TERRAN: A Transformer-based Electric Vehicle Routing Agent for Real-time Adaptive Navigation

Maojie Tang, Nanpeng Yu, Ioannis Karamouzas, and Zuzhao Ye

***Abstract*—The Electric Vehicle Routing Problem with Time Windows (EVRP-TW) poses significant challenges for sustainable logistics due to its tight coupling of spatial, temporal, and energy constraints. Classical optimization methods face trade-offs: exact solvers like CPLEX ensure optimality but require prohibitive runtimes, while metaheuristics like Variable Neighborhood Search struggle with feasibility under complex constraints. We propose TERRAN, a transformer-based reinforcement learning framework for real-time and scalable EVRP-TW optimization. TERRAN integrates three key components: (1) *Future-Feasibility Pruning (FFP)*, which proactively eliminates energy-infeasible actions by verifying reachability to charging stations or depots before each move; (2) *Staged Reward Scheduling*, which progressively transitions from dense auxiliary signals to task-aligned rewards to guide the agent from achieving feasibility to minimizing cost; and (3) *An End-to-End Transformer-Based RL Agent Tailored for EVRP-TW*, which directly integrates EV-specific constraints—including battery consumption, charging decisions, and delivery time windows—into the policy network and decoding process, enabling unified, post-processing-free optimization across varying instance scales. Experiments on Solomon benchmark instances with 5–100 customers demonstrate that TERRAN achieves 100% feasibility across all problem scales. It matches CPLEX's optimality on 5–customer instances, achieves up to 170,000× speedups with solutions within 1.5% of optimal on 15–customer instances (0.02 s vs. 3,500 s), and delivers feasible solutions for 100–customer instances in 0.47 s, where CPLEX fails on over 80% of cases within 1 hour. These results establish TERRAN as a practical and scalable solution for real-time electric vehicle routing in complex and constraint-rich environments.**

NOTE TO PRACTITIONERS

This paper addresses the challenge of real-time route planning for electric vehicles (EVs), which is critical for logistics platforms such as Amazon and FedEx that must quickly generate delivery routes while managing battery constraints, charging logistics, and strict customer time windows. We propose TERRAN, a deep reinforcement learning framework that produces feasible EV routing plans within seconds, achieving computational speeds that are orders of magnitude faster than classical solvers like CPLEX while maintaining solution quality far superior to traditional heuristics. By enforcing hard feasibility constraints during decision-making, TERRAN guarantees that all generated routes satisfy operational requirements without the need for post-processing or repair. Although the current implementation assumes simplified vehicle dynamics—including constant travel speed, fixed charging rates, and a compulsory full charging strategy—these idealized assumptions allow efficient initial modeling but limit applicability to more dynamic real-world environments. Future work will address these limitations by introducing nonlinear battery models, dynamic traffic conditions, and flexible partial charging strategies. Beyond last-mile delivery, the proposed approach may also benefit broader applications in smart city logistics, dynamic fleet management, and intelligent energy scheduling for electric vehicle networks.

***Index Terms*—Electric Vehicle Routing, Time Windows, Constrained Reinforcement Learning, Transformer Architectures.**

I. INTRODUCTION

THE rapid adoption of electric vehicles (EVs) in last-mile delivery systems is reshaping logistics, driven by growing environmental concerns and the demand for sustainable operations [1], [2]. Platforms like Amazon Flex and FedEx SameDay, which generate routes within minutes of order finalization, exemplify the urgency of intelligent routing strategies that balance delivery schedules with EV battery management. These routing decisions directly affect operational costs, delivery efficiency, and energy consumption, making real-time and feasibility-conscious planning indispensable [3]. Studies indicate that optimized EV routing can reduce last-mile delivery costs by 10% [4] and achieve energy savings ranging from 20% to 51% [5], [6], underscoring its economic and environmental impact.

At the heart of this challenge lies the **Electric Vehicle Routing Problem with Time Windows (EVRP-TW)**, an extension of the classical Vehicle Routing Problem (VRP) [7]. The VRP seeks to minimize travel costs while meeting customer demands. EVRP-TW introduces three additional groups of tightly coupled constraints that exacerbate the *NP-hardness* [8] and practical complexity of VRP. The first group of constraints imposes temporal dependencies by modeling *delivery time windows* that require each customer to be served within a specified interval [9]. The second group of constraints focuses on *battery energy consumption* and necessitates strategic planning of charging station (CS) visits to ensure the vehicle's state of charge (SoC) remains within operational limits [10]. Note that some of the energy related constraint may be relaxed if vehicle-to-vehicle energy sharing and trading is allowed. In such a setting, a blockchain- and PKI-based secure vehicle-to-vehicle energy-trading protocol can further enhance system security by ensuring authenticated participation, transaction

M. Tang and I. Karamouzas are with the Department of Computer Science and Engineering, University of California, Riverside, Riverside, CA 92521 USA (e-mail: mtang096@ucr.edu; ioannisk@cs.ucr.edu).

N. Yu and Z. Ye are with the Department of Electrical and Computer Engineering, University of California, Riverside, Riverside, CA 92521 USA (e-mail: nyu@ece.ucr.edu; zye066@ucr.edu).

Corresponding author: Nanpeng Yu.

This work was supported by the National Science Foundation under Award No. 2324940 and by the University of California Office of the President under Award No. L22CR4556.

integrity, and tamper-proof settlement [11]. The third group of constraints models *charging delays*, capturing the time and operational costs associated with charging activities. These additional groups of constraints render many traditional optimization methods impractical for *real-time* EVRP-TW, where platforms demand feasible solutions in minutes [12].

To this end, we propose **TERRAN** (**T**ransformer-based **E**lectric vehicle **R**outing agent for **R**eal-time **A**daptive **N**avigation), a reinforcement learning framework designed to generate efficient, scalable, and feasible EV routes under time and energy constraints in dynamic, real-world environments.

A. Existing Approaches and Limitations

Existing approaches to EVRP-TW fall into three categories: *exact solvers*, *metaheuristics*, and *deep reinforcement learning (DRL)*. While each has achieved notable progress, none fully satisfies the combined needs of real-time decision-making, scalability, and strict constraint feasibility.

Exact solvers, such as MILP formulations [13] and branch-price-and-cut frameworks [14], guarantee global optimality by exhaustively exploring the energy–time solution space. These methods can address partial charging and tight spatiotemporal coupling but require extensive engineering effort, e.g., hand-crafted dominance rules and labeling procedures. Moreover, they scale poorly [15], often necessitate dummy nodes to handle multiple charging station visits [16], and lack open-source implementations, limiting their applicability in real-time logistics. Among these, commercial solvers like **CPLEX** are widely used for their robustness but remain impractical for large instances under tight time budgets.

Metaheuristics, including ALNS [17], Tabu Search [18], and ACO [19], reduce computational burden via heuristic search but rely heavily on domain-specific rules and hyperparameter tuning. Some methods support partial recharging [20] or heterogeneous fleets [21], [22], yet most still operate on static structures and require manual feasibility recovery [23]. Reproducibility remains a challenge due to limited code availability and lack of standard benchmarks.

Deep Reinforcement Learning (DRL) solvers for routing are typically cast as a *centralized dispatcher* that constructs routes sequentially: at each step, the next node is selected, and once the current vehicle returns to the depot, a new route is initiated and the process repeats [24], [25]. This yields a discrete, stepwise decision process that aligns naturally with sequence-based backbones such as RNN/LSTM/Pointer decoders [26], [27]. In the case of EVRP-TW, however, the dispatcher must additionally ensure both time-window and state-of-charge (SoC) feasibility. Thus, each decision concerns not only *which node to visit next*, but also *whether the move is reachable* under energy and time constraints—commonly enforced through *hard feasibility masks* and, in some approaches, *post hoc* repair mechanisms [28], [29].

To better understand the evolution and current capabilities of DRL-based EVRP-TW solvers, we next discuss the progression of their architectural backbones and learning frameworks, and how these developments relate to broader trends in transportation electrification.

Backbone evolution: from RNN/Pointer to Transformer. Early neural decoders adopted *RNN/LSTM/Pointer* architectures [26], [27], which aligned well with the sequential nature of routing but suffered from limited long-range context and weaker parallelism. As attention models matured, *Transformer-style* backbones became prevalent for VRPs [25] and have since been extended to EV variants through edge- and energy-aware encoder-decoder designs [29]. Compared with RNNs, Transformers provide three key advantages: (i) *token-wise global context* at each decoding step, which is critical for long-horizon feasibility considerations (e.g., depot/CS/customer triads); (ii) *clean hard masking*, allowing direct injection of TW and SoC constraints at the logit level; and (iii) inherently *parallelizable* training, essential for real-time dispatch. Empirically, Transformer-based EV routing has shown improved feasibility and scalability [29]; related attention-based models have also advanced *heterogeneous vehicle routing problems (HVRP)* [30]–[32]; while energy-focused EVRP studies consistently report gains with attention backbones [33]. These properties collectively motivate our choice of a Transformer backbone for TERRAN.

Reinforcement learning framework: policy-gradient and beyond. On the learning side, most neural routing solvers rely on *policy-gradient* training with variance reduction techniques. For EVRP(-TW), both value-based DQN variants [28], [34] and end-to-end *policy-gradient* approaches [30]–[33], [35], [36] have been explored. DRL-TS [29] further proposes a two-stage curriculum that combines graph attention with edge features and separates soft versus hard constraint enforcement across phases. Despite these advances, several challenges remain. Reward functions are often hand-crafted, risking misalignment with long-horizon routing objectives [24], [25]. Feasibility mechanisms typically depend on soft penalties or static masks, which fail to capture dynamic SoC reachability [29]. Moreover, large-scale instances frequently suffer from degraded solution quality or incur substantial decoding overhead due to feasibility recovery [28]. To address these limitations, our method integrates a Transformer backbone with *constraint-aware attention* and explicit *future-reachability* modeling under a PPO/GAE objective. This design improves feasibility and training stability, while maintaining real-time inference efficiency.

Relation to trajectory optimization in transportation electrification. Beyond routing, a parallel line of research in transportation electrification focuses on optimizing *continuous-time* vehicle trajectories and grid–EV co-scheduling. Recent studies include the joint optimization of distribution networks and EV trajectories with explicit carbon-emission allocation [37], as well as eco-driving speed-trajectory optimization using optimal control or dynamic programming that minimizes traction energy [38], [39]. These works address a different decision layer—optimizing speed profiles and charging schedules along a *given path* or network operating point—whereas EVRP-TW targets *discrete node sequencing and charging station selection under TW/SoC constraints*. The two classes of approaches are complementary: trajectory-level results can refine arc-level energy and time estimates or serve as speed advisors within a dispatch

framework, while EVRP-TW determines which customers and CSs to visit and when.

Summary. Exact solvers guarantee optimality at the expense of scalability; metaheuristics trade computational speed for robustness and generalization; and existing DRL methods, though promising, exhibit critical limitations in reward shaping and feasibility enforcement. These challenges underscore the need for a scalable DRL framework that can ensure feasibility, improve training stability, and deliver high-quality real-time solutions—motivating our proposed TERRAN.

B. Our Contributions

We propose **TERRAN**, a constrained RL framework for real-time EVRP-TW that explicitly addresses spatiotemporal and energy constraints. TERRAN makes four contributions:

1) **First publicly available deep RL framework and reproducible benchmark for EVRP-TW.** To the best of our knowledge, there is no prior reproducible end-to-end DRL framework for EVRP-TW with matched constraints, public training scripts, and data generators. We release a repository with training code, data-generation tools, standardized instance sets, and solver baselines, establishing a transparent, reproducible, and extensible foundation for future research: https://github.com/NanpengYu/TERRAN.
2) **Future-Feasibility Pruning (FFP).** We introduce a look-ahead action-pruning mechanism that removes energy-infeasible actions by verifying reachability to a depot or charging station after each decision step. This rule is provably feasibility-preserving, shrinks the search space, and improves both training stability and inference efficiency.
3) **Staged Reward Scheduling.** We design a multi-stage reward schedule that guides learning from feasibility discovery to optimizing the primary objective (total travel distance). Early dense signals (e.g., for customer service or charging) mitigate sparse feedback and accelerate convergence.
4) **Unified Transformer-based, constraint-aware policy for EVRP-TW.** We integrate battery consumption, charging operations, and time-window constraints directly into decoding/masking within a Transformer-based policy, enabling end-to-end solution generation without ad-hoc post-processing.

TERRAN achieves up to **170,000× speedups** over CPLEX on 15-customer instances (0.02 s vs. 3,500 s), while maintaining solutions within 1.5% of the optimal. On 100-customer benchmarks, TERRAN generates 100% feasible solutions in just 0.47 s, whereas CPLEX fails to return a feasible solution in over half the cases within the 1-hour time limit. Across all problem scales, TERRAN consistently produces high-quality, constraint-compliant routes with exceptional computational efficiency and can serve as a transparent and extensible baseline for future work on EVRP-TW problems.

The rest of the paper is organized as follows. Section II models the EVRP-TW as a constrained graph-based optimization problem with spatiotemporal dynamics. Section III reformulates it as a Markov Decision Process (MDP) to enable RL solutions. Section IV introduces the TERRAN framework, detailing the model architecture, masking scheme, and multi-stage reward design. Section V presents experimental results on Solomon benchmark instances, alongside ablation studies. Section VI concludes the paper and discusses future work directions for scalability and real-world deployment.

TABLE I
LIST OF NOTATIONS

Symbol	Description
V	Set of all nodes (depot, customers, charging stations)
V_d	Depot node set (typically $\{0\}$)
V_c	Set of customer nodes
V_s	Set of charging station (CS) nodes
d_{ij}	Euclidean distance between node i and j
ω_{ij}	Travel time between node i and j; linearly proportional to d_{ij}
Q	Maximum battery capacity of EVs
C	Cargo capacity of EVs
g	Charging rate
η	Energy consumption rate per unit time
(x_i, y_i)	2-D spatial coordinates of node i
$[e_i, l_i]$	Time window at node i (arrival bounds)
q_i	Demand of customer i (0 for depot and CS)
t_{ser}	Constant service time at customer nodes
$T_{\max}$	Upper bound of global time horizon
$E_{\min}(i)$	Minimum energy required to reach a depot or CS from node i
SoC^t	Battery level (state-of-charge) at step t
c^t	Remaining cargo capacity at step t
τ^t, $\tilde{\tau}^t$	Departure time and arrival time at step t
S^t	System-level state tuple: $(\tau^t, \text{SoC}^t, c^t)$
s_t	RL state composed of: partial route, S^t, and current node u_t
a_t	Action at step t (selecting next node u_{t+1} to visit)
u_t	Node visited at step t (before taking a_t)
$\mathcal{A}$	Action space (candidate nodes at each step)
$M_t(i)$	Feasibility mask for node i at time t
Π	Permutation of visited nodes (full routing plan)
R_k	Route of vehicle k as a subsequence of Π

II. PROBLEM FORMULATION

EVRP-TW, introduced by Schneider et al. [13], extends the classical VRP by incorporating EV constraints involving battery capacity and delivery time windows (see Appendix A). In our setting, a central dispatcher agent plans routes for multiple EVs sequentially, where each vehicle operates independently but is coordinated centrally through a unified sequential decoding process. As in the standard benchmark setup, we adopt a homogeneous and unlimited fleet of vehicles, facilitating comparability and reproducibility with prior work. Table I summarizes the key notations used throughout the paper.

Following prior work on RL-based combinatorial optimization [25], we represent an EVRP-TW solution as a sequence of node selections constructed in an autoregressive decoding process. At each decoding step t, the vehicle observes the current system state and selects the next node $u_{t+1} \in V$ to visit, subject to feasibility constraints.

To support this sequential decision process, we formalize the problem on a complete directed graph $\mathcal{G} = (V, E)$, where the node set is defined as $V = V_d \cup V_c \cup V_s$, consisting of:

- **Depot** ($V_d = \{0\}$): the origin and destination for all EVs,

- **Customers** ($V_c = \{1, \ldots, n_c\}$): demand nodes with service time windows,
- **Charging Stations** ($V_s = \{n_c + 1, \ldots, n_c + n_s\}$): locations where vehicles must charge to full battery capacity before departure.

Each node $i \in V$ is represented by a 5-dimensional static feature vector:

$$X_i = \begin{cases} (x_i, y_i, 0, 0, T_{\max}), & \text{if } i \in V \setminus V_c \\ (x_i, y_i, q_i, e_i, l_i), & \text{if } i \in V_c, \end{cases}$$

where (x_i, y_i) denotes 2-D spatial coordinates of node i, q_i is the demand at node i (zero for depot and CS), $[e_i, l_i]$ represents the time window for serving the customer at node i.

For depot and CS nodes, demand and time window are set to $(0, 0, T_{\max})$ to standardize dimensionality. This unified representation enables consistent encoding of heterogeneous node types. The arc set is defined as $E = \{(i, j) \mid i, j \in V\}$, where the travel distance between node i and j is given by $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

A. Dynamic System State: Per-Vehicle Modeling

Model Assumptions. Following the classical EVRP-TW formulation [13], we adopt the following assumptions:

- EVs travel at a constant speed, making travel time ω_{ij} linearly proportional to the Euclidean distance d_{ij}.
- Energy consumption is proportional to travel distance, and since EVs travel at a constant speed, we express it as proportional to travel time, with a fixed rate η (kWh/min).
- The charging rate g (kWh/min) of EVs at CSs is constant.
- Each charging session restores the battery to full capacity; partial charging is not allowed.
- All customer nodes require a fixed service time t_{ser}, while depot and charging stations incur no service delays.

These assumptions govern the evolution of each EV's operational status throughout its assigned route. Since the agent constructs routes sequentially, we model the per-vehicle dynamics using a system state S^t at each decoding step t, capturing the current EV's time, battery, and remaining cargo capacity:

$$S^t = (\tau^t, \text{SoC}^t, c^t), \tag{1}$$

where $\tau^t \in [0, T_{\max}]$ denotes system time at step t, representing the vehicle's departure time from node u_t after completing service or charging, $\text{SoC}^t \in [0, Q]$ denotes the state-of-charge of the EV battery, and $c^t \in [0, C]$ denotes the EV's remaining cargo capacity.

This state representation encodes three core operational constraints of EVRP-TW: temporal deadlines, energy feasibility, and remaining load capacity.

Operational Constraints: Feasibility Conditions

1) *Time Window:* For any customer node $u_{t+1} \in V_c$, the vehicle must arrive within the service time window:

$$e_{u_{t+1}} \le \tau^t + \omega_{u_t, u_{t+1}} \le l_{u_{t+1}}, \quad t \in [0,\, T_{\max} - 1] \tag{2}$$

2) *Energy Feasibility:* The EV battery must have enough energy to reach the next node:

$$\text{SoC}^t \ge \eta \cdot \omega_{u_t, u_{t+1}}, \quad t \in [0,\, T_{\max} - 1] \tag{3}$$

3) *Cargo Capacity:* To serve a customer $u_{t+1} \in V_c$, the vehicle must have sufficient capacity:

$$c^t \ge q_{u_{t+1}}, \quad t \in [0,\, T_{\max} - 1]. \tag{4}$$

4) *Depot Return:* Each route must terminate at a depot $u_T \in V_d$ within the time horizon:

$$\tilde{\tau}_T \le T_{\max} \tag{5}$$

Solution Representation: Permutation-Based Routing

A feasible solution to EVRP-TW is represented by a permutation of node indices: $\Pi = (u_0, u_1, u_2, \ldots, u_m), u_i \in V$, where depot nodes (i.e., node 0) act as delimiters separating individual vehicle routes. This format allows a single agent to construct routes for multiple EVs in a unified decoding sequence. Each sub-sequence between depot nodes corresponds to a distinct vehicle's route, executed in parallel in the actual delivery process. The total number of vehicles is thus determined by the number of depot returns rather than being fixed a priori. For illustration, consider an instance with 5 customers (nodes 1–5), one CS (node 6), and a single depot (node 0). The permutation $\Pi = \{0, 1, 6, 2, 4, 0, 5, 0\}$ defines two vehicle routes: $R_1 : 0 \to 1 \to 6 \to 2 \to 4 \to 0$, $R_2 : 0 \to 5 \to 0$.

This permutation-based structure provides a unified representation that naturally incorporates depot returns and optional charging station stops. Each sub-route must satisfy all operational constraints, including time-window feasibility, battery capacity, and cargo limits. While routes are generated sequentially within a single permutation, they are executed concurrently by multiple vehicles in practice. Each depot token terminates the current route and resets a route-local clock τ to zero for the next one, ensuring that all temporal constraints are evaluated with respect to the local clock of the vehicle serving each customer.

Objective Function

The objective is to minimize the total travel distance across all EV routes:

$$\min_{\Pi \in \mathcal{F}} \sum_{k=1}^{K} \sum_{(i,j) \in R_k} d_{ij} \tag{6}$$

where $\mathcal{F}$ denotes the set of node permutations satisfying constraints (2)–(5).

This distance-based objective follows the classical EVRP-TW formulation [13], where travel distance is widely adopted due to its correlation with travel time, energy usage, and operational cost. Our work retains this standard assumption while focusing on constrained policy learning.

III. EVRP-TW AS AN RL PROBLEM

We model EVRP-TW as a finite horizon Markov Decision Process (MDP) with the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, H_{\max}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the dynamics model, r is the reward function, $H_{\max}$ denotes the step horizon, and $\gamma \in (0, 1]$ is the discount factor. Our goal is to find a policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$, represented as a neural network with weights θ, which maximizes the expected return, i.e., the discounted cumulative reward of the states visited: $G_t = \sum_{k=0}^{H_{\max}-t} \gamma^k r_{t+k}$.

State Space Design

At each decoding step t, the state $s_t \in \mathcal{S}$ represents the routing progress and vehicle status: $s_t = (\Pi_{1:t-1}, S^t, u_t)$, where $\Pi_{1:t-1}$ denotes the sequence of nodes visited so far, $S^t = (\tau^t, \mathrm{SoC}^t, c^t)$ is the current dynamic system state, and $u_t \in V$ is the current node index. An episode terminates when all customer nodes in V_c have been visited and the agent has constructed a complete sequence of vehicle routes, each starting and ending at the depot. Although the MDP operates in a sequential decision-making mode, the underlying solution corresponds to a parallel execution across multiple vehicles. The agent implicitly determines when to dispatch a new vehicle by inserting a depot node into the sequence, thereby eliminating the need for a predefined fleet size.

Action Space Design

The action $a_t \in \mathcal{A}$ corresponds to selecting the next node $u_{t+1} \in V$ to visit. Infeasible nodes are dynamically masked based on time window, battery, and cargo capacity constraints. The mask $M_t(i)$ is computed such that:

$$M_t(i) = \begin{cases} 0, & \text{if } \tilde{\tau}^{t+1} > l_i \text{ (time window violated)} \\ 0, & \text{if } \mathrm{SoC}^t - \eta \cdot \omega_{u_t,i} < 0 \text{ (insufficient energy)} \\ 0, & \text{if } c^t < q_i \text{ (infeasible demand)} \\ 1, & \text{otherwise} \end{cases}$$

This mask ensures single-step feasibility based on one-step ahead constraints. Future feasibility, such as ensuring sufficient battery energy to reach a subsequent charging station or the depot, is addressed in Section IV.

Environment Transition Dynamics

Given state s_t and a valid action $a_t = u_{t+1}$, the next state s_{t+1} is deterministically defined: $\mathcal{P}(s_{t+1} \mid s_t, a_t) = 1$. The updates are as follows:

The time-related state variables are updated as follows:
Arrival time:

$$\tilde{\tau}^{t+1} = \tau^t + \omega_{u_t,u_{t+1}}, \quad u_{t+1} \in V$$

Departure time:

$$\tau^{t+1} = \begin{cases} \max(e_{u_{t+1}}, \tilde{\tau}^{t+1}) + t_{\text{ser}}, & u_{t+1} \in V_c, \\ \tilde{\tau}^{t+1} + \dfrac{Q - (\mathrm{SoC}^t - \eta \cdot \omega_{u_t,u_{t+1}})}{g}, & u_{t+1} \in V_s, \\ 0, & u_{t+1} \in V_d. \end{cases}$$

The battery state-of-charge is updated as follows:

$$\mathrm{SoC}^{t+1} = \begin{cases} \mathrm{SoC}^t - \eta \cdot \omega_{u_t,u_{t+1}}, & u_{t+1} \notin V_s \cup V_d \\ Q, & u_{t+1} \in V_s \cup V_d \end{cases}$$

The cargo state is updated as follows:

$$c^{t+1} = \begin{cases} c^t - q_{u_{t+1}}, & u_{t+1} \in V_c \\ C, & u_{t+1} \in V_d \\ c^t, & u_{t+1} \in V_s \end{cases}$$

If $u_{t+1} \in V_d$, the vehicle is reset: time is re-initialized, the battery is fully recharged, and cargo capacity is restored. The episode terminates once all customers in V_c have been visited exactly once and the final route returns to the depot, ensuring that the constructed permutation Π consists of complete, depot-terminated routes. Early termination due to constraint violations does not occur, as infeasible actions are proactively masked by the environment.

Reward Function Design

Let $\Delta d_t \geq 0$ denote the travel distance incurred when moving from the current node u_t to the next node u_{t+1} (i.e., the action $a_t := u_{t+1}$). The per-step reward is defined as

$$\begin{aligned} r_t \;=\; & -\,\Delta d_t && (7) \\ & + r_{\text{re}}\,\mathbf{1}\{u_t \in V_s,\ \mathrm{SoC}^t \leq \theta_{\text{SoC}},\ u_{t-1} \notin V_s\} && (8) \\ & + r_{\text{ser}}\,\mathbf{1}\{u_t \in V_c\} && (9) \\ & + r_{\text{loop}}\,\mathbf{1}\{u_{t+1} \in \mathcal{L}_t,\ \text{serve}_t = 0\}, && (10) \end{aligned}$$

where $\theta_{\text{SoC}} \in (0, 1)$ is a low-SoC threshold; $\mathcal{L}_t$ is a short memory (e.g., the last m=3 nodes visited) used to detect short cycles; and $\text{serve}_t = 0$ indicates that no customer is served within $\mathcal{L}_t$. We use the distance term without additional scaling and keep the shaping magnitudes small relative to the (normalized) distance cost, with $r_{\text{ser}} > 0$, $r_{\text{re}} > 0$, and $r_{\text{loop}} < 0$. The charging bonus in (8) is granted only under low SoC and not for immediate repeated recharges at the same CS group (enforced by $a_{t-1} \notin V_s$). Equation (9) encourages service progress, and (10) suppresses degenerate short cycles (e.g., $i \to j \to i$ or CS–CS toggling) that do not contribute to service. Since the final objective remains distance minimization, the shaping terms can be annealed or removed in later training stages.

IV. PROPOSED METHOD

Given the EVRP-TW MDP formulation in Section III, we propose **TERRAN**, a transformer-based policy network trained via policy-gradient training to generate feasible EVRP-TW routes through autoregressive decoding. The framework explicitly incorporates time, energy, and capacity constraints into each routing decision (see Fig. 1).

Unlike prior DRL-based approaches that rely on retrospective constraint repair [28], [29], TERRAN embeds constraint feasibility directly into the decision-making process through three key innovations: (i) **Future-Feasibility Pruning (FFP)**, a proactive pruning mechanism that eliminates

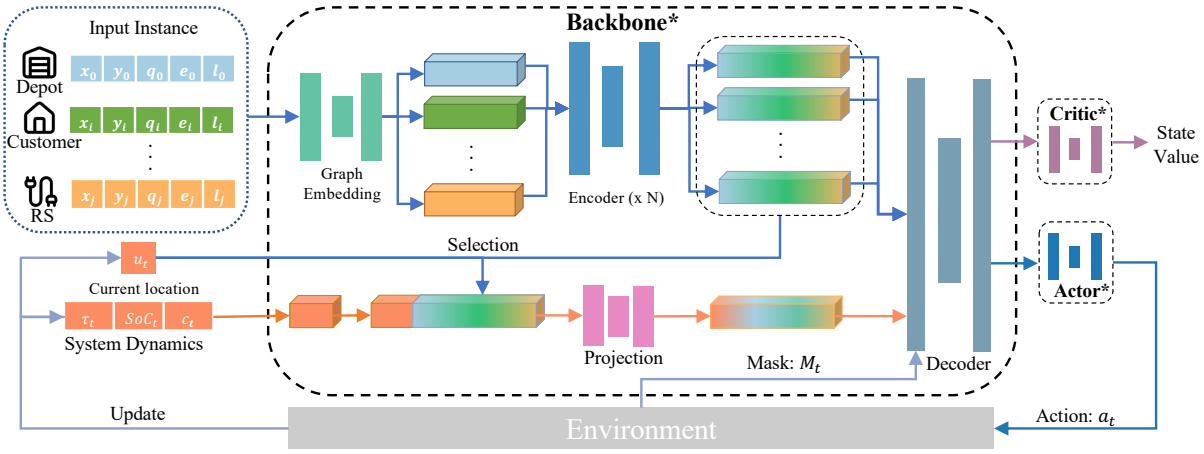


Fig. 1. **TERRAN overview.** Trainable modules are indicated by an asterisk (*): Transformer backbone*, Actor*, and Critic*. Constraint-handling components—feasibility masks M_t (time, capacity,), Future-Feasibility Pruning (FFP), and environment/state updates—are nontrainable and gate the action space before the policy step.

infeasible actions; (ii) **Staged Reward Scheduling**, a multi-phase reward strategy that progressively shifts from dense auxiliary signals to pure route optimization; and (iii) **End-to-End Transformer-based RL Agent Tailored for EVRP-TW**, which integrates battery, charging, and time-window constraints into the model architecture and decoding process, enabling feasibility-aware routing without post-processing.

A. Model Architecture

TERRAN adopts a transformer-based encoder-decoder architecture, augmented with dynamic state encoding, attention masking, and autoregressive decision-making. Fig. 1 summarizes the key modules and data flow.

a) Graph Embedding.: Each node is represented by a 5-D feature vector including spatial, temporal, and task-specific attributes (e.g., demand, charging eligibility), linearly projected into a shared embedding space.

b) Transformer Encoder ($\times N$).: We use N stacked encoder layers with multi-head self-attention to capture global node dependencies. The resulting contextualized embeddings are reused as attention keys and values.

c) Decoder with Dynamic Query and Feasibility Masking.: At each decoding step t, the decoder performs masked multi-head cross-attention. Encoder outputs $\{h_i\}_{i\in V}$ are linearly projected to keys and values, $K = W^K h_i$, $V = W^V h_i$. The query representation combines both local routing context and global graph information. Specifically, we concatenate the current-node embedding h_{u_t} with the dynamic state variables to form the input vector $w_t = [\tau^t; \mathrm{SoC}^t; c^t; h_{u_t}]$. This vector is processed by a feed-forward network to produce $z_t = \mathrm{FFN}(w_t)$, which is then combined with a graph-level aggregated feature $h_{\mathrm{graph}} = \frac{1}{|V|}\sum_{i\in V} h_i$, yielding the final query representation $\tilde{z}_t = z_t + h_{\mathrm{graph}}$. The attention query is given by $Q = W^Q \tilde{z}_t$. Cross-attention with (Q, K, V) produces an attention glimpse g_t, which is shared by both the actor (policy over feasible next nodes, decoded via greedy, sampling, or beam search) and the critic (state-value estimate). Before action selection, a binary feasibility mask M_t integrates time-window, capacity, and energy constraints together with Future-Feasibility Pruning (FFP) to suppress infeasible logits.

d) Actor and Critic.: The Actor network uses masked logits from g_t to sample the next action a_t, while the Critic network estimates the state value from g_t to guide PPO updates.

e) Environment.: After each action, the environment deterministically updates the routing state and recomputes the feasibility mask M_{t+1}, completing the autoregressive loop.

f) Discrete Decoding and Route Reconstruction.: At step t, masked logits over $V_t^{\mathrm{feas}} \subseteq V$ induce a categorical distribution over feasible candidates. Sampling from this distribution (e.g., greedy selection, stochastic sampling, or beam search) yields an *integer* index $u_{t+1} \in V_t^{\mathrm{feas}}$, which is appended to the growing permutation Π (Sec. II). The depot token 0 acts as a route delimiter: subsequences between two depots form vehicle routes $\{R_k\}$, and revisited customers are excluded by an instance-level visited mask. Decoding terminates when all customers have been visited and the last token is the depot, at which point Π is a complete discrete solution. Feasibility is ensured *by construction* because time-window, capacity, and energy masks—together with FFP—eliminate infeasible actions before selection; thus any decoded integer sequence corresponds to a feasible set of paths (e.g., $\{0 \to 5 \to 0\}$).

When multiple trajectories are generated (e.g., via restarts or beam search), we return the best instance solution under the distance objective.

B. Dynamic Staged Reward Scheduling

Optimizing only for total travel distance—while aligned with the final objective—leads to sparse and delayed rewards. In early training, the agent receives little feedback until it completes a valid route, often resulting in degenerate behaviors such as looping between charging stations or returning to the depot prematurely. To address this issue, we design a **dynamic multi-stage reward schedule** that transitions the agent from feasibility discovery to distance minimization. At each stage, we strategically retain terms from the full reward function (7), and gradually phase out others to avoid competing objectives. The training proceeds in three stages:

In the **Initialization Stage**, the reward function consists of three terms that encourage exploration and basic feasibility by providing dense feedback. The first term is the *distance penalty*, $r_{\text{dist}} = -d_{ij}$, which discourages longer travel distance. The second term is a *service reward*, $r_{\text{ser}} \geq 0$, which incentivizes customer visits. The third term is the *loop penalty*, $r_{\text{loop}} \leq 0$, which penalizes repetitive CS to CS moves.

In the **Shaping Stage**, two changes are made to the initialization stage's reward function to promote long-term planning by rewarding strategic charging decisions. First, we include a charging encouragement term, $r_{\text{re}} \geq 0$, which is applied when visiting a CS with low SoC. Second, we reduce the contribution of the service reward r_{ser} to help the agent shift its focus from simply meeting customer demand to improving the feasibility of the route.

In the **Optimization Stage**, all auxiliary reward terms are removed, and the reward is purely based on travel distance.

This staged design stabilizes training, promotes strategic behavior, and aligns the final policy with the EVRP-TW objective of minimizing total travel cost.

C. Feasibility Pruning and Masking Scheme

TERRAN enforces feasibility during decoding with a dual-layer masking mechanism. The first layer (*constraint-related*) encodes operational limits (time windows, capacity, energy), while the second layer (*instance-related*) captures routing logic (e.g., single service per customer and depot return).

Let $\mathcal{A}_t \subseteq V$ be the candidate set at step t. The feasible action set is

$$\mathcal{A}_t^{\text{feas}} \;=\; \{\, i \in V \;\mid\; M_t^{\text{con}}(i) = 1 \;\wedge\; M_t^{\text{inst}}(i) = 1 \,\}.$$

1) Constraint-Related Mask M_t^{con}. It is formed as a conjunction of three sub-masks:

$$M_t^{\text{con}}(i) \;=\; M_t^{\text{time}}(i)\, M_t^{\text{cap}}(i)\, M_t^{\text{energy}}(i). \tag{11}$$

(i) Time window.

$$M_t^{\text{time}}(i) = \begin{cases} 1, & \tau^t + \omega_{u_t,i} \leq l_i, \quad i \in V_c, \\ 1, & i \notin V_c, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

(ii) Capacity.

$$M_t^{\text{cap}}(i) = \begin{cases} 1, & c^t \geq q_i, \quad i \in V_c, \\ 1, & i \notin V_c, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

(iii) Energy via Future-Feasibility Pruning (FFP). Let $c(u,i)$ denote the energy to travel from u to i (under the constant-speed linear model $c(u,i) = \eta\, \omega_{u,i}$). Define the nearest-facility energy as:

$$E_{\min}(i) = \min_{j \in V_s \cup V_d} c(i,j), \qquad (E_{\min}(i) = 0 \text{ if } i \in V_s \cup V_d). \tag{14}$$

A candidate i is pruned at step t iff

$$c(u_t,i) + E_{\min}(i) > \text{SoC}^t \iff \text{SoC}^t - c(u_t,i) < E_{\min}(i). \tag{15}$$

Equivalently, the energy sub-mask is

$$M_t^{\text{energy}}(i) \;=\; \mathbf{1}\big[\, c(u_t,i) + E_{\min}(i) \leq \text{SoC}^t \,\big]. \tag{16}$$

2) Instance-Related Mask M_t^{inst}. Two rules are enforced.
(i) Visited-customer.

$$M_t^{\text{visit}}(i) = \begin{cases} 0, & i \in V_c \text{ and customer } i \text{ already served}, \\ 1, & \text{otherwise.} \end{cases} \tag{17}$$

(ii) Premature-return. If no unvisited customer satisfies both time and capacity, the next action must be the depot:

$$M_t^{\text{prem}}(i) = \begin{cases} 1, & i \in V_d, \\ 0, & i \notin V_d \text{ and } \sum_{j \in V_c} M_t^{\text{visit}}(j) M_t^{\text{time}}(j) M_t^{\text{cap}}(j) = 0, \\ 1, & \text{otherwise.} \end{cases} \tag{18}$$

The instance-level mask is

$$M_t^{\text{inst}}(i) = M_t^{\text{visit}}(i)\, M_t^{\text{prem}}(i), \tag{19}$$

and is updated after each action.

Final mask.

$$M_t(i) = M_t^{\text{con}}(i)\, M_t^{\text{inst}}(i), \tag{20}$$

which is used both to restrict the action-sampling space and to prune infeasible nodes during masked attention in the decoder.

D. Training

We train TERRAN using an on-policy actor-critic policy gradient framework, where a policy (actor) and its value function (critic) are learned in tandem [40]. Unlike prior DRL approaches for VRP that rely on REINFORCE [41]-style updates with sparse trajectory-level rewards, we support step-wise credit assignment through actor-critic optimization over state-action pairs (s_t, a_t). This enables the trained agent to exploit at each decision point intermediate feedback, such as service rewards, distance penalties, and constraint-driven incentives.

During training, the agent samples batches of solution trajectories via parallel environment rollouts. At each decoding

step t, it selects an action a_t from the masked feasible set $\{i \mid M_t(i) = 1\}$, and records the transition tuple (s_t, a_t, M_t, r_t). An episode ends when all customer nodes are served and the EV reaches the depot.

We optimize a standard on-policy policy-gradient objective by maximizing

$$\mathcal{L}_\pi = \mathbb{E}_t[A_t \log \pi(a_t \mid s_t) + \lambda_e H_{\text{ent}}(\pi_\theta)(\cdot \mid s_t)], \qquad (21)$$

where A_t is the advantage at time t and H^{ent} is an entropy regularization term that encourages exploration and mitigates premature convergence (λ_e is a small positive scalar). In its simplest form, the advantage can be defined as $A_t = R_t - V(\mathbf{s}_t)$, where $V(\mathbf{s}_t) = \mathbb{E}[R_t \,|\, \pi_\theta; s_t]$ is the value function estimated by the critic network. However, to reduce variance in policy updates, we employ the Generalized Advantage Estimation (GAE) method [42] to compute smoothed advantage estimates.

We use the Proximal Policy Optimization (PPO) algorithm [43] for policy training, where a surrogate loss $\mathcal{L}_\pi$ is used based on off-policy samples from an older policy $\pi_{\theta_{old}}$ to estimate the expected return of the current policy. This facilitates efficient learning of policies that generate feasible and high-quality EVRP-TW solutions under complex energy and time constraints.

The value function loss component corresponds to the critic's regression task: $\mathcal{L}_{\text{value}} = (V(s_t) - R_t)^2$, where R_t is the empirical return accumulated from step t.

V. Numerical Experiments

A. Experimental Setup

Before presenting results, we describe the training configuration, dataset construction, and evaluation protocol.

Hardware and Training Settings. All experiments are conducted on a workstation with an AMD Ryzen Threadripper 3970X (32-core CPU) and an NVIDIA RTX 2080 Ti GPU. The model is implemented in PyTorch and trained using the AdamW optimizer [44]. For PPO updates, we use a batch size of 64 and a total of 5 million environment steps. Additional training hyperparameters are provided in Appendix B.

Normalization and Units. To stabilize optimization and ensure comparability across instances, we normalize all inputs to a canonical $[0, 1]$ range on a per-instance basis. For each instance, node coordinates are translated and rescaled to fit within a unit square, customer demands are expressed relative to vehicle capacity, time windows and travel times are scaled by the instance horizon, and battery state-of-charge is scaled by vehicle battery capacity. The same procedure is consistently applied during training and inference; at deployment, each test instance (e.g., Solomon benchmarks) is normalized with its own per-instance rules before being processed by TERRAN. This reduces domain shift in scale, units, and spatial density, thereby enhancing cross-instance generalization.

Training Dataset. All training instances are synthetically generated following an extended Solomon-style EVRP-TW instance generator. Each instance contains a single depot, multiple charging stations, and a set of customers ($n \in \{5, 15, 100\}$) distributed under three spatial patterns: Random (R), Clustered (C), and Random–Clustered (RC). The number of charging stations is set to 3, 7, and 20 for the small-, medium-, and large-scale settings, respectively. Time windows are produced using standard Solomon benchmark rules to ensure consistency with VRPTW literature. To enhance structural diversity, two geographical layouts are used: an *inter-city* configuration (charging stations dispersed across the map) and an *intra-city* configuration (charging stations co-located with customer clusters). All instance-generation scripts have been released in our public repository to ensure full reproducibility.

Testing Dataset. Evaluation is conducted using: (i) the original Solomon benchmark dataset [45] for the 100-customer setting, which includes 56 standard instances across the R, C, and RC types; and (ii) 5-customer and 15-customer instances constructed by subsampling from a representative subset of Solomon instances. Specifically, we extract 12 sub-instances for each size and spatial type, ensuring structural coverage and consistency across evaluation scales.

Training Time. The 5-customer and 15-customer models are trained for 500,000 environment steps, taking approximately 2 days of wall-clock time. Due to the increased sequence length and action space, the 100-customer model is trained for 5 million steps over approximately 4 to 5 days.

Evaluation Metrics. We evaluate TERRAN and all baselines using three key metrics: (1) total travel distance (objective value), reflecting route quality; (2) inference time per instance (in seconds), measuring runtime efficiency; and (3) anytime objective values recorded at fixed time checkpoints (1, 5, 15, 30, 45, and 60 minutes), used to assess convergence speed and performance under time constraints.

B. Numerical Study Results

We present a comprehensive evaluation of TERRAN across benchmark scenarios of increasing complexity (5, 15, and 100 customers). The numerical studies are separated into four parts. The first part covers ablation studies to quantify the benefits of reward shaping and constraint pruning. The second part compares the proposed algorithm with baselines to assess the scalability and solution quality. The third part covers anytime behavior by analyzing algorithm convergence performance under runtime constraints. The final part visualizes the solution of a large-scale instance of EVRP-TW problem.

1) Ablation Studies: We conduct ablation studies to assess the contributions of two key components in TERRAN: **Dynamic staged reward scheduling**, which shifts the learning signal from feasibility heuristics toward cost minimization; and **Future-Feasibility Pruning (FFP)**, which removes actions that would strand the vehicle (i.e., insufficient energy to reach any charging station or the depot).

All reward-scheduling ablations are conducted on *15-customer subsets* to balance tractability and complexity. Specifically, we evaluate on *12 subsets derived from* Solomon instances spanning all three families: C-type (C103, C106, C202, C208), R-type (R102, R105, R202, R209), and RC-type (RC103, RC108, RC202, RC204). Unless otherwise stated, we report the mean across these 12 subsets. Hyperparameters are listed in Appendix B.

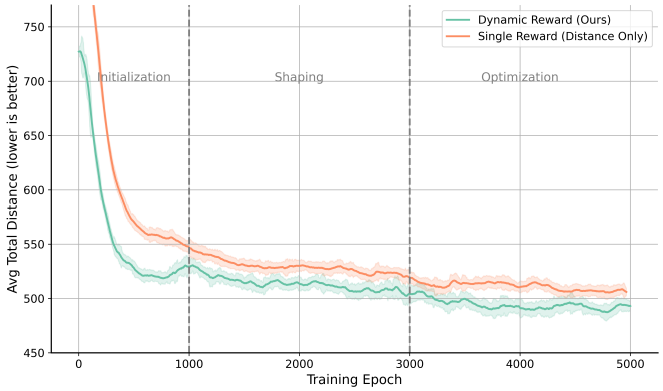


Fig. 2. Evaluation performance under dynamic vs. distance-only reward schedules. Solid lines indicate the average total distance across 12 Solomon test instances (C103, C106, C202, C208; R102, R105, R202, R209; RC103, RC108, RC202, RC204); shaded areas represent one standard deviation.

To isolate the effect of reward scheduling, we compare the full staged design against a baseline using only negative travel distance as the training signal, keeping FFP and all other components fixed to ensure a fair comparison. As shown in Fig. 2, dynamic reward scheduling accelerates convergence in early training and yields lower final objective values. It also improves generalization on evaluation instances, whereas the distance-only variant converges more slowly and often settles into suboptimal policies.

To assess the role of FFP, we disable the look-ahead energy-feasibility check during decoding. Without it, the agent may select nodes that are immediately reachable but ultimately become stranded due to insufficient battery to reach a charging station or depot. While soft-penalty methods attempt to discourage such actions via large negative rewards (e.g., $-1,000$), they do not guarantee constraint satisfaction at inference time, where stochastic sampling and policy noise can still yield invalid trajectories.

In contrast, FFP enforces feasibility through hard masking: it proactively excludes actions that would lead to energy-infeasible states, ensuring that the agent remains within the feasible solution space throughout decoding. This eliminates the need for artificial penalties or post hoc repairs, and maintains strict constraint compliance even under sampling noise.

2) Baseline Comparison: We benchmark TERRAN against two representative classical methods adapted from [13]: an exact MILP solved with **CPLEX** and a hybrid heuristic **VNS/TS**. To provide an end-to-end reference under identical objectives and constraints, we also implement a DRL baseline (**DRL–Sample, PPO**) with the same normalization, feasibility masks, Future-Feasibility Pruning (FFP), and distance-only reward as TERRAN. *The key distinction is that TERRAN employs* ***Staged Reward Scheduling*** *whereas DRL–Sample is trained solely with the distance objective.* All inference times are measured per instance on the same hardware. Deterministic decoders (CPLEX, VNS/TS, Ours–Greedy/Beam) have zero variance, whereas stochastic decoders (DRL–Sample, Ours–Sample) report both best and mean±std. Unless otherwise noted, we use a sample-based decoder with 100 parallel rollouts and select the lowest-cost trajectory, following [25], [36].

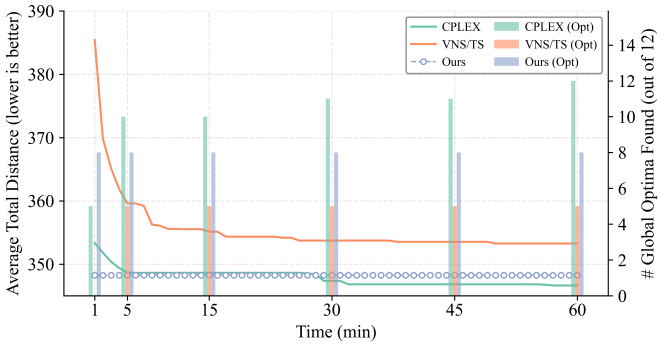


Fig. 3. Anytime performance on 15-customer EVRP-TW instances. TERRAN consistently converges faster and discovers a larger number of global optimal solutions within seconds.

a) Small instances ($n = 5$).: Ours–Sample attains the same best objective as CPLEX/VNS (187.1) while decoding in 0.01 s; the classical solvers require tens to hundreds of seconds. Relative to DRL–Sample, Ours–Sample substantially improves the mean objective (215 ± 8.9 vs. 312 ± 52.4).

b) Medium instances ($n = 15$).: CPLEX achieves the best solution (346.6) but requires 3,509 s. Ours–Sample reaches 351.7 (within ≈1.5% of CPLEX) and slightly outperforms VNS/TS (353.3), while decoding in 0.01 s. Compared with DRL–Sample, the mean objective improves from 607 ± 71.7 to 438 ± 34.5.

c) Large instances ($n = 100$).: Classical solvers are infeasible and not reported at this scale. Ours–Sample achieves a best objective of 1,249.6 and a mean of $1,457\pm177.6$ in 0.47 s, versus 2,571.1 (best) and $3,189\pm197.9$ (mean) for DRL–Sample (0.41 s), corresponding to improvements of about 51% (best) and 54% (mean). TERRAN attains near-optimal quality on small and medium Solomon instances—within ∼1–2% of CPLEX—while decoding orders of magnitude faster, and it markedly outperforms a matched DRL baseline at n=100 under the same protocol.

3) Anytime Behavior: To evaluate anytime performance, we focus on the 15-customer test set, which offers a nontrivial and balanced evaluation scenario. The 5-customer problem is too simple. Conversely, the 100-customer case is too challenging: CPLEX achieves feasibility on only 14.3% (8/56) of test instances, and VNS/TS fails to return any feasible solution within the time limit. Thus, the 15-customer case is an ideal middle ground for comparison.

We evaluate solution quality at six time checkpoints: 1, 5, 15, 30, 45, and 60 minutes. Fig. 3 presents the anytime performance of all algorithms, with lines indicating average objective values and bars showing the number of global optima found across 12 test instances. As shown in the figure, TERRAN identifies near-optimal solutions within seconds, significantly outperforming both CPLEX and VNS/TS throughout the time horizon. While CPLEX and VNS/TS gradually improve, neither achieves competitive performance within 28 minutes. TERRAN finds global optima in 8 out of 12 test cases—all within one minute—compared to 4 and 2 for CPLEX and VNS/TS, respectively. This highlights not only TERRAN's

TABLE II
PERFORMANCE ON SOLOMON BENCHMARKS. FOR EACH METHOD AND PROBLEM SIZE WE REPORT THE *best* OBJECTIVE (LOWER IS BETTER) AND RUNTIME. FOR STOCHASTIC DECODERS (DRL–SAMPLE, OURS–SAMPLE), WE ADDITIONALLY REPORT MEAN±STD ACROSS RUNS IN PARENTHESES. DETERMINISTIC METHODS (CPLEX, VNS/TS, OURS–GREEDY/BEAM) HAVE STD = 0.

Method	**5 Customers**			**15 Customers**			**100 Customers**		
	Best Obj	Mean±Std	Time (s)	Best Obj	Mean±Std	Time (s)	Best Obj	Mean±Std	Time (s)
CPLEX	187.1	–	148	346.6	–	3,509	–	–	–
VNS/TS	187.1	–	63	353.3	–	2,833	–	–	–
DRL–Sample	187.1	312±52.4	0.01	387.5	607±71.7	0.02	2,571.1	3,189±197.9	0.41
Ours–Greedy	189.7	–	0.01	493.3	–	0.02	1,579.2	–	0.22
Ours–Beam	189.3	–	0.01	367.0	–	0.02	1,509.5	–	0.24
Ours–Sample	**187.1**	**215±8.9**	0.01	**351.7**	**438±34.5**	0.02	**1,249.6**	**1,457±177.6**	0.47

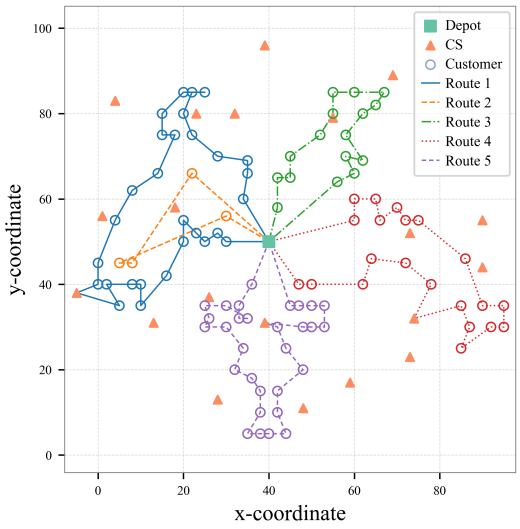


Fig. 4. Routing plan for a 100-customer EVRP-TW instance solved by TERRAN. Each color indicates a distinct vehicle route, with CSs strategically placed near customer clusters. The solution satisfies all constraints without post-processing, demonstrating the model's scalability and feasibility-aware planning.

ability to produce high-quality solutions rapidly, but also its robustness under tight time budgets.

In sum, TERRAN demonstrates superior anytime performance, combining short execution time, strong solution quality, and consistent feasibility—positioning it as a highly practical method for real-time EVRP-TW deployment.

4) Solution Visualization for a 100-Customer Instance: To qualitatively assess TERRAN's performance in large-scale settings, we examine a representative 100-customer instance. As shown in Fig. 4, the solution satisfies all time, energy, and capacity constraints without any post-processing. Each route (color-coded) forms a spatially coherent customer cluster, with strategically placed charging stops that align with battery limitations. Notably, TERRAN consistently respects time windows while minimizing detours, demonstrating effective long-horizon planning.

This case highlights TERRAN's ability to incorporate feasibility-driven decision-making into global route planning—validating the role of attention masking and FFP in producing robust, scalable solutions.

VI. CONCLUSION

We presented **TERRAN**, a Transformer-based reinforcement learning framework for the Electric Vehicle Routing Problem with Time Windows (EVRP-TW). TERRAN integrates Future-Feasibility Pruning (FFP) and staged reward scheduling into an end-to-end, feasibility-aware decoder, so that time, energy, and capacity constraints are enforced during action selection rather than via post-processing.

On Solomon benchmarks, TERRAN matches CPLEX on small instances and scales to large ones: it yields 100% feasible solutions for 100-customer cases in $\sim$0.47 s and achieves up to 1.7×10^5 speedups on 15-customer instances while remaining within 1.5% of optimal. These results indicate that the approach is both accurate and real-time capable.

This study focuses on the classic EVRP-TW benchmark: single depot, homogeneous fleet, and full charging [13]. We do not model heterogeneous vehicles or charging-station queuing. Nevertheless, TERRAN provides a reusable backbone: its constraint-aware Transformer and FFP can be extended to partial-charging models, heterogeneous or multi-depot fleets by appropriately augmenting the state and masking logic. Our open-source implementation offers a compact and transparent testbed for such extensions, as well as for exploring alternative learning signals (e.g., inverse or preference-based RL) and planning under uncertainty or partial observability.

APPENDIX

A. MILP Formulation for EVRP-TW

For benchmarking purposes, we adapt the MILP formulation from [13], which jointly models routing, charging, and time window constraints.

Objective:

$$\min \sum_{i \in V_0'} \sum_{\substack{j \in V_{N+1}' \\ j \neq i}} d_{ij} x_{ij} \tag{22}$$

Constraints:

$$\sum_{\substack{j \in V_{N+1}' \\ j \neq i}} x_{ij} = 1, \quad \forall i \in V \tag{23}$$

$$\sum_{\substack{j \in V_{N+1}' \\ j \neq i}} x_{ij} \leq 1, \quad \forall i \in V_s \tag{24}$$

$$\sum_{\substack{i \in V_{N+1}' \\ i \neq j}} x_{ij} = \sum_{\substack{i \in V_{N+1}' \\ i \neq j}} x_{ji}, \quad \forall j \in V' \tag{25}$$

$$\tau_i^t + (t_{ij} + s_i) x_{ij} \leq \tau_j^t + l_0(1 - x_{ij}), \quad \forall i \in V_d \tag{26}$$

$$\tau_i^t + t_{ij} x_{ij} + g^{inv}(Q - y_i) \leq \tau_j^t + (l_0 + g^{inv} Q)(1 - x_{ij}), \quad \forall i \in V_s \tag{27}$$

$$e_j \leq \tau_j^t \leq l_j, \quad \forall j \in V_0' \cup V_{N+1}' \tag{28}$$

$$0 \leq u_j \leq u_i - q_i x_{ij} + C(1 - x_{ij}), \quad \forall i \in V_0' \tag{29}$$

$$0 \leq u_0 \leq C \tag{30}$$

$$0 \leq y_j \leq y_i - h d_{ij} x_{ij} + Q(1 - x_{ij}), \quad \forall j \in V_{N+1}' \tag{31}$$

$$0 \leq y_j \leq Q - h d_{ij} x_{ij}, \quad \forall i \in V_s \tag{32}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j, i \neq j \tag{33}$$

Additional Notation Used in MILP:

Symbol	Description
V_0', V_{N+1}', V'	Extended node sets with dummy copies
τ_i	System time at node i
e_j, l_j	Time window bounds for customer j
s_i	Service time at customer i
u_i	Cargo load after visiting node i
u_0	Initial cargo capacity
y_i	Battery level after visiting node i
l_0	Large constant for time feasibility relaxation
g^{inv}	Inverse Charging Speed
h	Energy consumption rate per distance unit

The objective (22) minimizes the total travel distance across all routes. Constraints (23) and (24) govern node visitations: each customer must be visited exactly once, while charging station visits are flexibly handled via duplicated nodes. Constraint (25) ensures flow conservation, requiring that the number of arrivals and departures at each node are balanced. Temporal feasibility is maintained through constraints (26) and (27), which regulate travel and charging times between successive visits. Constraint (28) further enforces adherence to the time windows specified for each customer and depot node. Cargo capacity constraints are captured in constraints (29) and (30), ensuring the vehicle load remains non-negative and within maximum limits throughout the route. Similarly, constraints (31) and (32) maintain the energy feasibility of the EV, tracking battery levels along the journey. Finally, constraint (33) defines the binary nature of routing decisions, indicating whether an arc is traversed or not.

B. Hyperparameter Settings

The main hyperparameters used in training TERRAN are listed in Table III. These settings were selected based on preliminary experiments to ensure training stability and generalization across instance sizes. Values for staged reward coefficients and decoding strategies are included for completeness and can be adjusted based on problem scale.

TABLE III
TERRAN HYPERPARAMETER SETTINGS

Category	Hyperparameter Value
Optimization	
Optimizer	AdamW
Learning rate	3e-5
Discount factor (γ)	0.99
GAE parameter (λ)	0.95
PPO clip ratio (ϵ)	0.2
Value loss coefficient (λ_v)	0.5
Entropy loss coefficient (λ_e)	0.01
Training Setup	
Total environment steps	5 million
Training epochs	5,000
PPO batch size	64
Instance batch size per epoch	1,024
Number of parallel agents	100
Network Architecture	
Hidden size	128
Encoder layers	4
Attention heads	8
Dropout rate	0.1
Reward Parameters	
Charging warning threshold (θ_{SoC})	0.3
Customer service reward (r_{ser})	0.1
Charging reward (r_{re})	0.3
Loop penalty (r_{loop})	-100
Decoding Strategy	
Number of samples (sampling mode)	100
Beam width (beam search mode)	20

C. Baseline Solver Configuration

1) CPLEX Solver: CPLEX 22.1.1 was used with the following settings:

- Threads: 8 (to match hardware parallelism)
- MIP emphasis: `balance_optimality`
- Time limit: 1 hour for large instances
- Memory limit: 16GB

To address the modeling challenge that charging stations (CS) can be visited multiple times, whereas CPLEX natively enforces a single-visit constraint per node, we adopt a node duplication strategy [13]. Specifically, each CS is duplicated into multiple dummy nodes, each permitting a single visit. This approach enables CS nodes to be unified with customer nodes in a single routing graph without introducing additional structural complexity.

Each dummy node may be visited at most once, collectively allowing multiple visits to the same physical CS. However, there exists an inherent trade-off: increasing the number of dummy nodes expands the problem size and search space, leading to higher computational costs; conversely, insufficient duplication may restrict feasible routing options and exclude globally optimal solutions.

Prior studies commonly set the number of dummy nodes conservatively, e.g., up to three times the number of customers. However, our preliminary experiments suggest that such estimates can substantially enlarge the variable space without proportional gains in feasibility. Consequently, we empirically select a moderate number of dummy copies based on instance size:

- 5-customer instances: 3 dummy nodes per CS
- 15-customer instances: 5 dummy nodes per CS
- 100-customer instances: 7 dummy nodes per CS

We note that these settings are empirically tuned for the experimental benchmark used and are not guaranteed to be globally optimal. More adaptive strategies that adjust the number of dummy nodes based on instance-specific characteristics remain an open direction for future refinement.

2) VNS/TS Implementation.: We adopt the CPLEX solver and the classical VNS/TS heuristic [13] as competitive baselines for performance comparison. Since the original VNS/TS implementation was written in Java and not publicly released, we re-implemented it in Python based on the algorithmic descriptions provided in the paper. To ensure feasibility, we include the *AddNewVehicle* repair heuristic. While we follow the original design closely, some behavioral differences may exist due to unspecified implementation details.

For consistency with TERRAN, the reimplementation is done entirely in Python. We acknowledge that the choice of programming language can significantly affect runtime performance; therefore, our reported results are intended to reflect algorithmic behavior rather than absolute execution speed. All methods are evaluated under the same hardware and software conditions, with runtime limits of 1 minute (small instances) and 1 hour (large instances). Most hyperparameters follow the original paper, with reasonable assumptions made where necessary. The key settings are listed in Table IV.

TABLE IV
VNS SOLVER HYPERPARAMETER SETTINGS (15-CUSTOMER INSTANCES)

Parameter	Value	Description
ℓ_{tabu}	30	Max length of the tabu list
τ_{tabu}	100	Tabu Search iterations per restart
α, β, γ	10.0	Initial penalty weights
$\alpha_{\min}$, $\beta_{\min}$, $\gamma_{\min}$	0.5, 0.75, 1.0	Min penalties
$\alpha_{\max}$, $\beta_{\max}$, $\gamma_{\max}$	5,000	Max penalties
$k_{\max}$	15	VNS neighborhoods
η_{feas}, η_{dist}	700, 200	Iteration caps
δ_{SA}	0.08	SA cooling rate
λ_{div}	1.0	Diversity penalty
$N_{\text{route}}^{\text{pre}}$	1–10	Predefined route count
τ_{penalty}	2	Penalty update interval
δ	1.2	Penalty multiplier
T_0	adaptive	Init. SA temperature

D. Feasibility-Preserving Property of FFP

We formally show that the Future-Feasibility Pruning (FFP) mechanism preserves all feasible (and thus optimal) solutions by only eliminating actions that would lead to energy-infeasible states.

Suppose, for contradiction, that FFP prunes an action $a_t = i$ at step t, where node i lies on a feasible route. By the pruning rule, this action is excluded if:

$$\text{SoC}^t - \eta \cdot \omega_{u_t,i} < E_{\min}(i),$$

where SoC^t is the battery level before moving to node i, $\omega_{u_t,i}$ is the travel time from node u_t to i, η is the energy consumption rate, $E_{\min}(i)$ denotes the minimum energy required to reach any CS or depot from node i. Here, $E_{\min}(i) = \min_{j \in V_s \cup V_d} \eta \cdot \omega_{i,j}$ is precomputed for each node based on shortest-path travel time to its nearest charging point or the depot.

This condition implies that after moving to node i, the remaining energy would be insufficient to reach any charging point:

$$\text{SoC}^{t+1} = \text{SoC}^t - \eta \cdot \omega_{u_t,i} < E_{\min}(i),$$

which contradicts the feasibility requirement that the vehicle must always be able to reach a charging station or the depot from any visited node.

Hence, any action pruned by FFP cannot be part of a feasible route. FFP thus eliminates only actions that would inevitably violate the energy constraints, and preserves all feasible (and optimal) solution paths.

REFERENCES

[1] M. . Company, "The future of the last-mile ecosystem: Transitioning to sustainable delivery," 2022, accessed: 2025-04-28. [Online]. Available: https://www.mckinsey.com/industries/travel-logistics-and-infrastructure/our-insights/the-future-of-the-last-mile-ecosystem

[2] I. E. Agency, "Global ev outlook 2024: Catching up with climate ambitions," 2024, accessed: 2025-04-28. [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2024

[3] B. N. E. Finance, "Electric vehicle outlook 2024," 2024, accessed: 2025-04-28. [Online]. Available: https://about.bnef.com/electric-vehicle-outlook

[4] M. W. Ulmer, "Dynamic pricing and routing for same-day delivery," *Transp. Sci.*, vol. 54, no. 4, pp. 1016–1033, 2020.

[5] D. Sigler, Q. Wang, Z. Liu, V. Garikapati, A. Kotz, K. J. Kelly, M. Lunacek, and C. Phillips, "Route optimization for energy efficient airport shuttle operations–a case study from Dallas Fort Worth International Airport," *J. Air Transp. Manag.*, vol. 94, p. 102077, 2021.

[6] G. Wu, K. Boriboonsomsin, and M. J. Barth, "Eco-routing navigation system for electric vehicles," *arXiv preprint arXiv:2008.09674*, 2020.

[7] P. Toth and D. Vigo, *The vehicle routing problem*. SIAM, 2002.

[8] T. Paolo and D. Vigo, *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[9] J. Shi, Y. Gao, and N. Yu, "Routing electric vehicle fleet for ride-sharing," in *2018 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2)*, 2018, pp. 1–6.

[10] M. A. Bragin, Z. Ye, and N. Yu, "Toward efficient transportation electrification of heavy-duty trucks: Joint scheduling of truck routing and charging," *Transportation Research Part C: Emerging Technologies*, vol. 160, p. 104494, 2024.

[11] M. S. Hossain, C. Rodine, and E. E. Tsiropoulou, "A blockchain and PKI-based secure vehicle-to-vehicle energy-trading protocol," *Energies*, vol. 17, no. 17, p. 4245, 2024.

[12] D. Merchán, J. Arora, J. Pachon, K. Konduri, M. Winkenbach, S. Parks, and J. Noszek, "2021 Amazon last mile routing research challenge: Data set," *Transp. Sci.*, vol. 58, no. 1, pp. 8–11, 2024.

[13] M. Schneider, A. Stenger, and D. Goeke, "The electric vehicle-routing problem with time windows and recharging stations," *Transp. Sci.*, vol. 48, no. 4, pp. 500–520, 2014.

[14] G. Desaulniers, F. Errico, S. Irnich, and M. Schneider, "Exact algorithms for electric vehicle-routing problems with time windows," *Oper. Res.*, vol. 64, no. 6, pp. 1388–1405, 2016.

[15] Q. Zhang, F. Wang, and Z. Luo, "A bidirectional branch-and-price algorithm with pulse procedure for the electric vehicle routing problem with flexible deliveries," *Transp. Res. C Emerg. Technol.*, vol. 158, p. 104285, 2024.

[16] S. F. Roselli, P.-L. Götvall, M. Fabian, and K. Åkesson, "A compositional algorithm for the conflict-free electric vehicle routing problem," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 3, pp. 1405–1421, 2022.

[17] J. Duan, T. Wang, and W. Yang, "An adaptive large neighborhood search heuristic for the electric vehicle routing problems with time windows and recharging strategies," *J. Adv. Transp.*, vol. 2023, pp. 1–17, 2023.

[18] J.-F. Cordeau, G. Laporte, and A. Mercier, "A unified tabu search heuristic for vehicle routing problems with time windows," *J. Oper. Res. Soc.*, vol. 52, no. 8, pp. 928–936, 2001.

[19] H. Zhang, Q. Zhang, L. Ma, Z. Zhang, and Y. Liu, "A hybrid ant colony optimization algorithm for a multi-objective vehicle routing problem with flexible time windows," *Inf. Sci.*, vol. 490, pp. 166–190, 2019.

[20] M. Keskin and B. Çatay, "Partial recharge strategies for the electric vehicle routing problem with time windows," *Transp. Res. C Emerg. Technol.*, vol. 65, pp. 111–127, 2016.

[21] P. H. V. Penna, H. M. Afsar, C. Prins, and C. Prodhon, "A hybrid iterative local search algorithm for the electric fleet size and mix vehicle routing problem with time windows and recharging stations," *IFAC-Pap.*, vol. 49, no. 12, pp. 955–960, 2016.

[22] G. Hiermann, J. Puchinger, S. Ropke, and R. F. Hartl, "The electric fleet size and mix vehicle routing problem with time windows and recharging stations," *Eur. J. Oper. Res.*, vol. 252, no. 3, pp. 995–1018, 2016.

[23] J. Pierotti, "The EVRP-TW with heterogeneous recharging stations. an exact branch-and-price method," Ph.D. dissertation, Politecnico di Milano, 2016.

[24] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," *Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[25] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" *arXiv preprint arXiv:1803.08475*, 2018.

[26] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[27] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[28] B. Lin, B. Ghaddar, and J. Nathwani, "Deep reinforcement learning for the electric vehicle routing problem with time windows," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11 528–11 538, 2021.

[29] J. Chen, H. Huang, Z. Zhang, and J. Wang, "Deep reinforcement learning with two-stage training strategy for practical electric vehicle routing problem with time windows," in *International conference on parallel problem solving from nature*. Springer, 2022, pp. 356–370.

[30] Z. Zong, X. Tong, M. Zheng, and Y. Li, "Reinforcement learning for solving multiple vehicle routing problem with time window," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 2, pp. 1–19, 2024.

[31] A. Mozhdehi, M. Mohammadizadeh, Y. Wang, S. Sun, and X. Wang, "Efectiw-roter: Deep reinforcement learning approach for solving heterogeneous fleet and demand vehicle routing problem with time-window constraints," in *Proceedings of the 32nd ACM International Conference on Advances in Geographic Information Systems*, 2024, pp. 17–28.

[32] A. Mozhdehi, M. Mohammadizadeh, and X. Wang, "Edge-direct: A deep reinforcement learning-based method for solving heterogeneous electric vehicle routing problem with time window constraints," *arXiv preprint arXiv:2407.01615*, 2024.

[33] M. Tang, W. Zhuang, B. Li, H. Liu, Z. Song, and G. Yin, "Energy-optimal routing for electric vehicles using deep reinforcement learning with transformer," *Applied Energy*, vol. 350, p. 121711, 2023.

[34] M. A. Elbouzidi, M. Bendouda, and M. Oussalah, "A novel routing solution for ev fleets: A real-world case study leveraging double DQNs and graph-structured data to solve the EVRPTW problem," *IEEE Access*, vol. 11, pp. 151 492–151 509, 2023.

[35] Ö. Aslan Yıldız, İ. Sarıçiçek, and A. Yazıcı, "A reinforcement learning-based solution for the capacitated electric vehicle routing problem from the last-mile delivery perspective," *Appl. Sci.*, vol. 15, no. 3, p. 1068, 2025.

[36] M. Wang, Y. Wei, X. Huang, and S. Gao, "An end-to-end deep reinforcement learning framework for electric vehicle routing problem," *IEEE Internet Things J.*, vol. 11, no. 20, pp. 33 671–33 682, 2024.

[37] J. Chen, X. Zhang, Y. Zhou, G. Chen, Q. Zhai, and Y. Cao, "Optimal joint scheduling of distribution network and electric vehicles tracks considering actual carbon emission responsibility," *IEEE Transactions on Transportation Electrification*, 2025.

[38] H. Lee, K. Kim, N. Kim, and S. W. Cha, "Energy efficient speed planning of electric vehicles for car-following scenario using model-based reinforcement learning," *Applied Energy*, vol. 313, p. 118460, 2022.

[39] R. A. Dollar, A. Vahidi, B. Pattel, and H. Borhan, "A linear programming formulation for eco-driving over road slopes," *Automatica*, vol. 161, p. 111483, 2024.

[40] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[41] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

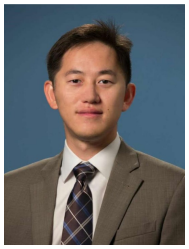
[42] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *ICLR*, 2016.

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[44] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[45] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Oper. Res.*, vol. 35, no. 2, pp. 254–265, 1987.

Maojie Tang (M'25) received the B.S. and M.S. degrees in Transportation Engineering from Tongji University, Shanghai, China, in 2016 and 2019, respectively, and the M.S. degree in Computer Science from Rice University, Houston, TX, USA, in 2022. He is currently pursuing the Ph.D. degree in Computer Science at the University of California, Riverside, CA, USA. His research interests lie at the intersection of deep reinforcement learning and combinatorial optimization, with applications in electric vehicle routing and sustainable logistics.



Nanpeng Yu (M'11–SM'16) received the B.S. degree in Electrical Engineering from Tsinghua University, Beijing, China, in 2006, and the M.S. and Ph.D. degrees in Electrical Engineering from Iowa State University, Ames, IA, USA, in 2007 and 2010, respectively. He is currently a Full Professor with the Department of Electrical and Computer Engineering and the Director of the Energy, Economics, and Environment Research Center at the University of California, Riverside, CA, USA. His research interests include physics-informed machine learning, optimization theory, smart grids, electricity markets, transportation electrification, and sustainable data-center systems.

Ioannis Karamouzas (Member, IEEE) received the Ph.D. degree in Computer Science from Utrecht University, The Netherlands, in 2014. He is an Associate Professor with the Department of Computer Science and Engineering, University of California, Riverside, USA. His research interests lie at the intersection of robotics, artificial intelligence, and computer graphics, focusing on motion-planning algorithms for autonomous agents in both physical and virtual worlds. His work has been integrated into commercial robotics navigation and pedestrian-simulation platforms, and he serves on the editorial boards of *IEEE Robotics and Automation Letters* and *IEEE Transactions on Visualization and Computer Graphics*.

Zuzhao Ye (Member, IEEE) received the B.E. degree in Thermal Energy and Power Engineering from the University of Science and Technology of China, Hefei, China, in 2015, and the Ph.D. degree in Electrical Engineering from the University of California, Riverside, CA, USA, in 2024. He is currently a Postdoctoral Scholar with the Department of Electrical and Computer Engineering, University of California, Riverside. His research interests include big-data analytics, machine learning, and optimization, with applications to the planning and operation of electric-vehicle charging infrastructure.