Universiteit Leiden

Multi-Objective Stochastic Electric Vehicle Routing Problem with Time Windows

Johannes Leppäkorpi (s3856348)

Supervisors:
Yingjie Fan & Wei Liu

Bachelor Thesis– Data Science & AI

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

July 1, 2026

Abstract

Most existing formulations of the Electric Vehicle Routing Problem with Time Windows (EVRPTW) assume a deterministic model with known travel times, energy consumption, and vehicle availability. Real-world delivery systems are subject to disruptions: traffic congestion, mechanical failure, and battery depletion can propagate through a route and produce late service. As such, deterministic models must be adapted to this stochastic reality to maintain operational resilience and service quality.

This thesis introduces the Multi-Objective Stochastic EVRPTW (MOS-EVRPTW), a framework that jointly optimizes expected energy consumption, time-window violation, and resilience. The problem is formulated as a two-stage stochastic program. First-stage routes are fixed before uncertainty is realized; second-stage recourse accounts for traffic congestion, vehicle breakdowns, and battery-depletion events during execution. Because the expected-value objectives have no closed form, the program is solved as a simheuristic, where candidate routes are evolved with the NSGA-II multi-objective genetic algorithm, and each candidate is evaluated by Monte Carlo (MC) simulation across many disruption scenarios. The energy consumption model uses a simplified version of a detailed physics model by Goeke and Schneider[14]. We conducted computational experiments on Goeke EVRPTW instances of up to 100 customers, comparing the stochastic Pareto-front knee point against a deterministic VRPy baseline re-evaluated under the same scenarios. Across the tested instances, the MOS-EVRPTW improves expected energy and resilience over the deterministic baseline, with the resilience advantage widening as disruption intensity increases; time-window performance depends strongly on customer geometry, time-window flexibility, and disruption levels.

Contents

1 Introduction

The transition toward electric mobility is accelerating. There are environmental concerns fueled by stricter regulations that are driving a growing interest in Electric Vehicles (EVs) for the distribution of goods, specifically for last-mile deliveries where EVs can operate with great efficiency. EVs offer a path to cleaner last-mile delivery, but they introduce operational challenges that conventional vehicles do not face: limited driving range, charging stations with long charging times, and energy consumption that varies with environmental and vehicle conditions.

These challenges are captured by the EVRPTW, which extends the classical VRP by adding battery capacity constraints, charging station visits, and time-window adherence[25]. Significant progress has been made on the deterministic variant of this problem, including partial recharge strategies[18]. However, the vast majority of EVRPTW models assume a deterministic environment with fixed travel times, energy consumption, and no vehicle breakdown or traffic.

In practice, delivery operations are subject to considerable stochasticity. Traffic congestion causes unpredictable delays, vehicles occasionally break down, and the resulting disruptions can cascade, leading to the failure of tightly fitted deterministically scheduled routes. A route plan that is optimal under deterministic assumptions may perform poorly when subject to real-world stochasticity[12]. This gap between model assumptions and operational reality motivates the need for a routing framework that explicitly account for stochastic disruptions.

1.1 Research Question

This thesis addresses the following problem:

> *To what extent does incorporating stochastic recourse through a simulation-based NSGA-II framework improve performance at the knee of the Pareto front in terms of energy efficiency, time-window violation, and resilience, compared to a deterministic EVRPTW baseline?*

1.2 Contributions

This thesis makes the following contributions:

1. MOS-EVRPTW formulation: We formulate the MOS-EVRPTW as a two-stage stochastic program with three objectives: energy consumption, time-window violation as a proxy for customer dissatisfaction, and recourse cost as a measure of operational resilience.
2. Simulation-based NSGA-II: We implement a method that combines NSGA-II with Monte Carlo simulation for fitness evaluation under uncertainty.
3. Strategic insights: We benchmark the stochastic framework against a deterministic baseline on the established Goeke EVRPTW instances and quantify the value of the stochastic solution, which measures the benefits of using a stochastic model instead of a deterministic one.

1.3 Thesis overview

The thesis is organized as follows. Section 2 provides the theoretical background on VRP, stochastics, multi-objective algorithms, and modeling. Section 3 introduces related work and identifies the gap that this thesis addresses. Section 4 presents the MOS-EVRPTW formulation and the simulation based NSGA-II solution method. Section 5 describes the experimental setup method, reports the results, and discusses the implications. Afterwards, limitations and further research are discussed in Sections 6, 7 followed by the conclusion in Section 8.

This bachelor thesis was conducted at the Leiden Institute of Advanced Computer Science under the supervision of Dr. Y. Fan.

2 Background

We introduce several concepts required to build upon the bigger MOS-EVRPTW framework. First, the classical vehicle routing problem. This thesis extends it to electric vehicles and then further introduces stochastic and multi-objective optimization techniques that form the methodological section of this thesis.

2.1 The Vehicle Routing Problem

The VRP was introduced by Dantzig and Ramser [5]. In its most basic form, a fleet of vehicles based at a depot must serve a set number of customers. These customers are dispersed throughout the map, and each customer has their own specific demand. The objective is to find a set of routes per vehicle that collectively visit every customer exactly once while minimizing the total travel distance or cost. Each route begins and ends at the depot, and the total demand served on any single route must not exceed the vehicle's capacity.

Formally, let $G = (V, E)$ be a complete directed graph where $V = \{0, 1, \ldots, n\}$ is the set of nodes, where node 0 is the depot and nodes 1 to n are customers, and $E = \{(i, j) \mid i, j \in V, i \neq j\}$ is the set of arcs. Each customer has a demand D_i and each arc (i, j) has a travel cost c_{ij}. A homogeneous fleet of K vehicles, each with a capacity of C, is available. The objective thus is to minimize the total routing cost subject to capacity and visit constraints.

The VRP is an NP-hard problem, meaning that there is no polynomial-time algorithm to solve it to optimality for large instances[28]. For practical-sized problems, a metaheuristic approach is therefore the dominant strategy.

2.2 The Vehicle Routing Problem with Time Windows

The VRPTW adds a scheduling dimension where each customer i has a time window $[l_i, u_i]$ during which service must begin. Solomon [26] proposed heuristics and benchmark instances for the VRPTW that have become widely used. A vehicle arriving before l_i must wait, and a vehicle arriving after u_i violates the constraint of time window adherence. The service at customer i takes s_i units of time. The depot also has a horizon time window $[l_0, u_0]$ which constraints the vehicles to arrive back at the depot by u_0.

The VRPTW significantly increases the difficulty of the VRP because routing and scheduling are tightly coupled factors. Changing the visit order on a route affects the arrival times at all subsequent customers.

2.3 The Electric Vehicle Routing Problem with Time Windows

The EVRPTW extends the VRPTW to a fleet of electric vehicles[25]. In addition to customer nodes (V_c) and the depot ($\{0\}$), the graph contains a set (V_f) of charging stations. Assuming parameter Q represents battery capacity rather than fuel tank capacity, vehicles have limited battery capacity Q and consume energy as they move along the arcs. If the battery state of charge drops to zero, the vehicle becomes stranded. Vehicles may detour to a charging station to replenish their battery before continuing.

Schneider [25] introduced the EVRPTW and solved it with a heuristic distinct from the approach of this thesis. Similarly to this thesis, their formulation assumed full recharging at every station visit, where each vehicle that visits a charging station would always charge its battery to full capacity. Keskin [18] later relaxed this assumption by introducing partial recharge strategies, allowing vehicles to charge only the amount needed to reach the next stop. The findings concluded that the increase in flexibility significantly reduced total route durations and improved solution quality.

2.4 Energy Consumption Model

Accurate energy modeling is important for electric vehicle routing because battery depletion depends on many factors such as: distance, vehicle speed, payload, friction, etc. Goeke and Schneider [14] proposed an energy consumption model that accounts for aerodynamic drag, rolling resistance, along with other factors that we will adopt in a simplified manner in this thesis. We assume a flat terrain and a vehicle that uses a single aggregate powertrain efficiency parameter.

The total force F_{total} opposing the vehicles motion at speed v and mass M on flat terrain is:

$$F_{total} = \underbrace{\frac{1}{2} C_d \rho A v^2}_{\text{Aerodynamic Drag}} + \underbrace{C_r M g}_{\text{Rolling Resistance}} \tag{1}$$

Where C_d is the drag coefficient, ρ is the air density, A is the frontal area of the vehicle, C_r is the coefficient of rolling resistance, and $g = 9.81$ is the gravitational acceleration. The vehicle mass is the sum of the curb weight M_{curb} and the current payload is $M_{load} \times \phi$, where M_{load} is the maximum cargo weight and $\phi \in [0, 1]$ is the load ratio representing how full the vehicle is.

$$M = M_{curb} + M_{load}\phi \tag{2}$$

The mechanical energy required to traverse a distance d is $E_{mech} = F_{total} \cdot d$. Accounting for a simple aggregate powertrain efficiency η and auxiliary power consumption P_{aux} (heating, air conditioning, and electronics), the electrical energy consumption on an arc (i, j) with distance d_{ij} and travel time t_{ij} is:

$$E_{ij} = \frac{F_{total} \cdot d_{ij}}{\eta} + P_{aux} \cdot t_{ij} \tag{3}$$

2.5 Two-Stage Stochastic Programming

Stochastic programming is a foundational concept used for optimization under uncertainty [2]. Decisions are made in two phases.

Within the first stage, decisions x are made before uncertainty is revealed. These decisions can be conceptualized as commitments that cannot be changed once the random events are realized. In the routing context, the first-stage decision is the set of vehicle routes.

Afterward, comes the second stage. After the uncertainty ξ is realized, recourse decisions $y(x, \xi)$ are made to compensate for any constraint violations or disruptions caused by the realized scenario.

The objective is to minimize the sum of the first-stage cost and the expected second-stage recourse cost.

$$\min F(x) = [E[f_{energy}(x, \xi)], E[f_{time}(x, \xi)], E[f_{resilience}(x, \xi)]] \tag{4}$$

This intends to produce solutions that are resilient to disruptions because they anticipate the average cost of recovery rather than optimizing solely for a single deterministic scenario.

2.6 Multi-Objective Optimization and Pareto Efficiency

A multi-objective optimization problem takes the form:

$$\min_{x \in X} \big(f_1(x),\ f_2(x),\ \ldots,\ f_k(x)\big) \tag{5}$$

A solution x^* is said to dominate another solution x' if $f_i(x^*) \leq f_i(x')$ for all objectives i and $f_j(x^*) < f_j(x')$ for at least one objective j. A solution that is not dominated by any other feasible solution is called Pareto-optimal. The set of all Pareto-optimal solutions forms the Pareto front, which is the set of best possible trade-offs among the objectives [31].

Unlike single-objective optimization, multi-objective optimization does not yield a single best solution. Rather, it produces a set of non-dominated solutions.

2.7 NSGA-II

The Non-dominated Sorting Genetic Algorithm II (NSGA-II), proposed by Deb et al. [8], is one of the most popular multi-objective evolutionary algorithms. It maintains a population of candidate solutions and evolves them over generations using selection, crossover, and mutation operators. It relies on two key mechanisms. The first is fast non-dominated sorting, which partitions the population into successive fronts $\mathcal{F}_1, \mathcal{F}_2, \ldots$ where $\mathcal{F}_1$ contains all non-dominated solutions, $\mathcal{F}_2$ contains all solutions dominated only by members of $\mathcal{F}_1$, and so on; solutions in earlier fronts are preferred during selection. The second is crowding distance, which measures how isolated solution is in objective space and ranks the members within each front accordingly. Solutions in less crowded regions are preferred, since these solutions promote diversity across the Pareto front.

The selection operator uses binary tournament selection based on rank and crowding distance. This allows the algorithm to converge toward the Pareto front while maintaining a diverse set of solutions.

2.8 Monte Carlo Simulation

Monte Carlo simulation is a computational technique that samples randomly in order to estimate the statistical properties of a system[21]. Given a stochastic model with random inputs ξ, Monte Carlo simulation generates N independent samples $\xi^1, \ldots, \xi^N$ and estimates the expected value of a function f as:

$$E[f(\xi)] \approx \frac{1}{N} \sum_{i=1}^{N} f(\xi^i) \tag{6}$$

The estimate converges to the true expectation as $N \to \infty$ by the law of large numbers. In the context of this thesis, Monte Carlo simulation is used to evaluate the expected performance of a route plan across many randomly generated disruption scenarios. This approach accommodates arbitrary disruption distributions and complex recourse logic without having to find a complicated closed-form analytical expression.

3 Related Work

The electric vehicle routing problem has a wide selection of past literature such as: deterministic EVRPTW models, stochastic vehicle routing, and multi-objective evolutionary approaches to routing problems. Within these works exists a gap that this thesis addresses.

3.1 Deterministic EVRPTW

The EVRPTW was formally introduced by Schneider et al.[25] who extended Solomon's classical VRPTW benchmark instances with charging stations and battery constraints. Their model assumed that vehicles must fully recharge at every station visit, and they solved the problem using a single objective metaheuristic, distinct from the NSGA-II approach used within this thesis. Keskin et al.[18] extended upon this work by relaxing the full-recharge assumptions, which allowed vehicles to partially recharge at charging stations. Their experiments demonstrated that integrating partial recharge reduces total travel distance and enables the use of fewer vehicles; particularly in environments with more restrictive time windows.

Goeke and Schneider[14] contributed a physics based energy consumption model that accounts for vehicle mass, speed, aerodynamic drag, rolling resistance, and road gradient. In contrast to plain distance-based models, their formulation captures how estimating the vehicle load can improve the quality of generated solutions. Specifically, they studied how a vehicle with a certain amount of load consumes more energy and how that, in turn, affects the driving range of the vehicle. Simply put, a fully loaded vehicle consumes more energy than an empty one. The work of Goeke and Schneider extends onto this thesis, where we utilize their EVRPTW instance dataset[13].

3.2 Stochastic Vehicle Routing

Work extending the VRP with stochastic factors is extensive. Gendreau et al.[12] review the field and identify key modeling choices, including the treatment of demand uncertainty, travel time variability, and recourse strategies. Where two-stage stochastic formulations are prevalent, as implied by the paper: first stage routes are fixed in advance, and second-stage recourse actions handle the disruptions caused by realized uncertainty.

Mendoza et al.[20] developed a multi-space sampling heuristic for the VRP with stochastic demands, being able to combine multiple solution representation spaces to find high quality solutions under a two-stage stochastic programming formulation. Rajabi et al.[24] addressed stochastic time-dependent routing with a multi-objective formulation solved by NSGA-II, their study was able to show that evolutionary methods are effective at handling the combination of stochasticity in combination with multiple objectives. Within their experiments they defined crucial parameters such as crossover and mutation rate, alongside population size and number of iterations. These parameters carry over their influence to this thesis.

Although the electric vehicle routing problem is widely explored, there exists a combination of stochastic modeling within the problem domain. Few studies have combined battery capacity, feasibility, and time-window adherence. For instance, battery feasibility is directly affected by the traffic, since if a given vehicle is stuck in traffic for prolonged periods of time it will eventually start depleting its battery from factors such as the constant auxiliary power consumption of external factors(eg. heating or radio). As such, this can push the vehicle below the state of charge necessary to reach the next stop, meaning that the recourse cost of a disruption is not separable from the energy model.

In the present implementation, the energy model is split into two partially distinct segments influenced by Goeke & Schneider: battery state of charge is represented as a linear function of distance ($r * d_{ij}$) according to the simplified model by Schneider et al.[25], while the energy objective itself is subject to traffic and load-dependent auxiliary draw following a simplified implementation of the energy model by Goeke et al.[14]. Fully coupling a comprehensive energy model to both battery state of charge and the energy objective is left to further research (Section 7).

3.3 Multi-Objective Evolutionary Approaches

NSGA-II[8] has been widely applied to multi-objective vehicle routing. Srivastava et al.[27] proposed objective specific variation operators for the multi-objective VRPTW and demonstrated that tailoring genetic operators to individual objectives improves convergence. Their work showed that a customized NSGA-II with problem specific operators can outperform prior state-of-the-art approaches for generating diverse Pareto fronts on routing problems.

Juan et al.[17] introduced the concept of "simheuristics", which is metaheuristics that embed simulation within the optimization loop. Through evaluating candidate solutions using Monte Carlo simulation, simheuristics can handle stochastic objectives without requiring explicit analytical expressions for the expected cost. This approach was naturally integrated within this thesis.

3.4 Research Gap

Existing literature covers EVRPTW, stochastic disruptions, and optimization of multiple objectives as largely separate dimensions. Stochasticity is incorporated into routing and it typically modifies existing objectives, creating factors such as uncertain travel times or uncertain demand, however operational resilience itself is rarely approached by existing literature. Yet, the impact of a disruption event is not purely captured by lateness alone. Although lateness is a big factor, beyond that factor there exists the rescue cost and cascading lateness that propagates along the disrupted route.

To the best of current knowledge, no prior work treats resilience to stochastic disruptions as a main objective alongside energy and time-window adherence in an EVRPTW context. This thesis aims to close this gap by formulating the MOS-EVRPTW which is a two-stage stochastic program with three competing expected-value objectives: energy, time-window adherence, and resilience. The objectives are optimized by using a simulation based NSGA-II approach. The deterministic baseline is used as a comparison to the MOS-EVRPTW under the same disruption distribution.

4 Approach

The first step to the formulation of the MOS-EVRPTW is to formalize the problem as a two-stage stochastic program with three objectives, then describe the recourse model that captures stochastic disruptions, and finally walk through each component of the metaheuristic: solution representation, fitness evaluation, charging repair, genetic operators, and the deterministic baseline used for comparison.

4.1 Problem Formulation

Let $G = (V, E)$ be a complete directed graph. The node set $V = \{0\} \cup V_c \cup V_f$, where every node in V is exactly one of $\{0\}$(depot), V_c(customer), or V_f(charger). Every customer $i \in V_c$, has a demand D_i, a service time s_i, and a time window $[l_i, u_i]$ during which service must begin. The depot has its own time window $[l_0, u_0]$. Every arc $(i, j) \in E$ has a Euclidean distance d_{ij} and a nominal travel time $t_{ij} = d_{ij}/v$, where v is the constant average velocity from the Goeke instance[13]. The route geometry is always assumed to be flat, as such the vehicle will not experience any of the effects that a route geometry with hills and valleys would impose on the vehicle.

Two distinct energy quantities appear in the model and should not be confused. Battery feasibility (Algorithm 1) uses Schneider et al.[25] linear consumption $r * d_{ij}$ in battery units, whereas the energy objective f_{energy} uses the physics model of Equation (14) in kWh. The energy objective therefore influences selection but not charger placement, this keeps charging logic distance-driven and identical to the baseline (Section 4.8).

A homogeneous fleet of electric vehicles are dispatched from the depot. Each vehicle has a load capacity C, battery capacity Q, battery consumption rate r per unit distance, and inverse recharge rate g_r per unit of energy refilled. Following Schneider et al.[25], chargers may be visited multiple times. Following Goeke [13] we treat the fuel tank capacity as the battery capacity. As discussed in Section 5.2, we adopt the simpler full-recharge model rather than the partial-recharge strategy of Keskin et al.[18].

A candidate solution is a set of routes $\mathcal{R} = \{R_1, R_2, \ldots, R_K\}$ where each route $R_k = (0, v_{k,1}, v_{k,2}, \ldots, 0)$ and $v_{k,i} \in V_f \cup V_c$. Each route begins and ends at the depot; it visits a sequence of customers and charger nodes while maintaining feasibility by respecting vehicle capacity, time windows, and battery constraints. The fleet size K is not fixed; however, it is implicitly bounded by the trade-off between energy and time objectives 5.2.

A simple lower bound on the fleet size follows from capacity feasibility:

$$K_{lower} = \left\lceil \frac{\sum_{i \in V_c} D_i}{C} \right\rceil \tag{7}$$

No explicit upper bound is imposed. Adding vehicles relieves time pressure through parallelized vehicle delivery, but it increases energy cost by adding more depot to customer legs. NSGA-II implicitly balances the upper bound of the fleet.

We adopt a stochastic programming approach [2] where the first-stage decision is the route set $\mathcal{R}$, which is committed before uncertainty is realized. The second-stage uncertainty ξ is the realized uncertainty of traffic congestion and vehicle breakdowns along the route, and the recourse function $y(\mathcal{R}, \xi)$, which accumulates the cost of disruptions encountered during execution. Our objective vector consists of three expectations:

$$F(\mathcal{R}) = [E[f_{energy}(\mathcal{R}, \xi)], E[f_{time}(\mathcal{R}, \xi)], E[f_{resilience}(\mathcal{R}, \xi)]] \tag{8}$$

where f_{energy} is the total kWh of energy consumed, f_{time} is the cumulative time-window violation penalty, and $f_{resilience}$ is the cumulative recourse cost from breakdowns, battery depletion, and congestion disruption. Each scalar component is broken down into two pieces: a deterministic planned cost $f_{plan}(\mathcal{R})$ that we can compute from the route alone, and a recourse cost $y(\mathcal{R}, \xi)$ that only materializes once the uncertainty is realized:

$$f(\mathcal{R}, \xi) = f_{plan}(\mathcal{R}) + y(\mathcal{R}, \xi) \tag{9}$$

The goal is to find the Pareto front of route sets that are non-dominated under (8).

4.2 Recourse Model

The second-stage scenario ξ is made up of independent random processes realized arc by arc as the vehicle traverses its specified route. Within this thesis the second-stage is not optimized. The first-stage decision is the route set $\mathcal{R}$. Once ξ is realized, no recourse decisions are chosen; instead a fixed simulation policy maps $(\mathcal{R}, \xi)$ to realized arrival times, breakdowns, and battery depletion events, and the recourse cost $y(\mathcal{R}, \xi)$ is read off from them. As such, this is a two-stage program with fixed recourse that is evaluated by Monte Carlo. Furthermore, there exist hard feasibility constraints that ensure each customer is visited once, ensuring capacity constraints are respected, and that vehicles start/end at the depot. These constraints are enforced by the operators (Section 4.6) and repair (Algorithm 1). Time-window and battery feasibility are soft constraints, and allow for the exploration of time-window or battery infeasible routes, which are then penalized in the resilience and time objectives. The method used within this thesis is therefore a simheuristic[17] that approximates the expected-value objective.

4.2.1 Traffic Congestion

Traffic congestion is assumed to be inescapable, a vehicle that is stuck in traffic cannot reroute or escape traversing the arc. Each arc (i, j) traversed during execution receives a traffic multiplier, where the realized traffic congestion is specified as:

```
if self.traffic_sigma > 0:
    traffic_multiplier = max(1.0, np.random.lognormal(mean=0.0,sigma=
       self.traffic_sigma))
else:
    traffic_multiplier = 1.0 # deterministic case
```

Figure 1: Traffic congestion modeled by sampling from a log normal distribution using Numpy[16], where $\sigma = 0$ implies a deterministic environment with fixed traffic.

let the traffic multiplier of an arc (i, j) be m_{ij}, then the realized travel time is $\tilde{t}_{ij} = m_{ij} * t_{ij}$. The lognormal floor at 1 models traffic congestion as a one sided phenomenon where traffic can only slow down a vehicle, but it may never speed it up beyond its average velocity v, where $v = 1$ according to Goeke [13]. The effective velocity during the arc is reduced to $\tilde{v}_{ij} = v/m_{ij}$.

4.2.2 Vehicle Breakdown

Independent of traffic congestion, each arc traversal generates a Bernoulli breakdown event with probability p_{break}. A breakdown adds a fixed cost c_{rescue} to the resilience objective and a fixed delay Δ_{break} to the route time, where the vehicles battery state is unchanged. The simulation applies this event at the end of the arc when the vehicle has traversed through (i, j) from i to j, which is a simplification that implies a breakdown cannot happen partway through, it only happens once the vehicle has traversed to j. As such there is no possibility to breakdown mid-arc which simplifies the problem. It should also be noted that the breakdown probability applies per arc, rather than per unit distance, which implicitly guides the algorithm toward routes with fewer but longer arcs covering the same total distance.

4.2.3 Battery Depletion

Battery depletion is an indirect consequence of the plan rather than a sampled disruption. If the amount of battery consumption on a route exceeds the available charge, the route is considered to have experienced an emergency rescue. In this scenario there is a penalty applied to the battery deficit $\alpha_{battery}$ per unit of battery depleted, plus a flat rescue cost c_{rescue} and delay Δ_{break}. The vehicle is then assumed to be rescued on the spot and restored to full charge, from which it continues along its planned route to the next node. This soft approach allows the genetic algorithm to explore regions that are battery infeasible, by paying for them in the objective, rather than rejecting them immediately. We reuse Δ_{break} and c_{rescue} across both vehicle breakdown and battery depletion events for simplicity. In practice dispatching a vehicle that can charge a battery depleted vehicle and dispatching a towing vehicle for a vehicle that has broken down are operationally separate concepts, as such parameterizing these concepts separately can be a natural extension of this work.

4.2.4 Recourse Cost

Putting all of these disruption factors together, the per-arc contributions to the three objectives are:

$$f_{\text{energy}}(\mathcal{R}, \xi) = \sum_{\mathcal{R}_k \in \mathcal{R}} \sum_{(i,j) \in \mathcal{R}_k} E_{ij}\big(d_{ij},\, \tilde{v}_{ij},\, \phi_{ij}\big), \tag{10}$$

$$f_{\text{time}}(\mathcal{R}, \xi) = \alpha_{\text{time}} \sum_{\mathcal{R}_k \in \mathcal{R}} \sum_{i \in \mathcal{R}_k} \big[\, T_i - u_i \,\big]^+, \tag{11}$$

$$f_{\text{resilience}}(\mathcal{R}, \xi) = \sum_{\mathcal{R}_k \in \mathcal{R}} \sum_{(i,j) \in \mathcal{R}_k} \Big(c_{\text{rescue}}\, 1_{B_{ij}} + \alpha_{\text{reroute}}\, \frac{\delta_{ij}}{t_{ij}}\, d_{ij} \Big) \tag{12}$$

$$+ \sum_{\text{depletions}} \Big(c_{\text{rescue}} + \alpha_{\text{battery}}\, |\text{batterydeficit}| \Big) \tag{13}$$

where ϕ is the load ratio (current payload divided by capacity C), T_i is the realized arrival time at i, δ_{ij} is the realized delay, and $1_{B_{ij}}$ is the Bernoulli indicator of breakdown on the arc (i, j). Because T_i accumulates arc by arc, any single disruption propagates into the time penalty of every downstream node on the same route, in addition to its own resilience term. This is the cascading lateness mechanism that the resilience objective is designed to capture. The reroute term is a penalty for congestion related disruption. Long arcs suffer a large proportional delay and contribute more than short arcs with the same relative slowdown. Furthermore, E_{ij} refers to the energy consumed traversing an arc, not to be confused with E in $G = (V, E)$.

4.2.5 Energy Model

The per arc energy E_{ij} in (10) follows the simplified Goeke model introduced in Section 2.4. We transform the Goeke unit distances and unit velocities with a distance scale factor $s_d = 1000m/unit$ and velocity scale factor $s_v = 11.11m/s$ to map the units to SI:

$$E_{ij}(d_{ij}, v_{ij}, \phi_{ij}) = \frac{F_{total} \cdot d_{ij} \cdot s_d}{\eta \cdot 3.6 \times 10^6} + P_{aux} \frac{d_{ij} \cdot s_d}{v_{ij} \cdot s_v \cdot 3600} \tag{14}$$

where we extend upon Goeke's model by incorporating realized velocity and current load:

$$E_{ij}(d_{ij}, \tilde{v}_{ij}, \phi_{ij}) = \frac{F_{total} \cdot d_{ij} \cdot s_d}{\eta \cdot 3.6 \times 10^6} + P_{aux} \frac{d_{ij} \cdot s_d}{\tilde{v_{ij}} \cdot s_v \cdot 3600} \tag{15}$$

where F_{total} is composed of both aerodynamic drag and rolling resistance at the realized velocity, as seen in (1)

$$F_{total} = \underbrace{\frac{1}{2}\, C_d\, \rho\, A\, \tilde{v}^2}_{\text{Aerodynamic Drag}} + \underbrace{C_r\, (M_{curb} + M_{load}\phi)\, g}_{\text{Rolling Resistance}} \tag{16}$$

The first term in (14) captures mechanical work against drag and rolling resistance, scaled by an efficiency factor η. The second term captures auxiliary draw from components such as heating, air conditioning or electronics. This power draw accumulates over the realized travel time.

4.2.6 Time-window Adherence

The route time T increments itself arc by arc during simulation. Once arrived at the customer i, if $T < l_i$ the vehicle waits without penalty until $T = l_i$; however if $T > u_i$ the lateness $T - u_i$ is multiplied by α_{time} and added to f_{time}. Service time s_i is then added to the route time. At charger nodes the vehicle recharges itself to full, with the charge time being $(Q - \text{currentbattery}) * g_r$ which is then added to the route time.

4.2.7 Expected Value Estimation

We approximate the expectations in (8) via Monte Carlo simulation with N_{MC} independent scenarios. For each objective $f \in \{\text{energy}, \text{time}, \text{resilience}\}$:

$$E[f(\mathcal{R}, \xi)] \approx \widehat{F}_f(\mathcal{R}) = \frac{1}{N_{MC}} \sum_{n=1}^{N_{MC}} f(\mathcal{R}, \xi^{(n)}) \tag{17}$$

By the law of large numbers, $\widehat{F}(\mathcal{R}) \to F(\mathcal{R})$ as $N_{MC} \to \infty$[2], where $\widehat{F}$ represents the collection of all three objectives into a vector. The convergence rate as a function of N_{MC} is studied in Section 5.2.2.

4.3 Solution Representation

A candidate solution is represented as a list of routes, where each route is a list of nodes (depot,customer,charger). These nodes begin and end at the depot as mentioned in Section 4.1.

A route may mix customer and charger nodes, and distinct routes may share chargers. Within this thesis we do not label which vehicle drives which route, as the vehicle fleet is homogeneous. Each individual carries a DEAP fitness object with weights $(-1, -1, -1)$, implying that all three objectives (8) are minimized. This thesis uses the python DEAP genetic algorithm framework[7].

4.4 Initial Population

The initial population of N_{pop} individuals is generated by a greedy randomized heuristic followed by a charging repair pass. The heuristic builds routes one route at a time. At each step the next customer is selected from a random sample of unassigned candidates, ranked by a noise perturbed score:

```
if vehicledemand <= current_capacity and (current_time + travel_time) <
   timewindow:
    noisy_distance = distance*random.uniform(0.5,3.0) # Added noise
       allows the population to be diverse
    if noisy_distance < bestcost:
        bestcost = noisy_distance
        bestcustomer = c
```

Figure 2: After passing the feasibility check, rank the candidate based on a highly noisy distance metric in order to encourage population diversity.

candidates are sampled at 50% of the customer set size, with a floor of 2, so the choice stays random until only a single unassigned customer remains. Furthermore, a candidate is feasible only if inserting it would not violate the vehicle capacity nor cause the arrival to exceed the customers time window. When no feasible candidates remain, the route returns to the depot and a new route is opened. The random sampling and the noise-perturbed distance metric ensure that the resulting population is diverse. The sampling uniformly explores the customer set, while the noise perturbed distance score introduces randomness within each sample.

After the routing has been constructed, each route is passed through the charging repair procedure which ensures that battery feasibility is enforced before the individual is evaluated.

4.5 Charging Repair

The charging repair procedure walks a route arc by arc, tracking battery state, while evaluating where and when the vehicle would need to charge itself. The process of finding a recharging station to recharging the vehicle is executed whenever the battery consumption from point A to point B drives the state of charge below zero. This process converts the initial population of customer only routes into battery feasible routes, and after every genetic operator (crossover, mutation) it re-establishes feasibility because those operators ignore chargers entirely.

The charging repair procedure follows:

Algorithm 1 Charging repair

Require: Route $R = (v_0, v_1, \ldots, v_k)$ with $v_0 = v_k$=depot;fuel consumption rate r; battery capacity Q; set of chargers V_f; distance function $d(\cdot, \cdot)$

committed $\leftarrow [v_0]$; battery_trace $\leftarrow [Q]$; pending $\leftarrow [v_1, \ldots, v_k]$ ▷ battery_trace holds the remaining battery after arriving at commited[i]

inserts $\leftarrow 0$; max_inserts $\leftarrow 4\,|V_f| + |R|$ ▷ Maximum bound on insert designated to stop infinite looping

if $V_f = \emptyset$ **then return** R

end if

while pending $\neq \emptyset$ **do**

$a \leftarrow$ committed.last; $b \leftarrow$ pending.first; $\beta \leftarrow$ battery_trace.last

if $a = b$ **then** ▷ Remove duplicate charger

pop(pending.first); **continue**

end if

if $\beta \geq r \cdot d(a, b)$ **then** ▷ Traverse arc if it is battery feasible

commit b; update battery and append it to battery_trace (and full recharge if $b \in V_f$); pop(pending.first)

else

inserts $\leftarrow$ inserts $+ 1$

if inserts $>$ max_inserts **then return** R as-is

end if ▷ Upper bound of inserts exceeded, return the route and the simulator will score the battery depletion event

$c^\star \leftarrow$ best charger between a and b that minimizes the distance $d(a, c) + d(c, b)$, is reachable on β, and from which b is reachable on Q

if $c^\star$ exists **then** commit $c^\star$ with battery Q and append it to battery_trace; **continue**

end if

backtrack: pop one committed stop and battery_trace at a time until only the depot is left and try inserting a charger $c^\star$

between the new tail and the popped stop; if $c^\star$ exists then put the popped stops

back at the front of pending in their original order **continue**

if backtracking exhausts **then return** R as-is

end if

end if

end while

return committed

The maximum insert bound guarantees termination on graphs where chargers are configured in a way where no insertion sequence would resolve the infeasibility. Within those scenarios the original route is returned and the Monte Carlo simulator scores the depletion event explicitly through penalizing the resilience objective.

4.6 Genetic Operators

4.6.1 Best Cost Route Crossover

We adopt the route-based crossover work of Ombuki et al.[23]. Given two parent individuals A and B, the operator deep copies each parent, and then selects a singular route at random from each parent:

$$\pi_A \in A, \qquad \pi_B \in B \tag{18}$$

From these routes, the original customer visitation order is extracted, while discarding chargers and the depot:

$$C_A = \{v | v \in \pi_A \wedge v \in V_c\} \qquad C_B = \{v | v \in \pi_B \wedge v \in V_c\} \tag{19}$$

Afterwards, the crossover mechanism removes the selected customers from each of the parents correspondingly, such that the route of parent A is removed from parent B and route of parent B is removed from parent A. Within the same pass all chargers from the every route of the parent are dropped as they are no longer guaranteed to be useful, since the original chargers were placed by Algorithm 1 to support specific customer ordering that no longer holds:

$$A' = \{R' | R' = filter(R, C_B \cup V_f) \wedge R \in A\} \qquad B' = \{R' | R' = filter(R, C_A \cup V_f) \wedge R \in B\} \tag{20}$$

where $filter(R, C)$ returns R with every node belonging to C removed, preserving the order of the remaining nodes. Routes that are reduced to only depots are then dropped from A' and B'.

The now removed customers are reinserted one at a time via the best insert method, which evaluates every position in every existing route of the modified parent and selects the insertion that minimizes the marginal distance: $d(a, c) + d(c, b) - d(a, b)$. Throughout the process feasibility constraints are checked when evaluating insertion into an existing route position, so that solutions which would violate vehicle capacity or arrival-time feasibility constraints are skipped. If no feasible insertion exists, a fresh route $(0, c, 0)$ with customer c is opened; this fallback does not itself re-check feasibility, so the opened route may still violate customer's time-window if it cannot be met directly from the depot. In that case the violation is not rejected, but rather captured later as penalty on f_{time} during fitness evaluation. Empty offspring are reduced to $\{(0, 0)\}$ to preserve representation, and every resulting route is passed through Algorithm 1 to re-establish battery feasibility.

Best Cost Route Crossover is applied with probability $p_{cx} = 0.65$ and each individual is mutated with probability $p_{mut} = 0.4$ following Rajabi et al.[24].

4.6.2 Mutation

Once mutation occurs, one of the three operators is chosen with equal probability:

Intra-route swap. Select a random route and if it contains at least two customers then sample all of the customer positions at random for two customers and swap the customers at those positions.

Inter-route swap. Select a source route with more than one customer, and pick a random customer within it, remove it, and reinsert it at a random position in a different randomly chosen destination route. If the source route is left with only the depots, it is dropped.

2-opt improvement. For each route which has at least 3 customers, enumerate through all the customers (i, j) with $j > i + 1$, shuffle them, and apply the first improving reversal where the segment $R[i, \ldots, j]$ is reversed if the sum of the new edge distances is smaller than the old[4].

According to Theorem 3 of Lin[19], 2-optimality is equivalent to inversion freedom, meaning that no improvement can be achieved by reversing any sub-sequence of the route. By using the first-improvement rather than best improvement it is possible to keep the computational cost per evaluation low, allowing the experiments to be conducted on time. According to Lin[19], using first-improvement rather than best improvement keeps the computational cost per-evaluation low; which Lin argues is generally preferable since best improvement tends to increase computation time disproportionately.

After mutation, the routes are passed through Algorithm 1 to restore battery feasibility.

Every mechanism (greedy initialization, BCRC, and 2-opt) ranks moves by Euclidean distance subject to feasibility constraints. These mechanisms do not evaluate energy, time penalty, or resilience. Likewise charger placement in Algorithm 1 is objective-blind in both the MOS-EVRPTW and the baseline. The three objectives enter only through NSGA-II selection over the Monte-Carlo fitness vector. The operators thus propose distance-feasible structures and selection retains the ones that score well under stochastic evaluation. Because energy and resilience correlate with distance, as seen by Section 5.3.3, while time-window penalty does not, this design predicts strong MOS gains on energy and resilience and weak performance on time penalty, as observed in Section 5.3.

4.7 NSGA-II

We use the standard NSGA-II algorithm of Deb et al.[8], implemented via the DEAP framework[7]. Parents are first chosen by binary tournament with dominance and crowding distance as the decision metrics. After crossover and mutation, survivors for the next generation are chosen by sorting the combined parent and offspring pool into Pareto fronts and creating the next generation from the best fronts first, which allows the best Pareto-front solutions to never be lost while propagating across generations.

Algorithm 2 NSGA-II main loop for MOS-EVRPTW

Require: population size N_{pop} (multiple of 4, required by selTournamentDCD[7]); generations N_{gen}; crossover probability $p_{cx} = 0.65$; mutation probability $p_{mut} = 0.4$[24]; MC iterations N_{MC}

$P_0 \leftarrow$ initial population (Section 4.4) ▷ population initialized with greedy random heuristic

evaluate P_0 via $\widehat{F}$ (17) with N_{MC} scenarios

$P_0 \leftarrow \texttt{selNSGA2}(P_0, N_{pop})$ ▷ assign Pareto rank and crowding distance

for $\tau = 1, 2, \ldots, N_{gen}$ **do**

  $\mathit{offspring} \leftarrow \texttt{selTournamentDCD}(P_{\tau-1}, N_{pop})$

  $\mathit{offspring} \leftarrow$ deep-clone each individual in $\mathit{offspring}$

  for $k = 1, 2, \ldots, N_{pop}/2$ **do**

    $(x, y) \leftarrow (\mathit{offspring}_{2k-1}, \mathit{offspring}_{2k})$

    if rand $< p_{cx}$ **then**

      $(x, y) \leftarrow \texttt{BCRC}(x, y)$

      repair charging on every route of x and y

      invalidate fitness of x and y

    end if

  end for

  for each $x \in \mathit{offspring}$ **do**

    if rand $< p_{mut}$ **then**

      pick mutation by choosing uniformly from (intra-route swap, inter-route swap, or 2-opt) and apply to x

      repair charging on every route of x

      invalidate fitness of x

    end if

  end for

  re-evaluate every $x \in \mathit{offspring}$ with invalid fitness via $\widehat{F}$

  $P_\tau \leftarrow \texttt{selNSGA2}(P_{\tau-1} \cup \mathit{offspring}, N_{pop})$

end for

return $P_{N_{gen}}$

The Pareto front $\mathcal{F}_1$ of the final population is then extracted from the algorithms output.

4.8 Deterministic Baseline

To determine the value of modeling stochastic disruptions, we will compare the MOS-EVRPTW against a deterministic EVRPTW model represented by the VRPy using the CVRPTW solver[22]. This baseline ignores both the stochastic disruptions and the battery.

As VRPy does not model recharging, we have to make the comparison fair in order to accurately represent a deterministic EVRPTW model. As such, we construct a derived graph G' that omits all charger nodes and sets the edge costs equal to distances. The solver is then run with `pricing_strategy="BestEdges1"` with the cspy backend disabled and an allocated time limit of 360s to converge to a high-quality solution, which through testing was determined to be enough in order for the solver to converge to an optimal path. The returned routes are translated back to

battery feasible deterministic EVRPTW standard through the Algorithm 1 which re-introduces charger stops wherever the battery would otherwise deplete. The resulting product is a single deterministic EVRPTW route plan $\mathcal{R}^{det}$ that respects capacity, time windows, and battery feasibility, and has been constructed without any knowledge of the stochastic disruptions.

Afterwards, we evaluate $\mathcal{R}^{det}$ on the same Monte Carlo scenarios used by the MOS-EVRPTW. In every experimental cell, the deterministic routes are re scored via $\widehat{F}(\mathcal{R}^{det})$ under the matching $p_{break}, \sigma, N_{MC}$ configuration. The end product is a single point in the objective space that can be plotted alongside the MOS-EVRPTW Pareto front.

From the MOS-EVRPTW side we summarize the Pareto front $\mathcal{F}_1$ by its knee point $\mathcal{R}^{knee}$, this represents the single trade-off solution at which improving any one objective would force a disproportionally large worsening of another, which in the absence of any preferences is the solution that a decision-maker is most likely to adopt[3]. It is important to note that under the assumption of no preferences, none of the Pareto optimal solutions can be said to be inferior in comparison to any other solution, as the point itself is superior in at least one criterion. However, we are using the knee point to move away from sacrificing a lot on one objective for a tiny gain on another, trying to avoid extremes and land near a solution which represents an ideal compromise.

Following Das[6], the knee is identified geometrically. Each objective is first min-max normalized across the front. For each of the three objectives, the front member that minimizes the corresponding normalized objective is taken as an extreme point, and the three extreme points span a hyperplane. The knee point is found by taking the 3 extreme points and stretching a flat plane across them. Then, we find which front solution sits furthest from the plane on the interior side, and that solution represents the knee point.

We borrow the term Value of the Stochastic Solution (VSS) loosely from Birge and Louveaux [2]. They define it as:

$$VSS = EEV - RP \tag{21}$$

where EEV is the expected cost of the expected-value solution, and RP is the recourse problem.

The Value of the Stochastic Solution on objective f is then defined as:

$$VSS_f = \widehat{F}_f(\mathcal{R}^{det}) - \widehat{F}_f(\mathcal{R}^{knee}), \qquad f \in \{\text{energy}, \text{time}, \text{resilience}\} \tag{22}$$

which is the gap between the baseline expected cost on objective f and knee-point expected cost on f of the MOS-EVRPTW Pareto front $\mathcal{F}_1$. A positive value indicates that on average the MOS-EVRPTW plan outperforms the baseline on objective f.

5 Experiments

5.1 Benchmark Instances

This thesis uses Goeke EVRPTW benchmark instances[13]. The benchmark is made up of 92 instances, where each instance follows one of the three customer distributions[25]:

- `c101_21:` instance follows a clustered distribution where customers are grouped in geographic clusters.
- `r101_21:` instance follows a random uniform distribution where customers are uniformly distributed.
- `rc101_21:` instance follows a mix of clustered/randomly distributed customers.

And each of the customer distributions are separated into either short scheduling horizons or long scheduling horizons[25]:

- `R1,C1,RC1`: Short scheduling horizon
- `R2,C2,RC2`: Long scheduling horizon

As visualized by (Figures 3,4,5) and the corresponding customer size sensitivity experiment (Figures 13,14,15), the greater the number of customers within the experiment, the more computationally costly it will be to find an optimal solution for each problem due to the NP-hard nature of the problem [28]. The random/uniform/mixed clustering can be clearly seen through these figures.

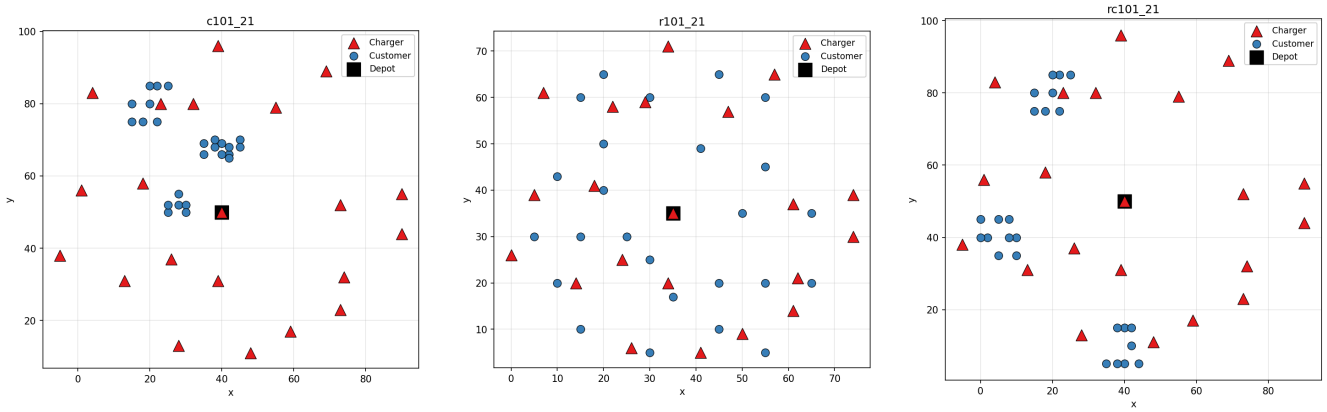


Figure 3: A diagram of a subset of 25 customers of each of the customer distribution instances where x and y represent distance units from the EVRPTW dataset[13].

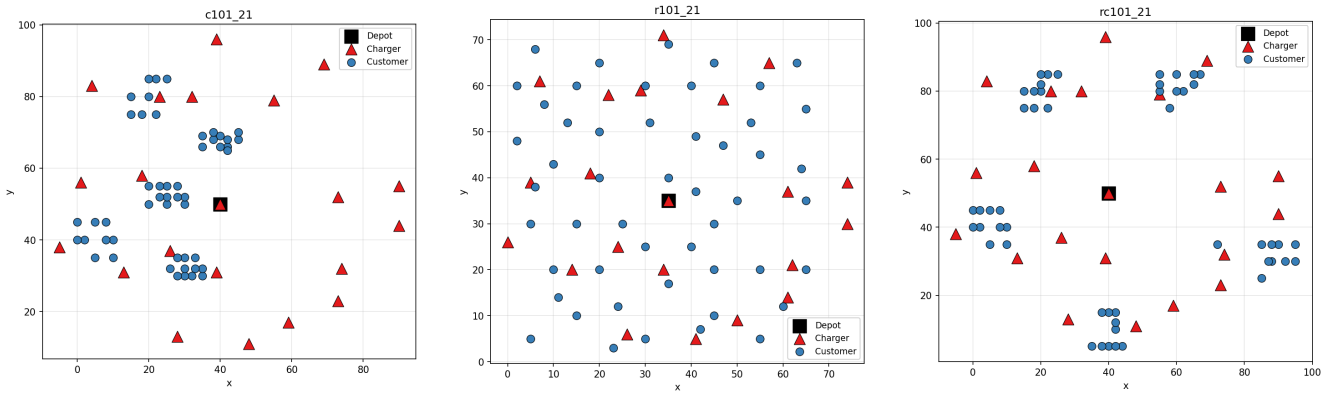


Figure 4: A diagram of a subset of 50 customers of each of the customer distribution instances where x and y represent distance units from the EVRPTW dataset[13].

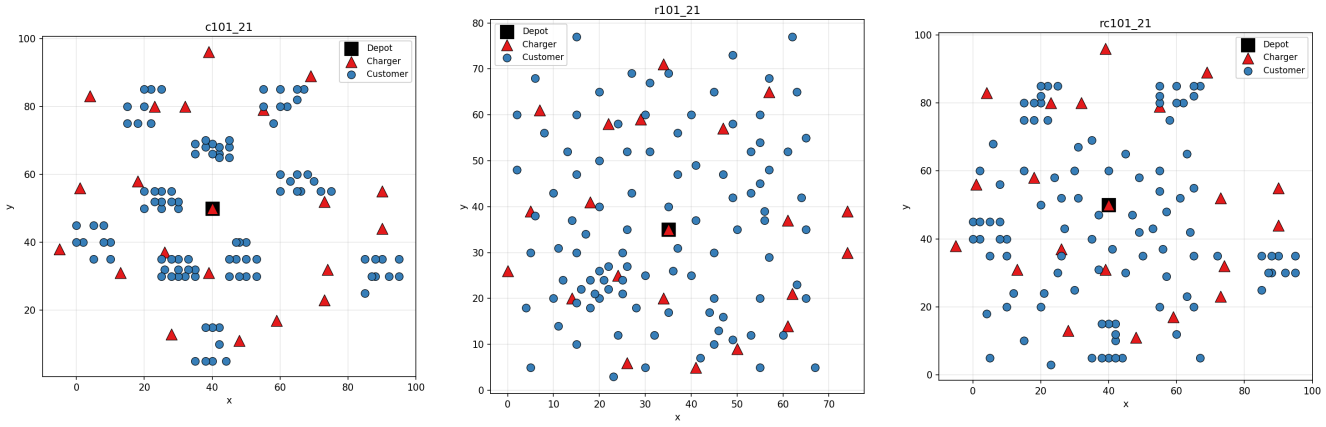


Figure 5: A diagram of all 100 customers of each of the customer distribution instances where x and y represent distance units from the EVRPTW dataset[13].

Each instance specifies the depot, customers with demands and time windows, charging station locations, and vehicle parameters(battery capacity (Q), load capacity(C), fuel consumption rate (r), inverse recharging rate (g_r), average velocity (v)). Each instance varies in these vehicle specific parameters, which creates a diverse set.

Table 1: Time-window width statistics for `c101_21` with horizon = 1236.0

Metric	Mean	Min	Max
Raw width (due − ready)	62.7	44.0	86.0
Slack (width − service)	-27.3	-46.0	-4.0

Table 2: Time-window width statistics for `r101_21` with horizon = 230.0

Metric	Mean	Min	Max
Raw width (due − ready)	10.0	10.0	10.0
Slack (width − service)	0.0	0.0	0.0

Table 3: Time-window width statistics for `rc101_21` with horizon = 240.0

Metric	Mean	Min	Max
Raw width (due − ready)	30.0	30.0	30.0
Slack (width − service)	20.0	20.0	20.0

Each customer instance has different properties and one of the most crucial distinctions across the customer instances is the strictness of time-windows. For instance, the horizon, which represents

the due date by which all vehicles must arrive at the depot, varies across instances. Instances can have narrow and wide time-windows or a mix of both, as indicated by Tables 1,2,3. Raw width represents how many time units each customers window spans, and the most important parameter is slack, which is the raw width minus the service time. The customer instance `rc101_21` has the most flexible time window, both the raw width and slack is high, meaning that the vehicle has a large span of time in which it can arrive and the vehicle can arrive late within the window and still complete service in time. The customer instances `c101_21` and `r101_21` are both comparatively strict, though for different reasons. `c101_21` has the largest raw width of the three instances, but a negative slack. As such, the service time itself exceeds the time-window, meaning that completing service at a customer pushes the route past that customers due date and propagates lateness to subsequent stops, independent of any stochastic disruption. `r101_21` instead has the narrowest raw width of all three instances with zero slack, leaving no margin for error. The characteristics of these time windows are subsequently visualized by Section 5.3, where MOS-EVRPTW struggles with time-window adhereance across `r101_21` and `c101_21`, while the wide and flexible time windows of `rc101_21` are the only setting in which MOS-EVRPTW is able to outperform the deterministic baseline on the time-penalty objective under a stochastic setting.

For the purposes of controlling the experimental procedure, this thesis conducts the experiments on a specific subset of instances. The experiment is limited in scope to these three instances: `c101_21`,`rc101_21`,`r101_21`. Therefore, we are able to gather representative information from each of the different customer distributions. All three instances belong to the short scheduling horizon group, which forces tight time windows and a larger fleet than their counterparts. A big chunk of the experiments is conducted using a subset of 25 customers, with the size-sensitivity experiment (Section 5.2) sweeping over $\{10, 25, 50, 100\}$ customers, where $n = 100$ is the maximum number of customers within the instance, capturing how the MOS-EVRPTW scales with larger customer sizes. As the MC fitness evaluation simulates the full route across many scenarios, which is computationally heavy, reducing the customer set was necessary due to limitations with the time allocated for the experimentation part.

5.2 Experimental Setup

Across each of the 3 different customer distribution instances, we run the experiments (5.2.1, 5.2.2, 5.2.3) with a fixed parent seed set of 42 in order to make the experiment reproducible. All of the experiments are independent of each other in terms of RNG state. Within the experiment (5.2.4) we utilize reseeding per cell, meaning that each experimental cell is independent of the previous cell (e.g. cell ($p_{break} = 0.05, \sigma = 0.00$) is independent of the previous experimental cell ($p_{break} = 0.00, \sigma = 0.00$)). This implies that if we have a cell k that conducts the stochasticity grid experiment on $p_{break} = 0.00, \sigma = 0.00$ then the next cell $k+1$ with parameters $p_{break} = 0.00, \sigma = 0.05$ would not start wherever cell k left state. The reseeding is derived through `numpy.random.SeedSequence`[16] where the parent seed of 42 is concatenated with the cell parameters, and the resulting seed is the child seed used to reseed the RNGs at the start of each cell. Because `SeedSequence` applies a hash to the cell parameter list, the child seeds are mutually uncorrelated even when their inputs differ by a single bit; hence, the cells of experiment 5.2.4 are statistically independent. Furthermore, since the mapping of the cell parameters to the RNG state is deterministic, it allows re-running the experiment to be reproducible.

Each of the experiments (5.2.1, 5.2.2, 5.2.3) are varying a single dimension while holding the remaining parameters constant at their default values. While experiment 5.2.4 averages the variance over 3 seeds within two environments of either high or low stochasticity.

Table 4 summarizes the algorithm parameters used in the experiment.

Table 4: Default experimental hyperparameters

Parameter	**Symbol**	**Value**
Population size	N_{pop}	500
Number of generations	N_{gen}	500
MC iterations	N_{MC}	500
Breakdown probability	p_{break}	0.05
Traffic noise	σ	0.15

Table 5: Experimental parameters

Parameter	**Symbol**	**Value**
Crossover probability	p_{cx}	0.65
Mutation probability	p_{mut}	0.4
Breakdown delay	Δ_{break}	30
Rescue cost	c_{rescue}	500
Rerouting penalty rate	$\alpha_{reroute}$	10
Time-window penalty rate	α_{time}	50
Battery depletion penalty	$\alpha_{battery}$	1000
Customer size	n	25
Random seed		42

Table 6: Vehicle and physics parameters

Parameter	**Symbol**	**Value**
Auxiliary power consumption	P_{aux}	1.5kW
Curb weight	M_{curb}	3009.585kg
Cargo weight	M_{load}	1269.605kg
Coefficient of aerodynamic drag	C_d	0.7
Frontal area	A	$3.912m^2$
Air density	ρ	$1.2041kg/m^3$
Coefficient of rolling resistance	C_r	0.01
Efficiency	η	0.7
Distance scale	s_d	$1000m/unit$
Velocity scale	s_v	$11.11m/s$
Gravitational acceleration	g	$9.81m/s^2$

The vehicle parameters (Table 6) are based off of Goeke[14] et al. and the 2025 Ford E-Transit [11] which is widely used by DHL [9]. Specifically this paper uses the Ford E-Transit Extended High model. The parameters that are assumed by this thesis are efficiency, auxiliary power consumption and distance/velocity scale. These parameters were selected as representative baselines for this experimental setup, and can be modified in future research to test for different operational effects. Auxiliary power consumption is modeled as an aggregation of various components such as AC or heating that consume energy as the vehicle travels. The efficiency factor is a single scalar that is applied to the conversion of mechanical energy to electrical energy, as a penalty which represents the efficiency loss of the vehicle.

Route geometry is assumed to be purely flat, with no hills or valleys. Furthermore, a vehicle cannot escape traffic on an arc, meaning that the vehicle will be stuck traversing the arc and cannot reroute. The traffic congestion is added into scenario resilience as a "reroute penalty" which acts as a proxy for the cost of conceptual rerouting, where the cost is approximated to be proportional to the arc distance and traffic congestion severity. Longer arcs incur a higher resilience penalty, and shorter arcs incur a lower resilience penalty. This pushes the vehicle to avoid long arcs that would be costly to detour from, instead opting to take shorter arcs from which it is easier to detour in case of traffic.

The generation size of 500 follows Deb et al., who uses the same value on their harder problems. Within this thesis, we choose to utilize a larger population size of 500 instead of Deb et al., who used a size of 100[8]. A high population size allows for greater exploration, while a high generation size enables better exploitation of the state space.

The deterministic baseline uses the VRPy package [22] to construct the optimal path for the deterministic model. The model has a time limit of 360s to find the optimal path, however from having observed several experiments, it seems that the VRPy package converges to a sound optimal path in less than 180s, hence the 360s upper bound can be considered generous.

The fleet size K in our experiment is unbounded, Although this is the case, the fleet size remains implicitly bounded as there are three competing pressure factors. First, the time-adherence relieves time pressure by spawning more vehicles to parallelize the tasks. Second, the energy objective penalizes the transfer from depot to customer arcs since each new vehicle adds two depot legs which implies a greater total distance and hence higher energy cost. Lastly, the capacity constraint ensures customer demands are met by having the sufficient amount of delivery vehicles. As such, there is a hard lower bound on the vehicle size, and that is:

$$K_{lower} = \left\lceil \frac{\sum_{i \in V_c} D_i}{C} \right\rceil \tag{23}$$

Where the numerator is the total demand of the customers and the denominator C is the capacity of the vehicle, ensuring that enough vehicles are dispatched to meet customer demand. The upper bound is implicit and maintained by a trade-off between the cost of energy required to dispatch more vehicles against the time-adherence benefits received from the parallelization effect of dispatching multiple vehicles.

This paper uses a simplified approach to Keskin et al. [18] partial recharge strategies by simply using full-recharge at every charger. This design choice abides by the simple original EVRPTW formulation.

5.2.1 Stochasticity Experiment

The main objective of this thesis is to study the MOS-EVRPTW model compared to a baseline EVRPTW model under controlled stochastic settings. The two stochastic parameters: breakdown probability p_{break} and traffic-noise σ are varied together across $p_{break} \in \{0.0, 0.05, 0.20\}$ and $\sigma \in \{0.0, 0.15, 0.60\}$, yielding a 3x3 grid that ranges from a fully deterministic environment where p_{break} and σ are 0 to a heavily disrupted one. The customer set is fixed at 25 and the amount of MC iterations is set to $N_{MC} = 500$. Each cell of the grid runs the full NSGA-II evolution and re-evaluates the deterministic baseline under the matching disruption distribution by simulating the deterministic baseline with the MC simulation. The grid is run across each of the different customer distributions so that the comparison covers clustered, random, and mixed customer distributions.

The experiment uses a population size of $N_{pop} = 500$ and a generation size of $N_{gen} = 500$. The generation size parameter is derived from the work of Deb et al., who ran NSGA-II for up to 500 generations on their harder problem settings. Deb et al. used a population of 100, however we adopt a larger $N_{pop} = 500$ to promote exploration of the substantially larger MOS-EVRPTW search space compared with the benchmark functions used by Deb et al.[8].

5.2.2 Monte Carlo Size Experiment

To determine the sufficient amount of MC iterations, we have conducted an experiment to run the MC simulation across $N_{MC} \in \{50, 250, 500\}$ with customers held fixed at 25, and stochastic disruption parameters $p_{break} = 0.05, \sigma = 0.15$ across all N_{MC} values. Furthermore, we calculate the Hypervolume of each N_{MC} across the three customer distributions using the Python library `moocore` which implements the Hypervolume calculation[1, 10, 29, 15]. This sub-experiment is aimed at quantifying both quantitatively and qualitatively how many simulated scenarios are required for the expected-value fitness estimates to stabilize.

5.2.3 Customer Size Experiment

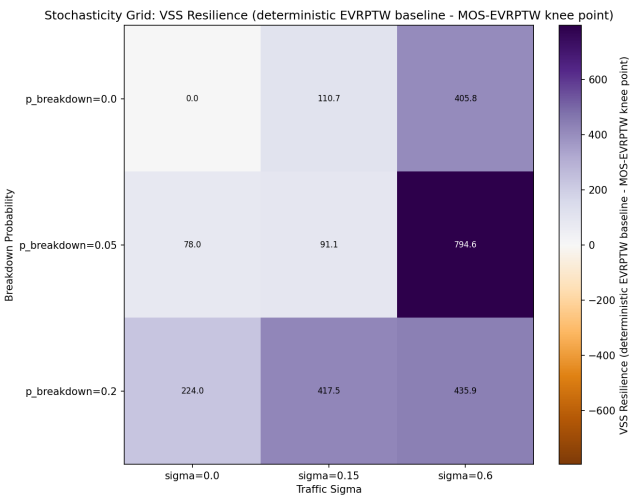
As mentioned in (Section 5), the experiments are conducted using 25 customers. However, as this is a limitation in our experiments, we have conducted an experiment to investigate how our framework would scale with different customer sizes. Experimenting from the smallest customer size to the maximum number of customers within the instance. We alter the customer size $N_{customers} \in \{10, 25, 50, 100\}$ with stochastic disruption parameters held constant at the slightly stochastic setting: $p_{break} = 0.05, \sigma = 0.15$ and $N_{MC} = 500$.

5.2.4 Seed Replication Experiment

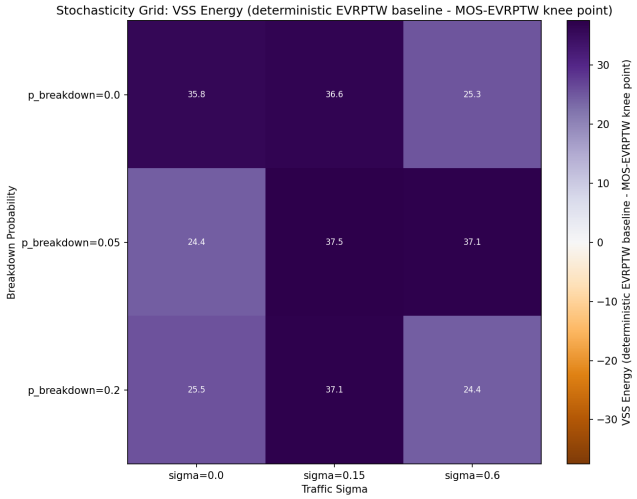
The seed replication experiment evaluates the best Pareto-front per-seed knee point per objective for the three customer instance distributions (C, R, RC) across three random seeds and two different customer instance sizes $n = 25$ and $n = 100$. Two stochasticity cells are reported: slightly stochastic setting ($p_{break} = 0.05$, $\sigma = 0.15$), highly stochastic setting ($p_{break} = 0.2$, $\sigma = 0.6$).

5.3 Results

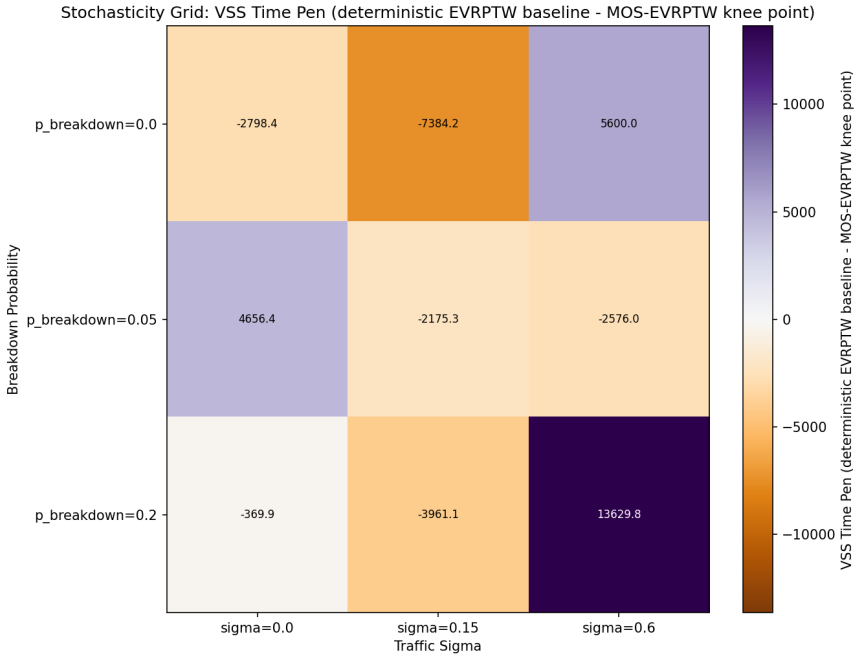
5.3.1 Stochasticity Experiment



(a) VSS Resilience.

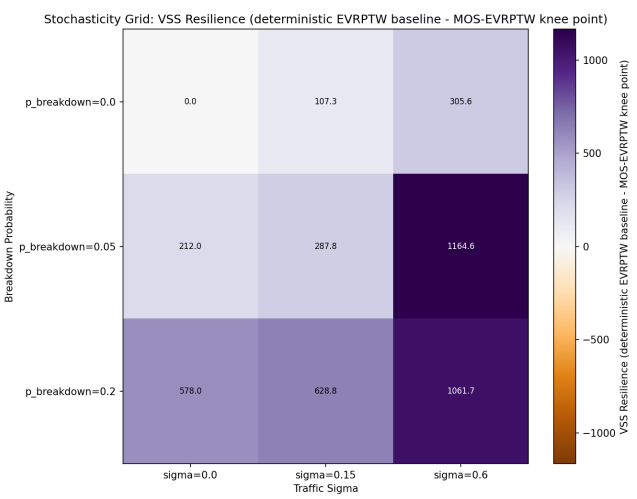


(b) VSS Energy.

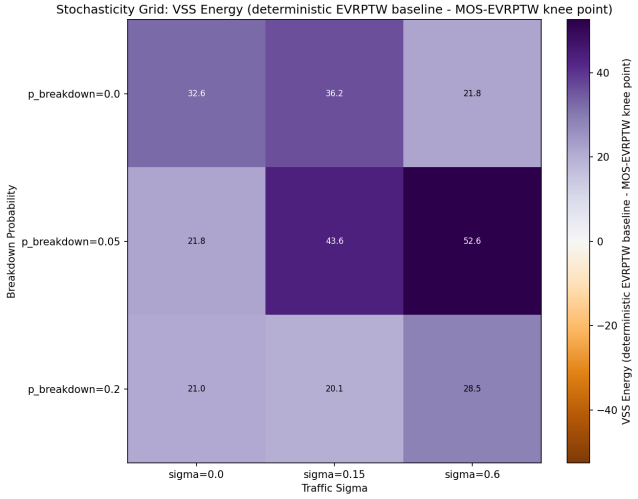


(c) VSS Time Penalty.

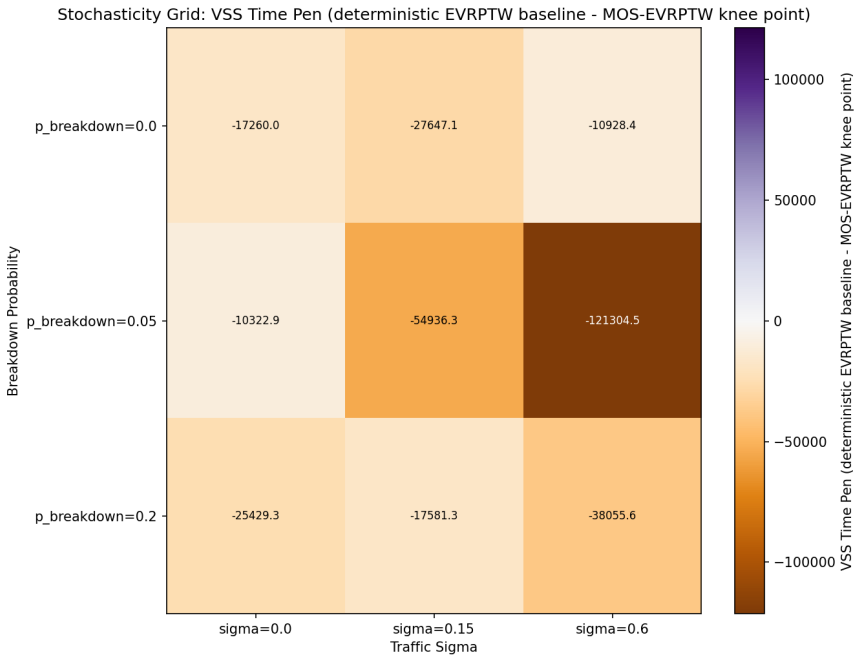
Figure 6: Value of the Stochastic solution on `rc101_21`, with default parameters ($n = 25, N_{pop} = N_{MC} = 500$), where the purple color corresponds to MOS-EVRPTW dominating the baseline and brown corresponds to baseline dominating the MOS-EVRPTW.



(a) VSS Resilience.

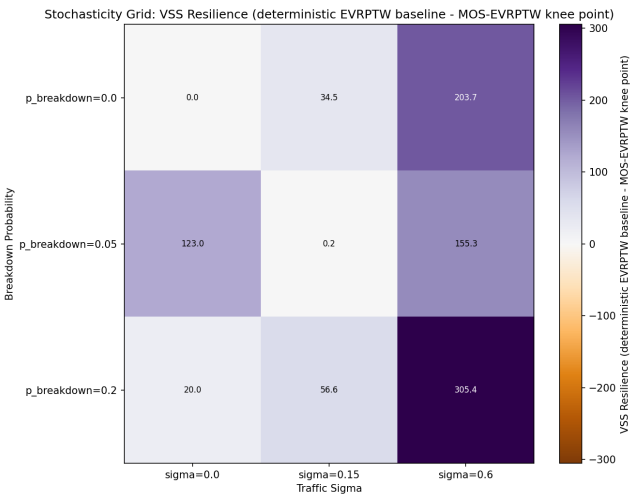


(b) VSS Energy.

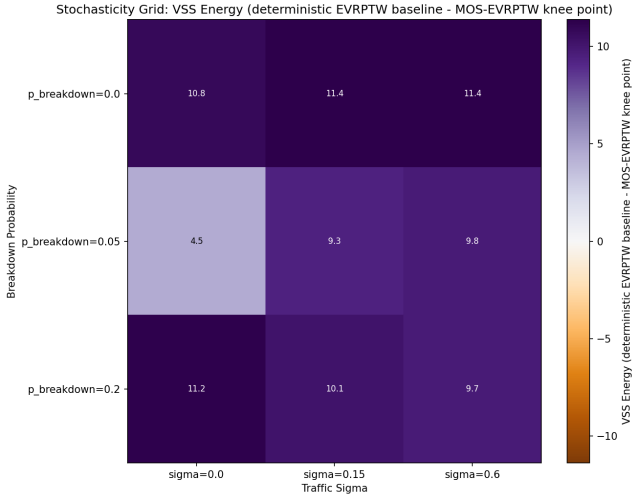


(c) VSS Time Penalty.

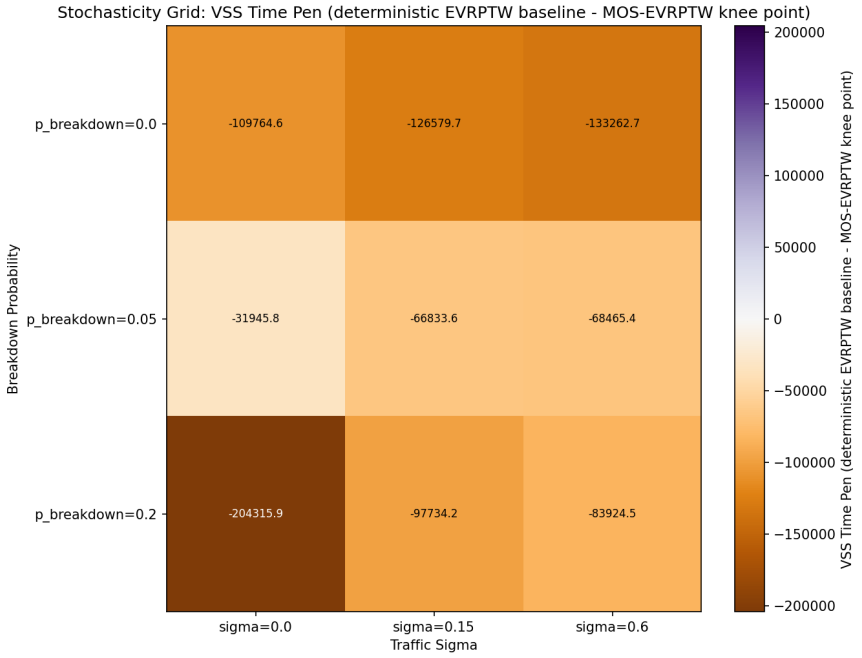
Figure 7: Value of the Stochastic solution on `r101_21`, with default parameters ($n = 25, N_{pop} = N_{MC} = 500$), where the purple color corresponds to MOS-EVRPTW dominating the baseline and brown corresponds to baseline dominating the MOS-EVRPTW.



(a) VSS Resilience.

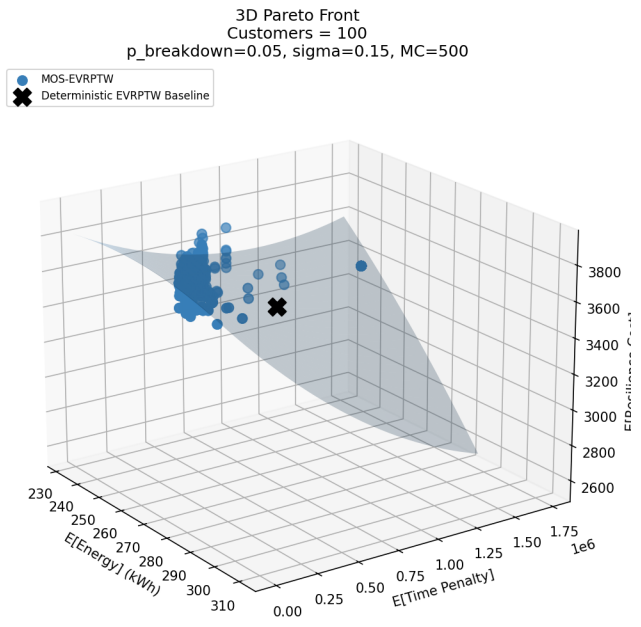


(b) VSS Energy.

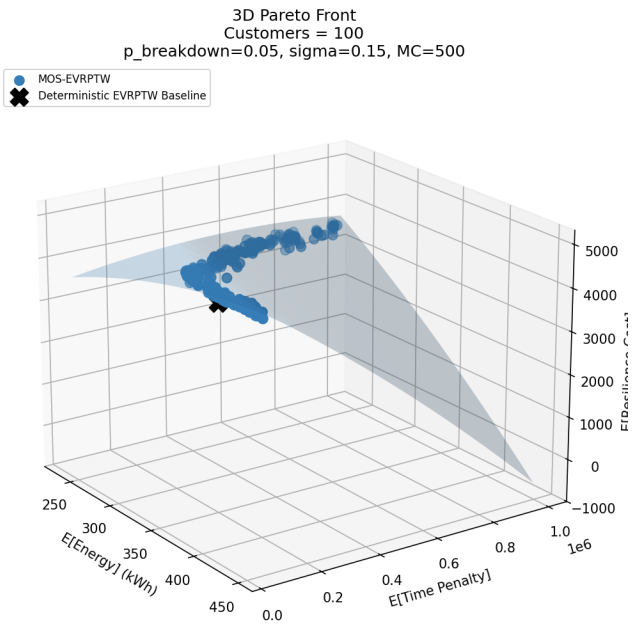


(c) VSS Time Penalty.

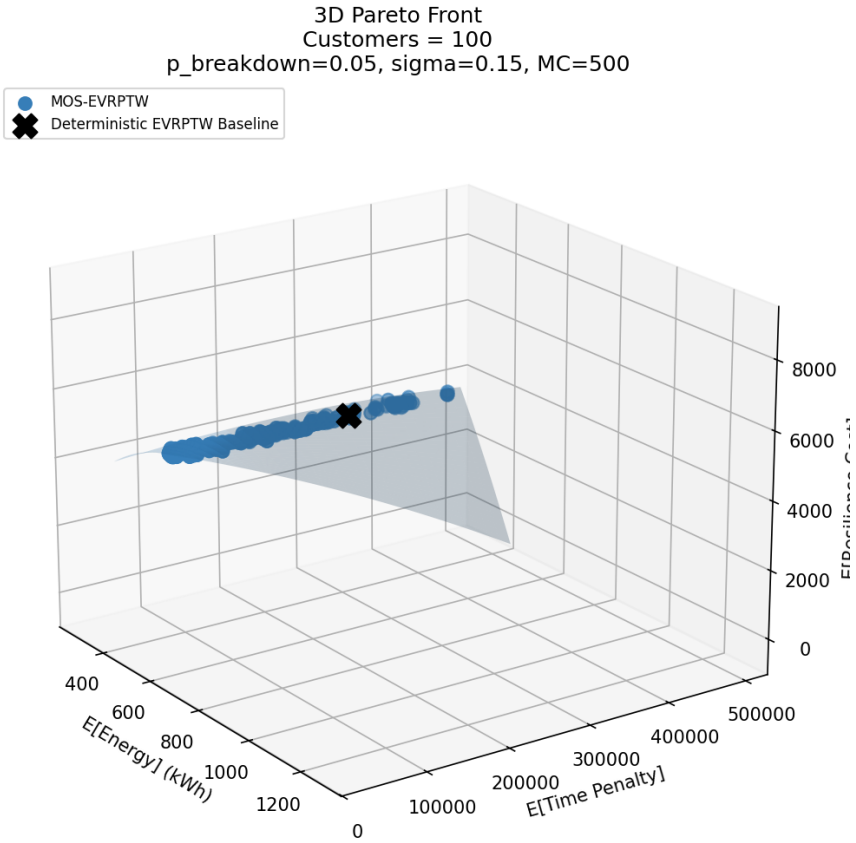
Figure 8: Value of the Stochastic solution on `c101_21`, with default parameters ($n = 25, N_{pop} = N_{MC} = 500$), where the purple color corresponds to MOS-EVRPTW dominating the baseline and brown corresponds to baseline dominating the MOS-EVRPTW.



(a) Clustered.



(b) Random Uniform.



(c) Mixed.

Figure 9: Pareto fronts of MOS-EVRPTW under slightly stochastic setting ($p_{break} = 0.05, \sigma = 0.15$) across the three customer distributions using all $n = 100$ customers and $N_{pop} = N_{MC} = 500$.

Out of each of the customer instances, the MOS-EVRPTW performs best on the `rc101_21` customer instance. It achieves substantially better energy, and resilience performance compared to the baseline. Time-window performance favors the baseline model in a low disruption risk environment, however MOS-EVRPTW becomes more advantageous under a high disruption risk environment. The energy gain is slight and roughly uniform across all cells in Figure 6, because the MOS-EVRPTW explicitly minimizes energy, whereas the deterministic baseline optimizes only for distance subject to time windows, with no energy objective. As stochasticity increases, the knee point shifts toward mainly time-penalty and slightly to resilience robustness (time penalty variance is large compared to the other objectives). This can be seen as trading away energy optimality for gains in the resilience and time penalty objectives.

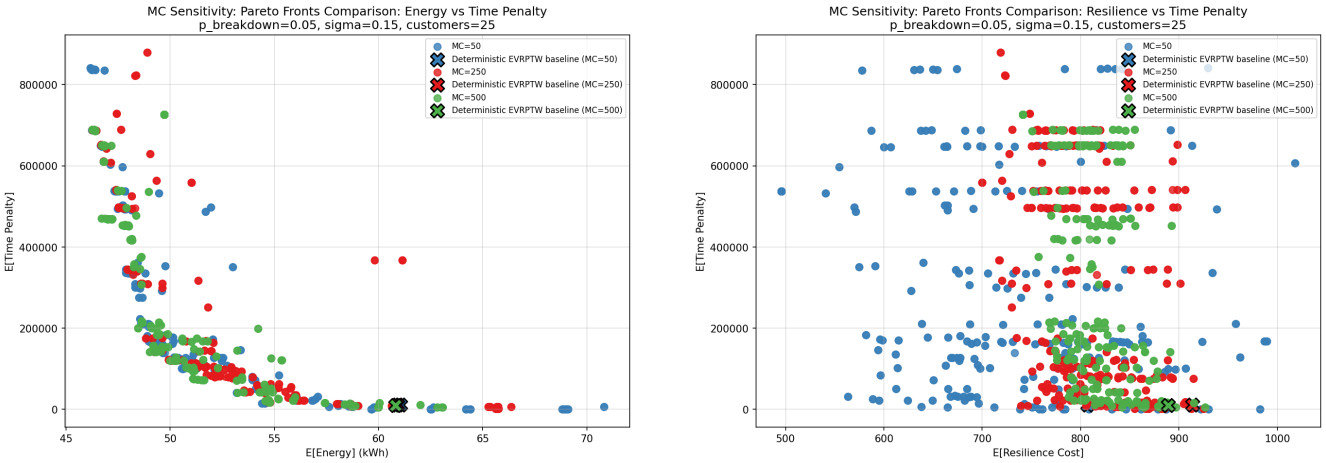
The resilience gains are largely dominated by σ parameter. The rerouting cost `delay_ratio * distance * delay_reroute_penalty` is proportional to arc distance. Longer arcs, when delayed, cost more in resilience than shorter arcs experiencing the same proportional delay. MOS-EVRPTW explicitly minimizes resilience and hence evolves routes that prefer shorter arcs, which also correlates with energy minimization. The baseline optimizes only deterministic distance subject to time windows, and it may use longer arcs where they are convenient. Under high σ the long arcs become expensive in resilience, as the congestion compounds over an arc and the vehicle cannot leave mid-traversal. This is why the VSS grows strongly with σ. The vehicle breakdown probability p_{break} has influence, but this influence is small compared to σ. This is because the breakdowns are arc independent Bernoulli events with a fixed cost, as such in every arc the route plan has the same probability of breakdown, as such, there is no strict systematic way that the MOS-EVRPTW can avoid breakdown costs. Although the MOS-EVRPTW can choose to take fewer shorter arcs, which would indeed systematically avoid breakdown costs as the exposure to breakdowns on the global scale would be smaller, as fewer arcs are traversed, and hence the likelihood of breaking down also becomes smaller. However, this has to be balanced with the resilience objective, which would prefer shorter arcs, since within those arcs traffic congestion would be easier to manage.

Throughout the instances, MOS-EVRPTW has struggled with time-window adherence. Only within the instance `rc101_21` has the MOS-EVRPTW been able to achieve results that have beat the baseline. The baseline wins the deterministic case $p_{break} = \sigma = 0$ in Figure 6 by the biggest margin, this is because the baseline VRPy solver[22] is explicitly designed to solve with respect to time windows. Under zero noise, the routes gather only the unavoidable time penalty from the tight customer time windows that exists in the customer instance. MOS-EVRPTW's knee under zero stochasticity points to a balanced solution that sacrificed some time window compliance for energy and resilience optimality. However, as the environment becomes more stochastic the MOS-EVRPTW is able to beat the baseline with a bigger margin at, but this is conditional on the route geometry and time-window tightness of the customer instance. The MOS-EVRPTW is aware of the stochasticity and can take an approach which accounts for breakdown delays and traffic congestion, allowing the routes to be constructed in a way that minimizes the effect of disruptions. However constructing these routes requires time-window slack, which allows for flexible route creation that leads to the MOS-EVRPTW beating baseline, as seen in Section 5.3.4.

Both `c101_21` and `r101_21` exhibit poor time-window adherence due to their unique properties: Figure 8 has clustered route geometry with negative slack (Table 1), and Figure 7 has uniform route geometry with 0 slack (Table 2). The difference is explained by the route geometry and

time-window slack. Instances with tight windows means that any disruption produce lateness, and with uniformly spread customers there is little routing freedom to recover, so the baseline is hard to beat. `r101_21` has tight time windows and uniform customer distribution, within this environment delays can hurt a lot more. However, since customers are uniformly spread, there is little routing freedom. Within both environments the VRPy deterministic solver is able to, with more ease, find an optimal solution that minimizes distance and adheres to time-windows, even under stochastic disruptions. However as the environment gets more stochastic, the VSS shifts towards favoring the MOS-EVRPTW in all objectives.

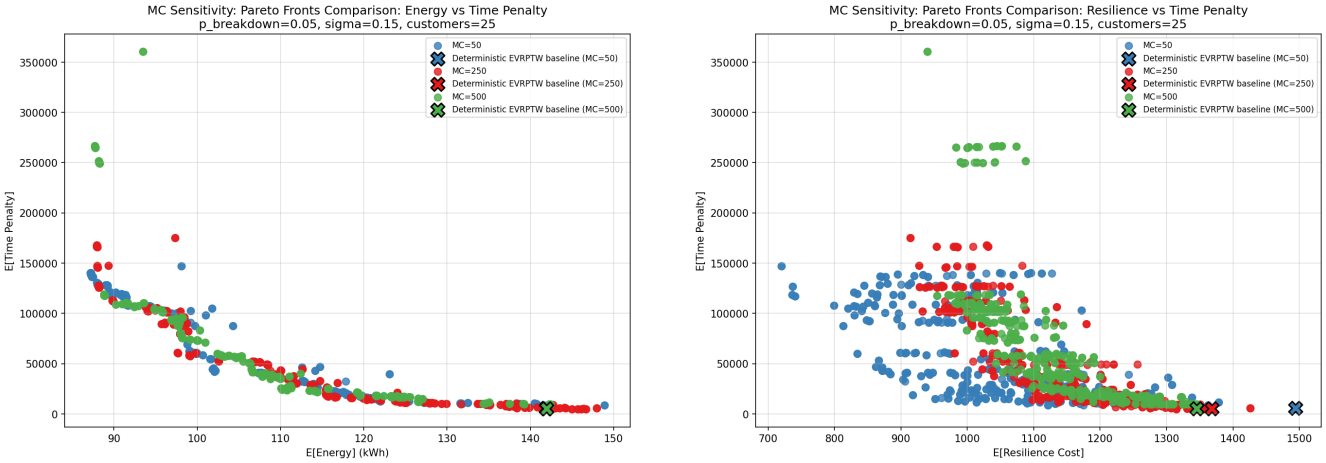
5.3.2 Monte Carlo Size



(a) Energy vs Time.

(b) Resilience vs Time.

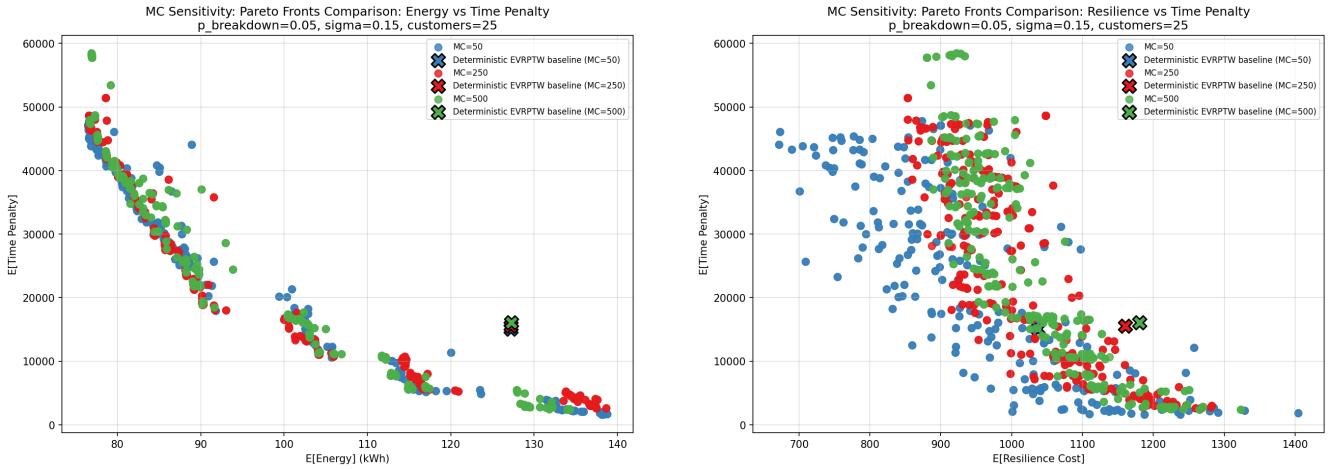
Figure 10: Monte Carlo iteration sensitivity across different N_{MC} within `c101_21` customer instance.



(a) Energy vs Time.

(b) Resilience vs Time.

Figure 11: Monte Carlo iteration sensitivity across different N_{MC} within `r101_21` customer instance.



(a) Energy vs Time.

(b) Resilience vs Time.

Figure 12: Monte Carlo iteration sensitivity across different N_{MC} within `rc101_21` customer instance.

The MC sensitivity experiment evaluates how the number of Monte Carlo simulations affect the quality and stability of the Pareto fronts produced by MOS-EVRPTW, holding all other parameters fixed at $p_{break} = 0.05, \sigma = 0.15, n = 25$.

We test across three MC levels: $\{50, 250, 500\}$. Each producing Pareto fronts that span across different energy and resilience levels in relation to time penalty. From these diagrams it is possible to derive the landscape geometry by looking at the energy and resilience axis. Clustered customer instance has customers that are tightly packed, and hence, in Figure 10 the average energy and resilience is lower than in Figure 11 & 12, this is due to the geometry of `c101_21` customer instance. Uniformly scattered customer instance has spread out customers over the space, widening the geometry, and hence increasing the amount of energy the average vehicle consumes. Furthermore, because of the uniformly distributed homogeneous and long-arc nature of the geometry, the disruptions naturally have a greater impact on resilience, as the longer arcs between the customers become expensive due to traffic congestion. As such, comparing the resilience axis of the clustered instance Figure 10 and uniformly distributed instance Figure 11, it is evident that the spread in the uniformly distributed instance is much wider. The mixed customer instance contains the smallest time penalty window, as seen in Figure 12. This is because the time windows in this instance are not too tight, allowing some room for disruptions which can be absorbed without any time-window adherence violation.

Across all three customer instances, the baseline (represented by the X marker) is clustered tightly around the same spot regardless of N_{MC}. This is to be expected, as the deterministic model produces fixed routes and the MC evaluation of those routes converges quickly. The baseline provides a reference point independent of the N_{MC} size.

The effect of N_{MC} on the front quality is most dominantly noticed in the dispersion of the fronts. $N_{MC} = 50$ produces a noticeably more dispersed Pareto front, with a great amount of outlier points, as seen in Figure 10. As a result $N_{MC} = 50$ produces much noisier solutions, which can propagate poor solutions to survive into the final Pareto front.

MC	C101	R101	RC101
50	0.2382	0.2311	0.1927
250	0.1741	0.1853	0.1478
500	0.1542	0.1826	0.1340

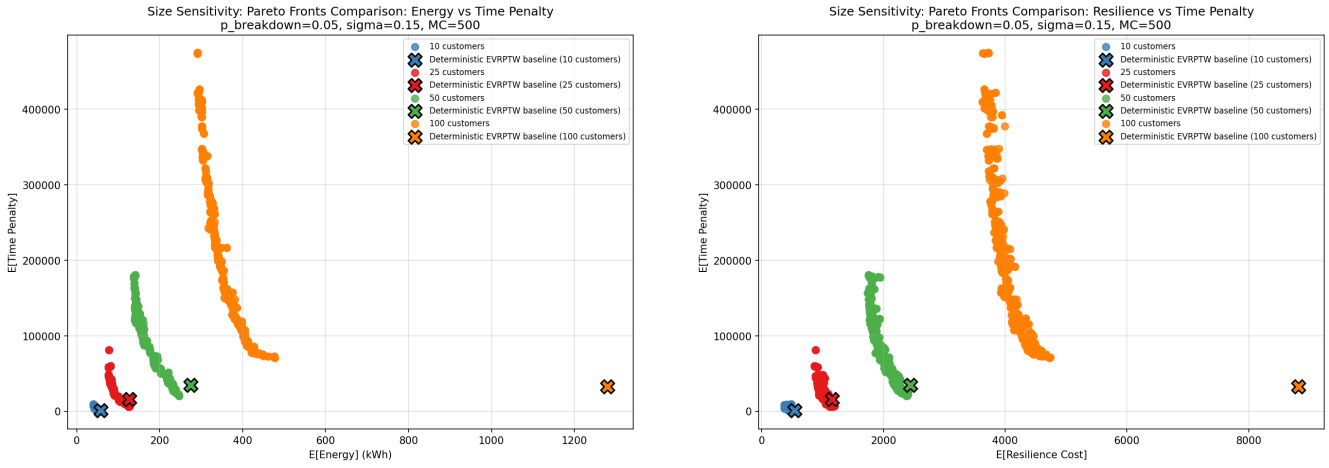
Table 7: Hypervolume (HV) indicator of the Pareto front across N_{MC} for all three customer distributions normalized by the worst-case objective space volume $V = \texttt{timepen} * \texttt{energy} * \texttt{resilience}$[1, 10, 29, 15] Higher values indicate larger dominated objective space.

The HV ratio drops sharply from $N_{MC} = 50 \rightarrow N_{MC} = 250$. Greater N_{MC} count beyond $N_{MC} = 250$ produce diminishing returns. The elevated HV at $N_{MC} = 50$ is an artifact of evaluation noise, since an optimistically biased fitness estimate pushes the apparent front toward the origin, which inflates the dominated volume. As N_{MC} increases this bias washes out and the HV decreases and stabilizes.

At $N_{MC} = \{250, 500\}$ the fronts become more compact, with greater overlap across both N_{MC} values. The change from 250 to 500 is smaller than the change from 50 to 250 as seen qualitatively by Figures 10,12,11, and quantitatively from the hypervolume in Table 7 which shows a significant drop in HV from 50 to 250, indicating that the estimates regressed to their true higher values, and the HV settles. This result is consistent with the law of large numbers (17).

The results support using $N_{MC} = 500$ in the experiments as the best available balance between accuracy and computation cost. However, using $N_{MC} = 250$ can be a sufficient sacrifice in order to trade accuracy for computational cost, since the fronts at $N_{MC} = \{250, 500\}$ have a great amount of overlap compared to $N_{MC} = \{50, 500\}$.

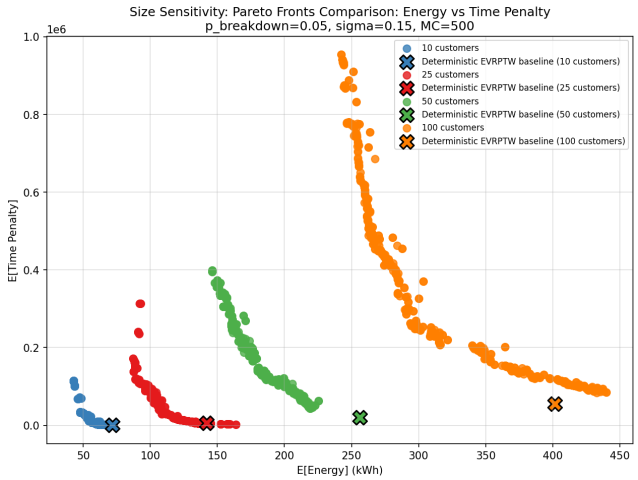
5.3.3 Customer Size



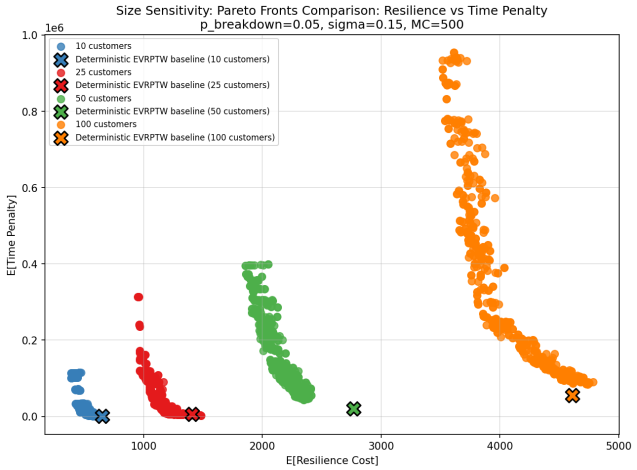
(a) Energy vs Time.

(b) Resilience vs Time.

Figure 13: Customer size sensitivity across different $n = \{10, 25, 50, 100\}$ within `rc101_21` customer instance.

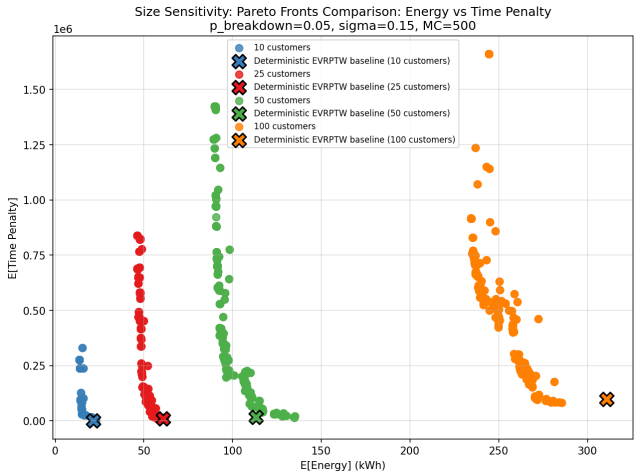


(a) Energy vs Time.

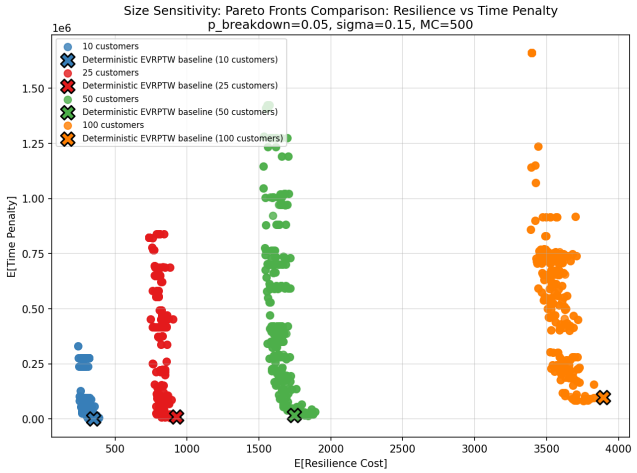


(b) Resilience vs Time.

Figure 14: Customer size sensitivity across different $n = \{10, 25, 50, 100\}$ within `r101_21` customer instance.



(a) Energy vs Time.



(b) Resilience vs Time.

Figure 15: Customer size sensitivity across different $n = \{10, 25, 50, 100\}$ within `c101_21` customer instance.

Within each customer instance size and each customer distribution, a similarly themed, tall shaped curve appears. At the low energy end, the points are spread over a huge vertical range of time penalty. At the high-energy end, the cloud narrows sharply into a tight low time-penalty tail.

The front is strongly convex, as indicated by the diminishing returns as the energy objective is pushed down further. Going from the high energy tip to the middle of each cluster buys a lot of time penalty improvement for cheap, going past that point costs disproportionately more. This balanced point is captured by the knee point.

Both energy vs time and resilience vs time figures are shaped roughly the same. As such, energy and resilience move together along the front, and they are not purely two independently competing objectives. Energy and resilience are correlated for the underlying reason that the disruptions incurred impact both objectives similarly; however, to a different degree. The implication of this finding is that it is possible to minimize the energy objective without large trade-offs in the resilience objective, and similarly, minimizing resilience would not incur large trade-offs in the energy objective. Minimizing the time penalty objective incurs heavy trade-offs in both energy and resilience objectives, as seen by the tall convex curve with huge vertical range in Figures 15,14,13.

In all three customer instances and at all size levels, the Pareto front scales upward and to the right as n increases. Larger problems have higher energy, resilience, and time penalty. This is to be expected, as the amount of customers increases, the vehicles will have to traverse a longer path, depleting more energy, and any given disruption event can have a cascading collateral effect on customer time-window adherence that increases resilience cost and time penalty.

The more interesting findings from the data reflect how the fronts scale relative to the baselines as n gets bigger, constrained by the amount of time-window slack (Tables 1, 2, 3) and customer geometry. In Figure 13 at $n = 10$, the baseline sits roughly close to the Pareto front; however, at $n = 25$, the baseline roughly starts shifting away from the front, which grows increasingly more prominent as $n = 50 \rightarrow n = 100$, where the deterministic baseline solution sits predominantly outside of the Pareto-front.

n customers	Objective	Baseline	MOS knee	Δ	% improve
10	Energy	57.76	42.67	15.09	26.12%
	Time Pen	1070.09	3807.32	-2737.23	-255.80%
	Resilience	545.98	433.46	112.52	20.61%
25	Energy	127.23	91.60	35.63	28.01%
	Time Pen	15857.51	20842.71	-4985.20	-31.44%
	Resilience	1154.90	945.50	209.39	18.13%
50	Energy	274.78	140.42	134.36	48.90%
	Time Pen	34631.90	122256.47	-87624.56	-253.02%
	Resilience	2444.83	1940.21	504.62	20.64%
100	Energy	1279.52	344.96	934.57	73.04%
	Time Pen	32572.63	191470.34	-158897.71	-487.83%
	Resilience	8817.95	4163.59	4654.36	52.78%

Table 8: Comparison between deterministic baseline and the knee point of the MOS-EVRPTW Pareto front in instance `rc101_21` across instance sizes $n = \{10, 25, 50, 100\}$.

n customers	Objective	Baseline	MOS knee	Δ	% improve
10	Energy	21.46	15.16	6.30	29.34%
	Time Pen	3.35	32810.07	-32806.73	-980452.69%
	Resilience	348.65	268.62	80.03	22.95%
25	Energy	60.86	49.24	11.62	19.10%
	Time Pen	10545.64	156141.72	-145596.09	-1380.63%
	Resilience	925.37	772.77	152.61	16.49%
50	Energy	113.05	96.39	16.66	14.74%
	Time Pen	16446.03	197200.95	-180754.91	-1099.08%
	Resilience	1746.83	1690.36	56.47	3.23%
100	Energy	310.80	241.37	69.43	22.34%
	Time Pen	96826.57	524898.45	-428071.88	-442.10%
	Resilience	3894.40	3545.45	348.94	8.96%

Table 9: Comparison between deterministic baseline and the knee point of the MOS-EVRPTW Pareto front in instance `c101_21` across instance sizes $n = \{10, 25, 50, 100\}$.

n customers	Objective	Baseline	MOS knee	Δ	% improve
10	Energy	71.75	47.54	24.21	33.74%
	Time Pen	244.73	33859.93	-33615.21	-13735.82%
	Resilience	649.10	493.37	155.73	23.99%
25	Energy	141.92	133.85	8.07	5.69%
	Time Pen	5705.15	9291.73	-3586.58	-62.87%
	Resilience	1408.45	1168.88	239.57	17.01%
50	Energy	256.14	179.94	76.19	29.75%
	Time Pen	19838.84	139288.72	-119449.89	-602.10%
	Resilience	2769.20	2198.65	570.55	20.60%
100	Energy	401.60	294.41	107.19	26.69%
	Time Pen	54581.57	256400.62	-201819.05	-369.76%
	Resilience	4608.07	3966.79	641.28	13.92%

Table 10: Comparison between deterministic baseline and the knee point of the MOS-EVRPTW Pareto front in instance `r101_21` across instance sizes $n = \{10, 25, 50, 100\}$.

The percentage improvement for time penalty reports extremely large and inconsistent values when compared to energy and resilience. For instance: -980452.69% in Table 9. This is a product of the relative change formula of percentage improvement. The deterministic baseline incurs near-zero time-window violation at small n compared to the MOS knee point that balances the three different objectives. The Time penalty is therefore better read from the absolute Δ column.

The rate at which the MOS-EVRPTW front separates from the baseline on energy and resilience differs strongly by customer instance. In `rc101_21` the energy gap between the baseline and the MOS-EVRPTW front grows with the problem size monotonically ($26.12\% \rightarrow 28.01\% \rightarrow 48.90\% \rightarrow 73.04\%$), on the other hand, resilience does not grow monotonically but it exhibits the greatest improvement over the baseline at $n = 100$. Inherently the monotonic gains in energy is due to MOS-EVRPTW's explicit energy objective, and the fact that the deterministic model optimizes distances subject to time windows purely. Because the baseline moves right faster than the front on energy as the customer size grows in `rc101_21`, there is potential for the MOS-EVRPTW to be used on larger sized customer problems similar to the customer instance environment of `rc101_21` to exploit the inherent energy advantage.

In `r101_21` and `c101_21`, by contrast, both energy and resilience improvement do not grow monotonically with n, but rather, both models exhibit their best energy/resilience performance compared to baseline on $n = 10$, then falling off at a higher customer instance size and then slowly regaining the energy advantage over the baseline as $n = 25 \rightarrow n = 100$. Resilience fluctuates roughly within a stable range rather than following any clear scaling trend.

There are strict conditions for the MOS-EVRPTW to scale better in energy compared to the baseline. The Figures 14,15 show the distinction of the energy advantage of the MOS-EVRPTW remain consistently smaller compared to 13. Following Tables 1,2,3 it is apparent that customer instances `c101_21` and `r101_21` have tight time-windows compared to the flexible time-windows in `rc101_21`. The MOS-EVRPTW is able to exploit its energy/resilience advantage most consistently, with a clear growth trend in instance `rc101_21` with flexible time windows; instances with tight-time windows give smaller and less predictable advantages to the MOS-EVRPTW model. The reasoning behind the trade-off in the energy/resilience objective as the time-window tightens is due to the fact that without enough time-window slack, the energy efficient and resilient route reordering cascades into time-window adherence violations, where resilient and energy efficient routes would cause an increase time penalty, as seen in Figure 14.

The Pareto front becomes more curved and well-defined with increasing n. At $n = 10$ there is very little trade-off to explore because of the tiny pool of customers, and within those instances almost all route structures are feasible and the three objectives are not in strong conflict. As the amount of customers increases the front becomes well-spread and smooth, spanning a wide range. As such, the higher the amount of customers, the more meaningful it is to use the MOS-EVRPTW approach to find trade-offs between the objectives.

5.3.4 Seed Replication

Instance	Objective	MOS knee mean	± std	BASE mean	± std	Delta	Winner
c101_21	Energy	50.62	0.51	60.88	0.01	10.26	MOS
	Time Pen	102106.69	18306.04	10528.67	28.68	-91578.02	BASE
	Resilience	835.15	66.15	894.73	40.90	59.58	MOS
r101_21	Energy	105.74	3.93	141.90	0.03	36.15	MOS
	Time Pen	37442.70	11504.35	5403.83	31.03	-32038.88	BASE
	Resilience	1074.59	95.09	1338.17	12.34	263.59	MOS
rc101_21	Energy	91.34	0.66	127.23	0.05	35.89	MOS
	Time Pen	20957.70	1366.00	16030.02	489.95	-4927.68	BASE
	Resilience	960.33	48.88	1178.82	62.65	218.49	MOS

Table 11: Seed replication, slightly stochastic cell ($p_b = 0.05$, $\sigma = 0.15$, MC=500, $N_{pop} = 500$, $N_{gen} = 500$, $n = 25$, seeds= $\{42, 84, 48\}$). For each seed a representative MOS-EVRPTW solution is represented as the per-seed knee which is averaged across the seeds. MOS-EVRPTW columns report mean ±std (across seeds) of the per-seed knee value on each objective. Baseline columns report mean ±std across seeds of the deterministic VRPy solution evaluated under the same stochastic environment. $\Delta = BASE - MOS$, where positive Δ favors MOS-EVRPTW.

Instance	Objective	MOS knee mean	± std	BASE mean	± std	Delta	Winner
c101_21	Energy	51.56	3.67	60.45	0.04	8.88	MOS
	Time Pen	123095.68	100218.11	14090.06	179.54	-109005.62	BASE
	Resilience	3616.77	76.29	3926.03	55.35	309.25	MOS
r101_21	Energy	106.39	3.33	140.83	0.00	34.44	MOS
	Time Pen	75122.51	26176.89	25476.19	543.66	-49646.33	BASE
	Resilience	4958.03	229.50	6095.78	78.54	1137.76	MOS
rc101_21	Energy	100.78	12.90	126.23	0.05	25.45	MOS
	Time Pen	44462.36	20240.12	55094.37	317.64	10632.00	MOS
	Resilience	4631.77	416.00	5230.97	37.91	599.20	MOS

Table 12: Seed replication, highly stochastic cell ($p_{break} = 0.20$, $\sigma = 0.60$, MC=500, $N_{pop} = 500$, $N_{gen} = 500$, $n = 25$, seeds= $\{42, 84, 48\}$). Same conventions as Table 11.

Across all instances MOS-EVRPTW wins on energy and resilience. The magnitude of MOS-EVRPTW win (Δ) depends on the customer instance: `c101_21` with clustered geometry and tightest time-windows shows the smallest gains in all objectives across low and high stochastic environments, `r101_21` shows the largest energy/resilience gains in both stochastic environments, and `rc101_21` shows the largest improvement in time penalty across both environments. Across all customer instances and in both high and low stochastic environments, the energy of the baseline remains consistent within the same environment. Within all of the customer instances the low

stochasticity environment baseline energy corresponds roughly to the high stochasticity environment baseline energy value. This confirms that energy is structurally determined by route geometry and it is largely insensitive to stochasticity level, since the same routes produce similar energy consumption regardless of how much traffic variance or breakdown probability is injected.

Instance	Objective	Δ (low)	Δ (high)	Δ %
c101_21	Energy	10.26	8.88	-13.37%
	Time Pen	-91578.02	-109005.62	-19.03%
	Resilience	59.58	309.25	419.04%
r101_21	Energy	36.15	34.44	-4.74%
	Time Pen	-32038.88	-49646.33	-54.96%
	Resilience	263.59	1137.76	331.65%
rc101_21	Energy	35.89	25.45	-29.08%
	Time Pen	-4927.68	10632.00	315.76%
	Resilience	218.49	599.20	174.25%

Table 13: Change in MOS-EVRPTW advantage from the slightly stochastic (Δ(low)) cell in Table 11 to the highly stochastic (Δ(high)) cell in Table 12. $\Delta = BASE - MOS$, where positive Δ favors MOS-EVRPTW. Δ % = (Δ(high)-Δ(low))/|Δ(low)|×100 is the growth of that advantage relative to the slightly stochastic cell; a positive Δ % means that the MOS-EVRPTW advantage widens under high stochasticity.

Table 13 isolates how the MOS-EVRPTW advantage changes with respect to stochasticity, and the three objectives behave differently. Resilience shows the clearest improvement signal, its advantage grows in every instance by roughly 175% to 420%. This is the objective through which the disruptions propagate, so as σ and p_{break} rise the fixed deterministic routes accumulate reroute and breakdown penalties that the stochastic-aware routes are built to avoid. Because the rerouting penalty term scales with delay × distance, the heavier congestion of the high cell disrupts the baselines long arcs disproportionately, which widens the gap between the baseline and the MOS-EVRPTW. `c101_21` shows the largest percentage growth in resilience, but the smallest base (Δ rises only from 59.58 to 309.25), so in absolute terms this advantage remains the weakest of the three, which is consistent with its clustered geometry with tightest time-windows.

The time penalty significantly favors the baseline model across all cells, except for `rc101_21` at high stochasticity where the MOS-EVRPTW beats the baseline by 315%. The MOS-EVRPTW performs well in terms of the time objective in the instance `rc101_21`, compared to the other customer instances, which reflects it's time-window flexibility. Given flexible time windows and heavily stochastic settings, MOS-EVRPTW can build route structures that absorb the delays, while the deterministic baseline continuously compounds breakdowns and traffic penalties across its fixed routes.

Instance	Objective	MOS knee mean	± std	BASE mean	± std	Delta	Winner
c101_21	Energy	234.66	11.41	360.51	0.08	125.86	MOS
	Time Pen	733931.18	196769.19	156823.60	200.82	-577107.58	BASE
	Resilience	3620.89	49.54	4186.01	30.65	565.12	MOS
r101_21	Energy	291.21	22.18	401.02	0.07	109.81	MOS
	Time Pen	339432.41	91613.43	53996.97	96.47	-285435.44	BASE
	Resilience	4059.33	101.48	4601.31	6.69	541.99	MOS
rc101_21	Energy	354.20	31.08	1266.93	0.16	912.73	MOS
	Time Pen	162007.80	33281.33	30007.32	62.33	-132000.48	BASE
	Resilience	3985.66	15.87	8806.79	109.36	4821.13	MOS

Table 14: Seed replication, slightly stochastic cell ($p_b = 0.05$, $\sigma = 0.15$, MC=500, $N_{pop} = 500$, $N_{gen} = 500$, $n = 100$, seeds= $\{42, 84, 48\}$). For each seed a representative MOS-EVRPTW solution is represented as the per-seed knee which is averaged across the seeds. MOS-EVRPTW columns report mean ±std (across seeds) of the per-seed knee value on each objective. Baseline columns report mean ±std across seeds of the deterministic VRPy solution evaluated under the same stochastic environment. $\Delta = BASE - MOS$, where positive Δ favors MOS-EVRPTW.

Instance	Objective	MOS knee mean	± std	BASE mean	± std	Delta	Winner
c101_21	Energy	237.42	12.99	417.35	0.10	179.94	MOS
	Time Pen	681451.20	352971.71	244273.29	1171.87	-437177.91	BASE
	Resilience	15053.79	43.68	19692.75	135.14	4638.97	MOS
r101_21	Energy	300.42	18.83	398.03	0.15	97.61	MOS
	Time Pen	433227.99	79591.60	170417.50	1394.55	-262810.50	BASE
	Resilience	17308.06	856.19	19982.79	123.29	2674.73	MOS
rc101_21	Energy	326.38	6.26	1257.08	0.13	930.70	MOS
	Time Pen	371258.72	17178.31	104791.74	819.97	-266466.98	BASE
	Resilience	16987.36	233.24	40714.35	172.82	23726.99	MOS

Table 15: Seed replication, highly stochastic cell ($p_{break} = 0.20$, $\sigma = 0.60$, MC=500, $N_{pop} = 500$, $N_{gen} = 500$, $n = 100$, seeds= $\{42, 84, 48\}$). Same conventions as Table 11.

Instance	Objective	Δ (low)	Δ (high)	Δ %
c101_21	Energy	125.86	179.94	42.97%
	Time Pen	-577107.58	-437177.91	24.25%
	Resilience	565.12	4638.97	720.88%
r101_21	Energy	109.81	97.61	-11.11%
	Time Pen	-285435.44	-262810.50	7.93%
	Resilience	541.99	2674.73	393.50%
rc101_21	Energy	912.73	930.70	1.97%
	Time Pen	-132000.48	-266466.98	-101.87%
	Resilience	4821.13	23726.99	392.15%

Table 16: Change in MOS-EVRPTW advantage from the slightly stochastic (Δ(low)) cell in Table 14 to the highly stochastic (Δ(high)) cell in Table 15. $\Delta = BASE - MOS$, where positive Δ favors MOS-EVRPTW. Δ % = (Δ(high)-Δ(low))/|Δ(low)|×100 is the growth of that advantage relative to the slightly stochastic cell; a positive Δ % means that the MOS-EVRPTW advantage widens under high stochasticity.

At $n = 100$ the energy and resilience gains of the MOS-EVRPTW hold similar to the pattern observed at $n = 25$, and the time penalty distinction becomes clearer. The MOS-EVRPTW wins in energy and resilience in every instance at both stochasticity levels, while the deterministic baseline now wins all time penalty scenarios across the six cells, removing the exception seen at $n = 25$, where `rc101_21` flipped to a MOS-EVRPTW win on time penalty under high stochasticity. At $n = 100$ that reversal does not occur, and the baseline time-penalty widens by 101.87% as seen in Table 16.

The knee's time penalty carries high variance across the three seeds because the front is nearly vertical, as seen in Figures 14,15,13 in the time direction, as such, small shifts in front geometry move the knee dominantly along time while leaving its energy and resilience coordinates stable. Therefore, the knee is a robust summary for energy and resilience, but for time-window penalty, it is noisy.

Resilience is the only objective whose response to stochasticity is preserved across instance sizes. The resilience advantage grows under high stochasticity in every cell at each customer instance size.

6 Limitations

Limited customer distribution variability and size. The experiments are conducted on three Goeke instances, one per customer distribution (`(r/c/rc101_21)`). There exists various other customer distribution instances that have their own unique geometry; this area is not explored. Furthermore, the seed experiment runs over 3 seeds, this can be expanded on future work to calculate statistical significance.

Charging policy always recharges to full. This goes against the findings of Keskin et al[18], who showed that partial recharging is superior to full recharging.

Utilization of purely first-improvement 2-opt instead of best-improvement 2-opt in mutation trades convergence speed of the Pareto front for computational speedup.

Implicit horizon enforcement, allows our study to be flexible with when vehicles must return to the depot. The horizon time is not enforced as a feasibility constraint by the genetic operators, but it is penalized in the MC fitness evaluation. Depending on the nature of the operations this can be justifiable or even a beneficial approach if a depot has flexible windows by which the vehicles must arrive back, however in the instance where a company has a depot with strict opening hours, it would not be realistic to assume that the vehicles could be able to return to the depot past their horizon time.

The disruption model is limited and treats vehicle breakdowns as a single per-arc Bernoulli variable with a fixed probability, independent of road condition, weather, time of day, vehicle age. Traffic congestion is treated as a single global log-normal multiplier with no spatial correlation across nearby arcs, and the correlation of traffic that changes depending on the time of the day not modeled.

The simplified energy model assumes flat terrain and uses a single aggregate powertrain efficiency, and treats auxiliary power as a constant that does not depend on conditions such as weather or driving mode. Furthermore, the energy objective and the battery state of charge are separate: the energy objective is made in the vision of the simplified model by Goeke et al.[14], while the battery state of charge is represented by a simple linear function of distance according to the simplistic implementation of Schneider et al. Additionally, the vehicle masses are taken from a single real-world reference vehicle rather than a distribution of fleet vehicles.

The deterministic baseline is a simplistic battery-blind model and solves a VRPTW with chargers stripped from the graph and is then converted to an EVRPTW by injecting the closest charging stations afterwards via the same charging-repair procedure used by the stochastic MOS-EVRPTW (Section 4.8). A true battery-aware EVRPTW solver would optimize for routing and charging together, and thus it would generally achieve roughly equal or lower cost. Our baseline can therefore be considered weaker than a true battery-aware EVRPTW, and the reported MOS-EVRPTW advantages may be overstated relative to a comparison against a fully battery-aware solver.

The resilience and time objective depend on the chosen penalty coefficients, and as such, the experimental data gathered is conditional on these parameters. Analysis over these coefficients is left to future work, and as such, the reported data should be read within the context of the chosen coefficient environment.

7 Further Research

Scale the problem instance, experiment on all available customer instances, using the full customer instance size $n = 100$. Sampling strategies can be incorporated to reduce the computational cost of experimenting on larger instances.

The MOS-EVRPTW framework can be extended to adapt to its environment. Allowing the vehicle to make real-time recourse actions, directly responding to realized disruptions. The disruption model should be extended by introducing new concepts such as rush-hour traffic and non rush-hour traffic

to model how traffic congestion correlates with the time of day. Furthermore, the deployed fleet can be extended from a homogeneous fleet to a heterogeneous fleet. Time-varying traffic congestion and heterogeneous fleet are concepts explored as the Green Vehicle Routing and Scheduling Problem within the work done by Xiao et al.[30]. The vehicle breakdowns on an arc should be extended from a plain per-arc Bernoulli with a fixed probability to a model that is specific to current conditions: weather, road condition, time of day, vehicle age.

The full-recharge at every station policy adopted in this thesis (Section 4.5) could be relaxed to the partial recharge strategies of Keskin et al.[18], which would require extending Algorithm 1 to choose a charge amount rather than always fully-recharging to Q, and re-evaluating whether the energy/resilience trade-offs reported in Section 5 still hold under partial recharging.

The uncertainty models can be enhanced by incorporating correlated travel times, where congestion on an arc can affect multiple nearby arcs simultaneously, varying the disruption probabilities with the amount of time that the vehicle has been operational for, and incorporating the full energy model by Goeke and Schneider with variable road geometry that is not purely flat[14]. Furthermore, an experiment should be conducted to study how different penalty coefficients affect the resilience and time objectives.

Furthermore, the NSGA-II implementation can be compared with other multi-objective algorithms such as NSGA-III to reveal whether different algorithm frameworks fit better for this problem structure.

8 Conclusions

This thesis presents the Multi-Objective Stochastic Electric Vehicle Routing Problem with Time Windows (MOS-EVRPTW), an extension of the deterministic EVRPTW that introduces stochastic disruptions and multi-objective optimization. The three objectives are to minimize energy consumption, time-window adherence violations as a proxy for customer dissatisfaction, and recourse cost as a measure of operational resilience. The results show that, in general, stochastic solutions improve expected energy and resilience performance relative to deterministic solutions.

The resilience advantage of MOS-EVRPTW grows consistently as the environment becomes more stochastic, given enough time-window slack and a customer geometry that provides meaningful route alternatives. The energy advantage, by contrast, is roughly insensitive to the level of stochasticity because energy is determined mainly by route geometry, and the baseline energy is similar in magnitude across the low and high disruption cells. Improvements in time-window adherence are less uniform because the deterministic baseline is highly specialized for time feasibility. As energy and resilience are not strongly competing objectives, it is possible to minimize resilience at only a small cost in the energy objective, however both energy and resilience are strongly competing with time penalty, and as such minimizing resilience or energy comes with a high time penalty trade-off, as visualized by the tall convex Figures 15,14,13 with huge vertical range. In a highly stochastic environment, with flexible time-windows (slack), the MOS-EVRPTW can dominate the baseline across all three objectives as observed for `rc101_21` at $n = 25$.

The simulation-based two-stage NSGA-II approach does improve Pareto-efficiency over a deterministic model in resilience and energy objective, falling short on time-window adherence in less

stochastic instances that have less slack. Overall, the framework is most valuable to decision-makers who require explicit trade-offs among energy efficiency, service reliability, and disruption resilience. The increase in flexibility allows the decision-maker to choose from many different alternatives, rather than being constrained to a single deterministic route plan.

References

[1] Nicola Beume, Carlos M. Fonseca, Manuel Lopez-Ibanez, LuÍs Paquete, and Jan Vahrenhold. On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation*, 13(5):1075–1082, 2009.

[2] John R. Birge and François Louveaux. Introduction to stochastic programming. *Springer Series in Operations Research and Financial Engineering*, 2011.

[3] Juergen Branke, Kalyan Deb, Henning Dierolf, and Matthias Osswald. Finding knees in multi-objective optimization. pages 722–731, 12 2004.

[4] G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812, 1958.

[5] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

[6] I Das. On characterizing the "knee" of the pareto curve based on Normal-Boundary intersection. *Structural optimization*, 18(2):107–115, October 1999.

[7] François-Michel De Rainville, Félix-Antoine Fortin, Marc-André Gardner, Marc Parizeau, and Christian Gagné. Deap: A python framework for evolutionary algorithms. pages 85–92, 07 2012.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Trans. Evol. Comp*, 6(2):182–197, April 2002.

[9] DHL Group. Milestone: 2,400 Ford Pro e-vans strengthen the electric delivery fleet of Deutsche Post and DHL in Germany, July 2025. Press release. Accessed: 2026-05-10.

[10] C.M. Fonseca, L. Paquete, and M. Lopez-Ibanez. An improved dimension-sweep algorithm for the hypervolume indicator. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1157–1163, 2006.

[11] Ford Motor Company. 2025 Ford E-Transit technical specifications, 2025. Accessed: 2026-05-10.

[12] Michel Gendreau, Ola Jabali, and Walter Rei. 50th anniversary invited article—future research directions in stochastic vehicle routing. *Transportation Science*, 50(4):1163–1173, 2016.

[13] Dominik Goeke. E-vrptw instances. 10.17632/h3mrm5dhxw.1, 2019. Accessed: 2026.

[14] Dominik Goeke and Michael Schneider. Routing a mixed fleet of electric and conventional vehicles. *European Journal of Operational Research*, 245(1):81–99, 2015.

[15] Andreia P. Guerreiro and Carlos M. Fonseca. Computing and updating hypervolume contributions in up to four dimensions. *IEEE Transactions on Evolutionary Computation*, 22(3):449–463, 2018.

[16] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[17] Angel A. Juan, Javier Faulin, Scott E. Grasman, Markus Rabe, and Gonçalo Figueira. A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, 2:62–72, 2015.

[18] Merve Keskin and Bülent Çatay. Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 65:111–127, 2016.

[19] Shen Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269, 1965.

[20] Jorge E Mendoza and Juan G Villegas. A multi-space sampling heuristic for the vehicle routing problem with stochastic demands. *Optimization Letters*, 7(7):1503–1516, October 2013.

[21] Nicholas Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949. PMID: 18139350.

[22] Romain Montagné, David Torres Sanchez, and Halvard Olsen Storbugt. Vrpy: A python package for solving a range of vehicle routing problems with a column generation approach. *Journal of Open Source Software*, 5(55):2408, 2020.

[23] Beatrice Ombuki-Berman, Brian Ross, and Franklin Hanshar. Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence*, 24:17–30, 02 2006.

[24] Mojtaba Rajabi-Bahaabadi, Afshin Shariat-Mohaymany, Mohsen Babaei, and Chang Wook Ahn. Multi-objective path finding in stochastic time-dependent road networks using non-dominated sorting genetic algorithm. *Expert Systems with Applications*, 42(12):5056–5064, 2015.

[25] Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520, 2014.

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

[27] Gaurav Srivastava, Alok Singh, and Rammohan Mallipeddi. Nsga-ii with objective-specific variation operators for multiobjective vehicle routing problem with time windows. *Expert Systems with Applications*, 176:114779, 2021.

[28] Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2002.

[29] Jin Wu and Shapour Azarm. Metrics for quality assessment of a multiobjective design optimization solution set. *Journal of Mechanical Design*, 123(1):18–25, 01 2000.

[30] Yiyong Xiao and Abdullah Konak. The heterogeneous green vehicle routing and scheduling problem with time-varying traffic congestion. *Transportation Research Part E: Logistics and Transportation Review*, 88:146–166, 2016.

[31] Eckart Zitzler, Lothar Thiele, Marco Laumanns, C.M. Fonseca, and Viviane Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *Evolutionary Computation, IEEE Transactions on*, 7:117 – 132, 05 2003.