SDU University

UDC: 004.83; 004.8.032.26 Manuscript Copyright

**MERALIYEV MERARYSLAN MERRUSLANOVICH**

Neural Combinatorial Optimization for Vehicle Routing Problem with Drones

6D070400 – Computing Systems and Software

Thesis for the Degree of

doctor of philosophy (PhD)

Scientific Supervisor

doctor PhD,

assoc. prof.

A. Bogyrbayeva

Foreign scientific supervisor

doctor PhD,

assoc. prof.

F. Hajiyev

Republic of Kazakhstan

Kaskelen, 2024

**CONTENT**

|  |  |
| --- | --- |
| [**NORMATIVE REFERENCES**](#_heading=h.30j0zll)………………………………..….…………. | 3 |
| [**ABBREVIATIONS**](#_heading=h.3znysh7)……………………………………………….……….….. | 4 |
| [**INTRODUCTION**](#_heading=h.tyjcwt)…………………………………………………………….. | 5 |
| **1** [**BACKGROUND AND LITERATURE REVIEW**](#_heading=h.1t3h5sf)…………………...…... | 12 |
| 1.1 [Overview of the Vehicle Routing Problem (VRP)](#_heading=h.2s8eyo1)…………..…………..… | 12 |
| 1.2 [Variants of the Vehicle Routing Problem (VRP)](#_heading=h.3rdcrjn)………………..….….….. | 13 |
| 1.3 [Methodological Advances in Solving VRPs](#_heading=h.35nkun2)……………………...……….. | 19 |
| 1.3.1 [Traditional Approaches](#_heading=h.35nkun2)…………………………………………..………. | 19 |
| 1.3.2 [Learning methods](#_heading=h.44sinio)……..…………………………………………………. | 21 |
| 1.4 [Computational studies](#_heading=h.z337ya)…………………………………………………….. | 35 |
| **2** [**METHODOLOGY**](#_heading=h.4i7ojhp)…………………………………………………………. | 38 |
| 2.1 [MDP Formulation of VRPD](#_heading=h.1ci93xb)………………………………………….……. | 38 |
| 2.2 [Solution Method](#_heading=h.2bn6wsx)………………………………………………………….… | 40 |
| 2.2.1 [The Deep Learning Model](#_heading=h.3as4poj)………………………………………..……… | 40 |
| 2.2.2 [The Training Algorithm](#_heading=h.2p2csry)………………………………………………….. | 47 |
| [**3** **EXPERIMENTAL RESULTS**…………………………………….………..](#_heading=h.23ckvvd) | 49 |
| 3.1 [Training and Evaluation Configurations](#_heading=h.32hioqz)…………………………………… | 51 |
| 3.2 [Results on RL models comparison](#_heading=h.41mghml)…………………………….……..……. | 54 |
| 3.3 [Results on VRPD with Unlimited Flying Range](#_heading=h.vx1227)……………………….….. | 56 |
| [**4 DISCUSSIONS**](#_heading=h.1v1yuxt)………………………………………….…………….…….. | 58 |
| **CONCLUSION**…………………………………………………………..…… | 59 |
| **REFERENCES**…………………………………………………………….….. | 60 |
| [**APPENDIX A**](#_heading=h.vgdtq7) – The performance comparison……………………………….. | 69 |

# 

# NORMATIVE REFERENCES

This thesis uses references to the following standards:

“Instructions for the preparation of a dissertation and author’s abstract” Ministry of education and science of the Republic of Kazakhstan, 377-3 Zh.

GOST 7.32-2001. Report on research work. Structure and design rules.

GOST 7.1-2003. Bibliographic record. Bibliographic description. General requirements and compilation rules.

GOST 7.32-2017. System of standards of information, librarianship, and publishing. Research report. Structure and design rule.

**ABBREVIATIONS**

|  |  |
| --- | --- |
| ADP | – Approximate Dynamic Programming |
| AR | – Autoregressive |
| AI | – Artificial Intelligence |
| ANN | – Artificial Neural Network |
| B&B | – Branch and Bound |
| CAGR | – Compound Annual Growth Rate |
| CSP | – Covering Salesman Problem |
| CVRP | – Capacitated Vehicle Routing Problem |
| DRL | – Deep Reinforcement Learning |
| DU | – dynamic and uncertain VRP |
| DVRP | – Dynamic Vehicle Routing Problem |
| DQN | – Deep Q-Network |
| EVRPTW | – Electric Vehicles Routing Problem with Time Windows |
| GAT | – Graph Attention Network |
| GCN | – Graph Convolutional Network |
| GNN | – Graph Neural Network |
| GRU | – Gated Recurrent Unit |
| GVRP | – Green Vehicle Routing Problem |
| HM | – Hybrid Model |
| IP | – Integer Programming |
| LSTM | – Long Short-Term Memory |
| MDP | – Markov Decision Process |
| MDVRP | – Multi-Depot Vehicle Routing Problem |
| MVRP | – Multi-Vehicle Routing Problem |
| NAR | – Non-Autoregressive |
| PPO | – Proximal Policy Optimization |
| RBFN | – Radial Basis Function Network |
| RL | – Reinforcement Learning |
| RNN | – Recurrent Neural Network |
| S2V | – Structure2Vec |
| SDVRP | – Split Delivery Vehicle Routing Problem |
| SL | – Supervised Learning |
| SVRP | – Stochastic Vehicle Routing Problem |
| TD | – Temporal Difference |
| TSP | – Traveling Salesman Problem |
| TSPTW | – Traveling Salesman Problem with Time Windows |
| VRP | – Vehicle Routing Problem |
| VRPD | – Vehicle Routing Problem with Drones |
| VRPTW | – Vehicle Routing Problem with Time Windows |

# INTRODUCTION

The retail industry underwent a significant transformation in the early 2000s due to the introduction of digital technology, particularly the Internet, which revolutionized competition among retailers [[1](#_heading=h.3tbugp1)]. This shift led to the rapid growth of online shopping and the adoption of interactive technologies by brick-and-mortar retailers to stay competitive [[2](#_heading=h.28h4qwu)]. The use of self-service technologies aimed at enhancing productivity and service quality while reducing costs became increasingly prevalent in retail [[3](#_heading=h.nmf14n)]. Additionally, the impact of e-commerce and information technology on the retail industry, including changes in competition dynamics, service quality, and customer interactions, has been substantial [[4](#_heading=h.37m2jsg)]. This change wasn’t just about moving from physical stores to online platforms; it marked a substantial shift in how consumers behaved and how the market worked. Online shopping made shopping more democratic, offering consumers unprecedented convenience and choices right at their fingertips. As online platforms advanced, they not only offered a vast selection of products but also created interactive shopping experiences that drew in a global audience.

This era of digital commerce broke down traditional geographic and logistical barriers, allowing consumers to shop anytime and from anywhere. The speed and convenience offered by digital platforms set new expectations for fast fulfillment and delivery, highlighting logistical challenges, especially in the last-mile delivery. The last-mile delivery, the final step in the e-commerce supply chain, became crucial in fulfilling the promise of online shopping by ensuring products reached consumers quickly and reliably.

**Problem Statement** As e-commerce platforms expanded their offerings, the diversity and volume of products have increased the complexity of logistics, particularly in last-mile delivery. Traditional delivery networks, designed for less frequent and more predictable shipping demands, struggle with the dual challenges of maintaining speed and efficiency while managing rising costs and minimizing environmental impacts. This has led to two key problems:

1. Optimizing fleet routes to ensure efficient deliveries.
2. Efficiently managing communication between multiple vehicles and drones to streamline operations.

**Aim of the Research Work** To enhance the efficiency and reliability of last-mile deliveries by optimizing delivery routes through the integration of drones and RL technologies.

**Research Objectives** The detailed objectives of this research are:

1. To design an MDP framework and develop an RL model based on it for efficient deliveries.
2. To develop algorithms that ensure effective communication and coordination between multiple vehicles and drones, enhancing logistical efficiency.

**Significance of the Study** This research holds substantial significance for the global e-commerce industry, with particular implications for Kazakhstan, a nation experiencing rapid digital transformation and growth in online retail sectors. Below are detailed reasons why this study is crucial for Kazakhstan:

1. Growing E-commerce Market: Kazakhstan’s e-commerce sector has shown impressive growth rates, spurred by increasing internet penetration and a growing young consumer base eager to embrace digital shopping. Optimizing last-mile delivery in Kazakhstan could significantly enhance the e-commerce experience, helping local businesses increase their competitiveness both domestically and in international markets [[5](#_heading=h.1mrcu09)].
2. Urbanization Trends: As urbanization continues to increase in cities like Almaty and Nur-Sultan, the challenges of urban logistics become more pronounced. Effective last-mile delivery solutions, such as the ones proposed in this research, are critical for managing the density and diversity of urban demands, improving the efficiency of deliveries in these congested areas [[6](#_heading=h.46r0co2)].
3. Environmental Concerns: Kazakhstan has committed to substantial environmental goals to reduce carbon emissions. Integrating drones into the delivery networks can help achieve these objectives by reducing the reliance on traditional motor vehicles, thus decreasing urban air pollution and contributing to a greener economy [[7](#_heading=h.2lwamvv)].
4. Technological Leadership: By adopting advanced technologies such as drones and Reinforcement Learning for logistical applications, Kazakhstan can position itself as a leader in technological innovation within the Central Asian region. This move can enhance its technological ecosystem, attracting investments and fostering innovation in related sectors [[8](#_heading=h.111kx3o)].
5. Policy and Regulatory Development: This research could also inform policymakers in Kazakhstan about the potential benefits and challenges of drone delivery. It could serve as a basis for developing regulations that support the safe and efficient use of drones in commercial applications, paving the way for broader adoption of new technologies.
6. Scalability Across Regions: While the focus might initially be on urban centers, the findings could have implications for rural areas in Kazakhstan, where vast distances and limited infrastructure often challenge delivery logistics. Drones could revolutionize how goods are delivered in remote areas, making providing faster service across the country feasible.

**Addressing National Priorities** This study aligns with Kazakhstan’s strategic priorities to boost digitalization across economic sectors and improve the efficiency of its logistics and transportation systems. By enhancing last-mile delivery, this research directly contributes to these national goals, supporting Kazakhstan’s vision for a digital-first economy and improving the quality of life for its citizens through better service delivery.

**Scientific Novelty** This thesis introduces a pioneering approach by combining drone technology with RL to address the vehicle routing problem, marking a significant innovation in logistics research.

**Personal Contribution of the PhD. Candidate** The PhD. candidate was instrumental in all research phases, excelling in data collection, sophisticated data analysis, and model development. His involvement extended to rigorous testing and optimization of the developed models, contributing significantly to the overall research design and execution. The candidate also took the lead in writing and revising the manuscripts, ensuring that each publication accurately reflected the findings and adhered to high academic standards. These efforts culminated in producing an influential article based on the research.

**Publications**

1. ”Machine learning to solve vehicle routing problems: A survey,” IEEE Transactions on Intelligent Transportation Systems, 2024.
2. ”Optimization of Data Segments and Number of Cores for Defining Popularity of Kazakh Words Using Apache Spark” - Engineering Journal of Satbayev University 143.3 (2021): 39-42.
3. ”Face extraction and recognition from public images using HIPI” - 2018 14th International Conference on Electronics, Computer and Computation (ICECCO).

**Statistical Insights and Industry Dynamics** The rise of e-commerce globally has fundamentally transformed consumer shopping behaviors, and Kazakhstan is experiencing a similar transformation. As of 2020, an impressive 65% of companies have embraced technology to streamline the critical logistics phase of last-mile delivery, reflecting a broader industry trend toward meeting evolving consumer expectations for speed and reliability [[9](#_heading=h.3l18frh)]. In Kazakhstan, the e-commerce market has grown significantly, fueled by increased digital connectivity and a shift in consumer preferences towards online platforms. This growth has placed new demands on logistical operations, especially in densely populated urban areas where most deliveries occur.

The global market for last-mile delivery services, valued at $31.25 billion in 2018, is projected to reach $55.2 billion by 2025, growing at a Compound Annual Growth Rate (CAGR) of 16.7%. This rapid expansion underscores the escalating consumer demand for delivery solutions that are not only efficient but seamlessly integrated into the fabric of modern digital life. More than a quarter of consumers worldwide are willing to pay premiums for expedited delivery services, highlighting a significant shift towards prioritizing speed and convenience [[9](#_heading=h.3l18frh)]. This trend is acutely felt in Kazakhstan, where urban consumers increasingly expect same-day or next-day delivery as standard service.

However, the demand for speed introduces challenges as consumer patience with delivery delays is noticeably diminishing. Research indicates that 39% of consumers would reconsider their purchasing decisions if faced with slow last-mile delivery, placing immense pressure on e-commerce businesses to enhance their logistical operations to maintain customer loyalty and market share [[9](#_heading=h.3l18frh)].

Integrating drones into the delivery networks of the future may offer a game-changing solution to the perennial challenges of speed, cost, and environmental impact. This is only dampened by the complexities presented within the regulatory landscape, most noticeably for the use of commercial drones, which is still very fresh within the country. The environmental ambitions in the country that technology is brought about by drones could serve, for the introduction of drone technology might make a small dent in its overreliance on traditional delivery vehicles equipped with combustion engines that, in turn, would reduce urban air pollution and greenhouse gases.

One of the most important aspects within this dynamically changing landscape is that the use of big data effectively can make companies operate efficiently, forecast customer behaviors, customize service provisions, and react to real-time challenges of last-mile delivery. This paper discusses the details of Reinforcement Learning, a very sophisticated tool using very complex algorithms to find the best way to optimize the delivery route and complex decision processes based on inputs of ever-changing data.

This study, therefore, shows the capital importance of technological innovation for the e-commerce sector. Drones and Reinforcement Learning, for example, would mean a strategic competitive logistic advantage answer to the challenge of logistics that comes with a borderless world for Kazakhstan. The findings of the current study offer invaluable insights that may guide businesses, policymakers, and stakeholders to successfully mark a path through the rapid change in e-commerce logistics landscapes.

**Scope and Limitations**

**Scope of the Study** This thesis proposes to explore some measures that will be applied to enhance the efficiency and sustainability of last-mile delivery within the e-commerce sector, including drones and Reinforcement Learning (RL) technologies. It will seek to do the following:

1. **Geographical Focus:** The subject of the proposed study covers mainly urban and suburban zones, where last-mile delivery difficulties are quite common and may offer major efficiency improvement by using drone delivery.
2. **Technological Implementation:** Development and testing of an RL model designed to optimize delivery routes for drones and ground vehicles. This will include delivery scenario simulations for the effectiveness appraisal of the model under different urban logistic conditions.
3. **Scalability and Adaptability:** This will be specific to the urban centers of Kazakhstan. However, the research model prescribes that its results can be replicated, scaled, and adapted as a general tool for other urban densities and the geographic regions of the country.
4. **Drone Integration:** The research examines the pragmatics of integrating drone technology in the current delivery framework, considering operational protocol, safety standards, and compliances as per the present aviation and local government regulations.
5. **Environmental Impact:** It seeks to assess the reductions in potential carbon emissions and other benefits to the environment, if any, from drone delivery, as opposed to the use of conventional vehicles for delivery.

**Limitations of the Study** This study seeks to recognize some of the limitations, while great insights are sought to be provided in the optimization of the last mile delivery:

1. **Data Availability:** The model’s efficiency through Reinforcement Learning is overwhelmingly based on the availability and quality of data. In real-world use cases, logistic operations may limit data collection due to privacy; for example, there may be no recent use of drones or availability of such data for commercial delivery operations.
2. **Regulatory Constraints:** These delivery drones have to go through regulatory approval that is really strict, differing in various regions and countries. It is such regulations that can, therefore, limit the full realization and testing of drone delivery systems within certain urban environments.
3. **Technology Adoption:** The integration of drone technology assumes a level of technological adoption and infrastructure readiness that may not be present in all target areas. Differences in technological adoption rates could affect the feasibility and scalability of the proposed solutions.
4. **Generalizability:** The findings from this study may not be universally applicable to all e-commerce logistics scenarios, particularly in rural or less densely populated areas where the logistics dynamics differ significantly from urban settings.
5. **Cost Implications:** While the study will explore the cost-effectiveness of drone deliveries, initial setup costs and ongoing maintenance for drones could be prohibitive, affecting the practical implementation of such systems in a real-world business context.

**Addressing Limitations** To address these limitations, the research will employ robust simulation techniques to test the RL model under various controlled scenarios, ensuring a comprehensive evaluation of its capabilities and limitations. Additionally, the study will review existing literature and case studies on drone delivery to provide a broader context and validation of the research outcomes.

This section presents the scope and limitations, sets realistic expectations for what the research will cover, and acknowledges the constraints under which the study will operate. This transparency is crucial for framing the research findings and clearly understanding where this thesis’s contributions fit within the broader field of e-commerce logistics optimization.

**Thesis Structure** The thesis is divided into chapters systematically; each chapter has been designed so as to investigate and analyze different aspects of the Vehicle Routing Problem (VRP) with Reinforcement Learning (RL), to target a specific Actor-Critic framework. It provides a logical continuum from basic ideas to advanced applications, which might help understand the VRP in logistics both theoretically and practically.

The thesis shall be briefly outlined as follows:

**Chapter 1: Background Information** This chapter examines the integration of advanced computational methods with traditional routing challenges, emphasizing the effectiveness of machine learning and reinforcement learning techniques. It highlights the deployment of radial basis function networks and reinforcement learning to enhance evolutionary algorithms and refine neighborhood search strategies. Additionally, the chapter explores the application of multi-agent systems, dynamic programming, and clustering algorithms in addressing complex Vehicle Routing Problems (VRP). It stresses the trend towards hybrid methodologies that blend sophisticated learning techniques with conventional heuristics, aiming to develop more efficient and scalable solutions tailored to the dynamic and stochastic nature of VRP challenges. The discussion delves deeper into how these advanced computational techniques not only improve solution accuracy and processing speed but also adapt to and efficiently manage the variability and constraints inherent in real-world logistics scenarios. Through a detailed analysis of current studies and methodologies, the chapter sets the stage for a comprehensive understanding of how modern techniques are transforming the landscape of vehicle routing to meet contemporary demands.

**Chapter 2: Methodology** This chapter delves into the research methodology aimed at enhancing routing strategies by integrating multiple vehicle groups, specifically trucks and drones. The necessity for sophisticated coordination arises from the interdependent decision-making processes among these groups. Conventional single vehicle routing models fall short in managing the dynamic interactions essential in multi-vehicle contexts. To address these shortcomings, the chapter introduces an advanced model termed the attention-encoder group-based LSTM-decoder, which substantially enhances the Hybrid Model (HM). This model leverages an attention-based encoder to thoroughly analyze the entire operational graph, capturing intricate interactions and dependencies. It also incorporates an LSTM-based decoder that integrates augmented inputs to reflect group interactions, thereby enabling the model to make well-informed routing decisions that account for both intra-group coordination and inter-group dynamics. This methodological advancement is designed to optimize routing strategies, improve efficiency, and bolster the robustness of solutions in complex multi-vehicle routing scenarios.

**Chapter 3: Computational Studies** In this chapter, the focus is on the computational studies of the proposed model, specifically using a well-known benchmark dataset for vehicle routing and optimization. This dataset consists of customer and depot locations distributed across a designated area and has been widely utilized in various studies to validate new algorithms. The methodology involves comparing the proposed model with traditional exact and heuristic methods and evaluating the efficiency and quality of solutions through a defined GAP metric. The experiments also consider different routing configurations and the comparative speed of drones and trucks to test the model’s computational efficiency and performance. Additionally, various decoding strategies and computational resources are described to provide a comprehensive overview of the experimental setup and the hyperparameters employed. This approach allows for a detailed assessment of the proposed model’s capabilities in solving complex, dynamic vehicle routing problems effectively.

**Chapter 4: Discussions** This chapter interprets the results within the broader context of RL’s application in logistics. It discusses the implications of the findings for practitioners in the field, addresses potential limitations of the study, and suggests areas for further research based on the observed outcomes. Through a series of detailed simulations under various theoretical conditions, the research has demonstrated that RL models offer substantial improvements over traditional logistics methods in terms of efficiency, adaptability, and scalability. These findings not only affirm the potential of RL to transform logistics operations but also highlight its capacity to contribute to more sustainable practices within the industry.

**Conclusion** and Recommendations The final chapter summarizes the main findings of the thesis, reflecting on the research objectives and the contributions made to the field of logistics optimization. It offers recommendations for logistics managers considering RL solutions and outlines future research directions that could build on this work. While the research has yielded promising results, the practical application of RL in real-world logistics operations remains largely theoretical. Future directions include empirical validation of these models in real-world settings, seamless integration with existing logistics systems, and addressing regulatory and ethical considerations as the use of automated systems expands. In conclusion, this thesis lays a strong foundation for the adoption of reinforcement learning in logistics, showcasing its potential to enhance operational efficiency, adaptability, and sustainability. The continued exploration and development of RL models will be critical in realizing their full potential, potentially revolutionizing logistics operations globally. As industries increasingly move towards automation and intelligent systems, the role of RL in logistics is poised to become a pivotal element in shaping the future of the sector.

**References** Here is a complete list of all scholarly sources and research materials cited throughout the thesis, formatted according to academic standards.

**Appendices** The appendices contain additional material, such as tables of extra experiment results, to support the thesis. These resources offer greater detail on the research’s technical aspects and support the study’s transparency and reproducibility (Appendix A).

This structure guides the reader from a general understanding of the VRP through coherent and logical development to an inquiry into advanced solutions using Reinforcement Learning, focusing on the derivation of practical insights and future directions.

1. **BACKGROUND AND LITERATURE REVIEW**

## Overview of the Vehicle Routing Problem (VRP)

*Historical Context of VRP*First defined by George Dantzig and John Ramser through their innovative paper, ”The Truck Dispatching Problem” [[10](#_heading=h.206ipza)] in 1959, this work developed one of the concepts to optimize routes for delivery vehicles and, besides that, has made a revolution in logistics and operational research, which is under constant dynamism even at the present moment. First intended for the optimization of delivery routes for gasoline to service stations, the application of VRP has grown widely and represents an essential framework for many logistic applications in different industries.

The redefinition of the industry’s approach towards supply chain management, in fact, turned out to be a watershed courtesy of the pathbreaking work of Dantzig and Ramser in the domain of logistics. Their new approach, which was based on mathematical models and precision, brought in huge savings in cost and operational efficiencies. They illustrated how businesses that utilized strategic route planning could cut down significantly on the unnecessary mileage and fuel usage associated with the disposition of goods. Their model would be the basis for the more sophisticated logistic models used today.

With the commercial world expanding and supply chains becoming more complicated, the routing of vehicles assumes paramount importance. Today, the Vehicle Routing Problem (VRP) has been suggested as a means to cater to these emerging challenges and different versions of the VRP have been proposed over time to serve the needs of particular industries. For example, the Capacitated Vehicle Routing Problem (CVRP) considers the vehicle capacity constraint, while the Vehicle Routing Problem with Time Windows (VRPTW) considers specific delivery time windows. These have been crucial alterations in tailoring the VRP to the specific needs of sectors like the postal service and distribution retail networks, where precision in timing and load management are required. As a result, the VRP has become a fundamental tool in optimizing vehicle routing along complex supply chains [[11](#_heading=h.4k668n3)].

It was in the latter half of the 20th century and marked the beginning of the 21st century when the integration of real-time data and communication technologies was integrated. As a result, with this new integration, there came the advent of an all-new era termed Dynamic Vehicle Routing Problems (DVRP). This means that the changes of the current time would dynamically influence the updated routes; this is to manifest either from the conditions of traffic, requests from customers, or the status of a vehicle. This is indispensable with respect to the modern logistics, specifically in the area of retail and e-commerce, where timely delivering of products plays a very vital role in customer satisfaction and retaining a business’s reputation [[12](#_heading=h.2zbgiuw)].

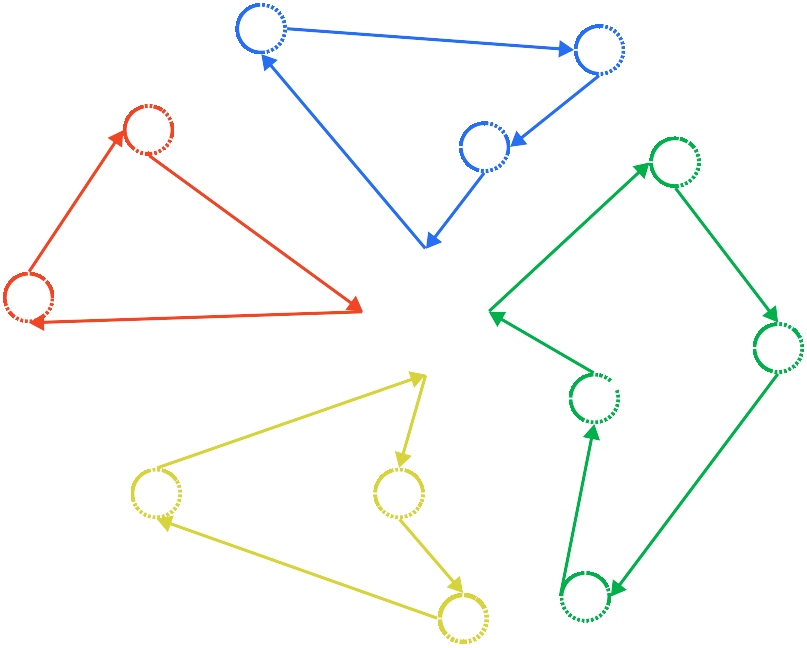
Modern VRP research applies artificial intelligence, mainly using machine learning and reinforcement learning techniques. The Actor-Critic approach is one of the different adaptations designed to learn the effective optimal routing strategies of their decision-making process from trial and error based on operational feedback from real-world experience. This is particularly helpful in a logistics environment, which is uncertain when static traditional models cannot efficiently match modern delivery activity, which is evidently described by their variability and complicated nature [[13](#_heading=h.1egqt2p)].

The concept of Vehicle Routing Problem (VRP) has evolved from basic to advanced AI-driven models, marking improvements in computational and algorithmic methods. This shift also signifies a change in the logistics industry’s focus from mere cost reduction and efficiency to adaptability, sustainability, and customer-centric service models. The dynamic global logistics of the 21st century are continuously transforming due to new technologies and changing consumer expectations, however, VRP remains an essential area for academic research and practical application. It offers the possibility of discovering new, creative solutions to some of the most complex problems presented by modern times.

## Variants of the Vehicle Routing Problem (VRP)

*Vehicle Routing Problem (VRP)*The Vehicle Routing Problem (VRP) is a mathematical problem that aims to find the optimal way to route a fleet of vehicles to visit a set of customers with non-negative demands while minimizing the total cost of the routes. In VRP, we consider an undirected graph with a set of nodes, V={v0, . . . , vn}, that represents the locations of the customers and a depot, where the vehicles start and finish their routes. The set of edges, E, represents the connections between the nodes. The VRP requires that each customer is visited only once by one of the M identical vehicles, which have a limited capacity. The goal is to determine the optimal set of routes that minimizes the total distance traveled by the vehicles and satisfies the capacity constraint. Different constraints can be added to VRP to reflect the realistic routing challenges faced by delivery companies [[14](#_heading=h.3ygebqi)]. The main components are shown in figure 1.1 and are as follows:

1. *Depots:* These are the storage points, places for repair, and dispatching points for vehicles. The central location of depots has to be very effective in enhancing operational efficiency concerning minimum travel time and expenses. Efficient depot management will have to sum up not only the ideal inventory levels attained but also the synchronization of dispatch schedules in tandem with delivery demands that ensure the flow of goods to customers in a timely and cost-effective manner.
2. *Vehicles:* Vehicles are central to the VRP as they carry out the delivery of goods. Factors such as the volume and character of goods, route characteristics, and delivery windows will determine the composition and capabilities of the vehicle fleet, from large trucks to compact delivery vans. Choices must be carefully made based on fuel consumption, maintenance costs, and suitability to the delivery environment for each vehicle. For example, electric vehicles may be used in urban deliveries to reduce emissions, but range and charging issues must still be incorporated into the route planning.
3. *Customers:* The arriving customers form the delivery route bases. The locations and demands define the endpoints and structure of the routes. Understanding their needs, including the amount of product required and the timing of deliveries, is critical. The challenge is to optimally cluster customer locations and plan delivery schedules so that there is a minimum distance covered but with great service quality. Customer satisfaction often hinges on timely and accurate deliveries, which depend on well-planned routes that consider traffic patterns and customer availability.
4. *Routes:*Routes define the path vehicles will follow to service customers. Optimizing these routes is the essence of solving the VRP since the travel distance and time between various points must be minimized. This involves complex decision-making to design stops in a manner that conforms to vehicle capacities and operational constraints, such as time windows for delivery and traffic conditions. The use of sophisticated routing algorithms can dramatically improve route efficiency by adapting to real-time changes and reducing the operational footprint.
5. *Constraints:* VRP is subject to multiple constraints that ensure the feasibility and efficiency of delivery routes. These range from vehicle load limits to delivery time windows, driver working hours, and even road weight limits. This calls for sophisticated solutions that can balance various logistical needs so that the routes are not only cost-efficient but also conform to regulatory and operational standards.
6. *Objective Function:* This guides the optimization process in VRP, detailing what the solution is supposed to achieve. Generally, the objective function involves balancing operational cost minimization or reduction in travel time with customer service improvement. These goals often need to be balanced against each other, especially when operational constraints limit flexibility. For example, reducing delivery times may increase costs, or maximizing vehicle loads could affect service quality. This makes the objective function a critical element of VRP, as it encapsulates the strategic goals of logistics operations.



1

2

2

3

1

1

2

4

3

1

2

3

Figure 1.1 – Vehicle Routing Problem (VRP) Network

Understanding these components in depth allows logistics managers to devise more effective and adaptive routing strategies, directly impacting the efficiency and success of supply chain operations. Businesses can achieve a harmonious balance between operational demands and service excellence by addressing each component thoughtfully and strategically.

Exploring VRP Variants After understanding the basic components, we can explore the specific VRP variants that address different operational challenges and industry-specific needs:

1. *Capacitated Vehicle Routing Problem (CVRP)* ensures that no established route for a fleet of delivery vehicles exceeds the cargo capacity of any vehicle. This makes it one of the basic forms of the Vehicle Routing Problem (VRP), as most delivery systems need to consider the physical constraints on the capacity of each vehicle. The primary objective of CVRP is to find a set of routes that minimizes overall travel costs (often measured in distance or time) in such a way that the sum of the loads on each route does not exceed the vehicle’s carrying capacity. For example, CVRP addresses practical cases, such as a distributor needing to deliver goods to supermarkets. Each truck has a maximum payload, either by weight or volume, and each supermarket has a specific demand. The goal is to minimize the total distance traveled by the entire fleet while ensuring that no truck is overloaded. The challenge in solving CVRP lies in balancing between finding the shortest possible route and fairly distributing the load among the vehicles. Techniques used range from simple heuristics, like the nearest neighbor approach to start building a route, to more complex metaheuristics, such as genetic algorithms and particle swarm optimization, which can provide near-optimal solutions for larger and more complex datasets. CVRP is pivotal in industries ranging from retail distribution, where products must be delivered to various outlets, to waste collection, where trucks collect trash within their capacity limits. Its applications are vast and critical to operational efficiency in logistics and distribution networks.
2. *Vehicle Routing Problem with Time Windows (VRPTW)*introduces an added complexity because each customer must be served within a preassigned time window [[15](#_heading=h.2dlolyb)]. This variant is highly applicable in cases where timing is of the essence, such as delivering perishable goods and pharmaceuticals or providing in-house services, with customers expecting visits within preassigned times. In VRPTW, the planner must ensure the carrying capacity of the vehicle against the length of the route and, at the same time, ensure compliance with the assigned delivery or service times at each stop. This can lead to a situation where a vehicle either waits for the start of a time window or, conversely, has to speed up service at other stops to arrive at a customer within the window. Successfully solving VRPTW typically requires the use of advanced heuristic or metaheuristic techniques that can effectively handle the dual constraints of routing and timing. These could include adaptations of traditional VRP solutions enhanced by time-checking mechanisms or entirely bespoke approaches like time-oriented ant colony systems, which mimic the way ants search for efficient paths through the problem space. The significance of VRPTW extends to healthcare (home visits by medical practitioners), e-commerce (customers expecting deliveries at convenient times), and even public services, such as school buses or municipal services operating within tight schedules.
3. *Dynamic Vehicle Routing Problem (DVRP)* addresses VRP in a changing environment where customer demands, delivery locations, or other elements may change in real-time [[16](#_heading=h.sqyw64)]. This becomes increasingly relevant with the growth of e-commerce, where orders are received dynamically, and logistics providers must adapt routes ”on the fly” to accommodate new or changing orders. The solution in DVRP is not static but continues to update as new information becomes available. This might involve incorporating last-minute orders into existing routes, adjusting routes in response to traffic conditions, or re-optimizing to cover for a vehicle that has developed a mechanical fault. The solutions for DVRP often involve sophisticated algorithms that can quickly recalculate routes based on real-time inputs from technologies like GPS tracking, IoT devices, and advanced predictive analytics to forecast and respond to changes. Thus, DVRP is of utmost importance to courier services, on-demand product deliveries, and any logistics operation that requires a high level of flexibility and rapid response to maintain service levels in the dynamic operational context.
4. *Multi-Depot Vehicle Routing Problem (MDVRP)* arises when more than one depot is used for vehicle dispatch during any operation – it does not have a fixed origin point. In this variant, every vehicle is assigned to a depot, and its starting and finishing locations must fall within one depot. This complicates the routing decisions due to the involvement of multiple starting and ending points. The primary challenge in MDVRP is the perfect matching of vehicle allocation to depots and customers to vehicles to minimize costs, which typically includes travel distances and time. MDVRP is applicable when an organization operates from regional distribution centers or has multiple storage locations. For example, a national retail company might have several warehouses across the nation to serve different regions efficiently. The solution ensures that there are effective routes from each depot to serve customers within the region, maintaining optimum vehicle capacity loads and reducing overlap between regions served by different depots. Solving MDVRP can be complex, involving algorithmic strategies that consider not only the routing of vehicles but also the initial allocation of vehicles to depots. Solutions are often found using techniques like cluster-first route-second, where customers are first clustered by proximity to depots and then routed, or through advanced metaheuristics like adaptive extensive neighborhood search. Strategic management of MDVRP significantly enhances operational efficiency by reducing travel times and costs, improving service levels, and better utilizing resources across multiple depots. This is crucial for large logistics networks to deliver goods rapidly and cost-effectively over large geographic areas.
5. *Green Vehicle Routing Problem (GVRP)* addresses the challenges arising from environmental pollution by focusing on reducing the environmental impact of vehicle routes. This issue is becoming increasingly important as firms are obligated to reduce their carbon footprints. GVRP presents an eco-friendly alternative that incorporates environmental targets within the traditional Vehicle Routing Problem (VRP) framework. These targets typically involve reductions in fuel consumption, emissions, or the use of alternative fuel vehicles. GVRP is a dual-objective problem that considers both economic and ecological costs, aiming to achieve cost efficiency and environmental sustainability simultaneously. In practical terms, GVRP could involve planning routes to minimize idle times and fuel consumption or scheduling specific routes where electric or hybrid vehicles are prioritized. For companies with a strong commitment to sustainability, managing GVRP may include strategic decisions about vehicle purchases and fleet composition, focusing on investments in greener technologies. This approach not only enhances efficiency but also aligns with broader environmental goals. Most methods for solving the Green Vehicle Routing Problem (GVRP) fall within multi-objective optimization frameworks, which are designed to balance different types of costs. Approaches such as genetic algorithms, simulated annealing, and other metaheuristic techniques can be tailored to accommodate the added complexity of environmental objectives. This adaptation enables the development of solutions that are not only economically sustainable but also environmentally conscious.
6. *Stochastic Vehicle Routing Problem (SVRP)* addresses uncertainties inherent in logistics, such as unknown customer demands, unpredictable road conditions, or varying service times. SVRP aims to develop strong and flexible routes that can accommodate these variations as they occur, using probability distribution models to predict such events. The main task in SVRP is to design routes that optimize expected performance, taking into account the probability of various scenarios. This may involve creating spare capacity on vehicles, choosing routes with less traffic variability, or planning buffer times into schedules. It’s crucial not to rely only on the best-case scenario for each route but to prepare for a range of possible outcomes. The approaches used for solving SVRP include scenario-based planning and robust optimization techniques. These methods can forecast potential issues and make route adjustments either in advance or on the delivery day. Tools commonly used to handle the variability and ensure that logistics operations remain adaptable and responsive to changes include Monte Carlo simulations, dynamic programming, and robust optimization.
7. *Vehicle Routing Problem with Drones (VRPD)* introduces unmanned aerial vehicles, commonly known as drones, into traditional vehicle routing frameworks as shown in figure 1.2. This integration aims to optimize the delivery process by combining the speed and accessibility of drones with the carrying capacity of ground vehicles. VRPD is particularly effective in urban areas and remote rural locations where conventional vehicles might face accessibility issues. In VRPD, drones can be launched from host vehicles or depots to complete the last-mile delivery of small, time-sensitive packages, significantly reducing delivery times. The main challenges include managing operational dynamics between drones and vehicles, such as optimizing drone flight paths for minimal energy consumption and coordinating the launch and retrieval of drones while adhering to regulatory and safety standards. Solving VRPD involves complex decision-making about which deliveries should be assigned to drones versus vehicles based on factors like package size, delivery priority, and geographical constraints. Advanced algorithms extend traditional VRP solutions with drone deployment strategies and integrate route planning with aerial path optimization. Simulation and heuristic-based approaches, as well as more sophisticated metaheuristics like hybrid evolutionary algorithms, are commonly employed to find practical solutions. VRPD is crucial for industries requiring timely deliveries, such as pharmaceuticals and food, offering cost savings and environmental benefits by reducing road travel and associated emissions.

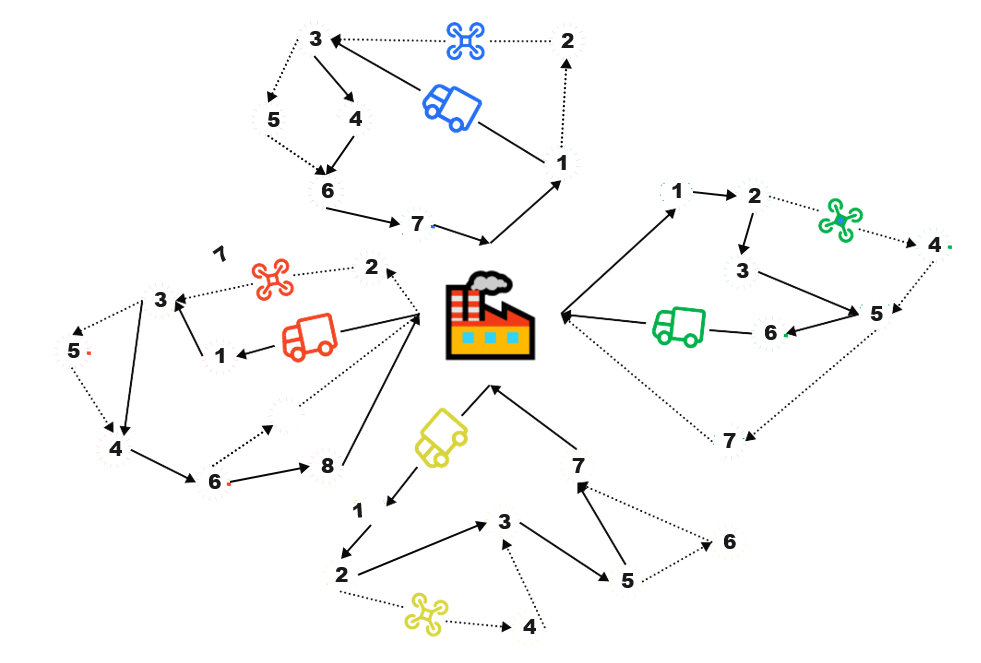


Figure 1.2 – Vehicle Routing Problem with Drone (VRPD) Network

Every version of the Vehicle Routing Problem provides distinct ideas and solutions to the intricate difficulties faced by modern logistics and supply chain management. From the essential concerns of capacity in CVRP to the creative incorporation of drones in VRPD, these variants cater to the varied requirements and changing dynamics of global distribution networks. They equip planners with the necessary tools to optimize routes that balance efficiency, cost, service quality, and sustainability.

As we move forward, we will delve deeper into specific methodologies and technologies developed to solve various variants of Vehicle Routing Problem (VRP). This will include the exploration of computational algorithms, the role of artificial intelligence in enhancing decision-making, and the impact of technological advancements like IoT and real-time data analytics on VRP solutions. The upcoming chapter will also highlight case studies and real-world applications demonstrating the practical implications and benefits of optimizing vehicle routing in different industrial and commercial contexts. This comprehensive exploration aims to enhance understanding of the theoretical underpinnings and showcase the logistics sector’s practical relevance and transformative potential

## Methodological Advances in Solving VRPs

### 

### 1.3.1 Traditional Approaches

*Exact Methods:* Exact methods are designed to guarantee the optimal solution to the Vehicle Routing Problem (VRP) by systematically exploring all potential solutions or proving that no better solutions exist. Given all variables and constraints, these methods ensure the most efficient route is identified:

1. *Integer Programming (IP)* is another exact method widely used in solving the Vehicle Routing Problem (VRP). In IP, the VRP is formulated as an integer linear program that utilizes linear equations to encode constraints such as vehicle capacity, customer demands, and route continuity. This approach is particularly effective for smaller datasets and is often used alongside cutting-plane methods to reduce the search space, enhancing efficiency and solution precision.
2. *Branch-and-Bound (B&B)* is a problem-solving technique that initially involves breaking down a complex problem into smaller, simpler subproblems (branching), and then calculating bounds to evaluate these subproblems. Bounds help to prune away ineffective branches, thereby improving the method’s efficiency. Branch-and-Bound is particularly suitable for Vehicle Routing Problems (VRP) where precise solutions are necessary. However, its applicability is generally limited to more minor instances due to computational constraints.
3. *Heuristic Methods*Heuristic methods are indispensable for efficiently solving Vehicle Routing Problems (VRP), providing solutions that are nearly optimal within a reasonable timeframe [[17](#_heading=h.3cqmetx)]. These methods are widely utilized due to the inherent complexity of VRPs and the practical necessity for feasible solutions over theoretically exact ones [[18](#_heading=h.1rvwp1q)]. The primary aim of these methods is to rapidly generate feasible solutions, even if they are not the globally optimal solutions. Heuristics play a critical role in addressing the challenges of VRP efficiently and effectively, helping to navigate the intricacies of routing logistics within manageable computational limits.
4. *Heuristics* are commonly used in routing to create simple and easy-to implement solutions. Popular techniques such as the Nearest Neighbor, which selects the closest unvisited location at each step, and the Cheapest Insertion Method, which prioritizes the lowest cost position to insert the next customer, are efficient in generating solutions. These methods offer simplicity and speed in solution generation but may not fully explore the entire solution space. This limitation is noted in studies like Yuliza et al. [[19](#_heading=h.4bvk7pj)], highlighting that while these methods are efficient, they do not al-ways yield the most optimal solution due to their simplicity. Additionally, Stopka et al. [[20](#_heading=h.2r0uhxc)] found that the Nearest Neighbor method was less advantageous for certain routes. To address these limitations, researchers have explored adaptive approaches such as the adaptive extensive neighborhood search to enhance routing solution efficiency, as discussed by Yu et al. [[21](#_heading=h.1664s55)]. Furthermore, Zhu et al. [[22](#_heading=h.3q5sasy)] proposed an Adaptive Elitist Genetic Algorithm with improved neighbor routing initialization to optimize Electric Vehicle Routing Problems. These adaptive methods aim to improve solution space exploration and enhance route quality. In conclusion, while heuristics like the Nearest Neighbor and Cheapest Insertion Method are valuable for their simplicity and speed in generating solutions, researchers continuously develop more advanced and adaptive techniques to overcome the limitations associated with these traditional methods and comprehensively explore the solution space.
5. *Improvement Heuristics:* The 2-opt and 3-opt algorithms are heuristic methods that refine solutions locally within route optimization processes. These algorithms work by iteratively swapping two or three segments of routes to minimize the total route distance, enhancing solutions originally generated by other algorithms or heuristics [[23](#_heading=h.25b2l0r)]. The 2-opt algorithm is a classical local search method known for accelerating convergence and enhancing solution accuracy, widely applicable in graph optimization challenges [[24](#_heading=h.kgcv8k)]. The 3-opt algorithm, a specific case of the K-Opt algorithm where K=3, has been integrated into the Ant Colony Optimization with Visibility Adaptation (ACOAV) to bolster its efficacy in solving the Traveling Salesman Problem [[25](#_heading=h.34g0dwd)]. Additionally, the 2-opt algorithm has been effectively combined with evolutionary algorithms to enhance the quality of solutions in various optimization scenarios [[26](#_heading=h.1jlao46)]. It has also been applied in hybrid genetic algorithms to optimize solutions for the Asymmetric Distance-Constrained Vehicle Routing Problem. In summary, the 2-opt and 3-opt algorithms are pivotal in refining solutions for optimization problems, particularly in route optimization. By adjusting route segments iteratively, these algorithms significantly improve the quality of solutions generated by different optimization techniques. This efficiency underscores their importance in contemporary optimization strategies.
6. *Metaheuristics* play a crucial role in efficiently solving large and complex Vehicle Routing Problems (VRPs) where exact methods are impractical. Techniques such as the Bees Algorithm, Iterated Local Search, and Harris Hawks Optimization employ flexible and scalable problem-solving approaches, utilizing intensification and diversification strategies to enhance solution quality [[27](#_heading=h.43ky6rz)-[29](#_heading=h.xvir7l)]. These methods have become widely adopted due to their robust local and global search capabilities, which enable them to find optimal solutions for VRPs effectively [[29](#_heading=h.xvir7l), p. 1-19]. They are particularly effective in escaping local optima by extensively exploring the search space [[30](#_heading=h.3hv69ve)]. Additionally, metaheuristics are invaluable for identifying new solution neighborhoods and providing practical solutions for real-world VRP instances [[31](#_heading=h.1x0gk37), [32](#_heading=h.4h042r0)]. The use of metaheuristics in VRPs is prevalent, often being favored over exact methods and classical heuristics due to their effectiveness and efficiency in handling complex optimization scenarios [[33](#_heading=h.2w5ecyt)]. These methods not only offer a robust framework for tackling VRPs but also ensure that solutions are both practical and adaptable to real-world conditions.
7. *Genetic Algorithms (GAs)* have been consistently recognized, both in literature and practice, as powerful tools for solving complex optimization problems like the Vehicle Routing Problem (VRP). This problem involves determining the most efficient routes for a fleet of vehicles delivering goods to customers, a task characterized by its combinatorial complexity [[34](#_heading=h.1baon6m)]. Research has explored various enhancements to traditional GAs to address different aspects of the VRP. For example, hybrid genetic algorithms have been developed by integrating GAs with other intelligent optimization algorithms, such as ant colony optimization, to tackle challenges in problems like the Multi-Depot Green Vehicle Routing Problem [[35](#_heading=h.3vac5uf)]. These hybrid approaches aim to combine the strengths of different algorithms to improve the quality of solutions and accelerate convergence.

### 1.3.2 Learning methods

Classifying the broad spectrum of learning methods used to solve routing problems is challenging due to the diversity of approaches and their applications. However, an effective way to categorize these methods is by examining the structure of the solutions they generate as shown in figure 1.3. This classification distinguishes between two main types: pure learning methods and hybrid methods.

Pure learning methods, often called end-to-end learning methods, utilize either Supervised Learning (SL) or Reinforcement Learning (RL) to manage the entire problem-solving process from start to finish. These methods rely solely on learning algorithms to formulate and refine routing solutions, making them well-suited for scenarios where historical data and predictive modeling can directly inform routing decisions.

In contrast, hybrid methods combine learning algorithms with traditional, nonlearning optimization techniques. In these approaches, learning methods might be used initially to develop feasible and efficient routing solutions. These initial solutions are further refined using traditional construction heuristics, enhancing their practicality and effectiveness. Alternatively, learning algorithms might be applied to solve specific complex sub-problems within a larger heuristic framework, thereby improving the overall problem-solving capability of traditional methods. Figure 1.3 summarizes the taxonomy of learning methods to solve VRPs.

Learning methods for Routing Problems

End-to-End

Hybrid

Supervised Learning

Reinforcement Learning

Q-learning

Policy gradient methods

Actor-Critic

Learning methods

for facilitating non-learning methods

Construction heuristics for improving learning methods

Figure 1.3 – The taxonomy of learning methods for VRPs

Routing problems can also be classified based on the number of vehicles involved, another crucial aspect in the design of routing algorithms. Single-vehicle routing problems (SVRPs), such as the Traveling Salesman Problem (TSP), the Capacitated Vehicle Routing Problem (CVRP), and the Vehicle Routing Problem with Time Windows (VRPTW), involve determining the optimal route for one vehicle to visit several locations. Multi-vehicle routing problems, on the other hand, involve coordinating multiple vehicles, necessitating more complex solutions that address the additional challenges of fleet management and route synchronization. These are often managed using a central controller to ensure efficiency across heterogeneous or homogeneous fleets. In the literature, these are denoted as mTSP, mCVRP, and mVRPTW when multiple vehicles are incorporated into the traditionally single-vehicle scenarios.

This dual approach to classification by solution structure and by vehicle number provides a clear framework for understanding and researching the diverse methodologies in vehicle routing, accommodating the vast array of strategies ranging from purely theoretical models to practical applications in complex logistical environments.

*End-to-End Supervised Learning for Vehicle Routing Problems (VRP)*

Supervised Learning (SL) is an approach used in Vehicle Routing Problems (VRP) that relies on high-quality, pre-computed solutions to serve as essential training labels. These labels contain optimal or near-optimal solutions to VRP instances, which the SL model uses to internalize effective routing strategies and patterns. The model trains on carefully selected datasets to understand which routing decisions typically yield the most cost-effective, time-efficient, or otherwise desirable outcomes based on specific operational criteria.

The accuracy of the training labels is critical for achieving optimal performance. These labels are often generated using exact methods or sophisticated heuristic algorithms. Therefore, the quality of the data used to train the SL model is essential for it to effectively generalize these learned strategies to new, unseen VRP scenarios. The better the quality of the training labels, the more skilled the model becomes at predicting optimal routes under various circumstances, thereby enhancing its effectiveness and applicability in real-world routing tasks.

Further, depending on how solutions are produced, they are divisible into autoregressive (AR) and non-autoregressive (NAR) categories. In AR SL methods, routes are constructed step by step, node by node, while in NAR methods, the routes are built in zero-shot, meaning the entire route is generated at once without iterative steps. This distinction is crucial as it affects the flexibility and speed of the routing process, with AR methods generally offering more detailed control over the sequence of locations, and NAR methods providing quicker, though potentially less refined, solutions.

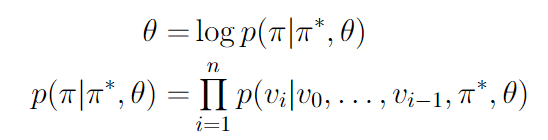
In addressing the Traveling Salesman Problem (TSP), the Non-Auto-regressive (NAR) approach has ushered in a transformative framework for generating solutions. This approach relies heavily on classical solvers such as the Held-Karp algorithm, Concorde, or LKH3, known for their ability to deliver optimal or near-optimal solutions. These solver outputs are used as benchmarks for training more advanced machine learning models. Specifically, for any given graph G(V, E) set within a two-dimensional Euclidean space, the outputs from these solvers π∗ are converted into a binary adjacency matrix, where for each ei,j ∈ E we can compute the probability pi, j to simplify the problem for further processing by deep learning models: pi,j = 1 if ei,j is in π∗, and pi,j = 0 otherwise.

A deep learning model, parameterized by θ, is then trained to predict this adjacency matrix. It learns to assign probabilities i,j for each edge ei,j ∈ E, indicating the likelihood of each edge being part of the optimal solution. The model’s training is driven by a binary cross-entropy loss function, which measures the divergence between the model’s predicted probabilities and the actual binary values from the solver’s output:

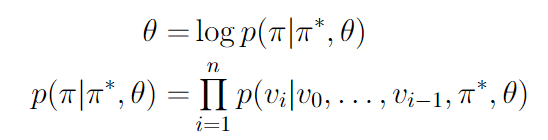
L(*θ*) = *pi,j* log *p*ˆ*i,j* − (1 − *pi,j*) log(1 − *p*ˆ*i,j*) (1.1)

This loss function effectively pushes the model towards higher accuracy in predicting the optimal route configuration. Depending on the solution strategy, the model may employ a greedy search that selects edges based on the highest probabilities or implement a beam search to keep multiple candidate solutions, thereby enhancing the selection process for the final tour.

Conversely, autoregressive (AR) models for Supervised Learning adopt a sequential approach where the model generates partial routes by building on an existing high-quality solution π∗ = [v0, . . . , vn]. These models focus on maximizing the conditional probability of the route given the predetermined optimal or near-optimal sequence:



(1.2)



(1.3)

This methodology emphasizes the importance of each routing decision’s context, allowing the model to learn from the incremental construction of proven optimal routes. Both modeling approaches leverage cutting-edge neural network technologies to enhance their predictive capabilities. Graph Neural Networks (GNNs) and Graph Convolutional Networks (GCNs) are particularly useful for extracting relational patterns and dependencies within the graph data. Furthermore, Memory Augmented Neural Networks and Recurrent Neural Networks (RNNs) facilitate the handling of sequence data, which is crucial for maintaining context and continuity across decisions in AR models.

These advanced models are meticulously trained across varied graph sizes, ensuring their ability to generalize and deliver near-optimal solutions consistently for different instances of TSP. This ongoing refinement and application of machine learning techniques highlight a robust development trajectory in solving complex routing problems like TSP with increased efficiency and precision. Table 1.1 summarizes studies focused on end-to-end supervised learning and continuity across decisions in AR models.

Table 1.1 – The summary of supervised learning methods with autoregressive (AR) and None-autoregressive (NAR) solutions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Problems | AR | NAR | Loss | Label |
| Joshi et al. | TSP | – | ✓ | Cross-entropy | Concorde |
| Milan et al. | TSP | ✓ | – | Log-likelihood | Nearest Neighbours |
| Prates et al. | a decision TSP | – | – | Binary cross-entropy | Concorde |
| Joshi et al. | TSP | ✓ | – | Cross-entropy | Concorde |
| Joshi et al. | TSP | ✓ | ✓ | Cross-entropy | Concorde |
| Van Knippenberg et al. | CVRP, TSP | ✓ | – | Cross-entropy | Held-Kalp algorithm, LKH-3, Concorde |
| ✓ – belongs to the class  Note – Compiled from sources [[36](#_heading=h.2afmg28)-41] | | | | | |

These advanced models are meticulously trained across varied graph sizes, en- suring their ability to generalize and deliver near-optimal solutions consistently for different instances of TSP. This ongoing refinement and application of machine learning techniques highlight a robust development trajectory in solving complex routing problems like TSP with increased efficiency and precision. Table 1.1 summarizes studies focused on end-to-end supervised learning.

*End-to-End Deep Reinforcement Learning (DRL) for Vehicle Routing Problems (VRPs)*

End-to-end Deep Reinforcement Learning (DRL) provides a robust framework for addressing Vehicle Routing Problems (VRPs) by enabling models to learn optimal policies through interactions with a simulated environment. This approach overcomes the limitations of relying on predefined rules to solve complex problems. Nazari et al. [[42](#_heading=h.1302m92)] demonstrated the effectiveness of an end-to-end framework for solving VRPs using DRL, outperforming classical heuristics and Google’s optimization tools. Additionally, proposed an end-to-end DRL framework for the Electric Vehicles Routing Problem with Time Windows (EVRPTW) [[43](#_heading=h.3mzq4wv)].

DRL methods in VRP applications are categorized into value-based and policy- based approaches. Value-based methods focus on learning action values in each state to maximize expected returns, while policy-based methods directly learn the state-action mapping. These methods are used to solve VRP problems represented as Markov Decision Processes (MDPs). The rollout algorithm, part of the Approximate Dynamic Programming (ADP) approach for MDPs, is applied to Multi-Depot Dynamic Vehicle Routing Problems with Stochastic Road Capacity. Furthermore, DRL involves training effective control policies using deep neural networks. Using DRL in VRP applications offers advantages such as enhanced efficiency and performance in solving combinatorial problems.

Utilizing specialized algorithms such as Deep Q-Networks (DQN) for value-based strategies and Proximal Policy Optimization (PPO) for policy-based strategies is crucial for optimizing complex routing decisions dynamically. DQN, based on Q-learning, has evolved beyond state discretization, making it suitable for modeling and optimizing complex routing decisions [[44](#_heading=h.2250f4o)-[47](#_heading=h.1gf8i83)]. The refinement of Deep Q-Networks (DQN) involves utilizing n-step Temporal Difference (TD) learning to calculate rewards that accrue over multiple steps, enriching the model’s capacity to evaluate the long-term consequences of decisions. This technique, coupled with experience replay, allows the model to re-utilize previous iterations to refine the policy by reassessing past actions and their outcomes. This is based on tuples consisting of the state at time t, the action at time t, the cumulative reward from time t to t + n, and the state at time t + n. The cumulative reward from time t to t + n is the sum of the rewards at each step from time t to time t + n.

The Q-value update is as follows: The estimated Q-value for the state at time t and the action at time t is updated by adding the learning rate times the difference between the target value and the current estimated Q-value. The target value is the sum of the cumulative reward and the discounted maximum estimated Q-value for the state at time t + n and the action at time t + n.

Experience replay is an innovative method employed in Reinforcement Learning (RL) that enhances the learning process by saving a selection of the agent’s past experiences and revisiting them during training. As highlighted by Dai et al. [[48](#_heading=h.40ew0vw)], this approach involves drawing a batch of these stored experiences to update the parameters of the neural network. This method accelerates the convergence of the network, making it particularly effective in tackling combinatorial optimization problems, where neural networks serve as approximation functions [[49](#_heading=h.2fk6b3p), [50](#_heading=h.upglbi)]. This technique ensures a more stable and efficient learning trajectory by diversifying the training data, which prevents overfitting to recent experiences and promotes a more robust generalization.

Proximal Policy Optimization (PPO) has shown significant potential to accelerate model convergence, enhancing the speed at which models adapt and optimize decision-making processes in complex environments [[51](#_heading=h.3ep43zb), [52](#_heading=h.1tuee74)]. The integration of advanced neural architectures like Graph Neural Networks (GNNs) further complements this capability by effectively capturing the complex topology of graphs, which is crucial for modeling and refining routing decisions in vehicle routing problems (VRPs) [[53](#_heading=h.4du1wux)-[55](#_heading=h.184mhaj)].

In this structured learning environment, the actor-critic framework utilizes two neural network sets to concurrently learn the policy πθ and the baseline value function Vθ(S0), enhancing the model’s predictive accuracy and decision-making capabilities [[56](#_heading=h.3s49zyc)]. The policy πθ(a|s) is approximated using conditional probabilities and the chain rule. This means that the policy is represented as the product of the probabilities of taking each action at the next step given the current state and a set of visited nodes. The set of visited nodes is denoted as Yt.

The critic’s parameters θc are updated using Mean Square Error, considering the rewards accumulated over a batch of experiences, to refine the policy and improve the decision-making process. This update involves adjusting θc by adding the learning rate times the average gradient of the squared difference between the accumulated reward and the estimated value of the initial state over all experiences in the batch.

Simultaneously, the actor’s parameters θa are fine-tuned using the REINFORCE algorithm, incorporating a baseline to stabilize training and enhance performance. This update involves adjusting θa by adding the learning rate times the average gradient of the log probability of the policy, multiplied by the difference between the accumulated reward and the estimated value of the initial state over all experiences in the batch.

To capture and model the VRP’s complex structure effectively, the encoder- decoder architecture initially developed for machine translation has been adapted. The encoder in VRPs, unlike in machine translation, focuses on embedding the graph structure efficiently to aid the decoder in sampling the next action based on the current state and historical data [[42](#_heading=h.1302m92), p. 1-20; [57](#_heading=h.279ka65)]. This adaptation leverages the similarities between sequencing problems in machine translation and VRPs, where both involve constructing sequences from a given set of elements.

Static encoders process the graph’s initial structure once per instance, capturing both static and dynamic node elements. Specifically, the static node features are captured by a function that processes the static attributes of each node, while the dynamic node features are captured by a function that processes the dynamic attributes of each node.

Dynamic encoders, on the other hand, update the graph embeddings at each timestep, reflecting the dynamic changes induced by routing decisions. This means that at each timestep, the dynamic attributes of each node are processed to update the graph embeddings, ensuring that the model reflects the current state of the graph.

This dynamic encoding allows the model to adapt to new information continuously, enhancing the routing strategy’s responsiveness and efficiency [[58](#_heading=h.meukdy), [59](#_heading=h.36ei31r)]. In vehicle routing problems (VRPs), the accurate representation of both static and dynamic node characteristics, such as geographic coordinates and customer demands, is crucial. This is because VRPs necessitate converting intricate and discrete information across graph nodes and edges into a seamless vector space. Several innovative models have been developed to accomplish this task of graph embedding effectively.

For instance, Bello et al. [[57](#_heading=h.279ka65)] utilizes recurrent neural networks (RNNs) within Pointer Networks to serve as encoders. However, Nazari et al. [[42](#_heading=h.1302m92), p. 1-20] questions the necessity of this approach due to the non-sequential nature of node sets typically found in VRPs.

Independently, Da et al. [[60](#_heading=h.1ljsd9k)] introduced a novel graph representation known as Structure2Vec (S2V), capable of dynamically encoding the graph and its evolving solutions. Additionally, Kool et al. [[56](#_heading=h.3s49zyc)] advocates for a fully attention-based encoder using transformers [[61](#_heading=h.45jfvxd)] to effectively tackle VRPs, while Graph Neural Networks (GNNs), known for their proficiency in graph data interpretation, are employed by Joshi et al., Jiang et al., Senuma et al. [[40](#_heading=h.48pi1tg), p. 70-97; [62](#_heading=h.2koq656), [63](#_heading=h.zu0gcz)]. These innovative encoding strategies have been widely adopted, with Pointer Networks [[64](#_heading=h.3jtnz0s)], multi-head attention [[65](#_heading=h.1yyy98l)-[69](#_heading=h.3x8tuzt)], and other models like S2V [[70](#_heading=h.2ce457m)-[72](#_heading=h.3bj1y38)] being particularly popular. Despite these advancements, no studies have conclusively determined which models excel at graph embeddings for solving VRPs.

The actor-networks decoder, activated at each decision-making step, integrates the current problem state with graph embeddings to dictate subsequent actions. It computes a compatibility score at each timestep to derive action probabilities using a softmax function. Specifically, a function computes the compatibility score, denoted as ut, by applying a sequence of complex, nonlinear transformations to the hidden state and the current state. The action probabilities are then derived by applying the softmax function to the compatibility score. The function *F* represents a series of these complex transformations. RNNs are frequently used to maintain information flow regarding the solution’s sequence, complemented by additive attention mechanisms [[73](#_heading=h.1qoc8b1)] that help align the graph embedding with the current state [42, p. 1-20; [57](#_heading=h.279ka65); [74](#_heading=h.4anzqyu)].

Fully attention-based models are also prevalent, where multi-head attention calculates the compatibility between the state representation and the graph embed- ding [[56](#_heading=h.3s49zyc); [58](#_heading=h.meukdy), p. 636-649; [66](#_heading=h.4iylrwe), p. 12042-12048; [75](#_heading=h.2pta16n)-[77](#_heading=h.3oy7u29)]. A recent innovation by Li et al. [[78](#_heading=h.243i4a2)] introduces a feature embedding refiner between the encoder and decoder to enhance model performance.

Table 1.2 outlines various end-to-end RL methods for VRPs, noting the use of the REINFORCE algorithm with a baseline due to VRPs’ episodic nature. The baseline’s formulation varies, with some researchers [42, p. 1-20; [57](#_heading=h.279ka65); [70](#_heading=h.2ce457m), p. 3806-3816] employing a separate neural network known as a critic to estimate the expected total reward from the initial graph state. In contrast, others [[56](#_heading=h.3s49zyc); [67](#_heading=h.2y3w247), p. 10418-10429] opt for a greedy approach to determine baseline values. Other methods like the Deterministic Policy Gradient and Actor-critic approaches have also been explored [[79](#_heading=h.j8sehv), [80](#_heading=h.338fx5o)].

Despite the efficacy of these trained models, their performance may still fall short compared to traditional heuristics, prompting the use of various search methods to enhance outcomes. Active search [[81](#_heading=h.1idq7dh)] adjusts model parameters to specific instances to improve solutions, while random sampling involves generating multiple solutions and selecting the most effective one to maximize rewards. In summary, the strategic integration of PPO, advanced neural networks, and the innovative use of policy-gradient methods within an actor-critic framework, complemented by sophisticated encoder-decoder architectures, outlines a comprehensive approach to solving VRPs that bridges theoretical research and practical application.

Reinforcement learning (RL) has emerged as a powerful approach for addressing the dynamic and complex scenarios inherent in Vehicle Routing Problems (VRPs) [[82](#_heading=h.3gnlt4p)]. RL systems are trained from their own experience, allowing them to operate in domains lacking human expertise. Applying deep reinforcement learning (DRL) in VRPs has shown promising results, combining theoretical principles with practical execution strategies. DRL models have been developed rapidly in recent years and have significantly impacted the solution approaches for combinatorial optimization problems such as VRPs [[13](#_heading=h.1egqt2p), p. 4754-4771]. Furthermore, RL-based approaches have been proposed to solve VRPs, demonstrating the capability of RL algorithms to learn to make decisions by interacting with the environment [[56](#_heading=h.3s49zyc)].

Table 1.2 – The summary of studies with end-to-end RL methods

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Study | Problems | Training Algorithm | Graph Embedding | Decoder |
| 1 | 2 | 3 | 4 | 5 |
| Bello et al. | TSP | REINFORCE with baseline | Pointer Network | Pointer Network |
| James et al. | DVRP | REINFORCE with baseline | S2V | Pointer Network |
| Dai et al. | TSP | DQN | S2V | - |
| Nazari et al. | TSP, CVRP | REINFORCE with baseline | Elementwise projections | RNN with Attention |
| Xin et al. | TSP, CVRP | REINFORCE with the greedy rollout | Pointer Network, Multi-head atten tion (Dynamic) | Pointer Network, Multi-head attention |
| Vera and Abad | mCVRP | Actor-critic (A2C) | RNN | RNN with Attention |
| Zhang et al. | mVRPSTW | REINFORCE with baseline | Multi-head attention | Multi-head attention |
| Kool et al. | TSP, CVRP, OP, PCTSP, SDVRP, SPCTSP | REINFORCE with baseline | Multi-head attention | Multi-head attention |
| Peng et al. | CVRP | REINFORCE with baseline | Graph Attention Network (GAT) | Multi-head attention |
| Sykora et al. | MAMP | REINFORCE | Linear projection | RNN with attention |
| Xin et al. | TSP, CVRP, OP, PCTSP, SDVRP, SPCTSP | REINFORCE with the greedy rollout | Multi-head attention | Multi-head attention |
| Xu et al. | TSP, CVRP, PCTSP, SDVRP | REINFORCE with baseline | Multi-head attention with gate aggregation | Self-attention with a attentive aggregation module |
| Kim et al. | TSP, CVRP, PCTSP | REINFORCE with greedy rollout | Multi-head attention | Multi-head attention |
| Kwon et al. | TSP, CVRP | REINFORCE with baseline | Multi-head attention | Multi-head attention |
| Emami and Ranka | TSP | Deterministic Policy Gradient (DPG) | RNN | Sinkhorn layer |
| Joshi et al. | TSP | REINFORCE with greedy rollout | Multi-head attention | Multi-head attention |
| Falkner and Schmidt-Thieme | mCVRPTW | REINFORCE with greedy rollout | Self-attention, Linear projection | Multi-head attention |
| Joshi et al. | TSP | REINFORCE with baseline | GNN | Multi-head attention |
| Li et al. | CSP | REINFORCE with the greedy rollout | Pointer Network | RNN with attention |
| Drori et al. | TSP, VRP | REINFORCE with baseline | GAT | Multi-head attention |
| Lin et al. | EVRPTW | REINFORCE with greedy rollout | S2V | RNN with Attention |
| Continuation of table 1.2 | | | | |
| 1 | 2 | 3 | 4 | 5 |
| Li et al. | MOTSP | Actor-critic (A2C) | 1D-Conv | Gated recurrent unit  (GRU) with Attention |
| Li et al. | PDP | REINFORCE with greedy rollout | Multi-head attention | Multi-head attention |
| Li et al. | HCVRP | REINFORCE with greedy rollout | Multi-head attention | Feed-Forward networks, Multi-head attention |
| Duan et al. | CVRP | REINFORCE with greedy rollout, SUPERVISE with policy-sampling | Graph Convolutional  Networks | GRU with context-based attention, Multi-layer Perceptron |
| Note – Compiled from sources [39; 40, p. 70-97; 42, p. 1-20; 48,p. 1-23; 56; 57; 58, p. 636-649; 59, p. 4861-4870; 64, p. 13142-13154; 65, p. 102861; 66, p. 12042-12048; 67, p. 10418-10429; 68, p. 21188-21197; 70, p. 3806-3816; 74, p. 1-5; 75; 79; 80, p. 3103-3114; 82, p. 9300-9309; 82, p. 2306-2314; 83-87] | | | | |

*Hybrid Methods for Vehicle Routing Problems (VRPs)* Hybrid methods for solving Vehicle Routing Problems (VRPs) innovatively merge the analytical strengths of traditional optimization techniques with the adaptive capabilities of machine learning algorithms. This synergy creates two distinct approaches within the hybrid framework, each leveraging the unique strengths of different methodologies to tackle the multifaceted challenges of VRPs. The first approach within the hybrid methodology utilizes machine learning as a supportive tool, seamlessly integrated into traditional optimization methods. Here, learning algorithms are specifically applied to tackle internal subproblems where conventional methods may falter, such as dynamic routing scenarios or complex constraint handling. This integration bolsters traditional algorithms by equipping them with adaptive insights derived from data, enhancing problem-solving strategies and potentially accelerating convergence times.

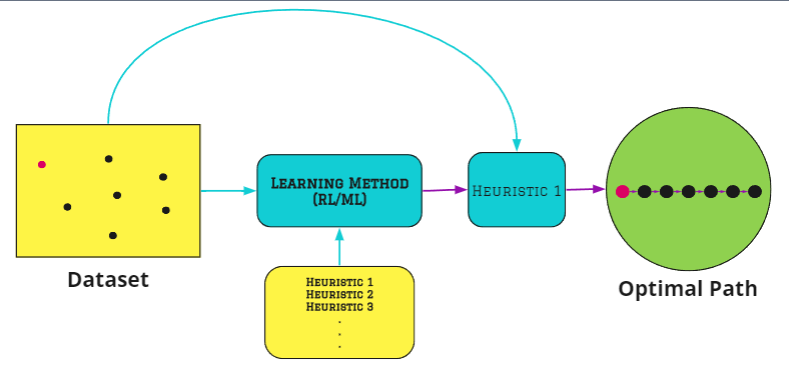


Figure 1.4 – Learning Methods as Improvement Methods used to choose best heuristic for provided dataset

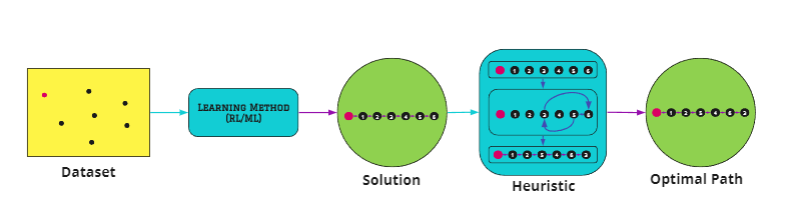


Figure 1.5 – Learning Methods as Improvement Methods used to improve solution generated by Heuristic solution

Figures 1.4, 1.5 the adaptive nature of machine learning fine-tunes solutions in response to real-time logistics data, improving the efficiency and accuracy of routing solutions. For a visual representation of these solutions, which illustrate the integration of machine learning into traditional routing algorithms. On the other side, the second approach positions machine learning as the primary mechanism for generating initial solutions. These initial solutions, often near-optimal, are subsequently refined using tried-and-true construction heuristics. This method harnesses the predictive power of algorithms like neural networks or reinforcement learning models to quickly formulate feasible routing plans that approximate a good solution. Subsequently, construction heuristics are applied to these initial plans to tweak and enhance them, ensuring that the solutions not only meet but exceed operational requirements in terms of efficiency and cost-effectiveness. Moreover, this approach often incorporates additional optimization layers that are tailored specifically to adjust the cost functions used in the learning models. This customization [[88](#_heading=h.2uxtw84), [89](#_heading=h.1a346fx)] ensures that the machine learning models are not only learning from data but are also aligning closely with the specific economic realities of VRP scenarios. For a detailed visualization of how heuristics used to improve solutions generated by machine learning models, refer to figure 1.6. Hybrid methods stand out for their ability to offer the best of both worlds: the innovative, adaptive response of learning methods and the reliability and robustness of heuristic optimizations. By drawing on the strengths of both learning and non-learning methods, hybrid solutions provide a comprehensive framework that can adapt to various complexities of real-world routing challenges. This adaptability makes hybrid methods a compelling choice for businesses looking to optimize their logistical operations effectively, ensuring that they can meet the demands of an ever-evolving marketplace with agility and precision. The fusion of Q-learning with meta-heuristics has catalyzed a transformative approach to solving Traveling Salesman Problems (TSP), beginning with the pioneering work by. This research introduced the concept of utilizing an ant-colony optimization framework in a distributed setting, where each ant is modeled as an autonomous agent responsible for constructing potential solutions. These agents contribute to a shared Q-table, focusing their efforts on shorter, more efficient tours. This innovative approach not only improved solution discovery but also laid the groundwork for enhanced collaborative mechanisms among agents through pheromone updates on graph edges, a technique further refined and expanded upon by Dorigo et al. [[90](#_heading=h.2981zbj)]. The updated approach demonstrated that enhancing cooperative strategies among agents could lead to superior results in solving TSPs, establishing a new benchmark for ant-colony systems integrated with local search techniques.

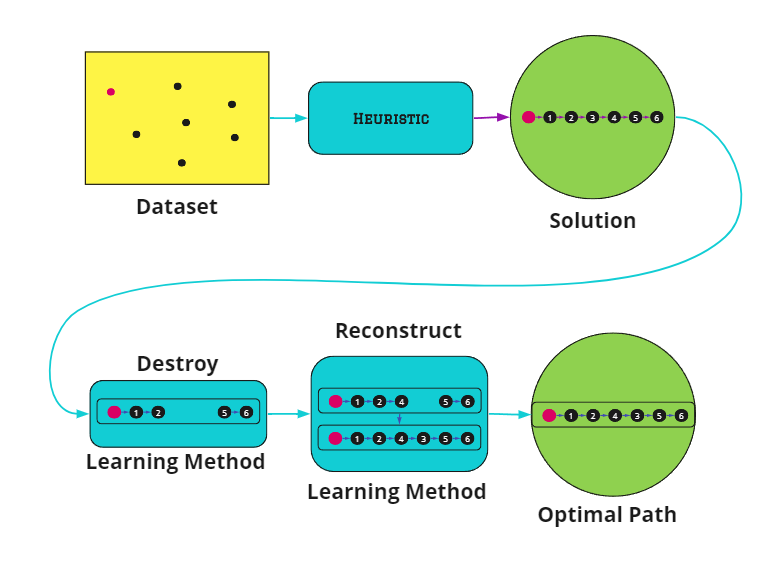


Figure 1.6 – Construction Heuristics to Improve Learning Methods

Continuing this trend, Liu et al. [[1](#_heading=h.odc9jc)] creatively merged genetic algorithms with reinforcement learning to tackle TSPs. By adapting the Q-learning algorithm to work in conjunction with genetic operations, specifically mutations, this hybrid method effectively generated high-quality tours that were subsequently refined through local search processes. This methodology highlighted the versatility of integrating evolutionary concepts with reinforcement learning to enhance solution quality.

In a similar vein, Alipour et al. [[92](#_heading=h.38czs75)] explored the synergistic combination of genetic algorithms and multi-agent reinforcement learning (MARL). Here, MARL served as a robust tool for generating initial solutions that were later polished using genetic algorithms, demonstrating the efficacy of using MARL as a preliminary solution generator in hybrid models. Additionally, Qi et al. [[93](#_heading=h.1nia2ey)] successfully integrated Q-learning into the NSGA-II optimization framework to refine the solution search process further. This approach showcased how strategic enhancements in the search mechanism, through the application of Q-learning, could lead to superior performance compared to traditional evolutionary algorithms that use randomly selected neighborhoods.

The application of radial basis function networks (RBFN) by Niu et al. [[94](#_heading=h.47hxl2r)] represents another significant advancement in enhancing evolutionary algorithms. The study illustrated that a well-calibrated RBFN could rapidly evaluate the fitness of solutions within evolutionary frameworks, substantially speeding up the convergence process and boosting overall algorithm performance by providing precise, real-time evaluations.

Moreover, Hottung et al. [[95](#_heading=h.2mn7vak)] introduced an innovative use of reinforcement learning for neighborhood search strategies aimed at addressing both Capacitated Vehicle Routing Problems (CVRP) and Split Delivery Vehicle Routing Problems (SDVRP). By employing an encoder-decoder model trained to connect segments of feasible solutions actively, this approach significantly outperformed existing models and demonstrated competitive results against advanced heuristic methods like LKH3.

Expanding the utility of reinforcement learning in routing solutions, Silva et al. [[96](#_heading=h.11si5id)] leveraged a multi-agent system to optimize heuristic selection for VRPTW. This study exemplifies the strategic use of Q-learning to sequence neighborhood explorations efficiently, thereby enhancing the effectiveness of local search methodologies.

The intersection of reinforcement learning with dynamic programming opens new avenues for research and application. For instance, Cappart et al. [[97](#_heading=h.3ls5o66)] integrates deep Q-learning and Proximal Policy Optimization with constraint programming to tackle a variety of routing challenges. This integration illustrates the potential of learned policies to overcome the inherent limitations of dynamic programming, particularly in handling complex constraints and large solution spaces.

Clustering algorithms also play a crucial role in decomposing complex routing problems into manageable sub-problems that can be addressed effectively using specialized non-learning methods. Studies like Alesiani et al. [[98](#_heading=h.20xfydz)] and Wang et al. [99] utilize clustering to strategically divide customer bases into clusters that are then tackled using tailored solutions that adhere to specific operational constraints, such as delivery time windows. In recent developments, Ma et al. [[100](#_heading=h.302dr9l)] and Wu et al. [[101](#_heading=h.1f7o1he)] have been at the forefront of designing sophisticated heuristics through deep learning frameworks. These frameworks are adept at embedding and processing complex spatial and relational data from VRPs, enabling the refinement and generation of innovative, efficient routing strategies. This approach not only demonstrates the potential for deep learning to enhance heuristic methods but also underscores the evolving landscape of VRP solutions, where traditional methodologies are being complemented and sometimes surpassed by modern, data-driven approaches.

*Improving Learning Methods with Heuristics*In the realm of complex routing problems, trained learning methods are increasingly being integrated with conventional construction heuristics to create innovative solutions. A notable approach involves using a greedy solution derived from a learning method as a starting point. For example, Deudeon et al. [[102](#_heading=h.3z7bk57)] tackles the Traveling Salesman Problem (TSP) by employing a Multi-Head Attention (MHA)-based encoder and a Pointer Network decoder, further refined by the 2-opt local search algorithm. Similarly, for addressing larger scale Vehicle Routing Problems (VRP) and Vehicle Routing Problems with Time Windows (VRPTW), Zhao et al. [[103](#_heading=h.2eclud0)] utilizes the model developed by Nazari et al. [[42](#_heading=h.1302m92), p. 1-20], enhanced with an adaptive critic, and subsequently augmented with local search techniques like OR-tools and Large Neighborhood Search (LNS).

Ma et al. [[104](#_heading=h.thw4kt)] introduce Graph Pointer Nets (GPNs) to resolve TSPs, combining a graph neural network (GNN) encoder with Pointer Networks to leverage the strengths of both architectures. This hybrid approach is extended to tackle Time-Dependent TSPs (TSPTW) through Hierarchical Reinforcement Learning, where each neural network layer is tuned to address TSP constraints specifically. The integration of GPNs with local search methods yields results on par with traditional construction heuristics, demonstrating efficacy on both real and synthetic datasets of TSPTW.

Zhang et al. [[105](#_heading=h.3dhjn8m)] explore a nuanced variant of TSP, focusing on minimizing rejection and distance costs in TSPs with Time Windows and Rejections (TSPTWR). The study employs an Attention Model (AM) to solve conventional TSP, followed by a heuristic evaluation to ensure feasibility concerning Time Windows and to assess order rejections. This mechanism aids in accurately calculating the rewards function for TSPTWR, enabling effective updates to neural network parameters based on baseline rollout strategies.

A recent advancement by Ma et al. [[106](#_heading=h.1smtxgf)] refines their model from Ma et al. [10[0,](#_heading=h.302dr9l) p. 11096-11106] by integrating it with Neighborhood Search (NS) to more adeptly solve the Pickup and Delivery Traveling Salesman Problem (PDTSP). They implement Neural Neighborhood Search to swiftly manage precedence constraints, allowing simultaneous operations on pairs of pickup and delivery nodes via specialized removal and reinsertion decoders within the neighborhood search framework.

The integration of learning methodologies with prescriptive approaches is also gaining traction. Lin et al. [[107](#_heading=h.4cmhg48)], for instance, addresses the management of a large fleet of homogeneous vehicles for online ride-sharing services. The authors propose using Contextual Deep Q-Networks (DQN) and Contextual Actor-Critic networks to develop shared state-value functions across all vehicle agents, which are then utilized in a linear programming framework to optimize vehicle allocation efficiently.

*Learning Methods for Facilitating non-learning methods*Furthermore, the combination of learning methods with Monte Carlo Tree Search (MCTS) represents a frontier in solving large-scale routing problems. [[108](#_heading=h.1xrdshw)] illustrates this by training a graph convolutional residual network with an attention mechanism on smaller TSP instances using solutions from the Lin-Kernighan Heuristic (LKH) as benchmarks. This model is then applied to larger graphs, divided into smaller subgraphs, to generate heat maps indicating connection probabilities. These heatmaps are aggregated and employed in an MCTS framework to enhance the final solution, showing the model’s capability to handle TSPs up to 10,000 nodes efficiently.

These hybrid methodologies, as summarized in table 1.3, showcase a diverse array of learning techniques – ranging from supervised learning and Q-learning to policy gradient methods – combined with various traditional approaches like Mixed Integer Programming (MIP), heuristics, and metaheuristics. This integrative approach not only broadens the applicability of learning methods but also enhances the effectiveness and scalability of solutions for complex routing challenges.

Table 1.3 – The summary of studies with hybrid methods.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Problems | Type | Learning Method | Non-learning method |
| 1 | 2 | 3 | 4 | 5 |
| Dorigo and Gambardella | TSP | First | Q-Learning | Ant-colony |
| Liu and Zeng | TSP | First | Q-learning | Genetic Algorithm |
| Alipour et al. | TSP | First | MARL | Genetic Algorithm, 2-opt |
| Hottung and Tierney | CVRP, SDVRP | First | Policy-based RL | NS |
| Syed et al. | VRPTW in ride-hailing services | First | SL | Large Neighborhood Search |
| Fernandes et al. | VRPTW | First | Q-learning | Adaptive Local Search |
| Cappart et al. | TSPTW | First | DQN, PPO | CP, DP |
| Kool et al. | TSP, CVRP, TSPTW | First | SL | DP |
| Xu et al. | TSP | First | Unsupervised Learning | DP |
| Delarue et al. | CVRP | First | Value-based RL | Mixed Integer Programming (MIP) |
| Ma et al. | CVRP | First | The actor-critic variant of PPO | 2-opt, swap and insert |
| Chen and Tian | CVRP | First | Actor-Critic | Halide rewriter |
| Li et al. | Large-scale CVRP | First | SL | MIP solver |
| Deudon et al. | TSP | Second | MHA-based encoder, Transfor mers Decoder | 2-opt local search |
| Zhao et al. | CVRP and VRPTW in large scale | Second | Policy Gradient | OR-tools and LNS |
| Ma et al. | TSPTW | Second | Hierarchical Policy based RL | Local search |
| Zhang et al. | TSPTWR | Second | Policy based RL | Tabu search |
| Lin et al. | Large-scale fleet management problem | Second | Contextual DQN and Contextual Actor-Critic | Linear programming |
| Chen et al. | SDDPVD | First | Deep Q-learning | A policy function approximation |
| Tyasnurita et al. | Open VRP | First | Hyper-heuristic classification | Modified Choice Function All Moves |
| Xing and Tu | TSP | Second | SL to learn heat maps | Monte Carlo Tree Search |
| Fu et al. | TSP | Second | SL to learn heat maps | Monte Carlo Tree Search |
| Hottung et al. | TSP, CVRP | Second | SL to learn latent space | Unconstrained continuous optimization |
| Gutierrez-Rodriguez et al. | mVRPTW | Second | SL | HMOEA-06,  MOGA-06 MMOEAD-15 |
| Continuation of table 1.3 | | | | |
| 1 | 2 | 3 | 4 | 5 |
| da Costa et al. | TSP | Second | DRL, Policy Gradient | 2-opt local search |
| Fu et al. | TSP | First | MCTS with RL | 2-opt local search |
| Wu et al. | TSP, CVRP | First | Policy gradient | Pairwise local operators |
| Xin et al. | TSP, CVRP, CVRPTW | First | SL | LKH Algorithm |
| Yang et al. | TSP | First | Deep Q-Learning | DP  HMPSO-16 |
| Note – Compiled from sources [60, p. 465-479; 91, p. 6995-7000; 92, p. 2935-2950; 93, p. 178-200; 98, p. 1995658; 101, p. 5057-5068; 102, p. 170-180; 103, p. 7208-7217; 104; 105, p. 1-7; 106; 108; 109-129] | | | | |

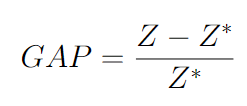
## 1.4 Computational studies

In this section, we delve into the experimental evaluations conducted to assess the performance of various models designed to tackle the Traveling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP) using a set of randomly generated data.

*Data Generation:*For our comparative analysis, we utilized a benchmark dataset introduced by Kool et al. [[56](#_heading=h.3s49zyc)], featuring coordinates for customer locations randomly distributed across a unit square. For the Capacitated Vehicle Routing Problem (CVRP), customer demands at each node were determined by randomly selecting integers between 1 and 9. Vehicle capacities were set based on the size of the graph: 30 for 20 nodes, 40 for 50 nodes, and 50 for 100 nodes.

*Baselines:* Our evaluation includes a range of end-to-end learning methods such as those documented in Kool et al., Kim et al., Kwon et al., Xin et al. [[56](#_heading=h.3s49zyc); 66, p. 12042-12048; [67](#_heading=h.2y3w247), p. 10418-10429; [68](#_heading=h.1d96cc0), 21188-21197] and hybrid approaches Ma et al., Wu et al., Da et al., Hottung et al. [[101](#_heading=h.302dr9l), p. 5057-5068; [102](#_heading=h.1f7o1he), p. 170-180; 108; [130](#_heading=h.2olpkfy)]. These were assessed against the classic LKH3 algorithm, utilizing their respective pre-trained models and implementations available on public GitHub repositories. This study particularly emphasizes the original configurations of these models, even though enhancements like the model-agnostic refiner proposed in Li et al. [[78](#_heading=h.243i4a2), p. 1-12] show potential.

*Experimental Setup:* The experiments were conducted on a high-performance computing setup featuring a single RTX2080 GPU and an Intel Core i9-9900K CPU with 16 cores, running on Ubuntu 20.04. We maintained a batch size of one, accommodating both greedy and sampling decoding strategies, provided the models’ codes support this configuration.

*Results:* The outcomes of these experiments are summarized in table [1.4](#_heading=h.3j2qqm3), which compares the average costs for solving 1,000 instances of CVRP. The discrepancy between the methods is calculated using the formula:

(1.4)

where *Z*∗ is the cost achieved by the LKH3 algorithm, and *Z* is the cost incurred by each tested method.

Table 1.4 – The performance comparison to solve CVRP on random instances

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Method | CVRP20 | | | CVRP50 | | | CVRP100 | | |
| cost | gap, % | time | cost | gap, % | time | cost | gap, % | time |
| LKH3 | 6.16 | 0.00 | 2h18m | 10.46 | 0.00 | 9h10m | 15.68 | 0.00 | 17h27m |
| Kool et al., AM  {M: 1280} | 6.28 | 1.93 | 26s | 10.71 | 2.42 | 1m22s | 16.24 | 3.62 | 3m40s |
| Kool et al., AM  {M: 2560} | 6.28 | 1.83 | 40s | 10.7 | 2.29 | 1m56s | 16.21 | 3.44 | 5m39s |
| Kool et al., AM  {M: 7500} | 6.27 | 1.67 | 1m3s | 10.68 | 2.09 | 4m4s | 16.17 | 3.18 | 14m8s |
| Kwon et al. | –1 | –1 | –1 | –1 | –1 | –1 | 15.87 | 1.27 | 10s |
| Kwon et al. x8 augment | –1 | –1 | –1 | –1 | –1 | –1 | 15.87 | 1.27 | 4hr23m |
| Ma et al. (T=1k) | 6.57 | 6.35 | 31s2 | 12.82 | 22.46 | 1m23s2 | 16.3 | 5.36 | 4m8s2 |
| Ma et al. (T=5k) | 6.28 | 1.94 | 2m33s2 | 12.02 | 14.78 | 4m2 | 15.97 | 3.25 | 20m43s2 |
| Ma et al. (T=10k) | 6.19 | 0.65 | 5m12s2 | 11.73 | 12.01 | 14m3s2 | 15.94 | 1.70 | 49m2 |
| Ma et al. x6 augment | 6.07 | -1.19 | 18m13s2 | 11.38 | 8.70 | 1hr8m2 | 15.79 | 0.70 | 4hr2 |
| Wu et al. (T=1,000) | –1 | –1 | –1 | 10.71 | 3.83 | 8m4s2 | 16.27 | 5.16 | 2hr2m2 |
| Wu et al. (T=3,000) | –1 | –1 | –1 | 10.65 | 3.25 | 8m20s2 | 16.1 | 4.06 | 2hr2m2 |
| Wu et al. (T=5,000) | –1 | –1 | –1 | 10.62 | 2.97 | 8m12s2 | 16.03 | 3.58 | 3hr36m2 |
| Hottung et al. | 6.16 | 0 | 4hr | 10.5 | 0.41 | 9hr23m | 15.89 | 1.4 | 28hr2 |
| Notes  1. Compiled from sources [56; 68, p. 21188-21197; 101, p. 5057-5068; 102, p. 170-180; 121, p. 470-480].  2. 1 – the pretrained models are not shared in the corresponding github repositories.  3. 2 – the batch size is set to 1,000 to record running times | | | | | | | | | |

We report both the average gaps and the total computational time for all test instances. Notably, while the hybrid model by [[101](#_heading=h.302dr9l), p. 5057-5068] equipped with data augmentation slightly outshines other models in terms of performance, it does demand considerable computational time. The CVRP remains a formidable challenge, demonstrating notable performance gaps between learning-based methods and traditional optimization algorithms such as LKH3. However [[68](#_heading=h.1d96cc0), p. 21188-21197] stands out by solving more significant instances with a minimal average gap of 1.27% in considerably less time. We also examined solutions for the Vehicle Routing Problem with Time Windows (VRPTW). Given the unique settings of each study, direct comparisons are challenging. Thus, in Table 1, we juxtapose the findings from [75] and [56] using data from the well-known Solomon dataset [15, p. 254-264], as reported by [75]. Furthermore, table [2](#_heading=h.3fg1ce0) showcases the models’ generalization capabilities on the renowned CVRPLib dataset, with OR-Tools [[131](#_heading=h.13qzunr)] serving as the benchmark for optimization methods. The comparative data is drawn from corresponding studies, with [[105](#_heading=h.thw4kt), p. 1-7] displaying superior performance overall, outdoing both OR-Tools and other learning-driven approaches.

This comprehensive evaluation not only highlights the strengths and weaknesses of various approaches but also underscores the ongoing need for innovative solutions in the realm of vehicle routing, especially as models continue to evolve and are pushed to solve increasingly complex scenarios.

**2 METHODOLOGY**

## 2.1 MDP Formulation of VRPD

A Markov Decision Process (MDP) is characterized by a tuple comprising several elements. The set of states is represented by **s**, and the set of actions is represented by *a.* The deterministic transition probabilities, which define the likelihood of transitioning from one state to another given a specific action, are represented by *P*. The reward function, which assigns a reward to each state-action pair, is denoted by *R*. Finally, the time horizon, indicating the finite number of decision-making stages, is represented by *T*.

The detailed components of this MDP are outlined below.

For the Vehicle Routing Problem with Drones (VRPD), the state of the Markov Decision Process (MDP) includes a set of trucks and drones, represented by a state vector at each time step ***t***. The state vector, denoted as **s*t***, consists of several components: the subset of customers already served up to time step *t* denoted as V*t*, the current destination nodes for the trucks c*ttr*, the current destination nodes for the drones c*tdr*, the remaining capacities of the trucks r*ttr*, the remaining capacities of the drones r*tdr*, the energy levels of the drones *εtdr*, and the locationsof the drones l*tdr*.

Specifically, V*t* represents the subset of customers already served up to time step *t*, and it is denoted as the subset of N, which is the set of all customers.

The variables *ck* and *ck,d* indicate the current destination nodes for truck *k* and itsdrone *d*, respectively. At any given time step *t*, not all vehicles may have reached their destinations.

If a truck or drone is traveling between nodes at step *t*, the current destination node for truck *k* (or its drone *d*) is set to the destination node. When a truck or drone arrives at a node, this current destination node is updated to reflect the node index. Consequently, we construct vectors to represent the current destination nodes for all trucks and drones. The vector for trucks includes the current destination nodes for all trucks, and the vector for drones includes the current destination nodes for all drones associated with each truck.

The remaining travel times to these destinations for trucks and drones are represented by separate variables for each truck and its drone. For trucks or drones that are not in transit, these remaining travel times are set to zero. We define vectors to represent these remaining travel times for all trucks and all drones.

The vectors representing the remaining battery life and load of drones at step *t* are denoted as the battery life vector and the load vector, respectively. The battery life vector includes the remaining battery life for all drones associated with each truck, and the load vector includes the load for all drones associated with each truck. The maximum battery level for drones is denoted as the maximum battery life.

Initially, at step *t* = 0, all vehicles are located at the depot. This gives us the initial conditions where the set of customers already served is the entire set of nodes, the current destination nodes for all trucks and drones are set to the depot, the remaining travel times for all trucks and drones are zero, the remaining battery life for all drones is at its maximum level, and the load for all drones is set to one. These initial conditions apply to all trucks and all drones.

At each time step, the central controller decides the next destinations for the trucks and drones. The decisions for the next destinations for all trucks are represented by a vector, and similarly, the decisions for the next destinations for all drones are represented by another vector. If a drone of a truck has not yet reached its assigned node (i.e., its remaining travel time is greater than zero), then its next destination remains the same as its current destination. Similarly, if a truck is still in transit, its next destination remains the same as its current destination. The combined action at each time step includes the decisions for both the trucks and the drones.

Given the deterministic nature of the problem, state transition probabilities are deterministic. We update the remaining travel times for each vehicle after its action is taken. The updated remaining travel time for each truck is calculated by adding the travel time from its current location to its next destination to its current remaining travel time. Similarly, the updated remaining travel time for each drone is calculated by adding the travel time from its current location to its next destination to its current remaining travel time. The travel time between the same locations is zero for both trucks and drones.

The time step advances when at least one vehicle reaches its destination. The time to complete the previous step is the minimum remaining travel time among all trucks and drones. The remaining travel times at the new step are then updated by subtracting this minimum time from the updated remaining travel times of all trucks and drones.

The drone’s battery level is updated as follows: If the truck and its drone are at the same location and both have zero remaining travel time, the drone’s battery level is reset to its maximum value. Otherwise, the drone’s battery level is decreased by an amount proportional to the time taken to complete the previous step.

If a drone arrives at a node with a demand at a particular time step, the load of the drone for the next time step is updated by reducing it by the demand at that node, ensuring it does not go below zero.

The demand at each node is then updated. For all trucks and drones, if a drone arrives at a node and its remaining travel time is zero, the demand at the node is reduced by the load of the drone, ensuring it does not go below zero. If a truck arrives at a node and its remaining travel time is zero, the demand at the node is reduced by one, which is assumed to be the maximum demand that can be fulfilled by a truck at each node.

The set of visited nodes is updated by including all nodes that were the next destination for any vehicle (truck or drone) and have zero remaining travel time at the new time step.

Additionally, the current destination for each vehicle at the next time step is updated to be the action taken at the previous time step. Let *T* be the index of the step when all drones and trucks return to the depot after serving all customers. The cost function, which represents the negative reward, is the total time spent in the system. This total time, also known as the makespan, is the sum of the time spent at each time step from the start until all vehicles return to the depot.

## 2.2 Solution Method

In this section, we will delve into the details of a sophisticated deep-learning model that has been developed to facilitate the efficient coordination of multiple drones and trucks. The model is designed to optimize the routing process while ensuring the safety of all vehicles involved, making it a useful tool for any organization that operates a fleet of vehicles.

The model employs a combination of advanced algorithms and techniques to ensure that it can handle complex routing scenarios with ease. For instance, it uses a combination of reinforcement learning, neural networks, and other state-of-the-art techniques to learn from real-world data and make intelligent routing decisions.

To ensure that the model is capable of handling a wide range of real-world scenarios with a high degree of accuracy and reliability, it was trained using a sophisticated training algorithm. The training algorithm is designed to fine-tune the model by adjusting its parameters and hyperparameters to optimize its performance on a given task.

Overall, this deep learning model has the potential to revolutionize the way in which fleets of vehicles are coordinated, making it possible to optimize routing and ensure safety while minimizing costs and maximizing efficiency.

### 2.1.1 The Deep Learning Model

The efficient routing of drones and trucks is crucial for successful last-mile delivery. In order to achieve this, we need to develop a stochastic policy known as *πθ*. This policy is designed to enable the optimal routing of delivery vehicles by considering a range of factors, including traffic patterns, delivery locations, and weather conditions. The *πθ* policy is defined as a product of conditional probabilities, each of which is associated with a specific set of parameters *θ*. By learning this policy, we can ensure that delivery vehicles reach their destination in the most efficient and cost-effective manner possible. The policy is represented as the product of the probabilities of taking each action at each time step given the state at that time.

In recent years, research in the domain of routing problems, particularly those involving autonomous vehicles like drones and trucks, has been significantly enhanced through the use of encoder-decoder-based deep learning models. These models have shown to be adept at learning a routing policy for a single vehicle, as demonstrated in pivotal studies conducted by Vinyals et al., Kool et al. and Bogyrbayeva et al. [56; [131](#_heading=h.3nqndbk); 132].

The framework of the encoder-decoder-based deep learning models typically consists of two parts - an encoder and a decoder. The encoder is responsible for extracting and encoding essential information from a graph, delineating its structure and the specifics of the problem at hand. This encoded information is then passed on to the decoder.

The decoder uses the embeddings from the encoder to learn the routing policy *πθ* directly. The embeddings are a compressed representation of the original graph that captures the most relevant information required for the routing policy. The decoder then uses this information to learn the routing policy, which determines the optimal path for the autonomous vehicle to follow.

Overall, these encoder-decoder-based deep learning models have proven to be a valuable tool in the field of routing problems, particularly with regards to autonomous vehicles like drones and trucks. By extracting and encoding essential information from a graph, these models are able to learn a routing policy that can help optimize the path taken by single vehicles.

The research conducted by Bogyrbayeva et al. [[133](#_heading=h.i17xr6)] presented a new hybrid model called the Hybrid Model (HM), which combines an attention-based encoder with an LSTM-based decoder. This model is designed specifically to address the routing problem, which involves determining the best path for a vehicle or group of vehicles to take from one point to another. \

The HM is a deep learning model that is capable of processing large amounts of data and making use of both short-term and long-term memory. It has proven to be effective in handling the routing of a single drone and truck, making it especially useful in simple routing scenarios.

The attention-based encoder in the HM allows the model to focus on specific parts of the input data that are most relevant to the routing problem, while the LSTM-based decoder is responsible for generating the output sequence that represents the optimal route. This unique combination of features makes the HM a powerful tool for tackling the routing problem in a wide range of scenarios.

The current model for routing multiple drones and trucks does not take into account the interactions between groups. Each truck and its set of drones can be viewed as a group, and it is crucial for the model to coordinate the actions taken by both the truck and its drones to ensure efficient launching and retrieval operations. This coordination is particularly important because the decision-making process within one group inherently impacts the choices of another group. Drones are exclusively launched and retrieved by their assigned trucks, so the actions of one group can impact the routing decisions of another group. To ensure that all customers, who are dispersed across various locations, are served efficiently and in minimal time, the model must account for the actions chosen by one group when making decisions about the routing of another group. Therefore, a more detailed and comprehensive approach is needed to account for these group interactions, including the coordination of actions and the impact on routing decisions.

The current routing models have proven to be quite useful in developing routing policies for individual vehicles, however, they come with a crucial limitation. These models are not capable of handling situations where multiple vehicles need to operate in a coordinated fashion across different groups. In other words, they cannot take into account the dynamic interactions between multiple vehicle groups.

To address this limitation, a more advanced modeling approach is required that not only considers individual vehicle routes but also takes into account the dynamic interactions between multiple vehicle groups. Such an advanced approach will be crucial for optimizing routing strategies and achieving higher efficiency in complex multi-vehicle environments. Moreover, this approach will enable the development of routing solutions that are more robust and can handle complex and unpredictable situations that may arise during multi-vehicle operations.

Therefore, it is of utmost importance to continue researching and developing new modeling approaches that can address these limitations and enable the development of more efficient and effective routing strategies for multi-vehicle environments.

We have built upon the research done by Bogyrbayeva et al. [[133](#_heading=h.i17xr6), p. 103981] in the field of routing multiple groups of drones and trucks. We have identified a need to enhance the Hybrid Model (HM) to better handle the complex dynamics involved in coordinating multiple vehicle groups. While the original HM is efficient at managing individual routes, it lacks the capability to handle interactions and coordination between multiple vehicle groups. To address this limitation, we have made strategic modifications to the inputs fed into the decoder of the HM. These adjustments enable the model to effectively incorporate and manage both the coordination within a vehicle group and the interactions between different vehicle groups within a shared operational graph.

We have made significant strides in improving our latest model, which we have named the attention-encoder group-based LSTM-decoder. This name reflects the model’s enhanced focus on group dynamics and attention-driven processes, which are critical in managing complex systems.

To achieve this, we have incorporated an attention-based encoder that meticulously processes the entire graph. This allows the model to capture complex dependencies and interactions between different nodes and vehicle groups, resulting in a highly detailed representation of the system. By doing this, the model can make informed routing decisions that take into account the system’s complexity.

The LSTM-based decoder component of the model has also been improved with augmented inputs that include information about group interactions. This means that the model can consider both intra-group coordination and inter-group dynamics, which are critical in managing complex systems. These enhancements ensure that the model is more accurate, efficient, and effective overall.

Overall, the attention-encoder group-based LSTM-decoder model is a significant improvement over previous models. It is more detailed, more accurate, and more effective in managing complex systems than its predecessors. It is a testament to our commitment to continuous innovation, improvement, and development of cutting-edge technologies.

Figure [2.1](#_heading=h.1pxezwc) provides a comprehensive overview of this revised model. This visual representation helps elucidate the operational framework of the attention-encoder group-based LSTM-decoder, showcasing how it integrates and processes information to optimize routing decisions across multiple groups. The figure illustrates the flow of data through the model, highlighting how the modified inputs enhance the model’s ability to synthesize and respond to the dynamic routing environment effectively. This visual tool is invaluable for understanding the complex mechanisms at play and the strategic enhancements that have been implemented to improve the model’s performance in multi-vehicle routing scenarios.

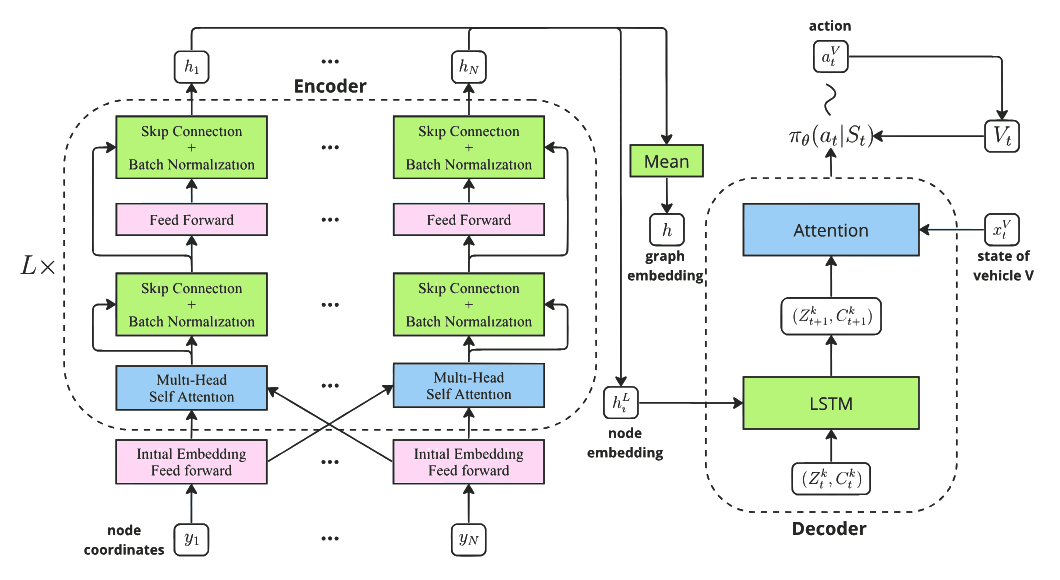


Figure 2.1 – An overview of the attention-encoder group-based LSTM-decoder model

*Encoder.*Our implementation leverages the efficacy of a multi-head attention-based encoder, which has been demonstrated to be highly effective in embedding graphs for solving routing problems. This encoder is capable of capturing complex patterns and dependencies within the graph structure, allowing for more accurate and efficient routing solutions. The effectiveness of this approach has been demonstrated in previous works, including studies by Kwon et al. (2020) and Kool et al. (2019), which provide strong evidence of the potential of multi-head attention-based encoders for graph embedding tasks. By utilizing this state-of-the-art technique, our implementation is able to deliver accurate and efficient routing solutions with a high degree of reliability and scalability.

Let’s consider a graph with nodes and corresponding coordinates for each node. The coordinates of each node are represented in a two-dimensional space. To embed a fully connected graph, we start with an initial transformation of node embeddings. The initial embeddings are generated using the coordinates of the nodes, which are transformed through a linear transformation. This transformation involves learnable parameters, which are updated during the training process to generate an optimal solution.

After we obtain the initial node embeddings represented as *h*0:*n ∈ N*, we make use of *L* attention layers. Each of these attention layers consists of two sublayers, namely, a multi-head attention (MHA) layer and a fully connected feed-forward (FF) layer. The MHA layer is responsible for taking input from the previous layer and supporting message passing between nodes. Before inputting to the MHA layer, we compute queries *q*, keys *k*, and values **v** for each input by projecting the input *hn* with the help of trainable parameters *WQ*, *WK*, and *WV*, respectively.

In order to determine the compatibility between nodes *i* and *j*, we make use of the fully connected graph structure. We express compatibility as a function of the parameter size, meaning that the larger the parameter size, the greater the compatibility between the nodes. Once we have calculated these compatibility values, we use them to compute attention weights by applying the softmax function, resulting in values between 0 and 1.

In our approach, we utilize a multi-head attention (MHA) layer with head size *M* to improve the attention mechanism’s expressive power. This layer enables us to compute and combine messages from each head for each node *n* in the network. The output of the MHA sublayer, along with the skip connections, undergoes batch normalization to reduce internal covariate shift and improve the training speed. This process achieves better accuracy and stability during the training phase, making our model more robust and efficient.

Our implementation for routing problems uses a multi-head attention-based encoder that effectively embeds graphs. The encoder architecture consists of multiple attention layers, which allow for message passing between nodes. We use batch normalization to produce a batch-normalized output after passing the input through the attention layers.

The resulting batch-normalized output is then passed through a fully connected feed-forward (FF) network, where the ReLU activation function is applied. The fully connected FF network enhances the model’s effectiveness and training speed. To further refine the output of the FF layer, we apply skip connections and batch normalization. The final representation of the output is obtained after *l* attention layers, which we denote as *hl*.

Overall, our approach provides an efficient and effective method for routing on graphs. The attention mechanism facilitates message passing between nodes, while the fully connected FF network with skip connections and batch normalization further enhances the model’s performance. Our implementation has the potential to improve routing efficiency and performance on graph-structured data.

*Decoder.*In order to represent embeddings for every node in a graph, a process is used that involves utilizing encoder outputs. This process is especially useful when we want to pass information from a graph as input to a machine learning model. By encoding the graph’s information into a vector format, we can more easily pass it to a machine learning model. The resulting encoding is represented as , which is the mean of all node embeddings in the graph.

The process of encoding involves taking a graph and representing it in a vector format. Each node in the graph is represented by an embedding, which is essentially a vector that represents the node’s features. To create these embeddings, the encoder takes the features of each node as input and outputs a vector that represents that node’s features. Once we have the embeddings for each node, we can average them to get an overall representation of the graph’s features. This is what is represented as .

To solve a transportation problem, we first group the trucks into different sets based on the problem’s specific requirements. The number of groups or sets, represented as *k*, determines the number of groups we have. For each group, we initialize an LSTM (Long Short-Term Memory) consisting of a tuple (z*tk,* c*tk*) forall *k* ∈ K. The LSTM is a type of neural network that can maintain information over time and is useful for predicting the next value in a time series or sequence of data. The LSTM’s architecture allows it to selectively remember or forget previous inputs based on the current input, making it an ideal tool for solving complex transportation problems. By using an LSTM, we can predict the optimal routes for each truck group, which helps to minimize the total transportation cost and increase efficiency.

To determine an appropriate action for a decision-making unit, such as a drone or truck in group *k*, at a particular time *t*, we make use of an LSTM model. The model is fed with the embedding of the last node that was selected by the most recent decision-maker in the group. This embedding is represented as *hiL*, where *i* corresponds to the node. By providing the LSTM with this information, we enable it to utilize its memory to predict the next best course of action for the decision-maker. This approach helps to optimize decision-making and ensure that the right actions are taken at the right time, based on previous actions and outcomes. The Long Short-Term Memory (LSTM) network updates the state and cell values for the next time step using the current hidden state and cell values along with the previous state and cell values.

During the process of sequence generation in a recurrent neural network, when we move to the decoder, we apply a technique called dropout to the hidden state z*kt+1* and cell state c*t+1k* at time *t* + 1. Dropout is a regularization technique that randomly sets a fraction of the output values to zero during training. This technique helps to prevent overfitting and improve the generalization performance of the model.

To be more specific, during training, each element of z*kt+1* and c*kt+1* is set to zero with a probability of *p*. This means that some elements of z*kt+1*and c*kt+1* are randomly dropped out, and the model learns to be more robust and less dependent on specific features.

The purpose of dropout is to force the network to learn more robust features that are useful in making predictions. By randomly dropping out some of the elements, the network learns to rely on a more diverse set of features, making it less likely to overfit to the training data. Dropout also helps to improve the generalization performance of the model by reducing the impact of individual neurons.

In summary, dropout is an important technique for preventing overfitting and improving the generalization performance of recurrent neural networks during sequence generation. By randomly dropping out some of the elements of the hidden and cell states, we can make the model more robust and less dependent on specific features. This is achieved by applying a dropout function to the hidden state and cell state at the next time step with a certain probability.

In our system, we use a set of binary values to represent the decision-maker’s state at a specific time. Specifically, we create two binary values, *x*1*t* and *x*2*t*, which indicate whether the decision-maker is currently in transit and whether they are carrying a load, respectively. These values are essential for determining the optimal routing and scheduling of drones in the system.

In addition to these binary values, we also calculate the current battery level of a drone. To do this, we normalize the drone’s current battery level with respect to the maximum battery level, *εmax*. We denote this normalized battery level as *x*3, which provides us with valuable information for determining the feasibility of drone delivery operations and ensuring that drones can complete their assigned tasks within their battery capacity.

Finally, we combine these three values, *x*1*t*, *x*2*t*, and *x*3*t*, into a vector called *vecxt*. By concatenating these values, we can represent the decision-maker’s current state concisely and meaningfully, which is essential for making informed decisions about the routing and scheduling of drones in the system.

The attention mechanism is an essential component of many machine learning models. It is used to compute the importance of each element in the input sequence, enabling the model to focus on the most relevant information.

The attention mechanism takes three inputs: the current state of the decision-maker represented by x*t*, the embedding of the graph represented by , and the hidden state of the LSTM represented by z*kt+1*. These inputs are then processed using trainable parameters *va*, *W a*, and *W x* to compute the attention vector.

The attention vector is represented as a matrix *ai,j*, where each element *ai,j* denotes the relevance of the *j*th input at time step *i*. The attention vector is then used to weigh the input sequence, allowing the model to focus on the most important information.

By using the attention mechanism, the model can efficiently process large amounts of data and make more informed decisions. The attention mechanism has been used in various applications, including natural language processing, computer vision, and speech recognition. Its effectiveness has made it a popular tool for many machine learning practitioners. The attention vector is calculated by taking the dot product of a learnable parameter vector with the tanh activation applied to a combination of the context vector, the updated hidden state, and the transformed input vector.

The attention vector is a vector of weights that determines the importance of different elements in a given context. In many machine learning algorithms, the attention vector is used to decide which nodes should be visited next. The attention vector is typically created by a neural network, and it assigns a weight to each node based on how relevant it is to the current context.

Once the attention vector is formed, the softmax function is typically applied to it. The softmax function is a mathematical function that normalizes the vector and produces a probability distribution. This distribution indicates the likelihood of visiting each node, with higher probabilities indicating that a node is more likely to be visited.

However, if a node has already been visited by any vehicle before, its probability value will be set to zero. This is because the algorithm needs to ensure that each node is visited only once to optimize efficiency. As a result, the probability distribution is updated based on the remaining unvisited nodes, and the algorithm selects the node with the highest probability value as the next node to visit.

Overall, the attention vector and softmax function approach is a powerful technique for optimizing efficiency in machine learning algorithms. By prioritizing the most relevant nodes, it allows algorithms to make more informed decisions and achieve better outcomes. The probability of selecting a node at a specific time step is calculated by taking the exponential of the attention value for that node and normalizing it over the sum of the exponentials of the attention values for all unvisited nodes. If a node has already been visited, its probability is set to zero.

### 2.2.2 The Training Algorithm

In order to efficiently train the attention-encoder group-based LSTM-decoder model, it is necessary to explore different actions and receive feedback as rewards. The model is designed to determine the probability distribution, denoted as *πθ*(V*T* |G), which generates the sequence of nodes that the drones and trucks should visit in the Vehicle Routing Problem with Drones (VRPD) given a graph. This is achieved by using an attention mechanism in the encoder to assign weights to the nodes based on their importance and a group-based decoder that generates a sequence of actions for the drones and trucks to follow.

The objective during the training phase is to minimize the makespan, which is the total time taken to complete all the deliveries. This is done by minimizing the cost function, which takes into account the distance traveled by the drones and trucks, as well as the time taken to complete each delivery. By minimizing the makespan, the model can generate an optimal sequence of actions that will minimize the time and distance required to complete all the deliveries.

The training objective function is defined to optimize the performance of the model. It is denoted as *J*(*θ*|G) and is calculated as the expected value of the difference between the total cost and a baseline function, multiplied by the logarithm of the policy probability over the sequence of actions. This expectation is taken with respect to the policy.

In reinforcement learning, it is common to use a baseline to reduce the variance of the learning process. In this context, *b*(G) serves as a baseline, which is estimated by a critic network. The policy *πθ* is learned by an actor-network, and the REINFORCE algorithm [[134](#_heading=h.320vgez)] is used to compute the gradients of the objective function, ∇*θJ*(*θ*|G).

The training process of the actor and critic networks is outlined in Algorithm [1](#_heading=h.147n2zr) (figure 2.2). This process involves sampling a batch of data from a predefined distribution and selecting actions for all drones and trucks at each time step. The final reward obtained at the end of the episode is used as an empirical mean to train both networks.

To explore the action spaces of drones and trucks efficiently, a masking scheme is deployed that considers the current state of the problem and the routing constraints. This scheme prevents some nodes from being visited, which is similar to what has been described in existing literature [[42](#_heading=h.1302m92), p. 1-20; [56](#_heading=h.3s49zyc)]. This approach allows the agent to efficiently explore the action space, which is essential in this context since the problem of routing drones and trucks is highly complex.

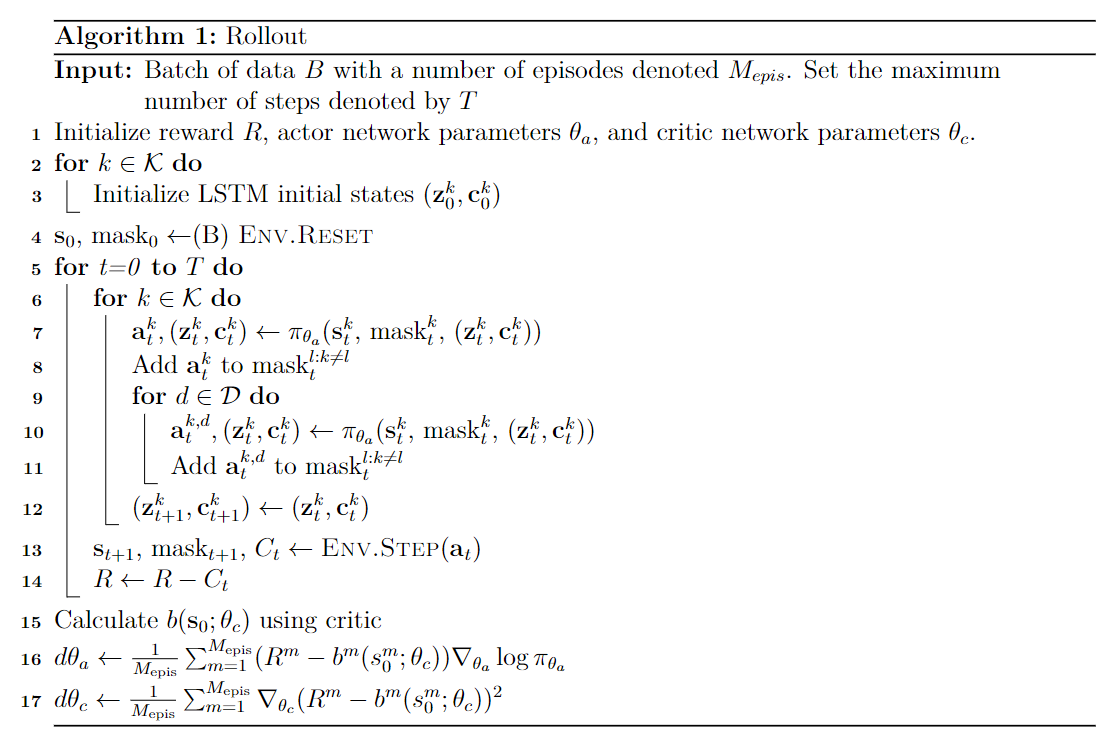


Figure 2.2 – Training algorithm

**3 EXPERIMENTAL RESULTS**

In this section, we focus on computational studies of the proposed model. Figure 3.1 shows training rewards for the VRPD problem on the network with 10 nodes, 2 vehicles, and 2 drones. Figure 3.2 shows training rewards for the VRPD problem on the network with 20 nodes, 2 vehicles, and 1 drone.

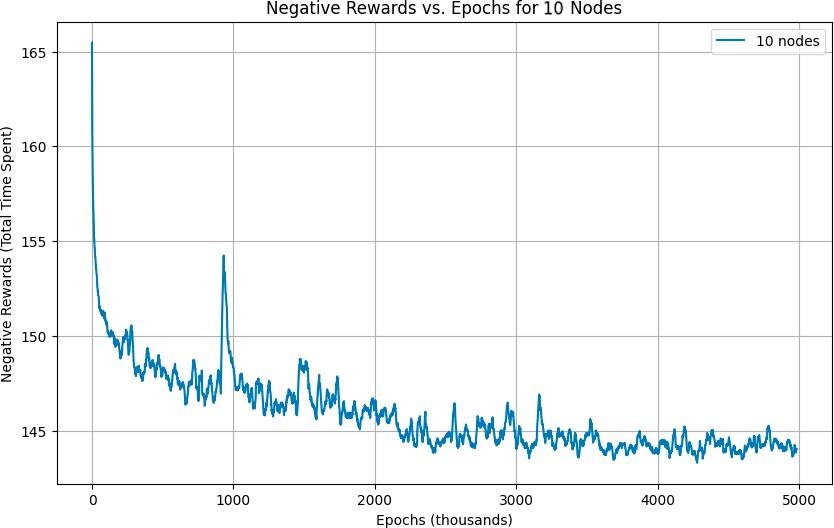


Figure 3.1 – Training rewards for experiments with 10 nodes

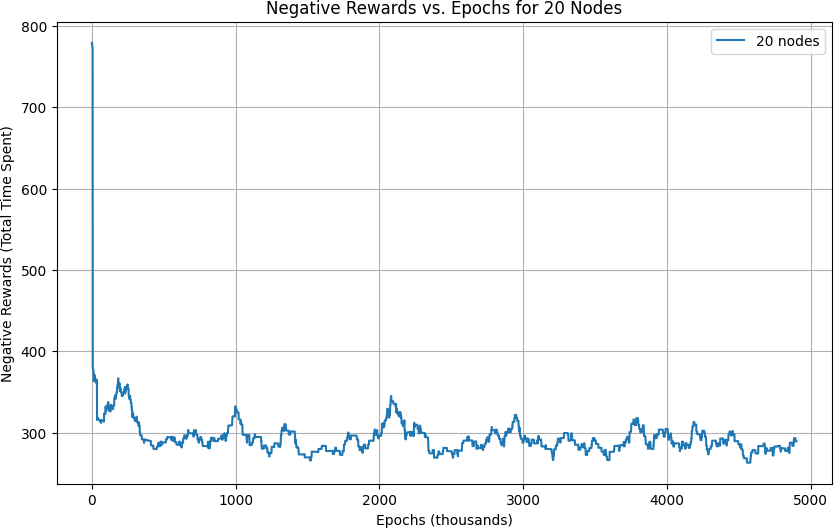


Figure 3.2 – Training rewards for experiments with 20 nodes

The study is based on the following assumptions, which are critical for accurately modeling and simulating the operations of both trucks and drones within the system:

*Drone Operations:*

* each drone can serve exactly one customer per operation, which includes both the delivery and retrieval phases. The drone has a limited endurance of *ε* distance units for each operation. Upon returning to the vehicle, the drone’s battery is recharged instantaneously.

*Customer and Vehicle Characteristics:*

* each customer has an equal demand, and the vehicles used in the model are homogeneous, each carrying a specified number of identical drones.

*Drone Travel Constraints:*

* drones are restricted to visiting only one customer per launch, provided the battery allows it. After serving a customer, a drone can travel to one of the following destinations;
  + an unserved customer who is not expected to be visited by any other trucks or drones;
  + the depot;
  + the expected location of its associated truck;
  + the expected location of another drone that has already served a customer assigned to the same truck;
* if the drone’s battery is insufficient to reach any of these nodes, the node cannot be visited by the drone.

*Truck Travel Constraints:*

* trucks have the flexibility to visit;
  + any unserved customer who is not expected to be visited by other trucks or drones;
  + the depot;
  + the expected locations of its drones that have already served customers;
  + locations where its drones have become immobilized.

*Depot Constraints:*

* once trucks or drones reach the depot, they are not permitted to leave again. This assumption aligns with the guidelines presented in Schermer et al. (2019).

*Drone Idle Condition:*

* if a drone arrives at a node after serving a customer, it must remain at that location until its associated truck arrives.

*Speed Matching:*

* when traveling together, drones will match the speed of their respective trucks.

*Zero-Time Actions:*

* if drones or trucks remain stationary at a node, the time required for this action is considered negligible and is set to zero.

*Operational Constraint:*

* a drone cannot return to the node from which it was launched within the same operation.

These assumptions form the basis of our computational study, ensuring that the model reflects realistic constraints and operational scenarios. The computational experiments were designed to assess the efficiency, reliability, and overall performance of the integrated truck-drone delivery system. By simulating various operational scenarios and analyzing the results, we aim to provide insights into the practical implications of using drones in vehicle routing problems and to identify potential areas for further optimization and improvement.

## 3.1 Training and Evaluation Configurations

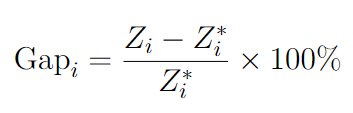
*Data Generation.*To highlight the computational efficiency of our proposed method, we conducted a study using a well-known benchmark dataset that is widely used in the field of vehicle routing and optimization. The dataset, initially introduced in a research article by Agatz et al. [[135](#_heading=h.1h65qms)], consists of both customer and depot locations.

The customer locations in the dataset were randomly generated using a uniform distribution within the [1,100]x[1,100] area, ensuring that the customers are spread across a large geographic region. The depot location was generated with a uniform distribution within the [0,1]x[0,1] range, ensuring that it is always located in a corner.

This dataset has been extensively used in a variety of studies, including those by Bogyrbayeva et al. and Schermer et al. [[136](#_heading=h.415t9al)]. By utilizing this dataset, we were able to compare our approach with other state-of-the-art methods and establish the effectiveness of our algorithm.

Throughout our experiments, we assumed that the drone’s speed is twice that of the truck, i.e., *α* = 2, consistent with previous literature. This allowed us to measure the efficiency of our approach in terms of computational efficiency and time taken to solve the problem. Our results indicate that the proposed algorithm is highly efficient and can solve the problem relatively quickly, making it a promising approach for future research in this field.

*Gap.*To evaluate the quality of solutions produced by different approaches to solve the VRPD, we measure the GAP, defined as:



(3.1)

where *Zi* is the cost of a solution method for each instance *i* and *Z*∗ is the cost of the best-performing solution method among all methods compared, for instance, *i*. In table [1.4](#_heading=h.3j2qqm3) we report the average GAP for each setting of the problem as the mean of GAPs for 10 instances of the problem.

*Baselines.*Our main goal is to assess the effectiveness of our neural architecture, which we designed to solve a complex problem. To achieve this, we are using the research paper by Schermer et al. (2019) [[136](#_heading=h.415t9al), p. 134-157] as a benchmark. In their paper, Schermer et al. presented a mixed-integer linear programming (MILP) formulation of the problem and reported the results of the small instances of the problem with and without lazy constraints. The problem involves finding an optimal delivery route for a fleet of vehicles, subject to certain constraints such as vehicle capacity, time windows, and geographical location.

Schermer et al. used two methods, namely the Solver and Solver Lazy methods for solving the problem, which are known to be efficient in solving the problem exactly, as demonstrated in table [1.4](#_heading=h.3j2qqm3). However, these methods are computationally expensive and may not be practical for large-scale problems.

To address this issue, Schermer et al. proposed a more practical approach that utilizes heuristic methods, namely the Variable Neighborhood Search (VNS) and Tabu Search (TS) methods. The VNS method explores various possibilities by altering the neighborhood structure, while the TS method learns from previous solutions to avoid retracing steps. These heuristic methods are referred to as ”Heuristic” in table [1.4](#_heading=h.3j2qqm3).

We are grateful to the authors of [[136, p. 134-157](#_heading=h.415t9al)] for sharing the computational time and costs for each instance of the problem. We will report the best solution obtained by their heuristic in five runs, along with its corresponding computational time. By comparing the results of our neural architecture with the results obtained using VNS and TS-based heuristics, we can analyze the performance of our architecture more accurately. This comparison will help us determine the efficiency of our neural architecture in solving the problem and whether it outperforms traditional methods. We will also analyze the computational time and costs associated with each method to determine the most efficient approach for solving the problem.

*Decoding Strategies.*This study aimed to explore and compare three different methods for extracting solutions from a trained model. The study was conducted to gain a comprehensive understanding of each method and identify their respective benefits and drawbacks.

The first method, known as the ”greedy” approach, involves selecting nodes with the highest probability of being visited at each time step. This method is simple and efficient, but it does not always guarantee the optimal solution.

The second method, the ”sampling” approach, involves generating multiple solutions independently from the trained model. To obtain multiple candidate solutions, the researchers used a sample size of 4800. The cost of each solution was evaluated by summing the weights of the edges that connect the nodes in the solution. The solution with the lowest cost among the samples was then selected and reported as the result. While this method has a higher probability of finding the optimal solution than the greedy approach, it is computationally more expensive.

The third method, the ”Ensemble” approach, involved using multiple models saved throughout the training process. The researchers selected the models’ weights in the range of [400*k, ...,* 1000*k*] with an increment of 100k to ensure the best possible results. To obtain multiple solutions, solutions were sampled using a sample size of 4800 for each model, and the best results were reported. This approach allowed the researchers to leverage the strengths of multiple models and obtain better solutions. However, it was the most computationally expensive approach among the three.

The study provided a detailed analysis of each method, including their advantages, disadvantages, and computational requirements. The researchers also evaluated the trade-off between computational efficiency and solution quality for each approach. The results of this study can be useful for researchers and practitioners in selecting the most appropriate method for their specific application, taking into account the trade-off between efficiency and solution quality. Overall, this study provided valuable insights into the methods used to extract solutions from a trained model.

*Computational Resources and Hyperparameters.*In our project, we utilized a combination of computer systems to carry out various tasks. Specifically, we used two systems that were equipped with an Intel Core i9 12900K processor and PCI-E 24576Mb Palit RTX 3090 GamingPro OC, GeForce RTX 3090 video card. These systems were highly efficient and allowed us to complete tasks that required high-end graphics processing.

For training purposes, we used another computer system with a Core i9 12900K processor and RTX 4090 24GB. This computer system proved extremely useful in handling the complex computations required for training neural networks.

To conduct testing and report computational time, we used an Intel Core i9 12900K processor and PCI-E 24576Mb Palit RTX 3090 GamingPro OC computer. This computer system was chosen for its reliability and high performance in running the required tests.

The model was trained over an extensive period to ensure its robustness and accuracy. The training process involved running the model for 1 million epochs. Each epoch represents a complete pass through the entire training dataset. Such a high number of epochs was chosen to allow the model to learn complex patterns and relationships within the data.

During training, a batch size of 128 was used. Batch size refers to the number of training examples utilized in one iteration. A moderate batch size of 128 was selected to balance the computational efficiency and the stability of the gradient updates.

Testing instances were incorporated into the training process to monitor the model’s performance and prevent overfitting. A total of 512 instances were used for testing, which occurred every 200 steps. This frequent testing allowed for continuous evaluation and provided valuable feedback on the model’s learning progress.

*Hyperparameters* Hyperparameters play a crucial role in the training of neural networks. The chosen hyperparameters for this model are as follows:

1. *Embedding Dimensions:* The embedding dimensions were set to 3. Embedding layers are used to transform categorical data into continuous vector spaces, which help in capturing the semantic relationships between different categories.
2. *Hidden Dimensions:*The hidden layer had 256 dimensions. Hidden dimensions refer to the size of the vectors used in the hidden layers of the neural network. A larger hidden dimension size allows the model to capture more complex features and patterns.
3. *RNN Layer:*The model incorporated 1 recurrent neural network (RNN) layer. RNNs are particularly useful for sequential data as they maintain a hidden state that captures information from previous time steps.
4. *Forget Bias:*The forget bias was set to 1.0. In the context of RNNs, a forget bias helps in controlling the extent to which the previous information is retained or forgotten. A forget bias of 1.0 ensures that the model retains a substantial amount of past information, which is crucial for understanding long-term dependencies.
5. *Dropout Probability:*A dropout probability of 0.1 was applied to prevent overfitting. Dropout is a regularization technique where a fraction of neurons is randomly set to zero during training, which helps in making the model robust and reducing overfitting.
6. *Gradient Clipping:*Gradients were clipped at a value of 2.0 to avoid exploding gradients. Gradient clipping is a technique used to prevent the gradients from becoming too large during training, which can cause the model to become unstable.
7. *Learning Rates:*The learning rates for both the actor and critic were set to 10−4. The learning rate controls how much to change the model in response to the estimated error each time the model weights are updated. A small learning rate was chosen to ensure gradual learning and to avoid overshooting the optimal solution.

These hyperparameters were carefully chosen based on preliminary experiments and best practices in neural network training. They collectively contributed to the effective training of the VRP model with drones, ensuring that it could learn efficiently from the data while maintaining stability and preventing overfitting. When training our neural networks, we followed the hyperparameters used in [[134](#_heading=h.i17xr6), p. 229-255], which were carefully selected based on our experimentation and testing. Specifically, we set the hidden dimension for embedding as 256, the learning rate as 0.0001, and performed 1 million iterations for training. These parameters were chosen to ensure optimal results and were adjusted based on our testing and analysis.

By meticulously tuning these hyperparameters and incorporating rigorous testing during the training process, the model was able to achieve a high level of performance, making it a valuable tool for solving complex vehicle routing problems with the integration of drones.

## 3.2 Results on RL models comparison

We tested our model with other RL solutions that were developed during this research work. The Hybrid Model (HM), considered the state-of-the-art for VRPD, showed competitive results but was generally outperformed by the RL sampling model in terms of the optimality gap and cost efficiency. The Custom model, which uses custom settings for different problem sizes and incorporates the same augmentation technique proposed by Yining Ma, also provided a strong baseline but did not match the consistency and performance of the RL sampling approach. For the 8-node problems, the RL sampling model performed well across all configurations. Specifically, with 1 vehicle and 2 drones, RL sampling had an average cost of 160.23 and a gap of 5.47%, showing the lowest cost and gap among the tested methods except for the Ensemble RL model, which had slightly better performance.

When we configured 2 vehicles with 1 drone each, RL sampling achieved an average cost of 148.32 and a gap of 0.59%, outperforming most methods except for the Ensemble RL, which showed a 0.00% gap. For the configuration of 2 vehicles with 2 drones each, RL sampling had a cost of 126.87 with a 0.00% gap, matching the best performance of the Ensemble RL model.

For the 10-node problems, the RL sampling model continued to show strong performance. With 1 vehicle and 2 drones, Ensemble RL had an average cost of 177.01 and a gap of 0.11%, outperforming all other methods. In the configuration of 2 vehicles with 1 drone each, Ensemble RL achieved an average cost of 167.56 and a gap of 0.00%, slightly outperforming the RL sampling model. When using 2 vehicles with 2 drones each, RL Ensemble achieved a cost of 134.20 with a 1.31% gap, but RL sampling showed better results with a cost of 132.40 and a 0.00% gap, demonstrating its superior performance in this specific configuration (table 3.1).

Table 3.1 – VRPD results with Infinite Flying Range of Drones for different RL solutions

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Category | Methods | |Kc|=1, |Dc|=2 | | |Kc|=2, |Dc|=1 | | |Kc|=2, |Dc|=2 | |
| costs | gaps% | costs | gaps% | costs | gaps% |
| |Nc| = 8 | HM Greedy | 168.71 | 9.39 | 168.94 | 0.83 | 127.17 | 0.21 |
| HM Sampling | 168.35 | 9.18 | 167.98 | *0.25* | *126.87* | *0.0* |
| Custom Greedy Augmented | 161.27 | 5.97 | 177.61 | 6.0 | – | – |
| Custom Sampling Augmented | 160.98 | 5.81 | 176.74 | 5.48 | – | – |
| RL greedy | 161.69 | 6.23 | 148.68 | 0.9 | 126.98 | 0.09 |
| RL sampling | 160.23 | 5.47 | 148.32 | *0.59* | *126.87* | *0.0* |
| Ensemble RL | *159.95* | *5.31* | *147.6* | *0.0* | *126.87* | *0.0* |
| |Nc| = 10 | HM Greedy | 185.81 | 4.83 | 168.94 | 0.83 | 137.31 | 3.51 |
| HM Sampling | 187.23 | 6.05 | 167.98 | 0.25 | 136.14 | 2.6 |
| Custom Greedy Augmented | 182.09 | 2.8 | 171.14 | 2.14 | 135.69 | 2.47 |
| Custom Sampling Augmented | 181.14 | 2.22 | 168.32 | 0.45 | 135.69 | 2.47 |
| RL greedy | 185.42 | 4.57 | 168.4 | 0.5 | 136.06 | 2.66 |
| RL sampling | 180.26 | 1.72 | 167.57 | 0.01 | *132.4* | *0.0* |
| Ensemble RL | *177.01* | *0.11* | *167.56* | *0.0* | 134.2 | 1.31 |
| |Nc| = 20 | Custom Greedy Augmented | 246.86 | 5.6 | 213.54 | 3.95 | – | – |
| Custom Sampling Augmented | 246.86 | 5.6 | 213.54 | 3.95 | – | – |
| RL greedy | 243.06 | 3.86 | 219.02 | 6.91 | 185.88 | 8.17 |
| RL sampling | 234.45 | 0.28 | 209.65 | 2.12 | 174.88 | 1.71 |
| Ensemble RL | *234.09* | *0.11* | *205.67* | *0.16* | *172.64* | *0.39* |
| Note – Averages of 10 problem instances. ‘Cost’ refers to the average cost value. ‘Gap’ is the mean relative difference to the cost of the best algorithm for each instance. ‘Time’ is the average solution time of the algorithm for a single instance. ‘Kc’ is the number of vehicles and ‘Dc’ is the number of Drones, ‘Nc’ is number of nodes | | | | | | | |

While we lacked comparative results from other methods for the 20-node problems, the Ensemble RL model demonstrated impressive performance. With 1 vehicle and 2 drones, the Ensemble RL model had an average cost of 234.09 and a minimal gap of 0.11%. For the configuration of 2 vehicles with 1 drone each, the model showed a cost of 205.67 and a gap of 0.16%. With 2 vehicles and 2 drones each, RL sampling achieved a cost of 172.64 and a gap of 0.39%.

Notably, the Ensemble RL model generally outperformed all other methods across most configurations. However, for the 10-node configuration with 2 vehicles and 2 drones, the RL sampling model showed better results. Overall, the Ensemble RL model consistently outperformed or closely matched the best-performing methods across different configurations and problem sizes, highlighting its superior efficiency and robustness in solving the Vehicle Routing Problem with Drones (VRPD).

## 3.3 Results on VRPD with Unlimited Flying Range

Our study involved an extensive evaluation of the performance of our RL sampling model on various graph instances with different numbers of nodes (8, 10, 20, and 50). We used 10 instances of these graphs to test the efficiency of our model, which were similar to the ones used in a previous study by Schermer et al. (2019).

During our evaluation, we focused on different configurations that involved adjusting the drone’s flight range, determining the number of drones assigned to each vehicle, and changing the number of vehicles. We tested three configurations to evaluate the model’s performance and compared it to other methods we tested. In the first configuration, we used a single vehicle with two drones. Our results showed that the RL sampling model was almost 70 times faster than other methods we tested for small-scale problems with 8 nodes. It also demonstrated a 1.45% optimality gap, which was better than other methods we tested.

The second configuration involved two vehicles with one drone assigned to each vehicle. The RL sampling model outperformed all other methods we tested for small-scale problems with 8 and 10 nodes. It showed a 0.96% optimality gap for 10-node problems.

In the third configuration, we used two vehicles with two drones assigned to each vehicle. The RL sampling model showed a 2.14% optimality gap for small-scale problems with 8 nodes, which was better than other methods we tested. For 10-node problems, it performed better than other methods in the initial two configurations and achieved a minimal 0.92% gap compared to exact methods in the third configuration.

We also tested the model on big-scale problems with 20 nodes. However, no results were reported for other methods, so we only reported the RL model results for this problem size. Overall, our study showed that the RL sampling model was efficient and outperformed other methods in most of the tested configurations for small-scale problems (table 3.2).

Table 3.2 – VRPD results with Infinite Flying Range of Drones

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Category | Methods | |Kc|=1, |Dc|=2 | | | |Kc|=2, |Dc|=1 | | | |Kc|=2, |Dc|=2 | | |
| costs | gaps % | time (s) | costs | gaps % | time (s) | costs | gaps % | time (s) |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| |Nc| = 8 | Solver | *158.53* | *0.51* | 175.53 | 149.53 | 1.1 | 601.33 | 125.55 | 1.09 | 481.62 |
| Continuation of table 3.2 | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|  | Solver Lazy | *158.53* | *0.51* | 76.97 | *147.6* | *0.0* | 148.52 | *123.9* | *0.0* | 161.35 |
| Heuristic | *158.53* | *0.51* | 154.55 | 150.32 | 1.94 | 61.84 | 125.5 | 1.06 | 65.06 |
| RL greedy | 161.69 | 2.38 | *0.02* | 148.68 | 0.9 | *0.03* | 126.98 | 2.25 | 0.21 |
| RL sampling | 160.23 | 1.6 | *0.87* | 148.32 | 0.59 | *1.31* | 126.87 | 2.14 | 4.0 |
| Ensemble RL | 159.95 | 1.45 | *6.02* | *147.6* | *0.0* | *9.27* | 126.87 | 2.14 | 28.18 |
| |Nc| = 10 | Solver | 196.48 | 11.25 | 601.29 | 193.75 | 16.42 | 601.52 | 171.78 | 30.89 | 601.69 |
| Solver Lazy | 195.26 | 10.78 | 556.72 | 200.91 | 20.42 | 600.67 | 176.63 | 34.64 | 602.05 |
| Heuristic | 178.93 | 1.42 | 308.29 | 167.59 | 0.82 | 63.68 | *131.36* | *0.23* | 99.1 |
| RL greedy | 185.42 | 4.74 | *0.01* | 168.4 | 1.22 | *0.4* | 136.06 | 3.58 | *0.03* |
| RL sampling | 180.26 | 1.89 | *0.43* | 167.57 | 0.71 | *1.43* | 132.4 | 0.92 | *2.06* |
| Ensemble RL | *177.01* | *0.28* | *3.05* | *167.56* | *0.7* | *9.38* | 134.2 | 2.23 | *14.65* |
| |Nc| = 20 | RL greedy | 243.06 | 3.86 | 0.02 | 219.02 | 6.91 | 0.02 | 185.88 | 8.17 | 0.32 |
| RL sampling | 234.45 | 0.28 | 1.31 | 209.65 | 2.12 | 4.24 | 174.88 | 1.71 | 8.01 |
| Ensemble RL | 234.09 | 0.12 | 8.94 | 205.67 | 0.15 | 29.42 | 172.64 | 0.39 | 55.6 |
| Note – Averages of 10 problem instances. ‘Cost’ refers to the average cost value. ‘Gap’ is the mean relative difference to the cost of the best algorithm for each instance. ‘Time’ is the average solution time of the algorithm for a single instance. ‘Kc’ is the number of vehicles and ‘Dc’ is the number of Drones, ‘Nc’ is number of nodes | | | | | | | | | | |

**4 DISCUSSIONS**

Experiments conducted in this thesis have explored the significant potential of reinforcement learning (RL) in optimizing logistics operations involving both ground vehicles and aerial drones through detailed simulations under various theoretical conditions. The methodological approach and the resulting findings underscore the adaptability and superior efficiency of RL models compared to traditional logistics methods, offering insights into their potential application in real-world logistics scenarios.

The methodology employed a series of advanced scenario simulations designed to test the RL models under a range of conditions. These simulations focused on vehicle capacities and operational constraints while excluding experiments on limited drone flight ranges. This approach facilitated a comprehensive assessment of how RL models manage logistics complexity and variability. The simulations were structured to progressively increase in complexity, thus providing a robust test of the RL algorithms’ scalability and adaptability.

By leveraging various RL algorithms, the models were tailored to dynamically adjust their strategies based on the specific requirements of each scenario. This dynamic adjustment is critical in logistics, where operational conditions can change rapidly due to external factors such as traffic congestion, weather conditions, and varying delivery schedules.

The results from these simulations demonstrated that RL models consistently outperformed traditional logistics methods across several key metrics:

1. *Efficiency:*RL models reduced the time and resources required to complete logistics operations by optimizing routes and vehicle utilization based on real-time scenario data.
2. *Adaptability:*The models exhibited a high degree of flexibility, effectively adapting to a wide range of operational scenarios and constraints without the need for extensive reconfiguration.
3. *Scalability:*RL models proved capable of scaling to handle larger logistics networks without a loss in performance, showcasing their potential for application in both small and large-scale operations.

Moreover, the ability of RL models to learn from and adapt to complex environments indicates their suitability for integration into increasingly automated and data-driven logistics processes.

An unexpected yet critical aspect of the research was the potential environmental impact of integrating RL-guided drone technology with traditional vehicle logistics. The simulations indicated that using drones could lead to a substantial decrease in the carbon footprint of delivery networks. Drones, when optimally deployed, reduce reliance on larger, more polluting vehicles and can access areas that might otherwise require multiple vehicle trips. This aspect of the research aligns with global efforts to reduce emissions and suggests that RL could play a significant role in creating more sustainable logistics practices.

# CONCLUSION

This thesis has systematically explored the application of reinforcement learning (RL) to optimize logistics operations involving both ground vehicles and aerial drones. Through a series of detailed simulations under a variety of theoretical conditions, the research has demonstrated that RL models offer substantial improvements over traditional logistics methods in terms of efficiency, adaptability, and scalability. These findings not only affirm the potential of RL to transform logistics operations but also highlight its capacity to contribute to more sustainable practices within the industry.

*Key Contributions to the Field of Logistics and Operational Management that align with the study objectives:*

1. Theoretical Innovation in MDP Formulation: The introduction of the first MDP formulation for the vehicle routing problem with drones provides a new methodological framework for addressing the complexities of dynamic and uncertain environments in logistics.
2. Development of a Novel Deep Learning Model: This model enhances the application of artificial intelligence in logistics by improving the predictive accuracy and operational efficiency of drone routing.
3. Creation of a Novel Reinforcement Learning Algorithm: The development of a new RL algorithm optimized for the efficient training and application of deep learning models in logistics. This algorithm demonstrates significant advancements in learning efficacy and computational efficiency, making it a valuable tool for real-time operational decision-making.
4. Empirical Validation through Advanced Simulations: The simulations conducted as part of this research provided robust empirical evidence of the superiority of RL models over traditional logistics methods across key performance metrics such as efficiency, adaptability, and scalability.

*Future Research Directions:*

1. Empirical Validation: Future studies should focus on implementing RL models in real-world logistics operations to empirically validate the findings and refine the models based on practical challenges.
2. Integration with Existing Systems: Exploring how RL models can be seamlessly integrated with existing logistics infrastructures, addressing compatibility issues, data integration, and system resilience.
3. Regulatory and Ethical Considerations: As the use of drones and automated systems in logistics expands, comprehensive research on regulatory and ethical issues will be essential to ensure responsible deployment.

In conclusion, this thesis lays a robust foundation for the adoption of reinforcement learning in logistics, showcasing its potential to enhance operational efficiency, adaptability, and sustainability. The continued exploration and development of RL models are critical to realizing their full potential, potentially revolutionizing logistics operations globally. As industries increasingly move towards automation and intelligent systems, the role of RL in logistics is poised to become a pivotal element in shaping the future of the sector.

# REFERENCES

1. Varadarajan R., Srinivasan R., Vadakkepatt G. et al. Interactive technologies and retailing strategy: a review, conceptual framework and future research directions // Journal of Interactive Marketing. – 2010. – Vol. 24, №2. – P. 96-110.
2. Currah A. Behind the web store: the organisational and spatial evolution of multichannel retailing in toronto // Environment and Planning a Economy and Space. – 2002. – Vol. 34, №8. – P. 1411-1441.
3. Weijters B., Rangarajan D., Falk T. et al. Determinants and outcomes of customers’ use of self-service technology in a retail setting // Journal of Service Research. – 2007. – Vol. 10, №1. – P. 3-21.
4. Bakos Y., Lucas H., Oh W. et al. The impact of e-commerce on competition in the retail brokerage industry // Information Systems Research. – 2005. – Vol. 16, №4. – P. 352-371.
5. Kazakhstan plans to increase share of e-commerce to 20% by 2030 // <https://primeminister.kz/en/news/kazakhstan-plans-to-increase-share-of.> 16.12.2024.
6. Housing stock statistics (2023) // <https://stat.gov.kz/en>. 28.02.2024.
7. Постановление Правительства Республики Казахстан. Об утверждении обновленного национального вклада Республики Казахстан в глобальное реагирование на изменение климата: утв. 19 апреля 2023 года, №313.https://adilet.zan.kz/rus/docs/P2300000313. 19.04.2023.
8. President addresses key AI and digital transformation priorities at Digital Bridge 2023 // <https://baiterek.gov.kz/en/pr/news/president-addresses>. 12.10.2023.
9. Lindner J. Online food delivery statistics [fresh research] // <https://gitnux.org/online-food-delivery-statistics/>. 16.12.2023.
10. Dantzig G.B., Ramser J.H. The truck dispatching problem // Management Science. – 1959. – Vol. 6, №1. – P. 80-91.
11. Toth P., Vigo D. The vehicle routing problem. – Philadelphia: SIAM, 2002. – 367 p.
12. Pillac V., Gendreau M., Guéret C. et al. A review of dynamic vehicle routing problems // European Journal of Operational Research. – 2013. – Vol. 225, №1. – P. 1-11.
13. Bogyrbayeva A., Meraliyev M., Mustakhov T. et al. Machine learning to solve vehicle routing problems: A survey // IEEE Transactions on Intelligent Transportation Systems. – 2024. – Vol. 25, №6. – P. 4754-4772.
14. Rizzoli A.E., Montemanni R., Lucibello E. et al. Ant colony optimization for real-world vehicle routing problems // Swarm Intelligence. – 2007. – Vol. 1, №2. – P. 135-151.
15. Solomon M.M. Algorithms for the vehicle routing and scheduling problems with time window constraints // Operations Research. – 1987. – Vol. 35, №2. – P. 254-265.
16. Psaraftis H.N. Dynamic vehicle routing: Status and prospects // Annals of Operations Research. – 1995. – Vol. 61, №1. – P. 143-164.
17. Szeto W., Wu Y., Ho S. An artificial bee colony algorithm for the capacitated vehicle routing problem // European Journal of Operational Research. – 2011. – Vol. 215. – P. 126-135.
18. Tu W., Li Q., Li Q. et al. A spatial parallel heuristic approach for solving very large-scale vehicle routing problems // Transactions in GIS. – 2017. – Vol. 21. – P. 1130-1147.
19. Yuliza E., Puspita F., Supadi S. Heuristic approach for robust counterpart open capacitated vehicle routing problem with time windows // Science Technology Indonesia. – 2021. – Vol. 6. – P. 53-57.
20. Stopka O., Gross P., Pečman J. et al. Optimization of the pick-up and delivery technology in a selected company: a case study // Technologies. – 2022. – Vol. 10. – P. 84-1-84-23.
21. Yu Z., Zhang P., Yu Y. et al. An adaptive large neighborhood search for the larger-scale instances of green vehicle routing problem with time windows. // Complexity. – 2020. – Vol. 2020. – P. 1-14.
22. Zhu Y., Lee K., Wang Y. Adaptive elitist genetic algorithm with improved neighbor routing initialization for electric vehicle routing problems // IEEE Access. – 2021. – Vol. 9. – P. 16661-16671.
23. Liu Y., Pan H., Sun G. et al. Joint scheduling and trajectory optimization of charging UAV in wireless rechargeable sensor networks // IEEE Internet of Things Journal. – 2022. – Vol. 9. – P. 11796-11813.
24. Shahadat A., Akhand M., Kamal A. Visibility adaptation in ant colony optimization for solving traveling salesman problem // Mathematics. – 2022. – Vol. 10. – P. 2448-1-2448-24.
25. Wang C., Zhu R., Jiang Y. et al. A scheme library-based ant colony optimization with 2-opt local search for dynamic traveling salesman problem. //Computer Modeling in Engineering Sciences. – 2023. – Vol. 135. – P. 1209-1228.
26. Ahmed Z., Hameed A., Mutar M. Hybrid genetic algorithms for the asymmetric distance-constrained vehicle routing problem // Mathematical Problems in Engineering. – 2022. – Vol. 2022. – P. 1-20.
27. Ismail A., Hartono N., Zeybek S. et al. Combinatorial bees algorithm for vehicle routing problem // Macromolecular Symposia. – 2021. – Vol. 396. – P. 2000284.
28. Máximo V. A hybrid adaptive iterated Local Search with diversification control to the Capacitated Vehicle Routing Problem // https://www.researchgate.net/publication/347534968\_A\_hybrid\_. 10.11.2023.
29. Alweshah M., Almiani M. et al. Vehicle routing problems based on Harris Hawks optimization // Journal of Big Data. – 2022. – Vol. 9. – P. 1-20.
30. Gokalp O., Uğur A. Solving vehicle routing problem for multistorey buildings using iterated local search // Turkish Journal of Electrical Engineering and Computer Sciences. – 2019. – Vol. 27. – P. 3516-3531.
31. Agárdi A., Kovács L., Bányai T. An attraction map framework of a complex multi-echelon vehicle routing problem with random walk analysis. // Applied Sciences. – 2021. – Vol. 11. – P. 2100-1-2100-23.
32. Linfati R., Escobar J. Reoptimization heuristic for the capacitated vehicle routing problem // Journal of Advanced Transportation. – 2018. –Vol. 2018. – P. 1-8.
33. Zhou Y., Liret A., Liu J. et al. Decision support system for green real-life field scheduling problems // Computational Logistics: proceed. 8th internat. conf. (ICCL 2017). – Southampton, UK, 2017. – P. 355-369.
34. Sajid M., Singh J. et al. A novel algorithm for capacitated vehicle routing problem for smart cities // Symmetry. – 2021. – Vol. 13. – P. 1923-1-1923-23.
35. Zhang W., Gajpal Y. et al. Multi-depot green vehicle routing problem to minimize carbon emissions // Sustainability. – 2020. – Vol. 12. – P. 3500-1-3500-19.
36. Joshi C.K., Laurent T., Bresson X. An efficient graph convolutional network technique for the travelling salesman problem // https://graphdeeplearning.github.io/publication/joshi-2019-efficient/. 10.11.2023.
37. Milan A., Rezatofighi S.H., Garg R. et al. Data-driven approximations to NP-hard problems // Proceed. of the 31st AAAI conf. on Artificial Intelligence (AAAI-17). – San Francisco, 2017. – P. 1453-1459.
38. Prates M., Avelar P.H.C., Lemos H. et al. Learning to solve NP-complete problems: A graph neural network for decision TSP // Proceed. of the AAAI conf. on Artificial Intelligence. – 2019. – Vol. 33. – P. 4731-4738.
39. Joshi C.K., Laurent T., Bresson X. On learning paradigms for the travelling salesman problem // https://arxiv.org/abs/1910.07210. 10.11.2023.
40. Joshi C.K., Cappart Q., Rousseau L.-M. et al. Learning the travelling salesman problem requires rethinking generalization // Constraints. – 2022. – Vol. 27, №1. – P. 70-98.
41. Van Knippenberg M., Holenderski M., Menkovski V. Complex vehicle routing with memory augmented neural networks // Proceed. 2020 IEEE conf. on Industrial Cyberphysical Systems (ICPS). – Tampere, 2020. – Vol. 1. – P. 303-308.
42. Nazari M., Oroojlooy A., Snyder L. et al. Reinforcement learning for solving the vehicle routing problem // Advances in Neural Information Processing Systems. – 2018. – Vol. 31. – P. 1-21.
43. Zhang Z., Qi G., Guan W. Coordinated multi-agent hierarchical deep reinforcement learning to solve multi-trip vehicle routing problems with soft time windows // IET Intelligent Transport Systems. – 2023. – Vol. 17. – P. 2034-2051.
44. Zheng Y., Tao J., Sun Q. et al. An intelligent course keeping active disturbance rejection controller based on double deep Q-network for towing system of unpowered cylindrical drilling platform // International Journal of Robust and Nonlinear Control. – 2021. – Vol. 31. – P. 8463-8480.
45. Quek Y., Koh L., Koh N. et al. Deep Q-network implementation for simulated autonomous vehicle control // IET Intelligent Transport Systems. – 2021. – Vol. 15. – P. 875-885.
46. Zhao Y., Wang Y., Tan Y. et al. Dynamic job-shop scheduling algorithm based on deep Q network // IEEE Access. – 2021. – Vol. 9. – P. 122995-123011.
47. Iqbal A., Tham M., Chang Y. Double deep Q-network-based energy-efficient resource allocation in cloud radio access network // IEEE Access. – 2021. – Vol. 9. – P. 20440-20449.
48. Dai H., Khalil E.B., Zhang Y. et al. Learning combinatorial optimization algorithms over graphs // Proceed. 31st conf. on Neural Information Processing Systems (NIPS 2017). – Long Beach, CA, 2017. – Vol. 30. – P. 1-24.
49. Mnih V., Kavukcuoglu K., Silver D. et al. Human-level control through deep reinforcement learning // Nature. – 2015. – Vol. 518, №7540. – P. 529-533.
50. Riedmiller M. Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method // Lecture Notes in Computer Science. – 2005. – Vol. 3720. – P. 317-328.
51. Schulman J., Wolski F., Dhariwal P. et al. Proximal policy optimization algorithms // https://arxiv.org/abs/1707.06347. 10.11.2023.
52. Ma Z., Cui T., Deng W.J. et al. Adaptive optimization of traffic signal timing via deep reinforcement learning // Journal of Advanced Transportation. – 2021. – Vol. 2021. – P. 1-14.
53. Ju M., Hou S., Yu F. et al. Adaptive kernel graph neural network // Proceed. of the AAAI conf. on Artificial Intelligence. – Palo Alto, 2022. – Vol. 36. – P. 7051-7058.
54. Li J. From uni-relational to multi-relational graph neural networks // Proceed. of the 15th ACM internat. conf. on Web Search and Data Mining. – NY., 2022. – P. 1551-1552.
55. Song Y., Kang Q., Wang S. et al. On the robustness of graph neural diffusion to topology perturbations // Advances in Neural Information Processing Systems. – 2022. – Vol. 35. – P. 6384-6396.
56. Kool W., van Hoof H., Welling M. Attention, learn to solve routing problems! // https://arxiv.org/abs/1803.08475. 10.11.2023.
57. Bello I., Pham H., Le Q.V. et al. Neural combinatorial optimization with reinforcement learning // https://arxiv.org/abs/1611.09940.
58. Peng B. et al. A deep reinforcement learning algorithm using dynamic attention model for vehicle routing problems // Procced. internat. sympos. on Intelligence Computation and Applications. – Singapore, 2019. – P. 636-650.
59. Xin L., Song W., Cao Z. et al. Step-wise deep learning models for solving routing problems // IEEE Transactions on Industrial Informatics. – 2020. – Vol. 17, №7. – P. 4861-4871.
60. da Costa P.R., Rhuggenaath J., Zhang Y. et al. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning // Proceed. of Machine Learning Research. – 2020. – Vol. 129. – P. 465-480.
61. Vaswani A., Shazeer N., Parmar N. et al. Attention is All You Need // In book: Advances in Neural Information Processing Systems. – Long Beach, 2017. – P. 5998-6008.
62. Jiang Y., Wu Y., Cao Z. et al. Learning to solve routing problems via distributionally robust optimization // Proceed. of the AAAI Conference on Artificial Intelligence. – Vancouver, 2022. – Vol. 36. – P. 9786-9794.
63. Senuma Y., Wang Z., Nakano Y. et al. GEAR: A graph edge attention routing algorithm solving combinatorial optimization problem with graph edge cost. // Proceed. of the 10th ACM SIGSPATIAL internat. Workshop on Analytics for Big Geospatial Data. – NY., 2022. – P. 8-16.
64. Li K., Zhang T., Wang R. et al. Deep reinforcement learning for combinatorial optimization: Covering Salesman Problems // IEEE Transactions on Cybernetics. – 2022. – Vol. 52, №12. – P. 13142-13155.
65. Zhang K., He F., Zhang Z. et al. Multi-vehicle routing problems with soft time windows: A multi-agent reinforcement learning approach // Transportation Research Part C: Emerging Technologies. – 2020. – Vol. 121. – P. 102861.
66. Xin L., Song W., Cao Z. et al. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems // Proceed. of the 35th AAAI conf. on Artificial Intelligence. – Palo Alto, 2021. – P. 12042-12049.
67. Kim M., Park J. et al. Learning collaborative policies to solve NP-hard routing problems // Advances in Neural Information Processing Systems. – 2021. – Vol. 34. – P. 10418-10430.
68. Kwon Y.-D., Choo J., Kim B. et al. POMO: Policy optimization with multiple optima for reinforcement learning // Advances in Neural Information Processing Systems. – 2020. – Vol. 33. – P. 21188-21198.
69. Goh Y.L., Lee W.S., Bresson X. et al. Combining reinforcement learning and optimal transport for the traveling salesman problem // https://www.researchgate.net/publication/358975362\_Combining\_. 10.11.2023.
70. James J.Q., Yu W., Gu J. Online vehicle routing with neural combinatorial optimization and deep reinforcement learning // IEEE Transactions on Intelligent Transportation Systems. – 2019. – Vol. 20, №10. – P. 3806-3817.
71. Lin B., Ghaddar B., Nathwani J. Deep reinforcement learning for the electric vehicle routing problem with time windows // IEEE Transactions on Intelligent Transportation Systems. – 2022. – Vol. 23, №8. – P. 11528-11538.
72. Pan W., Liu S.Q. Deep reinforcement learning for the dynamic and uncertain vehicle routing problem // Applied Intelligence. – 2023. – Vol. 53, №1. – P. 405-422.
73. Bahdanau D., Cho K., Bengio Y. Neural machine translation by jointly learning to align and translate // Procced. 3rd internat. conf. on Learning Representations (ICLR 2015). – San Diego, 2015. – P. 1-16.
74. Vera J.M., Abad A.G. Deep reinforcement learning for routing a heterogeneous fleet of vehicles // Procced. 2019 IEEE Latin American conf. on Computational Intelligence (LA-CCI). – Guayaquil, 2019. – P. 1-6.
75. Falkner J.K., Schmidt-Thieme L. Learning to solve vehicle routing problems with time windows through joint attention // https://www.researchgate.net/publication/342229883\_Learning\_to\_. 10.11.20223.
76. Lei K., Guo P., Wang Y. et al. Solve routing problems with a residual edge-graph attention neural network // Neurocomputing. – 2022. – Vol. 508. – P. 79-98.
77. Gunarathna U., Borovica-Gajic R., Karunasekara S. et al. Solving dynamic graph problems with multi-attention deep reinforcement learning // https://www.researchgate.net/publication/357823714\_Solving\_Dynamic. 10.11.2023.
78. Li J., Ma Y., Cao Z. et al. Learning feature embedding refiner for solving vehicle routing problems // IEEE Transactions on Neural Networks and Learning Systems. – 2023. – Vol. 10. – P. 1-13.
79. Emami P., Ranka S. Learning permutations with Sinkhorn policy gradient // <https://arxiv.org/abs/1805.07010>. 10.11.2023.
80. Li K., Zhang T., Wang R. Deep reinforcement learning for multiobjective optimization // IEEE Transact. on Cybernet. – 2020. – Vol. 51, №6. – P. 3103-3114.
81. Hottung A., Kwon Y.-D., Tierney K. Efficient active search for combinatorial optimization problems // https://arxiv.org/abs/2106.05126. 10.11.2023.
82. Li J., Xin L., Cao Z. et al. Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning // IEEE Transactions on Intelligent Transportation Systems. – 2022. – Vol. 23, №3. – P. 2306-2315.
83. Sykora Q. et al. Multi-agent routing value iteration network // Procced. Internat. conf. on Machine Learning. – Vienna, 2020. – P. 9300-9310.
84. Xu Y., Fang M., Chen L. et al. Reinforcement learning with multiple relational attention for solving vehicle routing problems // IEEE Transactions on Cybernetics. – 2022. – Vol. 52, №10. – P. 11107-11120.
85. Drori I., Kharkar A., Sickinger W.R. et al. Learning to solve combinatorial optimization problems on real-world graphs in linear time // Procced. 19th IEEE internat. conf. on Machine Learning and Applications (ICMLA). – Miami, 2020. – P. 19-24.
86. Li J., Ma Y., Gao R. et al. Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. //IEEE Transactions on Cybernetics. – 2021. – P. 1-14.
87. Duan L., Zhan Y., Hu H. et al. Efficiently solving the practical vehicle routing problem: A novel joint learning approach // Proceed. of the 26th ACM SIGKDD internat. conf. on Knowledge Discovery & Data Mining. – NY., 2020. – P. 3054-3063.
88. Alesiani F. Bigrad: Differentiating through bilevel optimization programming // https://paperswithcode.com/paper/bigrad-differentiating. 10.11.2023.
89. Pogancic M.V., Paulus A., Musil V. et al. Differentiation of blackbox combinatorial solvers // Procced. 8th internat. conf. on Learning Representations, (ICLR 2020). – Addis Ababa, 2020. – P. 1-19.
90. Dorigo M., Gambardella L.M. Ant-q: A reinforcement learning approach to the traveling salesman problem // Machine Learning, proceed. of the 12th internat. conf. on Machine Learning. – Tahoe City, 1995. – P. 252-260.
91. Liu F. et al. Study of genetic algorithm with reinforcement learning to solve the TSP // Expert Systems with Applications. – 2009. – Vol. 36, №3. – P. 6995-7001.
92. Alipour M.M., Razavi S.N. et al. A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem // Neural Computing and Applications. – 2018. – Vol. 30, №9. – P. 2935-2951.
93. Qi R., Li J., Wang J. et al. QMOEA: A Q-learning-based multiobjective evolutionary algorithm for solving time-dependent green vehicle routing problems with time windows // Information Sciences. – 2022. – Vol. 608. – P. 178-201.
94. Niu Y., Shao J., Xiao J. et al. Multi-objective evolutionary algorithm based on RBF network for solving the stochastic vehicle routing problem // Information Sciences. – 2022. – Vol. 609. – P. 387-410.
95. Hottung A., Tierney K. Neural large neighborhood search for the capacitated vehicle routing problem // https://arxiv.org/abs/1911.09539. 10.11.2023.
96. Silva M.A.L., de Souza S.R., Souza M.J.F. et al. A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems // Expert Systems with Applications. – 2019. – Vol. 131. – P. 148-171.
97. Cappart Q., Moisan T., Rousseau L.-M. et al. Combining reinforcement learning and constraint programming for combinatorial optimization // Proceed. of the AAAI Conference on Artificial Intelligence. – 2021. – Vol. 35. – P. 3677-3687.
98. Alesiani F., Ermis G., Gkiotsalitis K. Constrained clustering for the capacitated vehicle routing problem (CC-CVRP) // Applied Artificial Intelligence. – 2022. – Vol. 36, №1. – P. 1995658.
99. Wang Y., Ran L., Guan X. et al. Collaborative multicenter vehicle routing problem with time windows and mixed deliveries and pickups // Expert Systems with Applications. – 2022. – Vol. 197. – P. 116690.
100. Ma Y., Li J., Cao Z. et al. Learning to iteratively solve routing problems with dual-aspect collaborative transformer // Advances in Neural Information Processing Systems. – 2021. – Vol. 34. – P. 11096-11107.
101. Wu Y., Song W., Cao Z. et al. Learning improvement heuristics for solving routing problems // IEEE Transactions on Neural Networks and Learning Systems. – 2022. – Vol. 33, №9. – P. 5057-5069.
102. Deudon M., Cournut P., Lacoste A. et al. Learning heuristics for the TSP by policy gradient // Procced. internat. conf. on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research. – Delft, 2018. – P. 170-181.
103. Zhao J., Mao M., Zhao X. et al. A hybrid of deep reinforcement learning and local search for the vehicle routing problems // IEEE Transactions on Intelligent Transportation Systems. – 2021. – Vol. 22, №11. – P. 7208-7218.
104. Ma Q., Ge S., He D. et al. Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning // <https://www.researchgate.net> /publication/337208780\_Combinatorial\_Optimization\_by\_Graph\_. 10.11.2023.
105. Zhang R., Prokhorchuk A., Dauwels J. Deep reinforcement learning for traveling salesman problem with time windows and rejections // Procced. 2020 internat. Joint conf. on Neural Networks (IJCNN). – Glasgow, 2020. – P. 1-8.
106. Ma Y., Li J., Cao Z. et al. Efficient neural neighborhood search for pickup and delivery problems // <https://www.researchgate.net.> 10.11.2023.
107. Lin K., Zhao R. et al. Efficient large-scale fleet management via multi-agent deep reinforcement learning // Proceed. of the 24th ACM SIGKDD internat. conf. on Knowledge Discovery & Data Mining. – NY., 2018. – P. 1774-1783.
108. Hottung A. et al. Learning a latent search space for routing problems using variational autoencoders // https://paperswithcode.com. 10.11.2023.
109. Syed A.A., Akhnoukh K., Kaltenhaeuser B. et al. Neural network based large neighborhood search algorithm for ride hailing services // Procced. EPIA conf. on Artificial Intelligence. – Vila Real, 2019. – P. 584-595.
110. Fernandes F.C. et al. A multiagent architecture for solving combinatorial optimization problems through metaheuristics // Procced. 2009 IEEE internat. conf. on Systems, Man and Cybernetics. – San Antonio, 2009. – P. 3071-3076.
111. Kool W., van Hoof H. et al. Deep policy dynamic programming for vehicle routing problems // Integration of Constraint Programming, Artificial Intelligence, and Operations Research: procced. 19th internat. conf. (CPAIOR 2022). – Los Angeles, 2022. – P. 190-213.
112. Xu S., Panwar S.S. et al. Deep neural network approximated dynamic programming for combinatorial optimization // Proceed. of the AAAI conf. on Artificial Intelligence. – Vancouver, 2020. – P. 1684-1691.
113. Delarue A., Anderson R., Tjandraatmadja Ch. Reinforcement learning with combinatorial actions: An application to vehicle routing // Advances in Neural Information Processing Systems. – 2020. – Vol. 33. – P. 609-620.
114. Chen X., Tian Y. Learning to perform local rewriting for combinatorial optimization // Advances in Neural Information Processing Systems 32: procced. annual conf. on Neural Information Processing Systems (NeurIPS 2019). – Vancouver, 2019. – P. 6278-6289.
115. Ragan-Kelley J., Barnes C., Adams A. et al. Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines // Acm Sigplan Notices. – 2013. – Vol. 48, №6. – P. 519-530.
116. Li S., Yan Z., Wu C. Learning to delegate for large-scale vehicle routing // Advances in Neural Information Processing Systems. – 2021. – Vol. 34. – P. 26198-26211.
117. Chen X., Ulmer M.W., Thomas B.W. Deep q-learning for same-day delivery with vehicles and drones // European Journal of Operational Research. – 2022. – Vol. 298, №3. – P. 939-952.
118. Tyasnurita R. et al. Learning heuristic selection using a time delay neural network for open vehicle routing // Procced. 2017 IEEE Congress on Evolutionary Computation (CEC). – Donostia, 2017. – P. 1474-1481.
119. Xing Z., Tu S. A graph neural network assisted monte carlo tree search approach to traveling salesman problem // IEEE Access. – 2020. – Vol. 8. – P. 108418-108428.
120. Fu Z.-H., Qiu K.-B., Zha H. Generalize a small pre- trained model to arbitrarily large tsp instances // Proceed. of the AAAI conf. on Artificial Intelligence. – 2021. – Vol. 35. – P. 7474-7482.
121. Gutierrez-Rodríguez A. et al. Selecting meta-heuristics for solving vehicle routing problems with time windows via meta-learning // Expert Systems with Applications. – 2019. – Vol. 118. – P. 470-481.
122. Tan K.C., Chew Y.H., Lee L.H. A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows // Computational Optimization and Applications. – 2006. – Vol. 34, №1. – P. 115-151.
123. Ombuki B., Ross B.J., Hanshar F. Multi-objective genetic algorithms for vehicle routing problem with time windows // Applied Intelligence. – 2006. – Vol. 24, №1. – P. 17-30.
124. Qi Y., Hou Z., Li H. et al. A decomposition based memetic algorithm for multi-objective vehicle routing problem with time windows // Computers & Operations Research. – 2015. – Vol. 62. – P. 61-77.
125. Wu D.Q., Dong M., Li H.Y. et al. Vehicle routing problem with time win- dows using multi-objective co-evolutionary approach // International Journal of Simulation Modelling. – 2016. – Vol. 15, №4. – P. 742-753.
126. Fu Z.-H., Qiu K.-B., Qiu M. et al. Targeted sampling of enlarged neighborhood via monte carlo tree search for tsp // https://openreview.net/forum?id=ByxtHCVKwB9. 10.11.2023.
127. Xin L., Song W., Cao Z. et al. Neurolkh: Combining deep learning model with lin-kernighan-helsgaun heuristic for solving the traveling salesman problem // Proceed. of the 35th conf. on Neural Information Processing Systems. – San Diego, 2021. – P. 7472-7483.
128. Yang F., Jin T., Liu T.-Y. et al. Boosting dynamic programming with neural networks for solving np-hard problems // Procced. Asian conf. on Machine Learning. – Hanoi, 2018. – P. 726-739.
129. da Costa P., Rhuggenaath J. et al. Learning 2-opt heuristics for routing problems via deep reinforcement learning // SN Computer Science. – 2021. – Vol. 2, №5. – P. 1-16.
130. Perron L. et al. Or-tools // [https://developers.google.com. 10.11.2023.](https://developers.google.com/optimization/)
131. Vinyals O., Fortunato M., Jaitly N. Pointer networks // https://arxiv.org/abs/1506.03134. 10.11.2023.
132. Bogyrbayeva A. et al. A reinforcement learning approach for rebalancing electric vehicle sharing systems // https://arxiv.org/abs/2010.02369. 10.11.2023.
133. Bogyrbayeva A. et al. A deep reinforcement learning approach for solving the traveling salesman problem with drone // Transportation Research Part C: Emerging Technologies. – 2023. – Vol. 148. – P. 103981.
134. Williams RJ. Simple statistical gradient-following algorithms for con- nectionist reinforcement learning // Machine learning. – 1992. – Vol. 8, №3. – P. 229-256.
135. Agatz N., Bouman P., Schmidt M. Optimization approaches for the traveling salesman problem with drone // Transportation Science. – 2018. – Vol. 52, №4. – P. 965-981.
136. Schermer D., Moeini M., Oliver W. A hybrid vns/tabu search algorithm for solving the vehicle routing problem with drones and en route operations // Computers & Operations Research. – 2019. – Vol. 109. – P. 134-158.

# APPENDIX A

The performance comparison

Table 1 – The performance comparison to solve VRPTW on the Solomon benchmark problems

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Category | Model | VRPTW20 | | | | VRPTW50 | | | |
| Cost | K | Dist | Tinf | Cost | K | Dist | Tinf |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| TW1 | Falkner et al.  OR-Tools-AU | 2577.08 | 4.18 | 643.77 | 0.22s | 4344.88 | 6.3 | 1110.68 | 1.76s |
| Falkner et al.  OR-Tools-GLS | 2522.85 | 4.11 | 617.77 | 8.00s | 4213.23 | 6.15 | 1090.36 | 8.06s |
| random (1000) | 3036.39 | 5.68 | 1200.37 | - | 7297.53 | 11.8 | 2914.86 | - |
| Kool et al. AM+TW(greedy) | 3766.88 | 5.29 | 1264.4 | 0.05s | 7189.43 | 9.42 | 2882.53 | 0.12s |
| Kool et al. AM+TW(sampl.) | 3041.24 | 5.74 | 1202.64 | 0.12s | 7327.09 | 11.92 | 2921.39 | 0.38s |
| Kool et al. AM+TW(t10240) | 2750.06 | 5.27 | 1163.58 | 0.95s | 6878.84 | 11.28 | 2865.3 | 3.08s |
| Falkner et al. JAMPR(greedy) | 1862.4 | 2.25 | 966.74 | 0.10s | 3055.94 | 5.42 | 1733.24 | 0.24s |
| Falkner et al. JAMPR(sampl.) | 1716.6 | 2.29 | 965.42 | 0.86s | 2691.55 | 4.03 | 1811.06 | 3.07s |
| TW2 | Falkner et al.  OR-Tools-AU | 635.06 | 4.23 | 635.06 | 0.37s | 1123.82 | 6.72 | 1123.82 | 3.81s |
| Falkner et al.  OR-Tools-GLS | 619.57 | 4.14 | 619.21 | 8.30s | 1119.07 | 6.67 | 1118.01 | 8.09s |
| random (1000) | 1646.83 | 6.13 | 1202.66 | - | 8368.49 | 8.65 | 2897.99 | - |
| Kool et al. AM+TW(greedy) | 7615.69 | 2.0 | 1094.39 | 0.05s | 40245.4 | 2.0 | 2687.95 | 0.12s |
| Kool et al. AM+TW(sampl.) | 1572.31 | 6.56 | 1221.09 | 0.11s | 7712.35 | 9.34 | 2953.65 | 0.34s |
| Kool et al. AM+TW(t10240) | 1387.3 | 6.46 | 1170.49 | 0.90s | 6730.55 | 9.99 | 2954.76 | 2.70s |
| Falkner et al. JAMPR(greedy) | 674.72 | 4.32 | 626.8 | 0.11s | 1273.2 | 6.02 | 1126.74 | 0.25s |
| Falkner et al. JAMPR(sampl.) | 620.68 | 4.19 | 602.33 | 0.92s | 1116.76 | 5.64 | 1076.79 | 2.32s |
| TW3 | Falkner et al.  OR-Tools-AU | 1317.81 | 4.07 | 637.15 | 1.07s | 2707.72 | 6.12 | 1121.57 | 20.63s |
| Falkner et al.  OR-Tools-GLS | 1312.71 | 4.11 | 625.8 | 8.00s | 2753.66 | 6.18 | 1192.61 | 8.02s |
| random (1000) | 1409.35 | 3.34 | 953.48 | - | 4407.58 | 6.92 | 2692.78 | - |
| Continuation of table A.1 | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  | Kool et al. AM+TW(greedy) | 3101.21 | 2.0 | 1094.39 | 0.05s | 26467.34 | 2.0 | 2687.95 | 0.12s |
|  | Kool et al. AM+TW(sampl.) | 1412.16 | 3.42 | 951.92 | 0.12s | 4161.24 | 7.15 | 2674.03 | 0.34s |
| Kool et al. AM+TW(t10240) | 1318.56 | 3.23 | 899.83 | 0.88s | 3941.47 | 6.77 | 2575.28 | 2.67s |
| Falkner et al. JAMPR(greedy) | 1002.81 | 1.0 | 733.01 | 0.10s | 3158.26 | 2.01 | 1347.72 | 0.23s |
| Falkner et al. JAMPR(sampl.) | 844.35 | 1.39 | 660.48 | 0.78s | 1947.65 | 2.29 | 1358.29 | 2.13s |
| Notes:  1. TW1 - problems with hard time windows  2. TW2 - problem with soft constraint for upper bound  3. TW3 - problem with soft constraints for both upper and lower bounds  4. K is the average number of vehicles  5. Tinf is the time to generate a solution  6. Compiled from source [56; 75] | | | | | | | | | |

Table 2 – The performance comparison on CVRPLib

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instance | Depot | Customer  Type | Wu et al. (T=5k), % | Ma et al. (T=5k), % | OR Tools,  % | Kool et al. (N=10k), % | Kwon et al.  ×8 augment, % | Wu et al. (T=5k, M=100), % | Ma et al. (T=10k), % | Ma et al. ×6 augment, *%* |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| X-n101-k25 | R | R | 7.70 | 2.09 | 6.57 | 32.95 | 3.64 | 5.60 | 1.86 | *1.47* |
| X-n106-k14 | E | C | 4.86 | 2.93 | 3.72 | 6.78 | *1.85* | 2.83 | 2.75 | 1.87 |
| X-n110-k13 | C | R | 6.39 | 1.43 | 7.87 | 3.15 | 2.05 | 4.40 | 0.87 | *0.13* |
| X-n115-k10 | C | R | 13.32 | 3.29 | 4.50 | 7.52 | 3.49 | 5.19 | 3.26 | *1.68* |
| X-n120-k6 | E | RC | 16.16 | 3.50 | 6.83 | 4.54 | *2.12* | 5.56 | 3.20 | 2.38 |
| X-n125-k30 | R | C | 8.79 | 6.51 | 5.63 | 35.16 | 7.14 | *4.71* | 5.47 | 5.44 |
| X-n129-k18 | E | RC | 11.01 | 2.93 | 8.37 | 4.00 | *0.97* | 4.63 | 2.55 | 2.55 |
| X-n134-k13 | R | C | 16.06 | 6.98 | 21.61 | 20.13 | 4.22 | 8.88 | 5.56 | *2.63* |
| X-n139-k10 | C | R | 14.99 | 2.54 | 12.02 | 4.30 | 2.28 | 4.90 | 2.16 | *2.08* |
| X-n143-k7 | E | R | 20.20 | 7.80 | 11.27 | 8.88 | *2.79* | 6.61 | 6.47 | 3.55 |
| X-n148-k46 | R | RC | 16.38 | 2.69 | 7.80 | 79.53 | 19.88 | 3.60 | *2.22* | *2.22* |
| X-n153-k22 | C | C | 22.94 | 11.06 | 8.01 | 78.11 | 12.16 | *4.53* | 9.02 | 6.53 |
| X-n157-k13 | R | C | 17.15 | 4.64 | *2.57* | 16.30 | 2.79 | 3.60 | 4.44 | 3.12 |
| X-n162-k11 | C | RC | 19.16 | 4.43 | 6.31 | 6.37 | 4.77 | 5.26 | 3.04 | *2.62* |
| X-n167-k10 | E | R | 18.52 | 5.37 | 9.34 | 8.41 | 4.05 | 8.27 | 4.28 | *3.47* |
| X-n172-k51 | C | RC | 12.06 | 6.23 | 10.74 | 85.37 | 21.99 | 4.36 | 5.27 | *3.41* |
| X-n176-k26 | E | R | 19.49 | 10.29 | 8.99 | 20.39 | 10.27 | 6.16 | 8.07 | *5.93* |
| X-n181-k23 | R | C | 6.27 | 3.41 | 2.94 | 6.45 | *2.08* | *2.08* | 2.42 | *2.08* |
| X-n186-k15 | R | R | 17.71 | 5.99 | 7.75 | 6.01 | *2.15* | 7.65 | 5.30 | 4.94 |
| X-n190-k8 | E | C | 18.64 | 7.97 | *6.53* | 46.61 | 9.25 | 6.78 | 6.73 | 6.73 |
| X-n195-k51 | C | RC | 17.04 | 7.00 | 13.76 | 79.26 | 9.23 | 4.47 | 4.54 | *4.36* |
| X-n200-k36 | R | C | 9.60 | 5.93 | *4.15* | 26.25 | 5.01 | 4.26 | 5.87 | 5.86 |
| X-n129-k18 | E | RC | 11.01 | 2.93 | 8.37 | 4.00 | *0.97* | 4.63 | 2.55 | 2.55 |
| X-n134-k13 | R | C | 16.06 | 6.98 | 21.61 | 20.13 | 4.22 | 8.88 | 5.56 | *2.63* |
| Continuation of table A.2 | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| X-n139-k10 | C | R | 14.99 | 2.54 | 12.02 | 4.30 | 2.28 | 4.90 | 2.16 | *2.08* |
| X-n143-k7 | E | R | 20.20 | 7.80 | 11.27 | 8.88 | *2.79* | 6.61 | 6.47 | 3.55 |
| X-n148-k46 | R | RC | 16.38 | 2.69 | 7.80 | 79.53 | 19.88 | 3.60 | *2.22* | *2.22* |
| X-n153-k22 | C | C | 22.94 | 11.06 | 8.01 | 78.11 | 12.16 | *4.53* | 9.02 | 6.53 |
| X-n157-k13 | R | C | 17.15 | 4.64 | *2.57* | 16.30 | 2.79 | 3.60 | 4.44 | 3.12 |
| X-n162-k11 | C | RC | 19.16 | 4.43 | 6.31 | 6.37 | 4.77 | 5.26 | 3.04 | *2.62* |
| X-n167-k10 | E | R | 18.52 | 5.37 | 9.34 | 8.41 | 4.05 | 8.27 | 4.28 | *3.47* |
| X-n172-k51 | C | RC | 12.06 | 6.23 | 10.74 | 85.37 | 21.99 | 4.36 | 5.27 | *3.41* |
| X-n176-k26 | E | R | 19.49 | 10.29 | 8.99 | 20.39 | 10.27 | 6.16 | 8.07 | *5.93* |
| X-n181-k23 | R | C | 6.27 | 3.41 | 2.94 | 6.45 | 2.08 | 2.08 | 2.42 | *2.08* |
| X-n186-k15 | R | R | 17.71 | 5.99 | 7.75 | 6.01 | *2.15* | 7.65 | 5.30 | 4.94 |
| X-n190-k8 | E | C | 18.64 | 7.97 | *6.53* | 46.61 | 9.25 | 6.78 | 6.73 | 6.73 |
| X-n195-k51 | C | RC | 17.04 | 7.00 | 13.76 | 79.26 | 9.23 | 4.47 | 4.54 | *4.36* |
| X-n200-k36 | R | C | 9.60 | 5.93 | *4.15* | 26.25 | 5.01 | 4.26 | 5.87 | 5.86 |
| Avg. Gap for [100,150) | | | 12.35 | 3.88 | 8.74 | 18.81 | 4.58 | 5.17 | 3.31 | *2.36* |
| Avg. Gap for [150,200] | | | 16.24 | 6.57 | 7.37 | 34.50 | 7.61 | 5.22 | 5.36 | *4.46* |
| Avg. Gap for all | | | 14.29 | 5.23 | 8.06 | 26.66 | 6.10 | 5.20 | 4.33 | *3.41* |
| Notes – Compiled from source [[56](#_heading=h.3s49zyc); 68, p. 21188-21197; 101, p. 5057-5068; 102, p. 170-180] | | | | | | | | | | |

Table 3 – The performance comparison on TSPLib

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instance | Wu et al. (T=3k), % | Ma et al., DACT (T=3k), % | OR Tools, % | Kool et al. (N=10k), % | Kwon et al., POMO x8 augment, % | Wu et al. (T=3k, M=1k), % | Ma et al. (T=10k), % | Ma et al. x4 augment, % | da Costa et al., % | Kim et al., AM + LCP, % |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| eil51 | 2.82 | 1.64 | 2.35 | 2.11 | *0.00* | 1.17 | *0.00* | *0.00* | 0.23 | 0.73 |
| berlin52 | 6.34 | 0.03 | 5.34 | 1.67 | *0.03* | 2.57 | *0.03* | *0.03* | 5.73 | 0.10 |
| st70 | 4.59 | 0.44 | 1.19 | 2.22 | *0.30* | 0.89 | *0.30* | *0.30* | 0.74 | 0.74 |
| eil76 | 6.88 | 2.42 | 4.28 | 3.35 | *1.49* | 4.65 | 2.04 | 1.67 | 2.60 | 1.64 |
| pr76 | 1.40 | 1.02 | 2.72 | 2.84 | 19.97 | 1.37 | *0.03* | *0.03* | 2.60 | 0.44 |
| rat99 | 17.18 | 4.05 | 1.73 | 9.50 | 7.51 | 8.51 | 1.16 | *0.74* | 14.62 | 6.67 |
| KroA100 | 18.39 | 0.86 | 0.78 | 79.49 | 4.45 | 2.08 | 0.63 | *0.45* | 11.60 | 2.95 |
| KroB100 | 19.97 | 0.27 | 3.91 | 9.30 | 5.83 | 5.78 | *0.25* | *0.25* | 7.45 | 1.51 |
| KroC100 | 22.14 | 1.06 | 4.02 | 8.04 | 6.55 | 3.17 | *0.84* | *0.84* | 9.27 | 2.84 |
| KroD100 | 16.33 | 3.54 | 1.61 | 10.02 | 8.74 | 5.00 | 3.54 | *0.12* | 9.58 | 1.97 |
| KroE100 | 21.91 | 2.17 | 2.40 | 3.10 | 5.97 | 3.29 | 1.95 | *0.32* | 5.37 | 1.90 |
| rd100 | 0.06 | 0.08 | 3.53 | 1.93 | 0.00 | 0.06 | 0.06 | *0.00* | 0.43 | 0.13 |
| eil101 | 4.61 | 3.66 | 5.56 | 3.97 | *2.07* | 4.61 | 3.66 | 2.86 | 0.95 | 2.59 |
| lin105 | 26.53 | 3.41 | 3.09 | 32.13 | 12.00 | 2.48 | 3.35 | *0.69%* | 12.36 | 3.86 |
| pr107 | 19.76 | 5.86 | *1.74* | 43.26 | 5.66 | 3.87 | 5.01 | 3.81 | - | - |
| pr124 | 11.82 | 1.56 | 5.91 | 4.41 | *0.29* | 2.97 | 1.22 | 1.22 | 0.82 | 3.84 |
| bier127 | 20.65 | 4.08 | 3.76 | *1.71* | 60.56 | 3.48 | 3.79 | 2.46 | 2.40 | 8.92 |
| ch130 | 16.53 | 6.63 | 2.85 | 2.96 | *0.25* | 4.89 | 5.48 | 1.93 | 1.06 | 0.57 |
| pr136 | 9.14 | 5.54 | 5.62 | 4.90 | *1.06* | 6.33 | 5.14 | 4.54 | 1.74 | 1.56 |
| pr144 | 21.30 | 3.44 | 1.28 | 8.77 | *0.80* | 1.40 | 3.44 | 2.49 | 4.56 | 3.47 |
| ch150 | 21.26 | 3.60 | 3.08 | 3.45 | *0.83* | 3.55 | 3.45 | 1.23 | - | - |
| KroA150 | 17.80 | 6.93 | 4.03 | 9.98 | 13.15 | 4.51 | 3.91 | 3.91 | 13.40 | *3.68* |
| KroB150 | 20.20 | 6.10 | 5.52 | 9.87 | 11.72 | 5.40 | 4.10 | *2.82* | 7.80 | 3.18 |
| pr152 | 16.20 | 4.48 | 2.92 | 13.47 | 4.11 | *2.17* | 3.59 | 3.59 | 2.20 | 2.52 |
| u159 | 21.97 | 6.84 | 8.79 | 7.38 | 2.19 | 7.67 | 5.86 | 3.16 | *1.51* | 10.84 |
| rat195 | 25.40 | 6.93 | *2.84* | 16.57 | 29.06 | 9.90 | 5.81 | 4.99 | 27.21 | 10.81 |
| Continuation of table A.3 | | | | | | | | | | |
| 1 | 2 | 3 | *4* | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| d198 | 13.83 | 12.27 | *1.16* | 331.58 | 45.98 | 4.99 | 10.74 | 8.75 | - | - |
| KroA200 | 22.44 | 3.60 | 1.27 | 15.64 | 20.00 | 7.01 | *1.52* | 1.25 | 10.74 | 6.14 |
| KroB200 | 23.69 | 10.51 | *3.67* | 18.54 | 21.06 | 7.05 | 6.28 | 5.66 | - | - |
| Notes – Compiled from source [[56](#_heading=h.3s49zyc); 68, p. 21188-21197; 101, p. 5057-5068; 102, p. 170-180; 130] | | | | | | | | | | |

Table 4 – The performance comparison to solve TSP on random instances used in\*. We used the pre-trained models from the respective GitHub repositories

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Method | TSP20 | | | TSP50 | | | TSP100 | | |
| obj | gap, % | time | obj | gap, % | time | obj | gap, % | time |
| Concorde | 3.84 | 0.00 | 11s | 5.69 | 0.00 | 44s | 7.75 | 0.00 | 3m24s |
| LKH | 3.84 | 0.00 | 31s | 5.69 | 0.00 | 7m8s | 7.75 | 0.00 | 31m3s |
| Kool et al., AM(N=1,280) | 3.85 | 0.05 | 18s | 5.71 | 0.45 | 1m6s | 7.93 | 2.26 | 3m11s |
| Kool et al., AM(N=5,000) | 3.85 | 0.05 | 33s | 5.71 | 0.38 | 2m40s | 7.91 | 2.08 | 9m13s |
| Kim et al., AM+LCP {640,10} | 3.85 | *0.02* | 2m10s | 5.69 | 0.13 | 3m47s | 7.83 | 1.01 | 7m13s |
| Kim et al., AM+LCP {1280,10} | –1 | –1 | –1 | 5.69 | 0.09 | 6m16s | 7.82 | 0.88 | 12m50s |
| Kim et al., AM+LCP\* {1280,45} | –1 | –1 | –1 | 5.7 | 0.16 | 28m48s | 7.87 | 2.43 | 29m3s |
| Kwon et al., POMO | 3.84 | *0.00* | 23s | 5.69 | 0.10 | 45s | 7.78 | 0.38 | 1m30s |
| Kwon et al., POMO x8 augment | 3.84 | *0.00* | 19s | 5.69 | 0.02 | 46.3s | 7.76 | 0.13 | 1m54s |
| Xin et al.,  MDAM-BS | 3.85 | *0.01* | 1hr 4m | 5.7 | 0.20 | 2hr33m | 7.8 | 0.64 | 4hr59m |
| Ma et al., DACT (T=1k) | 3.84 | 0.02 | 16.7s2 | 5.7 | 0.16 | 43s2 | 7.91 | 2.04 | 2m7s2 |
| Ma et al., DACT (T=5k) | 3.84 | *0.00* | 1m22s2 | 5.69 | 0.02 | 3m29s2 | 7.81 | 0.69 | 10m37s2 |
| Ma et al., DACT (T=10k) | 3.84 | *0.00* | 2m48s2 | 5.69 | 0.01 | 7m4s2 | 7.79 | 0.44 | 21m22s2 |
| Ma et al., DACTx4 augment | 3.84 | *0.00* | 8m25s2 | 5.69 | *0.00* | 25m2 | 7.76 | *0.09* | 1hr25m2 |
| Wu et al. (T=1,000) | –1 | –1 | –1 | 5.73 | 0.81 | 1m7s2 | 8.0 | 3.31 | 1m34s2 |
| Wu et al. (T=3,000) | –1 | –1 | –1 | 5.7 | 0.29 | 3m23s2 | 7.9 | 2.00 | 4m37s2 |
| Wu et al. (T=5,000) | –1 | –1 | –1 | 5.7 | 0.18 | 5m29s2 | 7.87 | 1.55 | 6m |
| Hottung et al. | 3.84 | *0.00* | 2hr29m | 5.69 | 0.03 | 5hr37m | 7.78 | 0.34 | 17h13m |
| da Costa et al. | 3.84 | *0.00* | 3hr4m | 5.69 | 0.12 | 3hr31m | 7.81 | 0.76 | 4hr22m |
| \* – Compiled from source [[56](#_heading=h.3s49zyc)]  Notes  1. 1 The pretrained models are not shared in the corresponding github repositories.  2. 2 The batch size is set to 1,000 to record running times  3. Compiled from source [[56](#_heading=h.3s49zyc); 68, p. 21188-21197; 101, p. 5057-5068; 102, p. 170-180; 130] | | | | | | | | | |