POLITECNICO DI TORINO

Master's Degree in Mathematical Engineering

Master's Degree Thesis

Methods for Multi-period Vehicle Routing Problems under Uncertainty

March 2026

Supervisor
Prof. Paolo Brandimarte

Candidate
Giorgia Appolloni

Academic Year 2025-2026

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Today, unprecedented amounts of data are generated through the everyday activities of individuals, devices, and digital systems. Information is collected from how people move, interact, and consume services, and additional data is continuously produced through models and synthetic processes.

Yet a simple question remains: who actually benefits from all this information?

In many operational systems, those responsible for making decisions often have only partial access to the data generated by the system itself. As a result, planning and coordination frequently take place under conditions of incomplete or imperfect information.

Logistics offers a clear example of this situation. Dispatchers must coordinate routing decisions while facing uncertain demand, approximate service times, and the possibility that new requests will appear during operations. In such environments, routing is a problem of decision-making under evolving and uncertain information.

This challenge lies at the core of the Vehicle Routing Problem (VRP). Beyond its well-known combinatorial complexity, the difficulty of routing often arises from what Psaraftis [7] described as the *evolution* and the *quality* of information. Evolution refers to the fact that new information may appear over time, for example when customer requests arrive while vehicles are already in transit. Quality refers to the uncertainty associated with the information that is available.

Within this context, the present thesis investigates a variant of the VRP in which customers are organized in geographic clusters, reflecting the spatial structure of many urban distribution systems. In such an environment, the dispatcher faces a recurring operational decision: determining which areas should be served immediately and which should be postponed.

To address this problem, the thesis develops a decision framework based on the principles of Approximate Dynamic Programming. The core contribution is a rule-based dispatching mechanism that separates two decisions: first, *selecting* which zones are ready to be served on a given day, and then *routing* the vehicles accordingly. The selection step is governed by adaptive thresholds that evaluate cluster-level statistics against the current level of urgency, deciding daily whether dispatching to a remote zone is justified.

The remainder of the thesis is organised as follows. Chapter 2 introduces the theoretical background, covering the Vehicle Routing Problem and its stochastic and dynamic extensions, and presenting the Approximate Dynamic Programming framework used to formalise the dispatching decision. Chapter 3 defines the specific routing problem studied in this thesis: the spatial structure of the service region, the stochastic arrival process, the deadline constraints, and the performance objectives. Chapter 4 presents the proposed dispatching policy in detail. Starting from the trigger mechanism that determines which zones to serve, it describes the two-phase queue partitioning that separates high-priority from deferred customers, the vehicle loading procedure, and the calibration of the policy parameters. Chapter 5 evaluates the policy through a systematic simulation study, comparing it against conventional baselines across six scenarios that vary congestion level and deadline pressure. Chapter 6 summarises the main findings and outlines possible directions for future work.

Chapter 2

Theoretical Background

2.1 The Vehicle Routing Problem: Formulation and Variants

2.1.1 From TSP to VRP and Standard Extensions

The Vehicle Routing Problem (VRP) is a combinatorial optimization integer programming problem that can be seen as an extension of the Traveling Salesperson Problem (TSP). While the TSP focuses on finding an optimal route for a single vehicle to visit a set of customers and return to the depot, the VRP involves a fleet of vehicles. A core constraint of the VRP is that each customer must be visited exactly once by a single vehicle. For this reason, once a subset of customers is assigned to a specific vehicle, the sub-problem of determining the optimal sequence of visits reduces to a TSP[9].

In practical applications, the classic VRP is extended to include specific constraints related to the vehicles or the customers. The Capacitated VRP (CVRP) assigns a maximum capacity to each vehicle, which can represent weight, volume, or standardized units. Additionally, time constraints are frequently introduced to limit the maximum route duration, typically to model driver working hours or operational shifts.

2.1.2 Stochastic, Dynamic, and Multi-Period Extensions

Standard VRP formulations assume that all information (customer locations, demands, and constraints) is deterministic and known *a priori*, even though real-world logistics operate in uncertain environments. To capture this complexity, the Stochastic VRP (SVRP) incorporates uncertainty by modeling parameters, such as customer demands, service times, or the arrival of new requests, as random variables with known probability distributions.

The model can be extended to the Dynamic Multi-Period Vehicle Routing Problem (DMPVRP) if the planning horizon spans multiple days and new requests are revealed dynamically over time [10]. The timing of information revelation and decision making are the main keys to manage this dynamism. We have two main path:

- **Online routing:** Decisions are revised continuously. Routes are dynamically optimized while vehicles are in transit, reacting to new events (e.g., the arrival of a highly urgent order).
- **Offline routing:** The decision-making process is locked for a defined time interval, such as a full working day. Events occurring during execution do not alter the current routes, but rather update the system state for the subsequent decision period.

The offline approach is the one related to the optimization problem discussed this thesis. By locking routing decisions on a daily basis, the dispatching problem transforms into a multi-stage sequential decision process. Every day the decision-maker must choose whether to serve a pending request immediately or postpone it to future days, balancing immediate routing costs against expected future penalties. This formulation recalls the mathematical framework of stochastic control and Markov Decision Processes.

2.2 The Sequential Decision-Making Framework

2.2.1 Markov Decision Process (MDP)

Markov Decision Processes are the standard mathematical framework for modeling sequential decision-making under uncertainty. The MDPs are based on the *Markov property*: the conditional probability distribution of future states depends only upon the present state and the current action, being conditionally independent of the past sequence of events.

Following the standard notation for Approximate Dynamic Programming in transportation and logistics [6], an MDP is defined by the core components $(\mathcal{S}, \mathcal{X}, \mathcal{W}, S^M, C)$ evaluated over a discrete time horizon $t \in \mathcal{T}$:

- $S_t \in \mathcal{S}$ is the state variable at decision epoch t, representing all the necessary information to make a decision (e.g., the current queue of pending customers, vehicle locations).
- $x_t \in \mathcal{X}(S_t)$ is the decision (or action) taken at time t, such as the dispatching and routing assignments for the vehicles.
- $W_{t+1} \in \mathcal{W}$ represents the *exogenous information* arriving between epoch t and $t+1$. In a dynamic VRP, this captures the stochastic elements, primarily the newly revealed customer delivery requests.
- $S^M(\cdot)$ is the *transition function* (or system model), which describes how the state evolves. The new state S_{t+1} is determined by the current state, the action taken, and the new exogenous information:

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}) \tag{2.1}$$

- $C(S_t, x_t)$ is the immediate cost (or contribution) accrued by taking action x_t in state S_t, typically representing travel distances and penalties for delayed services.

The objective in an MDP is to find an optimal policy π^*—a mapping from states to actions that minimizes/maximize the expected cumulative cost/reward over the planning horizon.

2.2.2 Stochastic Dynamic Programming and Bellman's Equation

Stochastic Dynamic Programming (SDP) serves as the computational framework for solving MDPs by using the Principle of Optimality.

Given a Markov decision process, SDP derives recursive equations to compute optimal decisions by transforming a complex multi-stage problem into a sequence of simpler, single-stage problems[**?**].

Let $V_t(S_t)$ be the value function, defined as the optimal expected cumulative cost-to-go from time t until the end of the horizon, starting in state S_t. The optimal value function satisfies the backward recursion known as Bellman's equation:

$$V_t(S_t) = \min_{x_t \in \mathcal{X}(S_t)} \{C(S_t, x_t) + \mathbb{E}_{W_{t+1}} [V_{t+1}(S_{t+1}) \mid S_t, x_t]\} \tag{2.2}$$

In this formulation, the expectation $\mathbb{E}[\cdot]$ is taken over the random variable W_{t+1} (the newly arriving exogenous information). This recursion provides the mathematical characterization of the optimal policy.

In the context of the Dynamic Multi-Period VRP, the state S_t represents the queue of customers waiting at the depot at the start of day t; the action x_t defines which customers are loaded into the vehicles and how they are routed; and W_{t+1} is the new orders placed by customers during day t. The transition function (2.1) mathematically updates the queue by removing the served customers and appending the new arrivals. Consequently, any dynamic routing problem with uncertainty can be rigorously formulated as an MDP and, theoretically, solved via SDP.

2.3 Overcoming the Curse of Dimensionality

Solving the Bellman equation directly for stochastic dynamic problems is generally intractable due to the size of the state and action spaces. This section outlines this fundamental computational bottleneck and introduces the approximate resolution frameworks utilized to bypass it.

2.3.1 The Curse of Dimensionality

The term *curse of dimensionality*, describes the phenomenon where the state space of a decision problem grows exponentially with the number of variables, rendering exact Dynamic Programming (DP) computationally infeasible.

In the context of the Stochastic Dynamic VRP, the state variable S_t must encode all relevant information: the pending customer queue (exact locations, residual demands, specific deadlines), the fleet status (remaining volume capacities, current geographical locations, residual shift times), and the temporal progression of the system. Concurrently, the action space x_t involves combinatorial routing sequences and dispatching assignments. At the same time, the number of possible state-action pairs becomes astronomically large. In practice, computing the exact Bellman equation for a VRP of realistic dimensions requires prohibitive computational times, as it necessitates evaluating the expected future value over the entire probability space of exogenous arrivals for every possible configuration [4].

2.3.2 Approximate Dynamic Programming (ADP)

To overcome these structural limitations, researchers rely on Approximate Dynamic Programming (ADP), a set of algorithmic strategies designed to solve large-scale sequential decision problems. The core principle of ADP is to replace the exact global value function with tractable approximations, or to define simplified heuristic policies instead of searching for strict mathematical optimality.

ADP approaches are generally categorized into four fundamental classes[5]:

1. **Value Function Approximation (VFA):** This approach projects the exact value function onto a lower-dimensional space, often modeling it as a linear combination of basis functions (features) or through statistical regression over simulated states. Rather than computing the exact value of every state, VFA estimates a generalized function that approximates the future costs.

2. **Cost Function Approximation (CFA):** Instead of calculating a future value, CFA modifies the immediate cost (or reward) function through additional parameterized penalties or bonuses. The decision-making process remains myopic, optimizing the current step, but is modulated by parameters tuned to capture future operational impacts.

3. **Policy Function Approximation (PFA):** This class directly searches for an approximate decision rule, denoted as $\hat{\pi}(S_t)$, which maps any given state directly to a specific action. PFA entirely avoids the computation of a value function, focusing instead on finding a robust parametric or heuristic policy that can bring an high-quality global performance.

4. **Lookahead Policies (Rollout and Direct Simulation):** This approach evaluates available actions by performing a limited forward simulation from the current state, using a simplified base policy. At each decision epoch, it compares alternatives by summing the costs accrued over a defined horizon, often implemented via rollout algorithms or Monte Carlo tree search strategies.

These approaches are frequently hybridized in practice. For instance, a heuristic base policy (PFA) can be enhanced through a rollout mechanism (Lookahead), or its internal parameters can be dynamically updated via simulation. The common objective across all ADP methods is to provide high-quality solutions for realistically sized VRPs within acceptable computational times.

Chapter 3

Problem Formulation

This chapter formalises the operational problem addressed in this thesis: a *dynamic, capacitated vehicle routing problem with stochastic arrivals and deadlines.* First, the problem will be described within the VRP taxonomy (3.1), then described the service region and its cluster structure (3.2), the stochastic arrival process (3.3), the customer attributes and constraints (3.4), the fleet model (3.5), the dispatch cycle (3.6), and finally state the mathematical objective (3.8).

3.1 Problem Taxonomy

The classical Capacitated Vehicle Routing Problem (CVRP) asks for minimum-cost routes that start and end at a central depot, visit a *known* set of customers, and respect vehicle capacity constraints. In many real-world applications, however, the set of customers is *not* known in advance: service requests arrive over time according to a stochastic process, each carrying an individual deadline. The dispatcher must decide *when* and *which* customers to serve, and *how* to route the vehicles, without full knowledge of future demand.

Our problem extends the static CVRP along three dimensions:

1. **Dynamic arrivals.** Customers are not available at the beginning of the planning horizon, they arrive day by day according to a Poisson process. The dispatcher observes only the arrivals that have occurred so far.

2. **Deadline constraints .** Each customer is assigned an individual deadline. Service after the deadline could be penalized but not forbidden, making the deadline *soft.*

3. **Dispatch timing.** The dispatcher solves the routing problem at each period (day), choosing which subset of the waiting queue to serve. Customers not selected remain in the queue for future periods.

These features place the problem in the family of *Dynamic and Stochastic Vehicle Routing Problems*, as surveyed by Pillac et al. [3] and Psaraftis et al. [8].

This thesis builds upon the Multiperiod Dynamic Capacitated Traveling Salesperson Problem introduced by Mutailifu et al. [2]. Their work addressed real-world logistics

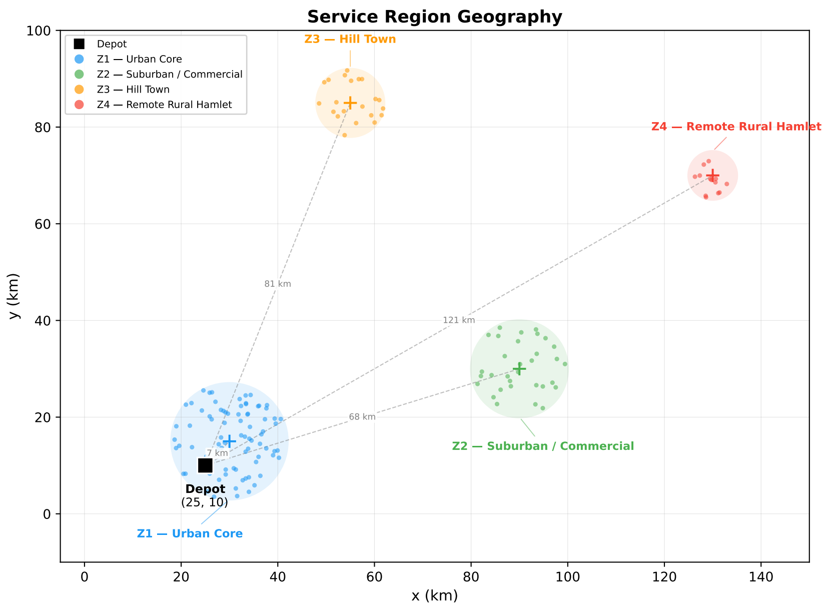


Figure 3.1. Service region geography - with depot, four demand zones, and sample customer locations. Zone 1 is the dense urban core; Zones 2–4 are satellite clusters at increasing distance from the depot.

challenges such as stochastic demand, irregularly clustered customers, and strict duration limits using an approximate dynamic programming (ADP) trigger-based dispatch policy. While their study was explicitly limited to a single-vehicle scenario, following the future research directions outlined by the authors, this thesis extends the framework to a multi-vehicle capacitated Vehicle Routing Problem (VRP).

3.2 Service Region and Cluster Geography

The service region is a bounded area in $\mathbb{R}^2$ containing a central *depot* and a set of Z demand zones (clusters). The geography is designed to model a realistic "city plus hinterland" layout, with one dense urban core close to the depot and several satellite clusters at increasing distances.

Notation. The depot is located at coordinates $\mathbf{d} = (d_x, d_y)$. Each zone $z \in \mathcal{Z} = \{1, 2, \ldots, Z\}$ is characterised by a triple

$$(\boldsymbol{\mu}_z,\ r_z,\ \lambda_z), \tag{3.1}$$

where $\boldsymbol{\mu}_z = (\mu_{z,x}, \mu_{z,y})$ is the centre of the zone, $r_z > 0$ is its radius, and $\lambda_z > 0$ is the Poisson arrival rate (customers per day).

Instance geography. The computational experiments in this thesis use $Z = 4$ zones whose parameters are reported in Table 3.1. Zone 1 is the *core cluster*: it is closest to the depot, has the highest arrival rate, and generates the bulk of daily demand. Zones 2, 3, and 4 are *satellite clusters* at progressively greater distance from the depot.

Table 3.1. Demand zone parameters. Distances are Euclidean from the depot at $\mathbf{d} = (25, 10)$. Arrival rates are base values per vehicle; the actual rate is $\lambda_z = \lambda_z^{\text{base}} \times |\mathcal{K}| \times m_\lambda$, where $|\mathcal{K}|$ is the fleet size and m_λ is the scenario-dependent congestion multiplier (see Section 3.7).

Zone z	Description	Centre $\boldsymbol{\mu}_z$	Radius r_z	λ_z^{base}	$d(\mathbf{d}, \boldsymbol{\mu}_z)$	Role
1	Dense urban core	$(30, 15)$	12	3.0	$\approx 7\,\text{km}$	Core
2	Suburban / commercial	$(90, 30)$	10	1.2	$\approx 68\,\text{km}$	Satellite
3	Hill town	$(55, 85)$	7	0.8	$\approx 81\,\text{km}$	Satellite
4	Remote rural hamlet	$(130, 70)$	5	0.5	$\approx 121\,\text{km}$	Satellite
Total $\lambda_{\text{tot}}^{\text{base}}$ per vehicle				5.5		

Customer location. A customer arriving in zone z is placed uniformly at random inside the disk of radius r_z centred at $\boldsymbol{\mu}_z$. The polar sampling procedure guarantees uniform coverage of the circular area:

$$\mathbf{x}_i = \boldsymbol{\mu}_z + r_z\sqrt{U_1}\begin{pmatrix}\cos(2\pi\, U_2)\\ \sin(2\pi\, U_2)\end{pmatrix}, \qquad U_1, U_2 \overset{\text{iid}}{\sim} \text{Uniform}(0,1). \tag{3.2}$$

Distance and travel time. Distances between any pair of locations (customers and depot) are measured in Euclidean metric. Let $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^2$ denote two locations; then

$$d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2, \qquad t_{ij} = \frac{d_{ij}}{v}, \tag{3.3}$$

where v is the constant vehicle speed (km/h) and t_{ij} is the travel time in hours. The resulting distance and travel-time matrices are symmetric and satisfy the triangle inequality.

3.3 Stochastic Arrival Process

Customer requests arrive over a discrete time horizon $\mathcal{T} = \{1, 2, \ldots, T\}$, where each period represents one working day. Arrivals in each zone follow independent Poisson processes.

3.3.1 Daily arrivals

Let $N_z(t)$ denote the number of customers arriving in zone z on day t. Then

$$N_z(t) \sim \text{Poisson}(\lambda_z), \qquad z \in \mathcal{Z}, \quad t \in \mathcal{T}, \tag{3.4}$$

and the processes $\{N_z(t)\}_{z\in\mathcal{Z}}$ are mutually independent.

The total expected daily demand (in number of customers) is

$$\bar{N} = \sum_{z\in\mathcal{Z}} \lambda_z = \lambda_{\text{tot}}^{\text{base}} \times |\mathcal{K}| \times m_\lambda, \tag{3.5}$$

where $|\mathcal{K}|$ is the number of vehicles and $m_\lambda \geq 0$ is a scenario-dependent *congestion multiplier* that controls the system load.

Derivation of m_λ. The base arrival rates per vehicle are $\lambda_1^{\text{base}} = 3.0$, $\lambda_2^{\text{base}} = 1.2$, $\lambda_3^{\text{base}} = 0.8$, $\lambda_4^{\text{base}} = 0.5$, giving a per-vehicle total of $\lambda_{\text{tot}}^{\text{base}}/|\mathcal{K}| = 5.5$ customers/day. Each zone rate is then scaled as

$$\lambda_z = \lambda_z^{\text{base}} \times |\mathcal{K}| \times m_\lambda, \tag{3.6}$$

so that m_λ acts as a uniform multiplier on all zones simultaneously. By substituting into the utilisation ratio we obtain a closed-form relationship between m_λ and the target congestion level ρ:

$$\rho = \frac{\bar{N}\cdot\mathbb{E}[q_i]}{|\mathcal{K}|\cdot Q} = \frac{5.5\times m_\lambda\times\mathbb{E}[q_i]}{Q}. \tag{3.7}$$

With $\mathbb{E}[q_i] = 30$ units and $Q = 250$ units per vehicle,

$$m_\lambda = \frac{\rho\cdot Q}{5.5\times\mathbb{E}[q_i]} = \frac{\rho\cdot 250}{5.5\times 30} = \frac{\rho}{0.66}.$$

Table 3.3 in Section 3.7 reports the three multiplier values used in the experiments ($m_\lambda \approx 1.06, 1.36, 1.82$ for $\rho = 0.70, 0.90, 1.20$).

When $\rho < 1$ the fleet has, on average, sufficient capacity to serve all daily arrivals; when $\rho > 1$ the system is structurally overloaded and some customers will necessarily be delayed or left unserved.

3.4 Customer Attributes

Each customer i arriving on day t_i^{in} in zone z_i is fully characterised by the tuple

$$c_i = \big(\mathbf{x}_i,\ z_i,\ q_i,\ s_i,\ t_i^{\text{in}},\ \bar{t}_i\big), \tag{3.8}$$

where the attributes are drawn independently:

- $\mathbf{x}_i \in \mathbb{R}^2$ — location, sampled uniformly in zone z_i (Equation (3.2));
- $z_i \in \mathcal{Z}$ — zone membership;
- $q_i \sim \text{Uniform}\{q_{\min},\ldots,q_{\max}\}$ — demand in units (default: $q_{\min} = 10$, $q_{\max} = 50$, hence $\mathbb{E}[q_i] = 30$);
- $s_i \sim \text{Uniform}(s_{\min}, s_{\max})$ — service time in hours (default: $s_{\min} = 0.25\,\text{h}$, $s_{\max} = 2\,\text{h}$);

- $t_i^{\text{in}} = t_i$ — arrival day;
- $\bar{t}_i = t_i^{\text{in}} + \Delta_i$ — absolute deadline, where $\Delta_i \sim \text{Uniform}\{d_{\min}, \dots, d_{\max}\}$ is the deadline window in days.

The deadline window parameters $d_{\min}$ and $d_{\max}$ are scenario-dependent and represent the urgency of service requests (see Section 3.7).

3.5 Fleet Model

The fleet consists of $|\mathcal{K}|$ homogeneous vehicles, each characterised by:

- Q — volume capacity per vehicle (units);
- H — daily working-time limit per vehicle (hours);
- v — constant travel speed (km/h).

All vehicles are based at the depot $\mathbf{d}$ and must return to the depot at the end of each working day. The aggregate daily capacity of the fleet is $|\mathcal{K}| \cdot Q$ in volume and $|\mathcal{K}| \cdot H$ in time.

Table 3.2. Fleet parameters used in the computational experiments.

Parameter	Symbol	Value
Number of vehicles	$\lvert\mathcal{K}\rvert$	3
Capacity per vehicle	Q	250 units
Working-time limit	H	10 h
Speed	v	50 km/h
Depot location	$\mathbf{d}$	$(25, 10)$

3.6 Daily Dispatch Cycle

The simulation operates on a discrete daily cycle. At each period $t \in \mathcal{T}$, the dispatcher performs the following steps:

1. **Arrival.** New customers $\{c_i : t_i^{\text{in}} = t\}$ arrive and join the waiting queue $\mathcal{Q}(t)$.
2. **Selection.** A *dispatch policy* π inspects the queue and selects a subset $\mathcal{S}(t) \subseteq \mathcal{Q}(t)$ to serve today. Not all customers in the queue need to be dispatched: the policy may decide to postpone satellite customers when fewer loads are accumulated (see Chapter 4).

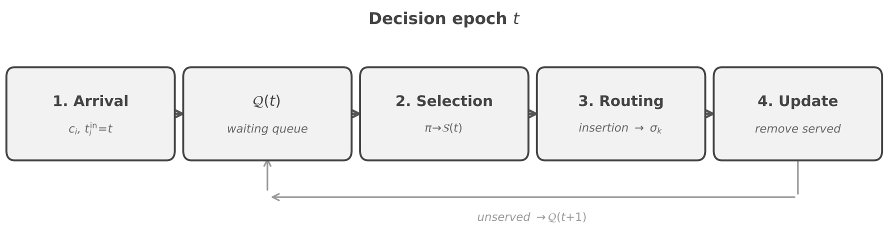


Figure 3.2. Daily dispatch cycle: arrivals join the queue; the policy selects a subset; vehicles are loaded and routed; served customers are removed.

3. **Routing.** The selected customers are assigned to vehicles and routed. Each vehicle k receives a sequence $\sigma_k(t)$ such that

$$\sum_{i\in\sigma_k(t)} q_i \le Q, \qquad \underbrace{\sum_{(i,j)\,\in\,\text{edges}(\sigma_k)} t_{ij}}_{\text{travel}} + \underbrace{\sum_{i\in\sigma_k(t)} s_i}_{\text{service}} \le H. \tag{3.9}$$

4. **Update.** Served customers are removed from the queue; unserved customers carry over to the next day:

$$\mathcal{Q}(t+1) = \big(\mathcal{Q}(t) \setminus \mathcal{S}(t)\big) \cup \{c_i : t_i^{\text{in}} = t+1\}.$$

Routing heuristic. Given the selected customers $\mathcal{S}(t)$, vehicle routes are constructed using a *cheapest insertion* heuristic. Each vehicle route σ_k is initialised as a degenerate depot–depot tour $(\mathbf{d}, \mathbf{d})$. Customers from $\mathcal{S}(t)$ are considered one by one in the order determined by the dispatch policy. For each candidate customer c_j, the heuristic evaluates every vehicle k and every feasible insertion position p within σ_k. A position is feasible if and only if *both* constraints are satisfied simultaneously:

1. **Capacity:** $\sum_{i\in\sigma_k} q_i + q_j \le Q$;
2. **Route duration:** $T(\sigma_k) + \Delta t_{\text{insert}}(p, j) + s_j \le H$,

where Δt_{insert} is the additional travel time caused by inserting c_j at position p (detour cost). Among all feasible (vehicle, position) pairs, the one with the minimum extra distance is selected; the customer is inserted into that route and the route's cumulative volume and time are updated. If no feasible insertion exists across *any* vehicle, the customer stays in the queue until the next day.

The heuristic iterates through the entire candidate list without early stopping: every customer is given the opportunity to be inserted. This means that vehicles are progressively filled until their capacity or time budget is exhausted, at which point subsequent candidates are deferred.

This heuristic, while not globally optimal, is computationally inexpensive and produces high-quality solutions. Its speed is essential for the PSO calibration loop, which evaluates thousands of dispatch policies over long simulation horizons (Section 3.9).

3.7 Experimental Scenarios

The behaviour of any dispatch policy depends on the interaction between demand intensity and deadline urgency. To systematically evaluate performance, we define six scenarios by varying two experimental factors *one at a time*, keeping the geography fixed.

3.7.1 Factor 1: Congestion Level (ρ)

The congestion multiplier m_λ scales all zone arrival rates simultaneously, while deadlines are fixed at $\Delta_i \sim \text{Uniform}\{1, 2, 3\}$. Three levels are defined:

Table 3.3. Congestion scenarios. The utilisation ratio $\rho = \bar{N} \cdot \mathbb{E}[q_i] \,/\, (|\mathcal{K}| \cdot Q)$.

Scenario	m_λ	$\bar{N}$/vehicle	ρ	Regime
LOW	1.061	5.83	0.70	Under-capacity
MEDIUM	1.364	7.50	0.90	Near-capacity
HIGH	1.818	10.0	1.20	Over-capacity

3.7.2 Factor 2: Deadline Pressure

Arrival rates are held at the MEDIUM level ($\rho = 0.90$), and the deadline window Δ_i is varied:

Table 3.4. Deadline scenarios. All share $\rho = 0.90$.

Scenario	Deadline Δ_i	Description
RELAXED	Uniform{3,4,5}	Generous margin
STANDARD	Uniform{1,2,3}	Default (= MEDIUM)
TIGHT	Uniform{0,1,2}	Urgent, little margin
CRITICAL	Uniform$\{0, 1\}$	Near-immediate delivery

The MEDIUM and STANDARD scenarios are identical, yielding six unique scenarios in total. This factorial structure allows us to isolate the effect of congestion from the effect of deadline pressure.

3.8 Objective Function

A dispatch policy must balance two competing goals:

- **Operational efficiency.** Minimise total travel time (or equivalently distance), which directly impacts fuel cost and driver wellness.
- **Service quality.** Minimise the fraction of customers who receive late service or remain unserved (*violation rate*), which drives customer satisfaction and contractual penalties.

These objectives are generally conflicting. Dispatching vehicles frequently (even when partially loaded) improves timeliness but increases travel costs due to partially utilised routes. Conversely, waiting to consolidate loads reduces travel but risks deadline violations.

3.8.1 Normalised Performance Indicators

We define four normalised metrics computed over a simulation of T days:

1. **Normalised travel ($\tilde{T}$).**

$$\tilde{T} = \frac{\bar{t}_{\text{travel}}}{|\mathcal{K}| \cdot H}, \qquad \bar{t}_{\text{travel}} = \frac{1}{T} \sum_{t=1}^{T} \sum_{k \in \mathcal{K}} t^k_{\text{travel}}(t), \tag{3.10}$$

   where $t^k_{\text{travel}}(t)$ is the travel time of vehicle k on day t. The denominator $|\mathcal{K}| \cdot H$ is the total fleet time, so $\tilde{T} \in [0, 1]$.

2. **Violation rate ($\tilde{L}$).** Accounts for every form of service failure—both customers served after their deadline (*late*) and customers never dispatched (*unserved*):

$$\tilde{L} = \frac{n_{\text{late}} + n_{\text{unserved}}}{n_{\text{arrivals}}}, \tag{3.11}$$

   where n_{arrivals} is the total number of customer arrivals over the simulation horizon. For diagnostic purposes the two sub-components—late rate $\tilde{L}_{\text{late}} = n_{\text{late}}/n_{\text{arrivals}}$ and unserved rate $\tilde{U} = n_{\text{unserved}}/n_{\text{arrivals}}$—are reported separately in the result tables.

3. **Throughput (Φ).**

$$\Phi = \frac{\text{total customers served}}{T} \quad \text{[customers/day]}. \tag{3.12}$$

3.8.2 Composite Objective (α-Weighted)

For calibration purposes, the two primary objectives are combined into a single scalar using a convex weight $\alpha \in [0, 1]$:

$$\boxed{J(\pi;\, \alpha) = \alpha \cdot \tilde{T} + (1 - \alpha) \cdot \tilde{L}.} \tag{3.13}$$

By varying α, the decision-maker expresses a preference along the efficiency–service trade-off:

- $\alpha = 0.3$ (**service** profile): heavier penalty on late deliveries; the policy tries to dispatch earlier to avoid deadline violations.
- $\alpha = 0.5$ (**balanced** profile): equal weight on travel efficiency and service quality.
- $\alpha = 0.7$ (**efficiency** profile): heavier penalty on travel; the policy consolidates loads more aggressively, tolerating some late deliveries to reduce mileage.

Rather than producing a single policy, this multi-profile approach yields a family of policies that can be chosen depending on business priorities.

3.8.3 Baseline Policies

Two priority-based dispatching rules serve as baselines. Unlike the single-key sorts, both baselines use a *lexicographic* composite key that adds geographic awareness as a secondary criterion, followed by demand as a tie-breaker:

FIFO (First-In, First-Out). Customers are sorted by the three-level key

$$\text{key}^{\text{FIFO}}(i) = \Big(\underbrace{t_i^{\text{in}}}_{\text{arrival (asc)}}, \underbrace{z_i}_{\text{zone (asc)}}, \underbrace{-q_i}_{\text{demand (desc)}} \Big).$$

The primary key respects arrival order; the secondary key groups customers from the same zone consecutively, improving the spatial coherence of the routes produced by the insertion heuristic; the tertiary key favours larger demands to maximise vehicle utilisation.

EDD (Earliest Due Date). Customers are sorted by the analogous key

$$\text{key}^{\text{EDD}}(i) = \Big(\underbrace{\bar{t}_i}_{\text{deadline (asc)}}, \underbrace{z_i}_{\text{zone (asc)}}, \underbrace{-q_i}_{\text{demand (desc)}} \Big).$$

The primary key prioritises the most urgent customers, while the secondary and tertiary keys serve the same geographic and load-packing purposes as in FIFO.

After sorting, the ordered customer list is passed to the *cheapest insertion* heuristic (Definition 4.2), which processes candidates one by one in the given order. For each customer, the heuristic evaluates all vehicles and selects the (vehicle, position) pair that minimises extra travel distance, subject to the capacity and time constraints of Equations (4.12)–(4.13). If no feasible insertion exists across any vehicle, the customer is left in the queue for the next day. The heuristic iterates through the *entire* sorted list—there is no early stopping—so that every customer is given the chance to be inserted.

Both baselines are thus "dispatch-everything" policies: at each period they attempt to serve the full queue, with no consolidation logic.

3.9 Preview: Calibration Approach

The trigger policy introduced in Chapter 4 exists in three modes—volume-only, time-only, and dual—each controlled by one or two continuous sensitivity parameters that determine when satellite clusters are dispatched. Finding optimal values requires many simulations of the stochastic system, which precludes exact methods. For the one-dimensional modes we use a grid search; for the two-dimensional dual mode we use *Particle Swarm Optimisation* (PSO). In both cases each candidate is evaluated by running a full multi-day simulation with the above arrival process and routing heuristic. The calibration procedure is described in detail in Chapter 4, Section 4.6.

Table 3.5 collects the principal notation used throughout the thesis.

Table 3.5. Summary of notation.

Symbol	Description
$\mathcal{Z} = \{1, \dots, Z\}$	Set of demand zones
$\mathcal{K} = \{1, \dots, \lvert\mathcal{K}\rvert\}$	Set of vehicles
$\mathcal{T} = \{1, \dots, T\}$	Planning horizon (days)
$\mathcal{Q}(t)$	Waiting queue at the beginning of day t
$\mathcal{S}(t)$	Set of customers dispatched on day t
$\boldsymbol{\mu}_z, r_z$	Centre and radius of zone z
λ_z	Poisson arrival rate in zone z (customers/day)
q_i, s_i	Demand (units) and service time (hours) of customer i
$t_i^{\text{in}}, \bar{t}_i$	Arrival day and deadline of customer i
Q, H, v	Vehicle capacity, time limit, speed
$\mathbf{d}$	Depot location
d_{ij}, t_{ij}	Distance and travel time between i and j
ρ	Congestion (utilisation) ratio
m_λ	Congestion multiplier
$\tilde{T}, \tilde{L}$	Normalised travel and late rate
α	Trade-off weight (efficiency vs. service)
$J(\pi; \alpha)$	Composite objective for policy π

Chapter 4

The Hybrid Trigger Policy

This chapter presents the core contribution of the thesis: an integrated, trigger-based dispatch policy for the dynamic capacitated VRP with stochastic arrivals introduced in Chapter 3. The policy adapts the trigger mechanism proposed by Mutailifu et al. [2] for the collection TSP and extends it to a multi-vehicle, multi-zone routing framework with capacity and time-window constraints.

The policy decomposes the daily dispatch decision into three sequential tasks (Figure 4.1):

1. **Trigger Evaluation** (4.2): for each satellite cluster, a threshold function determines whether accumulated demand is "ready" for dispatch. Three *trigger modes* are studied: volume-only, time-only, and dual (volume + time).
2. **Two-Phase Queue Partitioning** (4.3): the waiting queue is split into a *Phase 1* set (triggered satellite customers) and a *Phase 2* set (core-zone customers, overdue stragglers, and Phase 1 overflow); deferred satellite customers are excluded entirely.
3. **Sequential Vehicle Loading** (4.4): Phase 1 and Phase 2 candidates are inserted into vehicle routes in sequence using a cheapest-insertion heuristic.

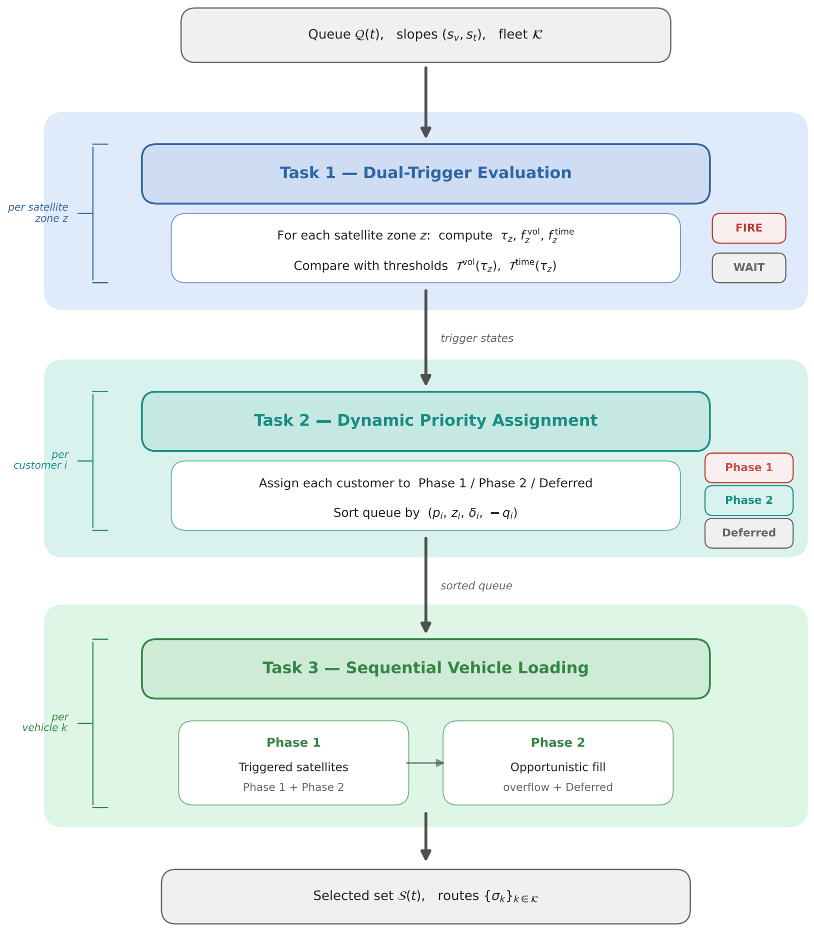
The numerical parameters of the trigger mechanism (one or two slopes, depending on the mode) are calibrated offline via grid search or Particle Swarm Optimisation under three trade-off profiles (4.6).

4.1 Motivation: When to Dispatch Satellite Clusters

In static VRP, the set of customers is known in advance and the optimiser decides only the routing. In the dynamic setting of this thesis, the dispatcher faces an additional decision: *which customers to serve today and which to defer to tomorrow.*

Core-zone customers are close to the depot and generate enough daily demand to fill a vehicle efficiently. Core customers can be dispatched daily with low marginal travel cost.

Satellite zones, however, pose a dilemma. They are far from the depot (68–118 km), so sending a vehicle to serve only one or two customers can waste most of the working



Queue $\mathcal{Q}(t)$, slopes (s_v, s_t), fleet $\mathcal{K}$
per satellite zone z
Task 1 — Dual-Trigger Evaluation
For each satellite zone z: compute τ_z, f_z^{vol}, f_z^{time}
Compare with thresholds $\mathcal{T}^{\text{vol}}(\tau_z)$, $\mathcal{T}^{\text{time}}(\tau_z)$
FIRE
WAIT
trigger states
per customer i
Task 2 — Dynamic Priority Assignment
Assign each customer to Phase 1 / Phase 2 / Deferred
Sort queue by $(p_i, z_i, \delta_i, -q_i)$
Phase 1
Phase 2
Deferred
sorted queue
per vehicle k
Task 3 — Sequential Vehicle Loading
Phase 1
Triggered satellites
Phase 1 + Phase 2
Phase 2
Opportunistic fill
overflow + Deferred
Selected set $\mathcal{S}(t)$, routes $\{\sigma_k\}_{k \in \mathcal{K}}$

Figure 4.1. Three-task pipeline of the Hybrid Trigger Policy. Task 1 evaluates trigger conditions per satellite zone; Task 2 partitions the queue into Phase 1, Phase 2, and Deferred; Task 3 loads vehicles via global cheapest insertion + residual filling.

day. Waiting to accumulate more demand in a satellite zone improves route efficiency, but risks violating deadlines. The fundamental trade-off is:

Dispatch early $\Rightarrow$ high travel per customer, low late rate.
Dispatch late $\Rightarrow$ low travel per customer, high late rate.

The trigger mechanism formalises this trade-off by defining *threshold functions* that relate the accumulated demand in a satellite zone to the remaining time before the earliest deadline. When accumulated demand crosses the threshold, the trigger "fires" and the zone's customers are promoted to the highest dispatch priority.

This idea originates from the line of research by Mutailifu et al. [2] on Multi-Depot Collection TSP with Stochastic Arrivals, where trigger policies were shown to outperform both myopic dispatching and policy-function-approximation approaches on single-depot pickup instances. Our contribution is to (i) define three trigger modes—volume-only, time-only, and a dual form that monitors both signals simultaneously—and empirically compare them, (ii) embed the trigger in a *multi-vehicle VRP* dispatcher with two-phase loading.

4.2 Task 1: Dual-Trigger Evaluation

At the beginning of each day t, the dispatcher evaluates, for each satellite zone $z \in \mathcal{Z}_{\text{sat}} = \mathcal{Z} \setminus \{1\}$, whether accumulated demand justifies scheduling a vehicle. Two aggregate statistics and one urgency measure are computed.

4.2.1 Cluster-Level Statistics

Let $\mathcal{Q}_z(t) = \{i \in \mathcal{Q}(t) : z_i = z\}$ denote the customers in the queue belonging to zone z. We compute:

1. **Urgency** (minimum days to deadline):
$$\tau_z(t) = \min_{i \in \mathcal{Q}_z(t)} (\bar{t}_i - t). \tag{4.1}$$
If $\tau_z(t) \leq 0$, at least one customer in the zone is overdue and the trigger fires unconditionally.
2. **Volume fraction**:
$$f_z^{\text{vol}}(t) = \frac{\sum_{i \in \mathcal{Q}_z(t)} q_i}{|\mathcal{K}| \cdot Q}. \tag{4.2}$$
3. **Time fraction** (approximate):
$$f_z^{\text{time}}(t) = \frac{\hat{T}_z^{\text{route}}(t)}{|\mathcal{K}| \cdot H}, \tag{4.3}$$
where $\hat{T}_z^{\text{route}}(t)$ is a nearest-neighbour estimate of the time needed to serve all customers in $\mathcal{Q}_z(t)$ (travel via an NN tour from the depot plus all service times). This provides a fast proxy for route duration without solving a full VRP.

4.2.2 Threshold Functions

The trigger is governed by two parameters, $s_v, s_t \in [0,1]$, referred to as the *volume slope* and *time slope* respectively. They define a pair of linear threshold functions of the urgency:

The threshold computation applies only when $\tau_z(t) > 0$ (i.e. no customer is overdue). If $\tau_z(t) \leq 0$, the trigger fires unconditionally and the threshold evaluation is skipped.

Normalisation. We set $\tau_{\max} = d_{\max}$, the upper bound of the scenario's deadline distribution (e.g. $\tau_{\max} = 3$ for U[1,3], $\tau_{\max} = 5$ for U[3,5], $\tau_{\max} = 2$ for U[0,2], and $\tau_{\max} = 1$ for U[0,1]). Because deadlines are drawn from $U[a,b]$, a customer arriving at day t has $d_i - t \leq b = \tau_{\max}$; customers that have already waited one or more days satisfy $d_i - t < d_i - t_i^{\text{in}} \leq b$. Therefore $\tau_z(t) \leq \tau_{\max}$ by construction and the *urgency ratio*

$$r_z(t) = \frac{\tau_z(t)}{\tau_{\max}} \in (0,\,1] \tag{4.4}$$

is well-defined whenever $\tau_z(t) > 0$.[1]

Threshold functions. The threshold lines are linear functions of $r_z(t)$, scaled by the calibrated slopes $s_v, s_t \in [0,1]$:

$$\mathcal{T}^{\text{vol}}\big(\tau_z(t)\big) = s_v \cdot r_z(t), \qquad \mathcal{T}^{\text{time}}\big(\tau_z(t)\big) = s_t \cdot r_z(t). \tag{4.5}$$

Geometric interpretation. When the zone's most urgent deadline is far away ($r_z \to 1$), the thresholds reach their maximum values s_v and s_t, requiring the accumulated volume or time fraction to be large before firing—the trigger is *selective.* As the deadline approaches ($r_z \to 0$), both thresholds drop toward zero, making the trigger progressively *permissive.* When $\tau_z \leq 0$ (at least one customer is overdue), the trigger fires unconditionally regardless of accumulated demand.

4.2.3 Trigger Rule

The trigger for zone z on day t fires if *any* of the following conditions holds:

$$\boxed{\text{FIRE}_z(t) \iff \begin{cases} \tau_z(t) \leq 0 & \text{(overdue)} \\ f_z^{\text{vol}}(t) \geq \mathcal{T}^{\text{vol}}\big(\tau_z(t)\big) & \text{(volume channel)} \\ f_z^{\text{time}}(t) \geq \mathcal{T}^{\text{time}}\big(\tau_z(t)\big) & \text{(time channel)} \end{cases}} \tag{4.6}$$

The disjunctive structure (OR) means that a satellite zone is dispatched whenever *either* its accumulated volume or its estimated route time crosses the respective threshold.

[1]The implementation includes a defensive guard $\min(\tau_z, \tau_{\max})$ in the numerator, which is redundant under the uniform-deadline assumption but would become necessary if the arrival distribution were changed to one with unbounded support.

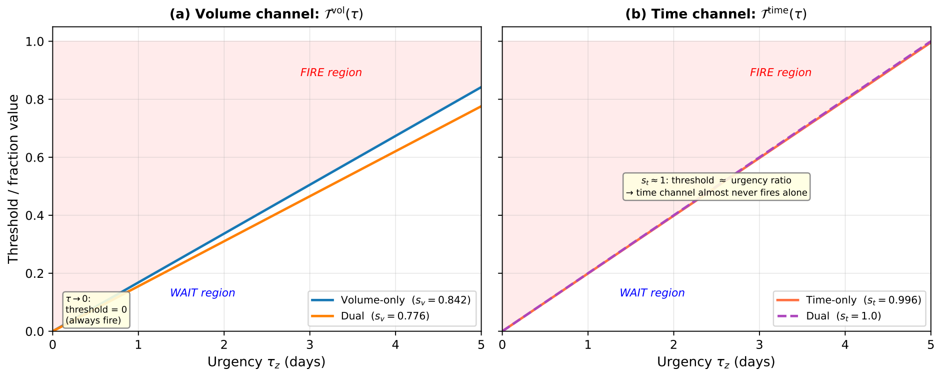


Figure 4.2. Threshold lines $\mathcal{T}^{\text{vol}}(\tau)$ and $\mathcal{T}^{\text{time}}(\tau)$ for the balanced profile ($\alpha = 0.5$). A satellite zone fires when its volume or time fraction exceeds the corresponding threshold. As $\tau \to 0$ (deadline imminent), both thresholds drop to zero, making the trigger increasingly permissive.

This dual activation provides robustness: a zone with few heavy-demand customers may not trigger on volume alone, but their concentrated enough travelling time may trigger the time channel, and vice versa.

Interpretation of the slopes. The slopes s_v and s_t control how "conservative" the trigger is:

- $s = 0$ **(most permissive):** The threshold is identically zero; the trigger fires for any non-empty queue, regardless of urgency. The policy dispatches satellite zones as soon as any customer appears — equivalent to a greedy "dispatch-everything" approach for that zone.
- $s = 1$ **(most selective):** The threshold equals the normalised urgency; the trigger fires only when accumulated demand (as a fraction of fleet capacity) meets or exceeds the remaining time fraction. The policy postpones dispatch to consolidate loads, at the risk of violating tight deadlines.
- $0 < s < 1$**:** Intermediate behaviour. As the deadline approaches ($\tau \to 0$), the threshold drops to zero and the zone is dispatched regardless of load. With more remaining time ($\tau \to \tau_{\max}$), a higher fraction of fleet capacity must be accumulated before triggering.

4.2.4 Volume-Only Trigger Mode

In the *volume-only* variant of the policy, the time channel is disabled:

$$\text{FIRE}_z^{\text{vol-only}}(t) \iff \tau_z(t) \le 0 \ \text{ OR } \ f_z^{\text{vol}}(t) \ge \mathcal{T}^{\text{vol}}(\tau_z(t)). \tag{4.7}$$

This reduces the parameter space from two dimensions to one (s_v only), simplifying calibration at the potential expense of some flexibility. The experimental comparison in

Chapter 5 evaluates whether the additional time channel provides a meaningful performance improvement.

4.2.5 Time-Only Trigger Mode

Symmetrically, the *time-only* variant disables the volume channel:

$$\mathrm{FIRE}_z^{\text{time-only}}(t) \iff \tau_z(t) \le 0 \;\text{ OR }\; f_z^{\text{time}}(t) \ge \mathcal{T}^{\text{time}}(\tau_z(t)). \tag{4.8}$$

Here the only tunable parameter is s_t, which controls how aggressively the policy reacts to the nearest-neighbour travel-time estimate $\hat{T}_z^{\text{NN}}(t)$. As shown in the calibration results (Table 4.2), the optimal s_t tends to values very close to 1.0 across all three operator profiles, meaning the time channel alone is nearly equivalent to a "do nothing special" policy—the threshold is so high that only zones whose travel time already saturates the fleet budget trigger a dispatch. This anticipates one of the main experimental findings: the volume signal carries most of the useful dispatch information, while the time channel adds little on its own.

4.3 Task 2: Two-Phase Queue Partitioning

Once the trigger states $\{\mathrm{FIRE}_z(t)\}_{z\in\mathcal{Z}_{\text{sat}}}$ have been determined, the waiting queue is partitioned into three disjoint subsets that govern the loading order.

Definition 4.1 (Queue partition)**.** ***(a) Phase 1 (triggered satellites).*** *Customers whose satellite zone has fired:* $\mathcal{P}_1(t) = \{i : z_i \in \mathcal{Z}_{sat} \text{ and } FIRE_{z_i}(t)\}$.

(b) Phase 2 (core + overdue + overflow). All *core-zone customers, plus any satellite customer from a non-triggered zone whose deadline has expired* ($\bar{t}_i \le t$)*, plus Phase 1 customers that could not be inserted for capacity reasons:* $\mathcal{P}_2(t) = \{i : z_i = 1\} \,\cup\, \{i : z_i \in \mathcal{Z}_{sat},\ \neg FIRE_{z_i}(t),\ \bar{t}_i \le t\} \,\cup\, overflow(\mathcal{P}_1)$.

(c) Deferred (wait). *Satellite customers from non-triggered zones whose deadline is still in the future:* $\mathcal{D}(t) = \{i : z_i \in \mathcal{Z}_{sat},\ \neg FIRE_{z_i}(t),\ \bar{t}_i > t\}$. *These customers are* not offered to any vehicle today *and remain in the queue for future days.*

Phase 1 customers are processed first and receive priority access to vehicle capacity. Phase 2 customers fill the remaining capacity. Deferred customers are the mechanism by which the trigger *actively postpones* satellite dispatch to accumulate volume, the key distinction from the FIFO and EDD baselines.

Sorting within Phase 1. FIRE zones are ordered by a composite key that prioritises the most urgent and most distant zones:

$$\text{zone\_key}(z) = \Big(\underbrace{\bar{\delta}_z}_{\text{mean dtd (asc)}},\ \underbrace{-d(\boldsymbol{\mu}_z, \mathbf{d})}_{\text{depot dist (desc)}},\ \underbrace{-\bar{q}_z}_{\text{mean demand (desc)}}\Big), \tag{4.9}$$

where $\bar{\delta}_z$ is the average days-to-deadline and $\bar{q}_z$ the average demand across the zone's queued customers. Within each zone, customers are sorted by (δ_i asc, $-q_i$ desc)—most urgent and heaviest first.

Sorting within Phase 2. All Phase 2 candidates (core customers, overdue stragglers, and Phase 1 overflow) are sorted by an EDD key:

$$\text{key}_{\text{P2}}(i) = \Big(\underbrace{\bar{t}_i}_{\text{deadline (asc)}}, \underbrace{-q_i}_{\text{demand (desc)}} \Big). \tag{4.10}$$

The ordered Phase 1 list is concatenated before the Phase 2 list, producing a single candidate sequence in which triggered satellites appear first. A *top-K* budget ($K = \min(|\text{candidates}|,\ 20\,|\mathcal{K}|)$) truncates the list to keep insertion cost manageable.

4.4 Task 3: Sequential Vehicle Loading

The final task assigns customers to vehicles and constructs routes. Unlike classical VRP solvers that seek a globally optimal assignment, this approach is a *sequential, zone-aware heuristic* that exploits the geographic structure of the problem. The loading operates in two phases.

4.4.1 Phase 1: Triggered-Satellite Insertion

Phase 1 processes the $\mathcal{P}_1(t)$ candidates in the zone-sorted order defined by Equations (4.9). For each candidate customer, the heuristic evaluates *all* $|\mathcal{K}|$ vehicles and selects the (vehicle, position) pair with the minimum extra distance, as defined by the cheapest-insertion rule:

Definition 4.2 (Cheapest insertion). *Given a partial route $\sigma = (\mathbf{d}, c_1, \dots, c_m, \mathbf{d})$ and a candidate customer c_j, the insertion position $p^* \in \{1, \dots, m+1\}$ minimises the extra travel distance:*

$$p^* = \underset{p \in \{1,\dots,m+1\}}{\arg\min} \big(d_{c_{p-1},c_j} + d_{c_j,c_p} - d_{c_{p-1},c_p} \big), \tag{4.11}$$

subject to the feasibility constraints:

$$\textit{volume:} \quad \textstyle\sum_{i \in \sigma} q_i + q_j \le Q, \tag{4.12}$$

$$\textit{time:} \quad T(\sigma) + \Delta t_{insert}(p, j) + s_j \le H, \tag{4.13}$$

where $T(\sigma)$ is the current route time (travel + service) and $\Delta t_{insert}(p, j) = t_{c_{p-1},c_j} + t_{c_j,c_p} - t_{c_{p-1},c_p}$ is the additional travel time caused by the insertion.

If no insertion position is feasible, c_j is added to the Phase 1 overflow set, which joins Phase 2.

Because Phase 1 candidates tend to belong to the same satellite zone and appear consecutively, the cheapest-insertion heuristic naturally concentrates them on one or two vehicles, producing geographically coherent routes without an explicit zone-to-vehicle assignment.

4.4.2 Phase 2: Residual Insertion (Core + Overflow)

After Phase 1, all Phase 2 candidates—core-zone customers, overdue stragglers from non-triggered satellites, and Phase 1 overflow—are processed in EDD order (Equation (4.10)). For each candidate, the algorithm evaluates all $|\mathcal{K}|$ vehicles and selects the (vehicle, position) pair that yields the minimum additional distance, subject to the same capacity and time constraints. If no vehicle can accommodate the customer, it stays in the queue until the next day.

This two-phase structure gives triggered satellite customers first access to vehicle capacity (Phase 1), while core-zone and overflow customers fill the remaining space (Phase 2). Deferred satellite customers ($\mathcal{D}(t)$) are never offered to any vehicle, preserving future consolidation opportunities.

4.5 Complete Algorithm

Algorithm 1 summarises the full dispatch logic executed at each period t.

4.6 Parameter Calibration via PSO

The trigger policy has a small but influential parameter vector. In *volume-only* mode $\mathbf{p} = (s_v) \in [0,1]$; in *time-only* mode $\mathbf{p} = (s_t) \in [0,1]$; in *dual* mode $\mathbf{p} = (s_v, s_t) \in [0,1]^2$.

Finding the optimal parameter values analytically is intractable because the objective $J(\pi; \alpha)$ defined in Equation (3.13) depends on the interaction of stochastic arrivals, queue dynamics, heuristic routing, and capacity constraints across hundreds of simulated days. Each evaluation of J requires running a complete simulation. We therefore resort to a simulation-based meta-heuristic.

4.6.1 Particle Swarm Optimisation

We employ *Particle Swarm Optimisation* (PSO) [1], a population-based global optimiser well-suited to continuous, low-dimensional search spaces with noisy or expensive objective functions.

A swarm of P particles explores the parameter space $[0,1]^{n_{\text{dim}}}$. Each particle j maintains a position $\mathbf{x}_j^{(g)}$ and velocity $\mathbf{v}_j^{(g)}$ at generation g, updated according to:

$$\mathbf{v}_j^{(g+1)} = \omega\, \mathbf{v}_j^{(g)} + c_1\, \mathbf{r}_1 \odot \left(\mathbf{x}_j^{\text{pbest}} - \mathbf{x}_j^{(g)}\right) + c_2\, \mathbf{r}_2 \odot \left(\mathbf{x}^{\text{gbest}} - \mathbf{x}_j^{(g)}\right), \tag{4.14}$$

$$\mathbf{x}_j^{(g+1)} = \text{clip}\left(\mathbf{x}_j^{(g)} + \mathbf{v}_j^{(g+1)},\ \mathbf{0},\ \mathbf{1}\right), \tag{4.15}$$

where ω is the inertia weight, c_1 and c_2 are the cognitive and social coefficients, $\mathbf{r}_1, \mathbf{r}_2 \sim \text{Uniform}(0,1)^{n_{\text{dim}}}$ are random vectors sampled independently at each update, and $\odot$ denotes element-wise multiplication. The positions $\mathbf{x}_j^{\text{pbest}}$ and $\mathbf{x}^{\text{gbest}}$ are the best positions found by particle j and by the entire swarm, respectively.

Algorithm 1 Hybrid Trigger Policy — daily dispatch at time t

```
Require: Queue $\mathcal{Q}(t)$, fleet $\mathcal{K}$, zones $\mathcal{Z}$, slopes $(s_v, s_t)$, mode $\in$
    $\{$volume_only, time_only, dual$\}$
Ensure: Selected customers $\mathcal{S}(t)$, routes $\{\sigma_k\}_{k\in\mathcal{K}}$
    — Task 1: Trigger Evaluation —
 1: for each satellite zone $z \in \mathcal{Z}_{\text{sat}}$ do
 2:     Compute $\tau_z, f_z^{\text{vol}}, f_z^{\text{time}}$ from $\mathcal{Q}_z(t)$                    ▷ Eqs. (4.1)–(4.3)
 3:     Compute thresholds $\mathcal{T}^{\text{vol}}(\tau_z)$, $\mathcal{T}^{\text{time}}(\tau_z)$                   ▷ Eq. (4.5)
 4:     FIRE$_z$ $\leftarrow \tau_z \le 0$ or (mode uses volume and $f_z^{\text{vol}} \ge \mathcal{T}^{\text{vol}}$) or (mode uses time and
    $f_z^{\text{time}} \ge \mathcal{T}^{\text{time}}$)
 5: end for
    — Task 2: Two-Phase Queue Partitioning —
 6: Phase 1 $\leftarrow$ customers from fired satellite zones
 7: Phase 2 $\leftarrow$ core customers $\cup$ overdue WAIT-zone customers $\cup$ Phase 1 overflow
 8: Deferred $\leftarrow$ non-overdue customers in non-fired satellite zones
 9: Sort Phase 1 zones by $(\bar{\tau}_z,\ -d_z,\ -\bar{q}_z)$; within zone by $(\delta_i,\ -q_i)$
10: Sort Phase 2 by (deadline$_i$, $-q_i$)                                            ▷ EDD
11: $\mathcal{C} \leftarrow$ top-$K$ from Phase 1 $\|$ Phase 2
    — Task 3: Two-Phase Vehicle Loading —
12: Initialise routes $\sigma_k \leftarrow (\mathbf{d}, \mathbf{d})$ for all $k \in \mathcal{K}$
        *Phase 1—Triggered-Satellite Insertion:*
13: for each Phase 1 candidate $i$ (in sort order) do
14:     $(k^*, p^*) \leftarrow \arg\min_k$ CheapestInsertion$(\sigma_k, i)$          ▷ global across all vehicles
15:     if feasible then insert $i$ into $\sigma_{k^*}$ at $p^*$
16:     else move $i$ to Phase 2 list
17:     end if
18: end for
        *Phase 2—Residual Insertion (core + overflow):*
19: for each Phase 2 candidate $i$ (EDD order) do
20:     $(k^*, p^*) \leftarrow \arg\min_k$ CheapestInsertion$(\sigma_k, i)$ subject to Eqs. (4.12)–(4.13)
21:     if feasible then insert into $\sigma_{k^*}$
22:     else $i$ stays in queue
23:     end if
24: end for
25: $\mathcal{S}(t) \leftarrow$ all customers assigned to some $\sigma_k$
26: return $\mathcal{S}(t)$, $\{\sigma_k\}_{k\in\mathcal{K}}$
```

4.6.2 Calibration Protocol

The calibration is performed *once* on the HIGH scenario ($\rho = 1.20$, $\Delta_i \sim$ Uniform$\{1,2,3\}$) and the resulting parameters are used without re-tuning across all six test scenarios. This “train-once, test-everywhere” design reflects the practical constraint that a logistics operator calibrates the policy on a representative demand pattern and then deploys it as conditions evolve.

The HIGH scenario was chosen as the training environment because it is the only regime where the trade-off between consolidation and urgency is clearly visible: with $\rho > 1$ the system is structurally overloaded, so the trigger slope has a meaningful effect on which customers are deferred and which are served. At lower congestion levels the objective landscape is nearly flat, and all slope values yield similar costs.

Table 4.1. PSO calibration settings.

Parameter	Symbol	Value
Number of particles	P	20
Maximum iterations	G	30
Early-stop stall	—	10 iterations
Inertia	ω	0.70
Cognitive coefficient	c_1	1.50
Social coefficient	c_2	1.50
Training horizon	T	1 500 days
Training seed	—	42
Fleet size	$\lvert\mathcal{K}\rvert$	3
Scenario	—	HIGH

Each particle evaluation consists of a full simulation of the dispatch system over $T =$ 1,500 days. With 20 particles, 30 iterations, and an additional 20-particle initialisation round, the calibration evaluates approximately $20 + 20 \times 30 = 620$ candidate parameter vectors. On the hardware used (Apple M-series, single core), each evaluation takes ~ 0.5 seconds, yielding a total calibration time of approximately 5 minutes per profile.

4.6.3 Grid Search for the Volume-Only Case

In the *volume-only* mode the parameter space reduces to a single continuous variable $s_v \in [0, 1]$. For such a low-dimensional problem, an exhaustive *grid search* provides a simpler and cheaper alternative to PSO. The search proceeds in two stages:

1. **Coarse sweep.** The interval $[0, 1]$ is discretised into $N_c = 51$ uniformly spaced points. Each candidate is evaluated by running a full simulation under the same protocol as PSO ($T =$ 1,500 days, seed 42, HIGH scenario). The point $s_v^{(c)}$ with the lowest objective value is retained.

2. **Local refinement.** A finer grid of $N_r = 21$ points is placed over the interval $[s_v^{(c)} - \Delta,\ s_v^{(c)} + \Delta]$, where Δ is the coarse step size (≈ 0.02). The best point from this stage is the final solution s_v^*.

The total budget is $N_c + N_r = 72$ simulation evaluations—roughly one ninth of the $\approx$ 620 evaluations used by PSO—yielding a calibration time of approximately 36 seconds per profile. Because the objective landscape along a single dimension is smooth and unimodal in practice, the grid search recovers the same optima as PSO while being substantially faster and fully deterministic.

For the *dual* mode ($s_v, s_t \in [0,1]^2$), a two-dimensional grid would require $51^2 \approx 2{,}600$ evaluations, making PSO the more efficient choice. The calibration therefore uses **grid search for the one-dimensional modes** (volume-only and time-only) and **PSO for the dual** mode.

4.6.4 Calibration Profiles

Three calibration runs are performed for each trigger mode, each with a different trade-off weight α:

Table 4.2. Calibration profiles and resulting parameters (1 500 days, 3 vehicles, HIGH scenario). Volume-only and time-only profiles were calibrated with grid search (51 points); dual profiles with PSO.

Profile	α	Mode	s_v^*	s_t^*	J^*	Interpretation
Service	0.3	volume-only	0.666	—	0.502	Moderate volume gating
Balanced	0.5	volume-only	0.842	—	0.466	Volume-based consolidation
Efficiency	0.7	volume-only	0.926	—	0.429	Strong volume gating
Service	0.3	time-only	—	0.996	0.503	Near-maximum time threshold
Balanced	0.5	time-only	—	0.996	0.468	Near-maximum time threshold
Efficiency	0.7	time-only	—	0.996	0.433	Near-maximum time threshold
Service	0.3	dual	0.752	0.999	0.503	$s_t \to 1$: time channel disabled
Balanced	0.5	dual	0.776	1.000	0.468	$s_t = 1$: time channel disabled
Efficiency	0.7	dual	0.977	1.000	0.433	$s_t = 1$: time channel disabled

Several patterns emerge from the calibrated parameters:

- As α increases (more weight on travel), the volume slope increases: the policy becomes more selective, postponing satellite dispatch until loads are fuller.
- The time-only mode calibrates to $s_t \approx 1.0$ regardless of α. At this value the time threshold is nearly equal to the urgency ratio, so the trigger fires only when the nearest-neighbour route estimate almost fills the fleet's total time budget—a condition that rarely occurs. The time signal is therefore too coarse to differentiate the three trade-off profiles.
- In the dual mode, the PSO optimiser *autonomously pushes* $s_t \to 1$ in every profile, effectively disabling the time channel. The volume slopes closely track the volume-only calibration.
- The volume-only mode achieves the lowest cost in every profile, confirming that the richer dual parameterisation does not improve upon a single, well-tuned volume threshold.

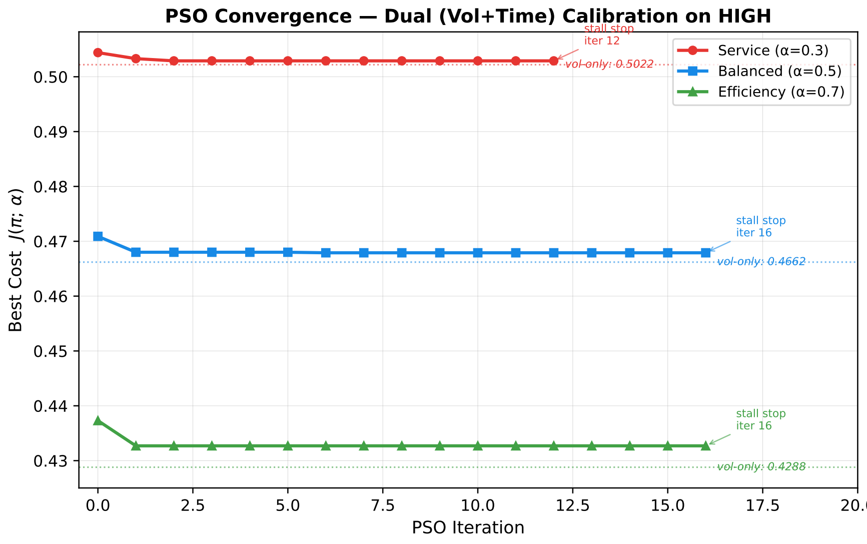


Figure 4.3. PSO convergence for the three α-profiles (dual mode, HIGH scenario). All profiles converge within 10–12 iterations. Dotted lines mark the corresponding volume-only grid-search optimum.

4.7 Policy Variants Summary

Combining three trigger modes with three α-profiles yields nine Hybrid policy variants. Together with the two baselines, the experimental evaluation in Chapter 5 compares eleven policies (Table 4.3).

Table 4.3. Complete policy set. Hybrid variants differ in trigger mode and trade-off profile; baselines use no trigger.

Policy	Strategy	Trigger Mode	Profile
FIFO	FIFO	—	—
EDD	EDD	—	—
HybridV_service	HYBRID	volume-only	service ($\alpha = 0.3$)
HybridV_balanced	HYBRID	volume-only	balanced ($\alpha = 0.5$)
HybridV_efficiency	HYBRID	volume-only	efficiency ($\alpha = 0.7$)
HybridT_service	HYBRID	time-only	service ($\alpha = 0.3$)
HybridT_balanced	HYBRID	time-only	balanced ($\alpha = 0.5$)
HybridT_efficiency	HYBRID	time-only	efficiency ($\alpha = 0.7$)
HybridD_service	HYBRID	dual	service ($\alpha = 0.3$)
HybridD_balanced	HYBRID	dual	balanced ($\alpha = 0.5$)
HybridD_efficiency	HYBRID	dual	efficiency ($\alpha = 0.7$)

The next chapter presents the experimental evaluation of all eleven policies across the six scenarios defined in Section 3.7.

Chapter 5

Experimental Evaluation

This chapter tests the Hybrid Trigger Policy and its three trigger modes—volume-only, time-only, and dual—against two standard baselines in a controlled simulation study. The analysis is built around three questions:

1. Does the time-based trigger channel add value, or is the simpler volume-only sufficient?
2. How much better does the Hybrid policy perform compared to standard dispatching rules?
3. Do parameters calibrated on a single scenario still work well under very different operating conditions?

The chapter is organised as follows. Section 5.1 describes the simulation environment and the experimental design. Section 5.2 compares the three trigger modes. Sections 5.3–5.3.4 present the scenario-by-scenario results. Section 5.4 summarises the main findings.

5.1 Experimental Setup

5.1.1 Fleet, Geography, and Demand

The simulation models a courier company that dispatches a homogeneous fleet of $K = 3$ vehicles every morning from a depot at coordinates $(25, 10)$. Each vehicle can carry $V_{\max} = 250$ units and drive for at most $T_{\max} = 10$ hours at a fixed speed of 50 km/h. Customer demands are generated from $U[10, 50]$ units (mean 30) and service times from $U[0.25, 2.0]$ hours.

The service region consists of four demand zones (see Table 3.1 and Figure 3.1). Zone 1, the urban core, sits about 7 km from the depot and produces most of the daily volume ($\lambda = 3.0$ arrivals per period). Zones 2–4 are more remote satellites. A round trip to the farthest satellite (Z4, at $\sim$121 km) takes several hours and is worthwhile only when enough customers have accumulated there.

5.1.2 Policies Under Comparison

We evaluate eleven policies. Two are conventional baselines described in Section 3.8.3: **FIFO**, which sorts customers by arrival time (with zone and demand as tie-breakers), and **EDD**, which sorts by deadline (earliest first) using the same secondary keys. Both baselines insert their sorted queue into the cheapest-insertion heuristic. Because the same routing engine is used for every policy, any performance difference is entirely due to the ordering in which customers are considered.

The remaining nine policies are Hybrid variants obtained by crossing three trigger modes (volume-only, time-only, dual) with three α-profiles (service, balanced, efficiency); see Table 4.3 for the full list. All parameters were calibrated once on the HIGH scenario as described in Section 4.6.4 and are used without re-tuning in every test scenario.

5.1.3 Scenario Design

We use a 2×3 factorial design that varies two stress factors—congestion and deadline pressure—each at three levels (Table 5.1). The HIGH scenario ($\rho = 1.20$, deadlines $U[1,3]$) is the calibration environment; every other scenario is used for out-of-sample evaluation.

Table 5.1. Six experimental scenarios. The HIGH scenario (bold) is the training environment.

Scenario	ρ	Deadline	Role
Congestion axis (deadlines $U[1,3]$*)*			
Low	0.70	$U[1,3]$	test
Medium	0.90	$U[1,3]$	test
High	**1.20**	**U[1,3]**	**train**
Deadline axis ($\rho = 0.90$*)*			
Relaxed	0.90	$U[3,5]$	test
Tight	0.90	$U[0,2]$	test
Critical	0.90	$U[0,1]$	test

Every combination of policy and scenario is replicated with 10 independent random seeds over 200 simulated days, giving a total of $11 \times 6 \times 10 = 660$ simulation runs. All reported figures are averages over these replications.

5.2 The Three Trigger Modes Compared

Before looking at individual scenarios, it is useful to compare the three trigger architectures at a global level. Table 5.2 reports the balanced cost J_α averaged across all six scenarios, which gives a single summary of each policy's robustness.

Table 5.2. Global balanced cost $J_{0.5}$ averaged over all six scenarios and 10 seeds. Lower is better.

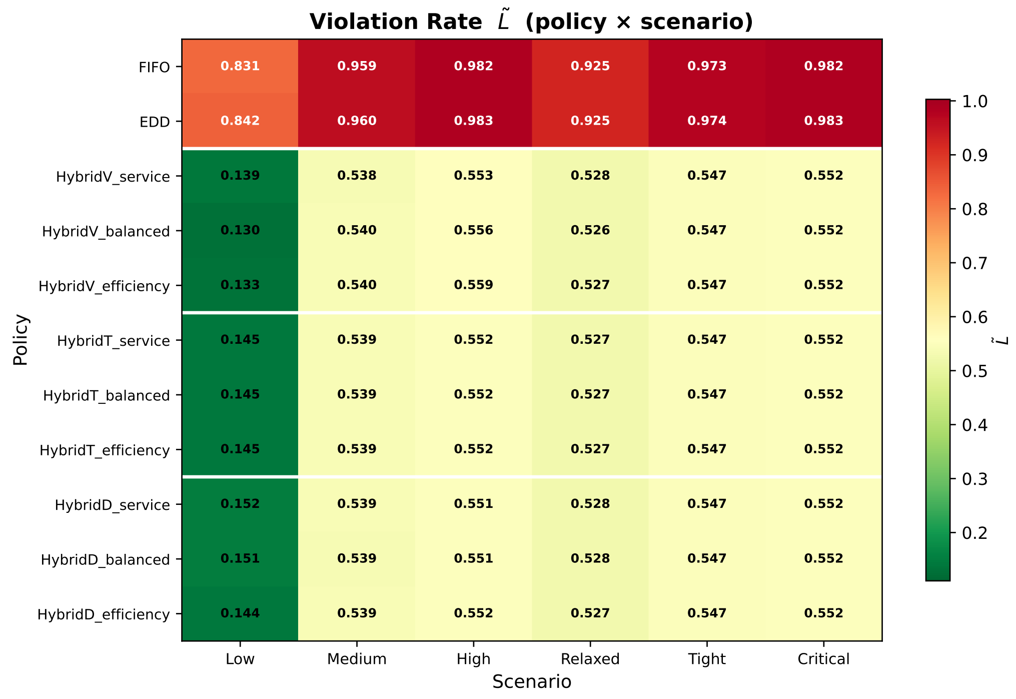
Policy	Mean $J_{0.5}$
HybridV_balanced	**0.4058**
HybridV_efficiency	0.4058
HybridV_service	0.4072
HybridD_efficiency	0.4084
HybridT_balanced	0.4086
HybridT_efficiency	0.4086
HybridT_service	0.4086
HybridD_balanced	0.4092
HybridD_service	0.4093
FIFO	0.6445
EDD	0.6460

From this results we can notice that the volume-only is the best trigger mode: the top three rows are all HybridV variants, and although the gap over the time-only and dual families is small in absolute terms ($\sim$0.003, or about 0.7%). Second, the time-only mode shows no profile differentiation at all. The three HybridT variants produce *identical* results to four decimal places; the reason is visible in the calibration table (Table 4.2), where all three profiles converge to $s_t \approx 1.0$. A threshold that high means the time channel never makes a meaningful decision—in practice, the trigger fires only when $\tau_z \leq 0$ (the urgency override), which shows no difference between the three profiles. Third, the dual mode collapses to volume-only: the PSO optimiser pushed $s_t \to 1.0$ in every profile (Table 4.2), disabling the time channel. The surviving volume slopes ($s_v = 0.752, 0.776, 0.977$) are close to but not identical to the pure volume-only values, which explains why HybridD is slightly worse—the two-dimensional search introduces noise without gaining information.

The time-based signal return to be redundant because the volume of orders already reflects how urgent the situation is. If a deadline is too close, the system overrides the trigger anyway; if many urgent orders arrive, they fill up the volume fraction quickly enough to start the dispatch. A separate time-based calculation adds more noise than useful data.

For this reason, the volume-only mode requires calibrating a single parameter for profile, uses a simple grid search instead of PSO, and dominates the alternatives. The rest of this chapter therefore focuses on the HybridV variants when comparing against baselines, though all eleven policies are included in the result tables for completeness.

Inside the figure 5.1 is available an overview of the violation rate across all eleven policies and six scenarios. The colour gradient makes two patterns immediately visible: (i) all Hybrid rows are substantially greener than the baseline rows, and (ii) within the Hybrid block the three families are nearly indistinguishable, confirming the redundancy of the time channel.



Policy	Low	Medium	High	Relaxed	Tight	Critical
FIFO	0.831	0.959	0.982	0.925	0.973	0.982
EDD	0.842	0.960	0.983	0.925	0.974	0.983
HybridV_service	0.139	0.538	0.553	0.528	0.547	0.552
HybridV_balanced	0.130	0.540	0.556	0.526	0.547	0.552
HybridV_efficiency	0.133	0.540	0.559	0.527	0.547	0.552
HybridT_service	0.145	0.539	0.552	0.527	0.547	0.552
HybridT_balanced	0.145	0.539	0.552	0.527	0.547	0.552
HybridT_efficiency	0.145	0.539	0.552	0.527	0.547	0.552
HybridD_service	0.152	0.539	0.551	0.528	0.547	0.552
HybridD_balanced	0.151	0.539	0.551	0.528	0.547	0.552
HybridD_efficiency	0.144	0.539	0.552	0.527	0.547	0.552

Figure 5.1. Violation rate $\tilde{L}$ for every combination of policy (rows) and scenario (columns). Cell values are averages over 10 seeds. White horizontal lines separate the four policy families. The colour scale runs from green (low violation) to red (high).

5.3 Congestion Scenarios

We try different congestion scenarios varing the arrival-rate multiplier ρ while keeping deadlines at $U[1{,}3]$. Figure 5.2 summarises the late rate across all six scenarios, aggregated by policy family. The gap between baselines and Hybrid is high in every scenario, while the three Hybrid families cluster together.

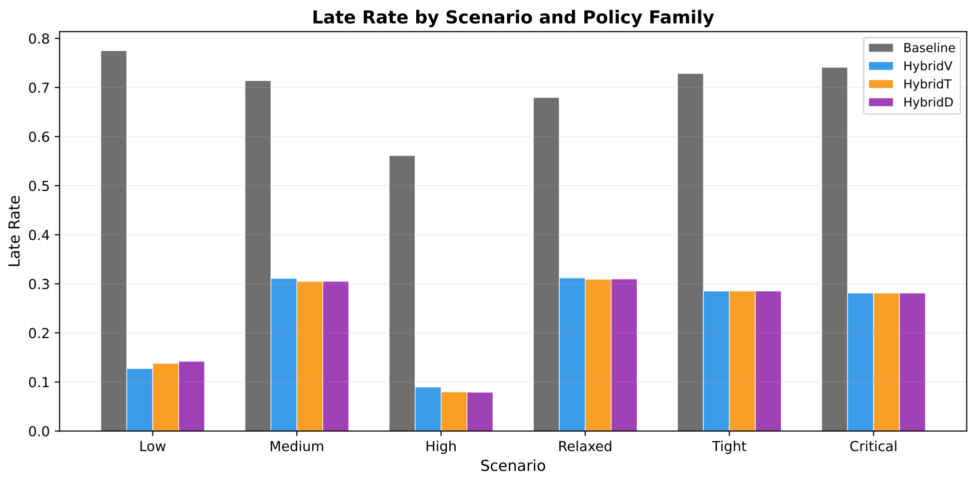


Figure 5.2. Late rate by scenario and policy family (Baseline, HybridV, HybridT, HybridD). Each bar is the family-level average over profiles and 10 seeds. The Hybrid policies reduce late deliveries by 40–65 percentage points relative to FIFO/EDD.

5.3.1 Low Congestion ($\rho = 0.70$)

Table 5.3. Low scenario ($\rho = 0.70$, deadlines $U[1,3]$).

Policy	Travel (h/d)	Late Rate	Unserved %	Throughput/d
FIFO	10.90	0.770	6.1%	16.5
EDD	10.92	0.780	6.2%	16.5
HybridV_service	8.59	0.132	0.7%	17.5
HybridV_balanced	8.49	0.124	0.7%	17.5
HybridV_efficiency	8.47	0.126	0.6%	17.5
HybridT (all)	8.70	0.138	0.7%	17.5
HybridD_service	8.71	0.145	0.7%	17.5
HybridD_balanced	8.70	0.144	0.7%	17.5
HybridD_efficiency	8.69	0.137	0.7%	17.5

When demand is below fleet capacity, the Hybrid policy works well. The best variant (HybridV_efficiency) delivers just 12.6% of customers late and leaves only 0.6% unserved, while travelling 22% less than FIFO (8.47 vs. 10.90 h/d).

The baselines, by contrast, are surprisingly poor: FIFO produces a late rate of 77% even though the fleet could handle all the demand. The problem is related to the geography. Without any spatial awareness, FIFO and EDD send vehicles across remote zones every day (sometimes sending a truck to Z4 for a single parcel) and waste hours on half-empty round trips.

The Hybrid avoids this by waiting until a satellite zone has accumulated enough volume to justify the trip. In a low-congestion environment, deadlines are rarely urgent enough to override the trigger, so the consolidation mechanism operates at full strength.

5.3.2 Medium Congestion ($\rho = 0.90$)

Table 5.4. Medium scenario ($\rho = 0.90$, deadlines $U[1,3]$).

Policy	Travel (h/d)	Late Rate	Unserved %	Throughput/d
FIFO	10.40	0.714	24.5%	17.0
EDD	10.42	0.714	24.6%	17.0
HybridV_service	9.80	0.307	23.1%	17.3
HybridV_balanced	9.70	0.312	22.8%	17.4
HybridV_efficiency	9.65	0.314	22.6%	17.5
HybridT (all)	9.89	0.305	23.4%	17.3
HybridD_service	9.90	0.305	23.4%	17.3
HybridD_balanced	9.89	0.305	23.4%	17.3
HybridD_efficiency	9.88	0.305	23.4%	17.3

At $\rho = 0.90$ the system is close to saturation and about a quarter of all arrivals go unserved regardless of the policy. The Hybrid still cuts the late rate in half (from 71% to 31%) and uses 7% less travel time than FIFO.

HybridV_efficiency is the best variant here: it achieves the lowest travel (9.65 h/d) and the lowest unserved rate (22.6%) among all Hybrid policies. An interesting detail is that HybridT and HybridD produce a marginally lower late rate (30.5% vs. 31.4%) but at the cost of higher unserved rates and more travel time—a sign that those modes trigger satellite dispatches slightly too often, sending vehicles out before the load is dense enough.

5.3.3 High congestion ($\rho = 1.20$, $U[1,3]$)

High is the training scenario: these are the operating conditions on which all Hybrid parameters were calibrated. With a demand-to-capacity ratio of 1.20, the fleet is structurally undersized, more than 40% of customers will go unserved, no matter which policy is used. We can check how the remaining capacity is spent.

Table 5.5. High scenario ($\rho = 1.20$, $U[1,3]$): all 11 policies.

Policy	**Travel (h/d)**	**Late Rate**	**Unserved %**	**Thru./d**
FIFO	9.94	0.561	42.2%	17.4
EDD	9.94	0.562	42.1%	17.5
HybridV_service	11.34	0.085	46.8%	16.0
HybridV_balanced	11.28	0.090	46.7%	16.1
HybridV_efficiency	11.21	0.095	46.4%	16.1
HybridT_service	11.48	0.080	47.2%	15.9
HybridT_balanced	11.48	0.080	47.2%	15.9
HybridT_efficiency	11.48	0.080	47.2%	15.9
HybridD_service	11.48	0.080	47.3%	15.9
HybridD_balanced	11.47	0.080	47.2%	15.9
HybridD_efficiency	11.47	0.079	47.2%	15.9

Table 5.5 shows a trade-off that does not appear at lower congestion levels. The Hybrid policies use more travel time than the baselines (11.3 h/d vs. 9.9) and serve fewer customers overall (46–47% unserved vs. 42%). In exchange, the customers they do serve are far more likely to be on time: the late rate drops from 56% under FIFO to less than 9% under HybridV and around 8% under HybridT/D.

Under overload the trigger still waits for remote satellites to accumulate volume, which means vehicles sometimes depart with partially empty capacity that could have been used for closer customers. The baselines fill every vehicle indiscriminately and therefore serve more people in total, but most of those deliveries arrive after the deadline. The Hybrid instead concentrates resources on feasible deliveries, accepting a throughput penalty for a large improvement in service quality.

All three trigger modes converge to nearly identical performance here, because the calibration was done on this very scenario: the learned thresholds are perfectly matched to the demand pattern.

Figure 5.3 shows the seed-level variability for this scenario. The boxplots confirm that the late-rate improvement is consistent across all 10 replications: the interquartile ranges for the Hybrid variants are tight and well separated from the baselines. The travel-time panel shows the corresponding cost: the Hybrid policies use roughly 1.3 h/d more than FIFO, a consequence of the trigger routing vehicles to remote satellites on a schedule.

5.3.4 The deadline scenarios

The three deadline scenarios keep congestion fixed at $\rho = 0.90$ (the same volume as Medium) and vary only the deadline window. This isolates the effect of temporal pressure on the trigger's consolidation strategy.

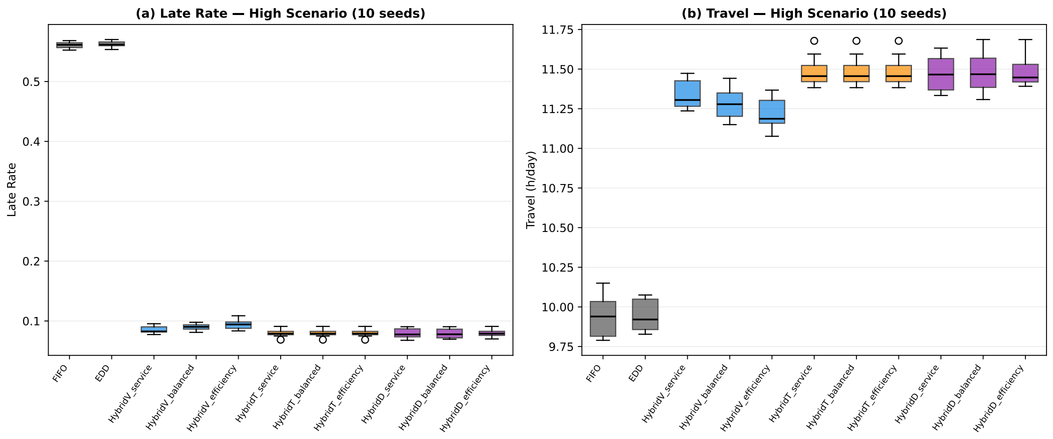


Figure 5.3. Seed-level variability on the High scenario ($\rho = 1.20$, $U[1,3]$, 10 seeds). (a) Late rate: all nine Hybrid variants cluster below 0.10, while both baselines sit near 0.56. (b) Travel: the Hybrid policies use 11.2–11.5 h/d vs. 9.9 h/d for FIFO, reflecting the consolidation detours.

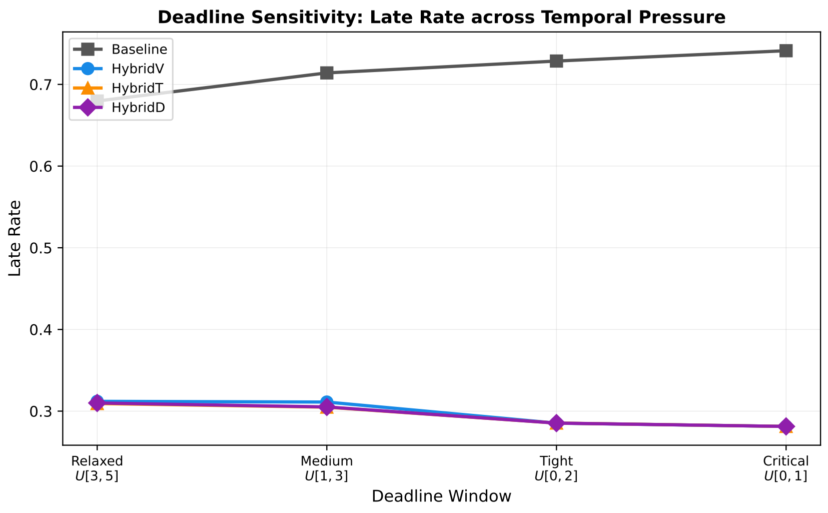


Figure 5.4. Late rate vs. deadline pressure ($\rho = 0.90$ for all four scenarios, averaged over 10 seeds). As deadlines tighten, the Hybrid curves rise but remain well below the baselines. In the Critical scenario all Hybrid variants converge to ∼28%.

Figure 5.4 plots the late rate as a function of increasing deadline pressure. The baselines are nearly insensitive to the deadline window, their late rate stays between 68% and 74%, while the Hybrid policies show a clear upward slope as deadlines tighten. The flattening between Tight and Critical reflects the trigger's saturation: once every zone fires daily, further deadline compression has no additional effect.

Relaxed (U[3,5])

Table 5.6. Relaxed scenario ($\rho = 0.90$, U[3,5]): all 11 policies.

Policy	**Travel (h/d)**	**Late Rate**	**Unserved %**	**Thru./d**
FIFO	10.40	0.680	24.5%	17.0
EDD	10.42	0.679	24.6%	17.0
HybridV_service	9.37	0.312	21.6%	17.7
HybridV_balanced	9.30	0.311	21.5%	17.7
HybridV_efficiency	9.27	0.313	21.4%	17.7
HybridT_service	9.39	0.309	21.8%	17.6
HybridT_balanced	9.39	0.309	21.8%	17.6
HybridT_efficiency	9.39	0.309	21.8%	17.6
HybridD_service	9.41	0.310	21.9%	17.6
HybridD_balanced	9.38	0.310	21.7%	17.7
HybridD_efficiency	9.38	0.310	21.7%	17.7

With deadlines extending to five days, the trigger has plenty of room to wait before dispatching remote satellites. HybridV_efficiency achieves the lowest travel among all medium-congestion scenarios (9.27 h/d, 11% below FIFO) with a late rate of 31% and an unserved rate three points lower than the baselines. The generous temporal window gives the consolidation mechanism what it needs to form dense, efficient routes.

Tight ($U[0,2]$)

Table 5.7. Tight scenario ($\rho = 0.90$, $U[0,2]$): all 11 policies.

Policy	**Travel (h/d)**	**Late Rate**	**Unserved %**	**Thru./d**
FIFO	10.40	0.729	24.5%	17.0
EDD	10.42	0.729	24.6%	17.0
HybridV_service	10.69	0.285	26.2%	16.7
HybridV_balanced	10.69	0.285	26.2%	16.7
HybridV_efficiency	10.69	0.285	26.2%	16.7
HybridT_service	10.69	0.285	26.2%	16.7
HybridT_balanced	10.69	0.285	26.2%	16.7
HybridT_efficiency	10.69	0.285	26.2%	16.7
HybridD_service	10.69	0.285	26.2%	16.7
HybridD_balanced	10.69	0.285	26.2%	16.7
HybridD_efficiency	10.69	0.285	26.2%	16.7

When most deadlines expire within two days, the picture changes substantially. All nine Hybrid variants collapse onto exactly the same performance: 10.69 h/d of travel, 28.5% late, 26.2% unserved. The trigger fires so frequently that the differences between modes and profiles are completely washed out. Travel time is now higher than FIFO (10.69 vs. 10.40) and the unserved rate is slightly worse (26.2% vs. 24.5%), because vehicles still detour to service remote satellites even when there is almost no consolidation benefit. Nevertheless, the late rate is more than the half (28.5% vs. 72.9%), indeed the Phase 1/Phase 2/Deferred loading mechanism continues to prioritise urgent customers regardless of how the trigger behaves.

Critical ($U[0,1]$)

Table 5.8. Critical scenario ($\rho = 0.90$, $U[0,1]$): all 11 policies.

Policy	**Travel (h/d)**	**Late Rate**	**Unserved %**	**Thru./d**
FIFO	10.40	0.742	24.1%	16.9
EDD	10.43	0.741	24.2%	16.9
HybridV_service	11.09	0.281	27.1%	16.3
HybridV_balanced	11.09	0.281	27.1%	16.3
HybridV_efficiency	11.09	0.281	27.1%	16.3
HybridT_service	11.09	0.281	27.1%	16.3
HybridT_balanced	11.09	0.281	27.1%	16.3
HybridT_efficiency	11.09	0.281	27.1%	16.3
HybridD_service	11.09	0.281	27.1%	16.3
HybridD_balanced	11.09	0.281	27.1%	16.3
HybridD_efficiency	11.09	0.281	27.1%	16.3

Under same-day or next-day deadlines all nine Hybrid policies produce identical numbers. The trigger is effectively neutralised, because the urgency component always exceeds the threshold, forcing dispatch to every satellite every day.

Yet even here the Hybrid late deliveries drop from 74% to 28%. The explanation is that the Phase 1/Phase 2/Deferred loading order, which is independent of the trigger, keeps placing the most urgent customers first on each vehicle.

Figure 5.5 shows the trade-off between travel time and violation rate for each scenario. The baselines occupy the upper part of each panel (high violation, moderate travel), while the Hybrid policies push toward the lower-left corner. In the Relaxed scenario the Hybrid Pareto front is clearly separated; in Tight and Critical the points collapse to a single cluster as the trigger saturates.

5.4 Discussion

After reporting the results of the 660 simulations, we can underline few findings:

- In every scenario the most impactful factor is whether the policy groups nearby customers before dispatching. Even in the Low scenario, where the fleet has spare capacity, FIFO delivers 77% of customers late because it sends vehicles to remote zones with half-empty loads. The Hybrid policies, by waiting for a critical mass in each satellite, bring that figure below 13%.

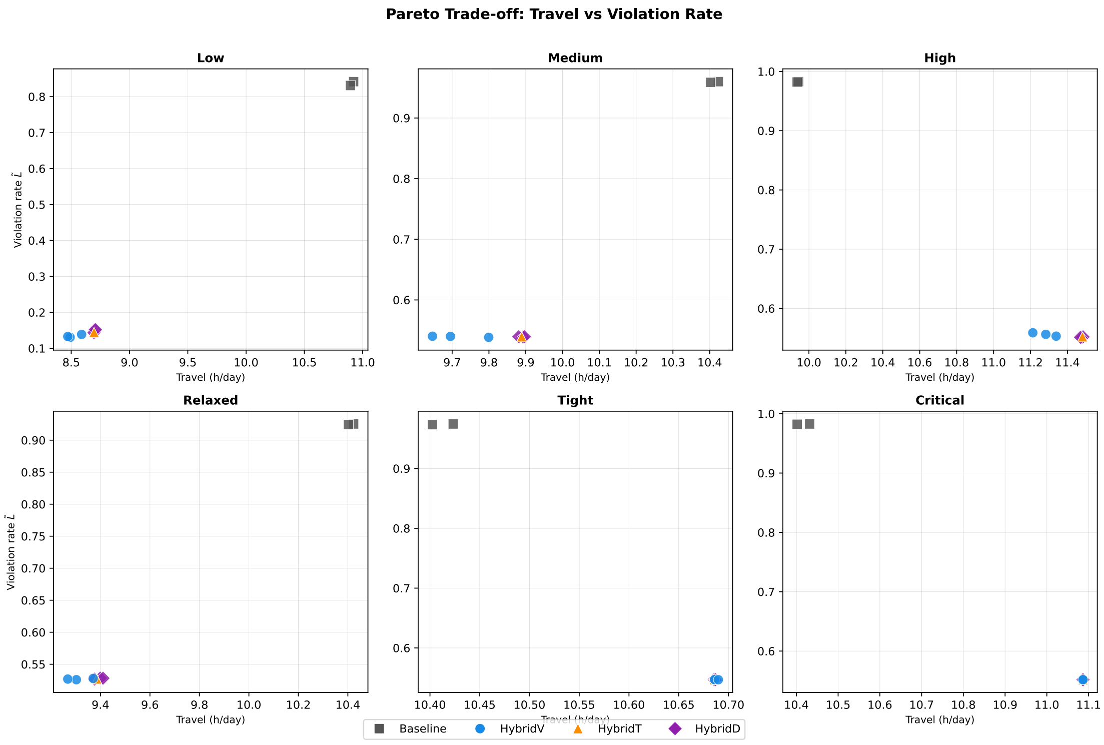


Figure 5.5. Travel time vs. violation rate $\tilde{L}$ for each scenario, averaged over 10 seeds. Each marker is one of the 11 policies, coloured by family. Lower-left is better. At low congestion the Hybrid policies dominate on both axes; under overload (High) they trade throughput for timeliness.

- The time-only trigger calibrates to $s_t \approx 1.0$ in every profile, disabling itself; the dual mode converges to the same behaviour. HybridV dominates globally (cost_balanced = 0.406 for the balanced profile). In practice, the queue length already contains all the temporal information the dispatcher needs: when a satellite grows fast the deadlines are approaching automatically.
- In the Relaxed scenario ($U[3,5]$), HybridV_efficiency cuts travel by 11% relative to FIFO while minimizing the late rate. As deadlines tighten, the trigger fires more often and we can benefit from the consolidation; in Critical ($U[0,1]$) all nine Hybrid variants produce identical numbers, because dispatch is forced daily to every satellite. Even in this limit case the late rate still drops from 74% to 28%—the Phase 1/Phase 2/Deferred loading order continues to prioritise urgent customers independently of the trigger.
- In the High scenario ($\rho = 1.20$) the policy serves fewer customers than FIFO (47% unserved vs. 42%) but the customers it *does* serve arrive on time far more often (late rate 9% vs. 56%). Whether this trade-off is acceptable depends on the operator's priorities, but the option is available through the α weight in the cost function.

- All Hybrid parameters were calibrated on the High scenario only (1 500 days, seed 42). Despite this, the policies exhibit stable advantages across all six test scenarios, including Low ($\rho = 0.70$) and Critical ($U[0,1]$). The learned thresholds encode a structural property of the cluster geography rather than an artefact of the specific demand level.
- FIFO and EDD produce nearly identical results in every scenario. The EDD policy sorts by deadline but does not consolidate spatially, so its sorting effort is wasted when the fleet visits the same geographic scatter as FIFO.

Chapter 6

Conclusions

This thesis has studied a stochastic capacitated vehicle routing problem in which customer requests arrive over time and carry individual delivery deadlines. The operational setting reflects a realistic urban distribution context: a depot serves a dense core zone and three progressively more remote satellite zones, and the dispatcher must decide each day which areas to serve immediately and which to postpone.

The central contribution of the work is a dispatching framework that separates two decisions. The first is a *selection* step: a zone-level trigger evaluates how much demand has accumulated in each satellite cluster relative to the fleet's capacity, and compares this fraction to an adaptive threshold that decreases as deadlines approach. If the fraction exceeds the threshold—or if any customer in the zone is already overdue—the zone is marked for dispatch. The second is a *loading* step: customers are partitioned into those belonging to triggered zones (Phase 1), those in the core or overdue in non-triggered zones (Phase 2), and those who are deferred to a future day. Vehicles are filled in this order, using a cheapest-insertion heuristic.

The experimental evaluation covered 660 simulation runs across six scenarios, varying both the congestion level (arrival rate multiplier ρ) and the deadline pressure (window width). The most important finding is that the volume-only trigger mode—which calibrates a single slope parameter s_v through a one-dimensional grid search—is sufficient and in fact dominates the more complex alternatives. Adding a time-based channel did not improve performance in any scenario: the calibration in both the time-only and dual modes consistently converged to the same solution found by the volume-only mode, effectively disabling the additional channel. This result is not only practically convenient—it eliminates one parameter and reduces calibration to a simple grid search—but also theoretically meaningful: it confirms that the volume fraction already captures urgency implicitly, because tight deadlines cause demand to accumulate rapidly and push the trigger past its threshold.

Two further structural properties emerged from the results. The benefit of spatial consolidation is closely tied to the amount of temporal slack available: when deadlines are loose, the trigger can afford to wait, form dense routes, and substantially reduce travel time; when deadlines are tight, the urgency override fires before the cluster is dense, and the advantage over the baselines narrows. Independently of these dynamics, the two-phase loading order provides a stable performance floor: even when the trigger fires every day across all zones, Phase 1 customers are still loaded before Phase 2, which alone halves the late rate relative to the FIFO and EDD baselines. This confirms that sorting *who* gets on the vehicle matters as much as deciding *whether* to send it.

Several directions remain open for future work. The most natural extension would be to replace the fixed threshold with a rolling-horizon lookahead: since each daily dispatch takes under one millisecond, the trigger policy could serve as the base policy for a rollout algorithm that samples the next day's likely arrivals and uses them to decide whether dispatching today is worthwhile. On the modelling side, incorporating time-dependent travel times and real road networks would bring the framework closer to operational deployment, as would testing the system under non-stationary demand patterns such as seasonal peaks or promotional bursts. Finally, the two-phase loading heuristic currently treats deferred customers as passive; allowing the system to consider predicted future arrivals when making the deferral decision could further reduce violations under tight deadline conditions.

Bibliography

[1] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. ieee, 1995.

[2] Subei Mutailifu, Paolo Brandimarte, and Aili Maimaiti. Learning decision rules for a stochastic multiperiod capacitated traveling salesperson problem with irregularly clustered customers. *Logistics*, 9(3):119, 2025.

[3] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225 (1):1–11, 2013.

[4] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[5] Warren B Powell. Clearing the jungle of stochastic optimization. In *Bridging data and decisions*, pages 109–137. Informs, 2014.

[6] Warren B Powell. A unified framework for stochastic optimization. *European journal of operational research*, 275(3):795–821, 2019.

[7] Harilaos N Psaraftis. A dynamic programming approach for sequencing groups of identical jobs. *Operations Research*, 28(6):1347–1359, 1980.

[8] Harilaos N Psaraftis, Min Wen, and Christos A Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2016.

[9] Paolo Toth and Daniele Vigo. *The vehicle routing problem.* SIAM, 2002.

[10] Min Wen, Jean-François Cordeau, Gilbert Laporte, and Jesper Larsen. The dynamic multi-period vehicle routing problem. *Computers & Operations Research*, 37(9):1615–1623, 2010.