribution strategy in the long-run (Gallego and Simchi-Levi 1994, Li et al. 2010, Gallego and Simchi-Levi 1990). In Gallego and Simchi-Levi (1990), the authors derived an explicit formula for evaluating how effective the direct delivery strategy is regarding the long-run average cost, concluding that it is at least 94% effective whenever the customer capacities exceed 71% of the vehicle capacity. In Li et al. (2010), the effectiveness of the direct deliveries can be represented by a function of some system parameters. Under some conditions on the usage rate and vehicle capacity, direct deliveries is an optimal distribution strategy, specially when the usage rate is high with respect to the vehicle capacity (e.g., when several deliveries to the customer are needed during a day).

Boland et al. (2017) introduced a Dynamic Discretization Discovery (DDD) algorithm for solving the continuous time Service Network Design Problem (SNDP), which uses extended integer programming formulations. The integer programs are constructed as a function of a subset of times, with variables indexed by times in the subset. These IPs are carefully designed to be tractable in practice and to yield a lower bound on the optimal continuous-time value. Once the *right* (very small) subset of times is discovered, the resulting integer programming model yields the continuous-time optimal value. The key to the approach is that it discovers exactly which times are needed to obtain an optimal, continuous-time solution, in an efficient way, by solving a sequence of (small) integer programs. The integer programs are constructed as a function of a subset of times, with variables indexed by times in the subset. These IPs are carefully designed to be tractable in practice and to yield a lower bound on the optimal continuous-time value.

In addition to the SNDP (Boland et al. 2017, 2019), there are other problems for which this methodology has been successfully applied, such as *Traveling Salesman Problem with Time Windows* (TSP-TW) (Vu et al. (2020)) and *Minimum Duration Shortest Path Problem* (MD-SSP). The opportunities for further research on time-dependent models are many (Boland and Savelsbergh 2019), so a growing literature, with applications to new problems and algorithmic developments, is expected.

In Lagos et al. (2020) we introduce the Continuous Time IRP (CIRP), the IRP variant in that optimal continuous visiting times are needed. Besides providing a mathematical formulation of the problem, we present properties of the problem and a lower bound value for the optimal cost. Among the properties, we show that the CIRP, when feasible, always has a rational optimal solution, i.e., a solution with rational visiting times and deliveries. In an extensive computational study, we provide proven optimal solutions for the CIRP using a simple version of the DDD algorithm. This algorithm is combined with sophisticated MIP models that provide a lower bound on the optimal solution value and a MIP that seeks to manipulate a set of delivery routes from this lower bound model to construct a feasible continuous-time solution. In this work, we extend the ideas presented in Lagos et al. (2020)

and develop and implement a full version of the DDD algorithm.

3 The Continuous Time Inventory Routing Problem with Out-and-Back Routes

We consider a vendor managed resupply environment in which a company manages the inventory of its customers, resupplying them from a single facility (depot) during a planning horizon H. Each customer i, in the set $N = \{1, \ldots, n\}$ of customers, has local storage capacity C_i, uses product at a constant rate u_i and has initial inventory I_i^0 at the start of the planning period (it is assumed to be equal or less than C_i). The set of customers and the depot (represented by $i = 0$) corresponds to $N_0 = N \cup \{0\}$. The company deploys a fleet of m homogeneous vehicles, each with capacity Q, to deliver product to its customers. Vehicles are assumed to be at the company's facility at the start of the planning period and have to return to the company's facility at the end of the planning period, but they can make multiple trips during this planning time. We assume that the loading of a vehicle at the depot is instantaneous.

We also assume that the company does not incur any holding cost for product, either at the company facility or at any of the customer sites. The company facility always has sufficient product to supply customers; it does not have either production or storage capacity constraints.

We consider an out-and-back routes only setting: a vehicle route starts at the depot visits a single customer and returns to the depot. Travel times τ_i and travel costs c_i, between the depot and a location $i \in N$, are assumed to be symmetric and strictly positive.

Vehicles are allowed to wait at customer location and, while they are there, to make multiple deliveries. This may be beneficial as it allows delivery of more that the customer's remaining capacity at the vehicle's arrival time, without an extra transportation cost. In practice, a customer may have sufficient space for several vehicles to wait at their premises, but usually at most one vehicle can deliver product at a time. As mentioned in Lagos et al. (2020), the situation in which multiple vehicles wait at a customer location can be modeled by the use of two locations for each customer (one for parking and one for making a delivery at the customer), but given the complexity of the ideas we wish to discuss in this paper, we make the simplifying assumption that all locations have the single-vehicle constraint. Thus, we assume that it is *not* possible for multiple vehicles to visit the same customer at the same time. Also, as assumed in Lagos et al. (2020), there is no cost for waiting.

We also assume that the product delivery at a customer site is instantaneous. If the delivery time is constant, it can be easily incorporated as part of the travel time between the depot and a customer. If the delivery time is quantity-dependent, incorporating delivery times becomes much more complicated (see, e.g., Campbell and Savelsbergh (2004b)). Studying this variant is left for future research.

The goal is to find a minimum cost delivery plan that ensures that none of the customers runs out of product during the planning period. A delivery plan specifies a set of vehicle

itineraries (routes), each of which consists of a sequence of out-and-back routes that start and end at the depot within the time horizon, to be performed by a single vehicle. Each route specifies a departure time from the facility, a visit to a customer, quantities delivered to the customer during the visit and the times of those deliveries, and a departure time from the customer location. We refer to this problem as the Continuous Time Inventory Routing Problem with Out-and-Back routes (CIRP-OB). Next, even though we assume only out-and-back routes, we show that the problem is still *strongly NP-Hard*.

Proposition 1. *The CIRP-OB is* strongly NP-Hard.

Proof. We show that CIRP-OB can be reduced to 3-PARTITION. Details in Appendix A. □

4 Exact Time Indexed Formulation

In Lagos et al. (2020) the authors prove that a rational optimal solution always exists for a feasible Continuous Time IRP (CIRP) instance. The CIRP-OB is a special case of the CIRP, thus an optimal solution whose deliveries and decision times are rational is guaranteed as long as a feasible solution exists. Then, there exists a time discretization sufficiently fine in which a time indexed formulation finds an optimal solution. In this section, we assume such time discretization with the property just mentioned is given and we present a time indexed formulation on it.

Let $\mathcal{T}_i = \{0, 1, \ldots, K_i\}$ be an index set and let $\{t_i^k\}_{k\in\mathcal{T}_i}$ with $t_i^0 = 0$, $t_i^{k-1} < t_i^k$ for $k = 1, \ldots, K_i$, and $t_i^{K_i} = H$ be the set of time points in the time discretization of location $i \in N_0$. Note that these time point sets can represent any time discretization, i.e., the difference between two consecutive time points, $t_i^{k+1} - t_i^k$ with $i \in N_0$ and $k \in \mathcal{T}_i \setminus \{0\}$, can be any rational number. We consider a time-expanded network that preserves travel times, and so, the following conditions hold to all time points:

- for all $i \in N$ and $k \in \mathcal{T}_0$ such that $t_0^k + \tau_i \leq H$, then $t_0^k + \tau_i \in \{t_i^\ell\}_{\ell\in\mathcal{T}_i}$;
- for all $i \in N$ and $k \in \mathcal{T}_0$ such that $t_0^k - \tau_i \geq 0$, then $t_0^k - \tau_i \in \{t_i^\ell\}_{\ell\in\mathcal{T}_i}$;
- for all $i \in N$ and $k \in \mathcal{T}_i$ such that $t_i^k + \tau_i \leq H$, then $t_i^k + \tau_i \in \{t_0^\ell\}_{\ell\in\mathcal{T}_0}$;
- for all $i \in N$ and $k \in \mathcal{T}_i$ such that $t_i^k - \tau_i \geq 0$, then $t_i^k - \tau_i \in \{t_0^\ell\}_{\ell\in\mathcal{T}_0}$.

Locations in N_0 and times $\{t_i^k\}_{i\in N_0, k\in\mathcal{T}_i}$ induce a time-expanded network. This network consists of the nodes set $\mathcal{N}^{\mathcal{T}} = \{(i, t_i^k) : i \in N_0, k \in \mathcal{T}_i\}$ and the timed arcs set,

$$\mathcal{A}^{\mathcal{T}} = \left\{ a_{ij}^{k\ell} := \big((i, t_i^k), (j, t_j^\ell)\big) \in \mathcal{N}^{\mathcal{T}} \times \mathcal{N}^{\mathcal{T}} : \begin{array}{l} \text{if } i = 0 \text{ and } j \in N \text{ then } t_0^k + \tau_j = t_j^\ell \\ \text{if } i \in N \text{ and } j = 0 \text{ then } t_i^k + \tau_i = t_0^\ell \\ \text{if } i = j \text{ and } k < K_i \text{ then } t_j^\ell = t_i^{k+1} \end{array} \right\}.$$

Let $\delta^+(i,t_i^k) = \{(j,\ell) : \exists \ell, a_{ij}^{k\ell} \in \mathcal{A}^{\mathcal{T}}\}$ be the set of locations and time indexes that can be reached from i at time t_i^k. Similarly, we define $\delta^-(i,t_i^k) = \{(j,\ell) : \exists \ell, a_{ji}^{\ell k} \in \mathcal{A}^{\mathcal{T}}\}$, as the set of locations and time indexes that reach the node $(i,t_i^k) \in \mathcal{N}^{\mathcal{T}}$. To simplify notation, we define $\bar{u}_i^k = u_i(t_i^k - t_i^{k-1})$ as the product consumed by a customer $i \in N$ during the time interval $[t_i^{k-1}, t_i^k]$, for $(i,t_i^k) \in \mathcal{N}^{\mathcal{T}}$ and $k \geq 1$.

We consider binary variables x_{ij}^k representing whether a vehicle travels from i to j at time t_i^k, $x_{ij}^k = 1$, or not, $x_{ij}^k = 0$, for all arcs $a_{ij}^{k\ell} \in \mathcal{A}^{\mathcal{T}}$. We consider continuous variables w_{ij}^k representing the amount of product that is transported from i to j at time t_i^k. Furthermore, the binary variable x_{ii}^k represents whether a vehicle waits at i from time t_i^k until (at least) time t_i^{k+1}, or not. If $x_{ii}^k = 1$, then the product "transported" from i to i at time t_i^k (to time t_i^{k+1}), i.e., w_{ii}^k, can be positive as there is a vehicle at i.

The variable y_i^k represents the product quantity delivered at i at time t_i^k and z_i^k represents the inventory level at i at time t_i^k **after** a delivery takes place (if any). The time indexed formulation for the CIRP-OB is the following.

$$
\begin{aligned}
\min \quad & \sum_{i \in N} \sum_{k \in \mathcal{T}_0} 2c_i x_{0i}^k, & & & \\
\text{s.t.} \quad & x_{i0}^k + x_{ii}^k = x_{ii}^{k-1} + x_{0i}^\ell, & & i \in N, k \in \mathcal{T}_i, (0,\ell) \in \delta^-(i,t_i^k), & \text{(1a)} \\
& x_{00}^k + \sum_{i \in N} x_{0i}^k = x_{00}^{k-1} + \sum_{(i,\ell) \in \delta^-(0,t_0^k)} x_{i0}^\ell, & & k \in \mathcal{T}_0 \setminus \{0, K_0\}, & \text{(1b)} \\
& x_{00}^0 + \sum_{i \in N} x_{0i}^0 = m, & & & \text{(1c)} \\
& w_{0i}^\ell + w_{ii}^{k-1} - w_{ii}^k = y_i^k, & & i \in N, k \in \mathcal{T}_i, (0,\ell) \in \delta^-(i,t_i^k), & \text{(1d)} \\
& x_{ii}^k + x_{i0}^k \leq 1, & & i \in N, k \in \mathcal{T}_i, & \text{(1e)} \\
& 0 \leq w_{ij}^k \leq Q x_{ij}^k, & & a_{ij}^{k\ell} \in \mathcal{A}^{\mathcal{T}}, & \text{(1f)} \\
& z_i^k = z_i^{k-1} + y_i^k - \bar{u}_i^k, & & i \in N, k \in \mathcal{T}_i \setminus \{0\}, & \text{(1g)} \\
& z_i^0 = I_i^0 + y_i^0, & & i \in N, & \text{(1h)} \\
& \bar{u}_i^{k+1} \leq z_i^k \leq C_i, & & i \in N, k \in \mathcal{T}_i, & \text{(1i)} \\
& 0 \leq y_i^k, & & i \in N, k \in \mathcal{T}_i, & \\
& x_{ij}^k \in \{0,1\}, & & a_{ij}^{k\ell} \in \mathcal{A}^{\mathcal{T}}. &
\end{aligned}
$$

Constraints (1a), (1b) and (1c) ensure vehicle flow balance and enforce that all m vehicles return to the depot at the end of the planning horizon. Constraints (1d) impose product flow balance and ensure that the product arriving at a customer is either delivered at that customer or remains on the vehicle. Note that no product can come back to the depot. Constraints (1e) together with the requirement that each x_{ij}^t variable is binary ensure that at most one vehicle can be visiting a customer at any one time. Constraints (1f) link the product flows to the vehicle flows. Constraints (1g) and (1h) model product usage at a customer and inventory balance. Constraints (1i) ensure that inventory at a customer is

sufficient, after each delivery, to meet the customer demand in the coming period and never exceeds the local storage capacity.

A lower bound value to the linear relaxation of formulation (1) can be found using the parameters of the problem. This value corresponds to the cost of the minimum number of vehicles required (possible fractional) to deliver the product that customers need to cover their usage during the planning horizon.

Proposition 2. *A lower bound for the formulation* (1) *linear relaxation optimal value is given by*

$$\sum_{i \in N} 2c_i \left(\frac{Hu_i - I_i^0}{Q} \right). \tag{2}$$

Proof. Note that the inventory balance constraints imply the following condition on the total delivery for any $i \in N$,

$$z_i^{K_i} = I_i^0 + \sum_{k \in \mathcal{T}_i} y_i^k - Hu_i \geq 0,$$

$$\sum_{k \in \mathcal{T}_i} y_i^k \geq Hu_i - I_i^0.$$

Also, note that summing up the product balance constraints (1d) leads to,

$$\sum_{\ell \in \mathcal{T}_0} w_{0i}^\ell = \sum_{k \in \mathcal{T}_i} y_i^k,$$

and since $w_{ij}^k \leq Q x_{ij}^k$, we have,

$$Hu_i - I_i^0 \leq \sum_{k \in \mathcal{T}_i} y_i^k = \sum_{\ell \in \mathcal{T}_0} w_{0i}^\ell \leq Q \sum_{\ell \in \mathcal{T}_0} x_{0i}^\ell.$$

Multiplying by $2c_i$ and summing up over all $i \in N$, we get that any feasible solution to formulation (1) is greater than or equal to (2). □

In Proposition 3, we show that the lower bound in (2) is tight on the linear relaxation optimal value when the number of vehicles is large.

Proposition 3. *Consider a feasible CIRP-OB instance and* $m = \sum_{i \in N} \lceil (C_i - I_i^0 + 2\tau_i u_i)/Q \rceil$. *Also, assume that the time points* $H - C_i/u_i$ *and* τ_i *are in* $\{t_i^k\}_k$, *for all* $i \in N$. *The optimal solution value to formulation* (1) *linear relaxation is equal to* (2), $\sum_{i \in N} 2c_i \left(\frac{Hu_i - I_i^0}{Q} \right)$.

Sketch of Proof. We construct a simple solution, in which a fraction of a vehicle delivers what is consumed in a time interval by the customers. Then, we show that the solution is feasible. Full details are given in Appendix A. □

In practice, for any feasible instance, the previous proposition holds and the value of the linear relaxation of (1) is (2). In consequence, the linear relaxation is weak: when the integrality condition on x variables is relaxed, the optimal solution might contain several "fractional" vehicles that deliver only what is consumed in a time interval, $\bar{u}_i^k$, $i \in N$, $k \in \mathcal{T}_i$.

This suggests that strengthening the formulation, by exploiting problem structure, will be critical in the development of an effective solution approach. In the next section we present some observations that can be used for strengthening the formulation (1).

5 Optimality Preserving Conditions (OPC)

In this section we present several conditions that can be used to limit the search for an optimal solution for the CIPR-OB. We call these conditions *optimality preserving conditions* (OPC), because there always exists an optimal solution that satisfies *all* these conditions. In order to derive these conditions, we first need to consider the following definitions.

Definition 1. ***Depot-time set of a vehicle****: For a CIRP-OB feasible solution, we define the depot-time set S_0^v of a vehicle indexed by $v \in \{1, \ldots, m\}$, as the union of time intervals in $[0, H]$ the vehicle v stays at the depot.*

Definition 2. ***Depot-time set of a solution****: For a CIRP-OB feasible solution, we define the depot-time set of the solution as the collection of depot-time sets of the vehicles, S_0^v, $v \in \{1, \ldots, m\}$.*

Definition 3. ***Maximal depot-time-set optimal solution (MDO)****: A MDO is an optimal solution to a CIRP-OB instance with depot time set $\bar{S}_0^v$, for all $v \in \{1, \ldots, m\}$, such that there is no other optimal solution with depot time set S_0^v, $v \in \{1, \ldots, m\}$, that satisfies the following conditions:*

- *for all $v \in \{1, \ldots, m\}$, $\bar{S}_0^v \subseteq S_0^v$;*
- *and for some vehicle $v' \in \{1, \ldots, m\}$, $\bar{S}_0^{v'} \subset S_0^{v'}$.*

That is, an MDO is an optimal solution in which the vehicles spend as much time at the depot as possible.

For the CIRP-OB several similar solutions might be optimal. Small perturbations to an optimal solution to the time a vehicle waits at a customer location; or to the delivered quantity; or to the time a vehicle stays at the depot; lead to different optimal solutions. We restrict the search for optimal solutions that are MDO only and that must satisfy all the OPC conditions. We use the following observation: if a solution does not satisfy any of the OPC conditions, then it is not an MDO, since there exists another optimal solution whose depot-time set is maximal.

All the proofs of the OPC propositions can be found in Appendix A.

The following OPC propositions provide conditions that an MDO solution holds with respect to vehicle waiting times at the customer location: in Proposition 4, the inventory

level at the customer restricts the delivered quantity after the waiting; in Proposition 5, the customer inventory level at the vehicle arrival time constraints the starting time of the waiting; and in Proposition 6, the duration of the waiting time is restricted.

Proposition 4. *In an MDO, if a vehicle waits at customer location $i \in N$ before making a delivery, then the inventory level immediately after the delivery is C_i.*

Proposition 5. *In an MDO, if a vehicle waits at customer location $i \in N$ before making a delivery, then the inventory level upon arrival is zero.*

Proposition 6. *In an MDO, if a vehicle waits at customer location $i \in N$ before making a delivery, then the waiting time is no longer than $\frac{\max\{Q-C_i,0\}}{u_i}$.*

In an MDO the visit times to customer are also restricted to specific time intervals, since the time vehicles are in the depot is maximal and no "useless" waiting at the customer location is allowed. The following OPC propositions indicate the time intervals for two classes of customers, customers whose inventory capacity is less than Q and customers whose capacity is greater than or equal to Q.

Let $N_\ell \subseteq N$ be the set of customers with $C_i < Q$, and let $N_g = N \setminus N_\ell$ be the set of customers with $C_i \geq Q$.

Proposition 7. *In an MDO, if a customer $i \in N_g$ has exactly $\eta_i = \left\lceil \frac{Hu_i - I_i^0}{Q} \right\rceil$ visits, then the visit arrival times s_k, $k = 1 \ldots, \eta_i$ must occur in the following intervals,*

$$s_k \in \left[\frac{Hu_i - C_i - (\eta_i - k)Q}{u_i}, \frac{I_i^0 + (k-1)Q}{u_i} \right], \qquad k = 1, \ldots, \eta_i. \tag{3}$$

Proposition 8. *In an MDO, if a customer $i \in N_\ell$ has exactly $\eta_i = \left\lceil \frac{Hu_i - I_i^0}{Q} \right\rceil$ visits, then the visit arrival times s_k, $k = 1 \ldots, \eta_i$ must occur in the following intervals,*

$$s_k \in \left[\frac{Hu_i - C_i + I_i^0 - (\eta_i - k + 1)Q}{u_i}, \frac{I_i^0 + (k-1)Q}{u_i} \right], \qquad k = 1, \ldots, \eta_i. \tag{4}$$

Proposition 9. *In an MDO, if a customer $i \in N_g$ has v visits, $v > \eta_i = \left\lceil \frac{Hu_i - I_i^0}{Q} \right\rceil$, then in at least one of the intervals,*

$$\left[\frac{kQ - (C_i - I_i^0)}{u_i}, \frac{I_i^0 + (k-1)Q}{u_i} \right], \qquad k = 1, \ldots, \eta_i, \tag{5}$$

there are no arrivals.

The last OPC proposition restricts the delivered quantity at the arrival time.

Proposition 10. *In an MDO, the delivered quantity at the time the vehicle arrives to a customer $i \in N$ is equal to the minimum between $C_i - I_i$ and Q, where I_i is the inventory level at the arrival time and C_i is the customer capacity.*

6 Lower Bound Model (LBM)

A formulation that provides a lower bound on the value of an optimal solution to an instance of CIRP-OB can be obtained from formulation (1) by removing time points from the discretization, i.e., removing one or more time points from the sets $\mathcal{T}_i$ for $i \in N$, appropriately adjusting input parameters, and relaxing some of the constraints. However, the time expanded network must always satisfy $\{0, H\} \subseteq \{t_i^k\}_{k \in \mathcal{T}_i}$ for all $i \in N_0$. We consider the following modifications:

1. The travel time at time t_i^k, $i \in N_0$, $k \in \mathcal{T}_i$, to a location $j \in N_0$ is given according to the following expression,

$$\tau_{ij}^k = \begin{cases} \max_{\ell \in \mathcal{T}_j} \{t_j^\ell : t_j^\ell \leq t_0^k + \tau_j\} - t_0^k & i = 0, j \in N, \\ \max_{\ell \in \mathcal{T}_0} \{t_0^\ell : t_0^\ell \leq t_i^k + \tau_i\} - t_i^k & i \in N, j = 0, \\ t_i^{k+1} - t_i^k & i = j, k < K_i. \end{cases} \tag{6}$$

   The travel times are rounded down, which ensures that $\tau_{ij}^k \leq \tau_j$, for all $i, j \in N_0$, $k \in \mathcal{T}_i$. Note that, depending on the time discretizations, some travel times τ_{ij}^k might be non-positive. Also for two different departure time points $t_i^{k_1}$ and $t_i^{k_2}$, $k_1, k_2 \in \mathcal{T}_i$, with $k_1 < k_2$, the arrival time at $j \in N_0$ can be potentially the same, i.e., $t_i^{k_1} + \tau_{ij}^{k_1} = t_i^{k_2} + \tau_{ij}^{k_2}$.

2. The customer storage capacity is enlarged by adding the amount $u_i(t_i^{k+1} - t_i^k) = \bar{u}_i^{k+1}$, i.e., the product consumption during the time interval $[t_i^k, t_i^{k+1}]$, with $i \in N$, $k \in \mathcal{T}_i$. We modify the z variables upper bound in (1i) as follows,

$$\bar{u}_i^{k+1} \leq z_i^k \leq C_i + \bar{u}_i^{k+1}. \tag{7}$$

   Note that the new storage capacity bounds depend on the customer time discretization; different time interval lengths have different inventory capacities.

3. Multiple vehicles can visit a customer at the same time. We remove one-visit-at-the-time constraints (1e) and we also let the variables x_{ij}^k be defined for any non-negative integer, $x_{ij}^k \in \mathbb{Z}_+$, $a_{ij}^{k\ell} \in \mathcal{A}^{\mathcal{T}}$.

We call this model the Lower Bound Model (LBM).

Theorem 1. *The optimal value of the LBM is a lower bound on the optimal value of the CIRP-OB.*

Proof. Consider any CIRP-OB feasible solution and any time discretization. Let $\{\upsilon_i^\ell\}_{\ell \in L_i}$ be the sequence of decision times, namely, arrival to; departure from; or delivery time of location $i \in N_0$, with set index L_i. Similarly, let η_i^ℓ be the delivered quantity to customer $i \in N$ at time υ_i^ℓ, $\ell \in L_i$. We show that this CIRP-OB solution can be mapped to a LBM solution at no extra cost. Consider the following mapping of a time υ_i^ℓ to a LBM time $T_i(\upsilon_i^\ell)$,

$$T_i(\upsilon_i^\ell) = \max_{k \in \mathcal{T}_i} \left\{ t_i^k : t_i^k \leq \upsilon_i^\ell \right\}.$$

Every time point v_i^ℓ is mapped to the closest time t_i^k in the time discretization, with $t_i^k \leq v_i^\ell$, $i \in N_0$.

The travel times defined in (6) guarantee the LBM solution is feasible in time dimension, i.e., the vehicle flow balance is preserved. Let $v_0^{\ell_1}$ be any departure time from the depot to some customer $i \in N$, $\ell_1 \in L_0$, and let $v_i^{\ell_2}$ be the corresponding arrival time to i, $\ell_2 \in L_i$ in the CIRP-OB solution. Since the solution is feasible, it holds $v_0^{\ell_1} + \tau_i \leq v_i^{\ell_2}$. Let $k_1 \in \mathcal{T}_0$ and $k_2 \in \mathcal{T}_i$ such that $T_0(v_0^{\ell_1}) = t_0^{k_1}$ and $T_i(v_i^{\ell_2}) = t_i^{k_2}$, then,

$$\begin{aligned} t_0^{k_1} + \tau_{0i}^{k_1} &= \max_{k \in \mathcal{T}_i} \{t_i^k : t_i^k \leq t_0^{k_1} + \tau_i\} \\ &\leq \max_{k \in \mathcal{T}_i} \left\{t_i^k : t_i^k \leq v_0^{\ell_1} + \tau_i\right\} \\ &\leq \max_{k \in \mathcal{T}_i} \left\{t_i^k : t_i^k \leq v_i^{\ell_2}\right\} \\ &= t_i^{k_2}. \end{aligned}$$

An equivalent argument can be used to show that departing from a customer $i \in N$ location to the depot in the LBM is time feasible.

If a vehicle is waiting at the customer $i \in N$ location, then for any two consecutive times $v_i^{\ell_1}$ and $v_i^{\ell_2}$ in the same vehicle itinerary, it holds either $t_i^{k_1} = T_i(v_i^{\ell_1}) = T_i(v_i^{\ell_2}) = t_i^{k_2}$ and there is no waiting in the LBM solution; or $t_i^{k_2} = t_i^{k_2} + \sum_{k=k_1}^{k_2-1} \tau_{ii}^k$.

So any decision time in the CIRP-OB solution is mapped to a time in the LBM solution that is before in the time horizon. Since in the LBM waiting times at customer place and more than one visit at the same time are allowed, the itinerary given by the LBM times is feasible.

The LBM solution keeps the deliveries η_i^ℓ values, for all $i \in N$ and $\ell \in L_i$. The CIRP-OB solution is product flow balanced, so we only show that the LBM solution is feasible for inventory levels. Let $I_i(t)$ be the inventory level at time $t \in [0, H]$ in the CIRP-OB solution, for a customer $i \in N$. Note that $0 \leq I_i(t) \leq C_i$, for all $t \in [0, H]$. Consider any integer $k < K_i$ and the interval given by $[t_i^k, t_i^{k+1}]$. Let v_i^ℓ be the time of the last visit to customer i in that interval, $\ell \in L_i$. Since the CIRP-OB solution is feasible it must be that $I(v_i^\ell) \geq (t_i^{k+1} - v_i^\ell)u_i$. It also holds $t_i^k = T(v_i^\ell)$. It follows,

$$z_i^k = I_i^0 + \sum_{s=1}^{\ell} \eta_i^s - u_i t_i^k = I_i^0 + \sum_{s=1}^{\ell} \eta_i^s - u_i v_i^\ell + u_i(v_i^\ell - t_i^k) = I_i(v_i^\ell) + u_i(v_i^\ell - t_i^k),$$

Combining the above, we get,

$$z_i^k = I_i(v_i^t) + u_i(v_i^t - t_i^k) \geq (t_i^{k+1} - v_i^\ell)u_i + u_i(v_i^t - t_i^k) = u_i(t_i^{k+1} - t_i^k) = \bar{u}_i^{k+1},$$

and also,

$$z_i^k \leq C_i + u_i(v_i^t - t_i^k) \leq C_i + \bar{u}_i^{k+1}.$$

Thus, the inventory levels are feasible.

We conclude, the CIRP-OB solution can be mapped to a LBM solution with no extra cost. In particular, this is true for an optimal CIRP-OB solution, so the LBM is a valid lower bound model. □

Figure 1 shows an example of a mapping of travel times in the LBM. Black dots represent time points for the customer $i \in N$ and the depot time discretization. The solid line represents a CIRP-OB solution and the crosses are times at which the customer is visited in that solution. This solution can be mapped as it is shown with dashed lines. Note that in this mapping the vehicle in the LBM has to wait at the customer location in order to maintain the CIRP-OB itinerary, even when the vehicle in the CIRP-OB solution does not.

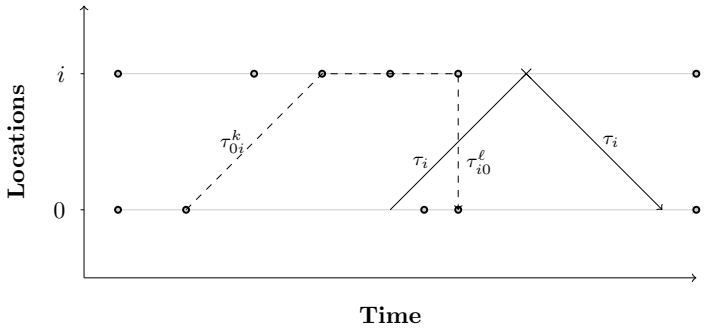

Figure 1: LBM Travel times example

Figure 2 shows an example of customer inventory level in a LBM solution mapped from a CIRP-OB solution. The solid line represents the inventory level for a feasible CIRP-OB solution whose delivery times are υ and delivery quantities are η. In this solution the inventory levels are always non-negative and below capacity C_i. The mapped LBM solution is represented with a dashed line, with deliveries at times t. The inventory levels in the LBM are above C_i at times t_i^{k-1} and t_i^k in order to deliver the same CIRP-OB amount.

As mentioned above, the travel times are rounded down, which means that it is possible that for two different time points in the departure location, the arrival time at the destination is the same. This condition gives more symmetry to the LBM, making it more difficult to solve in practice. We can remove some of the x variables that are redundant in the LBM, as stated in the following proposition.

Proposition 11. *In the LBM, if two points* $(0, t_0^{\ell_1})$ *and* $(0, t_0^{\ell_2})$ *in* $\mathcal{N}^{\mathcal{T}}$*, with* $t_0^{\ell_1} < t_0^{\ell_2}$ *have a timed arc connecting the same customer* $i \in N$ *at the same time* t_i^k*,* $k \in \mathcal{T}_i$*, then the variable* x *associated to the point* $(0, t_0^{\ell_1})$ *can be removed from the formulation.*

Proof. This result follows from the observation that the LBM without the time point $(0, t_0^{\ell_1})$ is still a lower bound model for the formulation, since the mapping T_i, $i \in N$, in Proposition 2 considers the closest time point in the time-expanded network for the CIPR-OB solution. □

In the LBM the OPC conditions cannot be imposed directly. However, we can check if the time-expanded network satisfies some conditions that allows to include the OPC in the model.

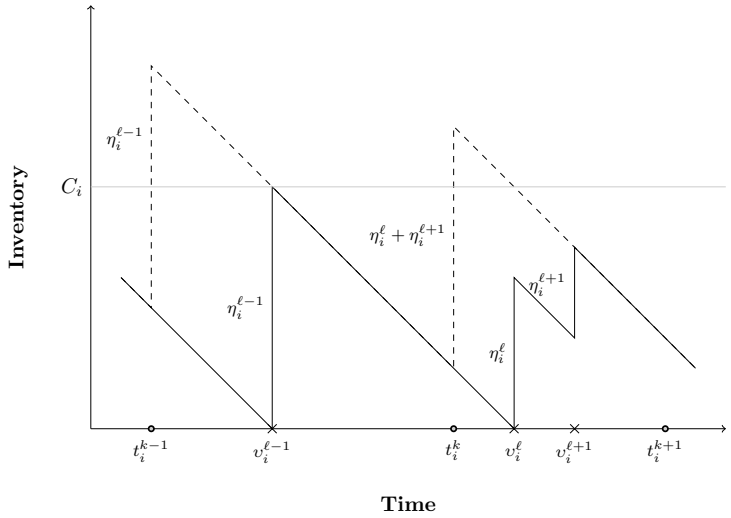

Figure 2: Example of a LBM inventory levels

6.1 Incorporating Vehicle Waiting Conditions into the LBM

In this section, we show how the waiting time conditions of Section 5 can be enforced by constraints in the LBM. We call a LMDO to a solution in the LBM for which there exists an MDO that can be mapped to the LBM solution. The proofs of all these results can be found in Appendix A.

Proposition 12. *(Condition Proposition 4): Any LMDO satisfies the following constraints,*

$$\begin{aligned} &C_i\nu_i^k \le z_i^k \le (C_i + \bar{u}_i^{k+1}) - \bar{u}_i^{k+1}\nu_i^k, && i \in N, k \in \mathcal{T}_i, \\ &x_{ii}^k \le m\nu_i^k, && i \in N, k \in \mathcal{T}_i, \\ &\nu_i^k \in \{0,1\}, && i \in N, k \in \mathcal{T}_i, \end{aligned} \tag{8}$$

provided $\bar{u}_i^{k+1} \le C_i$. The binary variable ν_i^k is equal to one if at least one vehicle is waiting at the customer $i \in N$ at time t_i^k, $k \in \mathcal{T}_i$; zero otherwise.

Proposition 13. *If for an index $k \in \mathcal{T}_i$ there exists a $\ell \in \mathcal{T}_0$, such that $t_0^\ell + \tau_{0i}^\ell = t_i^k$, then any LMDO satisfies the following constraints,*

$$\begin{aligned} &x_{ii}^{k-1} \le 1, && \text{if } i \in N_\ell, \\ &x_{ii}^{k-1} = 0, && \text{if } i \in N_g. \end{aligned}$$

Proposition 14. *(Condition Proposition 5): If for some customer $i \in N$ and $k \in \mathcal{T}_i$ there exist indexes $\ell_1, \ell_2 \in \mathcal{T}_0$ such that $t_0^{\ell_1} + \tau_{0i}^{\ell_1} = t_i^k$ and $t_0^{\ell_2} + \tau_{0i}^{\ell_2} = t_i^{k+1}$, then any LMDO satisfies the following constraint,*

$$(C_i - \bar{u}_i^k)(2 - x_{i0}^{\ell_1} - x_{ii}^k) + \bar{u}_i^{k+1} \ge z_i^{k-1} - \bar{u}_i^k. \tag{9}$$

Proposition 15. *(Condition Proposition 6): Consider a customer $i \in N_\ell$. For an index $k_1 \leq K_i - 1$ and an index $k_2 = \text{argmin}_{k \in \mathcal{T}_i}\{t_i^k \leq t_i^{k_1+1} + \frac{Q-C_i}{u_i}\}$ such that there exists a $\ell \in \mathcal{T}_0$, with $t_0^\ell + \tau_{0i}^\ell = t_i^{k_2}$. There exist a LMDO that satisfies the following constraint,*

$$\sum_{k=k_1}^{k_2} x_{ii}^k \leq k_2 - k_1.$$

6.2 Incorporating Visit Time Conditions into the LBM

Consider the condition given in Proposition 10: the delivery at time t_i^k must be the minimum of Q and $C_i - I_i^k$, where I_i^k is the inventory level before the delivery, for any customer $i \in N$. Let v_i^k be a binary variable that is equal to one if the minimum of the vehicle capacity Q and $C_i - I_i^k$ is Q; zero otherwise; for a customer $i \in N_g$ and a time t_i^k, $k \in k \in \mathcal{T}_i$. Any LMDO satisfies the following constraints,

$$y_i^k \geq (C_i + \bar{u}_i^k)(x_{0i}^\ell - v_i^k) - z_i^{k-1}, \tag{10a}$$
$$y_i^k \geq Q(x_{0i}^\ell + v_i^k - 1). \tag{10b}$$

For customers $i \in N_\ell$, the delivery is always $C_i - I_i^k$, so the following constraint is satisfied by any LMDO at a time t_i^k, $k \in k \in \mathcal{T}_i$,

$$y_i^k \geq (C_i + \bar{u}_i^k)x_{0i}^\ell - z_i^{k-1}, \qquad i \in N_\ell, k \in \mathcal{T}_i. \tag{11}$$

Note that $I_i^k = C_i + \bar{u}_i^k - z_i^{k-1}$, so when $x_{0i}^\ell - v_i^k = 1$ for customers $i \in N_g$, or when $x_{0i}^\ell = 1$ for customers $i \in N_\ell$, the inventory level after the delivery is C_i.

Inputs for the inequalities in this section are a lower and an upper bound for the number of visits to a customer. The lower bound η_i can be obtained computing the total product must be delivered $Hu_i - I_i^0$ divided by the vehicle capacity Q (maximum product can be carried by a vehicle), $\eta_i = \left\lceil \frac{Hu_i - I_i^0}{Q} \right\rceil$, for all $i \in N$. An upper bound M_i for the number of visits can be computed as the maximum total time that can be assigned to customer i divided by the total time it takes to visit it, $2\tau_i$. We have,

$$M_i = \left\lfloor \frac{mH - \sum_{j \in N: j \neq i} 2\tau_j \eta_j}{2\tau_i} \right\rfloor, \qquad i \in N.$$

For customers $i \in N$ with storage capacity $C_i > Q$, a system of inequalities can be derived from Propositions 7 and 9 for the LBM. Consider the following time indexes:

$$s_1^k = \max\{k \in \mathcal{T}_i : u_i t_i^k \leq Hu_i - C_i - (\eta_i - k)Q\} \qquad k = 1, \ldots, \eta_i,$$
$$s_2^k = \max\{k \in \mathcal{T}_i : u_i t_i^k \leq I_i^0 + (k-1)Q\} \qquad k = 1, \ldots, \eta_i,$$
$$s_3^k = \max\{k \in \mathcal{T}_i : u_i t_i^k \leq kQ - C_i + I_i^0\} \qquad k = 1, \ldots, \eta_i,$$
$$s_4^k = \max\{k \in \mathcal{T}_i : u_i t_i^k \leq I_i^0 + (k-1)Q - \epsilon\} \qquad k = 1, \ldots, \eta_i,$$

with $0 < \epsilon \leq C_i - Q$, given. Note that s_1^k is the index in $\mathcal{T}_i$ that represents the lower bound in the interval in (3) for the kth visit arrival time in the LBM, while s_2^k is the index that represents the upper bound in the interval. The index s_3 is the time index for the lower bound in the interval in (5) in the LBM and the s_4^k represents the upper bound of that interval.

Let ν_i a binary variable that is equal to one if the number of visits to customer i is η_i; zero otherwise. Let r_i^k be a binary variable that is equal to one if the interval given by $[t_i^{s_3^k}, t_i^{s_4^k}]$ has at least one visit; zero otherwise; for $k = 1, \ldots, \eta_i$. Any LMDO satisfies the following system,

$$(\eta_i + 1)(1 - \nu_i) \leq \sum_{\ell \in \mathcal{T}_0} x_{0i}^{\ell} \leq M_i - (M_i - \eta_i)\nu_i, \tag{12a}$$

$$\nu_i \leq \sum_{s=s_1^k}^{s_2^k} \sum_{\ell:(0,\ell)\in\delta^-(i,t_i^s)} x_{0i}^{\ell}, \qquad k = 1, \ldots, \eta_i \tag{12b}$$

$$\sum_{s=s_3^k}^{s_4^k} \sum_{\ell:(0,\ell)\in\delta^-(i,t_i^s)} x_{0i}^{\ell} \leq M_i r_i^k, \qquad k = 1, \ldots, \eta_i \tag{12c}$$

$$\sum_{k=1}^{\eta} r_i^k \leq \eta - 1 + \nu_i. \tag{12d}$$

provided the following conditions are satisfied in the time-expanded network,

1. there exists a index $\ell \in \mathcal{T}_0$ such that $t_0^{\ell} + \tau_{0i}^{\ell} = t_i^{s_1^k}$,
2. there exists a index $\ell \in \mathcal{T}_0$ such that $t_0^{\ell} + \tau_{0i}^{\ell} = t_i^{s_3^k}$.

The two conditions, time points in $\{t_0^{\ell}\}_{\ell \in \mathcal{T}_0}$ that permit to get to i at time $t_i^{s_1^k}$ and at time $t_i^{s_3^k}$, guarantee that a vehicle arriving at those times is not represented with a waiting arc in the LBM.

For customers $i \in N$ with storage capacity $C_i \leq Q$, a system of inequalities can be derived from Proposition 8. As before, we define time indexes representing the interval in (4) in the LBM:

$$\begin{aligned} s_1^k &= \max\{k \in \mathcal{T}_i : u_i t_i^k \leq H u_i - C_i + I_i^0 - (\eta - k + 1)Q\} && k = 1, \ldots, \eta_i, \\ s_2^k &= \max\{k \in \mathcal{T}_i : u_i t_i^k \leq I_i^0 + (k-1)Q\} && k = 1, \ldots, \eta_i. \end{aligned}$$

Let ν_i be a binary variable that is equal to one if the number of visits to customer i is

η_i; zero otherwise. Any LMDO satisfies the following system,

$$(\eta_i + 1)(1 - \nu_i) \leq \sum_{\ell \in \mathcal{T}_0} x_{0i}^{\ell} \leq M_i - (M_i - \eta_i)\nu_i, \tag{13a}$$

$$\nu_i \leq \sum_{s=s_1^k}^{s_2^k} \sum_{\ell:(0,\ell)\in\delta^-(i,t_i^s)} x_{0i}^{\ell}, \qquad k = 1, \ldots, \eta_i, \tag{13b}$$

provided there exists a index $\ell \in \mathcal{T}_0$ such that $t_0^{\ell} + \tau_{0i}^{\ell} = t_i^{s_1^k}$, so it is guaranteed the arrival time at $t_i^{s_1^k}$ is represented with a direct arc in the LBM.

6.3 Valid Inequalities for the Number of Visits

In Lagos et al. (2020) the authors present several valid inequalities for the CIRP. A large class of them corresponds to lower bounds on the number of visits to a customer, adaptations of inequalities presented in Coelho and Laporte (2014) and Archetti et al. (2007). In our setting, we keep the inequalities in Coelho and Laporte (2014) and we adapt them to the CIPR-OB. We consider a lower bound on the number of visits to a customer that must occur in an arbitrary time interval.

From a time $t_1 \in \{t_i^k\}_{k\in\mathcal{T}_i}$ and to a time $t_2 \in \{t_i^k\}_{k\in\mathcal{T}_i}$ with $t_1 < t_2$, the customer $i \in N$ consumption in that interval is $u_i(t_2 - t_1)$. At time t_1 the customer has at most C_i product in inventory, thus during $[t_1, t_2]$ the customer requires at least $u_i(t_2 - t_1) - C_i$ to be delivered. Since the maximum product that can be delivered during a visit is Q, we can compute a lower bound for the number of visits to i. For customers in N_ℓ, we also need to consider the waiting time at time t_1, having an extra variable for this case. Full details of the proof are found in Appendix A.

Proposition 16. *The following inequalities are valid for the LBM formulation,*

$$\left\lceil \frac{u_i(t_i^{k_2+1} - t_i^{k_1}) - C_i}{Q} \right\rceil \leq \sum_{k=k_1}^{k_2} \sum_{(0,\ell)\in\delta^-(i,t_i^k)} x_{0i}^{\ell}, \qquad i \in N_g, k_1 \in \mathcal{T}_i, k_1 < k_2, \tag{14}$$

$$\left\lceil \frac{u_i(t_i^{k_2+1} - t_i^{k_1}) - C_i}{Q} \right\rceil \leq x_{ii}^{k_1-1} + \sum_{k=k_1}^{k_2} \sum_{(0,\ell)\in\delta^-(i,t_i^k)} x_{0i}^{\ell}, \qquad i \in N_\ell, k_1 \in \mathcal{T}_i, k_1 < k_2. \tag{15}$$

Note that the previous inequalities are valid for any $k_1 < k_2$, with $k_1, k_2 \in \mathcal{T}_i$ and $i \in N$, but not all inequalities are useful in the LBM. We only add inequalities with $u_i(t_i^{k_2+1} - t_i^{k_1}) - C_i > 0$.

We also propose a Branch-and-Bound Strategy for solving the LBM. In Appendix B details about this strategy can be found.

6.4 Recovering Customer Storage Capacity Constraints

In order to get an LBM from the formulation (1), the travel times, customer storage capacities and number of visits at the same time are modified. If the time-expended network is fine enough, the travel times will be correct and the number of visits can be handled (as shown in Section 8.3), but the customer inventory bounds will be always $C_i + \bar{u}_i^k$, with $\bar{u}_i^k > 0$ for all $i \in N$, $k \in \mathcal{T}_i$. Even if the number of time points $\{t_i^k\}_{k \in \mathcal{T}_i}$ is large and they are no more than $\epsilon > 0$ apart, $\max_{k \in \mathcal{T}_i}\{t_i^{k+1} - t_i^k\} \leq \epsilon$, the inventory bound is strictly greater than C_i. How do we know LBM provides an useful solution for the problem? Theorem 2 states that for a fine time discretization the solution for x variables is the same for the LBM formulation with $C_i + \bar{u}_i^k$ and for the LBM with C_i.

The analysis in this section assumes the time discretization for any location $i \in N_0$ is Δ-homogeneous, i.e., for all $k \in \mathcal{T}_i$, $t_i^{k+1} - t_i^k = \Delta$. For notation, we represent by LBM$^+(\Delta)$ the LBM over a Δ-homogeneous time discretization and customer inventory capacity bounded by $C_i + u_i\Delta$. Similarly, the LBM(Δ) is the LBM over a Δ-homogeneous time discretization and customer inventory capacity bounded by C_i.

Consider an optimal solution for the LBM$^+(\Delta)$. Let $\omega_{i,a}^k(\Delta)$ be the kth arrival time to location $i \in N_0$ in the LBM solution, with $k = 1, \ldots, \eta_i$, and η_i the number of visits to i in the solution. Equivalently, let $\omega_{i,d}^k(\Delta)$ the kth departure time from the location $i \in N_0$, with $k = 1, \ldots, \eta_i$.

For a LBM(Δ) optimal solution, let $\upsilon_{i,a}^k(\Delta)$ be the kth arrival time to $i \in N_0$, and let $\upsilon_{i,d}^k(\Delta)$ be the kth departure time from location i of the optimal LBM solution, $k = 1, \ldots, \eta_i$.

Theorem 2. *There exist a $\Delta > 0$ and arrival and departure times $\omega_{i,a}^k(\Delta)$, $\omega_{i,d}^k(\Delta)$, $\upsilon_{i,a}^k(\Delta)$, $\upsilon_{i,d}^k(\Delta)$, with the same number of visits η_i, such that $\omega_{i,a}^k(\Delta) = \upsilon_{i,a}^k(\Delta)$ and $\omega_{i,d}^k(\Delta) = \upsilon_{i,d}^k(\Delta)$, for all $i \in N_0$ and $k = 1, \ldots, \eta_i$.*

Sketch of Proof. We show that this condition holds since, for a feasible instance of the problem, a rational optimal solution exists. For contradiction, we assume that there is no $\Delta > 0$ for which the times and delivered quantities of an optimal solution for LBM$^+(\Delta)$ are the same of a LBM(Δ) optimal solution and, using the fact the parameters are rational, we reach a contradiction (we get that for a non-empty polytope there is no rational solution). □

7 Finding Feasible Solutions for the CIRP-OB

In this section we present two different methods to get feasible solutions for a CIPR-OB instance. First, we consider a MIP formulation that, if feasible, provides feasible solutions for the continuous time problem for any time discretization. This formulation is derived in a similar way to LBM: the formulation (1) is changed and then we show the solution we get from the new formulation has a transportation cost that is an upper bound for the problem. Then, we present a MIP formulation that finds continuous visit times and deliveries for an optimal number of visits from the LBM; thus, we can determine whether a LBM optimal solution is feasible for the CIPR-OB or not.

7.1 Upper Bound Model (UBM)

An upper bound formulation for the CIRP-OB can be derived from formulation (1). The model we propose contains the same variables and constraints defined for formulation (1), but the travel times are modified: they are rounded up in the time-expanded network. The travel time at a time t_i^k, $i \in N_0$, $k \in \mathcal{T}_i$, to a location $j \in N_0$, is given according to the following expression,

$$\tau_{ij}^k = \begin{cases} \min_{\ell \in \mathcal{T}_j}\{t_j^\ell : t_j^\ell \geq t_0^k + \tau_j\} - t_0^k, & i = 0, j \in N, t_0^k + \tau_j \leq H \\ \min_{\ell \in \mathcal{T}_0}\{t_0^\ell : t_0^\ell \geq t_i^k + \tau_i\} - t_i^k, & i \in N, j = 0, t_i^k + \tau_i \leq H \\ t_i^{k+1} - t_i^k & i = j, k < K_i. \end{cases} \tag{16}$$

We call this model the Upper Bound Model (UBM). Even though the formulation decision times are restricted to the location time discretizations, the UBM finds feasible solutions for the CIRP-OB. Therefore, this model is a valid upper bound model for the problem.

Proposition 17. *Any feasible solution to the UBM is a feasible solution to the CIRP-OB.*

Proof. The UBM travel times ensure that a feasible solution for the model is feasible for the continuous time problem. Any vehicle whose travel times are given by (16), it gets to the destination after a vehicle traveling according to τ_i, $i \in N$. Note also that the continuous time solution can keep the UBM solution customers itinerary: if the continuous time vehicle gets too early to the customer, it can wait longer at the depot before departure. The constraints in formulation (1) ensure the solution is feasible for the CIRP-OB. □

Even if a feasible solution to an instance of CIRP-OB exists, UBM may not find it. The travel times and the constraints of the formulation might produce an infeasible model. For a Δ-homogeneous time discretization for the UBM, i.e., time intervals such that $t_i^{k+1} - t_i^k = \Delta > 0$, for all $i \in N_0$ and $k \in \mathcal{T}_i$, we show conditions on Δ that ensure a feasible model.

Proposition 18. *If $\Delta > 0$ satisfies,*

$$u_i \leq \frac{I_i^0}{\Delta\left(\left\lceil \frac{C_i}{Q} \right\rceil + \left\lceil \frac{\tau_i}{\Delta} \right\rceil\right)}, \tag{17}$$

$$\left\lceil \frac{C_i}{Q} \right\rceil \leq \frac{H}{\Delta} - 2\left\lceil \frac{\tau_i}{\Delta} \right\rceil, \tag{18}$$

for all $i \in N$, then the UBM with unlimited number of vehicles is feasible using a Δ-homogeneous time discretization.

The proof of Proposition 18 can be found in Appendix A.

7.2 Feasibility Check Model (FCM)

An optimal solution to LBM specifies a number of visits to each customer. These visits have a total transportation cost that is a lower bound for the CIRP-OB, but it is unknown whether this number of visits leads to a solution that is feasible for the continuous time problem or not.

In this section we present a model that allows us to determine whether a continuous-time feasible solution with this number of visits exists. We formulate a model that takes as an input the number of visits n_i, $i \in N$, and finds new continuous visiting times, defined in the interval $[0, H]$, and delivered quantities, defined in $[0, Q]$. These times and quantities must satisfy the CIRP-OB constraints, including the conditions we relax for the LBM: the travel times are given by the parameter τ_i, $i \in N$; the customer inventory capacities are bounded by C_i, $i \in N$; and the number of vehicles waiting at the same time at the customer location is at most one. We construct a mixed-integer programming (MIP) model to decide (revise) the visiting times and delivered quantities, while preserving the number of visits to each customer. We call this formulation Feasibility Check Model (FCM). Naturally, it may be that no feasible CIRP-OB solution using these number of visits exists, in which case a different (better) optimal LBM solution is needed.

Each visit to a customer is associated to an out-and-back route. Let $R = \{1, \ldots, \sum_{i \in N} n_i\}$ be the index set of routes. For each customer $i \in N$, let $R_i \subseteq R$ be the set of routes that visit customer $i \in N$. Let $c(r) \in N$ denote the customer visited on route $r \in R$. We assume customers have at least one visit, $n_i \geq 1$ and, w.l.o.g., the routes in R_i visit customer i in route index order, $R_i = \{r_1^i, r_2^i, \ldots, r_{n_i}^i\}$ with $r_{k-1}^i < r_k^i$ for all $k = 2, \ldots, n_i$. We have $c(r_k^i) = i$ for all $k = 1, \ldots, n_i$.

Let variable v_r be the arrival time at customer $c(r)$ and let w_r be the departure time from customer $c(r)$ of route $r \in R$. Inventory variables, z_r, denote the inventory level at customer $c(r)$ immediately prior to the first delivery on route r. Let y_{r_1,r_2} be a binary variable that is one if the same vehicle does route r and then r': zero otherwise. The variable ζ is the minimum slack time between any two consecutive visits.

$$
\begin{aligned}
\max \quad & \zeta, \\
\text{s.t.} \quad & \zeta \le v_{r_k^i} - w_{r_{k-1}^i}, && i \in N, k = 2, \ldots, n_i, && \text{(19a)} \\
& z_{r_1^i} = I_i^0 - u_i v_{r_1^i}, && i \in N, && \text{(19b)} \\
& z_{r_k^i} = z_{r_{k-1}^i} + f_{r_{k-1}^i} - u_i (v_{r_k^i} - v_{r_{k-1}^i}), && i \in N, k = 2, \ldots, n_i, && \text{(19c)} \\
& z_{r_{n_i}^i} + f_{r_{n_i}^i} \ge u_i (H - v_{r_{n_i}^i}), && i \in N, && \text{(19d)} \\
& z_r + f_r \le C_{c(r)} + u_{c(r)} (w_r - v_r), && r \in R, && \text{(19e)} \\
& v_{r'} \ge w_r + \tau_{c(r)} + \tau_{c(r')} - M(1 - y_{r,r'}), && r, r' \in R && \text{(19f)} \\
& \tau_{c(r)} \le v_r \le w_r \le H - \tau_{c(r)}, && r, r' \in R, r \ne r' && \text{(19g)} \\
& w_r = v_r, && r \in R, c(r) \in N_g && \text{(19h)} \\
& \sum_{r' \in R} y_{r,r'} \le 1, && r \in R && \text{(19i)} \\
& \sum_{r' \in R} y_{r',r} \le 1, && r \in R && \text{(19j)} \\
& \sum_{r \in R} \sum_{r' \in R} y_{r,r'} \ge |R| - m && && \text{(19k)} \\
& 0 \le f_r \le Q, && r \in R && \text{(19l)} \\
& y_{r,r'} \in \{0, 1\}, && r, r' \in R, && \\
& z_r \ge 0, && r \in R. &&
\end{aligned}
$$

Constraints (19a) ensure that the visit times at customers occur sequentially with at least ζ units of time apart. Constraints (19b) and (19c) set the inventory levels just prior to each visit at a customer. Constraints (19d) ensure that inventory after the last delivery is sufficient to meet demand until the end of the planning horizon. Constraints (19e) enforce that the vehicle remains long enough at the customer to deliver its whole load, while not violating the capacity limit. Constraints (19f) ensure that for each of the vehicles the visit times at customers properly account for travel times between locations, and, in case a vehicle performs multiple routes, properly account for travel times to and from the depot in between consecutive routes in its itinerary. Constraints (19g) impose bounds on arrival and departure times. Note that the w_r variables are only needed if $c(r) \in N_\ell$, but for simplicity of exposition, we include them for all r, and impose the constraints (19h) in the model. Constraints (19i)-(19k) ensure a sequence of routes for each vehicle itinerary that satisfies the vehicle availability m. Finally, constraints (19l) enforce deliveries cannot be more than the vehicle capacity Q.

If the FCM is feasible and has optimal value $\zeta^* > 0$, then the solution provides a feasible solution to the CIRP-OB with vehicle movement cost the same as the cost of the LBM solution. If no feasible solution with positive ζ exists in the FCM, then for the number of visits n_i given as an input, $i \in N$, there are no visit times and deliver quantities that result in a feasible CIRP-OB solution.

In order to guarantee the LBM solution is CIRP-OB feasible, we only need to find a feasible solution to the FCM with $\zeta > 0$. In our implementation we start with the sequence provided by the LBM solution for the FCM model (initial solution for binary y variables) and we run the solver until a solution with $\zeta > 0$ is found.

In practice, we have seen this model solves very fast, even for large instances it takes no more than few seconds. For the instances we present, there is no need of including OPC into the FCM.

8 Dynamic Discovery Discretization Algorithm (DDD) for the CIRP-OB

The DDD algorithm is presented in Boland et al. (2017) for the continuous time service network design problem. The central idea of the algorithm methodology is to work with a partial time discretization, that is sequentially and precisely refined, so it is guaranteed an optimal continuous time solution can be produced. This algorithm consists of the following steps:

1. find a lower bound solution for the continuous time problem in a time-expanded network;

2. given a lower bound solution, determine whether it is feasible for the (original) problem or can be converted into a feasible solution for the (original) problem (which implies it is optimal);

3. if the solution cannot be converted, improve the LBM by adding time points to the partial discretization.

In this section we develop the full algorithm for the CIRP-OB. Steps 1 (LBM) and 2 (FCM) have been already presented in previous sections, so we describe Step 3 and the execution of the full algorithm.

With respect to the CIRP-OB, the LBM considers three relaxations: travel times are rounded down; inventory capacity is enlarged by adding the consumption of the next time interval; and multiple visits to a customer at the same time is allowed. After checking that an optimal solution for the LBM is not feasible for the CIRP-OB problem, we detect one or more of the mentioned relaxations to correct in the time-expanded network, so the current optimal solution is no longer feasible for the LBM.

8.1 Correcting Travel Times

Since the travel times in the LBM are shorter than τ_i, $i \in N$, some difficulties can arise when trying to convert the LBM solution into a feasible continuous time solution: it is not possible to get to a customer before it runs out of product since the out-and-back route duration is too long, or the vehicle itinerary is larger than the total time H, or both.

Algorithm 1 takes as input the optimal LBM times at which vehicles depart from a location. For each $x_{ij}^k > 0$, $a_{ij}^{k\ell} \in \mathcal{A}^{\mathcal{T}}$, this algorithm checks whether the travel times starting at time t_i^k are correct. If a shorter travel time to the depot, $j = 0$, is detected, the algorithm adds the point $t_i^k + \tau_i$ to the set $\{t_0^\ell\}_\ell$, provided the new time point satisfies $t_i^k + \tau_i \leq H$. Equivalently, if a shorter travel time from the depot, $i = 0$, is detected, the time point $t_i^k + \tau_j$ is added to the set $\{t_j^\ell\}_\ell$, provided $t_i^k + \tau_j \leq H$.

```
forall x_{i0}^k > 0 do
    set s_1 = t_i^k + τ_i;
    if s_1 ∉ {t_0^ℓ}_{ℓ∈T_0} and s_1 ≤ H then
        add s_1 to {t_0^ℓ}_{ℓ∈T_0};
    end
end
forall x_{0j}^k > 0 do
    set s_2 = t_0^k + τ_j;
    if s_2 ∉ {t_j^ℓ}_{ℓ∈T_j} and s_2 ≤ H then
        add s_2 to {t_j^ℓ}_{ℓ∈T_j};
    end
end
```

Algorithm 1: Travel time correcting algorithm.

This algorithm refines the time-expanded network in order to get a LBM solution whose travel times are not rounded down.

8.2 Correcting Customer Storage Capacities

Increasing the customer storage capacity in the LBM implies that vehicles can deliver more than is possible when the inventory capacity is C_i, $i \in N$. More product can be accommodated at the customer storage, so fewer out-and-back routes are necessary to supply customers. Also, if a vehicle is waiting in the LBM solution, it can return to the depot before it is possible in continuous time since it does not have to wait the time needed for the delivery.

Algorithm 2 corrects the customer storage capacities. This algorithm checks whether customer storage capacity C_i is being violated by vehicles deliveries in the LBM solution, $i \in N$. If the inventory z_i^k, $k \in \mathcal{T}_i$, exceeds C_i for a large amount, then the time interval given by $[t_i^k, t_i^{k+1}]$ is too long and a new time point is needed to reduce it. A new time point in the middle of that interval, i.e., the time point $\frac{t_i^{k+1}+t_i^k}{2}$ is added to the set $\{t_i^k\}_k$. Since in the LBM the customer capacities are always larger than C_i, we consider a $\epsilon > 0$ tolerance. If $[t_i^k, t_i^{k+1}] \leq \epsilon$, this interval is not split by adding more time points.

input: $\epsilon > 0$
forall $i \in N$ **do**
  forall $k \in \mathcal{T}_i$ **do**
    if $C_i < z_i^k$ *and* $t_i^{k+1} - t_i^k > \epsilon$ **then**
      add time point $\frac{t_i^{k+1}+t_i^k}{2}$ to the set $\{t_i^l\}_l$;
    end
  end
end

Algorithm 2: Inventory capacity correcting algorithm.

Lemma 1. *For a given $\epsilon > 0$, the maximum number of points added by Algorithm 2 is $|N|\lceil 2H/\epsilon \rceil$.*

Proof. We show that in a interval $[t, t+\epsilon/2]$, $t \in [0, H-\epsilon/2]$, at most one point is added by the algorithm. Suppose, for contradiction, in that interval there are two points $t_1, t_2 \in [t, t+\epsilon/2]$, $t_1 < t_2$ and at least one of them is added by the algorithm. If t_1 is in the interval and then t_2 is included by the algorithm, then there exists a t_3 such that $t_2 = t_1 + (t_3 - t_1)/2 \leq t_1 + \epsilon/2$, so $t_3 - t_1 \leq \epsilon$. But $t_3 - t_1 > \epsilon$, otherwise the algorithm does not consider that interval, reaching the contradiction. If t_2 is in the interval and then t_1 is added, then there exists a t_3 such that $t_1 = t_3 + (t_2 - t_3)/2 \geq t_2 - \epsilon/2$, so $t_2 - t_3 \leq \epsilon$. But $t_2 - t_3 > \epsilon$, so we reach the contradiction.

We conclude by observing the maximum number of intervals of length $\epsilon/2$ for a given location $i \in N$ is $\lceil 2H/\epsilon \rceil$. □

8.3 Correcting Number of Visits

In the CIRP-OB vehicles are allowed to wait at the customer location but there must be at most one vehicle at the same time. In the LBM this constraint is relaxed. Therefore, if the solutions we get from the LBM are not continuous time feasible, then one possible problem is given by multiple vehicles waiting at a customer location at the same time.

We consider a $\epsilon > 0$ and a LBM solution as an input. Let $\mathcal{S}_i$ be the set of indexes for customer $i \in N$ whose times t_i^k, $k \in \mathcal{S}_i$, have more than one visit at the same time. For all time points t_i^k, $k \in \mathcal{S}_i$ we add the time $t = t_i^k + \epsilon$, provided $t_i^{k+1} > t$ and $t \leq H$. Since at time t_i^k there might be a vehicle that is waiting to do a delivery at time t_i^{k+1} or after, the travel times must also be corrected. We add the time $t - \tau_i$ to $\{t_0^\ell\}_{\ell \in \tau_0}$, provided $t - \tau_i \geq 0$.

The algorithm for correcting multiple visits at the customer is shown in Algorithm 3.

input: $\epsilon > 0$
1 detect time points $\{t_i^k\}_{i \in N, k \in \mathcal{S}_i}$ with multiple visits;
2 **forall** $\{t_i^k\}_{i \in N, k \in \mathcal{S}_i}$ **do**
3 | add one-visit-at-the-time constraint for t_i^k;
4 | add time $t_i^k + \epsilon$ to $\{t_i^\ell\}_{\mathcal{T}_i}$;
5 | add time $t_i^k + \epsilon - \tau_i$ to $\{t_0^\ell\}_{\mathcal{T}_0}$;
6 **end**

Algorithm 3: Correcting multiple visits at the same time algorithm.

In line 3, we include the condition that the customer $i \in N$ cannot have more than one visit at time t_i^k, $k \in \mathcal{S}_i$. This is done by removing the LBM modification 3 (Section 6), so the constraint $x_{ii}^k + x_{i0}^k \leq 1$ is added to the model (note that the constraint also imposes that the variables x_{ii}^k and x_{i0}^k must be binary).

The following lemma and proposition state Algorithm 3 only adds a finite number of time points and also preserves the valid lower bound condition of the LBM.

Lemma 2. *For a given $\epsilon > 0$, the maximum number of points added by Algorithm 3 is $2|N|\lceil 2H/\epsilon \rceil$.*

Proof. Note that for any customer $i \in N$ and any time interval $[t, t+\epsilon/2]$, with $t \in [0, H-\epsilon]$ at most one time point can be added by the algorithm, so no more than $2\lceil 2H/\epsilon \rceil$ points are added to i. Since for each time $t+\epsilon$ added to i there is a time $t+\epsilon-\tau_i$ added to the depot discretization, the algorithm generates at most $2|N|\lceil 2H/\epsilon \rceil$ time points. □

Proposition 19. *There exists a $\epsilon > 0$ such that Algorithm 3 preserves the valid lower bound condition of the LBM.*

Proof. In Lagos et al. (2020) the authors prove that for any feasible instance of the CIRP there exists a solution with visiting times and quantities rational. Since the CIRP-OB is a particular case, there exists a rational solution for this problem. In particular, there exists a $\epsilon > 0$ such that all visit times are at least ϵ apart. Thus, if the time-expanded network is dense enough, such that for all $i \in N$ and $k \in \mathcal{T}_i$, $t_i^{k+1} - t_i^k \leq \epsilon$, imposing the one-visit-at-the-time constraint at each $(i, t_i^k) \in \mathcal{N}^{\mathcal{T}}$ preserves at least one optimal solution for the problem. □

8.4 DDD Algorithm for the CIRP-OB

In this section we describe the DDD algorithm for the CIRP-OB, present properties and guarantees of the algorithm and mention some implementation details. The algorithm uses the correcting algorithms described in Sections 8.1, 8.2 and 8.3. The DDD algorithm is shown in Algorithm 4.

input: $\epsilon > 0$
1 set a initial time discretization $\{t_i^k\}_k$, for all $i \in N_0$;
2 solve the LBM, get optimal number of visits;
3 solve FCM with LBM number of visits;
4 **if** *FCM is feasible and* $\zeta^* > 0$ **then**
5 | return optimal solution;
6 **else**
7 | add time points using *travel time correcting Algorithm 1*;
8 | add time points using *inventory capacity correcting Algorithm 2* with ϵ;
9 | add time points and constraints using *visits correcting Algorithm 3* with ϵ;
10 **end**
11 **if** *no time points are added to the time-expanded network* **then**
12 | stop;
13 **end**
14 go to step 2;

Algorithm 4: DDD algorithm for the CIRP-OB

Algorithm 4 receives a $\epsilon > 0$ tolerance and it starts with a initial time-expanded network. It solves the LBM formulation and gets an optimal number of visits to each customer. Then, the LBM solution is checked using the FCM model, so it is determined whether it can be converted to a CIRP-OB feasible solution or not. If it can, then the FCM solution is optimal. If it not, then the time-expanded network and the LBM must be improved. The travel time, inventory capacity and visits correcting algorithms are executed. The inventory and visits correcting algorithm are executed using a ϵ-tolerance, i.e., time points are added if the time intervals are greater than ϵ. If at least one time point is added to the time-expanded network in steps 7, 8 and 9, the LBM is solved in the new network. If no points are added, then the algorithm stops.

The algorithm starts with time discretizations $\{t_i^k\}_{\mathcal{T}_i}$, $i \in N$, such that for times 0 and H are contained in all sets, $\{0, H\} \subseteq \{t_i^k\}_{\mathcal{T}_i}$. We also include time points needed for the visit time conditions in Section 6.2.

For all $i \in N_g$, the set $\{t_i^k\}_{\mathcal{T}_i}$ has the following time points:

- $s_{i,1}^k = H - \frac{C_i + (\eta_i - k)Q}{u_i}$, for all $k = 1, \ldots, \eta_i$;
- $s_{i,2}^k = \frac{I_i^0 + (k-1)Q}{u_i}$, for all $k = 1, \ldots, \eta_i$;
- $s_{i,3}^k = \frac{kQ - C_i + I_i^0}{u_i}$, for all $k = 1, \ldots, \eta_i$;
- $s_{i,4}^k = \frac{I_i^0 + (k-1)Q}{u_i} - \epsilon$, for all $k = 1, \ldots, \eta_i$.

For all $i \in N_\ell$, the set $\{t_i^k\}_{\mathcal{T}_i}$ has the following time points:

- $s_{i,1}^k = H - \frac{C_i - I_i^0 + (\eta_i - k + 1)Q}{u_i}$, for all $k = 1, \ldots, \eta_i$;
- $s_{i,2}^k = \frac{I_i^0 + (k-1)Q}{u_i}$, for all $k = 1, \ldots, \eta_i$;

For the depot, the set $\{t_0^k\}_{\mathcal{T}_0}$ has the following time points:

- $s_{0,i,1}^k = s_{i,1}^k - \tau_i$, for all $i \in N$ and $k = 1, \ldots, \eta_i$;
- $s_{0,i,2}^k = s_{i,2}^k - \tau_i$, for all $i \in N$ and $k = 1, \ldots, \eta_i$;

Theorem 3 below states that the DDD algorithm is an exact algorithm for the CIRP-OB, so for a feasible instance with rational data, it finds an optimal solution. For proving the theorem, the following proposition is needed: for any tolerance $\epsilon > 0$, the DDD algorithm stops in a finite time.

Proposition 20. *For a given $\epsilon > 0$, Algorithm 4 finishes after a finite number of steps.*

Sketch of Proof. Considering Lemmas 1 and 2, we know that the number of points added by Algorithm 4 is finite, for any initial discretization and $\epsilon > 0$, so the number of iterations is finite. □

Theorem 3. *Consider a feasible CIRP-OB instance. There exists an $\epsilon > 0$ such that Algorithm 4 finds an optimal solution.*

Sketch of Proof. Since there exists a ration solution (feasible instance), we know that there exists an $\epsilon > 0$ that guarantees solving the LBM with a homogeneous time discretization ϵ, the solution found is a lower bound and all the relaxations are corrected (travel times, inventory capacities and multiple visits at the same time). For this ϵ, the Algorithm 4 finishes at a finite time. □

In practice $\epsilon > 0$ needed for Algorithm 3 might be too small and solving the LBM with a time-expanded network whose intervals are less than ϵ can be intractable. In the implementation the time-expanded network is refined using Algorithms 1 and 2 (steps 7 and 8) only, and in case no points are added for those algorithms, it stops. Note that in this variant, no matter how large or small the input ϵ is, the LBM is always a lower bound model for the problem. This property allows us to use any discretization. We have seen that solving the LBM in course time discretizations leads to solution that are in many cases optimal. In addition, the model solves faster. We start the algorithm with a ϵ tolerance and in case no more refinements are made to the network, is reduced to $\epsilon/2$. This allows us to adjust the number of time points added by Algorithm 2 as needed.

As the time expanded network is refined, the LBM finds better (less travel times or capacity violations) integer solutions when running the branch-and-bound. We can take advantage of this exploration by running the FCM for each new integer solution is found, so it is possible to get feasible solutions before the algorithm finishes.

9 Computational Experiments

9.1 Instances

For our computational experiments, there are parameters and conditions that are the same for all generated instances. These are the following:

- vehicle capacity $Q = 50$;
- time horizon $H = 20$;
- initial inventory I_i^0 whose value is equal to customer storage capacity C_i for all $i \in N$;
- speed is equal to one;
- travel times are symmetric and equal to transportation costs.

The parameters that vary among the instances are determined in such a way that we study the impact of four factors on the performance of the algorithm:

- number of customers;
- geographical distribution of locations;
- percentage of customers whose storage capacity is less than the vehicle capacity Q, N_ℓ;
- customer usage rates.

The number of customers is a natural dimension to consider in our instances. We consider instances with up to 30 customers and three different values: 10, 20 and 30.

The customer locations with respect to the depot is a factor that impacts on the algorithm performance. Note that in the CIRP-OB problem, only the distance to the depot is relevant, so we consider that the customers are located in a line in which the depot is at position 0. Customer locations are randomly generated with an average distance from the depot of 5. The sample is given by a uniform $\text{Unif}(5-\delta, 5+\delta)$ distribution, and we consider three different values for $\delta = 0.5, 1.5, 2.5$.

Another factor that is important to include is the number of customers whose storage capacity is less than the vehicle capacity, since for these customers, vehicle waiting is allowed. We generate instances with 25%, 50% and 75% of the customers having capacity $C_i < Q$. More specifically, for each customer in N_ℓ, the capacity C_i is an integer uniformly randomly chosen from the interval $\left[\frac{Q}{2}, Q\right)$ and for each customer in N_g, the capacity C_i is an integer uniformly randomly chosen from the interval $\left[Q, \frac{3Q}{2}\right)$. In this way, we generate instances with different proportions of customers with large and small inventory capacity relative to the vehicle capacity Q.

Once the geographical distribution and the storage capacities are sampled, we generate customer usage rates. For each $i \in N$, we define $low_i = \frac{C_i}{H}$ as the lower limit for u_i; a lower value for u_i implies the customer needs no visits from the depot. We also define an

upper value for u_i, $up_i = \frac{\min\{C_i, Q\}}{2\tau_i}$. We consider three usage levels for instance generation, each level determine the range from which we sample usage values. For instances with a low usage rate (level 1) we get usage rates from Unif $\left(low_i, \frac{low_i + up_i}{2}\right)$; for instances with medium rate (level 2) we get rates from Unif(low_i, up_i); and for high rate (level 3) we get rates from Unif $\left(\frac{low_i + up_i}{2}, up_i\right)$, for all $i \in N$.

We generate 81 different instance types, each one is labeled N for the number of customers, G for the geographical distribution, P for the percentage of customers in N_ℓ, and U for the usage level. For example, an instance labeled N20G2P25U3 has 20 customers, the customer location distributions are sampled using Unif $(3.5, 6.5)$, it has 5 customers with storage capacity $C_i < Q$, and usage rates are sampled using Unif $\left(\frac{low_i + up_i}{2}, up_i\right)$ for all customers $i \in N$.

For each instance type, we generate three samples. In order to find the number of vehicles m for each sample, we run the UBM (described in Section 7.1) with an objective function that seeks to minimize the number of vehicles. The number of vehicles is a critical parameter for the instance: a small value can make the instance infeasible, while a large value can cause many vehicles not to be used (which may make the instances "too easy"). The time discretization for the UBM is homogeneous, $\Delta = t_i^k - t_i^{k-1}$, $i \in N$, $k \in \mathcal{T}_i$, with $\Delta > 0$. Let $\Delta' > 0$ be the maximum value that satisfies conditions (17) and (18). We consider $\Delta = \min\{\Delta', 0.25\}$ for the UBM. The model runs for up to two hours, and the number of vehicles, m, for an instance is set to the best value found. Then, we run the DDD algorithm. We have seen that randomly generated instances likely result in instances that are solved in one iteration of the DDD algorithm. We have identified these instances before running our experiments and replaced them with re-sampled instances that require more iterations. We repeat the procedure until we have 3 samples for each of the 81 instance types. The generated instances can be found at `https://github.com/felipelagos/cirplib`.

9.2 Results

We generate three samples for each of the 81 types of instances, a total of 243 instances. We run the DDD algorithm for each instance for up to 2 hours. The algorithm starts with $\epsilon = 1$, which is reduced by half ($\epsilon \leftarrow \epsilon/2$) if no new time points are added to the time-expanded network. The algorithm either finishes with a provable optimal solution, with a feasible solution (and an optimality gap), or with no solution. We have seen, in our experiments, that in the final solution (optimal or feasible) there are no multiple vehicles visiting a customer at the same time. In practice, for random instances generated as we do, multiple visits at the same time relaxation does not have to be addressed.

Table 1 summarizes the results by different instance dimensions: number of customers (N); usage rate level (U); geographical distribution (G); and percentage of customers in N_ℓ (P). The table shows the number of instances for which the algorithm finishes with an optimal solution (Optimal), with a feasible solution (Feasible), or with no solution (No Solution). We also show the average number of vehicles (Vehicles) of the instance and the average number of visits (Visits) of the found feasible solution. Finally, we report the average number of time points added to a customer time discretization (Time Points) and the percentage that

	N			U			G			P		
	10	**20**	**30**	**1**	**2**	**3**	**1**	**2**	**3**	**25**	**50**	**75**
Optimal	80	69	57	80	72	54	79	68	59	68	72	66
Feasible	1	10	14	0	8	17	2	9	14	5	8	12
No Solution	0	2	10	1	1	10	0	4	8	8	1	3
Vehicles	7.83	15.47	23.64	13.36	15.67	17.91	15.99	15.65	15.30	15.15	15.67	16.12
Visits	13.52	25.86	37.92	21.85	25.61	28.63	25.85	24.90	24.92	22.79	25.41	27.35
Time Points	23.22	36.07	42.09	34.43	33.16	33.79	29.20	33.61	38.56	37.42	32.61	31.35
Initial Points (%)	27.45	17.54	13.60	17.16	19.59	21.84	21.67	19.47	17.45	18.99	20.16	19.43

Table 1: Results summary.

represents the initial number of time points (starting time-expanded network) with respect to the total number at the last iteration (Initial Points).

Out of the 243 instances, the DDD algorithm optimally solves 206 of them (84.77%), finds a feasible solution (with a positive gap) for 25 instances (10.29%), and finds no solution for 12 instances (4.94%) (Recall that these instances were selected to be "difficult" for the DDD algorithm). In the detail for the number of customers, almost all of the instances are optimally solved for N10, but only 57 instances out of 81 are optimally solved for N30. In general, we see that as we increase the number of customers, the problem becomes more difficult to solve. We see a similar trend for the usage rate: as the level increases, fewer instances are solved optimally, and more instances have an optimality gap or no solution. When the geographical customer distributions are analyzed, the clustered instances (G1: $\delta = 0.5$, no more than 0.5 units of distance to the depot) are easier to solve than the non-clustered instances. The percentage of customers in N_ℓ does not show a clear impact on the algorithm.

In Table 1, we also see that the number of vehicles needed for the instance is related to the number of customers and the usage rate level. In both cases, as the number increases (N10 to N30, U1 to U3), more vehicles are necessary. However, the customer distribution (G) and the percentage of N_ℓ (P) have no clear impact on the number of vehicles.

The number of visits in the feasible solution depends on N, U, and P, but not on G. Note that if the instance has more customers in N_ℓ, then more visits have to be made, independent of the usage rate level: instead of waiting, the vehicles make more out-and-back trips to customers whose capacity C_i is less than Q.

For N10 instances, each customer has, on average, about 23 different time points in its time discretization set when the algorithm stops. When the number of customers is N20, the average is about 36, and for N30, about 42. This indicates that the more customers that are in the instance, the more time points that are needed, possibly because there is more flexibility in the vehicle itineraries (more options for how to combine customer visits in a vehicle itinerary). Another dimension that affects the number of visits is the geographical distribution: customers in clustered instances (G1) need fewer time points than customers in non-clustered instances (G3). Note that for clustered instances, the travel times τ_i are similar for all $i \in N$. Thus, the vehicle visit customers at similar times, which is not true for non-clustered instances.

Figure 3 reports the experiments running time. For each dimension (N, U, G, P), a plot

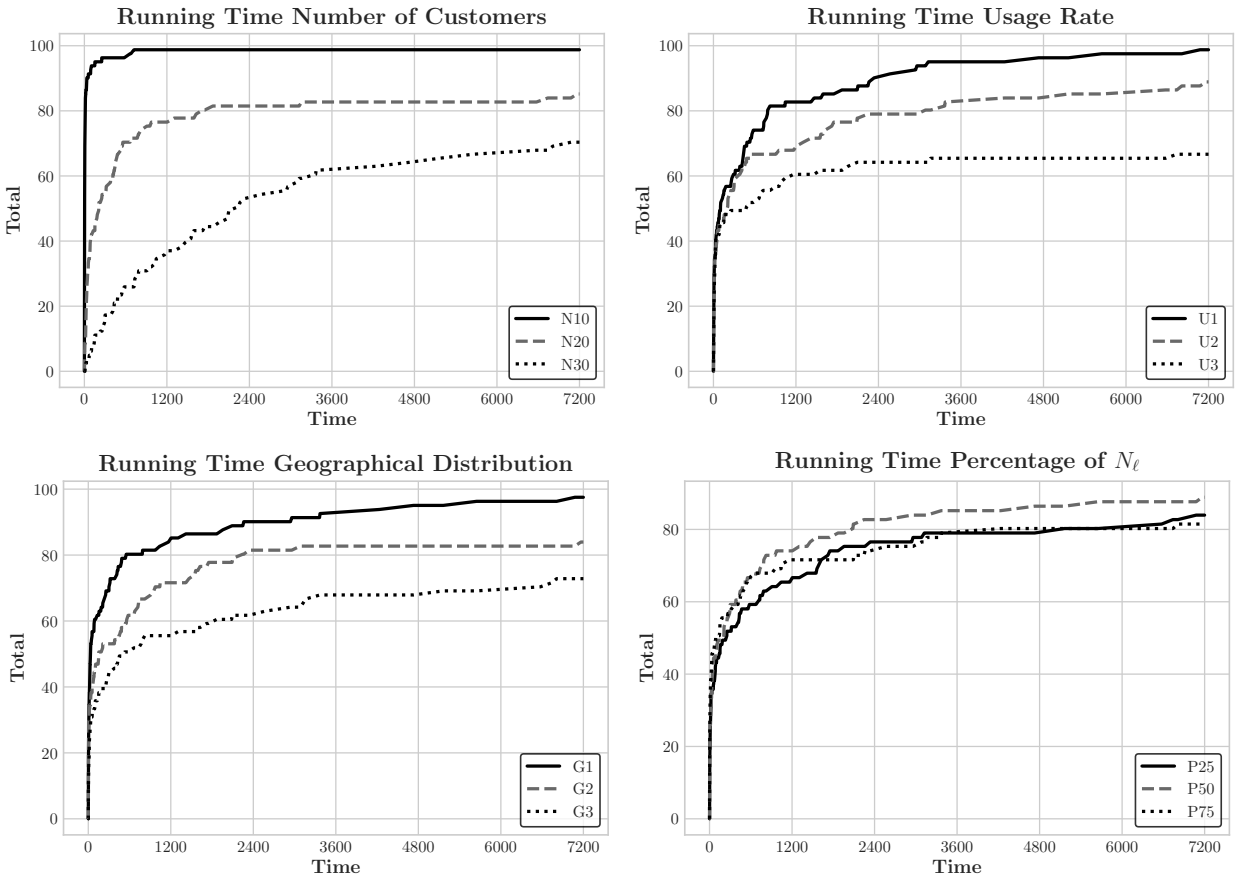

Figure 3: Running Time

with the percentage of instances solved for a given time, from 0 to 7200 seconds (2 hours), is set. The number of customers plot shows curves that are more separated each other, confirming that this dimension is critical in order to explain the difficulty of an instance. Most of the N10 instances are optimally solved in less than 20 minutes; about a 80% of the N20 instances are solved in less than one hour, while 70% of the N30 instances require two hours to be optimally solved. A similar trend is seen for usage and geographical dimensions. Instances with levels U1 and U2 require less solving time than U3 instances, while clustered instances are the fastest to solve for G. When the number of customers N_ℓ varies, the running time of the instance is not affected.

In Figure 4, we plot the optimality gap of the solution found by the DDD algorithm in two hours. The curve corresponds to the percentage of instances whose gap is less than a given value. For the number of customers (N) plot, the only instance with 10 customers, N10, the algorithm does not optimally solve in two hours has a small gap. For N20 instances, about 95% have a gap of less than 5%, and for N30 instances, about 85% have a gap of less than 10%. For the usage rate (U), almost all of the U1 and U2 instances have a gap of less than 8%. The plot also shows that, in general, the U3 instances have larger optimality gaps.

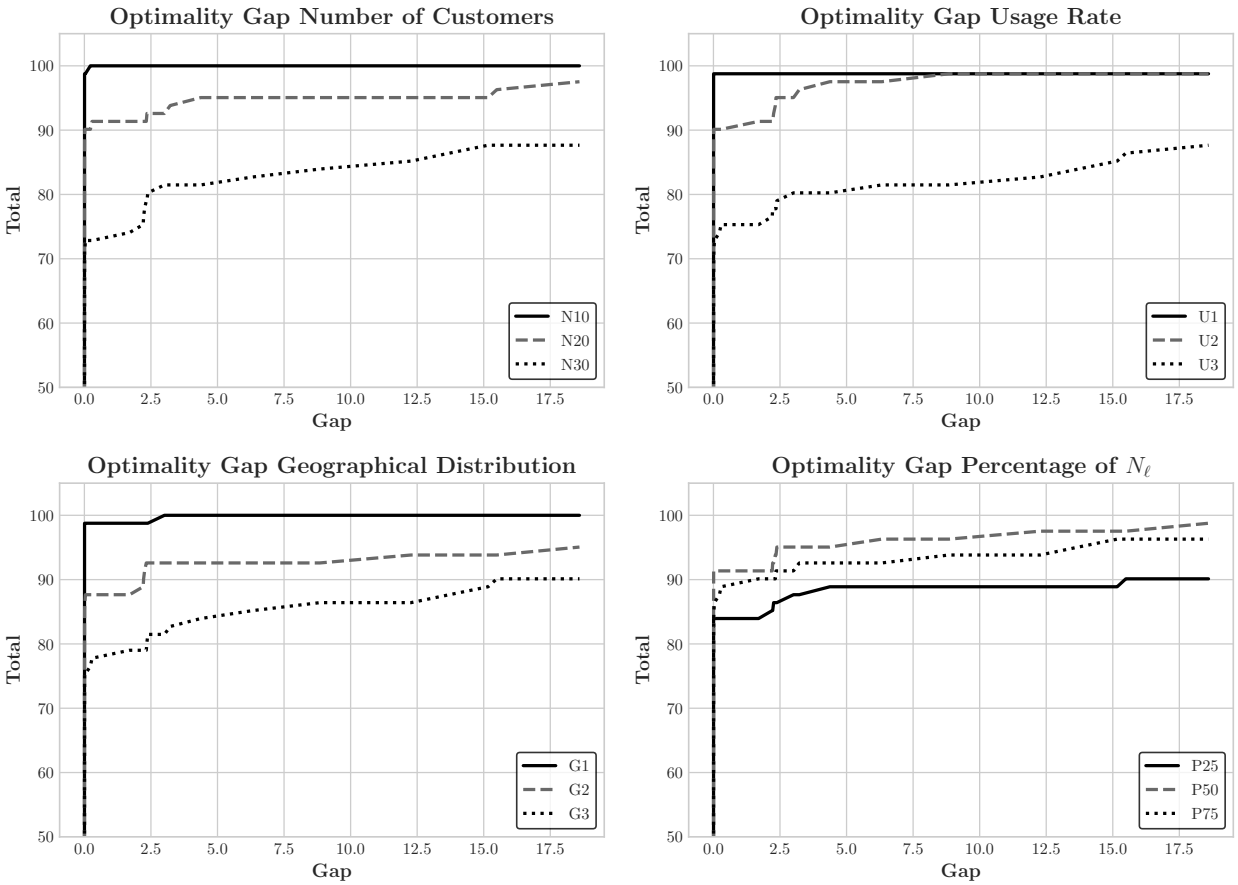

Figure 4: Optimality Gap

In Figure 5, we show the percentage of instances for which the DDD algorithm finishes in less than or equal to a given number of iterations (independent of the status of the solution). We see that a significant percentage of the instances requires more than 50 iterations. In general, the conclusions obtained from previous plots and tables are confirmed when analyzing the number of iterations.

	All	**No Branch**	**No Valid**	**No Visits**	**No Wait**
Optimal	69	69	10	68	66
Feasible	10	9	69	9	11
No Solution	2	3	2	4	4

Table 2: Status for DDD algorithm without strengthenings.

Table 2 shows five different variants of the DDD algorithm, in which a different strengthening mechanism, described in Section 6, is turned off:

- **All**: complete DDD algorithm;
- **No branch**: the branch-and-bound strategy in Appendix B is not used;

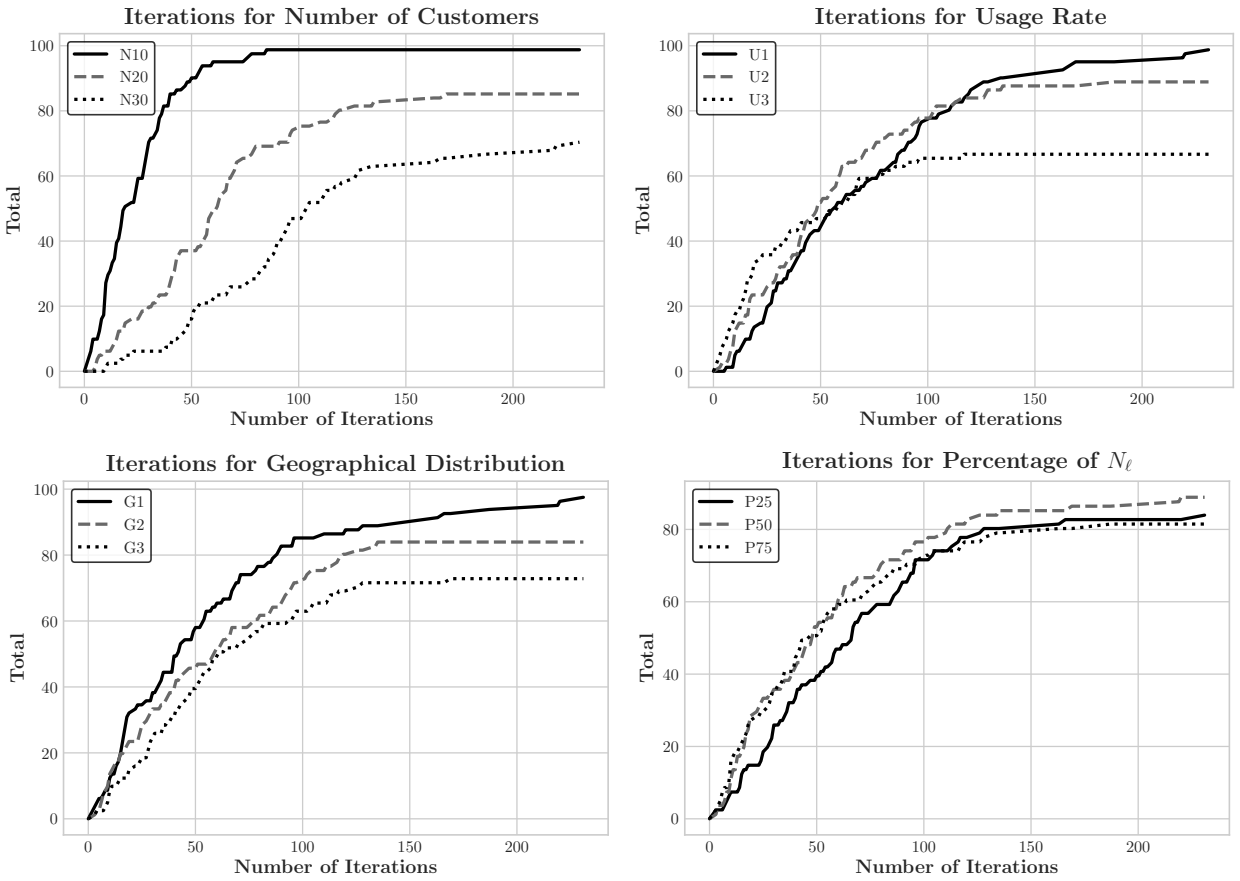

Figure 5: Number of iterations

- **No Valid**: the valid inequalities in Section 6.3 are not included;
- **No Visits**: the conditions in Section 6.2 for visits times in the LBM are not used;
- **No Wait**: the conditions in Section 6.1 for vehicles waiting in the LBM are not included.

For each variant, the number of instances with an optimal, a feasible or no solution status are computed for N20 instances (20 customers) after two hours running. All strengthenings lead to improvement in the algorithm's performance: for all the cases, the full DDD algorithm (All) has more instances with a feasible or an optimal solution when some variant does not. However, when the valid inequalities are turned off, the impact is more significant on the number of instances with optimal solutions than it is in any other case (not in the number of 'No Solution' status).

In Figure 6, we consider the same DDD algorithm variants presented in Table 2 in a running time plot, which shows the percentage of instances solved optimally in a given time. The plot considers the first 20 minutes and validates the previous observation: the valid

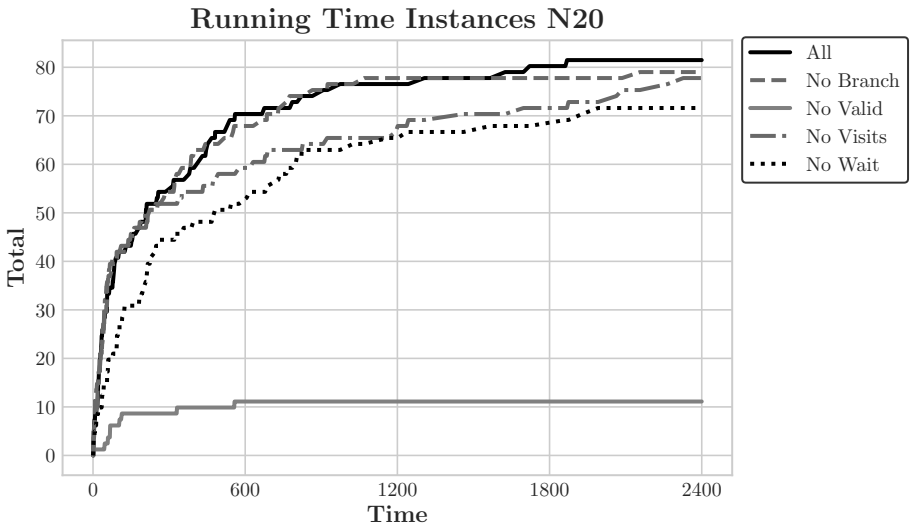

Figure 6: Running time for the DDD algorithm without strengthenings.

inequalities impact the performance the most. We also see that the waiting-vehicles and the visit times conditions (Section 6.1 and 6.2) have a significant effect on the algorithm running times, especially during the first 10 minutes. Figure 6 also indicates that when the Branch-and-Bound strategy is not included, the impact is not so relevant. One explanation for this is that the improvement of this strategy is already implied by other strengthening components.

As an example of the DDD algorithm, we plot the time expanded network for the instance N10G3P75U3-3 (the third sample of type N10G3P75U3) in Figure 7. This instance is solved optimally by the algorithm, and the network is the time discretization corresponding to the last iteration (54 iterations). The crosses represent the time points for each location (0 corresponds to the depot), and the gray lines indicate the time intervals in equations (14) and (15). These intervals are the visit arrival times when the number of visits is the minimum to deliver the customer usage during the time H. We first note that most of the time points are in the intervals, which shows that these intervals are relevant and that the algorithm does not need to fully discretize time in order to solve the problem. For Customer 9, for example, the time interval that is relevant for the (only) visit to the customer is from time 7 to 15. Customer 5 has two visits, one between time 3.5 and 9, another between 10 and 19. The DDD algorithm efficiently and dynamically discretizes the time-expanded network, focusing only on relevant time intervals.

More computational experiments and analysis can be found in Appendix C-D-E.

In Appendix C, we analyze the number of iterations needed for each instance type, showing that instances with 30 customers require more iterations (and, consequently, more time points) than instances with 10 and 20 customers. In Appendix C, we also include tables with the number of instances with optimal, feasible or no solution for each type.

In Appendix D, we compare the DDD algorithm with the approach used in Lagos et al.

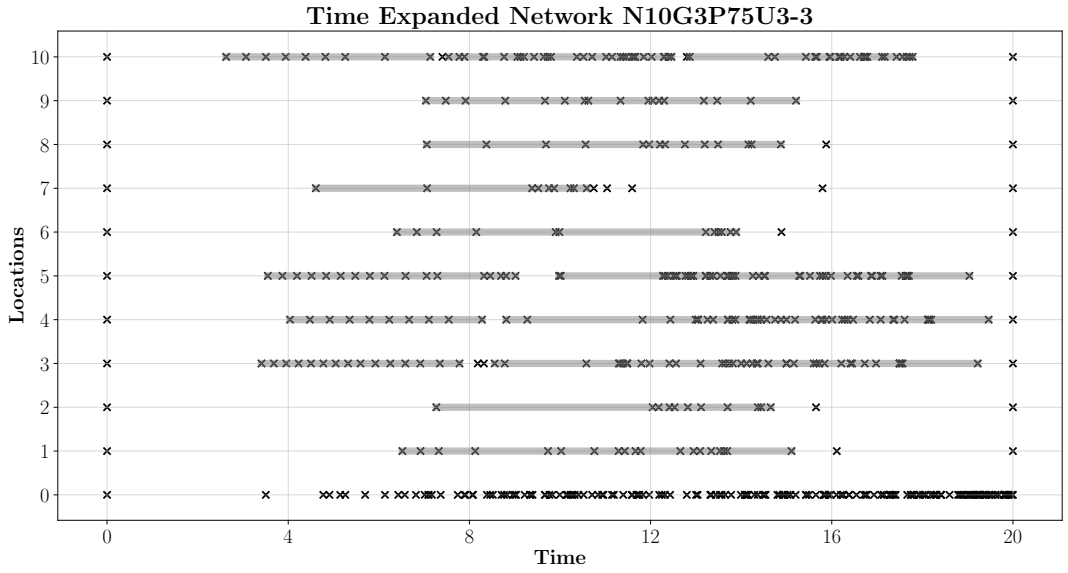

Figure 7: DDD algorithm time discretization example.

(2020) for the CIRP-OB. The results show that the DDD algorithm outperforms the homogeneous time discretization since for all instances, provable optimal solutions are found while, using the approach in Lagos et al. (2020), for several instances the optimally gap is large even after 2 hours of running time.

In Appendix E, we run an extra set of instances in which all customers have a storage capacity less than Q. The results show no significant difference in number of optimal, feasible or no solutions with respect to instances with 25%, 50% or 75% of customers in N_ℓ. We conclude this type of instance is no more difficult than the instances with 25%, 50% or 75% N_ℓ.

10 Discussion and Future Research

In this work, we study a special case of the CIRP, the CIRP-OB, and we propose a new and efficient algorithm for solving it. We successfully implement the DDD algorithm, an algorithm that has been used to solve several continuous time problems, and our computational experiments validate the potential of this algorithm. Out of 243 randomly generated CIRP-OB instances, the DDD algorithm finds a provable optimal solution for 84% of them in less than two hours, and for 95%, the algorithm returns a feasible solution with an optimality gap of less than 18%.

To the best of our knowledge, this is the first time a DDD algorithm has been developed for a continuous time variant of the IRP. Lagos et al. (2020), recently studied the CIRP and developed an algorithm that relies on an IP formulation over a time-expanded network, and can be viewed as a partial implementation of a DDD algorithm. Their methodology used models constructed over homogeneous time discretizations. In this work, we study complex

models and algorithms in non-homogeneous time expanded networks and show the power of using strategies that dynamically discretize time.

However, the DDD algorithm implementation is not straightforward for the CIRP-OB. The LBM formulation presents some challenges that need to be addressed. One is that a very fine time discretization does not lead to an exact formulation for the continuous time problem. In this formulation, the customer storage capacities are always strictly greater than the real capacity C_i, $i \in N$. For other problems in which the DDD algorithm has been studied (e.g. Boland et al. (2017)), this does not happen, and the lower bound model eventually becomes an exact model for the problem. Nevertheless, we overcome this problem by using Theorem 2, which allows us to prove that the DDD algorithm is an exact algorithm for the problem in that the DDD algorithm always returns an optimal solution for a feasible CIRP-OB instance.

In our experiments, we study several instance dimensions: the number of customers (N), the geographical distribution of the locations (G), the percentage of customers in N_ℓ (P), and the usage rate level (U). Of these, variations on the number of customers and the usage rate better explain the complexity of the problem (running time, optimality gap, iterations). We also see that the percentage of customers in N_ℓ (P) does not have a clear impact on the instance difficulty. The DDD algorithm can optimally solve instances with up to 30 customers, random geographical locations, and high usage rates. The algorithm efficiency results, in part, from the fact that the time discretization is more intensive in particular time intervals, so not all of time horizon H is considered.

In this research, we have demonstrated the potential of dynamic discretization discovery algorithms for solving a special class of continuous-time inventory routing problems (in which customers are served on dedicated routes). Some of the ideas presented in this research can be extended to more general continuous-time inventory routing problems. One complication that has to be overcome when solving continuous-time inventory routing problems is that the linear relaxations of time-indexed formulations are weak (as shown in Section 4), so it is critical to exploit both problem and solution structure as much as possible. Several of the ideas presented above can be used with little or no modification for the more general continuous-time inventory routing problem. An natural initial step would be to study a setting in which no more than 2 (possibly 3) customers can be visited on a route.

Finally, incorporating inventory holding costs might be of interest. The structure of optimal solutions might change as it may be beneficial to keep inventories at customers low (to reduce inventory holding costs), but that will require more (smaller) deliveries.

References

C. Archetti, L. Bertazzi, G. Laporte, and M. G. Speranza. A branch-and-cut algorithm for a vendor-managed inventory-routing problem. *Transportation Science*, 41(3):382–391, 2007.

P. Avella, M. Boccia, and L. Wolsey. Single-period cutting planes for inventory routing problems. *Transportation Science*, 52(3):497–508, 2017.

W. Bell, L. Dalberto, M. Fisher, A. Greenfield, R. Jaikumar, P. Kedia, R. Mack, and P. Prutzman. Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces*, 13(6):4–23, 1983.

L. Bertazzi, M. Savelsbergh, and M. G. Speranza. Inventory routing. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 49–72. Springer US, Boston, MA, 2008.

Luca Bertazzi. Analysis of direct shipping policies in an inventory-routing problem with discrete shipping times. *Management Science*, 54(4):748–762, 2008.

N. Boland, M. Hewitt, L. Marshall, and M. Savelsbergh. The continuous-time service network design problem. *Operations Research*, 65(5):1303–1321, 2017.

Natashia Boland, Mike Hewitt, Luke Marshall, and Martin Savelsbergh. The price of discretizing time: a study in service network design. *EURO Journal on Transportation and Logistics*, 8 (2):195–216, 2019.

Natashia L Boland and Martin WP Savelsbergh. Perspectives on integer programming for time-dependent models. *Top*, 27(2):147–173, 2019.

A. Campbell and M.W.P. Savelsbergh. A decomposition approach for the inventory routing problem. *Transportation Science*, 38:488–502, 2004a.

A. Campbell and M.W.P. Savelsbergh. Delivery volume optimization. *Transportation Science*, 38: 210–223, 2004b.

A. Campbell, L. Clarke, A. Kleywegt, and M.W.P. Savelsbergh. The inventory routing problem. In T. Crainic and G. Laport, editors, *Fleet Management and Logistics*, pages 95–113. Kluwer Academic Publishers, Norwell, MA, 1998.

A. Campbell, L. Clarke, and M.W.P. Savelsbergh. Inventory routing in practice. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, pages 309–330. Society for Industrial and Applied Mathematics, monographs on discrete mathematics and applications, Philadelphia, PA, USA, 2002.

L. C. Coelho and G. Laporte. The exact solution of several classes of inventory-routing problems. *Computers & Operations Research*, 40(2):558 – 565, 2013.

L. C. Coelho and G. Laporte. Improved solutions for inventory-routing problems through valid inequalities and input ordering. *International Journal of Production Economics*, 155(C):391 – 397, 2014. Celebrating a century of the economic order quantity model.

L C Coelho, J F Cordeau, and Gilbert Laporte. Thirty years of inventory routing. *Transportation Science*, 48(1):1–19, 2014.

G. Desaulniers, J. Rakke, and L. C. Coelho. A branch-price-and-cut algorithm for the inventory-routing problem. *Transportation Science*, 50:1060–1076, 2015.

Guillermo Gallego and David Simchi-Levi. On the effectiveness of direct shipping strategy for the one-warehouse multi-retailer r-systems. *Management Science*, 36(2):240–243, 1990.

Guillermo Gallego and David Simchi-Levi. Rejoinder to "a note on bounds for direct shipping costs". *Management Science*, 40(10):1393–1393, 1994.

Anton J Kleywegt, Vijay S Nori, and Martin WP Savelsbergh. The stochastic inventory routing problem with direct deliveries. *Transportation Science*, 36(1):94–118, 2002.

Felipe Lagos, Natashia Boland, and Martin Savelsbergh. The continuous-time inventory-routing problem. *Transportation Science*, 54(2):375–399, 2020.

Jianxiang Li, Haoxun Chen, and Feng Chu. Performance evaluation of distribution strategies for the inventory routing problem. *European Journal of Operational Research*, 202(2):412–419, 2010.

J.-H. Song and M. Savelsbergh. Performance measurement for inventory routing. *Transportation Science*, 41(1):44–54, 2007.

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

Appendix A - Mathematical Proofs

Proof of Proposition 1. Reduction for 3-PARTITION. Given an instance of 3-PARTITION, i.e., a set of integer items $S := \{a_1, a_2, \ldots, a_{3n}\}$ with $\sum_{i=1}^{3n} a_i = nB$, for some B positive integer, and $\frac{1}{4}B < a_i < \frac{1}{2}B$ for $i = 1, \ldots, 3n$. The question is whether or not there exist a partition of S into n triplets such that the sum of the number of each partition is equal to B. We construct the following instance of CIRP-OB. The instance has $3n$ customers. Customer $i \in N$ is located at travel time $\tau_i = \frac{1}{2}a_i$ from the depot, has usage rate $u_i = 1$, storage capacity $C_i = B$, and initial inventory $I_i^0 = B(1-\epsilon)$, with $0 < \epsilon < \min_{i\in N}\{a_i/2\}$. There are n vehicles with capacity Q equal to $3B$ and the planning horizon is $H = B$. Each customer needs at least one delivery during the planning horizon, because the usage during the planning horizon is $B > I_i^0/u_i$. Each vehicle can make at most 3 deliveries during the planning horizon, because the time required to make 4 deliveries is strictly greater than $4 \times \min_{i\in N}\{a_i\} > 4 \times \frac{1}{4}B = B = H$, which implies that the maximum number of deliveries that can be made during the planning horizon is $3n$ (there are n vehicles). This implies that each vehicle has to make 3 deliveries. Because $\sum_{i=1}^{3n} a_i = nB$, this implies that each vehicle k must have $\sum_{i\in S_k} a_i = B$, where S_k is the set of customers visited by vehicle k. Thus, a feasible delivery schedule exists if and only if the instance of 3-PARTITION is a *yes*-instance. □

Proof of Proposition 3. We show a feasible solution whose cost is equal to (2).

For all $i \in N$, consider $k = k_1, k_1+1 \ldots, k_2$, with $t_i^{k_1} = \tau_i$ and $t_i^{k_2} = H - C_i/u_i$. Also, let $\bar{k} = \text{argmin}_{k=k_1,k_1+1\ldots,k_2}\{I_i^0 + Q(k+1-k_1) - u_i t_i^k > C_i\}$, and take the following solution:

$$x_{0i}^{\ell} = \begin{cases} 1, & k = k_1, \ldots, \bar{k}-1, \\ \frac{C_i - (I_i^0 + Q(\bar{k}-k_1) - u_i t_i^{\bar{k}})}{Q}, & k = \bar{k}, \\ \frac{\bar{u}_i^k}{Q}, & k = \bar{k}+1, \ldots, k_2, \end{cases} \quad \text{with } \ell \in \delta^-(i, t_i^k).$$

In this solution, each customer $i \in N$ receives full vehicle deliveries at each time t_i^k, $k = k_1, \ldots, \bar{k}$, until the inventory is equal to C_i and then, after time $t_i^{\bar{k}+1}$, the deliveries are equal to what is consumed in a time interval $\bar{u}_i^k$. In this solution there is no waiting, thus all variables x_{ii}^k are equal to zero. Constraints (1a)-(1b) are satisfied because we have constructed the x variables as a flow. The number of vehicles constraint (1c) is guaranteed by the number m; the maximum number of vehicles a customer $i \in N$ needs in this solution is at most $\lceil (C_i - I_i^0 + 2\tau_i u_i)/Q \rceil$, corresponding to the product usage while a vehicle is traveling and the initial product needed to fill the inventory level, $C_i - I_i^0$.

Note that in this solution the inventory levels are feasible for all $i \in N$ and $k = k_1, \ldots, k_2$,

$$z_i^k = \begin{cases} z_i^{k-1} + Q - \bar{u}_i^k, & k = k_1, \ldots, \bar{k}-1, \\ C_i, & k = \bar{k}, \ldots, k_2, \end{cases}$$

and, since it also holds that $t_i^{k_2} = H - C_i^0/u_i$, inventory levels are feasible for all planning horizon.

We conclude this solution is feasible and its total cost is exactly (2),

$$\begin{aligned}\sum_{\ell\in\mathcal{T}_0} x^{\ell}_{0i} &= (k_1 - \bar{k}) + \frac{C_i - I_i^0 - Q(\bar{k} - k_1) - u_i t_i^{\bar{k}}}{Q} + \frac{u_i(t_i^{k_2} - t_i^{\bar{k}})}{Q},\\ &= \frac{C_i - I_i^0}{Q} + \frac{u_i t_i^{k_2}}{Q},\\ &= \frac{C_i - I_i^0}{Q} + \frac{Hu_i - C_i}{Q},\\ &= \frac{Hu_i - I_i^0}{Q}.\end{aligned}$$

□

Proof of Proposition 4. For contradiction, consider any MDO solution where a vehicle $v \in \{1,\dots,m\}$ makes a delivery and the inventory level is strictly less than C_i, $i \in N$. Let I_i be the customer inventory level before the delivery. Let $0 < q < C_i - I_i$ be the delivered quantity and let $r > 0$ be the remaining quantity at the vehicle immediately after the delivery. If $I_i + q + r \le C_i$, then the vehicle can deliver $q + r$ and go back to the depot. This is a contradiction; the solution cost is the same, so it maintains optimality condition, and the depot-time set of the vehicle v, S_0^v, is superset since the vehicle does not wait at the customer location. If $I_i + q + r > C_i$, then we can make the vehicle deliver $C_i - I_i$ and wait at the customer location. The remaining quantity at the vehicle is less than r and so the vehicle can return at time $t + \frac{q+r-C_i+I_i}{u_i} < t + \frac{r}{u_i}$, with u_i the customer usage rate and t the vehicle delivery time. The cost is the same and the S_0^v is also superset, so again, this is a contradiction. □

Proof of Proposition 5. Let $[t_1, t_2]$ be the time interval in which a vehicle $v \in \{1,\dots,m\}$ is waiting at the customer location of an MDO solution. Let I_i be the inventory level at time t_1 before delivery and let u_i be the usage rate, $i \in N$. For contradiction, assume I_i is positive. Then, the vehicle can arrive at time $t = \min\{I_i/u_i + t_1, t_2\}$, instead of t_1, and the solution is still feasible. If $t = t_2$, then there is no waiting; otherwise, the vehicle can deliver C_i quantity (the maximum quantity possible to be delivered at time t), and the depot-time set S_0^v is superset, since the vehicle v stays at the depot longer. The solution is feasible and the cost remains the same, reaching the contradiction. □

Proof of Proposition 6. For contradiction assume that the waiting time of a vehicle $v \in \{1,\dots,m\}$ is strictly greater than $\frac{\max\{Q-C_i,0\}}{u_i}$ for an MDO solution. By Proposition 5, we know the inventory is zero when the vehicle v arrives. If $Q \le C_i$, the vehicle can deliver Q and immediately after return to the depot. The waiting time at the customer is zero so we get the contradiction. If $Q > C_i$, since the inventory is zero, the vehicle can deliver C_i. Then it can stay until the customer consumes the difference $Q - C_i$ and return to the depot. In this case, the total waiting time is exactly $\frac{Q-C_i}{u_i}$, leading to the contradiction. □

Proof of Proposition 10. Note that if the arriving vehicle waits after the delivery, then by Proposition 4, the delivery must be equal to $C_i - I_i$. Consider a vehicle that does not wait after the delivery in an MDO solution. If the vehicle delivers the maximum quantity possible, in this case the minimum between the vehicle capacity Q and the remaining customer capacity $C_i - I_i$, then the solution cost is the same and the depot-time set of the solution is at least the same. Indeed, since the maximum possible quantity was delivered by the vehicle, any other vehicle $v \in \{1, \ldots, m\}$ that makes a delivery after can save time at the customer; less product must be delivered. Then, the vehicle v can potentially stay at the depot longer, so the set S_0^v is superset. □

Proof of Proposition 7. First we prove the intervals are correct. We know that $Q\eta_i \geq Hu_i - I_i^0$, so summing kQ both sides with $k = 1, \ldots, \eta_i$, and noticing that $Q \leq C_i$, we get,

$$\begin{aligned} Hu_i - I_i^0 + kQ &\leq Q(\eta_i + k), \\ Hu_i - (\eta_i - k)Q &\leq I_i^0 + kQ, \\ Hu_i - (\eta_i - k)Q - C_i &\leq I_i^0 + (k-1)Q. \end{aligned}$$

We prove that the visit times must be in the intervals. For contradiction, suppose there is a visit $k = 1, \ldots, \eta_i$ whose time s_k is not in the interval we defined. If the time $s_k > \frac{I_i^0 + (k-1)Q}{u_i}$, then the solution is not feasible since the maximum product delivered by $k-1$ visits is $(k-1)Q$, so at time s_k the inventory level is negative. Then, it must be $s_k < \frac{Hu_i - C_i - (\eta_i - k)Q}{u_i}$. Let q_ℓ be the total delivery done by the vehicle at visit $\ell = 1, \ldots, \eta_i$. The total delivery $\sum_{\ell=1}^{k} q_\ell$ up to time s_k satisfies,

$$\sum_{\ell=1}^{k} q_\ell \leq u_i s_k + C_i - I_i^0,$$

since for customer i no waiting is allowed (Proposition 6). The total delivery by all visits holds,

$$\begin{aligned} \sum_{\ell=1}^{\eta_i} q_\ell &= \sum_{\ell=1}^{k} q_\ell + \sum_{\ell=k+1}^{\eta_i} q_\ell \\ &\leq \sum_{\ell=1}^{k} q_\ell + (\eta_i - k)Q \\ &\leq u_i s_k + C_i - I_i^0 + (\eta_i - k)Q, \\ &< Hu_i - C_i - (\eta_i - k)Q + C_i - I_i^0 + (\eta_i - k)Q, \\ &= Hu_i - I_i^0, \end{aligned}$$

but this contradicts the fact the solution is feasible. □

Proof of Proposition 8. This proof is similar to Proposition 7's proof.

First we prove the intervals are correct. We know that $Q\eta_i \geq Hu_i - I_i^0$, so summing $(k-1)Q$ both sides with $k = 1, \ldots, \eta_i$, and noticing that $I_i^0 - C_i \leq 0$, we get,

$$\begin{aligned} Hu_i - I_i^0 + (k-1)Q &\leq Q(\eta_i + k - 1), \\ Hu_i - (\eta_i - k + 1)Q &\leq I_i^0 + (k-1)Q, \\ Hu_i - (\eta_i - k + 1)Q - C_i + I_i^0 &\leq I_i^0 + (k-1)Q. \end{aligned}$$

We prove that the visit times must be in the intervals. For contradiction, suppose there is a visit $k = 1, \ldots, \eta_i$ whose time s_k is not in the interval we defined. The case $s_k > \frac{I_i^0 + (k-1)Q}{u_i}$ is equivalent to 7's proof. Let q_ℓ be the total delivery done by the vehicle at visit $\ell = 1, \ldots, \eta_i$. Suppose the time s_k is before $\frac{Hu_i - C_i + I_i^0 - (\eta_i - k + 1)Q}{u_i}$. There are two cases, the vehicle during the visit kth waits or it does not. If it waits, then the inventory level before the delivery at the arrival time is zero (Proposition 5), so it must be $\sum_{\ell=1}^{k-1} q_\ell = u_i s_k - I_i^0$. In this case, the total delivery up to time s_k, $\sum_{\ell=1}^{k} q_\ell$, is at most $u_i s_k - I_i^0 + Q$, then,

$$\begin{aligned} \sum_{\ell=1}^{\eta_i} q_\ell &\leq u_i s_k - I_i^0 + Q + (\eta_i - k)Q, \\ &< Hu_i - C_i + I_i^0 - (\eta_i - k + 1)Q - I_i^0 + Q + (\eta_i - k)Q, \\ &= Hu_i - C_i, \\ &\leq Hu_i - I_i^0, \end{aligned}$$

which is a contradiction.

Let I_i^k be the inventory intermediately before the kth delivery. If the vehicle does not wait during the kth visit then, the inventory level after the delivery at time s_k is C_i (Proposition 10), so the q_k delivery is $C_i - I_i^k$ and also,

$$\begin{aligned} \sum_{\ell=1}^{k} q_\ell &= \sum_{\ell=1}^{k-1} q_\ell + q_k, \\ &= I_i^k - I_i^0 + s_k u_i + C_i - I_i^k, \\ &= s_k u_i + C_i - I_i^0. \end{aligned}$$

Thus, the total delivery satisfies,

$$\begin{aligned} \sum_{\ell=1}^{\eta_i} q_\ell &\leq s_k u_i + C_i - I_i^0 + (\eta_i - k)Q, \\ &< Hu_i - C_i + I_i^0 - (\eta_i - k + 1)Q + C_i - I_i^0 + (\eta_i - k)Q, \\ &= Hu_i - Q, \\ &< Hu_i - C_i, \\ &\leq Hu_i - I_i^0. \end{aligned}$$

leading to the contradiction. □

Proof of Proposition 9. For contradiction, assume the solution is MDO and all intervals have at least one visit. Consider the following procedure: for each interval keep the latest visit only by removing all the visits before within the same interval. Since each interval $k = 1, \ldots, \eta_i$ is defined from time $s_1^k = \frac{kQ-(C_i-I_i^0)}{u_i}$, the visit we left is guaranteed to delivery Q product. Indeed, the visit time t is greater than or equal to s_1^k thus the delivery q_k in this visit is the minimum between the capacity Q and the difference between the inventory before the visit and the capacity C_i,

$$\begin{aligned} q_k &= \min\{C_i - (tu_i - I_i^0 - (k-1)Q), Q\}, \\ &\geq \min\{C_i - (kQ - (C_i - I_i^0) - I_i^0 - (k-1)Q), Q\}, \\ &= Q. \end{aligned}$$

For each interval k the inventory is non-negative at the moment of the vehicle arrival t. Indeed, for each $\ell = 1, \ldots, k-1$ the delivery is Q, then the inventory is non-negative at $t \in [s_1^k, s_2^k]$, with $s_2^k = \frac{I_i^0+(k-1)Q}{u_i}$,

$$\begin{aligned} I_i^0 + (k-1)Q - tu_i &\geq I_i^0 + (k-1)Q - s_2^k, \\ &\geq I_i^0 + (k-1)Q - I_i^0 - (k-1)Q, \\ &\geq 0. \end{aligned}$$

We conclude the solution we propose is feasible and its cost is less than the MDO solution cost, reaching the contradiction. □

Proof of Proposition 12. Let's first consider the case with a single vehicle waiting at a customer location $i \in N$. Then at some time t_i^k, $k \in \mathcal{T}_i$, we have $x_{ii}^k = 1$. This means that for any CIRP-OB feasible solution mapped to this LBM solution there must be a vehicle waiting from some time t in $[t_i^k, t_i^{k+1})$ to some time t' in $[t_i^{k+1}, t_i^{k+2})$. Given that $\bar{u}_i^{k+1} \leq C_i$, we can assume w.l.o.g. that there are at most two deliveries in $[t, t_i^{k+1}]$, one at time t and another at time t_i^{k+1} (we can consolidate small deliveries in that interval). The Proposition 4 holds that at any time after a delivery the inventory level must be equal to C_i, in particular, this is true at t. In the LBM, the time t maps to t_i^k so the inventory at this time must be equal to C_i.

If $x_{ii}^k > 1$ for the LBM solution, then more than vehicle is waiting at the customer $i \in N$ location, but there is at most one doing deliveries, since the LBM is mapped from a feasible CIRP-OB solution and at most one vehicle can deliver at the same time. For the vehicle that is doing deliveries, using the previous result, every time x_{ii}^k is positive, the inventory must be equal to C_i. □

Proof of Proposition 13. Consider a customer $i \in N_\ell$. Every CIRP-OB feasible solution satisfies that no more than one vehicle is waiting at some customer location. If the LBM solution mapped from this solution has more than one vehicles are waiting at i at time t_i^{k-1}, $k \in \mathcal{T}_i$, means that at least $v - 1$, with $v = x_{ii}^{k-1} > 1$, vehicles make a delivery at some

time in $[t_i^k, H]$ and no delivery before time t_i^k. Assume those $v-1$ vehicles depart at time $t_0^{\ell'}$ from the depot, then we can re-route them from time $t_0^{\ell'}$ to time t_0^{ℓ}, arriving at time t_i^k to customer i. The time t_0^{ℓ} and travel time τ_{0i}^{ℓ} guarantee this new itinerary is feasible. We keep the same itinerary after this time t_i^k. Therefore the constraint $x_{ii}^{k-1} \leq 1$ is valid.

Consider a customer $i \in N_g$. A CIRP-OB feasible solution satisfies that waiting vehicles is not allowed for i. Similar to the case above, if $v = x_{ii}^{k-1} > 0$ vehicles are waiting we can re-route from time $t_0^{\ell'}$ to time t_0^{ℓ}, arriving at time t_i^k to customer i, so the constraint $x_{ii}^{k-1} = 0$ is valid. □

Proof of Proposition 14. From Proposition 13, we know that the number of vehicles waiting from some time in $[t_i^k, t_i^{k+1})$ to some time in $[t_i^{k+1}, t_i^{k+2})$ is at most one, since the depot time discretization has a time $t_0^{\ell_2}$ such that $t_0^{\ell_2} + \tau_{0i}^{\ell_2} = t_i^{k+1}$. Consider any optimal solution to the CIPR-OB, then by Proposition 5, any vehicle that arrives at some time in $[t_i^k, t_i^{k+1})$ and then waits, must get there when the customer is zero level inventory. This means that any LBM optimal solution in which a vehicle visits a customer at time t_i^k and then waits, the customer can have at most $\bar{u}_i^{k+1}$ units in inventory. □

Proof of Proposition 15. For any arrival time during the interval $[t_i^{k_1}, t_i^{k_1+1})$ the departure time cannot be later than $t_i^{k_1+1} + (Q - C_i)/u_i$ since the waiting time is at most $(Q - C_i)/u_i$ for customers $i \in N_\ell$ (Proposition 6). Thus, a vehicle can wait at most from time $t_i^{k_1}$ to $t_i^{k_2}$ in the LBM. □

Proof of Proposition 16. For a customer $i \in N$, let k_1 and k_2 be any two indexes in $\mathcal{T}_i$ such that $k_1 < k_2$. At time $t_i^{k_2}$, we know the inventory level is bounded from below as follows in the LBM,

$$z_i^{k_2} \geq \bar{u}_i^{k_2+1} = u_i(t_i^{k_2+1} - t_i^{k_2}).$$

Also, summing up the inventory balance constraints from time $t_i^{k_1}$ to $t_i^{k_2}$, we have,

$$z_i^{k_2} = z_i^{k_1-1} + \sum_{k=k_1}^{k_2} y_i^k - u_i(t_i^{k_2} - t_i^{k_1-1}),$$

so using the equality for $z_i^{k_2}$ and the usage inequality for $z_i^{k_2}$, we get,

$$\sum_{k=k_1}^{k_2} y_i^k \geq u_i(t_i^{k_2+1} - t_i^{k_1-1}) - z_i^{k_1-1},$$

and noticing that at time $t_i^{k_1-1}$ the inventory level is bounded from above by $C_i + u_i(t_i^{k_1} - t_i^{k_1-1})$, we finally have,

$$\sum_{k=k_1}^{k_2} y_i^k \geq u_i(t_i^{k_2+1} - t_i^{k_1-1}) - (C_i + u_i(t_i^{k_1} - t_i^{k_1-1})),$$

$$\sum_{k=k_1}^{k_2} y_i^k \geq u_i(t_i^{k_2+1} - t_i^{k_1}) - C_i,$$

which is a valid inequality for any customer $i \in N$. We can use this expression in order to get (14) and (15) by deriving an inequality for y and x variables. For customers $i \in N_g$, there is no waiting, so all the product that gets to a customer must have arrival time at the interval $[t_i^{k_1}, t_i^{k_2}]$, so,

$$\sum_{k=k_1}^{k_2} y_i^k = \sum_{k=k_1}^{k_2} \sum_{(0,\ell)\in\delta^-(i,t_i^k)} w_{i0}^\ell \leq Q \sum_{k=k_1}^{k_2} \sum_{(0,\ell)\in\delta^-(i,t_i^k)} x_{i0}^\ell,$$

then we have,

$$\frac{u_i(t_i^{k_2+1} - t_i^{k_1}) - C_i}{Q} \leq \sum_{k=k_1}^{k_2} \sum_{(0,\ell)\in\delta^-(i,t_i^k)} x_{i0}^\ell.$$

For customers $i \in N_\ell$, waiting is allowed, so some product can come from a vehicle waiting from the previous interval. We have,

$$\sum_{k=k_1}^{k_2} y_i^k = w_{ii}^{k_1-1} - w_{ii}^{k_2} + \sum_{k=k_1}^{k_2} \sum_{(0,\ell)\in\delta^-(i,t_i^k)} w_{i0}^\ell \leq Q x_{ii}^{k_1-1} + Q \sum_{k=k_1}^{k_2} \sum_{(0,\ell)\in\delta^-(i,t_i^k)} x_{i0}^\ell,$$

so we conclude,

$$\frac{u_i(t_i^{k_2+1} - t_i^{k_1}) - C_i}{Q} \leq x_{ii}^{k_1-1} + \sum_{k=k_1}^{k_2} \sum_{(0,\ell)\in\delta^-(i,t_i^k)} x_{i0}^\ell.$$

□

Proof of Theorem 2. For contradiction, we assume the following,

> **(A)**: *There is no $\Delta > 0$ and there is no times $\omega_{i,j}^k(\Delta)$ and $v_{i,j}^k(\Delta)$ such that the number of visits η_i is the same and $\omega_{i,j}^k(\Delta) = v_{i,j}^k(\Delta)$, for all $i \in N_0$, $j = a, d$, $k = 1, \ldots, \eta_i$.*

Take any $\Delta > 0$ such that for all $i \in N$, $\tau_i/\Delta \in \mathbb{Z}$ and $H/\Delta \in \mathbb{Z}$ (data is rational, so there is a Δ that satisfies those conditions). Let $\Delta_\ell = \Delta 2^{-\ell}$, $\ell \geq 0$, be an infinite sequence. Note that for all $\ell \geq 0$, in the LBM$^+(\Delta_\ell)$, the number of visits to a customer $i \in N$ is bounded. Indeed, since for all $\ell \geq 0$, it also holds $\tau_i/\Delta_\ell \in \mathbb{Z}$, $H/\Delta_\ell \in \mathbb{Z}$, then $\frac{\lceil H/\Delta_\ell \rceil}{2\lfloor \tau_i/\Delta_\ell \rfloor} \leq \frac{H}{2\tau_i}$ for all $i \in N$.

Let $\eta(\Delta_\ell) = [\eta_1(\Delta_\ell), \ldots, \eta_{|N|}(\Delta_\ell)]$ be a vector of number of visits to customers of an optimal solution to LBM$^+(\Delta_\ell)$, $\ell \geq 0$. Since the number of visits to each customer is bounded, the set $\{\eta(\Delta_\ell)\}_{\ell\in\mathcal{L}}$ is finite. Also, there are a finite number of feasible itineraries (i.e., the assignment of those visits to vehicles) for those visits. Then, we can take an infinite subsequence $\{\Delta_k\}_{k\in\mathcal{K}}$, with $\mathcal{K} \subseteq \mathbb{Z}_+$, such that there is an optimal solution to the LBM$^+(\Delta_k)$ whose number of visits is the same $\bar{\eta} = \eta(\Delta_k)$ and those visits occur in the same sequence, for all $k \in \mathcal{K}$. For each LBM$^+(\Delta_k)$, $k \in \mathcal{K}$, we consider an optimal solution (only one) whose

number of visits $\bar{\eta}$ and the itineraries are the same, so we refer to *the* optimal solution to LBM$^+(\Delta_k)$.

Note that the number of visits given by $\bar{\eta}$ does not induce a feasible solution to LBM(Δ'), for all $\Delta' > 0$, because any feasible solution to the LBM with inventory capacity C_i is a feasible solution to the LBM with inventory capacity $C_i + u_i\Delta'$. This contradicts **(A)**, since the number of visits is optimal for both LBM(Δ') and LBM$^+(\Delta')$ and the visit arrival and departure times are the same.

Let $f_i^\ell(\Delta_k)$ be the total delivery to $i \in N$ during the ℓth visit of the LBM$^+(\Delta_k)$ optimal solution, $k \in \mathcal{K}$, $\ell = 1, \ldots, \bar{\eta}_i$. Similarly, let $z_i^\ell(\Delta_k)$ be the inventory level before the delivery ℓth. Define,

$$\bar{f}_i^\ell = \lim_{k\to\infty} f_i^\ell(\Delta_k),$$
$$\bar{z}_i^\ell = \lim_{k\to\infty} z_i^\ell(\Delta_k).$$

Note that $\bar{f}_i^\ell$ and $\bar{z}_i^\ell$ represent feasible deliveries and inventory levels (in the limit there is no extra inventory capacity).

We also define $\bar{v}_i^\ell = \lim_{k\to\infty} v_{i1}^\ell(\Delta_k)$, and $\bar{w}_i^\ell = \lim_{k\to\infty} v_{i2}^\ell(\Delta_k)$, $\ell = 1, \ldots, \bar{\eta}_i$, $i \in N$.

There must be a $i \in N$ and a $\ell = 1, \ldots, \bar{\eta}_i$ such that either $\bar{v}_i^\ell$ or $\bar{w}_i^\ell$ is irrational. If not, then all arrival and departure times are rational, so there must exist a $\Delta' > 0$ such that Δ' is divisor for all times. The solution $(\bar{f}_i^\ell, \bar{z}_i^\ell, \bar{v}_i^\ell, \bar{w}_i^\ell)$ represents a feasible solution to LBM(Δ'), but this is a contradiction: we know there is no feasible solution to LBM(Δ') with number of visits given by $\bar{\eta}$, for all $\Delta' > 0$.

Using the number of visits $\bar{\eta}$ and the itineraries of LBM$^+(\Delta_k)$ optimal solutions, $k \in \mathcal{K}$, we can construct a polytope. Let v_i^ℓ be the ℓth visit arrival time and let w_i^ℓ be the ℓth visit departure time, $i \in N$. Let z_i^ℓ be the inventory level immediately before the ℓth visit and let f_i^ℓ the total delivery made by the ℓth visit, $i \in N$. Let $s(i,\ell) = (j,k) \in N \times \{1, \ldots, \bar{\eta}_j\}$ be the customer j and the visit kth that goes after the ℓth visit to i in the same vehicle itinerary, $i \in N$, $\ell = 1, \ldots, \bar{\eta}_i$. The polytope for $(v_i^\ell, w_i^\ell, f_i^\ell, z_i^\ell)$ variables is as follows,

$$w_i^{\ell-1} \le v_i^\ell, \qquad i \in N, \ell = 2, \ldots, \bar{\eta}_i, \tag{20a}$$
$$z_i^1 = I_i^0 - u_i v_i^1, \qquad i \in N, \tag{20b}$$
$$z_i^\ell = z_i^{\ell-1} + f_i^\ell - u_i(v_i^\ell - v_i^{\ell-1}), \qquad i \in N, \ell = 2, \ldots, \bar{\eta}_i, \tag{20c}$$
$$z_i^{\bar{\eta}_i} + f_i^{\bar{\eta}_i} \ge u_i(H - v_i^{\bar{\eta}_i}), \qquad i \in N, \tag{20d}$$
$$z_i^\ell + f_i^\ell \le C_i + u_i(w_i^\ell - v_i^\ell), \qquad i \in N, \ell = 2, \ldots, \bar{\eta}_i, \tag{20e}$$
$$v_i^\ell \ge w_j^k + \tau_i + \tau_j, \qquad i \in N, \ell = 2, \ldots, \bar{\eta}_i, (j,k) = s(i,\ell), \tag{20f}$$
$$\tau_i \le v_i^\ell \le w_i^\ell \le H - \tau_i, \qquad i \in N, \ell = 2, \ldots, \bar{\eta}_i, \tag{20g}$$
$$0 \le f_i^\ell \le Q, \qquad i \in N, \ell = 2, \ldots, \bar{\eta}_i, \tag{20h}$$
$$z_i^\ell \ge 0, \qquad i \in N, \ell = 2, \ldots, \bar{\eta}_i.$$

Constraints (20a) ensure that the visit times at customers occur sequentially. Constraints (20b) and (20c) set the inventory levels just prior to each visit at a customer. Constraints

(20d) ensure that inventory after the last delivery is sufficient to meet demand until the end of the planning horizon. Constraints (20e) enforce that the vehicle remains long enough at the customer to deliver its whole load, while not violating the capacity limit. Constraints (20f) ensure that for each of the vehicles the visit times at customers properly account for travel times between locations, and, in case a vehicle performs multiple routes, properly account for travel times to and from the depot in between consecutive routes in its itinerary. Constraints (20g) impose bounds on arrival and departure times. Finally, constraints (20h) enforce deliveries cannot be more than the vehicle capacity Q.

Note that there cannot be a rational solution for the previous polytope (20) since that solution would be feasible for LBM(Δ'), for some $\Delta' > 0$, which again, contradicts **(A)**. However, the solution $(\bar{f}_i^\ell, \bar{z}_i^\ell, \bar{v}_i^\ell, \bar{w}_i^\ell)$ is a feasible solution to that polytope, so it is not empty. Therefore, a non-empty polytope with rational data does not have a rational point (solution), reaching a contradiction.

□

Proof of Proposition 18. We prove that the following solution is feasible: visit each customer $i \in N$ at any possible time delivering as much as possible. Since the number of vehicles is unrestricted, we need to check if the inventory level are non-negative during the all time horizon for all customers. In particular, we show that for all $i \in N$ the inventory level is non-negative at time $\Delta\lceil I_i^0/(\Delta u_i)\rceil$ and the inventory is enough to cover the time $H - \Delta\lceil \tau_i/\Delta\rceil$ to return to the depot.

For each $i \in N$, the first visit time in the solution is at Δk_1, with $k_1 = \lceil \tau_i/\Delta\rceil$ and the last visit time is at Δk_2, with $k_2 = \lfloor H/\Delta\rfloor - k_1$. Note that (18) guarantees that,

$$k_2 - k_1 \geq \left\lceil \frac{C_i}{Q} \right\rceil.$$

Also, at time Δk with $k = \left\lfloor \frac{I_i^0}{u_i\Delta} \right\rfloor$, the inventory level z_i^k is equal to C_i since the total delivery up to k is at least $k\Delta u_i$. Indeed, using (17) we have,

$$k - k_1 = \left\lfloor \frac{I_i^0}{u_i\Delta} \right\rfloor - \left\lceil \frac{\tau_i}{\Delta} \right\rceil \geq \left\lceil \frac{C_i}{Q} \right\rceil,$$

so,

$$(k - k_1)Q \geq C_i \geq k\Delta u_i.$$

The inventory level after time $k\Delta$ is also C_i and the delivery of each visit is at most Q,

$$\Delta u_i \leq \frac{I_i^0}{\left\lceil \frac{C_i}{Q} \right\rceil + \left\lceil \frac{\tau_i}{\Delta} \right\rceil} \leq \frac{C_i}{\left\lceil \frac{C_i}{Q} \right\rceil + \left\lceil \frac{\tau_i}{\Delta} \right\rceil} < \frac{C_i}{\left\lceil \frac{C_i}{Q} \right\rceil} \leq Q.$$

Finally, at time Δk_2 the inventory is C_i and it is enough to cover the customer usage up to time H,

$$\begin{aligned} H - \Delta k_2 &\leq \Delta + \left\lceil \frac{\tau_i}{\Delta} \right\rceil \Delta, \\ &\leq \Delta \left(\left\lceil \frac{C_i}{Q} \right\rceil + \left\lceil \frac{\tau_i}{\Delta} \right\rceil \right), \\ &\leq \frac{I_i^0}{u_i}, \\ &\leq \frac{C_i}{u_i}, \end{aligned}$$

so the product consumption while the last vehicle returns to the depot satisfies $u_i(H-\Delta k_2) \leq C_i$. □

Proof of Proposition 20. We prove that for any $\epsilon > 0$ tolerance, a finite number of time points are added by Algorithms 1, 2 and 3, so the algorithm stops in a finite number of iterations. We assume the initial time discretization is $\{s_{0,i}^k\}_{i \in N_0, k \in \mathcal{T}_i}$.

Since travel times are rational, there exists a $\eta > 0$ such that $\tau_i/\eta \in \mathbb{Z}$, for all $i \in N$. This η guarantees that starting at any time t_i^k, $k \in \mathcal{T}_i$, the arriving times to other locations $j \neq i$, are at times $t_i^k + p\eta$, with $p \in \mathbb{Z}_+$. At a given iteration, the Algorithm 1 can correct the travel time starting from an initial time point, $s_{0,i}^k$; or from a time point added by Algorithm 1, $s_{1,i}^k$; or from a time point added by Algorithm 2, $s_{2,i}^k$; $i \in N$, $k \in \mathcal{T}_i$; or from a time point added by Algorithm 3, $s_{3,i}^k$; $i \in N$, $k \in \mathcal{T}_i$. For each time $s_{0,i}^k$, $s_{1,i}^k$, $s_{2,i}^k$ and $s_{3,i}^k$ we consider all possible new times can be added starting from that time, i.e., $|N_0|\lceil H/\eta \rceil$ points. However, note that times $s_{1,i}^k$ don't need to be counted, since they are already considered for some time point t_j^ℓ, $j \in N_0$, $\ell \in \mathcal{T}_j$, from where $s_{1,i}^k$ was generated. Using Lemma 1 at most $|N|\lceil 2H/\epsilon \rceil$ time points are added by Algorithm 2 and by Lemma 2 at most $2|N|\lceil 2H/\epsilon \rceil$, so no more than $\left(3|N|\lceil 2H/\epsilon \rceil + |\{s_i^k\}_{i \in N_0, k \in \mathcal{T}_i}|\right) \times |N_0|\lceil H/\eta \rceil$ are added by Algorithm 4 for a given ϵ. □

Proof of Theorem 3. For any feasible instance of CIRP-OB there exists a rational solution, then there must be a $\epsilon_1 > 0$ such that for any two consecutive visits to a customer $i \in N$, the time between those visits is at least ϵ_1. In addition, by Theorem 2, there exists a $\epsilon_2 > 0$ such that the optimal solution for the LBM does not change after restoring the customer storage capacity, i.e., changing the upper bound for inventory levels from $C_i + \bar{u}_i \leq C_i + u_i\epsilon_2$ to C_i, $i \in N$. Consider $\epsilon = \min\{\epsilon_1, \epsilon_2\}$.

By Proposition 20 we know the Algorithm 4 finishes after a finite number of steps with ϵ. If an optimal solution is found before the last iteration the algorithm stops. If not, then a solution whose cost is a lower bound to the problem is provided by the LBM. This solution has no multiple visits at the same time since the Algorithm 3 includes constraints at each time there are more than one visit. The algorithm maintains the valid lower bound condition of the LBM, since $\epsilon \leq \epsilon_1$. In addition, for this solution, the travel times are all corrected and the customers storage capacity is at most $C_i + u_i\epsilon$, $i \in N$. Since $\epsilon \leq \epsilon_2$, the inventory levels can be changed to C_i and the optimal solution is the same. Thus, the three relaxations for

the LBM listed in Section 6 are corrected, so the LBM is feasible and also is an optimal solution for the CIRP-OB. □

Appendix B - Branch-and-Bound Strategy

Consider the example in Figure 8. For this instance there is one customer c with usage rate $u_c = 1$, storage capacity $C_c = 10$ and initial inventory $I_c^0 = 10$. The time horizon is $H = 11$, so the customer needs at least one visit and the vehicle must deliver at least one unit of product. The vehicle capacity is $Q = 10$. The cost and travel times are equal, both with value 1. Assume that the time discretization for the customer and the depot is the set $\{1, 2, \ldots, 11\}$. For this instance and time discretization it is easy to check that there exists an optimal solution for formulation (1).

If we solve the linear relaxation of the formulation, the solution has a fraction 1/10 of a vehicle, that delivers 1 unit of product at time $t_c^1 = 1$. In order to get an integer solution, we consider the following branching scheme. In one branch there is a complete vehicle ($x_{0i}^0 = 1$) and the solution is integer. In the other branch, no vehicle departs at time t_0^0 from the depot ($x_{0i}^0 = 0$) and another fraction solution is optimal. A 2/10 fractional vehicle arrives at time 2 and delivers 2 units of product (note that 2 units are delivered because of the OPC in Proposition 10: the inventory after the delivery must be C_i). Again, it is needed to branch this solution: one branch finds the optimal solution ($x_{0i}^1 = 1$), while another has a 3/10 fractional vehicle arriving at time $t = 3$. Continuing in this fashion, after ten branching steps, a provable optimal solution is found.

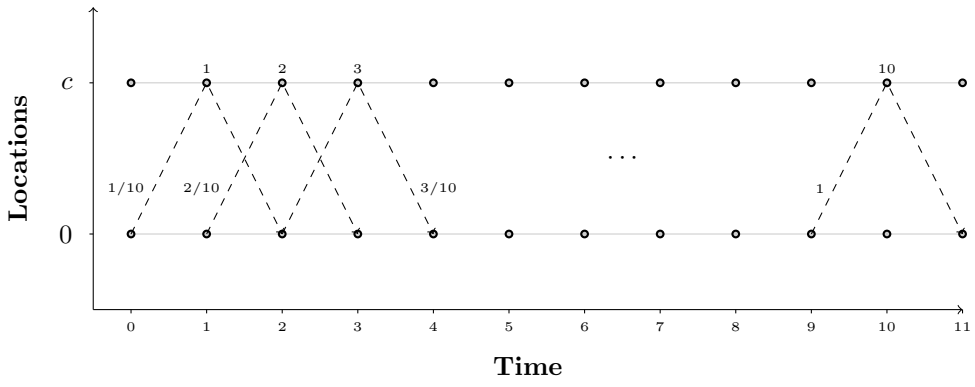

Figure 8: Simple Branch-and-Bound for a one-customer instance

To avoid this behavior, we seek a branching scheme that results in a more balanced search tree. We consider the variables x_{0i}^ℓ that represent visits to a customer $i \in N$ at time $k \in \mathcal{T}_i$, with $\ell \in \delta^-(i, t_i^k)$. In the branching strategy we propose, the first value we branch on is the total number of visits to i, i.e., we branch on the following sum,

$$S_i^{00} = \sum_{k=0}^{K_i} \sum_{\ell \in \delta^-(i,t_i^k)} x_{i0}^\ell.$$

When this sum is integer, the next step is branching on the following two sums,

$$S_i^{10} = \sum_{k=0}^{\lceil \frac{K_i}{2} \rceil - 1} \sum_{\ell \in \delta^-(i, t_i^k)} x_{i0}^{\ell},$$

$$S_i^{11} = \sum_{k=\lceil \frac{K_i}{2} \rceil}^{K_i} \sum_{\ell \in \delta^-(i, t_i^k)} x_{i0}^{\ell}.$$

Note that the value S_i^{00} is the sum of the values S_i^{10} and S_i^{11}. Once the two sums S_i^1 for this level have integer value, we continue branching on the following S_i^2 level. In general, for a level n, the sum $S_i^{n,k}$, $k = 0, \ldots, 2^n - 1$, corresponds to the following expression,

$$S_i^{n,k} = \sum_{k'=k_1}^{k_2} \sum_{\ell \in \delta^-(i, t_i^{k'})} x_{i0}^{\ell}, \tag{21}$$

with $k_1 = k \left\lceil \frac{K_i}{2^n} \right\rceil$ and $k_2 = \min \left\{ (k+1) \left\lceil \frac{K_i}{2^n} \right\rceil - 1, K_i \right\}$.

For the last level n, each value $S_i^{n,k}$ satisfies $k_2 - k_1 \leq 1$. Thus, the number of levels is $n = \lceil \log_2(K_i) \rceil - 1$.

The branching rule we propose starts branching on S_i^0 at the first level and, once it has integer value, the branching rule continues with S_i^1. Then, it branches on S_i^2, S_i^3, and so on, up to level $\lceil \log_2(K_i) \rceil - 1$. The level n is successfully branched when, for each customer $i \in N$, the sums S_i^n, that define the number of visits for a particular interval, have an integer value.

Appendix C - Iterations and Status Tables

In Table 3 the average number of iterations are computed for each instance type.

		N10			N20			N30		
		U1	U2	U3	U1	U2	U3	U1	U2	U3
G1	P25	38	30	5	87	52	32	162	80	117
	P50	31	14	11	75	52	31	186	53	41
	P75	35	23	20	35	56	15	111	111	79
G2	P25	23	23	46	62	73	97	101	104	88
	P50	34	17	6	77	79	84	81	89	76
	P75	21	17	32	75	17	75	99	70	83
G3	P25	30	44	18	77	73	110	84	109	71
	P50	21	14	35	111	78	55	72	58	68
	P75	22	15	57	69	88	85	82	94	78

Table 3: Average number of iterations.

Tables 4, 5 and 6 shows the number of instances with status Optimal, Feasible or No Solution after 2 hours for the number of customers and usage, geographical distribution and percentage.

Customers	10			20			30		
Usage	U1	U2	U3	U1	U2	U3	U1	U2	U3
Optimal	27	27	26	27	23	19	26	22	9
Feasible	0	0	1	0	4	6	0	4	10
No Solution	0	0	0	0	0	2	1	1	8

Table 4: Instance status for number of customers and usage rate

Customers	10			20			30		
Distribution	G1	G2	G3	G1	G2	G3	G1	G2	G3
Optimal	27	27	26	27	23	19	25	18	14
Feasible	0	0	1	0	3	7	2	6	6
No Solution	0	0	0	0	1	1	0	3	7

Table 5: Instance status for number of customers and geographical distribution

Customers	10			20			30		
Percentage	P25	P50	P75	P25	P50	P75	P25	P50	P75
Optimal	27	27	26	23	24	22	18	21	18
Feasible	0	0	1	2	3	5	3	5	6
No Solution	0	0	0	2	0	0	6	1	3

Table 6: Instance status for number of customers and percentage N_ℓ customers

Appendix D - CIRP-OB Instances and Results for Lagos et al. (2020)

In Lagos et al. (2020) randomly generated instances are solved using lower and upper bound models. We adapt those instances to the CIRP-OB and find new values for the number of vehicles, using the same methodology is used for finding a number of vehicles for CIRP instances. We run the lower bound model and the upper bound model with a homogeneous discretization $\Delta = 3$ and $\Delta = 1$, respectively, for 2 hours. The optimality gap is reported in Tables 7 and 8. Several instances do not have a provable optimal solution, specially random instances with more than 10 customers.

		Usage Level								
		U1			U2			U3		
		Capacity			**Capacity**			**Capacity**		
		Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3
Customers	5	0.00	0.00	18.85	0.00	0.00	0.00	0.00	0.00	0.00
	7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	8.67	10.66
	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	13.45	15.33
	12	0.00	6.09	14.61	0.00	5.38	6.09	3.90	0.00	0.00
	15	0.00	10.28	17.67	0.00	4.54	4.97	3.28	0.00	0.00

Table 7: Optimality gap (%) for clustered instances using LOWER(3) and UPPER(1) (2 hrs).

		Usage Level								
		U1			U2			U3		
		Capacity			**Capacity**			**Capacity**		
		Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3
Customers	5	0.00	15.80	17.50	0.00	0.00	0.00	0.00	0.00	0.00
	7	0.00	0.00	6.28	0.00	0.00	0.00	0.00	0.00	0.00
	10	12.79	3.03	8.27	2.26	12.79	21.89	1.86	2.37	8.14
	12	6.25	7.80	15.27	1.86	10.38	7.80	1.58	5.89	6.51
	15	8.97	6.54	10.45	4.68	17.51	15.28	4.08	8.17	13.14

Table 8: Optimality gap (%) for random instances using LOWER(3) and UPPER(1) (2 hrs).

The DDD algorithm runs for the same instances for up to 2 hours. In this case, all the instances have an optimal solution. In Tables 9 and 10 the running time is shown for each instance.

		Usage Level								
		U1			U2			U3		
		Capacity			**Capacity**			**Capacity**		
		Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3
Customers	5	0.19	0.19	0.73	0.20	0.19	0.19	0.22	0.20	0.18
	7	0.21	0.20	0.20	0.22	0.21	0.20	0.24	0.21	0.21
	10	1.14	0.99	1.11	1.14	1.08	1.07	1.33	1.09	1.05
	12	1.20	1.04	2.94	1.28	1.31	1.05	1.28	1.22	1.12
	15	1.29	1.24	4.36	0.33	0.32	0.24	0.37	0.29	0.25

Table 9: Running time (seconds) for clustered instances.

		Usage Level								
		U1			U2			U3		
		Capacity			**Capacity**			**Capacity**		
		Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3
Customers	5	0.24	49.42	44.79	0.25	0.21	0.23	0.23	0.23	0.21
	7	0.26	0.21	5.34	0.25	0.23	0.22	0.26	0.23	0.21
	10	141.04	33.81	102.25	0.26	4.84	262.97	0.34	0.25	6.72
	12	0.28	101.49	115.75	0.32	71.58	1.98	0.34	15.02	88.13
	15	0.31	596.33	645.91	0.34	434.38	1919.30	0.58	39.00	1159.75

Table 10: Running time (seconds) for random instances.

Appendix E - Results for instances with 100% customers in N_ℓ

In order to study the case when all customers have a storage capacity less than Q, we generate a new set of instances. We follow the same procedure we use for the initial set of instances, presented in Section 9.1, but in this case, we do not consider all the dimensions we used. Since the number of customers and the usage rate are the most important factors, we generate instances with 100% customers in N_ℓ for N=10,20,30 and U=1,2,3. As before, for each combination of factors, we sample three instances and for each instance, we find the minimum number of vehicles m using the UBM. We finally check that the DDD algorithm runs more than one iteration. In total we have 27 instances for each P=25,50,75,100 combination.

	25	**50**	**75**	**100**
Optimal (%)	81.48	85.19	85.19	74.07
Feasible (%)	7.41	14.81	11.11	22.22
No Solution (%)	11.11	0.00	3.70	3.70
Vehicles	14.93	15.74	16.30	16.74
Visits	22.54	25.89	26.04	28.38
Time Points	38.10	33.85	28.90	38.25
Initial Points (%)	17.86	19.55	20.99	13.97

Table 11: Results summary for instances with 25%, 50%, 75% and 100% of customers in N_ℓ.

In Table 11 we summarize the results for instances P=25,50,75,100. We report the percentage of instances with optimal, feasible (no optimality proved) or no solution for the DDD algorithm running up to 2 hours. We also compute the average number of vehicles, the average number of visits to customers, the average number of points added to the network and the percentage these points represent in the total number of points, once the algorithm finishes.

In general, from this Table, we cannot conclude that the P100 instances are more difficult to solve than the others. Even though the percentage of instances with optimal solution for P100 is less than in order cases, the percentage of instances with no solution is similar, and with respect to instances P25, is even more. We can conclude that both the number of vehicles and the number of visits are increasing on the number of customers in N_ℓ, which is an expected result. The number of time points added to the network by the algorithm does not show a clear correlation with the number of customers N_ℓ, but the number of initial time points is smaller for instances P100 than for other cases.

Appendix F - Result details

Instance	Lower	Best	Runtime	Iterations	Points
N10G1P25U1-1	110.08	110.08	4.79	33	220
N10G1P25U1-2	109.64	109.64	8.75	50	239
N10G1P25U1-3	110.98	110.98	3.64	30	183
N10G1P25U2-1	107.14	107.14	13.93	40	299
N10G1P25U2-2	109.36	109.36	7.20	35	252
N10G1P25U2-3	109.20	109.20	2.73	15	178
N10G1P25U3-1	107.20	107.20	0.90	3	98
N10G1P25U3-2	110.62	110.62	1.07	3	88
N10G1P25U3-3	110.92	110.92	2.27	10	154
N10G1P50U1-1	110.06	110.06	3.11	19	194
N10G1P50U1-2	110.56	110.56	14.01	55	296
N10G1P50U1-3	106.02	106.02	4.28	18	219
N10G1P50U2-1	108.62	108.62	1.79	8	133
N10G1P50U2-2	109.38	109.38	3.58	17	193
N10G1P50U2-3	109.44	109.44	2.80	17	174
N10G1P50U3-1	109.04	109.04	3.53	15	197
N10G1P50U3-2	107.30	107.30	2.77	13	171
N10G1P50U3-3	113.32	113.32	1.13	4	111
N10G1P75U1-1	109.02	109.02	12.61	53	311
N10G1P75U1-2	110.62	110.62	2.69	18	166
N10G1P75U1-3	108.74	108.74	4.82	35	224
N10G1P75U2-1	109.76	109.76	0.85	10	141
N10G1P75U2-2	98.90	98.90	1.92	18	174
N10G1P75U2-3	110.42	110.42	6.62	40	216
N10G1P75U3-1	106.90	106.90	2.07	16	180
N10G1P75U3-2	105.10	105.10	8.63	40	236
N10G1P75U3-3	105.02	105.02	0.35	3	90
N10G2P25U1-1	113.34	113.34	1.04	15	163
N10G2P25U1-2	112.04	112.04	3.21	25	238
N10G2P25U1-3	115.50	115.50	3.83	30	219
N10G2P25U2-1	110.38	110.38	2.33	24	183
N10G2P25U2-2	108.64	108.64	0.70	7	153
N10G2P25U2-3	108.96	108.96	20.45	37	314
N10G2P25U3-1	112.60	112.60	15.31	36	271
N10G2P25U3-2	123.18	123.18	721.17	85	565
N10G2P25U3-3	116.26	116.26	3.50	18	202
N10G2P50U1-1	100.98	100.98	1.46	13	190
N10G2P50U1-2	100.66	100.66	34.80	78	355
N10G2P50U1-3	111.62	111.62	0.78	11	143
N10G2P50U2-1	101.20	101.20	1.70	17	158
N10G2P50U2-2	110.08	110.08	1.65	10	188
N10G2P50U2-3	118.70	118.70	3.13	25	183
N10G2P50U3-1	88.76	88.76	0.96	8	153
N10G2P50U3-2	115.38	115.38	0.41	4	102
N10G2P50U3-3	106.68	106.68	0.72	7	140
N10G2P75U1-1	104.78	104.78	3.89	28	199
N10G2P75U1-2	113.52	113.52	0.66	10	143
N10G2P75U1-3	103.46	103.46	3.15	24	218
N10G2P75U2-1	108.10	108.10	0.85	12	129
N10G2P75U2-2	109.64	109.64	4.13	29	216
N10G2P75U2-3	108.72	108.72	1.04	10	161
N10G2P75U3-1	119.64	119.64	91.83	60	365
N10G2P75U3-2	122.54	122.54	33.94	34	294
N10G2P75U3-3	109.78	109.78	0.32	3	81
N10G3P25U1-1	107.38	107.38	10.77	37	278
N10G3P25U1-2	111.18	111.18	3.72	28	210
N10G3P25U1-3	102.28	102.28	3.78	25	248
N10G3P25U2-1	116.80	116.80	1.07	8	157
N10G3P25U2-2	99.68	99.68	16.96	47	284
N10G3P25U2-3	101.58	101.58	252.07	76	428
N10G3P25U3-1	119.72	119.72	11.14	15	304
N10G3P25U3-2	114.14	114.14	10.75	29	261
N10G3P25U3-3	118.28	118.28	1.63	9	184
N10G3P50U1-1	119.30	119.30	0.89	10	142
N10G3P50U1-2	108.22	108.22	3.15	24	180
N10G3P50U1-3	84.80	84.80	5.74	28	235
N10G3P50U2-1	121.62	121.62	58.05	30	314
N10G3P50U2-2	122.82	122.82	0.36	3	82
N10G3P50U2-3	112.10	112.10	1.60	10	181
N10G3P50U3-1	120.16	120.16	102.02	48	366
N10G3P50U3-2	123.70	123.70	12.31	22	247
N10G3P50U3-3	127.44	127.44	31.01	35	291
N10G3P75U1-1	115.14	115.14	1.18	14	175
N10G3P75U1-2	100.22	100.22	1.12	10	171
N10G3P75U1-3	98.08	98.08	14.76	43	294
N10G3P75U2-1	118.96	118.96	1.30	11	166
N10G3P75U2-2	123.12	123.12	6.24	31	199
N10G3P75U2-3	108.34	108.34	0.35	4	78
N10G3P75U3-1	120.20	120.20	152.49	29	396
N10G3P75U3-2	124.54	124.82	7200.00	88	749
N10G3P75U3-3	146.80	146.80	663.58	54	516

Table 12: Instances with 10 customers (N10).

Instance	Lower	Best	Runtime	Iterations	Points
N20G1P25U1-1	214.48	214.48	317.91	163	757
N20G1P25U1-2	213.28	213.28	21.72	27	528
N20G1P25U1-3	208.16	208.16	90.14	71	701
N20G1P25U2-1	210.92	210.92	188.32	58	795
N20G1P25U2-2	211.22	211.22	35.95	30	535
N20G1P25U2-3	213.18	213.18	157.25	67	653
N20G1P25U3-1	208.92	208.92	19.59	16	313
N20G1P25U3-2	210.00	210.00	251.76	67	701
N20G1P25U3-3	215.44	215.44	20.07	14	370
N20G1P50U1-1	207.84	207.84	55.25	45	580
N20G1P50U1-2	208.56	208.56	130.29	69	721
N20G1P50U1-3	209.42	209.42	257.36	110	770
N20G1P50U2-1	214.10	214.10	55.02	43	545
N20G1P50U2-2	205.80	205.80	208.76	79	672
N20G1P50U2-3	210.22	210.22	44.02	34	543
N20G1P50U3-1	225.68	225.68	481.25	63	652
N20G1P50U3-2	208.86	208.86	17.89	13	293
N20G1P50U3-3	209.84	209.84	19.70	16	339
N20G1P75U1-1	214.62	214.62	32.40	43	451
N20G1P75U1-2	198.96	198.96	34.08	40	474
N20G1P75U1-3	212.02	212.02	7.60	22	423
N20G1P75U2-1	205.46	205.46	83.07	55	565
N20G1P75U2-2	225.30	225.30	314.99	71	714
N20G1P75U2-3	211.38	211.38	29.19	42	420
N20G1P75U3-1	209.20	209.20	26.05	32	399
N20G1P75U3-2	206.64	206.64	1.97	7	231
N20G1P75U3-3	209.60	209.60	1.57	5	222
N20G2P25U1-1	203.28	203.28	65.21	66	561
N20G2P25U1-2	215.58	215.58	77.60	57	672
N20G2P25U1-3	212.24	212.24	84.63	62	634
N20G2P25U2-1	217.68	217.68	55.30	43	532
N20G2P25U2-2	218.16	218.16	1624.73	116	989
N20G2P25U2-3	228.10	228.10	467.96	59	889
N20G2P25U3-1	221.50	221.50	152.75	41	624
N20G2P25U3-2	221.56	-	7200.00	182	1132
N20G2P25U3-3	220.64	220.64	907.96	67	803
N20G2P50U1-1	202.72	202.72	383.61	96	844
N20G2P50U1-2	198.30	198.30	445.96	91	823
N20G2P50U1-3	221.64	221.64	40.74	45	522
N20G2P50U2-1	230.54	230.54	7164.19	135	1113
N20G2P50U2-2	225.42	225.42	207.92	40	721
N20G2P50U2-3	219.46	219.46	211.38	63	678
N20G2P50U3-1	230.50	230.50	675.38	58	810
N20G2P50U3-2	239.40	283.92	7200.00	114	1115
N20G2P50U3-3	224.06	224.06	974.09	80	764
N20G2P75U1-1	202.58	202.58	1.61	6	209
N20G2P75U1-2	205.24	205.24	539.34	101	906
N20G2P75U1-3	205.44	205.44	559.73	119	857
N20G2P75U2-1	202.68	202.68	4.11	9	272
N20G2P75U2-2	224.58	224.58	5.41	15	349
N20G2P75U2-3	218.58	218.58	12.89	27	372
N20G2P75U3-1	220.34	220.38	7200.00	103	951
N20G2P75U3-2	220.46	220.48	7200.00	117	1017
N20G2P75U3-3	211.58	211.58	1.82	6	244
N20G3P25U1-1	221.18	221.18	102.53	58	663
N20G3P25U1-2	189.56	189.56	431.48	96	843
N20G3P25U1-3	189.26	189.26	785.55	78	884
N20G3P25U2-1	209.94	209.94	81.29	41	579
N20G3P25U2-2	216.78	226.26	7200.00	109	1289
N20G3P25U2-3	228.32	228.32	1721.03	70	1070
N20G3P25U3-1	213.94	213.94	6739.51	117	1125
N20G3P25U3-2	238.22	-	7200.00	101	1136
N20G3P25U3-3	233.82	270.04	7200.00	113	1095
N20G3P50U1-1	199.52	199.52	33.47	39	513
N20G3P50U1-2	193.92	193.92	821.79	126	848
N20G3P50U1-3	200.52	200.52	1868.88	169	1210
N20G3P50U2-1	198.72	198.72	414.70	58	713
N20G3P50U2-2	257.66	257.68	7200.00	115	1104
N20G3P50U2-3	210.14	210.14	1306.51	60	872
N20G3P50U3-1	235.62	235.64	7200.00	128	1100
N20G3P50U3-2	231.50	231.50	26.43	19	399
N20G3P50U3-3	248.74	248.74	23.44	19	435
N20G3P75U1-1	156.96	156.96	180.76	74	633
N20G3P75U1-2	208.54	208.54	451.58	97	844
N20G3P75U1-3	191.74	191.74	46.35	35	582
N20G3P75U2-1	226.54	226.54	379.16	57	647
N20G3P75U2-2	206.52	213.18	7200.00	75	942
N20G3P75U2-3	236.56	242.12	7200.00	133	1048
N20G3P75U3-1	214.70	215.32	7200.00	139	898
N20G3P75U3-2	234.04	234.04	3158.80	64	905
N20G3P75U3-3	234.84	234.84	529.05	53	671

Table 13: Instances with 20 customers (N20).

Instance	Lower	Best	Runtime	Iterations	Points
N30G1P25U1-1	307.18	307.18	435.25	88	1165
N30G1P25U1-2	309.42	309.42	7077.08	231	2082
N30G1P25U1-3	310.52	310.52	2959.09	166	1767
N30G1P25U2-1	310.00	310.00	413.55	54	1093
N30G1P25U2-2	306.62	306.62	1418.79	96	1307
N30G1P25U2-3	321.04	321.04	1203.95	89	1306
N30G1P25U3-1	310.60	310.60	1040.41	96	995
N30G1P25U3-2	319.32	319.32	1956.07	68	1189
N30G1P25U3-3	304.54	313.70	7200.00	187	1529
N30G1P50U1-1	311.34	311.34	5647.27	219	1887
N30G1P50U1-2	301.48	301.48	789.43	120	1326
N30G1P50U1-3	310.54	310.54	4729.12	220	1613
N30G1P50U2-1	315.32	315.32	231.52	49	882
N30G1P50U2-2	320.22	320.22	552.07	60	1089
N30G1P50U2-3	312.18	312.18	300.87	49	1035
N30G1P50U3-1	313.48	313.48	83.43	23	638
N30G1P50U3-2	323.10	323.10	2084.87	90	1116
N30G1P50U3-3	309.22	309.22	29.26	11	461
N30G1P75U1-1	306.68	306.68	487.31	86	1088
N30G1P75U1-2	311.74	311.74	2261.30	163	1425
N30G1P75U1-3	296.98	296.98	422.31	84	1018
N30G1P75U2-1	312.36	312.36	19.98	17	507
N30G1P75U2-2	316.02	316.02	4233.20	187	1360
N30G1P75U2-3	316.24	316.24	3370.19	128	1207
N30G1P75U3-1	318.20	318.20	1166.61	78	891
N30G1P75U3-2	304.88	304.90	7200.00	149	1208
N30G1P75U3-3	308.38	308.38	10.12	10	389
N30G2P25U1-1	295.90	295.90	2335.74	126	1586
N30G2P25U1-2	319.74	319.74	733.21	86	1398
N30G2P25U1-3	314.22	314.22	579.88	90	1244
N30G2P25U2-1	309.66	309.66	1562.17	94	1388
N30G2P25U2-2	338.74	346.40	7200.00	113	1547
N30G2P25U2-3	309.40	309.40	1750.36	104	1346
N30G2P25U3-1	317.88	324.90	7200.00	104	1443
N30G2P25U3-2	340.06	-	7200.00	83	1435
N30G2P25U3-3	319.64	-	7200.00	76	1484
N30G2P50U1-1	286.36	286.36	730.75	95	1192
N30G2P50U1-2	330.86	330.86	106.11	38	915
N30G2P50U1-3	298.54	298.54	1463.70	111	1390
N30G2P50U2-1	322.46	322.46	2247.04	102	1182
N30G2P50U2-2	330.66	330.66	481.76	51	1008
N30G2P50U2-3	301.02	308.02	7200.00	114	1311
N30G2P50U3-1	331.76	331.76	1549.20	67	986
N30G2P50U3-2	328.10	335.40	7200.00	74	1370
N30G2P50U3-3	314.76	353.38	7200.00	88	1330
N30G2P75U1-1	304.86	304.86	2100.89	134	1386
N30G2P75U1-2	302.94	302.94	1044.95	117	1132
N30G2P75U1-3	304.52	304.52	151.35	47	877
N30G2P75U2-1	310.26	310.26	3083.55	92	1207
N30G2P75U2-2	312.82	312.82	148.60	41	843
N30G2P75U2-3	315.44	315.44	953.23	76	1005
N30G2P75U3-1	320.60	320.72	7200.00	125	1192
N30G2P75U3-2	318.20	318.20	72.45	19	607
N30G2P75U3-3	315.78	-	7200.00	105	1167
N30G3P25U1-1	298.04	298.04	1593.19	94	1496
N30G3P25U1-2	311.88	311.88	143.00	52	914
N30G3P25U1-3	302.08	302.08	3123.85	105	1569
N30G3P25U2-1	326.92	-	7200.00	86	1602
N30G3P25U2-2	330.38	330.38	6575.17	128	1543
N30G3P25U2-3	325.92	325.92	5162.23	113	1506
N30G3P25U3-1	331.44	-	7200.00	69	1452
N30G3P25U3-2	342.44	-	7200.00	82	1380
N30G3P25U3-3	346.22	-	7200.00	63	1304
N30G3P50U1-1	273.32	273.32	2940.29	113	1452
N30G3P50U1-2	302.80	302.80	768.84	61	1244
N30G3P50U1-3	327.88	327.88	291.60	41	913
N30G3P50U2-1	339.96	339.96	2091.10	82	1129
N30G3P50U2-2	348.34	348.34	300.00	45	941
N30G3P50U2-3	349.44	349.44	3362.65	48	1240
N30G3P50U3-1	347.56	369.54	7200.00	60	1197
N30G3P50U3-2	334.46	-	7200.00	52	1144
N30G3P50U3-3	339.98	348.08	7200.00	92	1269
N30G3P75U1-1	283.22	283.22	2568.24	82	1263
N30G3P75U1-2	318.98	318.98	266.97	51	1056
N30G3P75U1-3	272.84	-	7200.00	114	1596
N30G3P75U2-1	332.68	362.10	7200.00	78	1308
N30G3P75U2-2	342.06	342.06	6811.06	104	1408
N30G3P75U2-3	321.84	327.30	7200.00	101	1422
N30G3P75U3-1	330.64	380.78	7200.00	59	1233
N30G3P75U3-2	357.79	-	7200.00	109	1348
N30G3P75U3-3	330.62	375.94	7200.00	65	1371

Table 14: Instances with 30 customers (N30).

In Tables 12, 13 and 14 the result details are shown. The Instance column has the instance name, corresponding to the type slash (-) sample number. The Lower column shows the best lower bound value found and the Best column, the best feasible solution value. The Runtime column indicates the running time of the DDD algorithm and the Iterations column, the total number of iterations. The Points column shows the total number of time points added to the time-expanded network (including all locations in N_0).