A Multi-Period Vehicle Routing Problem with Driver Consistency and Arrival-Time Diversification

Jahir Llagas [a,*], Foteini Stavropoulou[b], Sanne Wøhlk[a]

[a]*Department of Business Development and Technology, Birk Centerpark 15, 7400 Herning, Denmark*
[b]*Liverpool Business School, Liverpool John Moores University, Liverpool L3 5UG, United Kingdom*

Abstract

We study a Multi-Period Vehicle Routing Problem that integrates driver consistency and arrival-time diversification. Driver consistency requires each customer to be served by the same vehicle across all service periods, while arrival-time diversification enforces temporal separation between visits in different periods. We first propose a Mixed-Integer Linear Programming formulation and an exact Logic-Based Benders Decomposition implemented through a Branch-and-Check scheme, where a multi-period VRP with driver consistency is handled in the master problem and arrival-time diversification is addressed in the subproblem. Building on this structure, we develop a matheuristic that exploits the subproblem in an efficient and targeted manner. As arrival-time diversification checks are computationally expensive, we propose a feasibility check procedure over a restricted search space. Computational results show that the branch-and-check approach solves medium-scale instances to optimality, while the matheuristic achieves near-optimal solutions in significantly shorter computation times. Lastly, the impact of arrival-time diversification is examined and managerial insights are derived.

Keywords: Logic-Based Benders Decomposition, Branch-and-Check, Constraint Programming, Arrival-time Diversification, Driver Consistency

1. Introduction

In many logistics systems, services must be provided repeatedly over several days, often following stable operational routines that support efficiency in terms of cost or time. However, in several application domains, such regularity may introduce significant security and operational risks. In cash-in-transit and high-value distribution, predictable routing patterns can be anticipated and exploited, increasing exposure to theft and targeted attacks. As a result, reducing route predictability has been recognized as a critical planning concern (Talarico et al., 2017; Fröhlich et al., 2023). Similar issues arise in overnight security and patrol services, where repeated visit sequences or schedules may compromise security

*Corresponding author

effectiveness, motivating the use of diversified or peripatetic routing strategies (Calvo and Cordone, 2003; Duchenne et al., 2007). In humanitarian and emergency logistics, routing decisions must also account for security and operational vulnerability in unstable environments. Empirical evidence from disaster response highlights that safety considerations often extend beyond classical efficiency objectives, and that some organizations deliberately vary routes and dispatch times to avoid establishing predictable patterns (De la Torre et al., 2012). More broadly, field-based studies emphasize that security has become a dominant concern in humanitarian operations, calling for planning approaches that better reflect these contextual risks (Pedraza-Martinez and Van Wassenhove, 2016).

To mitigate these risks, the literature has explored different strategies aimed at reducing route predictability. Early approaches focus on route randomization, introducing controlled perturbations to generate different path patterns across periods (Kuo et al., 2008). Other works propose peripatetic patterns, enforcing systematic alternation between zones or visit sequences to avoid repetition (Duchenne et al., 2007). A related stream of research explicitly models risk exposure, particularly in cash-in transit, where risk is typically proportional to the amount of valuables transported and the time or distance spent in transit (Talarico et al., 2017). In overnight security services, Calvo and Cordone (2003) address predictability by adopting a two-phase strategy, first clustering service locations to ensure feasible response times and then generating patrol routes within each cluster to reduce repetition. More recently, arrival-time diversification has been proposed as an alternative mechanism to reduce temporal regularity by introducing controlled variation in service start times across planning periods (Michallet et al., 2014). Although effective in reducing predictability, these strategies often lead to deviations from efficient routes, increased operational cost and variability, and additional planning and coordination effort, which may be difficult to sustain in recurrent operations.

Given these limitations, an alternative is to model the problem as a Multi-Period vehicle routing problem that incorporates an unpredictability mechanism. These approaches help balance workload and control operational variability across periods, but they typically allow a customer to be visited by different vehicles over time, which reduces familiarity with the customer's environment, limits the detection of unusual situations, and weakens operational traceability. A promising way to address these issues is to enforce driver consistency, understood as assigning the same vehicle to a customer in the periods when service is required. This continuity not only mitigates the disadvantages of rotating teams but also helps drivers develop contextual knowledge, improves communication with customers, enhances service reliability, and supports more stable planning by reducing coordination effort and facilitating long-term monitoring (Kovacs et al., 2014a).

Thus, we propose a Multi-Period Vehicle Routing Problem with Driver Consistency and Arrival-Time Diversification (VRPDC-ATD), whose objective is to design routes over multiple periods while minimizing travel time and satisfying capacity limits, route-duration constraints, driver consistency requirements, and minimum separation between service start times. Since spatial diversification strategies often incur additional costs or complexity, it is natural to explore sources of unpredictability along other dimensions of the problem. In this work, we focus on the temporal dimension as a mechanism better aligned with recurrent

operations. Arrival-time diversification introduces controlled variation in service times and may induce limited adjustments in visit sequences when required to satisfy temporal separation constraints. As a result, unpredictability arises primarily from timing decisions, while the spatial structure of routes is modified only when operationally justified. This paper makes the following contributions:

- We introduce a new Multi-Period Vehicle Routing Problem that jointly enforces driver consistency and arrival-time diversification under a no-waiting assumption.
- We propose a Branch-and-Check framework that decomposes the problem into a multi-period vehicle routing with driver consistency and a temporal feasibility subproblems for arrival-time diversification.
- We develop a matheuristic based on a hybrid Simulated Annealing and Variable Neighborhood Descent local search. Computational experiments demonstrate the effectiveness and scalability of the proposed approach.

The remainder of the paper is organized as follows: Section 2 reviews the related literature; Section 3 formally presents the problem; Section 4 presents the mathematical model; Section 5 introduces an exact approach based on a specialized Branch-and-Check decomposition; Section 6 proposes a matheuristic method denoted HAVS; Section 7 reports the computational experiments; and finally Section 8 offers our conclusions.

2. Literature Review

This section reviews the two structural features that motivate our work—driver consistency and arrival-time diversification—and examines how each has been addressed in the literature. We highlight their operational motivations, summarize the main methodological approaches, and discuss how time-window constraints limit diversification. We then analyze their interaction, which reveals a methodological gap that motivates our contribution.

2.1. Driver Consistency in Multi-Period Routing

Driver consistency has become an important feature in Multi-Period Routing Problems, particularly in services where repeated interactions between service provider and customer influence service quality. In settings such as parcel delivery or home care, assigning the same visiting employee improves familiarity, reduces service times, and manages customer expectations (Kovacs et al., 2014b). This idea was formalized by Groër et al. (2009) in the Consistent Vehicle Routing Problem (ConVRP), where each customer must be served by the same driver across the planning horizon. Evidence from home healthcare reinforces the operational relevance of continuity, showing that repeated assignments improve coordination and service quality, whereas frequent changes degrade performance (Gao et al., 2026). Similar consistency-driven requirements also arise in school transportation, where the Group-Consistent Dial-a-Ride Problem enforces stable passenger groupings across days to account for user well-being, particularly for children sensitive to change (Lindstrøem, 2022).

Early approaches relied on template routes to preserve consistent assignments throughout the planning horizon (Groër et al., 2009; Tarantilis et al., 2012; Kovacs et al., 2014b, 2015; Alvarez et al., 2024), but such methods limit routing flexibility. Subsequent work therefore introduced flexible consistency mechanisms, allowing a controlled number of drivers per customer while still promoting continuity (Coelho and Laporte, 2013; Braekers and Kovacs, 2016).

On the algorithmic side, driver consistency introduces strong inter-period dependencies. Goeke et al. (2019) shows that classical decomposition and column-generation approaches become ineffective under such coupling, motivating exact methods that model full Multi-Period route sets. Stavropoulou (2022) further show that consistency remains a dominant structural decision even in heterogeneous fleets, where customer assignments strongly influence feasibility and routing efficiency.

2.2. Arrival-Time Diversification in Multi-Period Routing

Arrival-time diversification enforces variability in service times across periods to reduce temporal predictability. It is particularly relevant in security-sensitive settings such as cash-in-transit logistics, security services, and humanitarian operations, where regular visit schedules increase exposure to risk. Diversification is typically modeled through a minimum temporal separation between visits, combined with a no-waiting assumption that ties arrival times directly to routing and departure decisions.

Two main modeling approaches exist. One embeds diversification within routing problems with time windows. Michallet et al. (2014) enforce minimum separation alongside classical time-window constraints and no waiting, evaluating diversification through penalty functions derived from route-level timing profiles. A second line of work modifies the feasible arrival-time domain directly. Hoogeboom and Dullaert (2019) construct admissible arrival-time intervals by excluding previously used time bands and an additional temporal margin around them, yielding routing problems with multiple time windows per customer. More recent extensions increase routing flexibility by allowing alternative paths with different travel times (Soriano et al., 2020), though this enlarges the decision space and complicates feasibility checks.

From an algorithmic perspective, these studies handle arrival-time diversification mainly through heuristic or reformulation-based mechanisms. Michallet et al. (2014) use a multi-start iterated local search with penalty functions for time-window and time-spread violations. Hoogeboom and Dullaert (2019) reformulate the problem as a rolling-horizon vehicle routing problem with multiple time windows and solve it with an iterated granular tabu search. Soriano et al. (2020) exploit a multi-graph representation to introduce alternative travel-time profiles. These methods are effective for temporal diversification, but they do not impose a fixed customer-to-vehicle assignment across the planning horizon.

When Time Windows Restrict Arrival-Time Diversification. Most diversification studies rely on time-window–based benchmark instances, where narrow or heterogeneous windows severely limit feasible temporal separation (Michallet et al., 2014; Hoogeboom and Dullaert, 2019; Soriano et al., 2020). To maintain feasibility, prior work often increases fleet size

substantially, reducing practical relevance. For example, Michallet et al. (2014) use the Solomon instances and enlarge the fleet up to 30 vehicles, while Hoogeboom and Dullaert (2019) remove fleet-size limits entirely, requiring up to 68 vehicles for instances with 100 customers.

By contrast, real-world datasets with broader operating hours support diversification more naturally. For example, Hoogeboom and Dullaert (2019) report bank operations with wide operating hours, while Michallet et al. (2014) present four-period instances with broad time windows. Overall, diversification is only effective when time windows are sufficiently wide to permit meaningful temporal variation.

2.3. Joint Driver Consistency and Arrival-Time Diversification

Combining driver consistency with arrival-time diversification is motivated by their complementary roles in security-sensitive multi-period operations. Consistency promotes continuity and operational awareness, as drivers develop local knowledge and are better able to detect and respond to unusual situations. Arrival-time diversification, in turn, reduces exposure by limiting temporal predictability in repeated visits. Together, these dimensions create a natural tension between stability and controlled variability across periods.

A closely related perspective appears in the overnight security service studied by Calvo and Cordone (2003), where management requires the generation of many low-cost yet distinct patrol schedules for security reasons. Repeating identical routes night after night would make patrol movements predictable and easier to exploit. To address this, the authors preserve stable service assignments, which are important for practicability and staff training, while inducing variability through modifications of the time-window constraints of routine services. However, this variability is obtained by solving separate Vehicle Routing Problem with Time-Windows (VRPTW) instances for each night, with time-window parameters adjusted between instances. Temporal variation is therefore introduced at the instance level rather than modeled as an explicit decision dimension within a unified structure. As a consequence, the degree of temporal separation between visits across nights is not directly controlled, and feasibility is verified independently for each night.

This separation highlights a broader methodological limitation. Methods developed for arrival-time diversification typically modify arrival-time domains, penalize timing violations, or exploit alternative paths (Michallet et al., 2014; Hoogeboom and Dullaert, 2019; Soriano et al., 2020), but they do not embed driver consistency as a global multi-period assignment decision. Conversely, ConVRP methods preserve stable customer-to-driver assignments through template routes, tabu search, adaptive large neighborhood search, or multi-period route-set formulations (Groër et al., 2009; Tarantilis et al., 2012; Kovacs et al., 2014b; Goeke et al., 2019). These approaches are designed to preserve regularity and, in several cases, arrival-time consistency, whereas our setting requires controlled temporal separation. Thus, when driver consistency and arrival-time diversification are enforced simultaneously, routing and timing decisions become inherently coupled across periods. A modification in one period may alter feasibility in another because customer assignments and arrival times are jointly constrained.

Therefore, to address this gap, we introduce a unified Multi-Period Vehicle Routing Problem that integrates driver consistency with explicit inter-period arrival-time separation constraints over a planning horizon.

3. Problem Description

We address the *Multi-Period Vehicle Routing Problem with Driver Consistency and Arrival-Time Diversification* (VRPDC-ATD), defined on a complete directed graph $G = (V, A)$, where $V = \{0, 1, \ldots, n\}$ is the set of nodes, with node 0 denoting the depot and $C = V \setminus \{0\}$ the set of customers. The arc set is defined as $A = \{(i, j) : i, j \in V, i \neq j\}$, where each arc $(i, j) \in A$ is associated with a travel time t_{ij}. The planning horizon is divided into a discrete set of periods P. Within each period $p \in P$, time is continuous over the interval $[0, H]$, where H denotes the route completion limit. Vehicles may depart from the depot at any time within this interval and must return to the depot no later than H. The vehicles operate under a no-waiting assumption and thus, arrival times are entirely determined by the depot departure time, travel times, and service durations.

A homogeneous fleet of K vehicles, each with capacity Q, is available at the depot. Routes are defined independently in each period. Thus, whenever a vehicle is used in period p, it starts from the depot and returns to the depot within the same period. Unused vehicles remain at the depot. Each customer $i \in C$ has a deterministic demand $q_i^p \geq 0$ in each period $p \in P$ and a fixed service time $s_i \geq 0$ per visit. We assume that $q_i^p \leq Q$ for all $i \in C$ and $p \in P$. Whenever $q_i^p > 0$, customer i must be served exactly once in period p and customer locations cannot be used as transit points without being served.

Beyond these classic routing considerations, this problem incorporates two additional operational requirements. First, the *driver consistency* requirement stipulates that each customer is served by the same driver (vehicle) in all periods in which the customer has a positive demand. Second, the *arrival-time diversification* requirement specifies that, for any customer served in at least two different periods, the pairwise difference between the corresponding arrival times must be at least ϵ time units. Figure 1 illustrates these two requirements in a simple two-period example. Panel (a) shows the driver-consistency assignment. Panels (b) and (c) show that, despite this fixed assignment, the route sequence may differ across periods. Finally, Panel (d) shows how different route departure times induce customer arrival times that satisfy the minimum separation requirement ϵ.

The objective is to determine feasible vehicle routes for each period that satisfy capacity limits, route completion constraints, and driver-consistency and arrival-time-diversification requirements, while minimizing total travel time across the planning horizon.

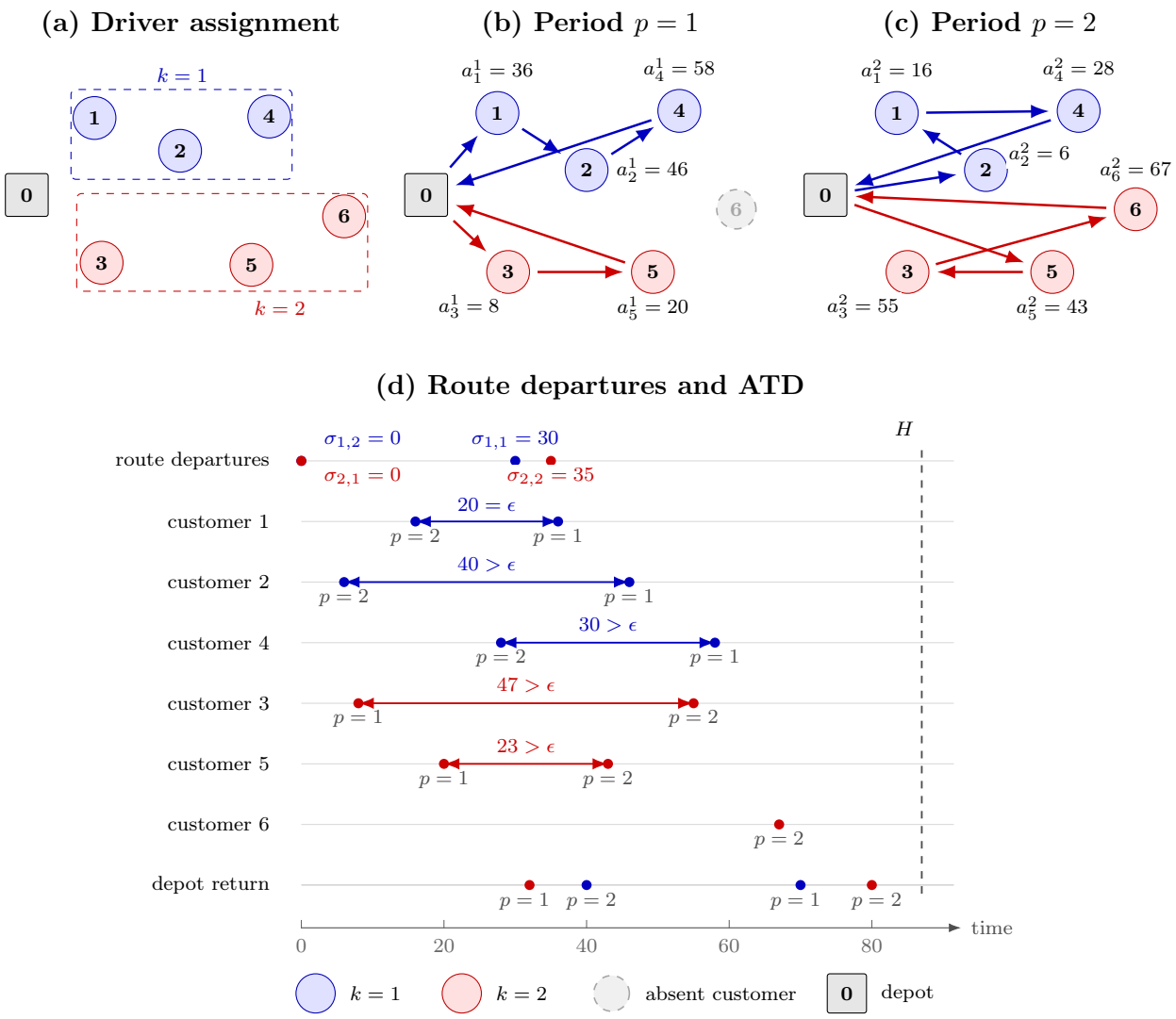

Figure 1: Illustrative VRPDC-ATD example with two periods and $\epsilon = 20$. The labels σ_{kp} indicate route departure times of vehicle k in period p, while the labels a_i^p indicate the arrival time of customer i in period p. Customers are served by the same vehicle, but route sequences and departure times may vary across periods. Customer 6 is absent in period 1 since $q_6^1 = 0$.

4. A Compact Formulation for the VRPDC-ATD

We adopt a four-index arc-flow formulation with assignment and timing variables. The formulation uses the input data introduced in Section 3. For each period $p \in P$, we define $C^p := \{i \in C : q_i^p > 0\}$ as the set of customers that must be visited in period p, and $A^p := \{(i,j) : i,j \in C^p \cup \{0\}, i \neq j\}$ as the corresponding set of arcs.

The decision variables are defined as follows. Let $x_{ij}^{kp} \in \{0,1\}$ indicate whether vehicle $k \in K$ traverses arc $(i,j) \in A^p$ in period $p \in P$. Let $y_i^k \in \{0,1\}$ indicate whether customer

$i \in C$ is assigned to vehicle k over the entire planning horizon. Let $a_i^p \geq 0$ denote the arrival time at customer i in period p. Finally, for each pair of periods $p, p' \in P$ with $p < p'$ and each customer $i \in C^p \cap C^{p'}$, let $z_i^{pp'} \in \{0,1\}$ select the ordering used to impose the minimum arrival-time separation between the two visits. A complete notation summary, including input parameters, derived sets, and decision variables, is provided in Appendix A.

Using this notation, the VRPDC-ATD can be formulated as follows:

$$\min \sum_{p \in P} \sum_{k \in K} \sum_{(i,j) \in A_p} (t_{ij} + s_i)\, x_{ij}^{kp} \tag{1}$$

s.t.:

$$\sum_{k \in K} y_i^k = 1, \quad \forall i \in C \tag{2}$$

$$\sum_{j \in C_p} x_{0j}^{kp} = \sum_{j \in C_p} x_{j0}^{kp} \leq 1, \quad \forall p \in P,\ k \in K \tag{3}$$

$$\sum_{\substack{j \in C_p \cup \{0\} \\ j \neq i}} x_{ji}^{kp} = \sum_{\substack{j \in C_p \cup \{0\} \\ j \neq i}} x_{ij}^{kp} = y_i^k, \quad \forall\, p \in P, i \in C_p,\ k \in K \tag{4}$$

$$\sum_{i \in C_p} q_i^p\, y_i^k \leq Q, \quad \forall p \in P,\ k \in K \tag{5}$$

$$a_j^p \geq a_i^p + t_{ij} + s_i - M\,(1 - \sum_{k \in K} x_{ij}^{kp}), \quad \forall p \in P,\ (i,j) \in A_p \tag{6}$$

$$a_j^p \leq a_i^p + t_{ij} + s_i + M\,(1 - \sum_{k \in K} x_{ij}^{kp}), \quad \forall p \in P,\ (i,j) \in A_p \tag{7}$$

$$t_{0i} \leq a_i^p \leq H - t_{i0}, \quad \forall i \in C_p,\ p \in P \tag{8}$$

$$a_i^p - a_i^{p'} \geq \epsilon - M\,(1 - z_i^{pp'}), \quad \forall p, p' \in P : p < p',\ i \in C_p \cap C_{p'} \tag{9}$$

$$a_i^{p'} - a_i^p \geq \epsilon - M\, z_i^{pp'}, \quad \forall p, p' \in P : p < p',\ i \in C_p \cap C_{p'} \tag{10}$$

$$a_i^p \in \mathbb{R}_{\geq 0}, \quad \forall p \in P,\ i \in C^p \tag{11}$$

$$y_i^k \in \{0,1\}, \quad \forall i \in C,\ k \in K \tag{12}$$

$$x_{ij}^{kp} \in \{0,1\}, \quad \forall p \in P,\ (i,j) \in A_p,\ k \in K \tag{13}$$

$$z_i^{pp'} \in \{0,1\}, \quad \forall p, p' \in P : p < p',\ i \in C_p \cap C_{p'} \tag{14}$$

The objective function (1) minimizes the total travel and service time across all vehicles and periods. Constraints (2) enforce driver consistency by assigning each customer to exactly one vehicle over the planning horizon. Constraints (3) and (4) ensure route connectivity: each vehicle departs from and returns to the depot at most once per period, and flow conservation holds for every visited customer. Constraints (5) enforce vehicle capacity limits. Constraints (6)–(7) calculate arrival times under the no-wait requirement, while also remove subtours. We enforce time propagation with a big-M linearization; a safe and tight choice

is $M = H$, the route completion limit. Constraint (8) bounds arrival times within the route completion limit. Constraints (9)–(10) enforce the arrival-time diversification requirement across periods by ensuring that the difference between the arrival times of a customer in two distinct periods is at least ϵ. If $z_i^{pp'} = 1$, the model enforces a minimum separation of ϵ with $a_i^p > a_i^{p'}$, whereas if $z_i^{pp'} = 0$, the minimum separation is enforced in the opposite direction, i.e., $a_i^{p'} > a_i^p$. Finally, constraints (11)–(14) define the variable domains.

Although route departure times are not explicit decision variables in the compact formulation, they are implicitly deduced from the arrival-time variables. In any feasible solution, if vehicle $k \in K$ is used in period $p \in P$, let j denote the first customer visited by vehicle k in that period, i.e., the unique customer satisfying $x_{0j}^{kp} = 1$. Then, under the no-waiting assumption, the departure time of route (k, p), defined as σ_{kp}, is given by

$$\sigma_{kp} = a_j^p - t_{0j}.$$

Hence, departure times are fully determined by the routing decisions and the arrival-time variables. They are therefore not needed as additional variables in the compact formulation.

4.1. Symmetry-Breaking Constraints

To reduce equivalent solutions among identical vehicles, we impose the following families of symmetry-breaking constraints.

$$\sum_{i \in C} y_i^k \geq \sum_{i \in C} y_i^{k+1}, \quad \forall k = 1, \ldots, K-1 \tag{15}$$

$$\sum_{j \in C_{p^*}} j\, x_{0j}^{kp^*} \geq \sum_{j \in C_{p^*}} j\, x_{0j}^{k+1p^*} - |V| \,(\sum_{i \in C} y_i^k - \sum_{i \in C} y_i^{k+1}), \quad \forall k = 1, \ldots, K-1 \tag{16}$$

Constraints (15) order vehicles according to the number of customers they serve, ensuring that vehicle k cannot visit fewer customers than vehicle $k+1$. Constraints (16) introduce a lexicographic tie-breaker based on the index of the first customer visited in the reference period $p^* \in P$, guaranteeing a unique ordering even when two vehicles perform routes of equal length. We select p^* as $p^* = \arg\max_{p \in P} |C_p|$.

5. Branch and Check Approach

This section presents a Branch-and-Check algorithm, a special case of Logic-Based Benders Decomposition (Hooker and Ottosson, 2003), in which feasibility cuts are generated within a branch-and-bound framework following the Branch-and-Check methodology of Thorsteinsson (2001). Our approach separates the problem into a master problem ($\mathcal{MP}$) and a subproblem ($\mathcal{SP}$): the master assigns customers to vehicles and determines route sequences across periods under capacity and duration constraints, while the subproblem checks feasibility by detecting subtours and, if none are found, verifying whether feasible route departure times can be assigned so that the induced arrival times satisfy the arrival-time diversification condition across periods. In this subproblem, route departure times are

auxiliary variables: once the master problem fixes the route sequence, each arrival time is obtained by adding the corresponding route offset to the departure time. Thus, the subproblem checks feasibility with respect to the arrival-time variables a_i^p of the compact formulation. To improve the efficiency of the search, the symmetry-breaking inequalities introduced in Section 4.1 are also incorporated into the master problem, helping to reduce significantly the number of generated cuts. Algorithm 1 summarizes the overall Branch-and-Check scheme.

Algorithm 1: Branch-and-Check for the VRPDC-ATD

```
Build the master problem MP                                   // Section 5.1
Solve MP within a branch-and-bound framework
foreach integer incumbent solution (x̄, ȳ) generated during the search do
    cutsAdded ← false
    foreach k ∈ K do
        Apply SP(k) to the routes induced by (x̄, ȳ)           // Section 5.2
        if SP(k) adds a cut to MP then
            cutsAdded ← true
    if cutsAdded = false then
        Accept (x̄, ȳ) as feasible for the VRPDC-ATD
return best feasible solution
```

5.1. Master Problem Formulation

Our initial $\mathcal{MP}$ keeps the binary variables x_{ij}^{kp} and y_i^k and preserves the objective function (1). The model assigns customers to vehicles and defines the route sequence for each period, subject to the connectivity, capacity, and symmetry-breaking constraints (2)–(5) and (15)–(16), as well as the domain definitions (12)–(13).

5.2. Sub Problem Algorithm

The subproblem $\mathcal{SP}$ admits a vehicle-wise decoupling. Once the master problem has fixed the assignment and routing variables, each customer is associated with a single vehicle over the planning horizon. Consequently, subtour detection and arrival-time diversification checks can be performed independently for each vehicle. Following the terminology of Hooker (2024), we refer to this structure as a decoupled subproblem. For each vehicle $k \in K$, we denote the resulting vehicle-level subproblem by $\mathcal{SP}(k)$. Algorithm 2 describes the corresponding procedure.

For a fixed vehicle k, $\mathcal{SP}(k)$ first detects subtours within each period using a breadth-first search (BFS) and adds the corresponding Subtour Elimination Constraints (SECs) when needed (see Section 5.2.1). If no subtours are found, the algorithm verifies whether feasible route departure times from the depot can be assigned across periods so that the induced arrival times of the customers assigned to vehicle k satisfy the arrival-time diversification constraints. This check is formulated as a multi-period disjunctive scheduling problem and

addressed through a Constraint Programming model, denoted as $\mathcal{CP}$ (see Section 5.2.2). If $\mathcal{CP}$ is infeasible, a Benders feasibility cut is generated for vehicle k.

Algorithm 2: $\mathcal{SP}(k)$

```
foreach period p ∈ P do
    detect subtours in the route of (k, p) using BFS
    if a subtour is found then
        add SEC (17) for (k, p) to MP
if cuts were added to MP then
    return
build CP for vehicle k
if CP is infeasible then
    add Benders feasibility cut (22) to MP
```

5.2.1. Check for Subtours

Two families of valid inequalities are added during the search to strengthen the master formulation. The first corresponds to a variant of the classical SEC, given by Inequality (17), which are generated for a specific vehicle k and period p whenever a disconnected route is detected in the current solution.

$$\sum_{i,j \in S} x_{ij}^{kp} \leq \sum_{i \in S} y_i^k - 1 + \frac{1}{|S|} \sum_{k' \in K \setminus \{k\}} \sum_{i \in S} y_i^{k'}, \quad \forall\, S \subseteq C_p,\ 2 \leq |S| \tag{17}$$

The summation over $k' \in K \setminus \{k\}$ acts as a lifting term. The cut is generated from a subtour detected for vehicle k, but after it is added to $\mathcal{MP}$, later incumbent solutions may assign only part of S to vehicle k. The additional term accounts for customers in S assigned to other vehicles and preserves validity in those cases. If all customers in S are assigned to vehicle k, the term is zero and the inequality reduces to the classical SEC. If none of them is assigned to vehicle k, the right-hand side becomes zero, which is consistent with vehicle k not serving any customer in S. The proof of this cut is provided in Theorem 1.

Theorem 1. For any subset $S \subseteq C_p$ and vehicle $k \in K$ in period $p \in P$, the Inequality (17) is valid.

Proof. Let $c = \sum_{i \in S} y_i^k$ denote the number of customers in S assigned to vehicle k. If vehicle k serves c customers in S, it can traverse at most $c-1$ arcs between nodes of S, which leads to the bound $\sum_{i,j \in S} x_{ij}^{kp} \leq c-1$. This bound strengthens the classical SEC when $0 < c < |S|$, but becomes invalid when $c = 0$.

To obtain a valid inequality for all values of c without introducing additional variables, we exploit the assignment structure. Since each customer in S is assigned to exactly one vehicle, it holds that $\sum_{k' \in K \setminus \{k\}} \sum_{i \in S} y_i^{k'} = |S| - c$. Adding a fraction of these assignments to the bound $c - 1$ yields

$$c - 1 + \frac{1}{|S|} \sum_{k' \in K \setminus \{k\}} \sum_{i \in S} y_i^{k'}.$$

Hence, Inequality (17) is valid: it coincides with the classical SEC when $c = |S|$, yields a zero bound when $c = 0$, and provides a valid relaxation otherwise. □

5.2.2. Check for Timing conflicts

The second family corresponds to Benders feasibility cuts, defined in Inequality (22), which are generated whenever the temporal subproblem associated with a given vehicle is found infeasible. This infeasibility is detected by solving a Constraint Programming model.

Given a vehicle $k \in K$, let $\bar{P}_k \subseteq P$ denote the set of periods in which k serves at least one customer. For each period $p \in \bar{P}_k$, let the fixed sequence of customers served by vehicle k be $R_{kp} = (0, i_1, i_2, \ldots, 0)$, and let $\bar{C}_{kp}$ denote the corresponding set of customers visited in this sequence. We define T_i^{kp} as the cumulative travel and service time from the depot to customer i along sequence R_{kp}, and D_{kp} as the total duration of this sequence.

To verify whether a solution of the master problem ($\mathcal{MP}$) satisfies the arrival-time diversification requirements, we use a constraint programming model. All parameters of this model are directly derived from the $\mathcal{MP}$ solution. The decision variables are continuous and non-negative: σ_{kp} represents the route departure time of vehicle k in period p, and $\hat{a}_i^p$ denotes the arrival time at customer i in period p. The resulting Constraint Programming formulation ($\mathcal{CP}$) determines feasible route departure times and checks temporal diversification feasibility for customers served across multiple periods.

$$\hat{a}_i^p = \sigma_{kp} + T_i^{kp}, \quad \forall\, p \in \bar{P}_k,\ i \in \bar{C}_{kp} \tag{18}$$

$$\sigma_{kp} \leq H - D_{kp}, \quad \forall\, p \in \bar{P}_k \tag{19}$$

$$(\hat{a}_i^p - \hat{a}_i^{p'} \geq \epsilon) \ \vee\ (\hat{a}_i^{p'} - \hat{a}_i^p \geq \epsilon), \quad \forall\, p < p' \in \bar{P}_k,\ i \in \bar{C}_{kp} \cap \bar{C}_{kp'} \tag{20}$$

$$\sigma_{kp} \in \mathbb{R}_{\geq 0}, \quad \hat{a}_i^p \in \mathbb{R}_{\geq 0}, \quad \forall\, p \in \bar{P}_k,\ i \in \bar{C}_{kp} \tag{21}$$

$\mathcal{CP}$ seeks feasibility only. Constraint (18) propagates arrival times according to fixed no-wait offsets, while constraint (19) ensures that each route fits within the route completion limit H. The disjunctive constraint (20) enforces arrival-time diversification across periods by requiring a minimum separation ϵ between visits of any customer served multiple times. Constraints (21) define the continuous domains of the route departure time and arrival-time variables.

This feasibility problem is NP-complete, as it can be formulated as a Disjunctive Temporal Problem (DTP). Each arrival-time diversification constraint introduces a disjunction on the temporal difference between two periods, and feasibility requires selecting one orientation for each such disjunction (20). In the worst case, each customer served in P periods induces up to $P(P-1)/2$ disjunctions, leading to an exponential number of combinations (Tsamardinos and Pollack, 2003). In practice, however, the number of periods per vehicle is small—typically $P \leq 5$—which keeps the effective search space manageable. Similar multi-period settings are considered in previous studies on the VRP with arrival-time diversification (Michallet et al., 2014; Hoogeboom and Dullaert, 2019; Soriano et al., 2020) and on the ConVRP variants (Groër et al., 2009; Goeke et al., 2019; Stavropoulou, 2022).

If $\mathcal{CP}$ is found to be infeasible for a given vehicle k and period p, the corresponding master solution contains temporal conflicts. In this case, the set of active arcs in the current solution is denoted by $\bar{A}_{kp}$, and Inequality (22) is added to the master problem to prevent the same infeasible combination of arcs from reappearing in subsequent iterations.

$$\sum_{(i,j)\in\bar{A}_{kp}} x_{ij}^{kp} \leq |\bar{A}_{kp}| - 1, \quad \forall\, k \in K,\ p \in P \tag{22}$$

6. Matheuristic

This section presents the matheuristic Hybrid Annealing with Variable Neighborhood Descent and memory-based Shake (HAVS) to address the VRPDC-ATD. An overview can be found in Algorithm 3. HAVS starts by generating an initial solution $\mathcal{S}$ using a template-based insertion–repair heuristic supported by $\mathcal{CP}$ as a temporal feasibility check (Section 6.2). Then a Local Search, denoted as Hybrid SA-VND, is applied. This mechanism relies on two complementary components: Simulated Annealing (SA), following the classical meta-heuristic of Kirkpatrick et al. (1983), and Variable Neighborhood Descent (VND), which is rooted in the Variable Neighborhood Search framework of Mladenović and Hansen (1997) and its deterministic Neighborhood-Descent variants (Hansen and Mladenović, 2001). Due to the inter-period dependencies induced by driver consistency, inter-vehicle moves are computationally expensive. Relocating a customer between vehicles changes its assignment over the entire planning horizon and therefore requires evaluating insertion positions in the destination vehicle's routes across all periods in which the customer must be served. As a result, the number of candidate insertion configurations grows exponentially with the number of active periods for that customer (Section 6.3). SA is therefore used to control both the selection and acceptance of inter-vehicle moves, reducing the number of costly inter-vehicle relocation evaluations while still allowing temporary deteriorations of the objective value in order to explore structurally relevant configurations that are inaccessible under strict improvement rules. VND performs deterministic intra-vehicle refinement through swap and relocate neighborhoods, restoring local solution quality after each accepted inter-vehicle move.

At each iteration n, the algorithm either resets the working solution to the best-so-far solution $\mathcal{S}^\star$ every γ iterations or applies a memory-based shaking mechanism to promote diversification (Section 6.5). The shaking mechanism targets customers that repeatedly induce infeasible inter-vehicle relocations. For each customer, the algorithm records the number of failed inter-vehicle move attempts caused by temporal conflicts. During the shaking phase, customers with the highest infeasibility counters are selectively removed from their current routes and reinserted. After the reset or shaking step, the algorithm applies the Hybrid SA-VND procedure to the current solution. The algorithm repeats this procedure for $N_{\max}$ iterations and returns the best solution $\mathcal{S}^\star$ at termination.

Algorithm 3: High-level structure of HAVS

```
Build an initial incomplete solution S                                  // Section 6.2
if S is feasible and κ(S) ≤ K then
    Set S* ← S
else
    Set S* ← ∅
Apply HybridSA-VND(S*, S)                                               // Section 6.3
for n = 1, 2, ..., N_max do
    if n mod γ = 0 and S* ≠ ∅ then
        Set S ← S*
    else
        Apply memory-based shaking from insertion failures              // Section 6.5
    Apply HybridSA-VND(S*, S)
if S* ≠ ∅ then
    return S*
else
    return no feasible solution found
```

6.1. A Solution Representation

We introduce an incomplete solution representation that separates routing from departure time decisions. The routing component consists of assigning customers to vehicles and defining their sequence per period, subject to capacity and no-waiting constraints. This corresponds to the decisions in $\mathcal{MP}$ and defines a Multi-Period VRP structure in which each vehicle has a fixed subset of customers and an ordered sequence per period, but no departure times. We refer to this as an incomplete solution.

The departure-time decisions are then obtained through a per-vehicle $\mathcal{CP}$ model that enforces arrival-time diversification. Given the fixed sequences, the $\mathcal{CP}$ either finds feasible departure times, completing the solution, or proves that no feasible timing exists, in which case the incomplete solution is discarded.

For an incomplete solution $\mathcal{S}$, let $\kappa(\mathcal{S})$ denote the number of vehicles serving at least one customer in $\mathcal{S}$. An incomplete solution may temporarily satisfy customer assignment and routing requirements while using more vehicles than the available fleet size. However, a solution is feasible for the VRPDC-ATD only if, after feasible departure times have been found, it also satisfies the fleet-size restriction $\kappa(\mathcal{S}) \leq K$. Therefore, the best solution $\mathcal{S}^\star$ is updated only with solutions satisfying all feasibility requirements, including $\kappa(\mathcal{S}) \leq K$.

6.2. An Initial Solution Heuristic

Based on this representation, we construct an initial incomplete solution using a template-based insertion–repair heuristic. Each vehicle is assigned a single template route into which customers are inserted while maintaining capacity feasibility across periods, and per-period routes are obtained by projecting the template order onto each period (Groër et al., 2009;

Stavropoulou, 2022). The $\mathcal{CP}$ model is then applied independently to each vehicle to determine feasible departure times that satisfy arrival-time diversification. If the $\mathcal{CP}$ fails, the vehicle is repaired by removing customers from its template route following their visit order until feasibility is restored. The removed customers are then evaluated for reinsertion into existing vehicles. Each candidate insertion is checked against capacity and completion limit constraints in all affected periods, and the $\mathcal{CP}$ model is then solved for the affected vehicle to verify whether feasible departure times satisfying arrival-time diversification still exist.

If no feasible reinsertion is found, an additional vehicle may be temporarily created to keep all customers assigned. This strategy follows a complete-first construction logic and is motivated by the difficulty of deciding feasibility under a fixed fleet size. A failed insertion into the current set of K vehicles does not prove that no feasible solution exists with K vehicles. It only reflects the current partial route structure. This distinction is important because, when $|P| = 1$, $\epsilon = 0$, and the temporal diversification constraints are inactive, the problem contains the fixed-fleet CVRP as a special case. Since the CVRP is strongly NP-hard, such a failure cannot be used as a certificate of fixed-fleet infeasibility (Lenstra and Rinnooy Kan, 1981). The construction then continues until all vehicles in the current incomplete solution admit feasible departure times.

The additional vehicle is only a temporary device used during construction. If it causes $\kappa(\mathcal{S}) > K$, the resulting incomplete solution is not feasible for the VRPDC-ATD and cannot update $\mathcal{S}^\star$. The subsequent local search attempts to remove any excess vehicles through the route-elimination criterion described in Section 6.3.

6.3. A Hybrid Local Search

Due to the driver-consistency requirement, relocating a customer across vehicles requires evaluating feasible insertion positions in every period in which the customer is served, which leads to a combinatorial explosion of candidate configurations. For this reason, inter-vehicle exploration in our method is guided by SA, while VND is used exclusively for intensification within vehicles. Algorithm 4 summarizes this hybrid SA-VND procedure and the interaction between exploration and exploitation.

The algorithm starts by initializing the temperature at $T_{\max}$, which denotes the initial SA temperature. The temperature is then gradually decreased until it reaches $T_{\min}$, the final SA temperature, over the maximum number of SA iterations $I_{\max}$. At iteration n, we use the exponential cooling schedule

$$T = T_{\max} \left(\frac{T_{\min}}{T_{\max}} \right)^{n/I_{\max}}.$$

At each iteration, the algorithm considers pairs of vehicles (k_o, k_d) and alternates between inter-vehicle swap and relocate moves depending on the iteration index. To limit the number of inter-vehicle moves evaluated at each iteration, candidate customers are selected from a restricted subset referred to as a batch. The batch is constructed using two complementary mechanisms: a granular neighborhood criterion and a long-term memory component. For

each customer i, a score

$$\eta_i = \zeta_i + \beta\alpha_i$$

is computed, where ζ_i denotes the number of granular neighbors of i in the origin vehicle, and α_i counts how many times customer i has previously been selected for inter-vehicle evaluation. The parameter $\beta \in [0, 1]$ is a continuous weight that controls the influence of the long-term memory component. When $\beta = 0$, the batch selection is based only on the granular neighborhood criterion. When $\beta = 1$, the memory counter receives its full weight relative to the granularity term, penalizing customers that have already been evaluated frequently. Intermediate values provide a gradual trade-off between these two effects. Only the $\theta\%$ customers with the lowest scores are included in the batch, so the search prioritizes customers whose reassignment is structurally relevant and has not been repeatedly explored.

For each customer (or customer pair), inter-vehicle candidate moves are generated by a neighborhood iterator that systematically explores insertion positions across the affected periods. A detailed description of the neighborhood iterator is provided in Appendix B. Each candidate move is first evaluated by computing its cost variation on the incomplete solution. This variation is then used in the SA acceptance criterion. Specifically, a candidate move with cost variation Δ is accepted if its probability

$$p(\Delta, T) = \begin{cases} 1, & \text{if } \Delta \leq 0, \\ \exp(-\Delta/T), & \text{if } \Delta > 0, \end{cases}$$

where T denotes the current temperature, and satisfies $u \leq p(\Delta, T)$, with $u \sim \mathcal{U}(0, 1)$.

In addition to this cost-based acceptance rule, SA uses a route-elimination criterion when the current incomplete solution uses more than K vehicles. A candidate move is also allowed to proceed to the feasibility checks if it empties its origin vehicle while $\kappa(\mathcal{S}) > K$. The empty vehicle is not removed at this stage. The move is first checked with respect to capacity, route duration, and arrival-time diversification. Only after the move is accepted and feasibility is confirmed, the empty vehicle is removed, and only if $\kappa(\mathcal{S}) > K$. Thus, the algorithm removes only excess vehicles and never deletes vehicles from a solution that already satisfies the fleet-size restriction.

The next step is to verify whether, for the current incomplete solution, there exist route departure times that satisfy the arrival-time diversification constraints. Although this verification can be performed using the $\mathcal{CP}$ model, integrating it within the local search would be computationally expensive. For this reason, we adopt a feasibility verification scheme based on a two-stage procedure (Section 6.4). First, a global infeasibility test discards moves that cannot satisfy arrival-time diversification for any choice of departure times. Second, if a move passes this test, its feasibility is verified using a local dynamic programming procedure $\mathcal{DP}$.

Only moves that are feasible under both checks are applied to the current solution. Once an inter-vehicle move is accepted, the solution is immediately intensified using VND on the affected vehicles. During this phase, VND evaluates intra-vehicle relocate and swap moves using the same incomplete-solution cost criterion and the same route-elimination criterion, and applies the same two-stage feasibility verification. If VND produces a new solution that

satisfies all feasibility requirements, including $\kappa(\mathcal{S}) \leq K$, the best solution $\mathcal{S}^\star$ is updated and the SA process is restarted from the initial temperature. At the end of each SA iteration, the temperature is updated according to an exponential cooling schedule. After completing all iterations, the algorithm returns the best solution $\mathcal{S}^\star$ found during the search.

Algorithm 4: `HybridSA-VND`

```
T ← T_max
S ← VND(S, K̄)
Update S* if S is feasible and κ(S) ≤ K
for n = 1, 2, ..., I_max do
    foreach vehicle pair (k_o, k_d) do
        if n is odd then
            Compute batches B_o and B_d from k_o and k_d
        else
            Compute batch B_o from k_o
        Initialize neighborhood iterator while next move m exists do
            Compute cost variation Δ
            if SAAccepted(Δ, T) then
                if GlobalInfeasibilityCheck(S, m) then
                    if LocalFeasibilityCheck(S, m) then
                        S ← VND(S ⊕ m, k_o, k_d)
                        Remove empty excess vehicle
                        Update S* only if κ(S) ≤ K
                        if S* is updated then
                            T ← T_max, n ← 1
                        break to next SA iteration
    T ← T_max × (T_min/T_max)^(n/I_max)
return S*
```

6.4. An Arrival-Time Diversification Feasibility Check

First, a global infeasibility check is applied to discard incomplete solutions that violate arrival-time diversification for any feasible choice of departure times. Then, a local Dynamic Programming-based feasibility check ($\mathcal{DP}$) is performed for a single vehicle of an incomplete solution (Section 6.4.2). In this procedure, the relative order of arrival times across periods is fixed for each customer according to the current solution, while the route departure times and arrival-time values remain decision variables. Under a fixed order, the feasibility of departure times is determined through a compact system of difference constraints involving only the departure times. Feasibility of this system is verified using the Bellman–Ford algorithm.

All notation in the remainder of this section is consistent with our $\mathcal{CP}$ model introduced in Section 5.2.2. In particular, σ_{kp} denotes the departure time of vehicle k in period p, T_{ip}

the cumulative travel and service time from the depot to customer i along the fixed route in period p, and D_{kp} the total duration of this route.

6.4.1. Global Infeasibility Check for Arrival-Time Diversification

To accelerate the verification process, we first perform a preliminary global check of arrival-time diversification before applying $\mathcal{DP}$. This check is termed *global* because, unlike $\mathcal{DP}$, it guarantees that any solution it rejects is completely infeasible with respect to diversification constraints.

Algorithm 5 presents this feasibility check. For each customer and each pair of periods, we analyze the extreme values that the arrival-time difference can attain over the rectangular domain defined by the departure-time bounds. If the maximum achievable separation is strictly smaller than ϵ, then no feasible choice of route departure times can satisfy the diversification constraint, and the solution is declared globally infeasible. The procedure terminates as soon as a violation is detected, ensuring computational efficiency.

Algorithm 5: Global infeasibility check via pairwise arrival-time

Input: a single vehicle of an incomplete solution k
Output: **true** if globally infeasible, **false** otherwise

foreach customer i assigned to k **do**
  let P_i be the set of periods in which i is served
  foreach pair of periods $(p, p') \in P_i$ with $p < p'$ **do**
    compute T_{ip} and $T_{ip'}$
    compute departure-time ranges $\sigma_{kp} \in [0,\ H - D_{kp}]$, $\sigma_{kp'} \in [0,\ H - D_{kp'}]$
    evaluate $\Gamma = (\sigma_{kp'} - \sigma_{kp}) + T_{ip'} - T_{ip}$ at the four corner points
    let $[\Gamma^{\min}, \Gamma^{\max}]$ be the resulting interval
    if $\max(|\Gamma^{\min}|, |\Gamma^{\max}|) < \epsilon$ **then**
      return true
return false

6.4.2. A Local feasibility check via order fixing

If the global infeasibility check described above does not detect infeasibility, we proceed with the local feasibility check. At this point, the algorithm has generated a candidate move m, but this move has not yet been accepted. Hence, the object being tested is the tentative incomplete solution $\mathcal{S} \oplus m$. For a given affected vehicle k, let $\bar{P}_k^m$ denote the set of periods in which vehicle k is active after temporarily applying move m. The route sequence of vehicle k in each period $p \in \bar{P}_k^m$ is kept fixed during the check. Arrival times in the tentative solution can therefore be written as $a_i^p = \sigma_{kp} + T_{ip}^m$, where T_{ip}^m is the route offset from the depot departure time to the arrival time at customer i in period p, after temporarily applying move m. The quantities σ_{kp} are not known at this stage. They are the departure-time variables whose feasibility is verified by the local check.

Arrival-time diversification is originally enforced in our $\mathcal{CP}$ model through the Disjunctive Constraint (20). Solving this disjunctive problem exactly inside the local search would be computationally expensive. We therefore perform a local check by fixing one branch of each

disjunction. The branch is selected using the departure times of the current solution $\mathcal{S}$, before the temporary move is applied.

More precisely, let $\bar{\sigma}_{kp}$ denote the stored departure time of vehicle k in period p in the current solution $\mathcal{S}$. If vehicle k is not active in period p in $\mathcal{S}$, we set $\bar{\sigma}_{kp} = 0$ by convention. These values are already available because $\mathcal{S}$ is the solution from which the candidate move is generated. They are not the departure times of the tentative solution $\mathcal{S} \oplus m$. They are used only to construct reference arrival times

$$\rho_i^p(m) = \bar{\sigma}_{kp} + T_{ip}^m.$$

For any customer i served by vehicle k in both periods p and p', with $p < p'$, we preserve the reference order induced by $\rho_i^p(m)$ and $\rho_i^{p'}(m)$. Hence, we select the corresponding branch of (20) as follows:

$$\rho_i^p(m) \leq \rho_i^{p'}(m) \Rightarrow a_i^{p'} - a_i^p \geq \epsilon, \qquad \rho_i^p(m) > \rho_i^{p'}(m) \Rightarrow a_i^p - a_i^{p'} \geq \epsilon.$$

Substituting $a_i^p = \sigma_{kp} + T_{ip}^m$, each fixed disjunction becomes a linear difference constraint involving only the departure times. If the reference order satisfies $\rho_i^p(m) \leq \rho_i^{p'}(m)$, then we impose

$$a_i^{p'} - a_i^p \geq \epsilon \Rightarrow (\sigma_{kp'} + T_{ip'}^m) - (\sigma_{kp} + T_{ip}^m) \geq \epsilon \Rightarrow \sigma_{kp} - \sigma_{kp'} \leq T_{ip'}^m - T_{ip}^m - \epsilon.$$

Conversely, if $\rho_i^p(m) > \rho_i^{p'}(m)$, then we impose

$$a_i^p - a_i^{p'} \geq \epsilon \Rightarrow \sigma_{kp'} - \sigma_{kp} \leq T_{ip}^m - T_{ip'}^m - \epsilon.$$

Thus, every fixed-order diversification constraint can be written in the generic form $\sigma_{kv} - \sigma_{ku} \leq w_{uv}$, where u and v are period indices. The scalar w_{uv} is the right-hand side of the corresponding difference constraint. It is not an additional input parameter. More precisely, the two possible cases are:

$$\rho_i^p(m) \leq \rho_i^{p'}(m) \Rightarrow (u, v) = (p', p), \qquad w_{uv} = T_{ip'}^m - T_{ip}^m - \epsilon,$$

$$\rho_i^p(m) > \rho_i^{p'}(m) \Rightarrow (u, v) = (p, p'), \qquad w_{uv} = T_{ip}^m - T_{ip'}^m - \epsilon.$$

The resulting system of difference constraints is represented as a directed graph. The graph contains one node for each active period $p \in \bar{P}_k^m$ and one dummy node ψ. A difference constraint is represented by an arc (u, v) with weight w_{uv}. If several customers generate constraints with the same ordered pair (u, v), we keep only the smallest weight, since it defines the tightest constraint and dominates the others.

The departure-time bounds are incorporated in the same graph. Let $U_{kp}^m = H - D_{kp}^m$ be the latest feasible departure time of vehicle k in period p after applying the tentative move m, where D_{kp}^m is the duration of the tentative route. The bounds

$$0 \leq \sigma_{kp} \leq U_{kp}^m$$

are written as difference constraints with respect to the dummy node ψ. The upper bound is $\sigma_{kp} - \sigma_{\psi} \leq U^m_{kp}$, and is represented by an arc (ψ, p) with weight U^m_{kp}. The lower bound is $\sigma_{\psi} - \sigma_{kp} \leq 0$, and is represented by an arc (p, ψ) with weight 0.

Thus, the resulting graph follows the standard transformation of systems of difference constraints into shortest-path problems. Under this transformation, each constraint $x_v - x_u \leq w_{uv}$ is represented by an arc (u, v) with weight w_{uv}, and feasibility is characterized by the absence of negative cycles (Cormen et al., 2009). We therefore solve the local feasibility problem using the Bellman–Ford algorithm, denoted by $\mathcal{DP}$, as summarized in Algorithm 6.

If a negative cycle is detected, no departure times can satisfy all fixed-order diversification constraints and the departure-time bounds induced by the route durations for the tentative routing structure of vehicle k. Otherwise, the Bellman–Ford labels are normalized with respect to the dummy node ψ, and the final departure times for the tentative solution are obtained as $\sigma_{kp} = b_p - b_{\psi},\ p \in \bar{P}^m_k$. This normalization fixes $\sigma_{\psi} = 0$ and guarantees that the returned values satisfy $0 \leq \sigma_{kp} \leq U^m_{kp}$. Since the graph contains $O(|\bar{P}^m_k|)$ nodes and $O(|\bar{P}^m_k|^2)$ arcs, the Bellman–Ford feasibility check runs in $O(|\bar{P}^m_k|^3)$ time.

6.5. Diversification Mechanisms

Although SA provides diversification through probabilistic acceptance and VND intensifies the search, the combined process may still become trapped in local minima. To mitigate this effect, we introduce a shaking mechanism guided by feasibility failures observed during the search.

Because departure-time feasibility is verified over a local search space, repeated failures may occur. We exploit this information by temporarily removing a fraction of the customers with the highest number of insertion failures during the SA phase. The proportion of removed customers is defined as $\nu/|P|$, where $\nu \in [0, 1]$ is a parameter. Removed customers are then considered for reinsertion into other vehicles, which are prioritized according to the number of closest neighbors associated with each customer.

Customer removal may render some vehicles infeasible. Feasibility is first restored using $\mathcal{DP}$, which is computationally inexpensive. If infeasibility persists, our $\mathcal{CP}$ model is then applied to explore a broader feasibility space. When both checks fail, an additional customer is removed and the process is repeated, preserving as much of the current route structure as possible, until all routes become feasible.

Once feasibility is restored, removed customers are reinserted, and the feasibility of each reinsertion is checked using $\mathcal{DP}$. If a customer cannot be inserted into any vehicle other than its original one, it is skipped. A new vehicle is created only if all reinsertion attempts fail, ensuring that all customers remain assigned. If this creates an incomplete solution with $\kappa(\mathcal{S}) > K$, the additional vehicle is treated as temporary and the solution cannot update $\mathcal{S}^{\star}$ until a subsequent local-search move eliminates the excess vehicle.

Algorithm 6: Bellman–Ford feasibility check for departure times

Input: Current solution $\mathcal{S}$, candidate move m, affected vehicle k, active periods $\bar{P}_k^m$, tentative route durations D_{kp}^m, tentative route offsets T_{ip}^m, stored departure times $\bar{\sigma}_{kp}$, and separation parameter ϵ

Output: Feasible departure times $(\sigma_{kp})_{p\in\bar{P}_k^m}$ for $\mathcal{S}\oplus m$, or infeasibility

Build a directed graph with node set $\{\psi\}\cup\bar{P}_k^m$
Set $b_v \leftarrow 0$ for all nodes $v \in \{\psi\}\cup\bar{P}_k^m$
foreach $p \in \bar{P}_k^m$ **do**
    Set $U_{kp}^m \leftarrow H - D_{kp}^m$
    Add arc (ψ, p) with weight U_{kp}^m `// upper bound` $\sigma_{kp} \le U_{kp}^m$
    Add arc (p, ψ) with weight 0 `// lower bound` $\sigma_{kp} \ge 0$
foreach customer i assigned to vehicle k in $\mathcal{S}\oplus m$ **do**
    Let $P_i^{k,m} \subseteq \bar{P}_k^m$ be the set of periods in which i is served
    foreach pair of periods $p, p' \in P_i^{k,m}$, with $p < p'$ **do**
        Compute $\rho_i^p(m) \leftarrow \bar{\sigma}_{kp} + T_{ip}^m$
        Compute $\rho_i^{p'}(m) \leftarrow \bar{\sigma}_{kp'} + T_{ip'}^m$
        if $\rho_i^p(m) \le \rho_i^{p'}(m)$ **then**
            Set $u \leftarrow p'$, $v \leftarrow p$, and $w \leftarrow T_{ip'}^m - T_{ip}^m - \epsilon$
        else
            Set $u \leftarrow p$, $v \leftarrow p'$, and $w \leftarrow T_{ip}^m - T_{ip'}^m - \epsilon$
        if arc (u, v) already exists **then**
            Set its weight to $\min\{w_{uv}, w\}$
        else
            Add arc (u, v) with weight $w_{uv} \leftarrow w$
for $r = 1, \ldots, |\bar{P}_k^m|$ **do**
    foreach arc (u, v) with weight w_{uv} **do**
        $b_v \leftarrow \min\{b_v,\ b_u + w_{uv}\}$
foreach arc (u, v) with weight w_{uv} **do**
    if $b_v > b_u + w_{uv}$ **then**
        return infeasible
foreach $p \in \bar{P}_k^m$ **do**
    Set $\sigma_{kp} \leftarrow b_p - b_\psi$
return feasible with departure times $(\sigma_{kp})_{p\in\bar{P}_k^m}$

7. Computational Results

This section presents the computational evaluation of the proposed methods. Throughout this section, the threshold separation is defined as $\epsilon = H\xi$, where H denotes the route completion limit and $\xi \in [0, 1]$ represents the relative separation level.

All algorithms were implemented in C++. The compact MILP formulation and the $\mathcal{MP}$ of the Branch-and-Check algorithm were solved using IBM ILOG CPLEX Optimization Studio 22.1.1. The $\mathcal{CP}$ feasibility subproblems used in LBBD and in HAVS were solved using the CP Optimizer component of the same optimization suite. All experiments were

conducted on a Windows 11 machine equipped with a 13th Gen Intel(R) Core(TM) i7-1365U processor at 1.80 GHz and 16 GB of RAM.

The parameter tuning phase and the small- and medium-scale experiments involving HAVS were performed using 10 independent runs to account for stochastic variability. The large-scale HAVS experiments were performed using 5 independent runs. All HAVS results reported in the computational tables correspond to feasible VRPDC-ATD solutions, satisfying the fleet-size, capacity, route-completion, driver-consistency, and arrival-time diversification constraints. A maximum time limit of 3 hours was imposed for both Branch-and-Check (LBBD) and MILP. Instances reaching this limit are denoted as "TL".

We begin by describing the data set and the parameter tuning procedure. We then evaluate the exact formulations on small-scale instances to validate modeling correctness and solution quality. Next, we compare LBBD and HAVS on medium-scale instances, where computational scalability becomes critical. For large-scale instances, the focus shifts to the performance and robustness of HAVS, as exact approaches become impractical. Finally, we derive managerial insights and analyze how routing structures adapt as diversification requirements increase.

7.1. Data Set

We use the benchmark instances from Groër et al. (2009), originally proposed for the ConVRP. Small instances are characterized by $|C| \in [10, 12]$ customers, $|P| = 3$ periods, vehicle capacity $Q = 15$, and route completion limit $H = 35$. For the large-scale instances, we consider the first five problems, with $|C| \in [50, 199]$, $Q \in [140, 200]$ and $|P| = 5$. Here, we evaluate planning horizons with $|P| \in \{3, 4\}$. This choice follows related VRP with Arrival-Time Diversification studies (Michallet et al., 2014; Hoogeboom and Dullaert, 2019; Soriano et al., 2020).

In the original data set, route duration is unrestricted. As this renders arrival-time diversification trivially feasible, we impose route completion limits $H \in [150, 180]$ to analyze the impact of time constraints on feasibility and diversification. Throughout the computational experiments, we label the five selected problems from Groër et al. (2009) as Instances 1–5. For the medium-scale experiments, each instance is restricted to the first |C| customers, with |C| ranging from 20 to 40 considering planning horizons with $|P| = 3$.

When reducing the planning horizon of the large-scale instances (originally defined with $|P| = 5$) to $|P| < 5$ periods, we select the first P periods of the original instance. If a customer has zero demand over these selected periods, their service requirements from subsequent periods are shifted forward to the reduced horizon.[1]

7.2. Parameter tuning

Parameter tuning is performed using a group-based strategy. Five medium-scale instances with $|P| = 3$ periods are used for calibration, and each configuration is evaluated with 10 random seeds. Algorithmic parameters are grouped by component and include the

[1]E.g., period-demand $(0, 0, 0, 5, 5)$ becomes $(5, 5, 0)$ for $P = 3$ and $(0, 0, 0, 5)$ for $P = 4$.

overall number of iterations ($N_{\max}$), the simulated annealing parameters ($T_{\min}, T_{\max}, I_{\max}$), the granularity threshold (π), the long-term memory and batch parameters (β, θ), the shaking procedure parameter (ν), and the return-to-best parameter (γ). These groups are tuned sequentially (Order), fixing previously calibrated parameters.

The parameter $N_{\max}$ controls the global search budget of HAVS. Larger values are therefore expected to improve or preserve solution quality, but at the cost of additional runtime. This trade-off is particularly relevant in multi-period settings, where increasing the planning horizon increases the number of inter-period feasibility interactions and, as shown later in the large-scale experiments, leads to substantially longer runtimes. We therefore tune $N_{\max}$ within a predefined computational budget and select the best-performing value in that range. The complete set of explored ranges and selected parameter values is reported in Table 1.

Regarding the $\mathcal{CP}$ feasibility model, although it is exact, it may require excessive computational effort during the search. Therefore, we limit its exploration by imposing a maximum number of failed solutions in the search τ_{fail}. This threshold was calibrated in a final tuning phase. If the limit is reached, the search is terminated and the configuration is treated as infeasible (i.e., feasibility is not determined).

Table 1: Parameter ranges explored and final values

Component	Order	Parameter	Range	Best
Global	1	$N_{\max}$	$[500, 1000]$	1000
SA	2	$I_{\max}$	$[500, 1250]$	1000
		$T_{\min}$	$[0.5, 2.5]$	2.0
		$T_{\max}$	$[10, 20]$	15
Granularity	3	π	$[0.3, 0.5]$	0.35
Long-term memory	4	β	$[0.5, 0.7]$	0.65
		θ	$[0.3, 0.5]$	0.45
Shaking	5	ν	$[0.3, 0.6]$	0.375
Return-to-best	6	γ	$[5, 25]$	10
Max Fails in $\mathcal{CP}$	7	τ_{fail}	$[10^3, 10^6]$	10^5

7.3. Comparison Between MILP and LBBD on small-scale instances

Table 2 reports the performance of the compact MILP formulation and the LBBD approach (Branch-and-Check) on small-scale instances. Throughout this analysis, we use a relative separation level $\xi = 10\%$. The first two columns indicate the number of customers ($|C|$) and the maximum fleet size (K). For each method, Opt/Tot denotes the number of instances solved to optimality over the total number tested, Gap(%) is the average optimality gap at termination, and T(s) represents the average computational time in seconds. For the exact methods, the upper bound, lower bound, and optimality gap are obtained from CPLEX at termination. The last row reports averages computed over all instances.

Table 2: Computational performance of MILP and LBBD on small instances

$\|C\|$	K	MILP			LBBD		
		Opt/Tot	Gap(%)	T(s)	Opt/Tot	Gap(%)	T(s)
10	2	4/4	0.0	214.2	4/4	0.0	188.9
12	2	5/5	0.0	1508.1	5/5	0.0	375.1
20	3	8/10	1.9	2825.8	9/10	0.5	1169.3
25	3	0/10	9.0	TL	9/10	1.0	1911.2
Avg			3.6	4993.4		0.5	1153.0

Detailed results are provided in Table C.2. For the smallest instances ($|C| \leq 12$), both approaches solve all instances to optimality with zero gap. Nevertheless, LBBD requires less computational time, particularly for $|C| = 12$. Thus, even at small scale, the decomposition yields computational advantages. When the problem size increases ($|C| = 20$), performance differences become more pronounced. LBBD solves more instances to optimality, achieves a smaller average gap, and requires substantially less computational time. Finally, for $|C| = 25$, MILP fails to solve any instance to optimality, reaches the time limit, and reports an average gap of 9.0%. In contrast, LBBD solves 9 of 10 instances to optimality with a significantly smaller gap (1.0%). These results indicate that although both methods perform similarly on very small instances, the LBBD approach exhibits superior robustness and scalability as instance size increases.

Table 3: Comparison between LBBD and the variant without symmetry-breaking constraints.

Instance	H	$\|C\|$	K	LBBD			No symmetry breaking		
				UB	Gap(%)	T(s)	UB	Gap(%)	T(s)
1	150	30	3	929.0	0.0	1517.6	929.0	0.0	2318.3
1	180	20	3	703.3	0.0	5.8	703.3	0.0	3.7
2	150	25	3	956.2	0.0	633.7	956.2	0.0	610.3
2	180	20	3	766.0	0.0	9.8	766.0	0.0	12.7
3	150	30	3	1006.3	0.0	2287.0	1006.3	0.0	2761.4
3	180	30	3	917.9	0.0	1918.9	917.9	0.8	TL
4	150	30	3	935.8	0.0	6829.2	935.8	0.0	8751.4
4	180	35	3	1029.6	0.0	3855.9	1029.6	0.0	5276.1
5	150	35	3	979.2	0.0	9546.7	979.2	0.6	TL
5	180	35	3	965.9	0.0	4697.2	965.9	0.0	2913.9
Avg					0.0	3130.2		0.1	4424.8

We also evaluate the impact of the symmetry-breaking constraints introduced in Section 4.1. Table 3 compares the full LBBD implementation with a variant in which these constraints are removed. Both versions obtain the same upper bounds in all tested instances, which confirms that the added inequalities do not modify the feasible solution space. On average, the full LBBD reduces runtime by 29.3% relative to the variant without symmetry breaking and avoids two time-limit cases observed without symmetry breaking. Although

the full version is not faster in every individual instance, the overall effect is positive. By reducing redundant permutations of identical vehicles, the symmetry-breaking constraints help the master problem focus on distinct assignments and improve the average efficiency of the Branch-and-Check scheme.

7.4. Comparison Between LBBD and HAVS on medium-scale instances

Table 4 compares LBBD and HAVS on medium-scale instances under a relative separation level of $\xi = 10\%$, grouped by route completion limit H and number of customers $|C|$. In Table 4, LBBD is evaluated through Opt/Tot, the average optimality gap at termination (Gap(%)), and the average runtime T(s). For HAVS, $\text{Gap}_{avg}(\%)$ measures the relative deviation between the average solution over 10 runs and the lower bound (LB) provided by LBBD,

$$\text{Gap}_{avg}(\%) = \left(\bar{z}^{\text{HAVS}} - LB^{\text{LBBD}}\right) / LB^{\text{LBBD}},$$

while Gap_{max} is defined analogously using the maximum solution cost of HAVS.

The indicator $\Phi(\%)$ measures the relative deviation of the average HAVS solution from the LBBD upper bound:

$$\Phi(\%) = \left(\bar{z}^{\text{HAVS}} - UB^{\text{LBBD}}\right) / UB^{\text{LBBD}}.$$

Since the objective is minimized, positive values of Φ indicate that the average HAVS solution is above the LBBD upper bound, whereas negative values indicate that HAVS improves upon the incumbent solution found by LBBD. Finally, $\Lambda(\%)$ reports the relative range across runs, defined as the difference between the worst and best solution values divided by the best value.

Table 4: Performance of LBBD and HAVS on medium-scale instances under different route completion limits and customer set sizes.

H	$\|C\|$	LBBD			HAVS					
		Opt/Tot	Gap(%)	T(s)	Opt/Tot	$\text{Gap}_{max}(\%)$	$\text{Gap}_{avg}(\%)$	$\Phi(\%)$	$\Lambda(\%)$	T(s)
150	20	4/5	1.0	2310.9	4/5	0.8	0.7	-0.4	0.3	38.2
	25	4/5	2.0	3382.2	4/5	0.7	0.5	-1.6	0.5	60.0
	30	4/5	0.7	4476.0	4/5	0.9	0.7	0.0	0.2	70.2
	35	1/5	9.7	10549.5	1/5	6.2	5.7	-4.8	0.8	81.6
	40	0/5	16.6	TL	0/5	13.5	12.7	-6.3	1.1	96.8
Avg/Sum		13/25	6.0	6303.8	13/25	4.4	4.1	-2.6	0.6	69.4
180	20	5/5	0.0	27.7	5/5	0.2	0.1	0.0	0.2	39.3
	25	5/5	0.0	440.3	3/5	0.3	0.1	0.1	0.3	60.5
	30	4/5	0.9	3432.3	4/5	1.4	1.0	0.0	0.6	74.3
	35	2/5	5.1	10350.7	2/5	5.0	4.4	-1.1	0.7	82.3
	40	0/5	13.4	TL	0/5	10.6	10.1	-5.0	0.7	92.5
Avg/Sum		16/25	3.9	5010.2	14/25	3.5	3.1	-1.2	0.5	69.8

For the instances solved to optimality by LBBD, HAVS reaches the corresponding optimal solution in 27 of 29 cases. For $H = 150$, LBBD performs strongly for $|C| \leq 30$, where

most instances are solved to optimality and gaps remain below 2%. However, performance deteriorates for $|C| = 35$ and 40, with only 1/5 and 0/5 optimal solutions, respectively, and larger gaps. HAVS shows a more gradual increase in the reported gap: Gap_{avg} remains below 1% for $|C| \leq 30$ and increases to 5.7% and 12.7% for $|C| = 35$ and 40. This increase is mainly due to weaker lower bounds from LBBD for $|C| > 30$. Furthermore, the values of Λ remain small in all configurations, indicating low variability across runs and therefore stable, even for the largest instances. In addition, HAVS requires significantly less computational time in all cases.

For $H = 180$, the additional slack markedly improves LBBD performance on sizes $|C| \leq 25$, where all instances are solved to optimality, and keeps good performance at $|C| = 30$. Nonetheless, for $|C| = 35$ and 40, LBBD again struggles to close the gap within the given limit. Aggregated over all sizes, the average gap of LBBD decreases from 6.0% ($H = 150$) to 3.9% ($H = 180$), yet HAVS still attains a slightly lower overall average gap (3.1%) while remaining consistently faster.

To further explain these results, Table 5 reports the callback-level effort of LBBD. The columns $\bar{t}_{sec}$ and $\bar{t}_B$ measure the average time spent in SEC separation and in the Constraint Programming temporal feasibility check, respectively, while $\bar{c}_{sec}$ and $\bar{c}_B$ report the corresponding average number of generated cuts. These times do not include the time spent by the branch-and-bound search of the $\mathcal{MP}$. Detailed results are provided in Tables C.3–C.4.

Table 5: LBBD computational effort by instance size and route duration

$\|C\|$	$H = 150$				$H = 180$			
	$\bar{t}_{sec}$	$\bar{t}_B$	$\bar{c}_{sec}$	$\bar{c}_B$	$\bar{t}_{sec}$	$\bar{t}_B$	$\bar{c}_{sec}$	$\bar{c}_B$
20	0.1	68.9	277	2700	0.0	10.9	202	220
25	0.0	29.7	403	2169	0.0	32.1	391	575
30	0.0	52.0	550	113	0.0	33.5	549	547
35	0.1	39.7	1022	514	0.0	65.1	737	70
40	0.1	6.2	1010	46	0.1	358.1	1039	74
Avg	0.1	39.3	652	1108	0.0	99.9	584	297

Consistent with the previous discussion, $H = 150$ induces a more restrictive solution space than $H = 180$, as reflected by larger average gaps and, in general, a higher number of Benders cuts. As H becomes non-binding, arrival-time diversification no longer requires structural route modifications, and the solution coincides with the baseline Multi-Period VRP with driver consistency for any ϵ.

The interaction between structure and diversification is evident for $|C| = 20, 25$, and 30. For $|C| = 20$ and 25, LBBD generates many Benders cuts, indicating that the solution under diversification deviates more from the baseline and that numerous infeasible master solutions must be excluded. In contrast, at $|C| = 30$, the number of cuts decreases substantially, which suggests fewer violations of the diversification constraints and a closer structural alignment

with the baseline. A structural explanation is given by the minimum fleet size,

$$\left\lceil \frac{\max_{p \in P} \sum_{i \in C} q_i^p}{Q} \right\rceil .$$

For $|C| = 30$, this value is typically three vehicles, enabling a more balanced customer distribution and greater temporal flexibility. For $|C| = 20$ and 25, the minimum is usually two vehicles, resulting in longer and more constrained routes, which increases the likelihood of diversification infeasibilities. In addition, the spatial distribution of additional customers may further influence this behavior, as certain configurations can facilitate temporal separation more naturally than others.

For larger instances ($|C| = 35, 40$), LBBD rarely reaches optimality. Table 5 shows that the time spent in SEC separation and temporal feasibility checks remains only a fraction of the total runtime reported in Table 4. Hence, most of the total runtime is associated with the branch-and-bound exploration of the $\mathcal{MP}$, where routing and assignment decisions become increasingly difficult. Nevertheless, the temporal check is still relevant for understanding the cost of each callback. In particular, for $H = 180$ and $|C| = 40$, the number of Benders cuts is small, but the average time spent in temporal feasibility checking increases. This suggests that larger and less restrictive routes may make temporal checks more expensive, even though they are not the dominant component of the total LBBD runtime.

7.5. Computational Results of HAVS on Large-Scale Instances

This section describes how expanding the planning horizon affects the behavior of HAVS, emphasizing its impact on solution quality, feasibility checking, and the distribution of computational effort across algorithmic components. Throughout this analysis, we use a relative separation level $\xi = 10\%$.

Table 6 reports the computational results for $|P| = 3$ and $|P| = 4$, including the average number of vehicles used ($\bar{K}$), the best solution value obtained, the variability indicator $\Lambda(\%)$ computed over five HAVS runs, and the average runtime $T(s)$ for each instance and route completion limit H. We also compute the relative separation between the average and best objective values:

$$\Delta_{\text{avg}}(\%) = 100 \times \frac{\overline{z} - z^{\min}}{z^{\min}},$$

where $\overline{z}$ is the average objective value over the five independent runs and $z^{\min}$ is the best value observed. Detailed results are given in Table C.5.

Table 7 presents the time distribution of HAVS across its main components. $T_{Prune+\mathcal{DP}}$(s) denotes the average time allocated to feasibility evaluation through the global infeasibility check (Algorithm 5) and the dynamic programming procedure (Algorithm 6). $T_{\mathcal{CP}}$(s) corresponds to the time required to solve $\mathcal{CP}$ models. The column #Prune(%) reports the percentage of candidate evaluations avoided through early pruning by the global infeasibility check mechanism. The percentages $T_{\text{SA-VND}}$ and $T_{\mathcal{CP}}$ indicate the share of total runtime allocated to local search and $\mathcal{CP}$ solving, respectively.

Increasing the planning horizon from $|P| = 3$ to $|P| = 4$ is associated with an average cost increase of about 37%. This effect is consistent across instance sizes and duration limits.

Although the cost escalation is stable, the additional period tightens the feasible search space by reinforcing inter-period dependencies and activating more temporal constraints, which increases the proportion of infeasible candidate moves and the complexity of feasibility verification, as well as increases the run time.

Table 6: Computational performance of HAVS on large-scale instances under different planning horizons.

H	Inst.	$\|C\|$	$\|P\|=3$					$\|P\|=4$				
			$\bar{K}/K$	Best	$\Lambda(\%)$	$\Delta_{\text{avg}}(\%)$	$T(s)$	$\bar{K}/K$	Best	$\Lambda(\%)$	$\Delta_{\text{avg}}(\%)$	$T(s)$
150	1	50	4/5	1233.8	0.8	0.3	104.5	4/5	1692.3	1.3	0.9	449.1
	2	75	8/10	2144.8	1.1	0.5	114.4	9/10	2864.1	3.4	1.5	333.8
	3	100	6/10	2142.5	3.6	1.7	298.6	6/10	2841.4	8.0	3.6	2412.4
	4⋆	150	10/15	2917.7	7.3	4.1	366.2	10/15	4176.2	3.6	1.9	2869.1
	5⋆	199	13/15	3629.2	8.4	4.2	497.0	13/15	5317.8	4.3	2.5	3657.8
180	1	50	4/5	1238.9	0.9	0.3	99.5	4/5	1686.8	1.8	0.9	560.8
	2	75	8/10	2113.5	1.9	1.0	62.9	9/10	2854.3	3.3	1.4	222.0
	3	100	6/10	2138.4	1.9	0.8	236.3	6/10	2741.8	6.5	3.3	2048.5
	4⋆	150	12/15	3009.5	3.0	1.9	368.7	10/15	3901.8	7.5	2.8	1937.1
	5⋆	199	14/15	3672.1	6.7	2.6	360.4	14/15	5160.3	2.0	1.2	2310.6
Avg					3.6	1.7	246.8			4.2	2.0	1680.1

(*) Customers with zero demand in all periods were assigned random positive demands and service periods.

Despite this, the runtime of our Hybrid SA–VND does not grow in proportion to the restrictiveness of the problem. Feasibility is evaluated through our dynamic programming procedure ($\mathcal{DP}$), which keeps the additional computational effort relatively low. As shown in Table 7, the proportion of time devoted to local search remains close to 54% on average, increasing by only about 2% under $|P| = 4$, while the average difference in total $\mathcal{DP}$ time, together with Algorithm 5, between horizons is just 0.2 s. This confirms that the $\mathcal{DP}$ evaluator effectively prevents local search from becoming the dominant computational bottleneck. The global check for infeasibility also becomes more effective as the horizon expands. Stronger inter-period coupling enables earlier detection of infeasible combinations, yielding evaluation reductions above 90% in several $|P| = 4$ instances. Thus, the increased restrictiveness of the problem enhances the pruning capability of the algorithm. These results indicate that the main limiting factor is the number of candidate evaluations performed, particularly in inter-vehicle moves. In other words, the computational effort is driven more by evaluation frequency than by evaluation complexity.

The other contributor to the overall increase in runtime is the $\mathcal{CP}$ model used in the shaking phase, which accounts for roughly 40% of the total runtime. While $\mathcal{CP}$ preserves structural feasibility and avoids destructive perturbations, it becomes the primary source of overhead as the planning horizon grows.

In addition, fleet usage remains largely stable when moving from three to four periods, with the exception of Instance 4, where a slight decrease occurs, indicating that fleet size has limited influence on the behavior of the Hybrid SA–VND. This deviation stems from the reduced-horizon instance construction (Section 7.1). Under $|P| = 3$, more customers

exhibit zero demand within the selected horizon, triggering demand shifting and altering route compactness and fleet utilization. With $|P| = 4$, fewer customers meet this condition, leading to a different redistribution pattern and, consequently, a different fleet structure.

Finally, the dispersion measure $\Lambda(\%)$ remains moderate, with average values of 3.6% for $|P| = 3$ and 4.2% for $|P| = 4$. This suggests some sensitivity to the stochastic search process, but not an unstable behavior. This interpretation is reinforced by $\Delta_{\text{avg}}(\%)$, which remains low on average: 1.7% for $|P| = 3$ and 2.0% for $|P| = 4$. Hence, the average solution quality stays close to the best solution found across independent runs, indicating that the method does not rely on isolated best runs.

Table 7: Detailed time decomposition of HAVS under different planning horizons.

$\|P\|$	Inst.	$T_{Pair+\mathcal{DP}}$(s)	$T_{\mathcal{CP}}$(s)	#Pair(%)	$T_{\text{SA-VND}}$(%)	$T_{\mathcal{CP}}$(%)
3	1	0.0	28.4	44.8	70.4	28.3
	2	0.0	31.5	5.5	60.6	37.8
	3	0.0	107.7	72.7	58.0	40.4
	4	0.0	221.5	51.7	37.4	60.3
	5	0.0	233.3	35.5	41.5	56.0
	Avg	0.0	124.5	42.1	53.6	44.6
4	1	0.2	166.1	88.5	66.1	32.9
	2	0.0	113.8	39.4	57.3	41.0
	3	0.6	796.5	97.2	62.8	35.7
	4	0.4	1166.6	96.1	48.2	48.5
	5	0.4	1526.4	92.4	43.8	51.1
	Avg	0.3	753.9	82.7	55.6	41.9

7.6. Managerial Insights

This section derives managerial implications from the computational results obtained with HAVS. We study the effect of different values of the relative separation level ξ through a sensitivity analysis on cost and fleet usage, and a structural analysis of route design under increasing diversification requirements. These perspectives highlight the operational trade-offs induced by temporal separation constraints.

7.6.1. Sensitivity Analysis with Respect to the Temporal Diversification Level

Figure 2 shows the relative cost increase with respect to the baseline solution without arrival-time diversification ($\epsilon = 0$) for our medium-scale instances based on Instance 3 from Groër et al. (2009) with $H = 180$. Each curve corresponds to a customer size $|C| \in [20, 40]$, and the two panels distinguish between $|P| = 3$ and $|P| = 5$ periods.

For $|P| = 3$, costs remain almost unchanged for $\xi < 10\%$. At $\xi = 10\%$, increases are between 1.3% and 2.5% for $|C| \in [20, 35]$, while no increase is observed for $|C| = 40$. This indicates that moderate diversification can be absorbed without major structural changes. Beyond $\xi = 10\%$, costs begin to rise more visibly. At $\xi = 15\%$, increases range from 2.2%

to 7.7%. The growth becomes steeper for $\xi \geq 20\%$, particularly for $|C| = 30$. At $\xi = 30\%$, cost increases reach 43.3%, 52.9%, and 67.4% for $|C| = 20, 25, 30$, respectively.

The effect is considerably stronger for $|P| = 5$. Even at $\xi = 5\%$, some instances already show noticeable increases (up to 3.9% for $|C| = 35$). At $\xi = 10\%$, costs rise to 14.1% and 11.6% for $|C| = 25$ and $|C| = 30$, and exceed 30% at $\xi = 15\%$. This confirms that distributing visits over more periods makes diversification significantly more restrictive and therefore increases the deviation from the baseline solution.

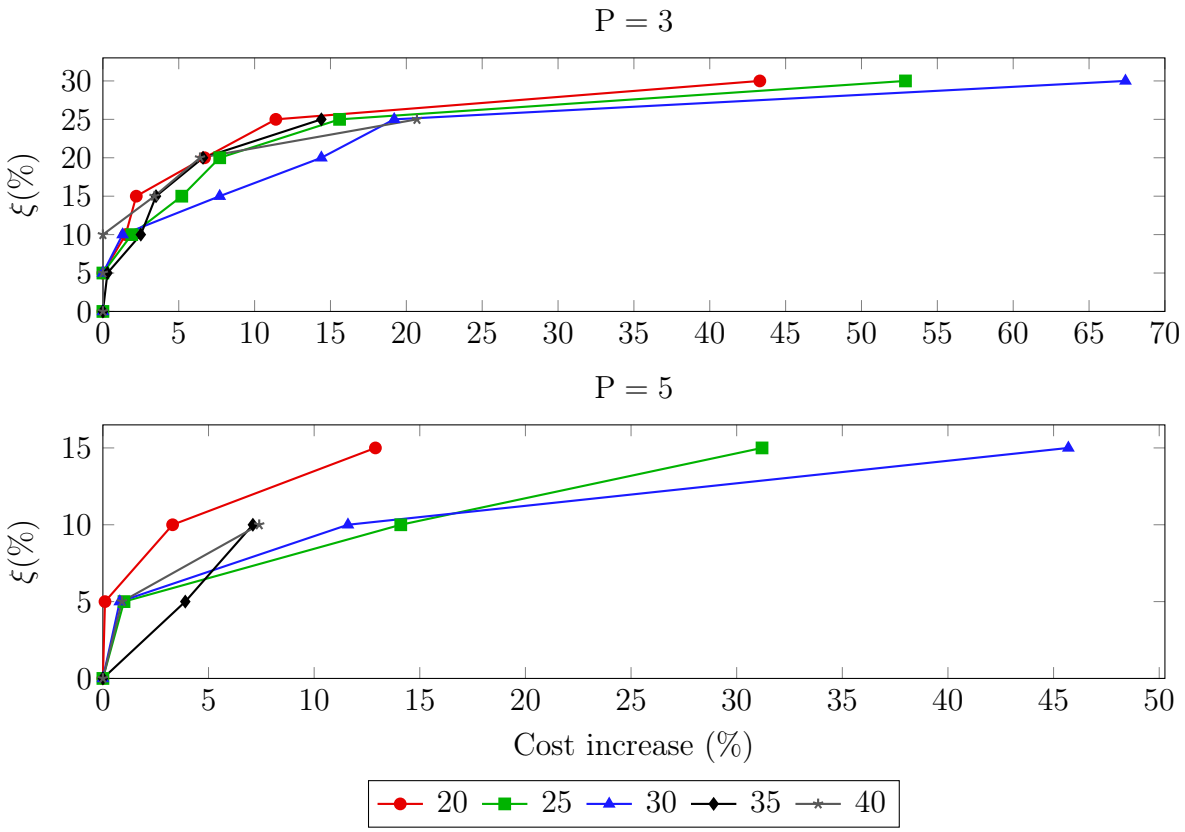

Figure 2: Cost sensitivity to arrival-time diversification across customer sizes, medium-size Instance 3.

Another important aspect in interpreting these results is the fleet size. Figure 3 provides additional insight, as each cell reports both the cost increase (through color intensity) and the number of vehicles used (embedded values). Blank cells indicate configurations for which HAVS was not able to find a feasible solution. The heatmap shows that introducing an additional vehicle leads to a substantial increase in cost, with average jumps of around 22.5%. The instance with $|C| = 35$ is particularly illustrative: although the diversification level ξ increases, the fleet size remains unchanged over a wide range, and the corresponding cost increase remains stable. In contrast, once a higher diversification level requires an additional vehicle, the cost increase becomes significantly larger. The same behaviour can be observed for the other customer sizes up to the point where the fleet size changes. This indicates that the main cost increases are associated with changes in fleet size rather than

with gradual adaptations of the routing structure. From a managerial perspective, the results suggest that maintaining the same number of vehicles stabilizes costs, even under increasing diversification requirements. The main cost shifts are driven by fleet size adjustments rather than by gradual modifications of route structures.

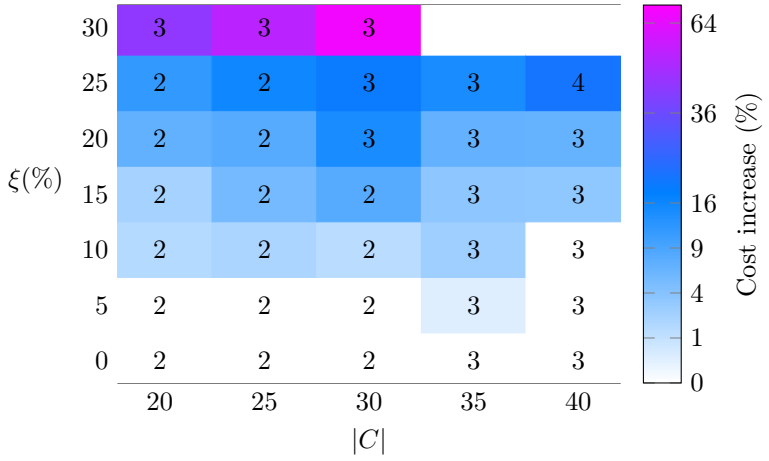

Figure 3: Impact of arrival-time diversification on routing cost and fleet size ($|P| = 3$).

Finally, diversification feasibility is structurally bounded by the planning horizon. Since route departure times lie in $[0, H]$ and arrival times are propagated without waiting, the $|P|$ visits of a customer must fit within an interval of length at most H. Enforcing a minimum separation ϵ between periods implies

$$(P - 1)\,\epsilon \leq H.$$

With $H = 180$, this yields $\epsilon \leq 90$ ($\xi = 50\%$) for $P = 3$ and $\epsilon \leq 45$ ($\xi = 25\%$) for $P = 5$. Although the tested values remain below these limits, the curves show that costs increase sharply well before reaching the theoretical bound, as fleet size and horizon constraints tighten simultaneously. For larger values of ξ beyond those tested, no feasible solutions were found.

7.7. Structural Response to Increasing Diversification Levels

To illustrate how solutions evolve as the diversification level ξ increases, we analyse a representative Instance 3 with 25 customers and track the changes in its routing structure across periods (Figure 4). Four main adaptation mechanisms can be identified:

- Adjustment of route departure times (orange interlined circles). For example, at $\xi = 10\%$, the green vehicle in periods 0–2 follows $0 \rightarrow 6$, and diversification is obtained by shifting the departure times.
- Reassignment of customers across existing routes (green interlined circles).
- Use of an additional vehicle, when the maximum fleet size is not reached (observed at $\xi = 30\%$).

- Changes in route patterns, involving a reorganization of customer sequences (blue interlined circles).

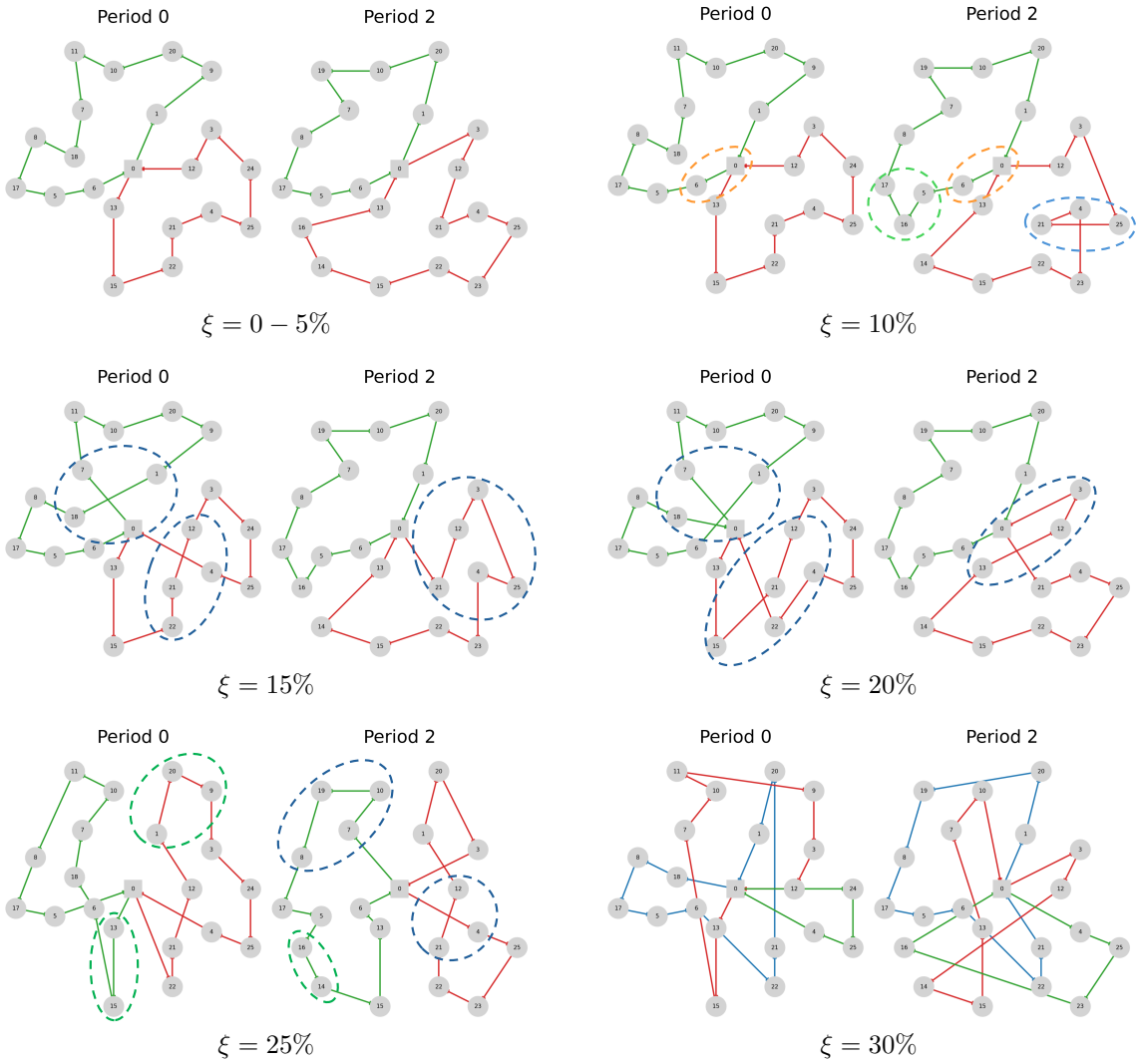

Figure 4: Evolution of routing patterns in Instance 3 ($|C| = 25$) as the diversification level increases. Circles indicate the changes compared to the previous diversification level.

The solution at $\xi = 0\%$ serves as the baseline configuration. For moderate diversification levels, such as the transition to $\xi = 10$–15%, most routing structures remain visually similar across periods. At this stage, the dominant mechanism is the adjustment of route departure times, complemented by limited customer reassignment. Route patterns exhibit only minor modifications, indicating that the algorithm exploits available temporal slack before altering the underlying structure.

For medium diversification levels (e.g., $\xi = 20$–25%), departure time adjustments and small structural changes are no longer sufficient. At this point, the algorithm must choose

between introducing more substantial modifications in route sequences or reassigning customers, or activating an additional vehicle (e.g., $\xi = 30\%$). This decision is not straightforward: small adjustments may be cheaper than opening a new vehicle, but only up to a certain threshold.

As ξ increases further, as shown at $\xi = 30\%$, structural changes alone are no longer enough. When the maximum fleet size has not yet been reached, the solution introduces an additional vehicle. This fleet expansion acts as a feasibility strategy and, in some cases, as a buffer that preserves structural similarity while absorbing the diversification requirement.

Once ξ increases beyond this point and the maximum fleet size is reached, adding vehicles is no longer possible. The solution must then rely on joint customer reassignment and more pronounced pattern changes. Route sequences differ more clearly from the baseline, and similarities across periods decrease. At this stage, the routing structure progressively resembles a peripatetic strategy, where patterns vary more strongly across periods and the probability of finding feasible solutions decreases.

8. Conclusions and Future Research

This paper addressed a Multi-Period Vehicle Routing Problem with driver consistency and arrival-time diversification. While arrival-time diversification has been studied previously to reduce temporal predictability, this paper is one of the first to integrate arrival-time diversification with driver consistency in a unified multi-period formulation. The joint enforcement of fixed driver assignments and inter-period temporal separation creates strong cross-period coupling, reducing the feasible region and significantly increasing computational complexity.

Two complementary solution approaches were developed: a logic-based Benders decomposition (LBBD) and a Hybrid SA–VND mathehuristic (HAVS). LBBD performs effectively on small-scale instances and remains competitive on medium instances under different route completion limits. However, as instance size or temporal restrictiveness increases, computational effort concentrates in the Benders phase, limiting scalability. HAVS, in contrast, shows strong robustness and scalability, consistently producing high-quality solutions with low variability across medium and large-scale instances.

The results indicate that the effect of diversification depends not only on the relative separation level ξ, but also on structural factors such as fleet size, route completion limits, planning horizon, and spatial customer distribution. Increasing the planning horizon intensifies inter-period coupling and tightens feasibility, leading to higher costs and, in some cases, provable infeasibility. Structural properties, particularly minimum fleet requirements and route balance, play a central role in mitigating the cost impact of arrival time-diversification requirements.

A promising direction for future research is the development of a Branch-and-Price algorithm based on multi-period clusters proposed by Goeke et al. (2019). Each column would represent a vehicle schedule over the entire planning horizon, rather than single-period routes. The associated pricing problem would correspond to a multi-period TSP with arrival-time diversification constraints. This pricing subproblem could be effectively

addressed using our LBBD approach, which is well suited to handling the temporal feasibility component while maintaining routing optimality.

Another potential extension concerns the dynamic programming ($\mathcal{DP}$) feasibility procedure. The evaluation order could be adapted dynamically, for example by prioritizing routes or periods according to structural characteristics observed during the search or based on the managerial insights identified in this study. Such adaptive ordering strategies may reduce computational effort by focusing resources on the most critical components of the solution.

Acknowledgments

The third author is partially funded by the Danish Council for Independent Research - Social Sciences. Project 'ComRoute' [grant number 3099-00003B]. This support is gratefully acknowledged.

References

Alvarez, A., Cordeau, J.F., Jans, R., 2024. The consistent vehicle routing problem with stochastic customers and demands. Transportation Research Part B: Methodological 186, 102968.

Braekers, K., Kovacs, A.A., 2016. A multi-period dial-a-ride problem with driver consistency. Transportation Research Part B: Methodological 94, 355–377.

Calvo, R.W., Cordone, R., 2003. A heuristic approach to the overnight security service problem. Computers & Operations Research 30(9), 1269–1287.

Coelho, L.C., Laporte, G., 2013. A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. International Journal of Production Research 51(23–24), 7156–7169.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., 2009. Introduction to Algorithms. MIT press.

De la Torre, L.E., Dolinskaya, I.S., Smilowitz, K.R., 2012. Disaster relief routing: Integrating research and practice. Socio-economic Planning Sciences 46(1), 88–97.

Duchenne, É., Laporte, G., Semet, F., 2007. The undirected m-peripatetic salesman problem: polyhedral results and new algorithms. Operations Research 55(5), 949–965.

Fröhlich, G.E., Gansterer, M., Doerner, K.F., 2023. Safe and secure vehicle routing: a survey on minimization of risk exposure. International Transactions in Operational Research 30(6), 3087–3121.

Gao, J., Zhu, J., Wang, J., Li, K., Zhen, L., Demeulemeester, E., 2026. The two-visit team orienteering problem considering time-interval-dependent profits and service consistency. Computers & Operations Research 188, 107370.

Goeke, D., Roberti, R., Schneider, M., 2019. Exact and heuristic solution of the consistent vehicle-routing problem. Transportation Science 53(4), 1023–1042.

Groër, C., Golden, B., Wasil, E., 2009. The consistent vehicle routing problem. Manufacturing & Service Operations Management 11(4), 630–643.

Hansen, P., Mladenović, N., 2001. Variable neighborhood search: Principles and applications. European Journal of Operational Research 130(3), 449–467.

Hoogeboom, M., Dullaert, W., 2019. Vehicle routing with arrival time diversification. European Journal of Operational Research 275(1), 93–107.

Hooker, J., 2024. Logic-Based Benders Decomposition: Theory and Applications. Springer.

Hooker, J.N., Ottosson, G., 2003. Logic-based benders decomposition. Mathematical Programming 96, 33–60.

Kirkpatrick, S., Gelatt Jr, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. science 220(4598), 671–680.

Kovacs, A.A., Golden, B.L., Hartl, R.F., Parragh, S.N., 2014a. Vehicle routing problems in which consistency considerations are important: A survey. Networks 64(3), 192–213.

Kovacs, A.A., Golden, B.L., Hartl, R.F., Parragh, S.N., 2015. The generalized consistent vehicle routing problem. Transportation Science 49(4), 796–816.

Kovacs, A.A., Parragh, S.N., Hartl, R.F., 2014b. A template-based adaptive large neighborhood search for the consistent vehicle routing problem. Networks 63(1), 60–81.

Kuo, C.F., Pang, A.C., Chan, S.K., 2008. Dynamic routing with security considerations. IEEE Transactions on Parallel and Distributed Systems 20(1), 48–58.

Lenstra, J.K., Rinnooy Kan, A.H.G., 1981. Complexity of vehicle routing and scheduling problems. Networks 11(2), 221–227.

Lindstrøem, C., 2022. Group-Consistent Dial-a-Ride Problems and Adaptive Large Neighborhood Search. Ph.D. thesis. Technical University of Denmark. Kongens Lyngby, Denmark.

Michallet, J., Prins, C., Amodeo, L., Yalaoui, F., Vitry, G., 2014. Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services. Computers & Operations Research 41, 196–207.

Mladenović, N., Hansen, P., 1997. Variable neighborhood search. Computers & Operations Research 24(11), 1097–1100.

Pedraza-Martinez, A.J., Van Wassenhove, L.N., 2016. Empirically grounded research in humanitarian operations management: The way forward. Journal of Operations Management 45, 1–10.

Soriano, A., Vidal, T., Gansterer, M., Doerner, K., 2020. The vehicle routing problem with arrival time diversification on a multigraph. European Journal of Operational Research 286(2), 564–575.

Stavropoulou, F., 2022. The consistent vehicle routing problem with heterogeneous fleet. Computers & Operations Research 140, 105644.

Talarico, L., Springael, J., Sörensen, K., Talarico, F., 2017. A large neighbourhood metaheuristic for the risk-constrained cash-in-transit vehicle routing problem. Computers & Operations Research 78, 547–556.

Tarantilis, C.D., Stavropoulou, F., Repoussis, P.P., 2012. A template-based tabu search algorithm for the consistent vehicle routing problem. Expert Systems with Applications 39(4), 4233–4239.

Thorsteinsson, E.S., 2001. Branch-and-Check: A hybrid framework integrating mixed integer programming and constraint logic programming, in: Walsh, T. (eds) Principles and Practice of Constraint Programming — CP 2001, Springer. pp. 16–30.

Tsamardinos, I., Pollack, M.E., 2003. Efficient solution techniques for disjunctive temporal reasoning problems. Artificial Intelligence 151(1–2), 43–89.

Appendix A. MILP Notation

Table A.1 summarizes the notation used in the compact MILP in Section 4. We distinguish between input data, derived sets used, and decision variables.

Table A.1: Notation used in the compact MILP formulation.

Sets and indices	
C	Set of customers.
P	Set of periods.
K	Set of available vehicles.
A	Set of directed arcs.
i, j	Customer or depot indices.
p, p'	Period indices.
k	Vehicle index.
Input parameters	
t_{ij}	Travel time from node i to node j.
q_i^p	Demand of customer i in period p.
s_i	Service time of customer i.
Q	Vehicle capacity.
H	Route completion limit.
ϵ	Minimum arrival-time separation between visits in different periods.
M	Big-M constant used in the time-propagation and diversification constraints.
Derived sets and parameters	
C^p	Set of customers with positive demand in period p, i.e., $C^p = \{i \in C : q_i^p > 0\}$.
A^p	Set of arcs available in period p, i.e., $A^p = \{(i,j) : i,j \in C^p \cup \{0\}, i \neq j\}$.
p^*	Reference period used in the symmetry-breaking constraints.
Decision variables	
x_{ij}^{kp}	1 if vehicle k traverses arc (i,j) in period p, and 0 otherwise.
y_i^k	1 if customer i is assigned to vehicle k, and 0 otherwise.
a_i^p	Arrival time at customer i in period p.
$z_i^{pp'}$	Binary variable selecting the ordering of the arrival-time separation between periods p and p'.

Appendix B. Neighborhood Iterator for Inter-Vehicle Moves

For each customer(or pair) selected in a batch(es), a neighborhood iterator enumerates insertion-position combinations across the periods affected by the inter-vehicle move. The iterator first identifies the set of periods that must be modified.

For each of these periods, insertion positions are considered between consecutive nodes of the route, starting from the position between the depot and the first customer. An insertion position is valid if it satisfies vehicle capacity and a granularity criterion, i.e., if at least one adjacent node is a neighbor of customer to be inserted. Insertion positions are explored in a lexicographic order across periods, advancing positions in one period at a time while keeping the others fixed.

In the case of swap moves, the iterator is applied only to periods in which a direct swap is not possible. Periods where both customers are served admit a fixed swap position and are therefore excluded from the lexicographic iteration. Once all combinations have been explored, the iterator proceeds to the next customer(or pair) in the batch(es).

Appendix C. Detailed Results

Table C.2: Average upper bound, computation time, and cut statistics for small instances with $|P| = 3$

		MILP		LBBD					
Instance	$\|C\|$	UB	t(s)	UB	t(s)	C^{sec}	C^{bd}	t^{sec}	t^{bd}
2	10	127.7	109.1	127.7	751.5	91	16357	0.2	257.5
3	10	149.6	719.0	149.6	2.6	109	150	0.0	1.9
4	10	151.0	17.0	151.0	1.0	78	45	0.0	0.8
5	10	131.0	11.7	131.0	0.4	48	5	0.0	0.2
6	12	172.1	1791.0	172.1	41.2	238	1905	0.0	20.7
7	12	111.1	3.5	111.1	0.7	71	24	0.0	0.4
8	12	155.7	4814.3	155.7	754.9	201	13652	0.1	191.1
9	12	166.0	428.9	166.0	17.2	134	988	0.0	12.9
10	12	144.4	502.7	144.4	1061.5	133	20121	0.2	254.7

Table C.3: Detailed LBBD results for medium-sized instances with $|P| = 3$ and $H = 150$

Instance	$\|C\|$	K	UB	LB	Gap(%)	t(s)	Nodes	c_{sec}	c_B	t_{sec}(s)	t_B(s)
	20	3	706.2	706.2	0.0	7.1	2078	205	63	0.0	2.5
	25	3	878.0	878.0	0.0	883.5	179939	329	43	0.0	3.2
1	30	3	929.0	929.0	0.0	1517.6	181205	405	22	0.0	28.6
	35	4	1079.9	976.4	9.6	TL	930000	722	50	0.0	4.2
	40	4	1195.6	1053.4	11.9	TL	367400	738	52	0.0	2.6
	20	3	766.0	766.0	0.0	11.5	5243	167	0	0.0	0.3
	25	3	956.2	956.2	0.0	633.7	135685	334	382	0.0	9.7
2	30	4	1031.5	995.7	3.5	TL	474010	631	17	0.0	67.0
	35	4	1250.5	1024.2	18.1	TL	873799	1132	56	0.1	96.7
	40	5	1309.2	1050.7	19.7	TL	299900	1242	37	0.1	2.9
	20	3	867.6	822.8	5.2	TL	2631911	381	9892	0.2	147.4
	25	3	988.8	988.8	0.0	1187.1	551296	369	220	0.0	12.5
3	30	3	1006.3	1006.3	0.0	2287.0	306465	657	274	0.0	9.2
	35	4	1266.3	1047.2	17.3	TL	623500	1265	338	0.1	11.3
	40	5	1570.4	1103.0	29.8	TL	333662	1161	41	0.2	3.0
	20	3	733.0	733.0	0.0	704.6	208082	378	3213	0.1	185.7
	25	3	907.7	816.2	10.1	TL	875100	578	6190	0.1	70.5
4	30	3	935.8	935.8	0.0	6829.2	904784	541	85	0.0	132.7
	35	3	1062.5	1024.6	3.6	TL	449209	1354	1406	0.2	56.6
	40	4	1265.0	1092.1	13.7	TL	278775	942	55	0.1	18.2
	20	3	714.7	714.7	0.0	31.3	13645	254	330	0.0	8.5
	25	3	897.7	897.7	0.0	3406.7	1166826	405	4009	0.1	52.5
5	30	3	933.8	933.8	0.0	946.2	116220	518	167	0.0	22.6
	35	3	979.2	979.2	0.0	9546.7	1506418	635	718	0.0	29.6
	40	4	1072.5	986.5	8.0	TL	401800	966	44	0.0	4.4

Table C.4: Detailed LBBD results for medium-sized instances with $|P| = 3$ and $H = 180$

Instance	$\|C\|$	K	UB	LB	Gap(%)	t(s)	Nodes	c_{sec}	c_B	t_{sec}(s)	t_B(s)
	20	3	703.3	703.3	0.0	5.8	2138	161	0	0.0	0.5
	25	3	878.0	878.0	0.0	540.7	95619	326	14	0.0	68.4
1	30	3	929.0	929.0	0.0	1115.4	149041	324	0	0.0	0.1
	35	4	1056.9	962.2	9.0	TL	435100	751	5	0.0	118.9
	40	4	1184.8	1048.4	11.5	TL	346300	686	18	0.0	1046.4
	20	3	766.0	766.0	0.0	9.8	4925	161	0	0.0	0.3
	25	3	945.3	945.3	0.0	425.6	81456	278	1	0.0	1.0
2	30	4	1037.3	991.4	4.4	TL	541096	575	0	0.0	0.9
	35	4	1180.5	1027.6	13.0	TL	772095	871	24	0.0	131.8
	40	5	1438.9	1037.6	27.9	TL	395231	1035	13	0.1	2.5
	20	3	798.6	798.6	0.0	61.9	24954	256	1062	0.0	20.9
	25	3	896.5	896.5	0.0	989.3	321108	592	2700	0.1	78.0
3	30	3	917.9	917.9	0.0	1918.9	381823	796	2553	0.1	72.0
	35	3	1117.4	1078.1	3.5	TL	755204	830	211	0.0	36.1
	40	4	1255.8	1116.9	11.1	TL	248970	1362	209	0.1	480.9
	20	3	722.4	722.4	0.0	44.4	4901	214	31	0.0	30.8
	25	3	813.5	813.5	0.0	161.3	22249	416	123	0.0	9.5
4	30	3	903.2	903.2	0.0	2527.4	309742	632	173	0.0	89.9
	35	3	1029.6	1029.6	0.0	3855.9	289087	652	88	0.0	23.6
	40	4	1181.2	1101.9	6.7	TL	272228	1152	101	0.1	255.7
	20	3	708.8	708.8	0.0	16.5	6805	218	5	0.0	2.0
	25	3	877.8	877.8	0.0	84.4	18421	345	35	0.0	3.6
5	30	3	932.6	932.6	0.0	799.7	118436	419	10	0.0	4.5
	35	3	965.9	965.9	0.0	4697.2	516474	581	21	0.0	14.9
	40	4	1094.9	985.1	10.0	TL	444400	960	30	0.0	4.8

Table C.5: Detailed HAVS results in medium-sized instances with different route duration limits H and $|P| = 3$.

		$H = 150$					$H = 180$				
Inst.	$\|C\|$	Best	Avg	T(s)	Gap(%)	OPT	Best	Avg	T(s)	Gap(%)	OPT
1	20	706.2	706.3	26.2	0.0	∗	703.3	703.3	30.7	0.0	∗
	25	878.0	878.0	48.9	0.0	∗	878.0	878.0	45.2	0.0	∗
	30	929.0	929.0	62.2	0.0	∗	929.0	929.0	56.5	0.0	∗
	35	1043.9	1047.0	60.6	-3.0	–	1041.0	1044.5	54.1	-1.2	–
	40	1147.5	1148.2	80.3	-4.0	–	1147.5	1148.4	64.5	-3.1	–
2	20	766.0	766.0	25.2	0.0	∗	766.0	766.0	22.7	0.0	∗
	25	956.2	956.4	42.5	0.0	∗	945.3	945.5	34.7	0.0	∗
	30	1031.5	1031.5	41.9	0.0	–	1031.5	1031.5	34.5	-0.6	–
	35	1126.9	1131.8	46.4	-9.5	–	1126.9	1130.2	43.0	-4.3	–
	40	1235.6	1237.4	49.0	-5.5	–	1235.6	1237.0	38.8	-14.0	–
3	20	843.9	849.3	48.5	-2.1	–	798.6	800.2	60.7	0.2	∗
	25	988.8	988.8	68.9	0.0	∗	897.2	900.4	98.9	0.4	x
	30	1006.3	1006.8	87.5	0.0	∗	917.9	924.3	115.9	0.7	∗
	35	1132.5	1134.9	109.4	-10.4	–	1114.3	1116.4	123.3	-0.1	–
	40	1305.8	1312.0	128.8	-16.5	–	1206.5	1214.1	124.3	-3.3	–
4	20	733.0	733.1	62.4	0.0	∗	722.4	722.4	51.2	0.0	∗
	25	826.0	833.0	93.5	-8.2	–	814.3	814.3	89.5	0.1	x
	30	935.8	936.3	99.5	0.1	∗	903.2	903.2	113.1	0.0	∗
	35	1042.8	1049.1	124.4	-1.3	–	1029.6	1030.2	125.8	0.1	∗
	40	1183.1	1199.7	142.8	-5.2	–	1153.5	1155.0	166.1	-2.2	–
5	20	714.7	714.7	28.9	0.0	∗	708.8	708.8	31.0	0.0	∗
	25	897.7	899.8	46.1	0.2	∗	877.8	877.9	34.3	0.0	∗
	30	933.8	934.1	60.1	0.0	∗	932.6	932.7	51.2	0.0	∗
	35	979.2	979.3	67.5	0.0	∗	965.9	966.3	65.1	0.0	∗
	40	1065.2	1066.1	82.8	-0.6	–	1065.2	1066.6	68.6	-2.6	–

Gap (%) denotes the relative deviation of Avg from the upper bound (UB) obtained by LBBD. OPT: ∗ optimal reached, x optimal not reached, – optimality not proven by LBBD.

Table C.6: Detailed HAVS results in large-sized instances with different route duration limits H and planning horizons $|P|$.

H	$\|P\|$	Inst.	Max	Min	Avg	T(s)	Λ(%)	Δ(%)
150	3	1	1243.8	1233.8	1237.3	101.7	0.8	0.3
		2	2167.7	2144.8	2154.5	105.4	1.1	0.5
		3	2220.1	2142.5	2179.6	291.5	3.6	1.7
		4	3129.5	2917.7	3037.2	348.9	7.3	4.1
		5	3935.2	3629.2	3781.1	476.7	8.4	4.2
	4	1	1714.5	1692.3	1706.8	449.1	1.3	0.9
		2	2961.6	2864.1	2907.6	333.8	3.4	1.5
		3	3067.4	2841.4	2944.1	2412.4	8.0	3.6
		4	4327.0	4176.2	4254.6	2869.1	3.6	1.9
		5	5546.5	5317.8	5449.6	3657.8	4.3	2.5
180	3	1	1250.1	1238.9	1242.4	98.9	0.9	0.3
		2	2154.4	2113.5	2134.6	64.2	1.9	1.0
		3	2179.9	2138.4	2155.8	242.2	1.9	0.8
		4	3115.6	3009.5	3066.3	385.3	3.5	1.9
		5	3916.8	3672.1	3767.7	352.9	6.7	2.6
	4	1	1717.4	1686.8	1701.2	560.8	1.8	0.9
		2	2948.6	2854.3	2895.0	222.0	3.3	1.4
		3	2919.2	2741.8	2831.3	2048.5	6.5	3.3
		4	4194.8	3901.8	4011.4	1937.1	7.5	2.8
		5	5264.3	5160.3	5221.3	2310.6	2.0	1.2
Avg.			3065.8	2955.8	3013.2	857.4	4.0	1.9