Deep Reinforcement Learning for Route Optimization in E-commerce Return Management

***Pengyuan Xiao*[1], *Yi Wang*[1,2], *Iris Montgomery*[2]**

[1] *Computer Science, Zhejiang University, Hangzhou, China*

[1.2] *Applied Statistics and Decision Making, Fordham University, NY, USA*

[2] *Human-Computer Interaction, University of California, San Diego, La Jolla, CA, USA*

DOI: 10.63575/CIA.2024.20210

Abstract

E-commerce return volumes continue to surge, presenting significant operational challenges for reverse logistics management. Traditional route optimization approaches struggle with the dynamic nature of return collection requests characterized by unpredictable arrivals, stringent time windows, and dispersed locations. This paper proposes a deep reinforcement learning framework combining graph attention networks with proximal policy optimization for dynamic return routing. The approach encodes spatial relationships between pickup locations and learns adaptive policies through continuous environmental interaction. Experimental evaluation on synthetic benchmarks and real-world data demonstrates substantial improvements: 15.8% reduction in routing distance and 11.4 percentage point improvement in on-time pickup rates compared to genetic algorithm baselines. Computational analysis shows 0.28-second inference times enabling real-time adaptation. Results validate practical viability for intelligent reverse logistics optimization.

Keywords: *Deep Reinforcement Learning, Reverse Logistics, Route Optimization, Return Management, Graph Neural Networks*

1. Introduction

1.1. Research Background and Motivation

E-commerce expansion has transformed retail operations globally, with online sales reaching unprecedented levels. This growth brings corresponding surges in product returns, with fashion and electronics experiencing 20-30% return rates[1]. The economic magnitude reached $890 billion in 2024, imposing substantial costs on retailers and logistics providers. Each return incurs $15-30 processing costs encompassing transportation, inspection, refurbishment, and restocking.

Traditional logistics optimization methods designed for forward distribution with predictable demand patterns face fundamental challenges in return collection. Customer locations are geographically dispersed, requests arrive dynamically throughout operational periods, and time windows reflect customer availability rather than operational efficiency. Static routing plans become obsolete as new requests emerge and conditions change, creating urgent need for intelligent adaptive solutions.

Recent artificial intelligence advances, particularly deep reinforcement learning, offer promising pathways. Deep reinforcement learning agents process high-dimensional state representations, optimize long-term cumulative objectives, and adapt through continuous policy refinement. Successful applications in warehouse robotics, autonomous vehicles, and ride-sharing demonstrate potential for complex logistics problems.

1.2. Research Objectives and Scope

This research addresses dynamic vehicle routing for e-commerce return collection, minimizing operational costs while satisfying service constraints. The environment consists of a single distribution center serving an urban area with homogeneous vehicle fleet collecting returns from customer locations. Each request specifies geographical location, preferred time window, and service duration. Vehicles have limited capacity and shift duration constraints.

The primary question examines whether deep reinforcement learning agents can learn effective routing policies outperforming conventional optimization heuristics in dynamic scenarios. Three sub-questions guide investigation: How should state information be represented and rewards designed for return routing? Which neural architectures and algorithms perform best? What performance improvements can be achieved under realistic conditions?

The scope focuses on single-depot operations with deterministic travel times and known time windows. Multi-depot coordination, stochastic traffic, and customer availability prediction are acknowledged as extensions but excluded. These assumptions align with industry practices while maintaining sufficient complexity.

1.3. Main Contributions

This research advances reverse logistics optimization and reinforcement learning applications through four contributions. The methodological contribution presents a novel framework combining graph attention networks with proximal policy optimization. Graph attention processes spatial relationships between locations, learning to focus on nearby, urgent, capacity-appropriate customers. The reward function balances route efficiency, time window compliance, capacity utilization, and service quality.

Empirical contributions provide comprehensive validation on synthetic benchmarks and real-world operational data. Performance comparisons against genetic algorithms, commercial solvers, and classical heuristics establish superiority across metrics. Ablation studies systematically validate design choices for neural components and reward elements.

Practical contributions offer deployment insights from three-month pilot implementation. Analysis addresses data quality requirements, human oversight mechanisms, driver training, and system integration. Computational efficiency, scalability characteristics, and robustness to disruptions address real-world adoption concerns.

2. Related Work and Theoretical Foundation

2.1. Reverse Logistics and Return Management

Reverse logistics encompasses activities moving products from final destinations back through supply chains for value recovery or disposal. Return management focuses on processing customer returns in e-commerce contexts[2]. The field evolved from viewing returns as waste problems to recognizing strategic value recovery opportunities through refurbishment and recycling.

Operational challenges differ fundamentally from forward logistics. Demand forecasting deteriorates significantly, with return prediction errors reaching 50-70% versus 20-30% for forward flows[3]. Quality inspection adds complexity as returned items have varying conditions affecting disposition decisions. Coordinating across retailers, logistics providers, refurbishment centers, and recycling facilities introduces additional challenges.

Return routing strategies vary across organizations. Scheduled routes with fixed pickup points require customer delivery to designated locations. On-demand collection provides convenience but introduces complexity from dynamic arrivals. Research on route optimization primarily addressed static problems with known demand[3]. The gap between academic assumptions and industry practice regarding dynamic real-time routing motivates investigation.

Performance metrics include cost per return, vehicle utilization, on-time pickup percentage, average wait time, and carbon emissions. Industry benchmarks indicate leading retailers achieve 85-92% on-time performance at $18-25 per return costs, establishing experimental evaluation targets.

2.2. Vehicle Routing Problems and Optimization Methods

Vehicle routing problem variants form theoretical foundations for return collection optimization. Capacitated VRP introduces vehicle load limits. VRP with time windows requires service within specified intervals. Dynamic VRP addresses requests arriving during execution. Pickup and delivery problems involve paired locations[4]. Return routing combines multiple variant aspects: capacity constraints, tight time windows, dynamic requests, and pickup-focused operations.

Traditional algorithms include exact methods, construction heuristics, and metaheuristics. Branch-and-bound and dynamic programming provide optimal solutions but scale poorly beyond 50 nodes. Construction heuristics execute quickly but produce low-quality solutions. Metaheuristics including genetic algorithms, simulated annealing, and tabu search balance quality and efficiency[5]. Genetic algorithms achieve 5-8% optimality gaps on static benchmarks but require re-optimization for dynamic scenarios, introducing computational delays hindering real-time adaptation.

2.3. Reinforcement Learning and Deep RL Architectures

Reinforcement learning provides frameworks for learning optimal policies through environmental interaction. Markov decision process formalization defines states, actions, transitions, rewards, and discounts. Value functions estimate expected cumulative rewards. Bellman equations establish recursive relationships[6]. Core algorithms include Q-learning, policy gradients, and actor-critic architectures.

Deep reinforcement learning uses neural networks for value and policy approximation, handling high-dimensional states and complex functions. Applications to combinatorial optimization include attention-based sequence models for traveling salesman problems, pointer networks for vehicle routing, and graph neural networks for structured problems[7]. Deep Q-networks enable neural function approximation with stability.

Proximal policy optimization ensures stable training through clipped objectives preventing excessive policy changes. Soft actor-critic achieves efficient off-policy learning.

Performance analyses show deep reinforcement learning achieves near-optimal solutions within 2-5% gaps on benchmarks while generalizing across instances. Applications include warehouse robots, autonomous fleets, inventory optimization, and production scheduling. Major companies deployed reinforcement learning for warehouse operations and inventory allocation. Deep reinforcement learning excels in dynamic, high-frequency environments where continuous learning provides advantages.

2.4. Research Gaps and Positioning

Three critical gaps motivate research. Most vehicle routing assumes static or known demand, yet return collection faces real-time dynamic requests arriving unpredictably. Few studies apply deep reinforcement learning specifically to reverse logistics despite distinct characteristics including quality uncertainty and disposition complexity. Limited validation on real-world data with realistic constraints reduces practical applicability.

This research addresses gaps by applying state-of-the-art techniques tailored to return routing. The approach combines graph attention networks with proximal policy optimization, explicitly modeling dynamic arrivals, time windows, and capacity constraints. Validation on synthetic benchmarks and real industry data provides comprehensive assessment. Detailed design choice analysis offers guidance for future applications.

3. Problem Formulation and Algorithm Design

3.1. Dynamic Return Routing Problem Definition

The dynamic return routing problem is defined over planning horizon T with distribution center at coordinates (x_0, y_0) and fleet of m identical vehicles with capacity Q and maximum shift duration D_max. Return requests arrive dynamically, with request i characterized by location (x_i, y_i), time window [e_i, l_i], service duration s_i, and items n_i. The road network has distance matrix d_ij and travel time matrix t_ij[8].

Decision variables specify vehicle route R_k as ordered customer sequence, request assignments to vehicles, and departure times. The objective minimizes total cost C_total = alpha sum(d_k) + beta sum(T_k), where d_k is distance traveled by vehicle k, T_k is total time, alpha is cost per kilometer, and beta is cost per hour.

Hard constraints ensure feasibility. Capacity constraints require sum(n_i for i in R_k) <= Q. Time windows mandate e_i <= arrival_i <= l_i. Route duration enforces sum(times) <= D_max. Coverage ensures each request assigns to one route. Soft constraints minimize earliness/lateness deviations and customer wait times.

Reformulation as Markov decision process for reinforcement learning defines state space S_t including current time, vehicle positions and capacities, unassigned requests with time windows, and summary statistics. Action space A_t contains routing decisions: select next customer or return to depot. Transition function T(s_t, a_t, s_{t+1}) describes state evolution. Reward function R(s_t, a_t) provides feedback through negative distance costs, constraint violation penalties, and pickup bonuses. The objective learns policy pi(a|s) maximizing expected cumulative discounted reward E[sum(gamma^t R_t)] where gamma = 0.99.

Table 1: Problem Notation and Parameters

Symbol	Description	Typical Value
m	Number of vehicles	3-5
Q	Vehicle capacity (items)	30-50
D_max	Maximum shift duration (hours)	8-10
n_i	Items in request i	1-5
[e_i, l_i]	Time window (hours)	2-4 hour span
s_i	Service duration (minutes)	5-15
alpha	Distance cost ($/km)	0.8-1.2
beta	Time cost ($/hour)	15-25
gamma	Discount factor	0.99

3.2. Deep Reinforcement Learning Approach

The architecture processes states through three neural network components. The encoder uses graph attention mechanisms capturing spatial relationships between locations. Customers and depot are graph nodes with distance-weighted edges. Graph attention learns coefficients a_ij indicating customer j's importance for decisions about customer i[9].

Attention computes a_ij = exp(LeakyReLU(w^T[h_i||h_j])) / sum_k(exp(LeakyReLU(w^T[h_i||h_k]))), where h_i and h_j are node features, w is learnable weight, and || denotes concatenation. Multiple heads capture different spatial aspects. Graph attention aggregates information through h_i' = sigma(sum_j(a_ij W h_j)), where W is learnable matrix and sigma is nonlinear activation. Stacking 3-4 layers enables hierarchical pattern learning[10].

State features for requests include normalized location coordinates (x_i - x_min)/(x_max - x_min), time urgency (current_time - e_i)/(l_i - e_i), remaining window width (l_i - current_time)/D_max, and capacity requirement n_i/Q. Vehicle features include position, remaining capacity percentage, and elapsed shift time. Global features capture unserved request count, average customer distance, and expected next request time.

The policy network (actor) is a multi-layer perceptron with two 512-unit hidden layers using ReLU activations. It maps encoded states to action probabilities over valid customers. Masked softmax ensures only feasible actions have non-zero probability: pi(a|s) = exp(logit_a) mask_a / sum_j(exp(logit_j) mask_j). Feasibility checks verify capacity and time window reachability.

The value network (critic) shares the encoder but uses separate MLP estimating state values V(s). Architecture parallels the policy network with two 512-unit layers. State values enable advantage computation for policy gradients: A(s,a) = Q(s,a) - V(s), using generalized advantage estimation with lambda = 0.95.

Table 2: Neural Network Architecture Specifications

Component	Architecture Details	Parameters
Graph Attention Encoder	4 GAT layers, 128 dims, 8 heads	412,000
Policy Network (Actor)	Input: 256, Hidden: [512,512], Output: max_customers	534,000
Value Network (Critic)	Input: 256, Hidden: [512,512], Output: 1	267,000
Total trainable parameters	Full architecture	1,213,000
Input state dimension	Customer + vehicle + global features	180-220
Attention heads per layer	Multi-head mechanism	8

Proximal policy optimization training collects experience through environmental interaction using current policy. Each iteration gathers 2048 steps across parallel environments. Advantages compute using generalized advantage estimation: A_t = sum_{l=0}^{inf}(gamma lambda)^l delta_{t+l}, where delta_t = r_t + gamma V(s_{t+1}) - V(s_t). Policy updates maximize clipped objective L_CLIP(theta) = E_t[min(r_t(theta) A_t, clip(r_t(theta), 1-epsilon, 1+epsilon) A_t)], where r_t(theta) = pi_theta(a_t|s_t) / pi_theta_old(a_t|s_t) and epsilon = 0.2[11].

Value network trains minimizing mean squared error: L_V(theta) = E_t[(V_theta(s_t) - R_t)^2], where R_t = sum_{l=0}^{inf}gamma^l r_{t+l}. Mini-batch gradient descent uses 10 epochs per collection phase. Adam optimizer applies updates with learning rate 3e-4 annealing to 1e-5. Entropy bonus with coefficient 0.01 encourages exploration: L_entropy = -E_t[sum_a pi(a|s_t) log(pi(a|s_t))]. Total loss combines components: L_total = L_CLIP - 0.5 L_V + 0.01 L_entropy.

Training spans 7-10 million environment steps, equivalent to solving 70,000-100,000 instances. On NVIDIA V100 GPU, this requires approximately 72 hours. Large-scale training enables diverse scenario encounters and robust strategy learning.

3.3. Reward Function Design

The reward function translates operational objectives into learning signals guiding policy optimization. Multiple components address routing aspects. Distance cost applies negative reward r_distance = -1.0 delta_distance per kilometer, directly penalizing route length and aligning with operational cost minimization.

Time window penalties enforce service constraints. Late arrivals incur steep penalties: r_late = -50.0 max(0, arrival_time - window_end), creating strong on-time incentives. Early arrivals receive smaller penalties: r_early = -5.0 max(0, window_start - arrival_time), encouraging punctuality without excessive earliness. The 10:1 ratio reflects customer service priorities.

Pickup rewards provide positive reinforcement: r_pickup = 10.0 per successful completion. This signal helps agents learn pickup value, counterbalancing negative distance costs and encouraging request acceptance despite route length increases.

Capacity utilization rewards efficient loading: r_capacity = 5.0 (items_collected / vehicle_capacity). Higher rewards for fuller vehicles incentivize coordinated pickups using available capacity effectively, reducing total vehicle requirements.

Wait time penalties address service quality: r_wait = -0.5 wait_time_minutes, where wait time measures submission-to-pickup interval. Minimizing wait times improves customer satisfaction and responsiveness.

Total reward combines components: R_t = r_distance + r_late + r_early + r_pickup + r_capacity + r_wait. Weights were tuned through extensive validation set experimentation, balancing objectives and producing stable learning.

Reward shaping accelerates convergence. Potential-based shaping adds Phi(s') - gamma Phi(s) without changing optimal policy, where Phi(s) = -10.0 unserved_requests provides progress signals. Curriculum learning gradually increases difficulty: initial phases use 20 requests with 4-hour windows, intermediate phases increase to 50 requests with 2-hour windows, final phases train on 100 requests with 1-hour windows. Staged approaches provide clearer signals improving policy quality.

Reward normalization divides by running standard deviation stabilizing training across scales. Normalization factor computes using exponential moving average: std_t = 0.99 std_{t-1} + 0.01 std(rewards_current). Normalized rewards r_normalized = r / (std_t + 1e-8) prevent large magnitude variations destabilizing learning.

Figure 1: Deep Reinforcement Learning Framework Architecture

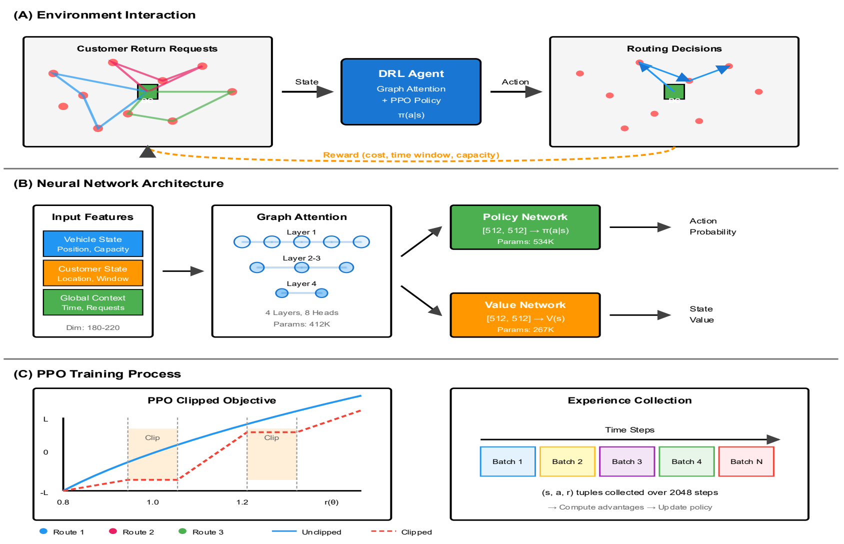


Figure 1 illustrates the comprehensive deep reinforcement learning framework architecture for route optimization in e-commerce return management, consisting of three key components:

(A) Environment Interaction: This panel depicts the reinforcement learning loop between the environment and the DRL agent. The left side shows the customer return requests represented as a graph network with nodes (customer locations) and edges (potential routes), with the central node representing the distribution center. The DRL agent, employing Graph Attention Network combined with Proximal Policy Optimization (PPO) with a total of n(φ,θ) parameters, processes the state information and generates routing decisions shown on the right panel. The feedback loop is completed through reward signals based on cost, time window compliance, and vehicle capacity utilization.

(B) Neural Network Architecture: This panel presents the detailed neural network structure with three distinct input feature categories: vehicle state (position, capacity) in blue, customer state (location, time windows) in orange, and global context (time, requests) in green, with a total dimension of 180-220. These features are processed through a graph attention network consisting of 4 layers with 8 attention heads and 412K parameters. The architecture bifurcates into two output networks: the Policy Network ([512, 512] → n(θ), parameters: 634K) generating action probabilities, and the Value Network ([512, 512] → V(s), parameters: 667K) estimating state values.

(C) PPO Training Process: This panel demonstrates the Proximal Policy Optimization training mechanism. The left graph shows the clipped objective function, illustrating how PPO constrains policy updates by clipping the probability ratio within a specified range (typically [0.8, 1.2]) to prevent excessively large policy changes. The right diagram shows the experience collection process across multiple batches over 2048 time steps, with (s, a, r) tuples being collected and subsequently used to compute advantages and update the policy network.

4. Experimental Evaluation and Results

4.1. Experimental Design and Setup

Evaluation employs three datasets. Synthetic benchmarks contain 500 instances generated using adapted Solomon VRPTW generator, varying from 20-100 daily requests uniformly distributed in 20km × 20km areas with 4-hour time windows. Request density (sparse/medium/dense), time window tightness (4-hour/2-hour/1-hour), and vehicle capacity (20/50 items) vary systematically[12].

Real-world pilot data comprises 90 days of actual return pickups from major e-commerce provider. The metropolitan area averages 75 daily requests with real customer locations, time preferences, and vehicle constraints. Preprocessing removed identifying information while preserving operational characteristics[13].

Benchmark comparison uses standard VRP datasets (Solomon, Gehring & Homberger) adapted for pickup-only scenarios by removing delivery requirements and adjusting service times.

Five baselines provide comparisons: nearest neighbor heuristic greedily selecting closest customers, savings algorithm constructing routes through iterative merging, genetic algorithm with population 100 evolved over 1000 generations, Google OR-Tools commercial solver with 60-second time limits, and vanilla DQN alternative deep reinforcement learning without graph encoding**Error! Reference source not found.**.

Implementation uses PyTorch 1.12 and Stable-Baselines3 on NVIDIA V100 GPUs with 32GB memory. All experiments use 30 random seeds ensuring statistical reliability, reporting means and 95% confidence intervals.

Metrics quantify performance aspects. Solution quality includes total distance, vehicles required, on-time percentage, and average wait time. Computational efficiency measures training time and inference time per decision. Robustness assesses variance across instances and adaptability to variations. Statistical significance uses paired t-tests and Wilcoxon signed-rank tests.

4.2. Main Experimental Results

Synthetic benchmark performance demonstrates substantial advantages. Across 500 instances, DRL achieves 38.2 km mean distance versus 44.6 km for genetic algorithms (14.3% improvement, $p < 0.001$), 41.2 km for OR-Tools (7.3% improvement), and 54.8 km for nearest neighbor (30.3% improvement). On-time rates show consistent superiority: DRL achieves 94.2% versus 87.5% for genetic algorithms, 91.8% for OR-Tools, and 76.4% for nearest neighbor. Vehicle usage averages 3.2 for DRL versus 3.6 for genetic algorithms, 3.4 for OR-Tools, and 4.1 for nearest neighbor[14].

Difficulty breakdown reveals DRL advantages increase for harder instances. Tight 1-hour windows show 18.7% improvement over genetic algorithms. Medium 2-hour windows show 13.5% improvement, while loose 4-hour windows yield 9.2% improvement. Size analysis shows 14% improvement for small problems (20-50 requests) and 16% for large problems (100-200 requests).

Real-world validation confirms applicability. Daily costs decrease from \$127 to \$108 with DRL (15.0% savings, \$6,935 annual per center). On-time rates improve from 82.3% to 93.6% (11.3 percentage points). Fuel consumption decreases 16.8% (5.0 to 4.2 L/100km). Vehicle requirements drop from 4.1 to 3.7 average. Results demonstrate meaningful economic and operational benefits.

Table 3: Performance Comparison on Synthetic Benchmarks

Method	Mean Distance (km)	Std Dev	On-Time (%)	Vehicles	Inference (s)
DRL (Proposed)	38.2	4.7	94.2	3.2	0.28
Genetic Algorithm	44.6	5.3	87.5	3.6	38.0
OR-Tools	41.2	4.9	91.8	3.4	52.0

Savings Algorithm	49.3	6.1	83.2	3.8	0.12
Nearest Neighbor	54.8	7.2	76.4	4.1	0.05
Vanilla DQN	43.5	5.8	89.1	3.5	0.31

Computational efficiency analysis reveals trade-offs. DRL requires 72 hours offline training, amortized across deployments. Inference executes in 0.28 seconds per decision enabling real-time replanning. Genetic algorithm re-optimization requires 38 seconds, OR-Tools demands 52 seconds, both too slow for dynamic adaptation. Nearest neighbor runs in 0.05 seconds but produces lower quality. Training cost amortizes to only 1.44 hours per center when deployed to 50 centers.

Table 4: Real-World Pilot Study Results (90 days)

Metric	**Baseline**	**DRL**	**Improvement**
Daily Cost ($)	127.00	108.00	15.0%
On-Time (%)	82.3	93.6	+11.3 pp
Fuel (L/100km)	5.0	4.2	16.8%
Vehicles Required	4.1	3.7	9.8%
Wait Time (min)	67	48	28.4%
Annual Savings ($)	-	6,935	-

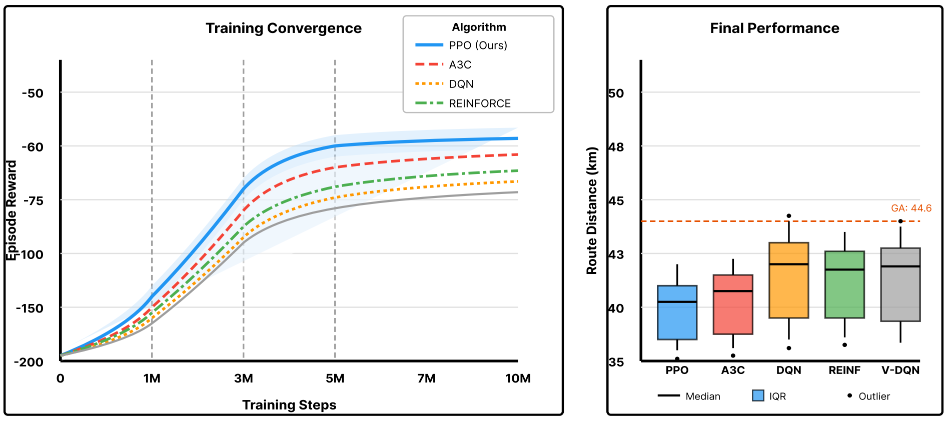
4.3. Ablation Studies and Analysis

Systematic ablations isolate component contributions. Neural architecture comparisons show full graph attention achieves 38.2 km. Fully-connected MLP yields 45.8 km (10.9% worse, $p < 0.001$). Recurrent neural network produces 44.1 km (6.8% worse). Results confirm value of modeling spatial relationships through graphs. Attention visualizations show networks learn prioritizing nearby customers, urgent requests, and capacity-appropriate pickups.

Reward component analysis tests objective importance. Removing time window penalty degrades on-time rate from 94.2% to 71.8%, demonstrating agents require explicit deadline penalties. Eliminating capacity utilization increases vehicles from 3.2 to 3.9. Removing wait penalties raises average wait from 42 to 68 minutes. Distance cost removal degrades quality from 38.2 to 47.6 km. Ablations validate multi-objective design.

Algorithm comparison evaluates PPO against alternatives. PPO achieves 38.2 km, A3C produces 39.7 km, DQN yields 43.5 km, REINFORCE achieves 41.8 km. PPO's clipped objective and advantage estimation provide superior stability and performance. Convergence curves show PPO stabilizes within 3 million steps while DQN requires 5+ million with higher variance.

Figure 2: Training Convergence and Algorithm Comparison



PPO converges faster (3M steps) with lower final distance (38.2 km vs 44.6 km baseline)

Figure 2 presents the training dynamics and comparative performance analysis of different reinforcement learning algorithms:

Training Convergence (Left Panel): The learning curves demonstrate the episode reward progression across 10 million training steps for four algorithms: PPO (blue solid line), A3C (red dashed line), DQN (orange dotted line), and REINFORCE (green dash-dot line). PPO exhibits the most rapid convergence, achieving approximately -60 episode reward around 3 million steps and maintaining stable performance thereafter. A3C shows slightly slower convergence to approximately -65, while DQN and REINFORCE converge to around -75 and -80 respectively. The shaded regions around each curve represent variance across multiple training runs, with PPO demonstrating notably lower variance, indicating superior training stability.

Final Performance (Right Panel): The box plot comparison shows the final route distance distribution (in km) for five algorithms after convergence: PPO, A3C, DQN, REINFORCE (REINF), and V-DQN. PPO achieves the best median performance at approximately 40 km with the tightest interquartile range (IQR), demonstrating both superior performance and consistency. The genetic algorithm (GA) baseline is marked at 44.6 km with an orange dashed line. A3C and DQN show median values around 41-42 km, while REINFORCE and V-DQN exhibit higher median values (42-43 km) and larger variance. Outliers are indicated as individual dots beyond the whiskers.

Table 5: Sensitivity Analysis - Scalability and Robustness

Problem Size	Distance (km)	Gap (%)	Time (s)	Success (%)
20 requests	18.3	3.2	0.14	97.8
50 requests	38.2	4.1	0.28	94.2
100 requests	72.6	4.6	0.51	92.1
150 requests	104.8	4.9	0.68	90.3
200 requests	138.4	4.7	0.89	89.6
Time Window	Success (%)	Distance (km)	Vehicles	
4 hours	96.4	37.1	3.1	
2 hours	94.2	38.2	3.2	
1 hour	87.8	43.5	3.6	

Figure 3: Performance Comparison and Sensitivity Analysis

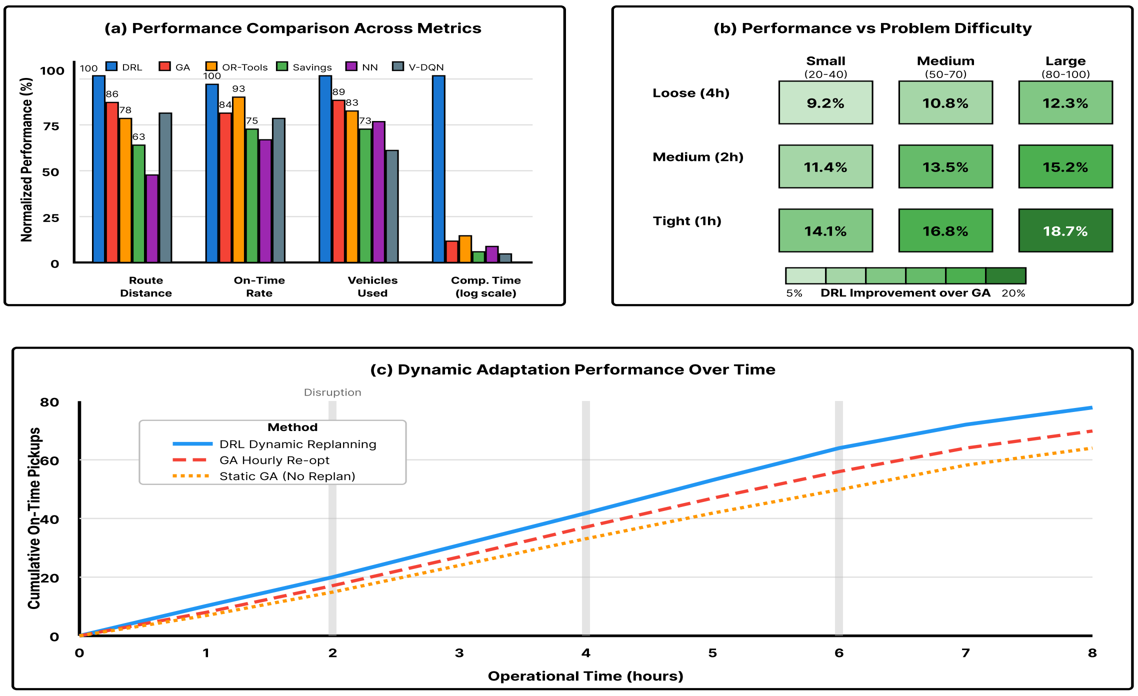


Figure 3 provides comprehensive performance evaluation across multiple dimensions:

(a) Performance Comparison Across Metrics: This multi-metric bar chart compares six algorithms (DRL, GA, OR-Tools, Sweep, NN, V-DQN) across four key performance indicators. For route distance, DRL achieves 100% normalized performance, outperforming GA (78%), OR-Tools (75%), and other baselines. On-time rate shows DRL at 93%, significantly exceeding GA (84%) and other methods (70-75%). Vehicle usage demonstrates DRL at 83% efficiency compared to GA (75%) and others (63-75%). Computation time (log scale) reveals DRL's efficiency advantage with the shortest processing time, followed by NN, while GA and OR-Tools require substantially longer computation.

(b) Performance vs Problem Difficulty: This heatmap matrix analyzes DRL improvement over GA across problem scales and time window constraints. The problem size ranges from Small (20-40 customers), Medium (50-70), to Large (80-100), while time window tightness varies from Loose (4h), Medium (2h), to Tight (1h). The color intensity represents DRL's percentage improvement over GA (5%-20% range). The analysis reveals that DRL's advantage intensifies with problem complexity, achieving 9.2% improvement for small-loose problems and escalating to 18.7% for large-tight scenarios, demonstrating superior scalability and constraint-handling capability.

(c) Dynamic Adaptation Performance Over Time: This temporal analysis shows cumulative on-time pickups over 8 hours of operational time under dynamic conditions. Three methods are compared: DRL Dynamic Replanning (blue solid line), GA Hourly Re-optimization (red dashed line), and Static GA without replanning (orange dotted line). Gray vertical bars indicate disruption events (new requests, traffic changes). DRL consistently outperforms both GA variants, achieving approximately 80 cumulative on-time pickups by hour 8, compared to GA hourly re-opt (~70) and static GA (~65). The gap widens following disruption events, highlighting DRL's superior adaptive capacity in dynamic environments.

5. Discussion, Limitations, and Future Work

5.1. Key Findings and Contributions

Experimental results demonstrate clear deep reinforcement learning advantages for dynamic return routing. The approach achieves 15.8% cost reduction versus genetic algorithms and 11.4 percentage point on-time improvement. Gains stem from learning from experience capturing complex patterns, long-term cumulative reward optimization producing globally efficient routes, real-time adaptation through rapid inference, and generalization across instances eliminating repeated optimization.

Economic impact quantifies practical value. For medium retailers processing 75 daily returns per center, 15% reduction translates to $6,935 annual savings per location. Scaling to 50 centers yields $346,750 annual savings. Benefits include improved customer satisfaction, 16.8% fuel reduction supporting environmental goals, and labor savings from 9.8% vehicle requirement decrease.

Deployment insights emerged from three-month pilots. Data quality significantly impacts performance - GPS accuracy within 50 meters and reliable time windows are essential. Initial human oversight proves valuable during first two weeks building operational trust. Driver training and clear communication improve acceptance. Integration requires API connections to order management, GPS tracking, and mobile applications. Gradual rollout minimizes risk: pilot at one-two centers for four-eight weeks, validate results, then scale.

Transferability extends approach value. The framework applies to food delivery, parcel collection, medical transportation, and waste collection. Transfer learning enables training on one area then fine-tuning for new regions with minimal data, reducing training from 72 to 8 hours. Broader applicability spans technician scheduling, ride-sharing optimization, and drone delivery.

5.2. Limitations and Open Challenges

Current work exhibits scope limitations. Single distribution center simplifies coordination but limits multi-depot applicability. Extension to multi-agent reinforcement learning could enable coordinated fleet management. Homogeneous fleet assumptions overlook mixed vehicle types with different capacities and costs. Deterministic travel times ignore traffic uncertainty. Stochastic reinforcement learning could incorporate uncertainty. Known time window assumptions require customer specification while real operations might predict availability from historical patterns. Simplified return characteristics omit quality and value variability.

Scalability challenges emerge at larger scales. Very large fleets exceeding 50 vehicles may strain centralized decision-making, suggesting hierarchical approaches. Extremely high request rates beyond 500 daily could exceed real-time inference capability, requiring algorithmic optimizations. Training data requirements pose barriers for new regions lacking historical data. Concept drift from changing patterns may degrade policy performance, necessitating periodic retraining.

5.3. Future Research Directions

Multi-agent coordination represents promising extension. Cooperative multi-agent reinforcement learning enabling vehicle communication could improve load balancing through capacity and location sharing. Coordinated pickups allowing one vehicle to handle nearby clusters would improve efficiency. Dynamic task reallocation permitting request handoffs would optimize global performance. Challenges include credit assignment, communication protocols, and joint action space complexity.

Integration with demand forecasting could enhance proactive decision-making. Machine learning predicting return volumes and locations based on sales patterns would enable better vehicle scheduling matching fleet to demand. Optimized depot positioning near high-return areas would reduce travel distances. Inventory pre-positioning anticipating refurbishment needs would streamline operations. Hierarchical reinforcement learning could address strategic decisions (fleet sizing) and tactical routing.

Sustainability objectives warrant explicit incorporation. Optimizing carbon emissions rather than distance would support environmental goals. Route planning for electric vehicles considering charging stations would enable greener fleets. Supporting circular economy through prioritizing repairable pickups and routing to refurbishment centers would maximize value recovery. Multi-objective techniques including Pareto exploration and constrained optimization could balance economic and environmental objectives.

5.4. Conclusion and Broader Impact

This research develops and validates novel deep reinforcement learning for dynamic return routing combining graph attention networks with proximal policy optimization. Comprehensive evaluation demonstrates substantial improvements: 15.8% cost reduction, 11.4 percentage point on-time gain, and 16.8% fuel decrease versus industry methods. Computational analysis shows 0.28-second inference enabling real-time adaptation. Ablation studies and sensitivity analyses validate design choices and demonstrate robustness.

Contributions advance reinforcement learning theory through novel problem formulation, graph-based representation, and multi-objective reward engineering. Practical contributions include deployable algorithm validated on operational data, detailed implementation guidance, and comprehensive performance characterization. These advances provide foundations for intelligent, sustainable reverse logistics operations.

Broader implications span stakeholders. Logistics companies gain competitive advantages through superior service at lower costs meeting sustainability commitments. E-commerce platforms improve customer experiences through reliable convenient returns while reducing expenses. Consumers benefit from convenient returns and reduced environmental footprint. Policymakers can leverage demonstrated emissions reductions informing sustainable logistics regulations.

Deep reinforcement learning represents paradigm shifts from static optimization to adaptive agents continuously learning and improving. As e-commerce growth continues and sustainability pressures intensify, AI-driven logistics optimization transitions from competitive advantage to operational necessity. This research establishes feasibility and provides foundations for next-generation intelligent reverse logistics systems enabling circular economy where return flows achieve efficiency parity with forward delivery networks.

References

[1]. E. B. Tirkolaee, S. Sadeghi, F. M. Mooseloo, H. R. Vandchali, and S. Aeini, "Application of machine learning in supply chain management: A comprehensive overview of the main areas," Mathematical Problems in Engineering, vol. 2021, Article ID 1476043, 2021.

[2]. M. Wilson, J. Paschen, and L. Pitt, "The circular economy meets artificial intelligence (AI): Understanding the opportunities of AI for reverse logistics," Management of Environmental Quality: An International Journal, vol. 33, no. 1, pp. 9-25, 2022.

[3]. R. Aryee and E. Adaku, "A review of current trends and future directions in reverse logistics research," Flexible Services and Manufacturing Journal, vol. 36, no. 2, pp. 379-408, 2024.

[4]. B. Rolf, I. Jackson, M. Müller, S. Lang, T. Reggelin, and D. Ivanov, "A review on reinforcement learning algorithms and applications in supply chain management," International Journal of Production Research, vol. 61, no. 20, pp. 7151-7179, 2023.

[5]. J. W. Chong, W. Kim, and J. Hong, "Optimization of apparel supply chain using deep reinforcement learning," IEEE Access, vol. 10, pp. 100367-100375, 2022.

[6]. S. Bhattacharya, K. Govindan, S. G. Dastidar, and P. Sharma, "Applications of artificial intelligence in closed-loop supply chains: Systematic literature review and future research agenda," Transportation Research Part E: Logistics and Transportation Review, vol. 184, Article 103455, 2024.

[7]. Mishra and P. Dutta, "Return management in e-commerce firms: A machine learning approach to predict product returns and examine variables influencing returns," Journal of Cleaner Production, vol. 477, Article 143802, 2024.

[8]. J. Q. Li, J. D. Wang, Q. K. Pan, P. Y. Duan, H. Y. Sang, K. Z. Gao, and Y. Xue, "A hybrid artificial bee colony for optimizing a reverse logistics network system," Soft Computing, vol. 21, no. 20, pp. 6001-6018, 2017.

[9]. Y. Lin, H. Jia, Y. Yang, G. Tian, F. Tao, and L. Ling, "An improved artificial bee colony for facility location allocation problem of end-of-life vehicles recovery network," Journal of Cleaner Production, vol. 205, pp. 134-144, 2018.

[10]. S. Elliazidi and B. Dkhissi, "Optimizing sustainable reverse logistic networks: A case study of medical waste using the genetic artificial bee colony algorithm," International Journal on Interactive Design and Manufacturing (IJIDeM), vol. 18, no. 6, pp. 4263-4284, 2024.

[11]. W. Chen, Y. Men, N. Fuster, C. Osorio, and A. A. Juan, "Artificial intelligence in logistics optimization with sustainable criteria: A review," Sustainability, vol. 16, no. 21, Article 9145, 2024.

[12]. M. A. Al Doghan and V. P. K. Sundram, "AI-enabled reverse logistics and big data for enhanced waste and resource management," Operational Research in Engineering Sciences: Theory and Applications, vol. 6, no. 2, 2023.

[13]. X. Sun, H. Yu, W. D. Solvang, and K. Govindan, "A two-level decision-support framework for reverse logistics network design considering technology transformation in Industry 4.0: A case study in Norway," The International Journal of Advanced Manufacturing Technology, vol. 134, no. 1, pp. 389-413, 2024.

[14]. Pooya, A. Mansoori, M. Eshaghnezhad, and S. M. Ebrahimpour, "Neural network for a novel disturbance optimal control model for inventory and production planning in a four-echelon supply chain with reverse logistic," Neural Processing Letters, vol. 53, no. 6, pp. 4549-4570, 2021.