Smart Ride and Delivery Services with Electric Vehicles: Leveraging Bidirectional Charging for Profit Optimisation

Jinchun Du[a,*], Bojie Shen[a], Muhammad Aamir Cheema[a], Adel N. Toosi[b]

[a]*Faculty of Information Technology, Monash University, Melbourne, 3800, Victoria, Australia*
[b]*School of Computing and Information Systems, University of Melbourne, Melbourne, 3052, Victoria, Australia*

Abstract

With the rising popularity of electric vehicles (EVs), modern service systems, such as ride-hailing delivery services, are increasingly integrating EVs into their operations. Unlike conventional vehicles, EVs often have a shorter driving range, necessitating careful consideration of charging when fulfilling requests. With recent advances in Vehicle-to-Grid (V2G) technology—allowing EVs to also discharge energy back to the grid—new opportunities and complexities emerge. We introduce the Electric Vehicle Orienteering Problem with V2G (EVOP-V2G): a profit-maximisation problem where EV drivers must select a subset of customer requests while managing when and where to charge or discharge. This involves navigating dynamic electricity prices, charging station selection, and route constraints. We formulate the problem as a Mixed Integer Programming (MIP) model and propose two near-optimal metaheuristic algorithms: one evolutionary (EA) and the other based on large neighborhood search (LNS). We compare these three algorithms with a greedy baseline on real-world data, showing that the proposed methods achieve up to twice the profit. V2G contributes about 20% of the total profit in the default settings. MIP finds optimal solutions for small cases (30 orders, 3 stations) but does not scale well. EA and LNS give near-optimal results for small cases and handle large ones (900 orders, 70 stations) efficiently. Our work highlights a promising path toward smarter, more profitable EV-based mobility systems that actively support the energy grid.

Keywords: Electric Vehicles, Vehicle-to-Grid, V2G, Orienteering Problem, Mixed Integer Programming, Evolutionary Algorithm, Large Neighborhood Search

1. Introduction

Achieving net zero emissions is a global goal, with transportation being a major source of greenhouse gas (GHG) emissions. According to a 2024 analysis [1], the transportation sector contributes about 16% of global greenhouse gas emissions, with road transport making up approximately 70% of that total. Electric vehicles (EVs) present a promising solution to mitigate these environmental impacts, leading to a projected decline in transport-related GHG emissions [2]. The adoption of EVs also supports several United Nations Sustainable Development Goals (SDGs), particularly SDG 7 (Affordable and Clean Energy), SDG 11 (Sustainable Cities and Communities), and SDG 13 (Climate Action), by fostering clean energy use, sustainable mobility, and climate resilience [3].

Beyond reducing emissions, EVs possess the potential to revolutionise the energy landscape through Vehicle-to-Grid (V2G) technology. V2G technology enables EVs to both store energy and transfer it back to the grid, homes, and other buildings. By enabling EVs to feed electricity back into the grid, V2G can optimise energy costs, enhance grid stability, and further contribute to emissions reduction. To fully harness the benefits of V2G and accelerate EV adoption, innovative approaches are required to seamlessly integrate travel and energy management for diverse EV users, including both private and commercial applications [4].

*Corresponding author.
Email addresses: jinchun.du@monash.edu (Jinchun Du), bojie.shen1@monash.edu (Bojie Shen), aamir.cheema@monash.edu (Muhammad Aamir Cheema), adel.toosi@unimelb.edu.au (Adel N. Toosi)

This research investigates profit maximisation for a single EV driver operating in commercial services like ride-hailing or delivery [5]. Specifically, we focus on a variation of the Electric Vehicle Orienteering Problem (EVOP) [6, 7] where a set of potential orders, each with specified pick-up and drop-off locations and time windows is given, and the EVOP seeks to determine the optimal subset of orders to accept and the sequence in which to complete them. The objective is to maximise profit while considering factors such as travel distance, energy consumption, and charging infrastructure. The EV must be able to complete all accepted orders within their respective time windows without depleting its battery, and charge the EV if and when required.

Previous studies on the EVOP have largely focused on optimising routes and energy consumption, without leveraging the potential of V2G technology to enhance profits. Although V2G presents significant opportunities for EV owners to increase revenue by purchasing energy at low prices and selling it back to the grid at higher prices, it also adds considerable complexity to the problem, which only a few previous works have investigated. Also, most existing studies have not only failed to consider the potential of V2G but also oversimplify the problem by assuming uniform charging rates and prices at all locations, which does not align with real-world conditions. For details, please refer to Section 2.

To the best of our knowledge, we are the first to formulate the Electric Vehicle Orienteering Problem with the integration of V2G technology to maximise profit. We call this the Electric Vehicle Orienteering Problem with V2G (EVOP-V2G). Fig. 1 demonstrates a simple instance of the EVOP-V2G problem where an EV owner with a work shift between 9:00 am to 5:00 pm has to maximise their profit. The order pool consists of 10 orders (o_1 to o_{10}), each with a time window (shown within square brackets) within which the order must be completed. For each order, the pick-up location (p) is represented by a dashed circle, while the drop-off location (d) is indicated by a solid circle. The battery icon represents the current state of charge (SoC) of the EV, indicating the amount of energy available. The clock symbol denotes the specific time at which the EV arrives at a given location. The figure shows one possible schedule for the EV owner where it leaves home at 9:00 am, completes o_2, o_3 and o_5 in this order before stopping at the charging station c_2 to discharge to 20% SoC as there is additional energy. Then, it processes the order o_7 which depletes the battery to 10%. It decides to visit charging station c_3 to charge to 80% before proceeding to complete o_9. After completing o_9, with the end of the work shift approaching, it returns home. With bidirectional charging available at home, the EV discharges its battery down to a user-defined threshold (in this case, 30%) when electricity prices are high, generating additional revenue. In total, the scheduled route yielded a profit of 63. Fig. 2a and Fig. 2b present alternative scheduled routes under varying operational conditions. In Fig. 2a, en route charging is unavailable; consequently, the driver cannot replenish the battery midday and must return to the depot after completing o_7. In Fig. 2b, discharging is unavailable both at charging stations and at home. Although the driver completes the same number of orders as in Fig. 1, the total profit is lower due to the loss of discharging revenue.

While the example above shows one possible schedule, the EVOP-V2G requires finding the schedule that maximises the profit. Finding a high-quality solution for EVOP-V2G is challenging because the algorithm must determine which subset of orders to serve and the sequence in which to complete them, while also considering charging and discharging between orders to maximise profit. Deciding which charging stations to use and when to charge or discharge, amidst different charging rates and prices, adds an additional layer of complexity.

To address these challenges, we first propose a Mixed Integer Programming (MIP) model that formulates the problem using mathematical equations. While the MIP model solves the problem optimally with off-the-shelf solvers, it suffers from poor scalability in large-scale instances. To mitigate this limitation, we design two suboptimal algorithms: Evolutionary Algorithm (EA) and Large Neighbourhood Search (LNS).

The EA initialises a population pool containing a large number of feasible solutions and relies on bio-inspired processes to iteratively evolve these solutions towards better quality. Although simple, the main drawback of EA is its requirement to maintain a large population pool, leading to slow convergence in solution quality. Conversely, LNS, a local search-inspired algorithm, overcomes this issue by maintaining a single solution and systematically exploring a broader range of neighbourhoods in the search space. We design an efficient LNS algorithm by designing advanced strategies to enhance its ability to effectively explore neighbourhoods and quickly find high-quality solutions.

To evaluate our proposed algorithms, we derive order requests and charging station information directly from real-world data sources. We compare the three proposed algorithms with a greedy baseline approach, demonstrating that our algorithms can produce solutions with twice the profit. The results show that V2G contributes approximately 20% of the total profit under our default settings. Among the three proposed algorithms, our MIP approach can optimally solve small-scale instances containing 30 orders and 3 charging stations, but it fails to scale for larger instances. In

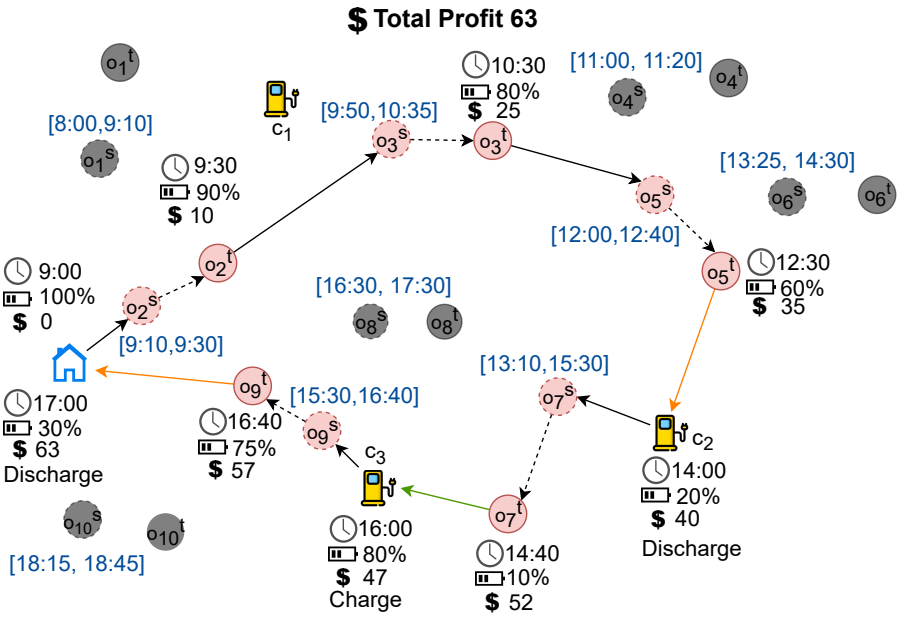


Figure 1: Representation of Electric Vehicle Orienteering Problem with V2G (EVOP-V2G) including 10 orders in order pool and 3 charging stations. Black arcs indicate order-serving travel, orange arcs indicate discharging, and green arcs indicate charging.

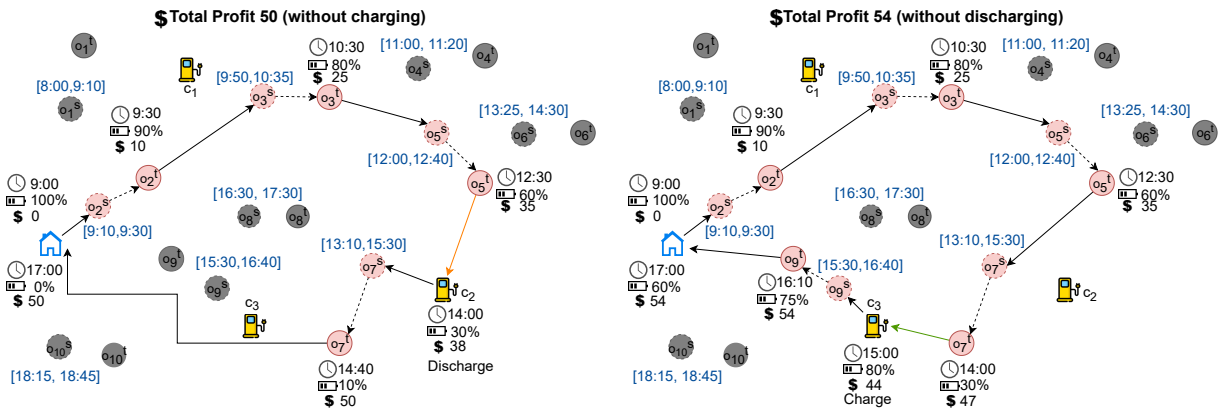
contrast, EA and LNS achieve near-optimal solutions for small-scale instances and effectively scale for larger instances with 900 orders and 70 charging stations. LNS demonstrates faster convergence rates than EA, quickly finding initial solutions and rapidly improving them to high quality, outperforming EA in most instances.

Our contributions in this paper are summarized below:

- We formulate a novel optimisation problem, EVOP-V2G, to maximise the profit of EV drivers in commercial settings such as ride-hailing. EVOP-V2G assumes a centralised server that broadcasts available orders for EV drivers, with each order associated with a monetary reward (e.g., ride fare). The objective of EVOP-V2G is to maximise the overall profit for the EV driver by optimising both the selection of orders and the charging/discharging activities.
- We generate and release a curated benchmark suite of order instances spanning multiple variation settings, derived from a real-world dataset. We vary spatial demand (clustered vs. dispersed orders), temporal structure for order requests (tight vs. broad time windows), and varying ride-length to impose complementary spatial, scheduling, and energy stressors. This benchmark suite enables reproducible evaluation and future comparisons on EVOP–V2G.[1]
- We conduct extensive experiments using a real-world dataset that includes realistic charging/discharging rates and prices, our customer order benchmark suite, fare prices, and charging station information. We quantify how V2G participation and infrastructure availability affect driver profit and solution structure, and we report scalability/solution-quality trade-offs across exact MIP model and two suboptimal heuristic methods, EA and LNS. These results serve as empirical guidance for practitioners and as baselines for subsequent research.

The structure of the paper is as follows: Section 2 reviews related work. Section 3 formalises the problem and describes its key aspects. We also include the mathematical formulation of our proposed MIP model. Section 4 details our proposed methods: EA, and LNS. Sections 5 and 6 present the experimental results and conclusions, respectively.

[1] `https://github.com/goldi1027/evop_benchmark`



(a) EVOP-V2G without en route charging

(b) EVOP-V2G without en route discharging

Figure 2: Alternative EVOP–V2G scenarios. Black arcs denote order-serving travel; orange arcs denote discharging segments; green arcs denote charging segments.

2. Related Work

This section situates EVOP–V2G within prior research. We review EV routing and orienteering formulations for both single vehicle and fleet settings, covering heterogeneous charging models, time-window constraints, and V2G participation. We highlight gaps at their intersection that motivate our formulation and evaluation.

2.1. Electric Vehicle Routing Problem

The Vehicle Routing Problem (VRP), introduced in 1959 [8], is an optimisation problem frequently encountered in logistics and has been extensively studied over the decades. In VRP, a fleet of vehicles is tasked with serving a set of customers. The primary objective of the VRP is to determine optimal routes for each vehicle, aiming to minimise overall transportation costs or maximise operational efficiency. A recent survey by [9] provides a comprehensive review of the VRP literature, focusing on a wide range of problem variants and associated solution methods. The authors introduce a taxonomic framework that classifies VRP models into four broad categories: classical problems, problems with additional decisions, integrated problems, and recent or emerging variants. The survey offers a structured account of the field's evolution, covering well-established variants such as the Capacitated VRP (CVRP) [10], as well as extensions involving dial-a-ride [11], and multi-echelon routing where distribution of goods are organised through multiple layers [12]. Particular attention is given to emerging domain-specific variants, notably the Electric Vehicle Routing Problem (EVRP), which addresses the challenges of limited battery capacity and the need for recharging during operations. Building on this foundation, we present a focused overview of recent literature on EVRP, highlighting its core modelling features and the diverse solution approaches proposed in the past few years.

2.1.1. Single EV

Existing EVRP research has extensively studied this problem by exploring numerous variations (see [13] for a survey), however, the primary focus remains on fleets of EVs due to their critical role in urban logistics applications. In the context of a single EV in EVRP, the EV must depart from and return to the same depot, while visiting each customer exactly once. The primary objective of EVRP is to minimise energy consumption after servicing all customers, taking into account charging requirements. Several studies on single EVRP have focused on optimising the routes of individual EVs while addressing different operational and logistical aspects such as non-linear charging [14] and the incorporation of multi-depot [15].

2.1.2. Fleets of EVs

A fleet of EVs consist of multiple EVs, each of which requires departure and return to the same depot. While the EV fleet comprises more EVs, the objective of the EVRP remains same, i.e., the EVRP plans routes, one for each EV,

to collectively meet the requirement of visiting each customer once while minimising the energy consumption of all EVs. Managing EV fleets models the practical urban logistics scenarios [16], yet introduces additional complexities related to vehicle coordination and energy constraints. Fleets are generally categorised as homogeneous, where all vehicles share the same specifications [17], or heterogeneous, where vehicles differ in parameters such as capacity or range [18]. The latter remains less explored due to its increased computational complexity. In addition to that, EVRP studies have further examined diverse charging strategies, including full charging, where vehicles recharge to full capacity at charging stations [15]; partial charging, which allows flexible energy replenishment [19]; and battery swapping, where depleted batteries are exchanged for fully charged units [20]. Time window constraints have also been considered [21], requiring service at customers within predefined intervals and significantly increasing routing complexity.

To address these complexities, solution approaches for EVRP for both single and fleet are broadly categorised into exact and heuristic methods. Exact methods employ mathematical formulations and utilise off-the-shelf solvers such as Gurobi or Cplex. Despite advancements like branch, cut, and price methods [22], which further enhance efficiency, exact methods often suffer from scalability issues and are typically suitable only for smaller-scale instances. On the other hand, heuristic approaches commonly employ techniques such as local search [23], genetic algorithms [24], and others. While these heuristics are better suited for larger-scale instances, they often trade off runtime efficiency for solution quality. This often leads to suboptimal solutions, with the quality of the solution dependent on the design of the heuristic approaches.

In recent years, researchers have increasingly explored reinforcement learning (RL) for EVRP and related variants, using model-free and deep RL to learn routing and charging policies [25]. These learning-based methods offer fast inference once trained and adapt well to dynamic, real-time settings; however, they typically require substantial training data and careful reward/hyperparameter tuning, and they do not provide optimality guarantees. Hybrid metaheuristics form another line of work, combining multiple algorithms or embedding learning within heuristics to offset the limitations of single methods in solution quality and robustness. For example, adaptive simulated annealing (SA) has been integrated with RL to tackle the capacitated EVRP [26]. These hybrids aim to balance diversification and intensification, often yielding stronger performance than standalone metaheuristics. However, hybrid approaches also present several challenges. Each combined heuristic introduces its own set of parameters, making parameter tuning and sensitivity a significant concern. Additionally, these methods often suffer from increased computational overhead and scalability issues, as running multiple algorithms in tandem can be resource-intensive. In practice, hybrid metaheuristics typically require more iterations or involve computationally expensive operations compared to standalone approaches.

2.2. Electric Vehicle Orienteering Problem

Differing from the VRP, where each customer must be visited exactly once, the Orienteering Problem (OP) aims to maximise total profit by visiting a subset of customers within time or distance constraints. A related variant, Vehicle Tour Planning (VTP), models tourist attractions instead of customers, assigning each a score to be maximised [27]. Similar to EVRP, the Electric Vehicle Orienteering Problem (EVOP) has emerged recently which extends OP by incorporating EV-specific constraints, including charging needs, and aims to maximise profit by selecting an optimal subset of customers while accounting for energy limitations.

2.2.1. Single EV

For the single EV, EVOP assumes that the EV must both start and end its journey at the same depot. Each customer is associated with a profit value and can be distributed arbitrarily across the map. EVOP typically necessitates users to input constraints such as the time or distance that an EV can travel. By factoring in the locations of charging stations and the duration required for charging, EVOP selects a subset of customers that can maximise profit and plan an optimal route that satisfies the input constraint. Studies have explored various charging strategies in EVOP. For example, [28] assumes full recharging at each station, while [29] incorporates waiting times before charging begins. Several EVOP variants introduce additional constraints, such as time windows [30], requiring visits within specified intervals, and range anxiety [7], which emphasises maintaining a higher average state-of-charge due to limited charging infrastructure.

Literature	Problem Type	Problem Characteristics					
		Single EV	Fleet of EV	Time Window	Charging	Discharging	Time-variant prices
Abdulaal et al. [15]	EVRP	✓		✓	Full	✓	✓
Narayanan et al. [33]	EVRP		✓	✓	N/A	✓	✓
Lin et al. [34]	EVRP		✓	✓	Partial	✓	✓
Lee et al. [29]	EVOP	✓			Partial		
Karbowska-Chilinska and Chociej [28]	EVOP	✓			Full		
Wang et al. [30]	EVOP	✓		✓	Battery swap		
Chen et al. [7]	EVOP	✓		✓	Partial		
Our work (EVOP-V2G)	EVOP	✓		✓	Partial	✓	✓

Table 1: Classification of the relevant literature based on problem characteristics closely related to our work (EVOP-V2G).

2.2.2. Fleets of EVs

For a fleet of EVs that contains multiple EVs, the objective of EVOP remains consistent, i.e., EVOP plans an individual route for each vehicle, aimed at maximising the collected profit from each customer collectively. Each vehicle is required to depart from and return to the same depot, while ensuring that each customer is serviced by only one vehicle. There is only limited number of studies focus on the EVOP for a fleet of EV [6]. Additionally, the EVOP has been extended in various scenarios, including surveillance operations [31] and logistical operations in smart cities [32]. These scenarios underscore the necessity of managing limited resources effectively, often necessitating the deployment of a team or fleet of EVs and Unmanned Aerial Vehicles (UAVs) to fulfil task requirements.

2.3. Discussion

Our work differs significantly from prior studies in several important ways. Firstly with advances in battery technology and a growing focus on renewable energy, there is increasing interest in repurposing idle EV batteries for energy storage through V2G systems, which enable bidirectional energy flow between EVs and grid/building infrastructure [35]. V2G adds extra charge–discharge cycles beyond normal driving, accelerating lithium-ion battery wear. Studies show additional V2G cycling shortens useful battery life [36]. For example, one V2G cycle every 1–2 days increases total degradation by 0.31% per year (approximately 9–14% more capacity fade over 10 years) relative to a vehicle not doing V2G. Another challenge is distribution-network stress: clusters of V2G-enabled EVs charging or discharging at the same time can create large coincident power flows at feeder nodes, overloading lines or transformers [37]. Balanced against these risks are substantial system-level benefits: when coordinated through smart scheduling and grid-aware controls, V2G can provide ancillary services (frequency regulation, reserves, voltage support), enable peak shaving and load shifting, and capture price arbitrage by charging when prices are low and discharging when they are high thus improving operational flexibility, supporting renewable integration by absorbing surplus and supplying during deficits, and creating potential revenue streams for EV owners and aggregators [38, 39]. These challenges and opportunities motivate a planning framework that jointly considers routing/fulfilment and energy decisions.

Secondly, while our problem shares some similarities with EVOP for a single EV, it is considerably more sophisticated and fundamentally different. Specifically, our problem integrates charging (or discharging) as a deduction (or addition) to the total profit collected, with charging/discharging prices fluctuating throughout the day (also known as time-variant prices). Hence, the strategic timing of charging or discharging operations also plays a crucial role in our problem. Currently, only a handful of EVRP studies, such as [15, 33], consider time-variant pricing at charging stations, and to the best of our knowledge, no EVOP problems have addressed this aspect. Moreover, without loss of generality, our problem accounts for the different charging rates at different stations during EV charging, leading to varying charging duration across stations. Consequently, the decision-making process regarding which station to charge at becomes pivotal.

Recall that EVOP differs from EVRP in its routing aspects: EVRP designs routes that serve all required customers while minimising cost, subject to EV-specific constraints such as battery limits and visits to charging stations. By contrast, EVOP selects a subset of customers under time/distance limits and maximises total profit, trading off coverage for value. Although a few EVRP studies have incorporated V2G (e.g., [15, 33, 34]), their scope is limited. For instance, [33] considers only the discharging phase of V2G, assuming EVs begin fully charged and do not recharge en route. To the best of our knowledge, V2G integration has not been explored within the EVOP context. In this

paper, we address this gap by incorporating V2G into EV route planning for order fulfilment, enabling vehicles to both charge and discharge strategically to maximise user profit while efficiently serving diverse customer orders.

To better differentiate our work from related studies, Table 1 outlines the most closely related works to our problem, highlighting the key problem characteristics. Compared to related studies, our problem setting is more generalised where our work allows EVs to depart from and end at different locations, supports partial charging, incorporates dynamic pricing, leverages V2G, and includes time windows for orders.

3. Problem Description

This section formalises EVOP–V2G. We specify assumptions, notation, decision variables, and constraints, and present an exact MIP model that jointly optimises order selection and charging/discharging to maximise driver profit.

3.1. Problem Formulation

To formulate EVOP-V2G, we construct a directed query graph $G = (V_{c,o,s,d}, E, W)$. Here, $V_{c,o,s,d}$ denotes the combined set of vertices[2], comprising charging locations (V_c), service orders (V_o), as well as the source (V_s) and destination (V_d) locations for the EV driver. The edge set E connects each vertex in $V_{c,o,s,d}$ to every other vertex to create a complete graph, i.e., $E = V_{c,o,s,d} \times V_{c,o,s,d}$. The weight function $w \in W$ assigns each edge $e_{ij} \in E$ a tuple (d_{ij}, t_{ij}), where d_{ij} and t_{ij} represent the travel distance and time from vertex v_i to v_j, respectively.

The EV-specific information is often provided by the EV driver which includes the battery capacity B (in kWh) and driving efficiency γ (in kWh per km). Given the driving efficiency γ, we can compute the energy consumed along an edge with edge weight (d_{ij}, t_{ij}) as $e_{ij} = \gamma \times d_{ij}$. In our formulation, energy consumption is modelled at the edge level of the graph G. Each edge $e_{ij} \in E$ represents a travel segment and is assigned an energy consumption value. While we adopt a linear model $e_{ij} = \gamma \times d_{ij}$ for clarity and computational simplicity, the optimisation framework itself does not rely on this linear assumption. The energy consumption term appears only as an attribute of each edge and can be computed by any energy-consumption function, e.g., one that accounts for vehicle speed, accessory loads, temperature, or non-linear state-of-change effects. Substituting a more sophisticated or data-driven model requires only updating the mapping $e_{ij} = f(v_i, v_j)$, without modifying the graph structure, constraints, or solution algorithm. This design choice ensures that the proposed EVOP-V2G formulation is modular and robust to future improvements in energy modelling. Our proposed techniques can thus easily integrate more sophisticated and accurate energy consumption estimation models if required.

The services system contains a set of order V_o that are available for the EV driver to choose from. Each order $v_i \in V_o$ is represented as a tuple $(l_i^p, l_i^d, p_i, d_i, t_i, [\tau_i^b, \tau_i^e])$, where l_i^p represents the pick up location and l_i^d represents the drop off location, p_i denotes the profit earned for completing the order v_i, d_i and t_i represent the distance and time, respectively, of v_i when traveling between l_i^p and l_i^d, and $[\tau_i^b, \tau_i^e]$ indicates the time window within which an EV does not arrive earlier for pick-up than τ_i^b and completes the drop-off no later than τ_i^e. Note that, when calculating the travel distance/time from an order vertex v_i to another vertex in the graph G, we consider the EV departing from the drop-off location l_i^d. Similarly, when computing the travel distance/time from a vertex to an order vertex v_i, we consider the EV reaching the pick-up location l_i^p.

For each charging station $v_i \in V_c$, P_i is given to denote the charging/discharging rate in kW. The planning horizon of charging/discharging is discretised into $|T|$ number of disjoint, consecutive timeslots, $T = \{t_0, \cdots, t_t\}$, each of which has the length δ. We assume the charging/discharging price are represented as piece-wise constant function at each charging station v_i. $P_i^{Bt_k}$ denotes the charging price at v_i (in \$ per kWh) at timeslot $t_k \in T$, while $P_i^{St_k}$ is the discharging price at v_i (in \$ per kWh) at timeslot $t_k \in T$. By default, the planning horizon T is set to 24 hours to accommodate EV charging/discharging at both the source and destination locations. Table 2 summarises the notation used throughout this paper. Using this specified notation, we formulate the EVOP-V2G query as follows:

Definition 1. ***EVOP-V2G Query****: A query is defined as a tuple:*

$$Q = (V_s, V_d, B, [B^s, B^d], [\tau_w^b, \tau_w^e]),$$

[2]In this paper, we often use $V_{x,y}$ to denote the union of two sets of vertices V_x and V_y, i.e., $V_{x,y} = V_x \cup V_y$.

Table 2: Summary of Notation

Symbol	Definition
G	A complete and directed graph.
E	A set of edges in G.
V_o	A set of vertices each of which associate with an order.
V_c	A set of vertices each of which associate with a charging station.
V_s	The source location of EV.
V_d	The destination location of EV.
T	The planning horizon T of charging/discharging at station.
δ	The length of each timeslots in the planning horizon T.
B	The battery capacity of the EV.
γ	The driving efficiency of the EV.
P_i	The charging rate at charging station $v_i \in V_c$.
$P_i^{Ct_k}$	The charging price at charging station $v_i \in V_c$ at time period t_k.
$P_i^{Dt_k}$	The discharging price at charging station $v_i \in V_c$ at time period t_k.
l_i^p	The pick up location of an order $v_i \in V_o$.
l_i^d	The drop off location of an order $v_i \in V_o$.
d_i	The travel distance between l_i^p and l_i^d when completing order $v_i \in V_o$.
t_i	The travel time between l_i^p and l_i^d when completing order $v_i \in V_o$.
p_i	The profit for an order $v_i \in V_o$.
$[\tau_i^b, \tau_i^e]$	The time window for order $v_i \in V_o$ starts at τ_i^b and ends before τ_i^e.
$[\tau_w^s, \tau_w^e]$	The working hours of an EV driver starts at τ_w^s and ends before τ_w^e.
$[B^s, B^d]$	The initial battery level B^s at source, and minimal battery required B^d at destination.

where an EV driver starts from source location $V_s \in V$ with initial battery level B^s, and must reach destination $V_d \in V$ with at least B^d battery remaining. The EV has a total battery capacity of B and operates within a working hour window $[\tau_w^b, \tau_w^e]$, during which it may serve orders ($v_o \in V_o$) or charge/discharge at stations ($v_c \in V_c$). Outside this window, only charging/discharging at V_s and V_d is allowed.

The output is a sequence of actions (serving orders or charging/discharging) aligned with a discretised planning horizon T. EVOP-V2G evaluates orders from the order pool O and selects a subset that optimally balances route planning and charging/discharging schedules. The objective is to maximise the profit potential for EV drivers operating within predefined working hours $[\tau_w^b, \tau_w^e]$.

$$\max \sum_{v_i \in V_o} \sum_{v_j \in V_{o,c,d}} x_{ij} p_i + \sum_{v_i \in V_c} \sum_{t_k \in T} \left(dc_i^{t_k} P_i^{Dt_k} - rc_i^{t_k} P_i^{Ct_k} \right) \tag{1}$$

$$\text{s.t.} \quad \sum_{v_i \in V_{o,c,d}} x_{ji} = 1, \qquad v_j \in V_s \tag{2}$$

$$\sum_{v_i \in V_{o,c,s}} x_{ij} = 1, \qquad v_j \in V_d \tag{3}$$

$$\sum_{v_i \in V_{o,c,s}} x_{ij} \leq 1, \qquad \forall v_j \in V_o \tag{4}$$

$$\sum_{v_i \in V_{o,c,s}} x_{ij} - \sum_{v_i \in V_{o,c,d}} x_{ji} = 0, \qquad \forall v_j \in V_{o,c} \tag{5}$$

$$\tau_i = T[0], \qquad v_i \in V_s \tag{6}$$

$$\tau_i = T[|T| - 1], \qquad v_i \in V_d \tag{7}$$

$$\tau_w^b \leq \tau_i \leq \tau_w^e, \qquad \forall v_i \in V_{o,c} \tag{8}$$

$$\tau_i^b \leq \tau_i \leq \tau_i^e, \qquad \forall v_i \in V_o \tag{9}$$

$$\tau_i + (t_{ij} + t_i) x_{ij} - M(1 - x_{ij}) \leq \tau_j, \qquad \forall v_i \in V_{o,c,s}, \forall v_j \in V_{o,c,d} \tag{10}$$

$$(k+1)\delta(rc_i^{t_k} + dc_i^{t_k}) + t_{ij} x_{ij} - M(1 - x_{ij}) \leq \tau_j, \qquad \forall v_i \in V_c, \forall v_j \in V_{o,c,d}, \forall t_k \in T \tag{11}$$

$$\tau_i - k\delta \leq M(1 - rc_i^{t_k} - dc_i^{t_k}), \qquad \forall v_i \in V_c, \forall t_k \in T \tag{12}$$

$$(k+1)\delta(rc_i^{t_k} + dc_i^{t_k}) \leq \tau^d, \qquad \forall v_i \in V_c, \forall t_k \in T \tag{13}$$

$$b_i = B^s, \qquad v_i \in V_s \tag{14}$$

$$b_i \geq B^d, \qquad v_i \in V_d \tag{15}$$

$$b_j \leq b_i - \frac{(d_{ij} + d_i^s)\gamma}{\lambda} x_{ij} + M(1 - x_{ij}), \qquad \forall v_i \in V_{o,s}, \forall v_j \in V_{o,c,d} \tag{16}$$

$$b_j \leq b_i + \sum_{t_k \in T} \delta rc_i^{t_k} - \sum_{t_k \in T} \delta dc_i^{t_k} - \frac{d_{ij}\gamma}{\lambda} x_{ij} + M(1 - x_{ij}), \qquad \forall v_i \in V_c, \forall v_j \in V_{o,c,d} \tag{17}$$

$$rc_i^{t_k} + dc_i^{t_k} \leq 1, \qquad \forall v_i \in V_c, \forall t_k \in T \tag{18}$$

$$\sum_{t \in T} \delta rc_i^t \leq B - b_i, \qquad \forall v_i \in V_c \tag{19}$$

$$\sum_{t \in T} \delta dc_i^t \leq b_i, \qquad \forall v_i \in V_c \tag{20}$$

$$0 \leq b_j \leq B \sum_{v_i \in V_{o,c,s}} x_{ij}, \qquad \forall v_j \in V_c \tag{21}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall v_i \in V_{o,c,s}, \forall v_j \in V_{o,c,d} \tag{22}$$

$$rc_i^{t_k}, dc_j^{t_k} \in \{0, 1\}, \qquad \forall v_i \in V_c, t_k \in T \tag{23}$$

3.2. Mixed-Integer Programming (MIP) Model

In this subsection, we introduce MIP model to find optimal solution for EVOP-V2G. Given the input graph G, we first introduce the binary and continue decision variable as follow:

- The binary decision variables are:
  $x_{ij} = \begin{cases} 1 & \text{if edge } (v_i, v_j) \in E \text{ is traveled by an EV,} \\ 0 & \text{otherwise,} \end{cases}$
  $rc_i^{t_k} = \begin{cases} 1 & \text{if the EV is recharging at } v_i \in V_c \text{ at timeslot } t_k \\ 0 & \text{otherwise,} \end{cases}$
  $dc_i^{t_k} = \begin{cases} 1 & \text{if the EV is discharging at } v_i \in V_c \text{ at timeslot } t_k \\ 0 & \text{otherwise,} \end{cases}$

- The continuous decision variables are:
  - τ_i : arrival time of EV at node v_i,
  - b_i : remaining energy upon arrival to node v_i

The binary decision variable x_{ij} indicates whether an EV travels between vertex v_i and v_j or not. When $v_j \in V_o$ is an order vertex, the binary decision variable $x_{ij} = 1$ indicates that the EV driver will serve the order v_j at pick-up location l_j^p. Conversely, when $v_i \in V_o$, $x_{ij} = 1$ indicates that the EV driver will complete the order v_i at drop-off location l_i^d. Additionally, we restrict that charging and discharging of EVs cannot occur simultaneously at the same timeslot t_k, this constraint will be imposed later on. Given the binary and continuous decision variable, the objective function is defined as Equation (1). The objective function is designed to maximise the total profit collected for EV drivers, as well as the benefits received from discharging energy back to the grid through V2G technology. This sum of profit is then subtracted by the costs associated with charging the EV at the stations. Next, we define the constraints for our MIP model.

3.2.1. Constraints of MIP model

The constraints (2)–(23) comprise the full set of constraints in the MIP model. These constraints are classified into three categories: (i) node-visiting constraints; (ii) travel-time constraints; and (iii) EV-battery constraints. The constraints (21)–(23) specify the variable types and bounds. Next, we provide a brief discussion of the constraints in each category:

Node Visiting Constraints: Constraints (2) to (5) enumerate the node-visiting constraints. Specifically, constraints (2) and (3) ensure that the EV traverses exactly one edge from the source vertex (or other vertices) to other vertices (or the destination vertex), thereby enforcing departure from the source and arrival at the destination location. Constraint (4) states that each order in V_o may be visited at most once, preventing the EV from revisiting the same order multiple times. Constraint (5) ensures that if an EV travels to a vertex v_j that is an order or charging station, it must leave afterwards, thereby implying $x_{ij} - x_{ji} = 0$. Collectively, constraints (2) to (5) constrain the EV to traverse a complete path from source V_s to destination V_d, and to serve the order $v_o \in V_o$ or engage in charging/discharging at a station $v_c \in V_c$ in between. Subtours are essentially eliminated because constraint (5) requires that if an EV visits a vertex, it must depart afterwards. Collectively, constraints (2)–(4) ensure that each vertex can be visited at most once by the EV. Together, these make it impossible to create cycles or subtours within a complete path.

Our MIP model permits multiple visits to each charging station, including those at the source and destination, by introducing additional dummy vertices[3] for each charging station. However, this significantly increases the computational cost, as demonstrated by [40]. Therefore, we create these dummy vertices only when required for specific instances (further details are provided in Section 5).

Travel Time Constraints: Constraints (6)–(13) cover the travel-time constraints for visits to each vertex. Specifically, constraints (6) and (7) set the arrival times at the source V_s and destination V_d to $T[0]$ and $T[|T|-1]$, respectively, aligning with the start and end times of the planning horizon T. Constraint (8) ensures that an EV driver can visit orders in V_o and charging stations in V_c only during working hours $[\tau_w^b, \tau_w^e]$. Constraint (9) enforces that the arrival at each order $v_i \in V_o$ occurs within its time window $[\tau_i^b, \tau_i^e]$, ensuring the EV driver reaches the pick-up location l_i^p and picks up the customer on time. Constraint (10) ensures that travel between vertices v_i and v_j accounts for the travel time of edge (v_i, v_j) and, if $v_i \in V_o$ is an order, that the arrival time at v_j allows sufficient time to complete the order. Note that M denotes the big-M constant [41]. The big-M notation incorporates a large constant, M, into constraints to activate or deactivate them based on the value of a binary variable, ensuring that a constraint is enforced only when required; for example, constraint (10) is valid only if the edge (v_i, v_j) is travelled (i.e., $x_{ij} = 1$).

Similarly, Constraint (11) ensures that the EV driver has adequate time to complete charging/discharging at the charging station $v_i \in V_c$ before reaching the next vertex v_j. Therefore, for each time step $t_k \in T$, the constraint verifies that, if the EV completes its charging or discharging (i.e., $(k+1)\delta(rc^{t_k} i + dc^{t_k} i)$), it has sufficient time to travel to the next vertex v_j. Constraint (12) ensures that charging/discharging at charging stations respects practical constraints; that is, the EV driver cannot charge/discharge at a charging station before arriving. Finally, Constraint (13) ensures that charging/discharging operations are completed before the working hours of an EV driver. Recall that we include two dummy vertices in V_c to represent charging stations at the source and destination. Therefore, Constraints (8) and (13) are relaxed for these two dummy vertices, allowing charging/discharging outside working hours.

EV Battery Constraints: Constraints (14)–(20) specify the battery constraints for the EV. At the start of the day, Constraint (14) initialises the EV's battery level at V_s to B^s, allowing the EV to charge or discharge before working hours begin. Similarly, Constraint (15) requires the battery level at V_d to be at least B^d. The EV may arrive with a lower battery level during the day, but it must have at least B^d at the end of the day. Constraint (16) monitors the battery level of an EV when travelling between vertices v_i and v_j, ensuring that the battery level decreases according to the travel distance d_{ij}. Additionally, if v_i is an order vertex, the constraint ensures that the battery level accounts for the travel distance d_l^s when serving order v_i. Similarly, Constraint (17) accounts for charging $rc^{t_k} i$ (and, respectively, discharging $dc^{t_k} i$) at the charging station $v_i \in V_c$ and ensures that the battery level increases (respectively, decreases) accordingly. Constraints (18)–(20) constrain charging and discharging at a charging station $v_i \in V_c$. Constraint (18) ensures that, when an EV arrives at v_i, charging and discharging cannot occur simultaneously. When the EV starts charging, Constraint (19) ensures that the battery level does not exceed the capacity B. Conversely, when the EV starts discharging, Constraint (20) ensures that the amount discharged does not exceed the current battery level b_i upon reaching v_i.

[3]The dummy vertex v_c' essentially replicates the vertex $v_c \in V_c$ in the graph G.

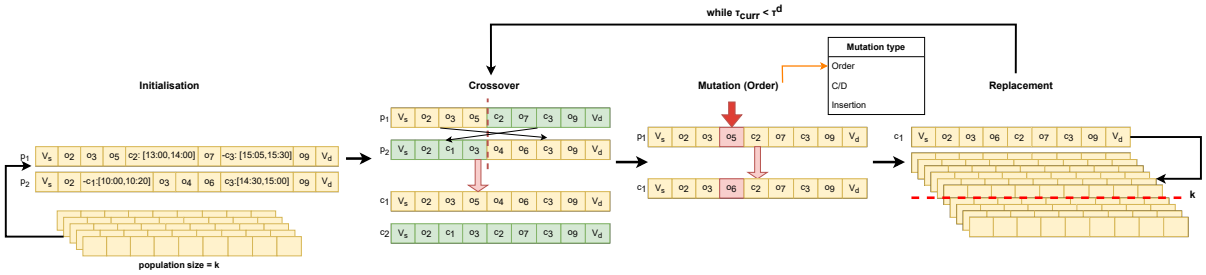


Figure 3: Illustration of the EA process, showing the stages of initialisation, crossover, mutation, and replacement.

Algorithm 1: Evolutionary Algorithm (High-Level)

Input: EVOP-V2G Instance; Fitness Function $f(.)$.
Parameters: Max Generations G; Time Limit T; Population Size k; Mutation Rate p_m.
Output: S_{best}

Initialise // see Sec. 4.1.1
    $P \leftarrow$ Initializing population P with k number of chromosome; // see Algorithm 2
    $S_{best} \leftarrow$ best individual in P evaluated by fitness f;
    generation $\leftarrow 0$;
while not reached termination (generation $\geq G$ or time limit $\geq T$) **do**
    Selection: $P' \leftarrow$ perform tournament selection to select individuals from P; // see Sec. 4.1.2
    Crossover: $O \leftarrow$ perform crossover to generate offsprings O from P';
    Mutation: $O \leftarrow$ Mutate(O, mutation strategy, p_m) ; // see Sec. 4.1.3
    Replacement: $P \leftarrow$ select the top-k individuals from $(P' \cup O)$; // see Sec. 4.1.4
    if $f(\text{best}(P)) < f(\text{bestSolution})$ **then**
        bestSolution $\leftarrow$ best(P);
    generation ++;
return S_{best}

4. Proposed Solutions

In this section, we present two novel algorithms for solving EVOP-V2G. We formulated the problem as a MIP model to achieve optimal solutions in Section 3. However, the MIP model suffers from computational burden and struggles with scalability. To address this, we propose two suboptimal algorithms: the Evolutionary Algorithm (EA), a bio-inspired method that iteratively evolves solutions, and Large Neighbourhood Search (LNS), a local search-based approach that efficiently explores the search space to find solutions.

4.1. Evolutionary Algorithm

Similar to the NP-hard EVRP and EVOP problems [13], our problem becomes more complex with the inclusion of V2G capabilities and dynamic charging/discharging prices. While MIP models can be solved by off-the-shelf solvers such as Gurobi or CPLEX, they face two main limitations: (i) they are computationally intensive and do not scale well, as shown in our experiments; (ii) they track the arrival time at each vertex, thereby restricting each charging station to a single visit by default. However, in certain scenarios, revisiting a charging station can be beneficial, particularly if it offers the lowest charging cost. In our experiments, we observed that enabling a single dummy vertex for a charging station can, in some cases, increase computational time exponentially.

To address these issues, we explore EA, a meta-heuristic inspired by natural selection, that iteratively evolves feasible solutions. EA has been successfully applied to various combinatorial problems, including EVRP. We propose EA-based algorithm in order to efficiently solve EVOP-V2G. Specifically, EA defines two important components: (i) Chromosomes, which maintains the feasible solution of the problem. (ii) Fitness Function, which evaluate the solution quality of the chromosomes. Algorithm 1 presents the pseudocode of the proposed EA for solving EVOP-V2G. Fig. 3

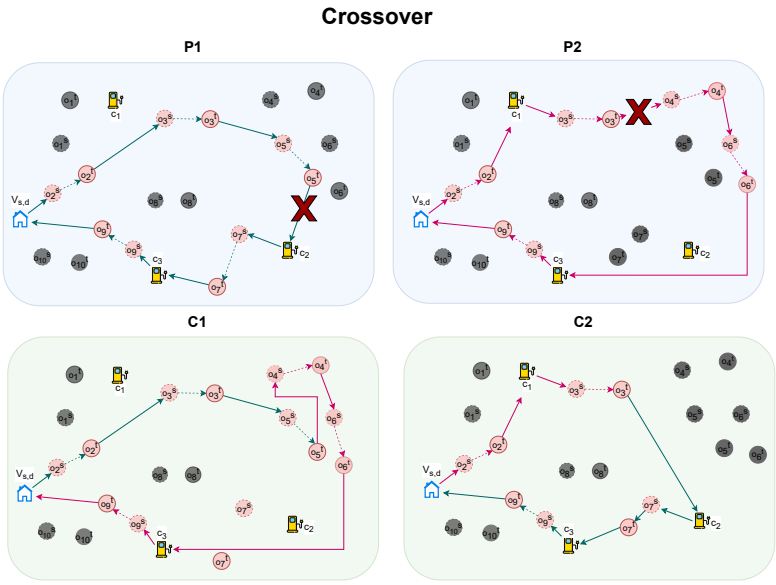


Figure 4: Representation of crossover operations. Parents p_1 and p_2 are shown in blue boxes; their routes are drawn in blue and pink, respectively. The red cross marks the crossover point. Offspring c_1 and c_2 (green boxes) inherit route segments from both parents from the crossover point onward.

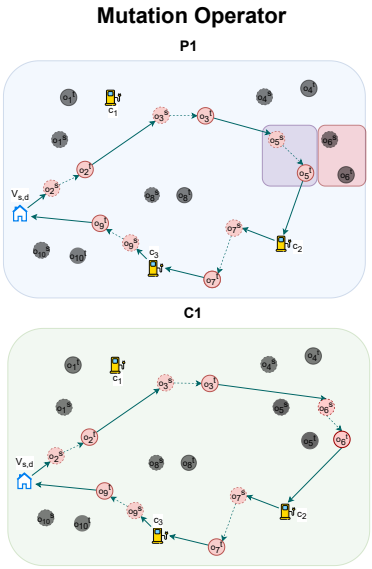


Figure 5: Representation of mutation operation. In chromosome p_1, the selected order o_5(purple outline) is swapped with order o_6 (red outline). The resulting c_1 after mutation is shown below in a green box.

illustrates the process of the EA, providing an overview of its key steps and mechanisms. In board stoke, EA can be outlined as follows:

1. **Initialisation**: Given the Chromosomes and Fitness Function defined, the initialisation phase generates a pool of chromosomes, known as the population, which are then evaluated using the fitness function.
2. **Selection & Crossover**: The selection & crossover process simulates genetic exchange. Initially, the selection process chooses a pair of chromosomes from the population. Subsequently, the crossover selects a portion of the chromosome from each parent and exchanges them to produce new solutions, called offspring. The feasible offspring is then inserted back into the population pool. Fig. 4 illustrates the crossover operation described in Example 1.
3. **Mutation**: For the chromosomes in the population, the mutation simulate genetic mutation by introducing random changes. In Fig. 5, an example chromosome demonstrating order mutation is shown and explained in detail in Example 1.
4. **Replacement**: The steps 2 - 3 repeats until the chromosomes in the population have undergone crossover and mutation (given the probability). The replacement process maintains the population size by removing chromosomes with lower fitness values, ensuring that the population remains at the predefined size k for the next generation.

Steps 2–4 are often referred to as the evolutionary phases. Generally, EA maintains a large population of chromosomes, ranging from hundreds to thousands, depending on the application. To solve a given problem, EA runs the evolutionary phases for a specified number of iterations or terminates based on a predefined cut-off time. The best solution in the population is then returned as the final result.

4.1.1. Initialisation:

For EA, an important phase involves designing an appropriate chromosome to represent a feasible solution to EVOP–V2G. To formulate an EA-based algorithm for EVOP–V2G, we define the chromosome as follows:

Algorithm 2: Generating Chromosome

Input: $[\tau_w^s, \tau_w^d]$: EV driver's working hours, starting at τ_w^s and ending at τ_w^d. $[B^s, B^d]$: Initial battery level B^s at source, and minimum battery required B^d at destination.

Initialisation: $\tau_{curr} = \tau^s$, $B_{curr} = B^s$, $chromosome = \emptyset$

1 V_o' = retrieve orders from V_o within working hour $[\tau_w^s, \tau_w^d]$;
2 append action (V_s) to C
3 **while** $\tau_{curr} < \tau^d$ **do**
4 v_i = randomly select an order from V_o';
5 **if** v_i is feasible under (τ_{curr}, B_{curr}) **then**
6 append action (v_i) to C and update (τ_{curr}, B_{curr});
7 remove order v_i from V_o';
8 append action (V_d) to C
9 $\tau_{curr} = \tau^s$, $B_{curr} = B^s$;
10 **for** each pair of order $(v_i, v_j) \in C$ **do**
11 update (τ_{curr}, B_{curr}) according to order v_i;
12 **for** i = 0 to randomNumber(0, m)-1 **do**
13 $(v_c, [\tau_{c_b}, \tau_{c_e}])$ = generate charging/discharging action;
14 **if** $(v_c, [\tau_{c_b}, \tau_{c_e}])$ is feasible under (τ_{curr}, B_{curr}) **then**
15 insert action $(v_c, [\tau_{c_b}, \tau_{c_e}])$ to C and update (τ_{curr}, B_{curr});
16 **return** C;

Definition 2. ***Chromosomes.*** *Given the input graph of EVOP–V2G, we represent a chromosome C as a sequence of actions, starting at V_s and ending at V_d, categorised as follows: (i) Order-serving actions: a single vertex (v_i), where $v_i \in V_o$. Each order must be served within its time window $[\tau_i^b, \tau_i^e]$, so we do not track specific arrival times. (ii) Charging/discharging actions: represented as $(v_i, [\tau_{c_b}, \tau_{c_e}])$, where v_i is a charging station. A positive v_i indicates charging, while a negative v_i indicates discharging. τ_{c_b} and τ_{c_e} denote the start and end times of charging/discharging.*

Verifying chromosome feasibility is straightforward and will be discussed in subsequent steps. As for the Fitness Function, as in the MIP model, we discretise charging/discharging prices over time steps $t_k \in T$ and calculate the profit accrued by the chromosome as follows:

$$f(C) = \sum_{(v_i) \in C} p_i + \sum_{(v_i, [\tau_{c_b}, \tau_{c_e}]) \in C} \Big(\sum_{\substack{t_k \in [\tau_{c_b}, \tau_{c_e}] \\ v_i < 0}} P_i^{Dt_k} - \sum_{\substack{t_k \in [\tau_{c_b}, \tau_{c_e}] \\ v_i > 0}} P_i^{Ct_k} \Big) \tag{24}$$

Generating Chromosome. As an important step in the initialisation phase, the EA initialises a population pool of k containing chromosomes. To efficiently generate these chromosomes, we design a two-phase greedy algorithm that first selects order-serving actions from V_o, then inserts up to m random charging/discharging actions between each pair. Algorithm 2 shows the pseudocode of our proposed algorithm. The algorithm takes as input the working hours $[\tau_w^s, \tau_w^d]$ of an EV driver, as well as the initial and minimum battery requirements $[B^s, B^d]$. Initially, the chromosome C is empty, and the variables τ_{curr} and B_{curr} track the arrival time and battery level at the current position and are set to τ^s and B^s, respectively.

Initially, the chromosome C is empty, and the variables τ_{curr} and B_{curr}, which track the arrival time and battery level at the current position, are set to τ^s and B^s, respectively.

Phase 1. (lines 1–8) To begin, the algorithm iterates through the order pool V_o and retrieves orders whose time windows $[\tau_i^b, \tau_i^e]$ fall within the working hours $[\tau_w^b, \tau_w^e]$ (line 1). The algorithm then appends the source (V_s) to C. From the available orders V_o', the algorithm initiates a while loop, iteratively appending random feasible orders to the chromosome C until the current insertion time τ_{curr} exceeds the end of the working hours τ_d, so no further actions can be inserted (line 3). In each iteration, the algorithm randomly selects an order $v_i \in V_o'$ and checks its feasibility by ensuring that (i) τ_{curr} allows the EV to reach v_i within its time window $[\tau_i^b, \tau_i^e]$, and (ii) B_{curr} is sufficient for the EV to complete v_i and return to the destination V_d. If v_i is feasible, the algorithm appends the order-serving action (v_i) to the chromosome C and updates the current time τ_{curr} and battery level B_{curr} to reflect the state after completing

order v_i (line 6). Otherwise, the algorithm removes v_i from V'_o to avoid selecting the same order twice (line 7). After completing the selection of orders from V'_o, the algorithm appends the destination V_d to C, concluding the first phase.

Phase 2. (lines 9–15) The algorithm then proceeds to its second phase by inserting charging/discharging actions between consecutive order-serving actions in C. Initially, τ_{curr} and B_{curr} are reset to τ^s and B^s (line 9). The algorithm then iterates through each consecutive pair of orders $(v_i, v_j) \in C$, updating τ_{curr} and B_{curr} to the time and battery level after completing order v_i (lines 10–11). By default, the algorithm assumes that a chromosome cannot contain more than m consecutive charging/discharging actions (we set $m = 4$ in our experiments). It then selects a random number between 0 and m (i.e., randomNumber$(0, m)$) to determine how many charging/discharging actions to insert. At each insertion step, the algorithm randomly chooses to charge or discharge and selects contiguous time steps $[\tau_{c_b}, \tau_{c_e}]$ between the current time τ_{curr} and the start time τ^b_j of the next order v_j. Let $p_e(v_i, v_j)$ denote the energy cost of travelling from v_i to v_j, based on energy consumption and the highest daily charging price. The algorithm ranks charging stations according to a combined factor of travel energy cost and charging/discharging price, as shown in Equation (25).

$$\sum_{\substack{t_k \in [\tau_{c_b}, \tau_{c_e}] \\ v_c < 0}} P_i^{Dt_k} - \sum_{\substack{t_k \in [\tau_{c_b}, \tau_{c_e}] \\ v_c > 0}} P_i^{Ct_k} - p_e(v_{curr}, v_c) - p_e(v_c, v_j) \tag{25}$$

The vertex v_{curr} represents the current location of the EV, while $v_c \in V_c$ denotes a charging station. We rank the charging stations by the charging (i.e., $v_c > 0$) or discharging (i.e., $v_c < 0$) price minus the energy cost to travel from v_{curr} to v_c and from v_c to v_j. From the top five ranked charging stations, we randomly select one to form $(v_c, [\tau_{c_b}, \tau_{c_e}])$ (line 13) and check its feasibility (line 14) by ensuring that: (i) departure at τ_{curr} allows the EV to reach v_c before τ_{c_b}; (ii) leaving v_c at τ_{c_e} permits arrival at v_j before τ^e_j; and (iii) B_{curr} is sufficient to reach v_c, charge/discharge during $[\tau_{c_b}, \tau_{c_e}]$, and then reach v_j. If the charging/discharging action $(v_c, [\tau_{c_b}, \tau_{c_e}])$ is valid, the algorithm inserts the action into C and updates τ_{curr} and B_{curr} accordingly (line 15). Finally, the algorithm terminates the insertion phase after iterating through every pair of orders (v_i, v_j) and returns the chromosome C (line 16). Note that, when generating the population, Algorithm 2 may be called multiple times, depending on the user settings.

4.1.2. Selection and Crossover:

The selection process involves identifying a pair of chromosomes from the population. Each instance of selection involves choosing an individual chromosome, thus necessitating two runs. We employ tournament selection [42] as the selection strategy, which is a popular and efficient method in evolutionary algorithms. This approach randomly selects a subset of n chromosomes from the population and chooses the one with the highest fitness. Each chromosome can be chosen only once per generation, with no repeats allowed.

For the selected pair of chromosomes, the crossover operation exchanges portions of the chromosomes to produce two offspring. Employing the single-point crossover method [43], we randomly select a point on each chromosome for recombination. At these points, the chromosomes undergo recombination, yielding two offspring. Note that the sequence of order-serving and charging/discharging actions undergoes crossover at the same randomly selected point. We ensure the feasibility of the generated offspring, adding only valid ones back to the population.

4.1.3. Mutation:

For chromosomes in the population pool, the mutation process introduces random changes to simulate genetic mutation. While random changes can diversify the offspring, they are unlikely to result in a valid solution of reasonable quality. To address the EVOP-V2G problem, we customise the mutation strategies as follows:

- **Order Mutation:** The order mutation randomly selects an order-serving action $(v_i) \in C$ from the offspring. Within the time window $[\tau^b_i, \tau^e_i]$ of v_i, we select another order $v_j \in V_o$ whose time window falls within the same interval $[\tau^b_i, \tau^e_i]$ to replace v_i.
- **Charging/Discharging Mutation:** The charging mutation randomly selects a charging/discharging action $(v_i, [\tau_{c_b}, \tau_{c_e}]) \in C$ from the offspring. Subsequently, a new charging/discharging action is generated to replace the original one. This action involves determining both the charging station and amount of charge/discharge, using the charging/discharging insertion phase of Algorithm 1.

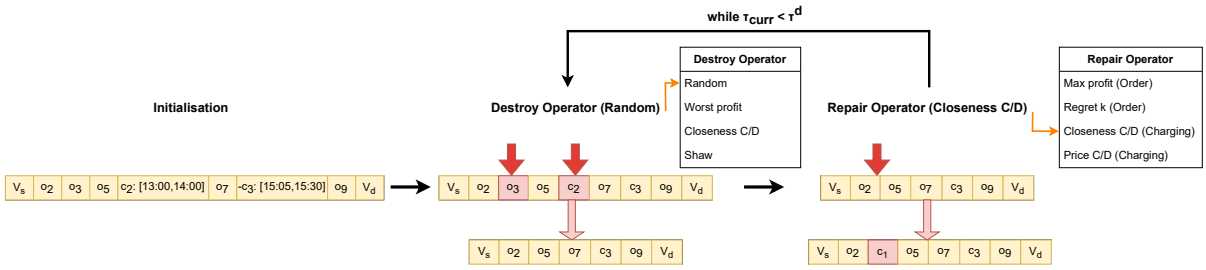


Figure 6: Illustration of the LNS process, showing the stages of initialisation, destroy operator, replacement operator, and optimisation using MIP.

- **Insertion Mutation:** Insertion mutation focuses on introducing the new orders into the offspring C. This process involves randomly selecting a position from C, followed by the random selection of an order from V_o, which is then inserted into the previously determined position.

For each mutation iteration, we set the probability of performing each mutation strategy mentioned above to 0.8. The feasibility of the mutated offspring is then checked, and invalid offspring are ignored. Each mutation is attempted up to 10 times, or until the fitness value of a generated offspring surpasses that of the original parent selected for mutation. This approach ensures that the offspring produced have a higher probability of improving upon the parental traits.

4.1.4. Replacement:

In each generation, the algorithm performs **selection & crossover** followed by **mutation** on the feasible chromosomes, according to their respective probabilities, introducing a varying number of new chromosomes into the population. The population is then sorted in descending order based on fitness values, and adjusted to the predefined size by removing the chromosomes with the lowest fitness values. This approach ensures that the elite chromosomes of the current population are retained for the next generation. After the replacement phase, the algorithm concludes one generation and starts the next generation.

Example 1. *Fig. 3 provides an example of the EA process applied to the toy instances depicted in Fig. 1. During the initialisation phase, the EA generates a population pool of k chromosomes using Algorithm 2. Each chromosome represents a sequence of actions starting at V_s and ending at V_d, involving the servicing of orders and performing charging or discharging along the way. For example, chromosome p_1 represents an EV driver starting at V_s, serving orders o_2, o_3, and o_5, charging at station c_2 from 13:00 to 14:00, then serving order o_7, discharging at c_3 from 15:05 to 15:30, serving order o_9, and finishing at destination V_d.*

Once initialisation is complete, the evolutionary process begins with the crossover operation, where two parent chromosomes (p_1 and p_2) are selected from the population, and a random crossover point is determined. The genes after this point are exchanged between the parents, producing two new offspring (c_1 and c_2), with the exchanged segments highlighted in yellow and green. Both offspring c_1 and c_2, are feasible and are inserted back into the population. Next, the mutation process begins after the crossover is completed for all chromosomes in the initial population pool. The mutation process introduces random changes to each chromosome selected by applying one of three mutation types: order, charging/discharging (C/D), or insertion. For example, Fig. 3 shows an order mutation for p_1 (with mutation probability higher or equal to 0.8) where o_5 is replaced by o_6, which has a similar time window. The resulting offspring is feasible and is added to the population. Chromosome c_2 does not undergo mutation, as its mutation probability is below 0.8. Finally, in the replacement phase, all chromosomes are sorted by fitness, and those beyond the k-th position are removed from the population. This completes one generation of the EA, and the process repeats until the current time τ_{curr} surpasses the predetermined termination time τ^d.

4.2. Large Neighborhood Search

Although EA offers a promising solution to the EVOP-V2G problem, it faces two key drawbacks: (i) maintaining a large population of chromosomes leads to slow convergence due to the need to improve multiple solutions simultaneously, and (ii) its reliance on random evolution-inspired operations (e.g., selection, crossover, mutation) results

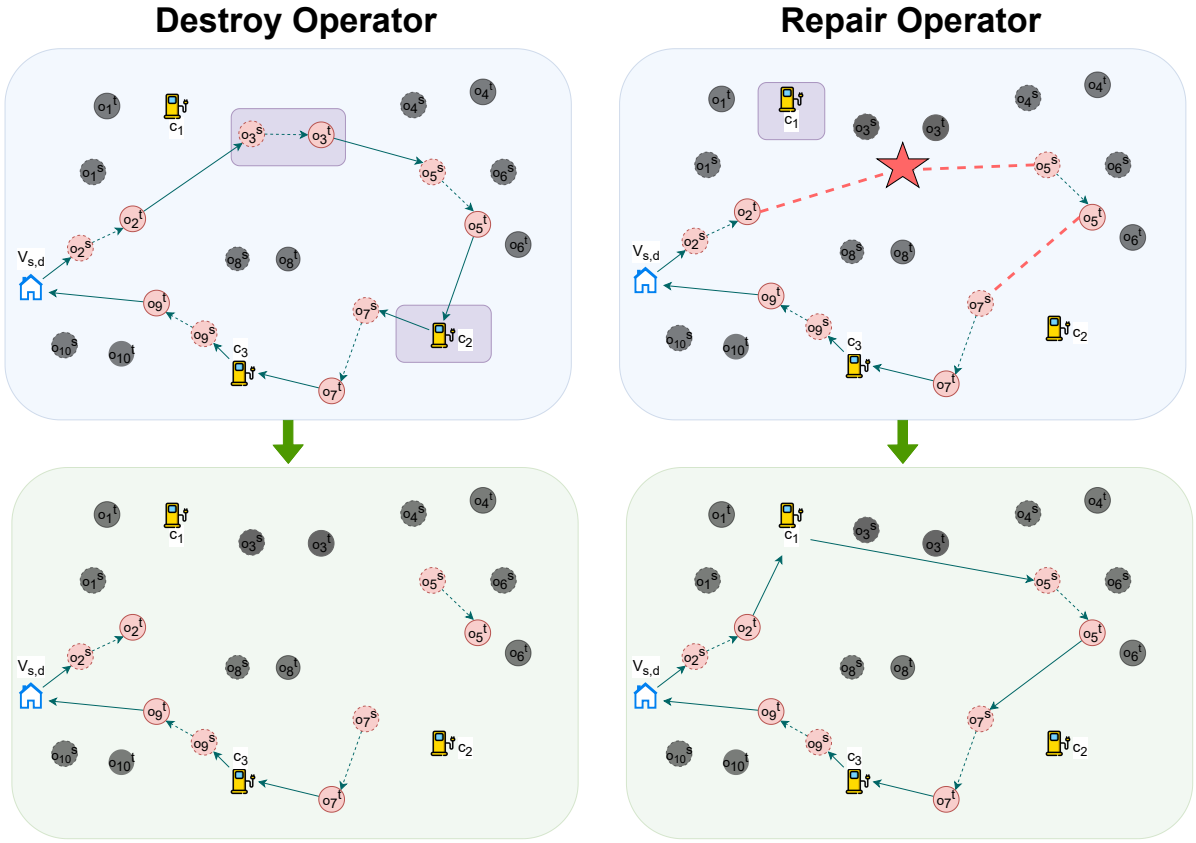


Figure 7: Representation of the destroy–repair process. The initial solution is shown in the blue box. A random-destruction strategy removes actions o_3 and c_2 (highlighted in purple), yielding a partial solution in the green box with the corresponding edges removed. The partial is then repaired using the closeness charging strategy: action c_1 (purple) is inserted at the position marked by the red star. The repaired route is indicated by red dashed lines, and the final repaired solution appears in the lower green box.

in solution quality being heavily influenced by chance. To address these issues, we investigate another efficient and successful family of algorithms: Large Neighbourhood Search (LNS) [44]. In combinatorial optimisation problems, the "neighbourhood" often refers to a local region or configuration space around a feasible solution. Unlike local search, which explores a single, narrowly defined neighbourhood, LNS systematically investigates a broader range of neighbourhoods.

Following the success of LNS, we developed an LNS-based algorithm to solve the EVOP-V2G problem. LNS maintains the best solution (i.e., the solution with the highest cost) so far and iteratively improves it by exploring the neighbourhood around this solution through a "destroy and repair" process. Algorithm 3 outlines the proposed LNS approach to EVOP-V2G. Fig. 6 shows the overall process of LNS. In broad strokes, LNS can be described as follows:

1. **Initialisation:** Initially, the LNS generates a feasible solution. The algorithm then evaluates the cost of solution and records it as the best solution, S_{best}.
2. **Destroy Operator:** In each iteration, LNS calls the destroy operator to remove the actions from S_{best}, making S_{best} a partial solution, S'_{best}.
3. **Repair Operator:** Taking the partial solution S'_{best}, the repair operator explores the neighborhood around S_{best} by repairing S'_{best} into a feasible solution S. LNS updates S_{best} to S if the cost of S improves upon the previous S_{best}. An example of the destroy and repair operation is shown in Fig. 7 and described in Example 2.

Similar to EA, LNS can run the "destroy and repair" operations for a certain number of iterations or terminate

Algorithm 3: Large Neighbourhood Search (High-Level)

Input: EVOP-V2G Instance; Evaluation Function $f(\cdot)$.
Parameters: Max Iterations: I; Time Limit: T; Random Degree: m; Reaction Factor: α; Charging Margin: θ.
Output: S_{best}

Initialise // see Sec. 4.2.1
  $S_{best} \leftarrow$ InitialSolution() ;
  iteration $\leftarrow 0$;
while not reached termination (iteration $\geq I$ or time limit $\geq T$) **do**
  Destroy Operator: $S'_{best} \leftarrow$ Destroy(S_{best}, destroy strategy, w^D, m); // see Sec. 4.2.2
  Repair Operator: $S \leftarrow$ Repair(S'_{best}, repair strategy, w^R, θ, m); // see Algorithm. 4
  MIP Refiner: $S \leftarrow$ run suboptimal MIP model to refine charging/discharging action of S; // see Sec. 4.2.4
  if $f(S) < f(S_{best})$ **then**
    $S \leftarrow S_{best}$
  Adaptive Update: update the weight of destroy strategy selection w^D and destroy strategy selection w^R with reaction factor α using Equation (29) // see Sec. 4.2.4
  iteration++;
return S_{best}

based on a predefined cut-off time. The current best solution S_{best} is then returned as the final result. Unlike EA, which improves multiple solutions simultaneously at each iteration, LNS focuses on enhancing a single solution at a time. While efficient, achieving high-quality solutions with LNS requires careful design of its destroy and repair operators. Additionally, we introduce optimisation to further improve the algorithm.

4.2.1. Initialisation

To represent a feasible solution, we inherit the same chromosome representation used in EA, where each feasible solution is represented as a set of actions $(v_i, [\tau_{c_b}, \tau_{c_e}])$. Since the objective of LNS is to maximise the amount of profit gained, we reuse the fitness function of EA (i.e., equation (24)) to evaluate the cost of the solution. Essentially, the initialisation phase only requires generating a feasible solution; the quality of the solution often does not play an important role, as the algorithm quickly performs many iterations to improve the solution. To keep the process simple, we initialise S_{best} with actions to visit the source and destination vertices only, without any additional actions in between. In the first iteration, the algorithm skips the destroy operator and calls the repair operator to enhance S_{best} into a high-quality solution.

4.2.2. Destroy Operator

Given the current best solution S_{best}, the destroy operator deconstructs the solution by removing a set of actions between source and destination. The number of actions removed is a random integer k, chosen within minimal and maximal thresholds based on the actions between the source and destination. The main purpose of the destroy operator is to determine which actions to eliminate, enabling S_{best} to escape local optima. This, in turn, allows the repair operator to reconstruct the solution to a neighbour with higher cost. We designed four different strategies for removing actions from S_{best}:

- **Random Removal**: To diversify neighborhood exploration sufficiently, random removal strategies select a random subset of k actions to remove from S_{best}.
- **Worst Profit Removal**: The worst profit removal is a heuristic strategy based on the observation that low-profit actions may prevent the solution from achieving higher costs. This strategy removes k actions from S_{best}. To evaluate the profit of an action $(v_i, [\tau_{c_b}, \tau_{c_e}]) \in C$, we consider:
  - For serving order $v_i \in V_o$, we take its profit p_i.
  - For charging at a station $v_i \in V_c$, we compute the profit is the sum of the differences between the maximum charging price at all stations and the current charging price, i.e., $\sum_{t_k \in [\tau_{c_b}, \tau_{c_e}]} \left(\max(P^{Ct_k}) - P_i^{Ct_k}\right)$.

Algorithm 4: Repairing the Partial Solution

Input: S'_{best}: The partial solution returned by destroy operator.
Output: S: A feasible solution returned by repair operator.

```
1  B_last = compute the battery level at the last action of S'_best;
2  while an action is inserted successfully do
3      if B_last < B × θ then
4          Insert a charging/discharging action in S'_best;
5          Insert an order-serving action in S'_best;
6      else
7          Insert an order-serving action in S'_best;
8          Insert a charging/discharging action in S'_best;
9      B_last = update the battery level at the last action of S'_best;
10 return S = S'_best;
```

  - For discharging at a station $v_i \in V_c$, the profit is the sum of the discharging price, i.e., $\sum_{t_k \in [\tau_{c_b}, \tau_{c_e}]} \left(P_i^{Ct_k}\right)$.

- **Closeness Charging/Discharging Removal**: The closeness charging removal method leverages the observation that EV charging/discharging often coincides with nearby orders, e.g., when an EV charges/discharges at a station, it tends to handle nearby orders. This approach aims to reduce these clusters by randomly selecting and removing k actions associated with a charging/discharging event.

- **Shaw Removal**: The shaw removal is a heuristic strategy based on the observation that removing correlated actions can lead to better solutions, while removing dissimilar actions often reverts the solution to a similar or worse state. Therefore, shaw removal randomly selects an action from S_{best} and removes the k most relevant actions. To measure the relatedness of two actions $(v_i, [\tau_{c_b}, \tau_{c_e}])$, $(v_j, [\tau'_{c_b}, \tau'_{c_e}])$, we adapt the measurement [45] as follow:

$$d(l_i^d, l_j^d) + d(l_i^p, l_j^p) + |\tau_{c_b} - \tau'_{c_b}| + |\tau_{c_e} - \tau'_{c_e}| \tag{26}$$

  The measurement includes the sum of distances between the pick-up and dropoff locations for two actions and their time difference. For an order-serving action (v_i), the action time $[\tau_{c_b}, \tau_{c_e}]$ is set to its time window $[\tau_i^b, \tau_i^e]$. For a charging/discharging action $(v_i, [\tau_{c_b}, \tau_{c_e}])$, the pick-up l_i^p and dropoff l_i^d locations are set to the same location as v_i.

4.2.3. Repair Operator

After the destroy operator removes a set of actions from S_{best} to form a partial solution S'_{best}, the repair operator then reconstructs the partial solution into a new solution S by adding new actions. The primary objective of the repair operator is to utilise the partial solution S'_{best} to explore the best solution in the neighbourhood nearby. Therefore, the repaired solution must be (i) feasible and (ii) of high quality. Achieving both simultaneously is challenging. To address this, we have designed a simple yet efficient algorithm that iteratively inserts actions to form the new solution S.

Algorithm 4 presents the pseudo-code of our proposed algorithm. Initially, the algorithm takes the partial solution S'_{best} and evaluates the remaining battery level, B_{last}, after the last actions of S'_{best} (line 1). The algorithm then enters a while loop, attempting to insert new actions, which are either order-serving or charging/discharging actions. In each iteration, the algorithm tries to insert both types of actions. If B_{last} is below a threshold θ of the battery capacity B (line 3), the algorithm prioritises charging/discharging actions to maintain flexibility for order-serving actions, and vice versa (line 4 - 8). After each insertion, B_{last} is updated. The loop continues as long as at least one insertion is successful and terminates when no more actions can be inserted. Finally, the algorithm returns the repaired solution S. In this paper, we set θ to 0.15 as charging margin, meaning the algorithm prioritises charging/discharging actions when the battery level B_{last} falls below 15%. Next, we describe different repair strategies for inserting order-serving or charging/discharging actions.

Inserting Order-Serving Actions. Each order in the order pool V_o can only be served once. To insert an order-serving action, the algorithm scans V_o and skips orders already served in S'_{best}. For each remaining order v_o, it checks if the EV driver has sufficient time and battery to complete the order while preserving other actions in S'_{best}. If no valid orders are available, the algorithm returns a failure and may attempt to insert charging or discharging actions instead. For valid orders, the algorithm selects based on the following strategies:

- **Max Profit Insertion:** For each valid order, it may be possible to insert it at multiple positions in S'_{best}. To determine the order and insertion position, we define the insertion profit $I(v_j)$ for inserting order v_j between $(v_i, [\tau_{c_b}, \tau_{c_e}])$ and $(v_k, [\tau'_{c_b}, \tau'_{c_e}])$. Let $p_e(v_i, v_j)$ represent the energy cost from v_i to v_j, calculated based on energy consumption and the highest charging price. The insertion profit $I(v_j)$ is then computed as follows:

$$I(v_j) = p_j - p_e(v_i, v_j) - p_e(v_j, v_k) \tag{27}$$

  The max profit insertion simply selects the order and position with the highest insertion profit. The intuition here is to maximise the total profit for S'_{best}.

- **Regret k Insertion:** The regret-k insertion improves the maximum profit insertion by incorporating a look-ahead mechanism. Let $I_1(v_j)$ denote the highest insertion cost for inserting the order v_j into S'_{best}. The regret value of inserting order v_j measures the profit lost between inserting v_j at the position with the highest insertion cost and the second highest insertion cost (i.e., $I_1(v_j) - I_2(v_j)$). This regret value can naturally be extended to consider up to k of the highest insertion costs as follows:

$$R(v_j) = \sum_{n<k}^{n=1} (I_n(v_j) - I_1(v_j)) \tag{28}$$

  The regret-k value indicates the importance of inserting the order at the current iteration, which would otherwise result in significant profit loss if inserted later. Hence, the strategy selects the order with the highest regret-k value and inserts it at the position with the highest insertion cost.

Inserting Charging/Discharging Actions. Unlike inserting an order-serving action, charging/discharging actions are not bound by strict time constraints, i.e., an EV can charge or discharge whenever it arrives at a station, while an order must be served within its time window. Consequently, considering all possible insertion positions for each charging station is time-consuming. To address this, we determine charging/discharging actions by iteratively selecting unique positions from S'_{best} and evaluating potential actions at each charging station. In each iteration, an EV charges (or discharges) if its battery level is below (or above) the required level for remaining actions, plus a threshold $B \times \theta$. Each action is verified to ensure the EV can charge/discharge for at least one timestep and has enough battery to reach and leave the charging station. Similar to EA, we limit the EV to no more than m consecutive charging/discharging actions. If valid actions are identified, the algorithm selects the charging/discharging station based on the following strategies:

- **Closeness Charging/Discharging Insertion:** Among all charging/discharging stations, this strategy selects those where the charging station is closest to the adjacent actions in S'_{best}. The intuition is to choose nearby charging stations, making S'_{best} more flexible for subsequent order-serving or charging/discharging activities.

- **Price Charging/Discharging Insertion:** For each charging/discharging station, we consider the variable price of station throughout the day, assuming the EV can charge/discharge for the maximum available timesteps. A lower price is preferred for charging, while a higher price is better for discharging. The price charging insertion selects the charging/discharging station with the best prices, aiming to maximise the profit collected in S'_{best}.

Once the station is selected, we randomly chooses a number of timesteps between 1 and the maximum available at the insertion position and return the charging/discharging actions accordingly. If no valid action is found, it checks the next random position and returns a failure after evaluating all positions.

Example 2. *Fig. 6 illustrates the main steps of the LNS algorithm applied to the toy instance in Fig. 1. In the initialisation phase, the solution generated after the first iteration is formed by repairing the trivial initial solution $\{V_s, V_d\}$. Once established, the algorithm iteratively performs a destroy and repair process to enhance the solution. Four strategies are available for the destroy operator: random, worst profit, closeness charging/discharging (C/D), and Shaw. In each iteration, the LNS selects one strategy to transform the current best solution into a partial solution; for example, the random destroy strategy removes two actions, o_3 and c_2. The subsequent repair step refines the partial solution using operators from two categories: order-serving actions (max profit and regret k) and charging/discharging actions (closeness and price C/D). In this instance, the closeness charging/discharging repair operator is selected, which adds a charging decision, c_1, at position 2. This iteration concludes, and the destroy and repair process continues in subsequent iterations, provided that the current computational time τ_{curr} does not exceed the predetermined termination time τ^d.*

4.2.4. Optimisation

So far, we have introduced various strategies for the destroy and repair operator, each incorporating a greedy approach that ensures efficiency. While each iteration is fast, achieving a high-quality solution requires guidance to select a suitable strategy. To enhance LNS, we introduce a set of optimisations:

Adaptive LNS [45] (ALNS). ALNS is an enhanced version of LNS that employs multiple strategies, tracking their success in improving the current solution and using the most promising strategy to choose the next neighborhood. To implement the strategies selection, assume we have a set of strategies $\mathcal{S}$. Initially, each strategy is assigned a weight value $w_i = 1$. In each iteration, strategies are selected using the roulette wheel selection method [46], where a strategy i is chosen with a probability of $\frac{w_i}{\sum w_j}$ for $j \in \mathcal{S}$. After an iteration, the weight of the selected strategy i is updated based on the improvement in the solution. Specifically, w_i is adjusted as follows:

$$\alpha \times max\{S - S_{best}, 0\} + (1 - \alpha)w_i \tag{29}$$

The parameter α is a user-defined reaction factor that determines the rate at which the weights adjust in response to variations in the relative success of enhancing the current solution. In our experiments, we set α to 0.01. Note that the strategy selection for the destroy and repair operators is conducted independently. For the repair operator, we simplify the selection process by considering permutations of strategies for inserting the order-serving and charging/discharging actions.

Introducing Randomness to Strategies. LNS uses various strategies for selecting destruction or repair actions, typically through a greedy approach: (i) rank actions based on a single measurement and (ii) select the top one or k actions to remove or insert. While different strategies may be employed in each iteration, relying on a single measurement can lead to local optima. To address this, we introduce randomness into the action selection process. Assuming we have N actions sorted by measurement, instead of selecting the top action, we choose an action at the position of N based on the following criteria:

$$r \in [0, 1] : r^m \times N \tag{30}$$

The m value represents the degree of freedom. Higher m values indicate a greater likelihood of selecting an action from the top few options. To ensure the algorithm maintains a high probability of selecting the best action based on the measurement, we set $m = 5$ in our experiments.

Improving Solution Using MIP Solver. Recall that when inserting a charging/discharging action, the repair operator determines the amount by randomly assigning a number of time steps to each action. This method can frequently result in suboptimal charging/discharging solutions. i.e., charging/discharging more at one station may result in an unnecessary visit to a subsequent station, thereby reducing overall profit. To address this issue, we reuse the MIP model proposed in Section 3.2 to further optimise the charging/discharging actions.

While the overall MIP model remains the same, we modify the input graph. Specifically, using the repaired solution S, we build an input graph containing a single path that starts from the source and ends at the destination, visiting orders and charging stations in between. Each order vertex must be visited once, but for each charging station,

we add an edge between adjacent vertices to allow the model to skip the charging station. Given the modified input graph, the MIP model only needs to reason about the charging/discharging period and the selection of the station, thereby the model often find the solution fast. To further improve efficiency, we set the suboptimality gap to 5% and allow the solver to terminate upon finding a solution within 5% of the optimal. This prevents the solver from running excessively long at each iteration.

5. Evaluation and Results

In this section, we assess the proposed approaches and investigate the conditions under which V2G yields the greatest benefit. We describe the dataset and benchmark instances, detail the protocol and metrics, report comparative results for MIP, EA, LNS and the baseline, and conduct experiments on spatial clustering, time-window structure and ride-length distributions.

5.1. Settings

To evaluate our proposed algorithms, we use a dataset assembled by integrating real-world ride-sharing trip statistics, EV specifications and ride-sharing fares. The EVOP-V2G algorithms rely on two main inputs: a set of orders and a set of charging stations. First, we describe how we derived candidate orders and candidate charging stations from real-world datasets. We then use these candidates in experiments to evaluate the proposed algorithms.

Order Candidates: We use ride-sharing orders from a public dataset [47], which contains real-world trip records in and around Melbourne, Australia. Among the three available dataset sizes, we chose the largest, with 68,625 orders. Each order includes a pick-up location l_i^p, drop-off location l_i^d, travel distance d_i, travel time t_i, and a time window $[\tau_i^b, \tau_i^e]$. Focusing on the Melbourne metropolitan area, we exclude orders where both pick-up and drop-off lie outside this region, resulting in 64,487 valid orders. The profit p_i for each order is based on a fare structure: a 2.75 base fare, 1.49/km distance fare, and 0.39/min time fare, with a 30% deduction reflecting typical ride-sharing service fees [48]. All amounts are in AUD.

To simulate realistic scenarios, we vary order selection according to three criteria: (i) Bounding box, filtering orders using 10%, 40%, 70% and 100% of the Melbourne metropolitan area; (ii) Ride length, grouping trips into 5–10 km, 10–25 km and > 25 km; and (iii) Time period, selecting orders within 2-hour (9–11 am), 5-hour (9 am–2 pm), and 8-hour (9 am–5 pm) windows. Unless otherwise stated, we set the bounding box to 40%, the ride-length category to 10–25 km and the time period to 8 h, varying one criterion at a time while keeping the others fixed.

Charging Station Candidates: We obtained charging station data from PlugShare[4], initially identifying 440 EV charging stations across Australia. After removing 171 stations that lacked charging rate data or lay outside the Melbourne metropolitan area, 70 stations remained. PlugShare reports a single, uniform charging price per station that is constant throughout the day. Accordingly, our experiments adopt these station-level prices and assume that dynamic pricing is not available at the selected charging locations. For the discharging price at each station, we use Energy Australia's time-of-use (TOU) feed-in tariff[5]. As of 2024, this scheme specifies three time periods: peak (4pm - 9pm), shoulder (9pm - 10am and 2pm - 4pm), and off-peak (10am - 2pm). The corresponding prices are \$0.117 (peak), \$0.061 (shoulder), and \$0.043 (off-peak).

We assume that both the source and destination (e.g., the EV owner's home) are equipped with charging infrastructure, with a charging rate of 7 kW. The charging and discharging prices at these locations follow a standard two-period time-of-use (TOU) tariff: during the peak period (3pm - 9pm), the price is \$0.412 per kWh. During the off-peak period, covering all other times, the price is \$0.2665 per kWh [49]. All prices are in AUD.

EV Specifications: In our experiments, we utilize the specifications of the Tesla Model S for our EV setting, with a battery capacity of 70 kWh and a driving range of 400 km. The driving efficiency γ can be calculated as the ratio of battery capacity to driving range which is 0.175 kWh/km in our experiments. Please note that these EV settings are taken in as parameters and can be easily adjusted to match other EV specifications. The working hours for the EV owner are set to the standard normal working hours, with τ_w^s set to 9 AM and τ_w^e set to 5 PM. We set the initial battery level B^s at source as 100% and set minimal battery required B^d at destination as 0%.

[4] `https://www.plugshare.com/`

[5] `https://www.energyaustralia.com.au/home/solar/feed-in-tariffs`

Competitor and Implementation: Since there is no existing work that addresses the same problem, we propose a simple local search algorithm as a baseline approach (BL). The BL is designed to represent a simple, human-inspired decision-making process. The BL is formulated as a greedy local search algorithm intended to emulate the behaviour of an EV driver making intuitive operational decisions in real time, without global optimisation. Its purpose is to establish a realistic and interpretable reference point that reflects straightforward decision logic rather than computational sophistication. The design of the BL algorithm is motivated by the observation that EV drivers typically act based on immediate operational feasibility—serving available orders as long as sufficient battery charge is available and charging only when necessary. Accordingly, the BL greedily prioritises order-serving actions whenever the vehicle's battery level exceeds a predefined threshold and initiates charging actions only when the charge drops below this level. This behavioural design mirrors the natural trade-offs that an EV driver would make when balancing service fulfilment, energy constraints and time availability.

This baseline approach first sorts the available orders into time windows and then greedily selects the nearest time window. From this selected time window, it randomly inserts order-serving actions until the current action sequence exceeds the working-hours limit, at which point the vehicle returns to its destination. BL handles charging and discharging in a greedy manner. When the battery level is greater than 20%, BL preferentially inserts order-serving actions. When the battery level is less than 20%, BL inserts charging actions. We set the charging threshold at 20% battery level in the baseline algorithm, aligning with empirical evidence on driver behaviour. A recent large-scale Norwegian survey of 1,005 EV owners found that drivers' range anxiety increases sharply below 30% SOC, with comfort at lower SOC levels strongly associated with reduced anxiety and higher "range confidence" [50]. Thus, adopting a 20% threshold reflects a psychologically realistic lower bound consistent with observed real-world charging behaviour. Charging stations are selected using a similar approach to EA: we compute a score for each charging station candidate based on distance and charging price, then the BL randomly selects one from the top five candidates. When charging, we assume charging continues until the battery reaches full capacity. Discharging actions are inserted only when the EV returns to its destination, at which point we assume the EV discharges all remaining energy down to the battery threshold. The BL executes a while loop until the first feasible solution is found and returned.

All implementations (MIP, EA, LNS and the baseline) are written in C++. We solve the MIP model using the off-the-shelf solver Gurobi (version 10.0.3)[6] . For EA, we set the population size to 2,000. In initial experiments, we also varied the population size, but this did not significantly affect results and in some cases worsened them. We conduct all experiments on an Apple M1 Pro with 16,GB RAM. For all algorithms except MIP, we set run-time limits of 60 seconds for small-scale instances and 300 seconds for large-scale instances; after these limits, we return the best solution found by each algorithm. We allow MIP to run until an optimal solution is found (which can take up to 27 hours for some instances). For reproducibility, our implementation is available online[7].

5.2. Experiment Results

5.2.1. Small-Scale Instances

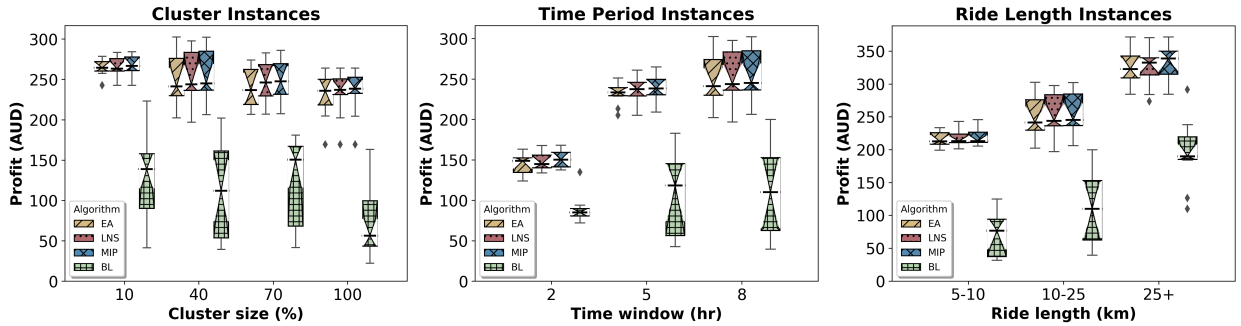
Instance: As mentioned in Section 4, the MIP model faces scalability limitations, rendering it impractical to solve larger instances. Therefore, to compare its performance with other algorithms, we construct small-scale instances. For each instance, we set the source and destination to the same location (i.e., the EV owner's home) and designate three charging locations: one at the EV owner's home and two others selected at random from the charging station candidates. In each experiment, we generate 10 instances; each instance consists of 30 orders sampled at random from the corresponding order candidates. We report the results for these 10 instances.

Results: Fig. 8 presents boxplots for the proposed algorithms (MIP, LNS and EA) and BL on the small-scale instances. Each boxplot shows the distribution of profit across the 10 instances. The box in each plot represents the interquartile range (IQR), covering the middle 50% of observations, with the line inside marking the median. The whiskers extend to the smallest and largest values within 1.5 times the IQR from the first and third quartiles, respectively. Observations that fall outside this range are considered outliers and are denoted by diamond-shaped points. Notches visualise an approximate 95% confidence interval for the median, computed as:

$$\text{median} \pm 1.57 \times \frac{\text{IQR}}{\sqrt{n}}$$

[6] `https://www.gurobi.com`

[7] `https://github.com/goldi1027/evop`

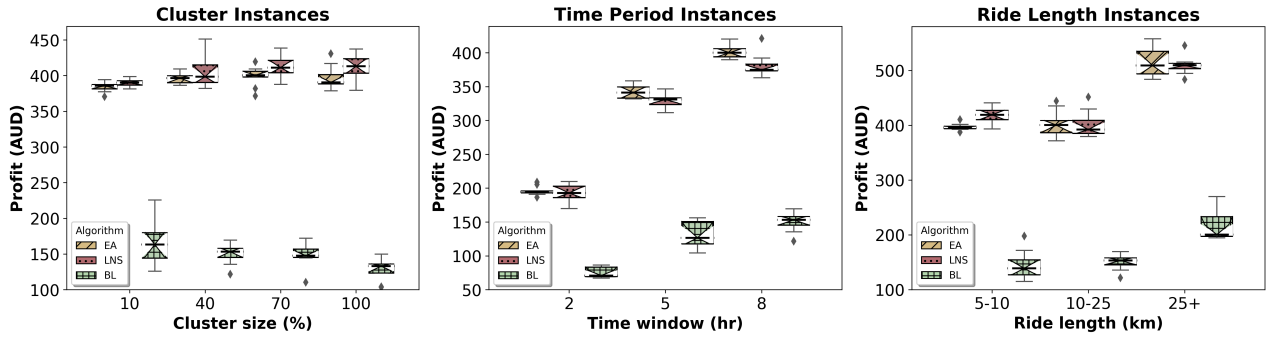


(a) (b) (c)

Figure 8: Small-scale instances. The x-axis represents the variation in parameters for each criterion. Fig. 8a shows the cluster bounding box as a percentage (%), Fig. 8b illustrates the time window in hours (h), and Fig. 8c displays the ride-length range in kilometres (km). The y-axis shows the overall profit in AUD. Each boxplot for each competitor represents the distribution of profit over ten repeats of the same parameter setting. The box represents the interquartile range (IQR), covering 50% of the distribution, with the line inside marking the median. Whiskers extend to the smallest and largest values within 1.5 times the IQR from the first and third quartiles, respectively. The diamond-shaped points denote outliers that lie outside this range. Notches represent an approximate 95% confidence interval for the median; non-overlapping notches between two boxes suggest a statistically significant difference in medians.

Non-overlapping notches between two boxes provide visual evidence that the corresponding medians differ at roughly the 5% significance level. We employ notched boxplots because each experiment includes 10 instances, which is sufficient to estimate a robust median and its uncertainty. The notch leverages these replicates to provide a distribution-free, outlier-resilient indicator of variability. With 10 runs, the median is already stable to occasional extremes, and the notches enable immediate visual comparison across algorithms: non-overlapping notches imply meaningfully different medians without reliance on parametric assumptions. Using the notches, BL's notches lie strictly below and do not overlap with those of EA, LNS and MIP across all settings, indicating a significantly lower median profit. By contrast, EA, LNS and MIP exhibit substantial notch overlap, implying no statistically clear differences among their median profits. The narrower notches for EA, LNS and MIP indicate greater stability, whereas BL's wider notches reflect higher variability across runs. In our inspection of the solution outputs from EA and LNS, we found frequent multiple revisits to a single charging station due to low or free charging prices. To ensure a fair comparison, we determined the maximum number of station revisits across EA and LNS (up to 3) and imposed this as a constraint in the MIP model. Because of computational limits (solve times exceeding 48 hours), we restricted revisits to 2, which still enabled MIP to outperform EA and LNS on all instances. MIP computation times ranged from 5 seconds to 27 hours. Overall, the proposed algorithms (MIP, LNS and EA) substantially outperform the baseline (BL), achieving more than twice the profit. Among these, the MIP approach consistently attains the highest overall profit across all instances. Nevertheless, EA and LNS remain highly competitive, delivering near-optimal solution quality relative to MIP.

Fig. 8a illustrates the effect of the bounding box size at 10%, 40%, 70% and 100%. As the bounding box enlarges, the overall profit remains broadly similar but shows a slight decrease at 100% for the proposed algorithms. This reduction arises because a larger bounding box disperses orders over a wider area, increasing inter-order travel distances and non-revenue time. Fig. 8b shows the effect of increasing the time period from 2 to 5 and 8 hours. With longer time periods, orders are more temporally dispersed, enabling EVs to serve more requests within the same working hours and thereby increasing total profit. Finally, Fig. 8c examines the effect of ride-length categories 5–10,km, 10–25,km and > 25,km. As ride length increases, total profit also increases, primarily due to higher fares per order. Although the number of completed orders is relatively consistent across methods, longer rides contribute more to profit. For rides over 25,km, however, profit variability is higher: missing a small number of high-fare orders due to suboptimal sequencing can yield lower performance than for mid-range rides. This underscores the importance of effective order selection and routing, particularly in high-value settings.



(a) (b) (c)

Figure 9: Large-scale instances. The x-axis represents the variation in parameters for each criterion. Fig. 9a shows the bounding box as a percentage (%) of the metropolitan area, Fig. 9b illustrates the time window in hours (h), and Fig. 9c displays the ride-length range in kilometres (km). The y-axis shows the overall profit in AUD. Each boxplot for each competitor represents the distribution of profit over 10 repeats of the same parameter setting. The box represents the interquartile range (IQR), covering 50% of the distribution, with the line inside marking the median. Whiskers extend to the smallest and largest values within 1.5 times the IQR from the first and third quartiles, respectively. The diamond-shaped points denote outliers that lie outside this range. Notches represent an approximate 95% confidence interval for the median; non-overlapping notches between two boxes suggest a statistically significant difference in medians.

5.2.2. Large-Scale Instances

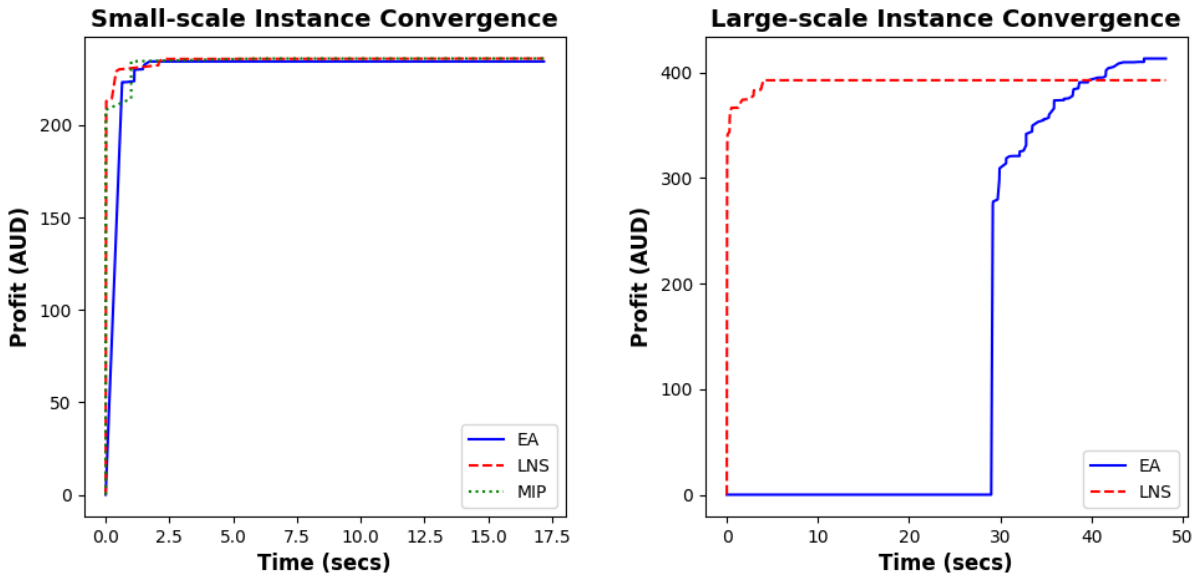
Instances: To further assess the relative performance of LNS and EA against each other and the baseline, we evaluate them on large-scale instances with an increasing number of orders. Similar to the small-scale experiments, we generate 10 instances for each order-candidate variation. Each large-scale instance comprises 900 randomly selected orders and all 70 charging stations. The choice of 900 reflects the size of the filtered dataset under the most restrictive criteria. For each instance, the EV's source and destination are identical and are randomly selected within the bounding box of the order set.

Results: Fig. 9 shows the results of our algorithms (LNS and EA) and BL on large-scale instances. It is evident from the figure that both LNS and EA outperform BL, confirming the significant advantage of utilising our proposed algorithm in real-world scenarios. As with the small-scale instances, the notches in the boxplots show the same pattern: BL's notches lie below and do not overlap with those of EA and LNS, indicating a lower median profit. EA and LNS, by contrast, exhibit overlapping notches, implying no statistically clear difference between their median profits. Their notches are comparatively narrow, suggesting higher precision and stability. Relative to EA, LNS attains higher profit in most instances. This further demonstrates the advantage of LNS; the higher profit arises primarily from its ability to adaptively select destroy–repair strategies and thereby escape local optima efficiently. By contrast, EA simulates a bio-inspired evolutionary process. Although comparatively simple relative to the more sophisticated mechanisms in LNS, improvements in solution quality often depend on the stochasticity introduced by selection, crossover and mutation. However, as shown in Fig. 9, randomisation in EA can be advantageous for certain problem instances (such as longer time periods) where diverse exploration of the solution space is required. In instances where EA outperforms LNS, this advantage stems from EA's capacity to leverage randomisation through evolutionary operators such as crossover and mutation, enabling exploration of a wider set of solutions and avoiding entrapment in local optima. This broader exploration enables EA to discover higher-quality solutions that LNS, which tends to focus on local improvements, may miss.

Across variations in order candidates, the overall trends in large-scale instances mirror those in small-scale instances. However, in large-scale instances the total profit increases and the boxplots exhibit more tightly clustered points. This arises because large-scale instances comprise a greater number of orders and charging stations, which facilitates the discovery of better, higher-profit solutions.

5.2.3. The convergence rate of different algorithms

To further evaluate the performance of the different algorithms, we randomly selected one instance from each of the small-scale and large-scale instances and plotted their convergence. Fig. 10a shows the convergence of EA, LNS and MIP on a small-scale instance. With each new solution found, we plotted its profit and the elapsed time in seconds



(a) Small-Scale Instance

(b) Large-Scale Instance

Figure 10: Convergence plots for small- and large-scale instances comparing the three approaches. The x-axis represents the elapsed time to discover improved solutions (in seconds), while the y-axis shows profit in AUD. The x-axis extends to the final time point at which any approach identified an improvement.

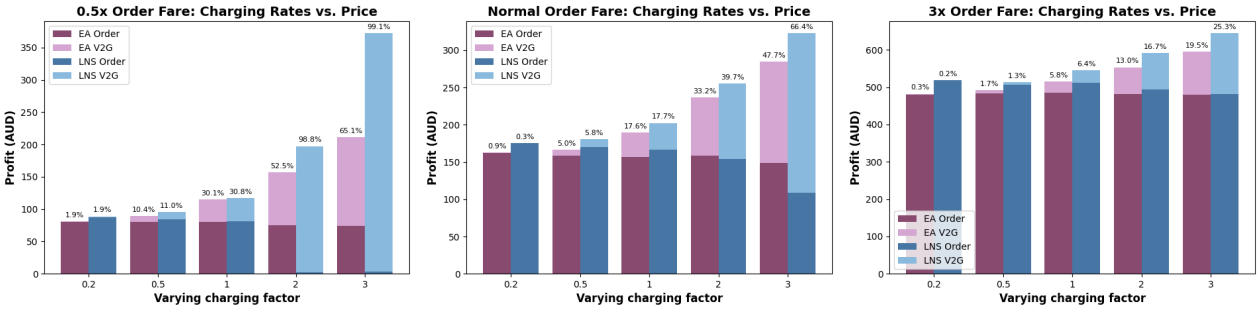
required to obtain it. The plot ends when all methods have found their best solutions. As shown in the figure, the small-scale instance is relatively easy to solve and all algorithms converge quickly. Even MIP rapidly finds a solution of reasonably high quality.

For the large-scale instance, Fig. 10b presents results for LNS and EA under the same settings. Despite the increased complexity, LNS quickly finds the first feasible solution and converges to the best solution within 2 seconds. By contrast, EA takes longer to initialise the population, requiring approximately 29 seconds to find its first feasible solution, and it then converges more slowly than LNS, taking up to 50 seconds to reach its best solution. The convergence curves therefore quantify both time-to-first-feasible and time-to-best, highlighting LNS's advantage when rapid responsiveness is required. Importantly, because each order-candidate variation is run with 10 independent repeats and a 300-second time limit, both algorithms have ample budget to stabilise: the trajectories flatten well before the cut-off, indicating that the reported solutions are near each method's performance envelope. Consequently, despite different convergence speeds, both LNS and EA consistently deliver high-quality solutions under the prescribed time budget.

5.2.4. Effect of varying charging rate, price, and order fares

Instances: With the increasing adoption of EVs and advances in fast-charging technologies, charging rates and prices are expected to evolve significantly. To account for these developments, we examine the impact of varying charging rates and prices within the EVOP-V2G problem. Specifically, we consider large-scale instances with a 2-hour time window for orders. This choice is informed by observations from previous experiments, which indicate that a shorter operational order time frame provides greater opportunities for charging and discharging activities outside this period. In our experiments, we vary the charging rates and prices at each station by scaling factors of 0.2, 0.5, 1, 2 and 3. Additionally, we vary order fares by factors of 0.5 and 3. We run LNS and EA for each setting and report the average values over 10 instances.

Results: Fig. 11 presents bar charts of the total profit attained by each algorithm for scaling factors of charging rate and price. Figs. 11c and 11a show the effect of additionally scaling order fares alongside charging rate and price. We increase charging rate and price simultaneously both because they are commonly correlated and to better illustrate



(a) 0.5 factor decrease in order fare

(b) Normal order fare

(c) 3 factor increase in order fare

Figure 11: Variation in charging rates, prices and order fares for 2-hour instances. The x-axis represents the varying factors for both charging rate and price, while the y-axis shows profit in AUD. Fig. 11a shows the results when order fares are decreased by a factor of 0.5, Fig. 11b shows the results for normal order fares, and Fig. 11c shows the results when order fares are increased by a factor of 3. The lighter shades for each algorithm denote profit from discharging, with annotations above the bars indicating the percentage contribution of the discharging incentive to overall profit.

the trends observed when analysing them separately. For each bar, the lighter-coloured area denotes V2G profit (i.e., buying and selling energy), and the annotation above the bar indicates the percentage of total profit attributable to V2G.

Charging rate and price: At the default charging and order settings (scaling factor 1), based on real-world data, V2G profit is about 20% of total profit, as shown in Fig. 11b. This indicates that V2G can substantially increase overall profit. As expected, when charging rate and price increase, the share of V2G in total profit becomes more pronounced; for example, when both are increased threefold, the percentage of V2G profit rises to 47.7% for EA and 66.4% for LNS. This highlights that our algorithms, particularly LNS, effectively identify and exploit more profitable V2G opportunities. As charging rate and price increase, V2G becomes more profitable and, consequently, profit from orders decreases as the algorithms adapt by prioritising V2G over some orders, treating it as a dominant factor in profit maximisation.

While profit from serving orders shows a downward trend for both EA and LNS, the effect is more pronounced for LNS. Overall profit for both algorithms increases as charging rate and price increase. This suggests that, as charging rates and prices rise in future, the benefits of V2G will become more pronounced. When charging rates and prices decrease (i.e., scaling factors 0.5 and 0.2), V2G profit declines relative to order profit. As expected, lower charging rates and prices reduce the incentive to discharge, which in turn lowers overall profit; consequently, the primary contributor to overall profit shifts to profit from orders. Comparing LNS and EA, LNS achieves better performance in discharging decisions and thus higher profits. This is because LNS utilises the MIP model to optimise charging and discharging actions at each iteration. By contrast, EA initially applies a two-phase algorithm that inserts order-serving actions first, followed by charging and discharging actions. More advanced charging and discharging strategies in EA arise only through the mutation process.

Order fares: Fig. 11a shows the results when order fares are scaled by 0.5 (halved) across different charging rate and price settings. When order fares decrease, total profit falls markedly, especially at lower charging rates and prices. However, as charging rates and prices increase, V2G profit becomes the dominant component—particularly for LNS, where the V2G share reaches as high as 99.1%.

Fig. 11c shows the results when order fares are tripled. In this setting, total profit increases substantially, driven primarily by profit from orders. Although V2G profit is modest under most settings, when charging rates and prices are tripled it still contributes 19.5%–25.3% of total profit for EA and LNS. This indicates that both EA and LNS effectively optimise profit from orders, achieving comparable solution quality.

5.2.5. Effect of varying battery size

Instances: Battery capacities span a wide range across modern EVs owing to the growing heterogeneity of models (compact cars, SUVs, vans and light trucks). To reflect these real-world conditions, we evaluate the effect of battery size on EVOP–V2G performance under the default instance settings (bounding-box 40%, ride-length 10–25,km and an 8-hour time window). We consider three representative capacities aligned with popular vehicles: 40 kWh (Nissan

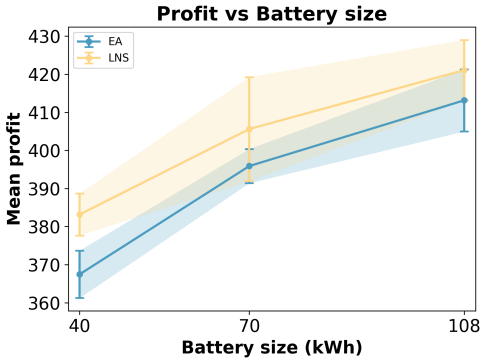


Figure 12: Effect of battery capacity on profit under the default instance settings (bounding box 40%, ride-length 10–25,km, 8-hour time window). Lines correspond to EA and LNS; the x-axis shows battery capacity (kWh) and the y-axis shows profit (AUD). Points denote means; error bars and shaded bands indicate 95% confidence intervals (CIs) computed over 10 repeats per capacity.

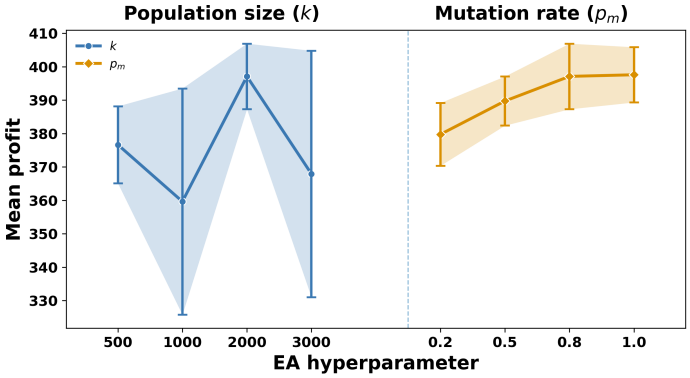


Figure 13: Effect of EA hyperparameters on profit under 100% bounding box settings (bounding box 100%, ride-length 10–25,km, 8-hour time window). The left panel shows population-size (k) levels (500, 1,000, 2,000, 3,000); the right panel shows mutation-rate (p_m) levels (0.2, 0.5, 0.8, 1.0). The x-axis lists EA hyperparameter levels, and the y-axis shows profit (AUD). Curves plot means; shaded bands and error bars indicate 95% confidence intervals.

Leaf), 70 kWh (Tesla Model S) and 108 kWh (Mercedes-Benz EQS SUV). For each battery capacity, we run EA and LNS on the default settings and aggregate results over 10 instances to report the mean profit and 95% confidence intervals (CIs), thereby quantifying both central tendency and uncertainty for each algorithm–battery combination.

Results: Fig. 12 reports total profit for EA and LNS across battery capacities of 40, 70 and 108,kWh. Both methods exhibit a clear positive slope, indicating that larger batteries yield higher profit by relaxing energy constraints and expanding flexibility for charging and discharging decisions during service. Across all capacities, LNS consistently outperforms EA, achieving higher average profit at each battery size. The error bars (95% confidence intervals, CIs, over 10 repeats) widen with capacity—short at 40,kWh and notably larger at 70,kWh for both EA and LNS—signalling increased run-to-run variability at higher capacities. Larger batteries give the EV greater flexibility to serve orders and to charge or discharge for profit. However, this flexibility also increases sensitivity to randomness; given the fixed numbers of orders and charging stations, profits show greater run-to-run variability.

5.2.6. Effect of Metaheuristic Hyperparameters

Instances: To assess the sensitivity of our metaheuristics to their design choices, we vary key hyperparameters for both EA and LNS under the largest bounding-box settings (bounding-box 100%, ride-length 10–25,km, 8-hour time window). We focus on this challenging scenario to evaluate robustness, as the larger spatial footprint increases travel distances. For each hyperparameter level, we run 10 instances and report the mean profit with 95% confidence intervals (CIs). All hyperparameters are evaluated in independent sweeps while holding all other settings fixed.

For EA, we consider population size 500, 1,000, **2,000**, 3,000 and mutation rate 0.2, 0.5, **0.8**, 1.0. For LNS, we consider Random degree (m) 2, **5**, 10, Reaction factor (α) 0.005, **0.01**, 0.05, 0.1, and Charging margin (θ) 0.05, **0.15**, 0.3. The bolded values denote the default configurations used in our main experiments.

Results: Fig. 13 summarises the effects of population size (k) and mutation rate (p_m) on profit for EA. Profit generally rises as the p_m increases from 0.2 to 0.8, indicating that moderate–high mutation enhances exploration and helps escape local optima within the available time budget. However, performance does not improve further at 1.0, suggesting that mutating every offspring is counterproductive. A similar pattern holds for k: profit improves from 500 to 2,000 individuals, then declines at 3,000. Under a 300-second budget, very large populations incur higher initialisation and evaluation costs, yielding fewer generations and thus weaker exploration. Confidence intervals (CIs) are wider at extreme settings, particularly for population size, indicating greater run-to-run variability when exploration is either too weak (small populations/low mutation) or too diffuse (very large populations). Overall, these results favour a relatively large population with a moderate–high mutation rate, while recognising that optimal choices depend on the available computational budget and problem scale.

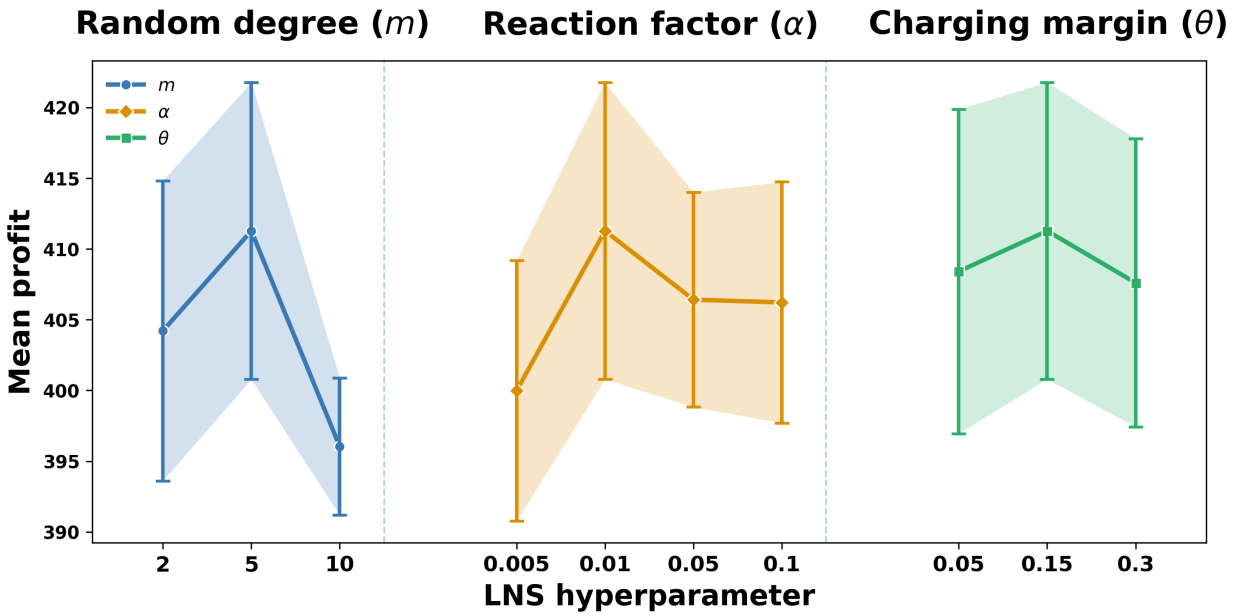


Figure 14: Effect of LNS hyperparameters on profit under bounding box size 100% settings (bounding box 100%, ride length 10–25 km, 8-hour time window). The three panels show Random degree (m) levels (2, 5, 10), Reaction factor (α) levels (0.005, 0.01, 0.05, 0.1), and Charging margin (θ) levels (0.05, 0.15, 0.3), respectively. The x-axis lists the LNS hyperparameter levels and the y-axis is profit (AUD). Curves plot means and shaded bands (with error bars) indicate 95% confidence intervals.

Fig. 14 summarizes the effects of the LNS hyperparameters on profit. For the Random degree (m), performance peaks at $m = 5$. A lower value of $m = 2$ (less randomness in greedy selection) and a higher value of $m = 10$ (more randomness) both yield lower mean profit, suggesting that $m = 5$ provides an effective balance between greedy selection and exploration. For the Reaction factor (α), profit peaks at the default value of 0.01. Performance declines for both the lower value tested ($\alpha = 0.005$) and as α increases (to 0.05 and 0.1). This indicates that while a slow, stable adaptation of strategy weights (a longer 'memory') is generally effective, adapting too slowly ($\alpha = 0.005$) may hinder learning within the time budget, just as reacting too quickly (higher α) makes the algorithm overly sensitive to noisy performance variations. Similarly, the Charging margin (θ) shows a clear peak at the default value of 0.15. This suggests that prioritising charging when the battery is below 15% (as $\theta = 0.15$ implies) is more profitable than intervening too early ($\theta = 0.3$) or waiting until the battery is nearly depleted ($\theta = 0.05$). The confidence intervals for m and θ are widest at their peak profitability points ($m = 5$ and $\theta = 0.15$), which is expected as these settings allow for more diverse solutions, leading to higher run-to-run variability. Overall, these results confirm that our chosen default values ($m = 5$, $\alpha = 0.01$, $\theta = 0.15$) are well-positioned for strong performance, capturing a near-optimal balance for each parameter's trade-off.

5.3. Discussion

In this study, we assume piecewise-constant electricity prices and a batch setting in which all orders' locations and time windows are known in advance. These choices keep the problem tractable and align with time-of-use tariffs and offline planning, but they necessarily simplify reality. Coarse price slots smooth over short-lived spikes and dips, which may slightly overstate the value of perfectly timed charging and discharging actions or miss brief opportunities. Full advance knowledge of orders likewise yields an upper bound compared with online markets featuring stochastic arrivals and cancellations. We omit battery degradation to focus on routing decisions and V2G scheduling under time-varying prices. Nevertheless, degradation is a realistic concern for EVs, especially under operating patterns characterised by frequent partial cycling at elevated state of charge and episodes of fast charging, both of which can accelerate wear.

For future work, several concrete extensions are promising. First, the single-EV setting can be generalised to coordinated multi-driver and fleet operations that explicitly model shared charging resources and charger queueing. This would enable comparative analyses of centralised (platform-optimised) versus decentralised (driver-level) dispatch, with attention to efficiency–equity trade-offs, incentive alignment and the role of charger reservations. From a problem perspective, extending EVOP-V2G to fleet scale introduces combinatorial coupling due to a larger order pool, vehicle–order assignment and charging station contention, necessitating scalable joint order selection, assignment and coordinated charging and discharging schedules. Second, uncertainty can be modelled more explicitly. Replacing piecewise-constant tariffs with forecast-driven price trajectories would enable robust and stochastic optimisation variants as well as model predictive control (MPC) formulations with recourse. Third, renewable-energy-linked tariff designs (e.g., dynamic feed-in and time-of-export rates, and negative-price events during surplus renewables) could be incorporated to co-optimise charging and discharging under high-renewables conditions, capturing arbitrage opportunities and curtailment avoidance. Beyond prices, integrating joint forecasts of order arrivals, travel times and station occupancy, together with a realistic battery degradation model, would enable a fully online, rolling-horizon EVOP–V2G formulation, effectively recasting the problem as a dynamic ride-hailing setting. From a methodological standpoint, a promising avenue for future research involves hybridising the EA by incorporating a local-search component. This approach could enhance solution quality by combining the broad search capabilities of EA with fine-grained neighbourhood refinement, potentially offering a different trade-off between exploration and exploitation compared with the standalone EA or LNS methods presented here.

6. Conclusion

This study introduces the Electric Vehicle Orienteering Problem with Vehicle-to-Grid (EVOP-V2G) as an optimization problem that jointly considers mobility and energy management decisions for profit maximisation in electric vehicle–based commercial services. By formulating an exact MIP model and developing two scalable heuristic approaches, EA and LNS, we establish a rigorous computational foundation for integrating V2G capabilities into operational planning. The experimental results confirm that our proposed algorithms substantially outperform a greedy baseline, achieving solutions with up to twice the profit and demonstrating that V2G can contribute approximately 20% of total profit under realistic settings. The MIP model provides optimal solutions for small-scale instances and serves as a performance benchmark, while EA and LNS yield near-optimal, scalable solutions for large problem sizes involving up to 900 orders and 70 charging stations. Among these, LNS exhibits superior convergence and solution quality, highlighting its suitability for practical deployment in large-scale, data-driven environments. Sensitivity analyses further reveal that increases in charging/discharging rates or decreases in order profitability enhance the relative benefits of V2G, underscoring its growing economic importance as charging infrastructure and dynamic energy markets mature. These findings demonstrate that integrating bidirectional charging into routing decisions can significantly improve both driver profitability and system-wide energy efficiency.

Future work can extend the EVOP-V2G to multi-EV and fleet-level settings, where multiple vehicles coordinate order allocation, routing, and energy trading under shared charging infrastructure and grid constraints. Such extensions would enable the study of inter-vehicle interactions, cooperative charging strategies, and dynamic fleet scheduling under stochastic demand and price conditions. In addition, incorporating forecast-driven, renewable-energy-linked tariff designs, moving from offline planning to online, and modelling battery degradation would further enhance the model's applicability to real-world mobility platforms, contributing to the development of coordinated, sustainable, and grid-aware electric transportation systems. Overall, this research establishes a foundational step toward the joint optimisation of mobility and energy decisions in electric vehicle systems, paving the way for next-generation fleet intelligence and data-driven, sustainable energy-transport integration.

CRediT authorship contribution statement

Jinchun Du: Conceptualization, Data curation, Methodology, Software, Validation, Formal analysis, Visualization, Writing-original draft, Writing-review & editing. **Bojie Shen**: Conceptualization, Methodology, Software, Writing-review & editing. **Muhammad Aamir Cheema**: Conceptualization, Supervision, Methodology, Project Administration, Writing-review & editing. **Adel N.Toosi**: Conceptualization, Supervision, Methodology, Writing-review & editing.

Funding

Muhammad Aamir Cheema and Adel N. Toosi are supported by the Australian Research Council (ARC) [DP230100081].

Data availability

Data will be made available on request.

Declaration of generative AI and AI-assisted technologies in the writing process.

During the preparation of this work the author(s) used ChatGPT in order to assist with grammatical checks and presentation improvements. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the published article.

References

[1] A. Rivera, S. Movalia, E. Rutkowski, Global greenhouse gas emissions: 1990–2022 and preliminary 2023 estimates, Tech. rep., Rhodium Group, accessed: 2025-06-25 (November 2024).
URL https://rhg.com/research/global-greenhouse-gas-emissions-1990-2022-and-preliminary-2023-estimates/

[2] Department of Climate Change, Energy, the Environment and Water (DCCEEW), Australia's emissions projections 2023, Tech. rep., Australian Government, accessed: 2024-03-29 (2023).
URL https://www.dcceew.gov.au/climate-change/publications/australias-emissions-projections-2023

[3] United Nations, Transforming our world: The 2030 agenda for sustainable development, https://sdgs.un.org/2030agenda, accessed: 2025-04-30 (2015).

[4] M. A. Cheema, H. Wang, W. Wang, A. N. Toosi, E. Tanin, J. Qi, H. Samet, Beyond the commute: Unlocking the potential of electric vehicles as future energy storage solutions (vision paper), in: ACM SIGSPATIAL, 2024.

[5] M. Zalesak, S. Samaranayake, Real time operation of high-capacity electric vehicle ridesharing fleets, Transportation Research Part C: Emerging Technologies 133 (2021) 103413.

[6] L. d. C. Martins, R. D. Tordecilla, J. Castaneda, A. A. Juan, J. Faulin, Electric vehicle routing, arc routing, and team orienteering problems in sustainable transportation, Energies 14 (16) (2021) 5131.

[7] R. Chen, X. Liu, L. Miao, P. Yang, Electric vehicle tour planning considering range anxiety, Sustainability 12 (9) (2020) 3685.

[8] G. B. Dantzig, J. H. Ramser, The truck dispatching problem, Management science 6 (1) (1959) 80–91.

[9] C. Archetti, L. Coelho, M. Speranza, P. Vansteenwegen, Beyond fifty years of vehicle routing: Insights into the history and the future, European Journal of Operational Research (2025).

[10] N. Christofides, The vehicle routing problem, Combinatorial optimization (1979).

[11] J. Paquette, J.-F. Cordeau, G. Laporte, Quality of service in dial-a-ride operations, Computers & Industrial Engineering 56 (4) (2009) 1721–1734.

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

[13] I. Kucukoglu, R. Dewil, D. Cattrysse, The electric vehicle routing problem and its variations: A literature review, Computers & Industrial Engineering 161 (2021) 107650.

[14] C. Lee, An exact algorithm for the electric-vehicle routing problem with nonlinear charging time, Journal of the Operational Research Society 72 (7) (2021) 1461–1485.

[15] A. Abdulaal, M. H. Cintuglu, S. Asfour, O. A. Mohammed, Solving the multivariant ev routing problem incorporating v2g and g2v options, IEEE Transactions on Transportation Electrification 3 (1) (2016) 238–248.

[16] S. Hulagu, H. B. Celikoglu, An electric vehicle routing problem with intermediate nodes for shuttle fleets, IEEE Transactions on Intelligent Transportation Systems 23 (2) (2020) 1223–1235.

[17] A. Ceselli, Á. Felipe, M. T. Ortuño, G. Righini, G. Tirado, A branch-and-cut-and-price algorithm for the electric vehicle routing problem with multiple technologies, in: Operations Research Forum, Vol. 2, Springer, 2021, pp. 1–33.

[18] L. Wang, Y. Ding, Z. Chen, Z. Su, Y. Zhuang, Heuristic algorithms for heterogeneous and multi-trip electric vehicle routing problem with pickup and delivery, World Electric Vehicle Journal 15 (2) (2024) 69.

[19] R. Basso, B. Kulcsár, I. Sanchez-Diaz, Electric vehicle routing problem with machine learning for energy prediction, Transportation Research Part B: Methodological 145 (2021) 24–55.

[20] J. Chen, M. Qi, L. Miao, The electric vehicle routing problem with time windows and battery swapping stations, in: 2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), IEEE, 2016, pp. 712–716.

[21] M. Schneider, A. Stenger, D. Goeke, The electric vehicle-routing problem with time windows and recharging stations, Transportation science 48 (4) (2014) 500–520.

[22] E. Lam, G. Desaulniers, P. J. Stuckey, Branch-and-cut-and-price for the electric vehicle routing problem with time windows, piecewise-linear recharging and capacitated recharging stations, Computers & Operations Research 145 (2022) 105870.

[23] U. Breunig, R. Baldacci, R. F. Hartl, T. Vidal, The electric two-echelon vehicle routing problem, Computers & Operations Research 103 (2019) 198–210.
[24] S. Karakatič, Optimizing nonlinear charging times of electric vehicle routing with genetic algorithm, Expert Systems with Applications 164 (2021) 114039.
[25] Ö. Aslan Yıldız, İ. Sarıçiçek, A. Yazıcı, A reinforcement learning-based solution for the capacitated electric vehicle routing problem from the last-mile delivery perspective, Applied Sciences 15 (3) (2025) 1068.
[26] E. Rodríguez-Esparza, A. D. Masegosa, D. Oliva, E. Onieva, A new hyper-heuristic based on adaptive simulated annealing and reinforcement learning for the capacitated electric vehicle routing problem, Expert Systems with Applications 252 (2024) 124197.
[27] J. Ruiz-Meza, J. R. Montoya-Torres, A systematic literature review for the tourist trip design problem: Extensions, solution techniques and future research lines, Operations Research Perspectives 9 (2022) 100228.
[28] J. Karbowska-Chilinska, K. Chociej, Genetic algorithm for generation multistage tourist route of electrical vehicle, in: International Conference on Computer Information Systems and Industrial Management, Springer, 2020, pp. 366–376.
[29] J. Lee, S.-W. Kim, G.-L. Park, A tour recommendation service for electric vehicles based on a hybrid orienteering model, in: Proceedings of the 28th Annual ACM Symposium on Applied Computing, 2013, pp. 1652–1654.
[30] Y.-W. Wang, C.-C. Lin, T.-J. Lee, Electric vehicle tour planning, Transportation Research Part D: Transport and Environment 63 (2018) 121–136.
[31] F. Mufalli, R. Batta, R. Nagi, Simultaneous sensor selection and routing of unmanned aerial vehicles for complex mission plans, Computers & Operations Research 39 (11) (2012) 2787–2799.
[32] J. Panadero, A. A. Juan, C. Bayliss, C. Currie, Maximising reward from a team of surveillance drones: A simheuristic approach to the stochastic team orienteering problem, European Journal of Industrial Engineering 14 (4) (2020) 485–516.
[33] A. Narayanan, P. Misra, A. Ojha, V. Bandhu, S. Ghosh, A. Vasan, A reinforcement learning approach for electric vehicle routing problem with vehicle-to-grid supply, arXiv preprint arXiv:2204.05545 (2022).
[34] B. Lin, B. Ghaddar, J. Nathwani, Electric vehicle routing with charging/discharging under time-variant electricity prices, Transportation Research Part C: Emerging Technologies 130 (2021) 103285.
[35] B. Shen, J. Du, M. A. Cheema, Beyond transport: V2x integration turning evs into smart energy assets (2025).
[36] S. Sagaria, M. van der Kam, T. Boström, Vehicle-to-grid impact on battery degradation and estimation of v2g economic compensation, Applied Energy 377 (2025) 124546.
[37] P. Yang, Y. Cao, J. Tan, J. Chen, C. Zhang, Y. Wang, H. Liang, Electric vehicle charging capacity of distribution network considering conventional load composition, Energy Engineering: Journal of the Association of Energy Engineers 120 (3) (2023) 743.
[38] A. Goncearuc, C. De Cauwer, N. Sapountzoglou, G. Van Kriekinge, D. Huber, M. Messagie, T. Coosemans, The barriers to widespread adoption of vehicle-to-grid: A comprehensive review, Energy Reports 12 (2024) 27–41.
[39] M. I. Al-Amin, J. Du, M. A. Cheema, I. F. Siddiqui, M. Salehi, Ev energy trading dashboard: Cost-emission reduction through spatiotemporal forecasts and smart charging, in: Proceedings of the 33nd ACM International Conference on Advances in Geographic Information Systems, 2025.
[40] S. Zhang, Y. Gajpal, S. Appadoo, M. Abdulkader, Electric vehicle routing problem with recharging stations for minimizing energy consumption, International journal of production economics 203 (2018) 404–413.
[41] R. H. Mladineo, Model building in mathematical programming (H. p. williams), SIAM Rev. 36 (2) (1994) 313–315.
URL https://doi.org/10.1137/1036082
[42] B. L. Miller, D. E. Goldberg, et al., Genetic algorithms, tournament selection, and the effects of noise, Complex systems 9 (3) (1995) 193–212.
[43] E. D. Goodman, Introduction to genetic algorithms, in: D. V. Arnold, E. Alba (Eds.), Genetic and Evolutionary Computation Conference, GECCO '14, Vancouver, BC, Canada, July 12-16, 2014, Companion Material Proceedings, ACM, 2014, pp. 205–226. doi:10.1145/2598394.2605335.
URL https://doi.org/10.1145/2598394.2605335
[44] P. Shaw, Using constraint programming and local search methods to solve vehicle routing problems, in: International conference on principles and practice of constraint programming, Springer, 1998, pp. 417–431.
[45] S. Ropke, D. Pisinger, An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows, Transportation science 40 (4) (2006) 455–472.
[46] D. E. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison-Wesley, 1989.
[47] A. Najmi, D. Rey, T. H. Rashidi, Novel dynamic formulations for real-time ride-sharing systems, Transportation research part E: logistics and transportation review 108 (2017) 122–140.
[48] Canstar Blue, Ridesharing apps in australia: Which is best?, https://www.canstarblue.com.au/travel/ridesharing-apps-australia/, accessed: 2023-09-15 (2023).
[49] Essential Services Commission of Victoria, Victorian default offer price review 2023-24, https://www.esc.vic.gov.au/electricity-and-gas/prices-tariffs-and-benchmarks/victorian-default-offer/victorian-default-offer-price-review-2023-24, accessed: 2023-11-28 (2023).
[50] J. Zatsarnaja, K. Reiter, M. Mehdizadeh, A. Nayum, T. Nordfjaern, Charged up with peace in mind: Unraveling the factors of range anxiety among norwegian electric vehicle drivers, Transportation Research Part F: Traffic Psychology and Behaviour 110 (2025) 15–28.