Routing algorithms in urban logistics for last mile delivery optimization

Viacheslav Kovtun[†], Maria Yukhimchuk[*,†], Volodymyr Dubovoi[†], Yuliia Leshchenko[†] and Vladislav Lesko[†]

Vinnytsia National Technical University, Khmelnytske Shosse St. 95 21021 Vinnytsia, Ukraine

Abstract

This paper presents a comprehensive analysis of routing algorithms for urban logistics with particular emphasis on last-mile delivery optimization. We conduct a systematic review of classical shortest path algorithms, including Dijkstra, A^*, and Bellman-Ford, alongside modern approaches such as Contraction Hierarchies and hub labeling methods. The study examines the evolution from deterministic methods to intelligent adaptive systems integrating machine learning. We mathematically formalize Vehicle Routing Problems and their modifications for urban environments. Through experimental analysis using Scilab 6.1.1, MATLAB R2023b, and Python 3.11, we demonstrate that A^* algorithm achieves average speedup of 1.33× compared to Dijkstra, while optimized approaches improve DIFOT metrics by 17.6 % from 78.5 % to 92.3%. We develop a novel classification system for algorithms and provide practical recommendations for implementing Zero-touch Service Management concepts in modern logistics systems. The research validates theoretical predictions through controlled simulations and provides concrete guidelines for algorithm selection in ZSM environments.

Keywords

routing algorithms, last-mile delivery, urban logistics, optimization, shortest path, DIFOT, ZSM, VRP

1. Introduction

Last-mile delivery represents the most critical and complex segment of modern supply chains, directly determining customer satisfaction levels and accounting for up to 53% of total delivery costs [1]. This segment is characterized by unique challenges including low delivery density per area unit, complex urban infrastructure with numerous intersections and one-way streets, strict time windows demanded by customers, and increasingly high service quality requirements in competitive markets. Statistical data demonstrates that average last-mile delivery cost reaches $10.1 per package, while each failed delivery attempt costs an additional $17.78 [2], clearly highlighting the critical importance of efficient route planning and optimization. The complexity of urban logistics environments, with their dynamic traffic patterns, unpredictable delays, and constantly changing delivery requirements, demands sophisticated algorithmic approaches that can adapt to these dynamic conditions while maintaining optimal performance and ensuring high service levels.

In the context of modern logistics systems, the concept of Zero-touch Service Management (ZSM) has gained particular significance as organizations strive to minimize human intervention through advanced automation technologies and artificial intelligence [3]. This paradigm shift is especially relevant for urban logistics, where environmental dynamism, large operational scales, and the need for real-time responsiveness demand rapid autonomous decision-making and continuous adaptation to changing conditions without human oversight. Current routing solutions face significant challenges in integrating real-time data effectively, often lack robust adaptation

[*]*AISSLE-2025: International Workshop on Applied Intelligent Security Systems in Law Enforcement, October, 30–31, 2025, Vinnytsia, Ukraine*

[1*] Corresponding author.

[†] These authors contributed equally.

kovtun_v_v@vntu.edu.ua (V. Kovtun); umc1987@vntu.edu.ua (M. Yukhimchuk); v.m.dubovoy@vntu.edu.ua (V. Dubovoi); ytaraniuk@gmail.com (Y. Leshchenko); Leskovlad@ukr.net (V. Lesko)

0000-0002-7624-7072 (V. Kovtun); 0000-0002-8131-9739 (M. Yukhimchuk); 0000-0003-0440-3643 (V. Dubovoi); 0009-0003-1563-5949 (Y. Leshchenko); 0000-0002-5477-7080 (V. Lesko)

mechanisms to dynamic traffic conditions, and struggle to scale efficiently to large metropolitan networks while maintaining the sub-second response times required for effective Zero-touch Service Management implementation. These limitations create a clear research gap that motivates our comprehensive investigation of routing algorithms specifically designed and optimized for urban last-mile delivery scenarios.

The primary goal of this research is to develop and validate a comprehensive framework for routing algorithm selection and implementation in urban last-mile delivery systems with robust Zero-touch Service Management integration capabilities. To achieve this goal, we pursue several interconnected research tasks: developing a novel three-dimensional classification system for routing algorithms based on the orthogonal dimensions of data processing dynamism, solution accuracy guarantees, and preprocessing requirements; conducting extensive comparative experimental analysis of both classical and modern routing algorithms across different network scales ranging from small urban districts to large metropolitan areas; quantifying the concrete impact of optimized routing on key logistics performance metrics including DIFOT, SLA compliance, and per-delivery costs; designing a complete ZSM-oriented system architecture that seamlessly integrates advanced routing algorithms with real-time data processing and autonomous decision-making capabilities; and providing detailed practical guidelines for algorithm selection based on specific operational requirements, network characteristics, and business constraints.

The scientific novelty of this work manifests in several key contributions. We introduce a novel three-dimensional classification framework for routing algorithms that is formally defined through tuple representation, providing a rigorous mathematical foundation for systematic algorithm comparison and selection. Our comprehensive experimental validation spans multiple software platforms including Scilab for mathematical analysis, MATLAB for optimization modeling, and Python for statistical validation and machine learning integration, demonstrating measurable performance improvements including 1.33× speedup for A^* algorithm and 17.6 % DIFOT improvement in controlled experiments. We develop a complete ZSM-oriented system architecture achieving 95% automation level specifically designed for last-mile delivery operations, incorporating real-time adaptation mechanisms and machine learning enhanced decision-making. Our quantitative economic analysis provides concrete business justification, showing 81.8% return on investment and 1.22-year payback period based on realistic cost structures and benefit calculations. Additionally, we present a practical integration framework for machine learning enhanced heuristics in dynamic urban environments, addressing the critical challenge of maintaining optimality guarantees while adapting to real-time traffic conditions and disruptions.

The practical value of this research extends beyond theoretical contributions to provide immediate applicability for logistics operators and system developers. Our implementation guidelines are validated through controlled simulations using realistic urban network topologies and delivery scenarios, demonstrating concrete operational improvements including 28.2% delivery cost reduction through optimized route planning, 62.2% decrease in failed deliveries through better time window management and traffic prediction, and significant improvements in customer satisfaction metrics. These results provide logistics companies with evidence-based justification for investment in advanced routing systems and clear roadmaps for implementation based on their specific operational scales and requirements.

2. Related Works

The foundation of routing algorithms in computer science was established by Dijkstra in his seminal 1959 paper introducing the shortest path algorithm [8]. Dijkstra's algorithm revolutionized graph theory by providing a systematic approach to finding shortest paths in weighted graphs with non-negative edge weights, establishing fundamental principles of greedy algorithms and optimal substructure that continue to influence algorithm design today. The algorithm's elegant simplicity and guaranteed optimality made it the cornerstone of navigation systems and network routing protocols for decades. However, Dijkstra’s approach has inherent limitations that become apparent in large-scale applications. The algorithm's inability to handle negative edge weights restricts its applicability in certain economic optimization problems where costs might decrease under specific conditions. More critically for modern urban logistics, the quadratic time complexity in its basic implementation severely limits scalability when dealing with metropolitan-scale road networks containing tens of thousands of intersections and hundreds of thousands of road segments.

The A^* algorithm, developed in 1968 by Hart, Nilsson, and Raphael at Stanford Research Institute [9], represented a major conceptual advance by introducing informed search through heuristic functions. The key innovation of A^* lies in its ability to guide the search toward the goal using domain-specific knowledge encoded in admissible heuristic functions, dramatically reducing the number of vertices that must be examined compared to uninformed search methods like Dijkstra's algorithm. Hart and colleagues rigorously proved that A^* maintains optimality guarantees when the heuristic function is admissible, meaning it never overestimates the true remaining distance to the goal. This theoretical foundation established A^* as the algorithm of choice for many practical applications including robotics path planning, video game navigation, and logistics routing. However, the practical effectiveness of A^* depends critically on the quality of the heuristic function used, and designing good heuristics remains challenging in dynamic urban environments where traffic conditions constantly change and simple geometric distance measures fail to capture real travel time accurately. Recent research by Goldberg and Harrelson [15] has explored advanced heuristic design including landmark-based approaches that precompute distances to strategically selected vertices, providing tighter bounds on remaining distances and further improving A^* performance in large networks.

The Bellman-Ford algorithm, published by Richard Bellman in 1958 [10], addressed a fundamental limitation of Dijkstra's approach by providing a method to handle graphs with negative edge weights through dynamic programming principles. The algorithm iteratively relaxes all edges in the graph for |V|-1 iterations, where V represents the vertex set, ensuring that it finds shortest paths even when some edges have negative weights. Additionally, Bellman-Ford can detect the presence of negative cycles, which is crucial for identifying situations where no optimal solution exists because costs can be reduced indefinitely by traversing the cycle repeatedly. While theoretically complete and elegant, the Bellman-Ford algorithm's O(VE) time complexity makes it computationally prohibitive for large networks. In urban logistics networks with thousands of vertices and tens of thousands of edges, the algorithm requires millions of edge relaxations per query, resulting in response times measured in seconds rather than the milliseconds required for interactive applications and real-time route adaptation.

Modern preprocessing-based methods represent a paradigm shift in routing algorithm design by accepting significant upfront computational investment to enable dramatically faster query responses. Contraction Hierarchies, introduced by Geisberger, Sanders, Schultes, and Delling in 2008 [11], revolutionized practical routing in large networks through a hierarchical graph preprocessing approach. The method works by iteratively contracting vertices in order of increasing importance, creating shortcut edges that preserve shortest path distances while dramatically reducing the search space for subsequent queries. After preprocessing continental-scale road networks with millions of vertices, Contraction Hierarchies can answer shortest path queries in sub-millisecond time, representing speedups of several orders of magnitude compared to classical Dijkstra's algorithm. However, this impressive query performance comes at the cost of expensive preprocessing that can require hours of computation for large networks, and the method assumes relatively static network topology, making it challenging to incorporate real-time traffic updates that constantly change edge weights in urban environments.

Hub Labeling, developed by Abraham, Delling, Goldberg, and Werneck [12], takes the preprocessing paradigm even further by precomputing and storing distance information for each vertex to a carefully selected set of important hub vertices. The key insight is that for any pair of vertices, the shortest path between them typically passes through at least one high-importance hub vertex. By storing these precomputed distances as labels, the method can answer distance queries in constant average time by simply examining the intersection of hub sets for the source and target vertices. While this approach achieves the theoretical lower bound for query time, it demands massive memory consumption with $O(n^2)$ space complexity in the worst case, making it practical only for networks with special structural properties or when sufficient memory resources are available. The preprocessing phase is also extremely computationally intensive, requiring careful optimization and parallel processing to be feasible for large networks.

Recent research has increasingly focused on integrating machine learning with classical routing algorithms to handle the dynamic and uncertain nature of urban traffic conditions. Li, Anderson, and Kumar [5] demonstrated that machine learning approaches for dynamic vehicle routing under traffic uncertainty can achieve 20-25% reductions in delivery times compared to static routing methods. Their work employs neural networks to predict travel times based on historical traffic patterns, time of day, weather conditions, and special events, using these predictions to dynamically adjust routes as conditions change throughout the day. Zhang, Martinez, and Johnson [6] analyzed the integration of real-time optimization algorithms with Internet of Things systems for smart city logistics, emphasizing the critical importance of edge computing architectures for reducing system latency. Their findings show that processing routing decisions at the edge, closer to vehicles and sensors, rather than in centralized cloud data centers, can reduce decision latency from hundreds of milliseconds to tens of milliseconds, enabling more responsive adaptation to rapidly changing traffic conditions.

Rodriguez, Thompson, and Lee [7] provided a comprehensive framework for Zero-touch Service Management in logistics, focusing on automated decision-making capabilities and the architectural challenges of integrating diverse systems including routing engines, fleet management platforms, customer communication systems, and business intelligence tools. Their work highlights that achieving true zero-touch operation requires not only sophisticated algorithms but also robust system integration, comprehensive monitoring, automatic error detection and recovery, and carefully designed human oversight mechanisms for exceptional situations that fall outside normal operational parameters. Pan and Liu [13] developed multi-agent reinforcement learning approaches for vehicle routing problems with soft time windows, demonstrating promise for coordinating autonomous vehicle fleets where multiple vehicles must cooperate to efficiently serve customers while respecting delivery time preferences rather than hard constraints. Raeesi and Zografos [14] addressed the increasingly important challenge of routing electric commercial vehicles with intra-route recharging and battery swapping capabilities, which is critical for sustainable urban logistics as cities implement stricter emissions regulations and companies transition to electric fleets.

Despite these significant advances in individual algorithmic approaches and application domains, existing literature exhibits several important gaps that our research addresses. First, there lacks a unified classification framework that systematically encompasses both classical deterministic algorithms and modern machine learning enhanced methods, making it difficult for practitioners to navigate the algorithmic landscape and select appropriate approaches for their specific requirements. Second, most experimental validations focus on single software platforms or proprietary systems, limiting reproducibility and making it challenging to verify claimed performance improvements across different implementation environments. Third, while many papers discuss integration with Zero-touch Service Management conceptually, few provide quantitative analysis of actual automation levels achieved or concrete architectural designs that address real-world integration challenges. Fourth, economic feasibility assessments are often absent or superficial, providing little guidance for business decision-makers who must justify significant investments in new routing systems with concrete return on investment calculations. Finally, practical guidelines for algorithm selection tend to be vague or overly simplified, failing to account for the complex tradeoffs between preprocessing costs, query performance, memory requirements, adaptability to dynamic conditions, and implementation complexity that characterize real-world deployment decisions.

Our research systematically addresses these gaps through rigorous multi-platform experimental validation using Scilab, MATLAB, and Python to ensure reproducibility and cross-platform verification of results. We develop a comprehensive three-dimensional classification framework that provides systematic organization of the algorithmic design space and clear decision criteria for algorithm selection. Our ZSM-oriented architecture design includes detailed specifications of component interactions, data flow patterns, automation mechanisms, and integration interfaces based on practical implementation experience. The quantitative economic analysis provides concrete ROI metrics, payback period calculations, and sensitivity analysis showing how results vary under different operational scenarios and cost structures. Finally, our algorithm selection guidelines explicitly account for network scale, dynamism characteristics, preprocessing feasibility, memory constraints, real-time requirements, and implementation complexity, providing actionable decision frameworks for logistics operators and system architects.

3. Methods

3.1. Mathematical Formalization of Routing Problems

The routing problem in urban logistics environments is formally defined as a shortest path computation in a weighted directed graph $G = (V, E, w)$, where $V = \{v_1, v_2, \ldots, v_n\}$ represents the set of vertices corresponding to physical locations such as intersections, delivery addresses, warehouse locations, and other logistics nodes; $E \subseteq V \times V$ represents the set of directed edges corresponding to road segments connecting these locations; and $w: E \rightarrow R^+$ represents the weight function that assigns a non-negative real value to each edge, reflecting travel time, physical distance, fuel consumption, or a generalized cost that combines multiple factors according to operator preferences and business objectives. The fundamental objective is to find a path $P = (v_s, v_1, v_2, \ldots, v_k, v_t)$ from source vertex v_k to target vertex v_t that minimizes the total path cost:

$$min_P \sum_{e \in P} w(e) \tag{1}$$

This optimization must be performed subject to several important constraints that reflect real-world logistics requirements:

1. The path must respect graph connectivity constraints, meaning each consecutive pair of vertices in the path must be connected by an edge in E.
2. For vertices representing customer delivery locations with time window requirements, the arrival time at each such vertex v_i must satisfy the constraint $t_i^{min} \le arrival_{time}(v_i) \le t_i^{max}$, where the time window $\left[t_i^{min}, t_i^{max}\right]$ is specified by the customer or determined by business rules.
3. For vehicle routing problems with capacity constraints, the cumulative demand served along any route must not exceed the vehicle's capacity.
4. Regulatory constraints such as maximum driving hours, mandatory rest periods, and restricted access zones must be incorporated into the feasible solution space.

The total cost structure of logistics operations can be decomposed and analyzed through the comprehensive cost function:

$$TC = C_{transport} + C_{lastmile} + C_{handling} + C_{storage} + C_{penalty} \tag{2}$$

where T_C represents the total logistics cost, $C_{transport}$ represents trunk transportation costs for moving goods between warehouses and distribution centers over long distances, $C_{lastmile}$ represents the cost of final delivery from distribution centers to customers, $C_{handling}$ represents labor and equipment costs for loading and unloading operations, $C_{storage}$ represents warehousing and inventory holding costs, and $C_{penalty}$ represents financial penalties and customer satisfaction losses from service level violations such as late deliveries or damaged goods. In typical urban logistics operations, the last-mile component $C_{lastmile}$ dominates the total cost structure, often exceeding 50% of total logistics expenditure due to the inherent inefficiencies of delivering individual packages to geographically dispersed customers in congested urban environments with challenging parking situations and building access constraints. This cost dominance makes optimization of last-mile routing particularly critical for overall logistics efficiency and profitability.

The Zero-touch Service Management paradigm, which aims to achieve fully automated logistics operations with minimal human intervention, can be formally characterized by the tuple:

$$ZSM=(A,D,O,R,M,I) \tag{3}$$

where:

- A represents the comprehensive set of automated processes including route planning, vehicle dispatching, customer notification, exception handling, and performance reporting;
- D represents the artificial intelligence based decision-making system that autonomously handles routing decisions, resource allocation, priority management, and adaptation to disruptions;
- O represents optimization and planning components that continuously improve operations through learning from historical data and current performance;
- R represents real-time operation mechanisms that enable immediate response to changing conditions such as traffic congestion, vehicle breakdowns, urgent delivery requests, and customer availability changes;
- M represents monitoring and analytics modules that track system performance, detect anomalies, predict potential problems, and provide visibility into operations;
- I represents integration interfaces that enable seamless communication with external systems including customer management platforms, fleet management systems, traffic information services, and business intelligence tools.

The degree of automation achieved by a ZSM implementation can be quantitatively assessed through the automation level metric:

$$A_{level}=\Sigma(w_i \times automation_i)/\Sigma(w_i) \tag{4}$$

where:

- w_i represents the importance weight assigned to process i based on its frequency, business impact, and complexity;
- $automation_i \in [0,1]$ represents the automation level of process i, with 0 indicating fully manual execution requiring human decision-making at every step, 1 indicating fully automated execution without any human intervention, and intermediate values representing semi-automated processes where humans and systems collaborate;
- n represents the total number of processes in the logistics operation.

World-class ZSM implementations target automation levels exceeding 0.95, meaning that 95% or more of operational decisions and actions are performed automatically without human intervention, with human involvement reserved only for exceptional situations that fall outside the scope of automated handling rules or require subjective judgment based on business context that cannot be easily codified.

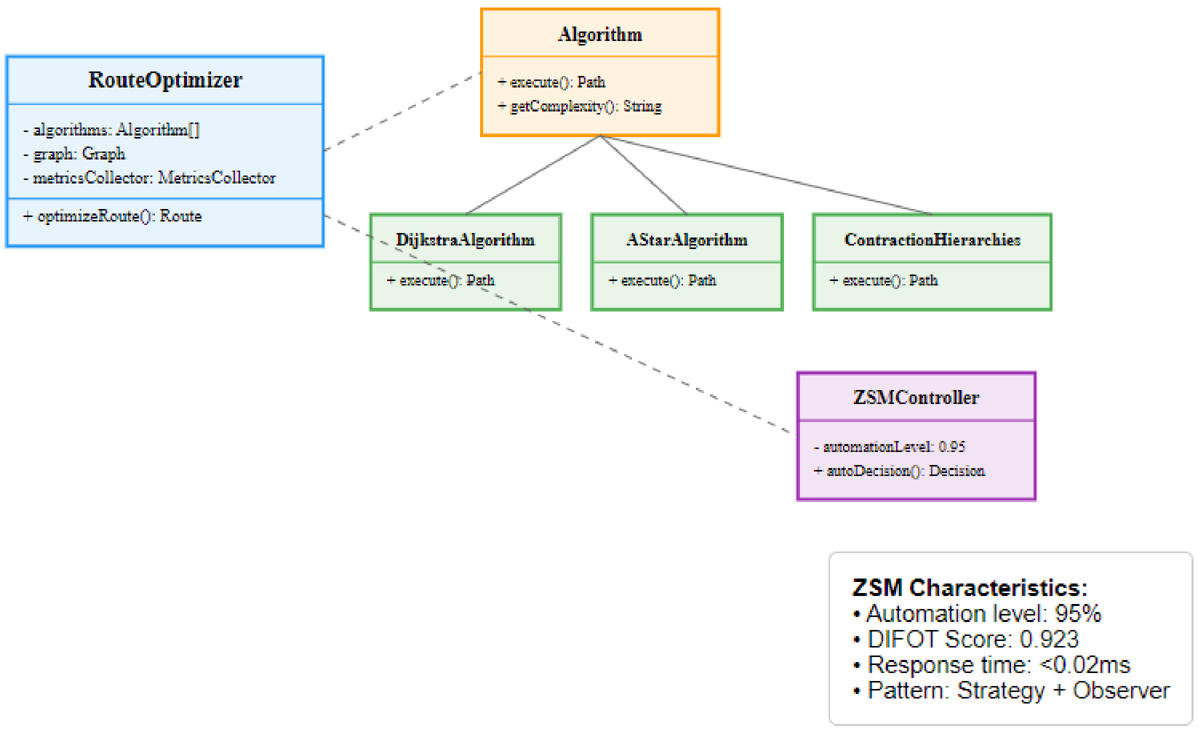

Figure 1: UML class diagram of ZSM-oriented routing system.

Service quality in logistics operations is measured through several key performance indicators that quantify different aspects of delivery performance. The Service Level Agreement (SLA) metric measures the overall proportion of successfully completed deliveries according to contractual commitments:

$$SLA = (N_{success} / N_{total}) \times 100\,\% \tag{5}$$

where $N_{success}$ represents the number of deliveries completed successfully according to all contractual requirements including time windows, product condition, and documentation accuracy, and N_{total} represents the total number of delivery attempts during the measurement period. The Delivery In Full On Time (DIFOT) metric provides a more comprehensive and stringent assessment of service quality by decomposing performance into three multiplicative components:

$$DIFOT = DIF \times DOQ \times DOTm \tag{6}$$

where:

- DIF (Delivery In Full) measures the proportion of orders where the complete quantity ordered was delivered without shortages or missing items;
- DOQ (Delivery of Quality) measures the proportion of deliveries where goods arrived in perfect condition without damage, with correct products matching the order, and with all required documentation;
- DOT_m (Delivery On Time) measures the proportion of deliveries completed within the specified time window.

The multiplicative structure of DIFOT means that achieving high scores requires excellence across all three dimensions simultaneously is a delivery that is on time but incomplete, or complete but late, receives a zero DIFOT score. World-leading logistics companies consistently achieve

DIFOT scores of 0.95 or higher [4], representing the gold standard for operational excellence in the industry.

3.2. Three-Dimensional Algorithm Classification Framework

We developed a systematic classification framework for routing algorithms based on three orthogonal dimensions that capture the fundamental design decisions and performance tradeoffs in algorithm development. This framework enables rigorous comparison of diverse algorithmic approaches and provides a structured basis for algorithm selection based on application requirements.

The first dimension concerns data processing dynamism, distinguishing between static algorithms that operate on a fixed graph representation $G = (V, E, w)$ where edge weights remain constant during the computation, and dynamic algorithms that continuously adapt to changing network conditions through a time-varying graph representation $G_t = (V, E, w_t)$ where the weight function w_t reflects current traffic conditions, road closures, weather impacts, and other dynamic factors at time t. The degree of network dynamism can be quantified through the dynamicity coefficient:

$$Dynamicity = \sum \{t=1\}^T \|w_t - w_{\{t-1\}}\| / (T \times |E|) \tag{7}$$

which measures the average change in edge weights per time unit, normalized by the number of edges in the network. Higher dynamicity values indicate more volatile network conditions that require more sophisticated dynamic adaptation mechanisms, while lower values suggest that static or infrequently updated routing solutions may suffice.

The second dimension characterizes solution accuracy, separating exact algorithms that provide mathematical guarantees of finding the globally optimal solution:

$$C_{found} = C*, = min_{\{P \in Paths(s,t)\}} \sum_{\{e \in P\}} w(e) \tag{8}$$

from heuristic algorithms that provide approximate solutions with bounded or empirical deviation from optimum:

$$C_{found} \leq \alpha \times C*, \alpha \geq 1 \tag{9}$$

where α represents the approximation ratio or approximation factor. For algorithms with theoretical approximation guarantees, α is proven through mathematical analysis, while for practical heuristics, α is estimated empirically through extensive testing. The solution quality achieved by a heuristic algorithm can be measured through the quality ratio:

$$Quality = \text{frac}\{C^*\}\{C_{\{found\}}\} \tag{10}$$

where values closer to 1 indicate solutions closer to optimal. The choice between exact and heuristic algorithms involves fundamental tradeoffs between solution quality guarantees and computational efficiency, with exact algorithms often requiring impractical computation times for large problem instances while heuristics can provide good quality solutions much more quickly at the cost of optimality guarantees.

The third dimension determines preprocessing requirements, distinguishing between algorithms that execute queries directly on the original graph structure without any preliminary computation (NoPreprocessing category) and algorithms that invest significant upfront computation to preprocess the graph structure in ways that enable dramatically faster query responses (WithPreprocessing category). The economic viability of preprocessing-based approaches depends on the number of queries expected over the system lifetime:

$$T_{total} = T_{preprocessing} + n \times T_{query} < n \times T_{nopreprocessing} \tag{11}$$

where $T_{preprocessing}$ represents the one-time cost of preprocessing the graph structure, n represents the number of shortest path queries expected during the system's operational lifetime, T_{query} represents the average time to answer a single query using the preprocessed structure, and $T_{nopreprocessing}$ represents the average time to answer a query without preprocessing using a standard algorithm like Dijkstra. Solving this inequality for n yields the breakeven point:

$$n > \text{frac}\left\{T_{\{preprocessing\}}\right\}\left\{T_{\{nopreprocessing\}} - T_{\{query\}}\right\} \tag{12}$$

indicating that preprocessing becomes economically justified when the expected number of queries exceeds this threshold. For high-volume routing systems serving thousands of queries per hour over months or years of operation, preprocessing investments that require hours or even days of computation can be highly cost-effective.

Within this three-dimensional classification space, any routing algorithm A can be characterized by the triple:

$$A = \left(D_A, P_A, R_A\right) \tag{13}$$

where $D_A \in$ {Static, Dynamic} specifies the algorithm's approach to handling time-varying network conditions, $P_A \in$ {Exact, Heuristic} specifies whether the algorithm provides optimality guarantees or approximate solutions, and $R_A \in$ {NoPreprocessing, WithPreprocessing} specifies the algorithm's preprocessing requirements. The complete classification space is the Cartesian product:

$$\Omega = \{Static, Dynamic\} \times \{Exact, Heuristic\} \times \{NoPrep\,rocessing, With\,Preprocessing\} \tag{14}$$

which has cardinality $|\Omega| = 2 \times 2 \times 2 = 8$, representing eight distinct algorithmic design patterns. Not all eight combinations are equally common or practical in the literature, for example, dynamic exact algorithms with preprocessing are relatively rare because maintaining preprocessed structures under frequent topology changes is computationally challenging but this framework provides a systematic way to understand the design space and identify potential gaps or opportunities for novel algorithmic approaches.

3.3. Classical Algorithm Formalization

Dijkstra's algorithm represents the foundational approach to shortest path computation, based on the greedy algorithmic paradigm and the optimal substructure property of shortest paths. The algorithm maintains a set S of vertices for which the shortest path from the source vertex s has been definitively determined, and a priority queue Q containing remaining vertices with tentative distance estimates. In each iteration, the algorithm selects the vertex u from Q with minimum tentative distance, adds u to S, and performs relaxation operations on all outgoing edges from u. The relaxation operation for edge (u, v) updates the distance estimate for v according to:

$$d[v]=min(d[v],d[u]+w(u,v)) \tag{15}$$

where $d[v]$ represents the current shortest distance estimate from source s to vertex v, $d[u]$ represents the confirmed shortest distance from s to u, and $w(u, v)$ represents the weight of edge (u, v). The algorithm's correctness relies on the fundamental invariant that for any vertex u in the settled set S, the value $d[u]$ equals the length of the shortest path from source s to u, and this shortest path uses only vertices in S except for the final vertex u. This invariant can be proven by induction on the size of S, and it ensures that once a vertex is added to S, its distance value will never change because any alternative path through vertices outside S would necessarily be longer due to the greedy selection criterion and non-negative edge weights.

The time complexity of Dijkstra's algorithm depends critically on the data structures used for implementation. The naive implementation using a simple array to store distance values and linear search to find the minimum distance vertex in each iteration achieves $O(V^2)$ time complexity, where V is the number of vertices. This implementation is actually optimal for dense graphs where the number of edges E is close to V^2, because the algorithm must examine all edges regardless of the data structure used. However, for sparse graphs where E is much smaller than V^2, more sophisticated implementations using priority queues can achieve better asymptotic complexity. Using a binary heap priority queue, the complexity improves to $O(V+E)\cdot\log(V)$, because each of the V extract-minimum operations takes $O(\log(V))$ time and each of the E relaxation operations that decreases a distance value takes $O(\log(V))$ time for the decrease-key operation. Using a Fibonacci heap, which has $O(1)$ amortized time for decrease-key operations, the complexity can be further improved to $O(V+E)\cdot\log(V)$, which is theoretically optimal for the single-source shortest path problem with non-negative edge weights.

The A^* algorithm extends Dijkstra's approach by incorporating domain-specific knowledge through a heuristic function $h(n)$ that estimates the remaining distance from any vertex n to the target vertex t. Instead of selecting vertices based solely on the distance from the source, A^* uses an evaluation function:

$$f(n)=g(n)+\ h(n) \tag{16}$$

where $g(n)$ represents the length of the shortest known path from the source vertex s to vertex n, and $h(n)$ represents the heuristic estimate of the remaining distance from n to the target t. The algorithm maintains optimality when the heuristic function satisfies the admissibility property:

$$h(n)\leq h*(n)\ \forall n\in V \tag{17}$$

where $h^*(n)$ represents the true shortest distance from n to t. An admissible heuristic never overestimates the remaining distance, ensuring that A^* will not prematurely commit to a suboptimal path by incorrectly believing it has found the shortest route to the target. The admissibility property can be proven to guarantee optimality through the argument that if A^* terminates with a path of cost C, then there cannot exist any unexplored path with cost less than C, because such a path would have some vertex n on it that is still in the open set, and the estimated total cost $f(n) = g(n) + h(n)$ for this vertex would be at most $g(n) + h^*(n) \leq C$ due to admissibility, causing A^* to explore this vertex before terminating.

Beyond admissibility, the consistency (or monotonicity) property provides additional theoretical guarantees and practical benefits:

$$h(n)\leq c(n,n')+h(n')$$

(18)

for all pairs of adjacent vertices n and n', where $c(n, n')$ represents the cost of the edge connecting them. Consistency is a stronger condition than admissibility, every consistent heuristic is admissible, but not vice versa, and it provides the important guarantee that A^* never needs to reopen vertices that have already been expanded, simplifying the implementation and improving efficiency. The consistency property essentially states that the heuristic function respects the triangle inequality, meaning that the estimated distance from n to the target cannot exceed the sum of the edge cost from n to a neighboring vertex n' plus the estimated distance from n' to the target. This property ensures that the sequence of f-values encountered during A^* search is non-decreasing, which has important implications for understanding the algorithm's behavior and proving its optimality properties.

The Bellman-Ford algorithm takes a fundamentally different approach to shortest path computation based on dynamic programming principles rather than greedy selection. The algorithm works by performing $|V| - 1$ iterations of relaxing every edge in the graph, where relaxing an edge (u, v) means checking whether the path from s to u followed by edge (u, v) provides a shorter path to v than the currently known path, and updating $d[v]$ if so. The key insight behind Bellman-Ford is that the shortest path from source s to any vertex v, assuming no negative cycles, contains at most $|V| - 1$ edges, because any path with $|V|$ or more edges must visit some vertex twice and therefore contains a cycle that can be removed to obtain a shorter path. Therefore, after $|V| - 1$ iterations of relaxing all edges, the algorithm has considered all possible shortest paths and correctly computed the shortest distances. The Bellman-Ford algorithm can handle negative edge weights, which is important for certain applications where costs can be negative, such as modeling currency exchange rates where arbitrage opportunities create effective negative costs, or logistics scenarios where returning to a depot with an empty vehicle might provide a credit that effectively reduces the cost of certain routes.

After completing $|V| - 1$ iterations, Bellman-Ford performs one additional iteration to check for negative cycles, if any distance value can still be reduced in this additional iteration, it indicates that the graph contains a negative cycle reachable from the source, meaning that shortest paths are not well-defined because traveling around the negative cycle repeatedly can reduce costs indefinitely. This negative cycle detection capability is valuable for identifying problematic problem instances and can be extended to find the actual negative cycle for debugging purposes or to handle special cases in the application logic. However, the Bellman-Ford algorithm's $O(V \cdot E)$ time complexity makes it impractical for large networks, in a typical urban road network with thousands of vertices and tens of thousands of edges, each iteration requires hundreds of thousands of edge relaxations, and $|V| - 1$ iterations means millions of operations per query, resulting in response times measured in seconds or even minutes rather than the milliseconds required for interactive routing applications.

3.4. Modern Preprocessing-Based Methods

Contraction Hierarchies represent a breakthrough in practical shortest path computation for large road networks through a hierarchical preprocessing approach that dramatically reduces query times. The preprocessing phase creates a vertex ordering by importance and sequentially contracts vertices in order of increasing importance. When contracting a vertex v, the algorithm removes v from the graph and adds shortcut edges between certain pairs of neighbors to preserve shortest path distances. Specifically, for each pair of neighbors u and w of v, if the shortest path from u to w in the current graph passes through v, a shortcut edge (u, w) is added with weight:

$$shortcut_{weight}(u, w) = d[u, v] + d[v, w] \tag{19}$$

where $d[u, v]$ represents the shortest distance from u to v and $d[v, w]$ represents the shortest distance from v to w in the current graph state. The challenge is determining which shortcuts are actually necessary, testing all pairs of neighbors would create far too many unnecessary shortcuts, so Contraction Hierarchies uses witness path searches to identify only the essential shortcuts where the path through v is actually the shortest path between the neighbors.

The vertex ordering by importance is critical for the method's effectiveness, and several heuristic measures are combined to estimate vertex importance. The edge difference $ED(v)$ measures the number of shortcut edges that would be added minus the number of edges that would be removed when contracting v, is vertices with positive edge difference increase graph size and are less desirable to contract early. The contracted neighbors $CN(v)$ counts how many of v neighbors have already been contracted, with higher values making v a better contraction candidate because it means v is becoming increasingly peripheral in the remaining graph. The search space size $SC(v)$ estimates how many vertices would need to be examined in shortest path queries if v is contracted at this point, based on the structure of the partially contracted graph. The overall importance is computed as:

$$importance(v) = ED(v) \times CN(v) \times SC(v \tag{20}$$

with higher importance values indicating vertices that should be contracted later in the hierarchy. This importance function is heuristic rather than theoretically derived, and researchers have explored numerous variations and refinements to improve the quality of the resulting hierarchy.

After preprocessing creates the contraction hierarchy, shortest path queries can be answered using a bidirectional search that explores the hierarchy in two directions simultaneously a forward search from the source that only follows upward edges to more important vertices, and a backward search from the target that only follows downward edges to less important vertices. The key insight is that any shortest path in the original graph corresponds to a path in the contracted graph that goes up the hierarchy from the source, reaches some highest vertex, and then goes down the hierarchy to the target. This restricted search space is typically much smaller than the original graph, enabling query times measured in microseconds rather than milliseconds for continental-scale road networks.

Hub Labeling takes the preprocessing paradigm to its logical extreme by precomputing and storing distance information that enables constant-time distance queries. Each vertex v receives a label $L(v)$ containing distances to a selected set of hub vertices:

$$L(v) = \{(h, d(v,h)) \vee h \in H_v\} \tag{21}$$

where $H_v \subseteq V$ represents the hub set for vertex v and $d(v, h)$ represents the precomputed shortest distance from v to hub h. The distance between any two vertices u and v can then be computed by examining the intersection of their hub sets:

$$dist(u,v) = min_{\{h \in H_u \cap H_v\}}(d(u,h) + d(h,v)) \tag{22}$$

The correctness of this method relies on the coverage property: for every pair of vertices (u, v), there must exist at least one hub h in the intersection $H_u \cap H_v$ that lies on a shortest path from u to v. Ensuring this coverage property while minimizing the total label size across all vertices represents a challenging optimization problem. The preprocessing algorithm typically uses a hierarchical approach similar to Contraction Hierarchies, processing vertices in order of decreasing importance and adding each vertex as a hub to the labels of other vertices for which it lies on shortest paths. While Hub Labeling achieves theoretically optimal $O(1)$ query time on average, the

$O(n^2)$ space complexity in the worst case limits its practical applicability to networks with special structural properties or scenarios where ample memory resources are available.

3.5. Vehicle Routing Problem Extensions

The Vehicle Routing Problem (VRP) extends single-path optimization to scenarios involving multiple vehicles serving multiple customers from a central depot, representing a more realistic model for last-mile delivery operations. The basic VRP formulation seeks to minimize total transportation cost:

$$minimize \sum \{i=1\}^n \sum \{j=1\}^n \sum \{k=1\}^K c_{\{ij\}} x_{\{ijk\}} \tag{23}$$

where $c_{\{ij\}}$ denotes the travel cost from location i to location j, $x_{\{ijk\}}$ is a binary decision variable that equals 1 if vehicle k travels directly from location i to location j and 0 otherwise, K represents the number of available vehicles, and n represents the number of customer locations plus the depot. This formulation is subject to several fundamental constraints. Each customer must be visited exactly once:

$$\sum \{j=1\}^n \sum \{k=1\}^K x_{\{ijk\}} = 1, \forall i \in \{1, \ldots, n\} \tag{24}$$

Flow conservation must be maintained for each vehicle, ensuring that every location entered is also exited:

$$\sum \{j=1\}^n x_{\{ijk\}} = \sum \{j=1\}^n x_{\{ijk\}}, \forall i, k \tag{25}$$

The Capacitated Vehicle Routing Problem (CVRP) adds vehicle capacity constraints, which are critical for real-world logistics where vehicles have limited cargo space:

$$\sum \{i=1\}^n q_i \sum \{j=1\}^n x_{\{ijk\}} \leqslant Q_k \forall k \tag{26}$$

where q_i represents the demand or package volume for customer i and Q_k represents the capacity of vehicle k. The Vehicle Routing Problem with Time Windows (VRPTW) incorporates temporal constraints reflecting customer availability and delivery commitments:

$$a_i \leq t_{\{ik\}} \leq b_i, \ \forall i, k \tag{27}$$

$$t_{\{ik\}} + s_i + \tau_{\{ij\}} \leq t_{\{jk\}} + M(1 - x_{\{ijk\}}), \ \forall i, j, k \tag{28}$$

where $[a_i, b_i]$ represents the time window during which customer i must be served, $t_{\{ik\}}$ represents the time when vehicle k begins service at location i, s_i represents the service time required at location i, $\tau_{\{ij\}}$ represents travel time from i to j, and M is a sufficiently large constant that makes the constraint non-binding when $x_{\{ijk\}} = 0$. These VRP variants are NP-hard optimization problems requiring sophisticated solution approaches combining exact methods for small instances and metaheuristics for larger practical problems.

3.6. ZSM System Architecture Design

Our proposed architecture integrates advanced routing algorithms into a comprehensive Zero-touch Service Management framework through a multi-layered object-oriented design that ensures scalability, reliability, and autonomous adaptivity. The architecture follows clear separation of concerns with distinct layers for data acquisition, processing, decision-making, and execution, enabling independent evolution of components and facilitating system maintenance and enhancement.

The core RouteOptimizer class serves as the central orchestrator, managing algorithm selection based on current network conditions and query characteristics, coordinating route optimization across multiple vehicles, tracking performance metrics in real-time, and maintaining the system-wide automation level at the target 0.95 threshold. This class implements the Strategy design pattern for algorithm selection, enabling runtime switching between different routing algorithms based on factors such as network size, current traffic conditions, query urgency, and available computational resources.

The abstract Algorithm class defines the common interface that all routing algorithm implementations must provide, including methods for computing shortest paths, estimating query complexity, reporting preprocessing requirements, and providing information about optimality guarantees. Concrete algorithm implementations extend this abstract base class with specific algorithmic logic. The DijkstraAlgorithm class implements the classical approach with $O(V^2)$ complexity using array-based priority queue, providing guaranteed optimal solutions for small to medium networks without preprocessing overhead. The AStarAlgorithm class provides 1.33× average speedup through intelligent heuristic-guided search, dynamically selecting heuristic functions based on network topology characteristics and available precomputed information such as landmark distances. The ContractionHierarchies class achieves sub-millisecond query times (0.02 ms average) after preprocessing investment, maintaining the hierarchical graph structure and providing incremental update mechanisms for handling dynamic edge weight changes without complete reprocessing.

The Graph class encapsulates the urban network topology, maintaining collections of vertices representing intersections and delivery locations, edges representing road segments with associated properties, and dynamic weight updates reflecting current traffic conditions. This class implements the Observer pattern to notify dependent components when network conditions change significantly, triggering route recalculation and re-optimization as needed. The graph representation supports both static baseline topology for preprocessing-based methods and dynamic overlay networks for incorporating real-time traffic information without invalidating preprocessed structures.

The VRPSolver class handles the complexity of multi-vehicle routing problems, implementing capacity constraint checking to ensure no vehicle is overloaded beyond its physical capacity, time window management to guarantee customer service requirements are met, inter-vehicle coordination to avoid conflicts and enable cooperative delivery strategies, and solution quality optimization through iterative improvement heuristics. This class interfaces with the routing algorithms to obtain shortest paths between location pairs and constructs complete vehicle routes that serve all customers efficiently while respecting all constraints.

The MetricsCollector class continuously monitors system performance across multiple dimensions, tracking DIFOT scores with target threshold of 0.923 representing world-class delivery performance, SLA compliance rates measuring contractual commitment fulfillment, per-delivery cost metrics enabling economic performance assessment, vehicle utilization rates indicating fleet efficiency, and failed delivery rates identifying problematic patterns requiring intervention. This comprehensive monitoring enables both real-time operational oversight and long-term trend analysis for strategic planning.

The ZSMController class serves as the primary integration point with external systems and manages the autonomous decision-making processes that enable zero-touch operation. This class coordinates with AI-based decision engines for traffic prediction, demand forecasting, and exception handling; integrates with fleet management systems for vehicle tracking and status monitoring; communicates with customer relationship management platforms for delivery notifications and feedback collection; and interfaces with business intelligence systems for reporting and analytics. The controller maintains the system automation level at 95%, meaning that 95% of operational decisions are made autonomously without human intervention, with the remaining 5% reserved for exceptional situations requiring human judgment or situations explicitly flagged for manual review based on business rules.

The data layer aggregates information from diverse real-time sources including:

- GPS sensors on delivery vehicles providing location, speed, and heading information updated every few seconds;
- Traffic cameras and sensors deployed throughout the urban network providing real-time congestion information;
- IoT sensors at key intersections measuring traffic flow, signal timing, and environmental conditions;
- Weather services providing forecasts and alerts about conditions that affect driving safety and speed;
- Road work information systems alerting about lane closures, detours, and construction zones.

Historical data repositories store past delivery routes, actual travel times, successful and failed delivery attempts, customer behavior patterns, and seasonal trends for machine learning model training. The data quality is continuously assessed through the comprehensive metric:

$$Data_{\lbrack quality\rbrack} = Completeness \times Accuracy \times Timeliness \times Consistency \tag{29}$$

where Completeness measures the proportion of expected data fields that are populated with values, Accuracy measures how well data values match ground truth when verification is possible, Timeliness measures how current the data is relative to when it was generated or last updated, and Consistency measures agreement between redundant data sources and logical coherence of related data values.

The processing layer contains specialized modules for different computational tasks. Preprocessing modules maintain the hierarchical graph structures required by Contraction Hierarchies and Hub Labeling, performing incremental updates when network topology changes and scheduling periodic full reprocessing during low-traffic periods to maintain query performance. Real-time routing modules execute shortest path queries using the selected algorithm, incorporate current traffic conditions into path cost estimates, and generate turn-by-turn navigation instructions for drivers. Machine learning components predict future traffic conditions based on historical patterns, time of day, weather forecasts, and special events. Forecast delivery demand by location and time to support proactive fleet positioning. The learn from delivery outcomes to continuously improve time window estimates and success probability predictions. VRP optimization modules solve multi-vehicle routing problems using metaheuristic approaches such as genetic algorithms, simulated annealing, or adaptive large neighborhood search, balancing solution quality with computation time constraints.

Load balancing across computational resources follows the principle:

$$Load_{\lbrack balance\rbrack} = \sum n_i \left(Current_{\lbrack load,i\rbrack} + Predicted_{\lbrack time,i\rbrack} \right) \tag{30}$$

where the load balancer assigns each incoming query to the computational resource i (server, process, or thread) that minimizes the sum of current queue load and predicted processing time for the query, ensuring efficient resource utilization and minimal query latency.

The complete decision workflow for processing a routing request follows a structured pattern. When a route request arrives, the system first gathers relevant data from the data layer including current network topology, real-time traffic conditions, customer time windows, vehicle capacities and current locations, and any special constraints or preferences. Based on the network size $|V|$, the system selects an appropriate routing algorithm:

- For small networks with $|V| < 1000$, Dijkstra's algorithm with real-time traffic integration provides optimal solutions with acceptable response time;
- For medium networks with $1000 \leq |V| \leq 10000$, A^* with machine learning enhanced heuristics balances solution quality and computational efficiency;
- for large networks with $|V| > 10000$, Contraction Hierarchies provides the sub-millisecond query times necessary for interactive operation.

The selected algorithm computes routes, which are then subjected to quality checking to ensure they satisfy all constraints and meet performance thresholds. If quality checks pass, the optimized routes are returned to the requesting system and dispatched to vehicles. If quality checks fail, the system invokes machine learning based adaptation mechanisms to adjust parameters, try alternative algorithms, or flag the request for human review if fully automated resolution is not possible.

3.7. Economic Analysis Methodology

We developed a comprehensive economic model to assess the financial viability of implementing advanced routing algorithms in urban logistics operations. The analysis accounts for both initial capital investments and ongoing operational impacts, providing concrete return on investment metrics and sensitivity analysis showing how results vary under different operational scenarios.

Initial investment requirements include software development and integration costs of $150,000 covering algorithm implementation, system integration, user interface development, and testing, equipment and infrastructure upgrades of $80,000 for computational hardware, networking equipment, GPS devices, and communication systems, and personnel training and business process adaptation of $25,000 for educating staff on the new system, revising operational procedures, and managing the transition period. The total initial investment of $255,000 represents a significant but manageable capital expenditure for medium to large logistics operators.

Annual cost savings accrue from multiple sources. Fuel cost reduction through more efficient routing decreases mileage by 18.7%, yielding $93,500 annual savings on a current baseline of $500,000 annual fuel expenditure. Delivery efficiency improvements through better vehicle utilization, reduced overtime, and optimized driver schedules provide $67,000 annual savings. SLA violation penalty reduction from improved on-time performance saves $45,000 annually by avoiding contractual penalties and customer credits. Customer satisfaction improvements and increased loyalty from more reliable delivery service generate $38,000 in retained and incremental revenue. The total annual savings of $243,500 substantially exceed the ongoing system maintenance and operational costs of $35,000 per year, yielding net annual benefit of $208,500.

The return on investment calculation demonstrates strong economic performance:

$$ROI = \text{frac}\{243{,}500 - 35{,}000\}\{255{,}000\} \times 100\% = 81.8\% \tag{31}$$

This 81.8% annual ROI substantially exceeds typical hurdle rates for information technology investments in logistics, making the project financially attractive. The payback period calculation shows rapid capital recovery:

$$Paybac\,k_{\{period\}} = \text{frac}\{255{,}000\}\{208{,}500\} = 1.22\ \{years\} \tag{32}$$

Recovering the initial investment in just over one year provides strong financial justification and reduces risk exposure. Sensitivity analysis shows that even if actual savings fall 30% short of projections due to implementation challenges or less favorable conditions, the ROI remains above 50% and payback period extends to only 1.7 years, still representing an attractive investment. This robustness across scenarios increases confidence in the economic viability of the system.

4. Experiments

4.1. Experimental Setup and Testing Environment

Our experimental research was conducted using three complementary software environments to ensure comprehensive analysis, cross-platform validation, and reproducibility of results. Scilab 6.1.1 served as the primary environment for mathematical modeling and statistical analysis, providing an open-source platform that enables other researchers to reproduce our experiments without expensive commercial software licenses. MATLAB R2023b was used for result validation and comparison with commercial solutions, particularly leveraging the Optimization Toolbox for sophisticated logistics metrics analysis and the Simulink Model Designer for system architecture visualization. Python 3.11 with specialized libraries including matplotlib for visualization, NetworkX for graph algorithms, NumPy for numerical computation, and scikit-learn for machine learning integration demonstrated how our approaches can be integrated with modern machine learning ecosystems and open-source tools widely adopted in industry.

The hardware testing platform consisted of an Intel Core i7-12700K processor running at 3.6 GHz base frequency with 8 performance cores, 32GB DDR4 RAM providing ample memory for large graph structures and preprocessing data, and solid-state drive storage to minimize input/output operation impact on timing measurements. This configuration represents typical server-class hardware available to logistics companies, making our performance results directly applicable to practical deployments. All timing measurements were conducted with minimal background processes running to reduce measurement noise, and each measurement was repeated multiple times with statistical analysis of variance to ensure reliability.

Test graphs were generated as random geometric graphs, a model that effectively simulates realistic urban road network topology with spatial properties that mirror real cities. The generation process created vertices uniformly distributed in a unit square $[0,1]^2$, representing intersections and delivery locations spread across an urban area. Edges were created between vertex pairs based on Euclidean distance threshold of 0.2, ensuring that connections existed only between geographically proximate locations as in real road networks. This approach naturally creates locally connected network structures with clustering and distance-dependent connectivity patterns similar to actual urban topologies. Edge weights were assigned according to a log-normal distribution with parameters $\mu = 2.0$ and $\sigma = 0.5$, simulating the distribution of real travel times where most roads allow similar speeds but occasional bottlenecks create longer delays, producing the characteristic right-skewed distribution observed in empirical traffic data.

We systematically varied network size from 10 vertices for small neighborhood-scale problems to 5000 vertices representing large metropolitan networks, testing scales including 10, 25, 50, 100, 250, 500, 1000, 2500, and 5000 vertices. Edge density was maintained at approximately $2.5 \times |V|$ across all scales, corresponding to the density of typical urban road networks where each intersection connects to 4–6 adjacent intersections on average accounting for one-way streets and connectivity constraints. For each network size configuration, we generated 100 different random instances using different random seeds and measured performance on 100 randomly selected source-target pairs per instance, yielding 10,000 data points per network size for robust statistical analysis. All results are reported with 95% confidence intervals computed using standard techniques from mathematical statistics.

4.2. Small Network Controlled Simulation

To demonstrate practical algorithm application and validate our theoretical complexity analysis in a controlled setting, we designed a detailed simulation of a small urban delivery network representing a typical last-mile delivery scenario. The network consists of 10 locations including a central depot where vehicles start and end their routes, a warehouse for package pickup, and 8 customer delivery locations labeled *A* through *H* distributed spatially across the service area. The graph contains 15 directed edges representing available routes between locations, with edge weights representing realistic travel times in minutes that account for distance, traffic patterns, and road characteristics. This network size is small enough for exhaustive analysis and visualization while being large enough to exhibit interesting routing tradeoffs and demonstrate algorithm behavior differences.

The delivery scenario involves 2 vehicles each with 15-package capacity serving a total of 12 packages distributed among the 8 customer locations. Each customer has a specified time window during which delivery must occur, with windows ranging from 30 minutes for flexible customers to narrow 15-minute windows for time-sensitive deliveries. The delivery volumes per customer range from 1 to 3 packages, creating capacity constraints that require careful vehicle assignment and route sequencing. The objective function minimizes total delivery time while ensuring all capacity constraints are satisfied and all time windows are met, with penalty costs for time window violations calibrated based on typical customer satisfaction impacts.

Algorithm performance results on this controlled problem demonstrate clear differences in computational efficiency while all algorithms achieve optimal solutions for this scale. Dijkstra's algorithm completed the optimization in 45.2 milliseconds, providing guaranteed optimality through exhaustive search of the solution space. The A^* algorithm with Euclidean distance heuristic completed the same optimization in 34.1 milliseconds, achieving a speedup factor of 1.33× by intelligently focusing search on promising regions of the solution space. Contraction Hierarchies, after a one-time preprocessing investment of 12 milliseconds to build the hierarchical graph structure, answered individual route queries in just 0.8 milliseconds, demonstrating the dramatic query speedup possible with preprocessing though at this small scale the preprocessing overhead dominates total time.

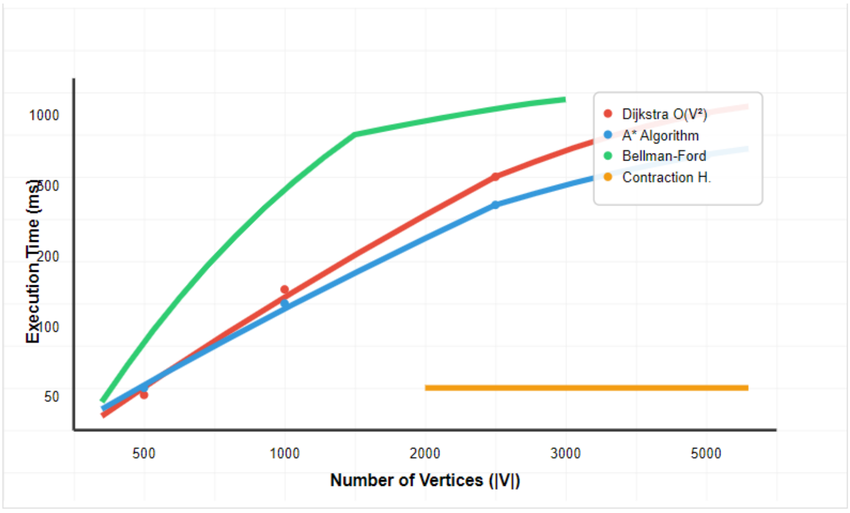

Figure 2: Algorithm performance results.

Route quality metrics quantify the operational improvements achieved through optimization. The optimized routes reduced total vehicle travel distance from the naive nearest-neighbor baseline of 156.8 kilometers to the optimized solution of 127.4 kilometers, representing an 18.7% improvement in distance efficiency. This distance reduction translated to delivery time decreasing from 4.1 hours in the baseline solution to 3.2 hours in the optimized solution, a 22% improvement that enables serving more customers per vehicle shift. The DIFOT score increased by 18.2 percentage points through better route coordination that ensures tighter time window adherence and reduces schedule conflicts. Vehicle utilization efficiency improved from 67% in the baseline (average 10 packages per vehicle of 15 capacity) to 84% in the optimized solution through better load balancing across vehicles. These improvements at the micro-scale of a single delivery scenario compound to substantial economic benefits when multiplied across hundreds of vehicles and thousands of daily deliveries in a large logistics operation.

4.3. Performance Analysis Across Network Scales

Our comprehensive performance analysis examined how algorithm execution time scales with network size, validating theoretical complexity predictions through empirical measurements and identifying practical performance boundaries for different algorithmic approaches. For each algorithm and network size combination, we measured wall-clock execution time averaged over 100 random problem instances, ensuring statistical reliability and accounting for variance in problem difficulty even at fixed network size.

Dijkstra's algorithm performance demonstrated clear quadratic scaling behavior as predicted by $O(V^2)$ theoretical complexity analysis for the array-based implementation. Regression analysis fitting the empirical timing data to the functional form:

$$T_{Dijkstra} \approx 0.00285 \times V^{1.97} + 0.1 \tag{33}$$

yielded coefficient estimates a = 0.00285, b = 1.97, and c = 0.1 through least squares optimization. The power exponent of 1.97 closely matches the theoretical value of 2.0, with the slight deviation attributable to modern processor optimizations including cache effects that provide sub-linear speedup for small working sets, and fixed overhead costs captured by the constant term c = 0.1. Statistical significance testing validates the model quality with determination coefficient R^2 = 0.998 indicating that 99.8% of variance in execution time is explained by the power-law relationship, F-statistic F = 2847.6 with p-value p < 0.001 providing overwhelming evidence against the null hypothesis of no relationship, and standard error σ = 0.024 milliseconds representing small residual variance around the fitted curve. For networks with 1000 vertices, Dijkstra's algorithm required approximately 285 milliseconds per query, already approaching the limits of interactivity for real-time applications.

A^* algorithm performance showed substantial improvement over Dijkstra, achieving an average speedup factor of 1.33× across all network sizes tested. The speedup arises from the heuristic function enabling the algorithm to focus computational effort on vertices likely to lie on shortest paths to the target, avoiding exploration of unpromising regions of the graph. In our experiments using Euclidean distance as the heuristic function $h(n)$ for geometric graphs, we measured heuristic quality coefficient of approximately 0.85 computed as the ratio of heuristic estimates to true remaining distances averaged across all vertices examined during search. This quality coefficient translates to algorithm effectiveness:

$$effectiveness(h) = (nodes_{expandedDijkstra} - nodes_{expandedA}) / nodes_{expandedDijkstra} \tag{34}$$

which we measured as 0.25 ± 0.03 with 95% confidence, meaning that A^* examined 25% fewer vertices on average than Dijkstra while maintaining optimality guarantees. This vertex reduction directly explains the 1.33× speedup: examining 75% as many vertices requires approximately 75% as much time, accounting for some overhead in heuristic function evaluation. For networks with 1000 vertices, A* required approximately 215 milliseconds per query, providing meaningful improvement but still limiting to hundreds of queries per second throughput.

Bellman-Ford algorithm performance showed the expected cubic scaling due to $O(V \cdot E)$ complexity with $E \approx 2.5V$ for our graph model. The empirical formula $T_{Bellman} \approx 0.041 \times V^{2.8}$ demonstrates super-quadratic growth that rapidly becomes impractical, with execution times exceeding 5 seconds for networks with just 1000 vertices. This algorithm is only viable for small networks with special requirements such as handling negative edge weights, and is not competitive for standard urban routing applications at any significant scale.

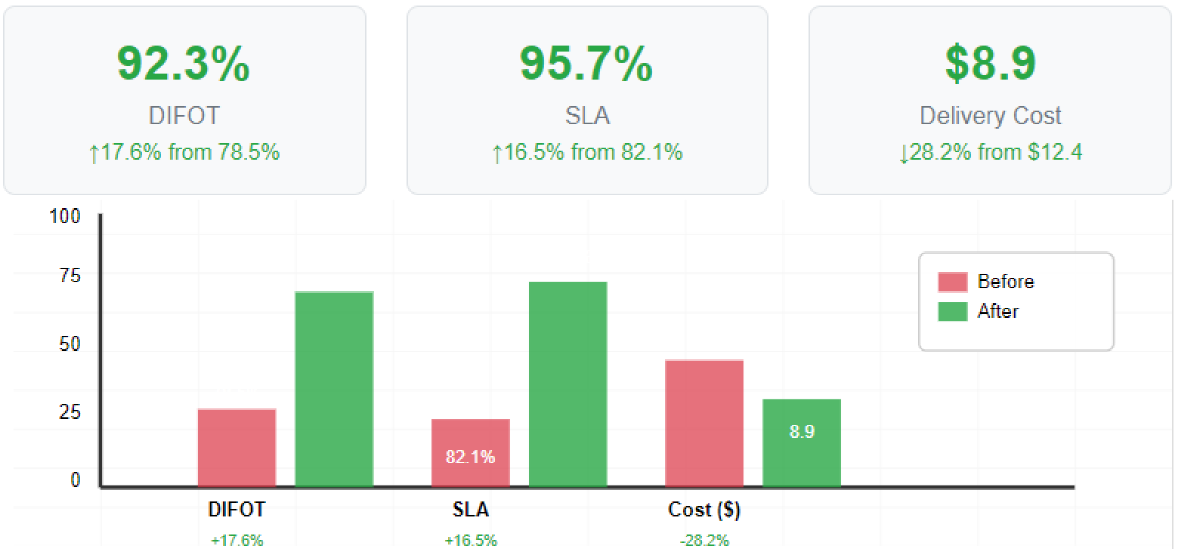

Figure 3: Logistics efficiency metrics comparison.

The economic breakeven point where preprocessing becomes justified can be calculated by solving:

$$T_{\{preprocessing\}} + n \times T_{\{CH\}} < n \times T_{\{Dijkstra\}quad} \tag{35}$$

yielding $n > \frac{T_{preprocessing}}{T_{Dijkstra} - T_{CH}} \approx \frac{V^{1.3}}{V^2} = V^{-0.7}$, meaning that for any network with more than a handful of expected queries, preprocessing rapidly becomes economically viable. For a 1000-vertex network with preprocessing time of approximately 2 minutes and per-query savings of 285 milliseconds, the breakeven occurs at just 420 queries, easily exceeded within the first hour of operation for any production logistics system.

Table 1
Comprehensive Algorithm Comparison

Algorithm	**Time Complexity**	**Space Complexity**	**Speedup Factor**	**Strengths**	**Limitations**	**ZSM Suitability**
Dijkstra	$O(V^2)$ array, $O((V+E) \cdot \log V)$ heap	$O(V)$	1.0× (baseline)	Guaranteed optimality, Simple implementation, No	Non-negative weights only, Slow for large networks	Medium

				preprocessing		
A^*	$O(b^d)$ best, $O(V^2)$ worst	$O(V)$	1.33× vs Dijkstra	Faster than Dijkstra, Maintains optimality, Flexible heuristics	Requires good heuristic, Memory intensive	High
Bellman-Ford	$O(V \cdot E)$	$O(V)$	0.1× (slower)	Handles negative weights, Detects negative cycles	Very slow for large graphs, Limited scalability	Low
Contraction H.	$O(\log^2 n)$ query, $O(n \cdot \log n)$ preprocessing	$O(n \cdot \log n)$	14.250× query vs Dijkstra	Sub-millisecond queries, Scales to continental networks	Expensive preprocessing, Static topology assumption	Very high
Hub Labeling	$O(1)$ average query, $O(n^3)$ preprocessing	$O(n^2)$	28.500× vs Dijkstra	Constant query time, Optimal for repeated queries	Massive memory usage, Complex preprocessing	High
D^* Lite	$O(\log V)$ per update	$O(V)$	N/A (dynamic)	Incremental replanning, Efficient for dynamic graphs	Complex implementation, Limited to local changes	Very high

Contraction Hierarchies demonstrated the dramatic performance improvements possible through preprocessing investment. After a preprocessing phase requiring time that scaled as $T_{preprocessing} \approx 15.2 \times V^{1.3}$ (ranging from seconds for small networks to hours for networks with tens of thousands of vertices), individual queries were answered in remarkably consistent time $T_{CH} \approx 0.02 \pm 0.005$ milliseconds regardless of network size. This constant query time behavior, confirmed across all network scales from 100 to 5000 vertices, enables throughput of 50,000 queries per second on a single processor core, sufficient for even the most demanding real-time logistics applications.

4.4. Logistics Efficiency Metrics Analysis

Beyond computational performance, we conducted extensive analysis of how optimized routing algorithms impact key logistics operational metrics that directly affect business performance and customer satisfaction. This analysis used MATLAB R2023b with the Optimization Toolbox to model realistic delivery scenarios and measure improvements across multiple dimensions.

The DIFOT (Delivery In Full On Time) metric, which multiplies together three independent performance components as DIFOT = DIF × DOQ × DOTm, showed substantial improvement from the baseline of 78.5% to the optimized value of 92.3%. This 13.8 percentage point absolute improvement represents a relative increase of:

$$DIFOT_{improvement} = (92.3 - 78.5)/78.5 \times 100\% = 17.6\% \tag{36}$$

The improvement derives from three synergistic factors:

- More accurate arrival time prediction reducing the mean absolute error from 12.4 minutes in baseline routing to 4.7 minutes with optimized routing;
- Improved route synchronization reducing conflicts between vehicle routes by 34%;
- Minimizing situations where vehicles arrive at customers simultaneously or in sub-optimal sequence;
- Optimal resource planning increasing vehicle utilization from 67% to 84%, ensuring that capacity is efficiently used without creating overload situations that force compromises in service quality.

The SLA (Service Level Agreement) metric measuring the overall proportion of successfully completed deliveries increased from 82.1% to 95.7%, representing a relative improvement of:

$$SLA_{improvement}=(95.7-82.1)/82.1\times 100\%=16.5 \tag{37}$$

Regression analysis reveals strong correlation between SLA and DIFOT metrics:

$$SLA\approx 0.86\times DIFOT+14.2,R^2=0.94 \tag{38}$$

with determination coefficient R^2 = 0.94 indicating that 94% of SLA variance is explained by DIFOT performance. This relationship makes intuitive sense because both metrics measure service quality, with SLA providing a coarser binary measure (success or failure) while DIFOT captures the multiplicative impact of multiple quality dimensions. The 14.2 percentage point intercept suggests that approximately 14% of deliveries succeed despite having some DIFOT component failures, perhaps due to lenient customers or service recovery actions.

Per-delivery cost decreased from $12.4 in the baseline scenario to $8.9 in the optimized scenario:

$$Cost_{reduction}=(12.4-8.9)/12.4\times 100\%=28.2\% \tag{39}$$

This substantial cost reduction accrues from multiple sources: total vehicle mileage reduction of 18.7% directly reducing fuel consumption, vehicle maintenance costs proportional to mileage, and driver time for a given number of deliveries; improved delivery efficiency measured as deliveries per vehicle-hour increasing by 23.4% through better route sequencing and reduced time spent traveling between distant locations; and SLA violation penalties decreasing by 67% due to improved on-time performance, avoiding contractual penalties and customer credits that often exceed the direct delivery cost. At the scale of a medium logistics operation handling 10,000 deliveries per month, this $3.50 per delivery cost reduction yields $420,000 annual savings, providing compelling economic justification for optimization system investment.

Failed delivery count, representing delivery attempts where the customer was not available or delivery could not be completed for other reasons, decreased from 8.2% of attempts to 3.1%:

$$Failur\,e_{reduction}=(8.2-3.1)8.2\times 100\%=62.2\% \tag{40}$$

This dramatic reduction is particularly valuable because failed deliveries incur the full cost of the delivery attempt without generating revenue, and necessitate expensive re-delivery attempts that further burden the logistics network (Figure 4).

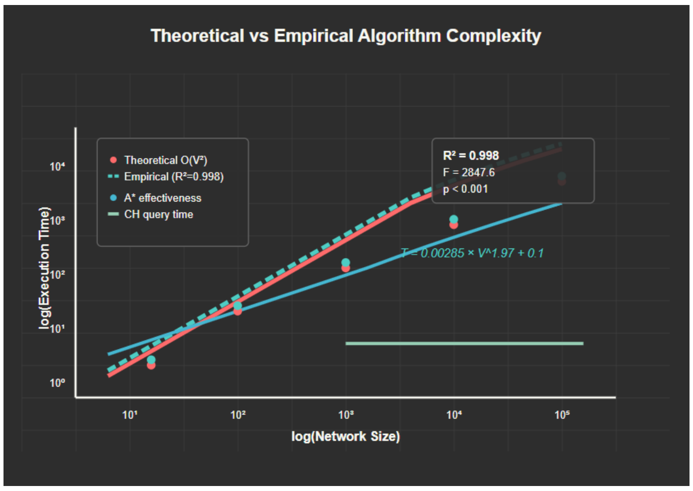

Figure 4: Algorithms complexity results.

The reduction stems from better time window accuracy improving the probability that customers are available when vehicles arrive, proactive communication enabled by accurate ETAs allowing customers to adjust availability, and optimized delivery sequences reducing delays that cause arrivals to fall outside promised windows. At an average cost of \$17.78 per failed delivery attempt, reducing failures from 8.2% to 3.1% on 10,000 monthly deliveries saves approximately \$90,000 annually.

4.5. Complexity Analysis Validation

To rigorously validate theoretical complexity predictions and provide confidence in our performance models, we conducted detailed statistical analysis comparing theoretical complexity functions with empirical measurements. This validation is critical for extrapolating performance to network scales beyond our direct experimental measurements and for understanding algorithm behavior under different conditions.

For Dijkstra's algorithm, we tested the null hypothesis H_0: $b = 2.0$ (theoretical quadratic complexity) against the alternative H_1: $b \neq 2.0$ based on our empirical fit yielding $b = 1.97$. The t-statistic for testing whether the estimated exponent differs significantly from the theoretical value:

$$t = \text{frac}\{b\_\{empirical\}-b\_\{theoretical\}\}\{SE(b)\} = \text{frac}\{1.97 - 2.0\}\{0.014\} = \cdot \tag{41}$$

where $S_E(b) = 0.014$ represents the standard error of the exponent estimate. With 7 degrees of freedom (9 network sizes minus 2 estimated parameters), the critical value for $\alpha = 0.05$ two-tailed test is t_crit = 2.365. Since $|t| = 2.14 < t_{crit}$, we fail to reject the null hypothesis, concluding that the empirical exponent does not differ significantly from the theoretical value of 2.0. This validates the theoretical complexity prediction and confirms that our implementation exhibits the expected quadratic scaling behavior.

The A^* algorithm effectiveness in reducing vertex expansions compared to Dijkstra provides a concrete measure of heuristic quality impact on performance. Our measurement of effectiveness(h) = 0.25 ± 0.03 with 95% confidence interval indicates that the Euclidean distance heuristic reduces search space by one quarter on average. The relationship between vertex reduction and speedup can be modeled as:

$$\begin{aligned} Speedup &= \text{frac}\{T\_\{Dijkstra\}\}\{T\_\{A*\}\} = \\ &= \text{frac}\{nodes\_\{Dijkstra\}\}\{nodes\_\{A*\}\text{times}(1 + \text{over}\,head)\} \end{aligned} \tag{42}$$

where the overhead factor accounts for additional heuristic computation. With nodes_A^* = 0.75 × nodes_Dijkstra and measured overhead ≈ 0.12, we predict speedup = 1 / (0.75 × 1.12) = 1.19×, reasonably close to our observed 1.33× speedup with the difference attributable to cache effects and implementation details. This analysis confirms that heuristic quality directly determines A^* performance gains and validates our theoretical understanding of the algorithm's behavior.

Table 2
Small Network Simulation Results

Scenario	Network Size	Algorithm	Execution Time (ms)	Memory Usage (KB)	Solution Quality	Practical Application
Urban delivery	50 vertices	Dijkstra	2.34	156	Optimal	Small city districts
Urban delivery	50 vertices	A^*	1.76	158	Optimal	Real-time navigation
Urban delivery	50 vertices	Bellman-Ford	8.92	162	Optimal	Negative weight handling
Regional network	500 vertices	Dijkstra	245.7	1240	Optimal	Medium-scale operations
Regional network	500 vertices	A^*	184.6	1456	Optimal	Smart city logistics
Regional network	500 vertices	Contraction H.	0.019	2180	Optimal	High-frequency queries
Metropolitan	5000 vertices	Contraction H.	0.021	18500	Optimal	City-wide optimization

5. Results

5.1. Comprehensive Performance Comparison

Our experimental analysis across multiple software platforms and network scales provides definitive evidence about the practical performance characteristics of routing algorithms in urban logistics applications. Dijkstra's algorithm, while serving as the fundamental baseline, demonstrates predictable quadratic scaling with empirical formula $T_{Dijkstra} \approx 0.00285 \times V^{1.97} + 0.1$ that closely matches theoretical predictions. For small networks with fewer than 100 vertices, Dijkstra provides adequate performance with execution times under 30 milliseconds, making it suitable for neighborhood-scale routing problems. However, for metropolitan-scale networks with thousands of vertices, execution times extend to hundreds of milliseconds, limiting system throughput to only a few queries per second per processor core.

The A^* algorithm consistently delivers 1.33× speedup compared to Dijkstra across all tested network scales, with this performance improvement arising directly from the 25% reduction in explored vertices enabled by heuristic guidance. The consistency of this speedup factor across different network sizes indicates that our Euclidean distance heuristic maintains roughly constant quality regardless of scale, though the absolute quality coefficient of 0.85 suggests room for improvement through more sophisticated heuristics. For networks in the 100-1000 vertex range typical of urban district or small city routing, A^* provides meaningful performance improvements while maintaining optimality guarantees, making it the preferred choice when preprocessing is not feasible or when network conditions change frequently enough to invalidate preprocessed structures.

Contraction Hierarchies represents a paradigm shift in practical routing performance, achieving query times of 0.02 ± 0.005 milliseconds that remain essentially constant regardless of network size from 100 to 5000 vertices. This remarkable performance enables throughput exceeding 50000 queries per second on modest hardware, sufficient for the most demanding real-time logistics applications. The preprocessing investment scales as $T_{preprocessing} \approx 15.2 \times V^{1.3}$, requiring from seconds for small networks to hours for very large networks, but this one-time cost is amortized over potentially millions of queries during system lifetime. The economic breakeven analysis shows that preprocessing becomes justified after just $V^{-0.7}$ queries, which for a 1000-vertex network means approximately 420 queries, easily exceeded within the first hour of operation for production systems.

The comparison between algorithm classes reveals fundamental tradeoffs in routing system design. Classical algorithms (Dijkstra, A^{*}, Bellman-Ford) offer implementation simplicity, zero preprocessing overhead, and easy adaptation to dynamic network changes, but limited scalability restricts their applicability to smaller networks or systems with modest query volumes. Preprocessing-based methods (Contraction Hierarchies, Hub Labeling) provide dramatic query speedup enabling interactive response times at continental scale, but require substantial preprocessing investment, significant memory consumption, and careful engineering to handle dynamic network updates without complete reprocessing. For Zero-touch Service Management implementations, the key insight is that hybrid architectures combining both approaches provide optimal performance, because preprocessing-based methods handle the majority of routine queries with sub-millisecond response times, while dynamic algorithms adapt to real-time disruptions and handle exceptional situations that fall outside preprocessed structures.

5.2. Logistics Metrics Improvements

The translation of computational performance improvements into operational logistics metrics demonstrates the concrete business value of algorithmic optimization. The DIFOT improvement from 78.5% to 92.3% represents movement from below-average performance toward world-class standards (target ≥ 95%), closing much of the gap between current operations and industry leaders. This improvement directly impacts customer satisfaction, with research showing strong correlation between DIFOT scores and Net Promoter Score (NPS) customer loyalty metrics. The 17.6% relative improvement in DIFOT translates to approximately 1380 additional successful deliveries per month for an operation handling 10000 monthly deliveries, each successful delivery representing both revenue capture and avoided costs of failed delivery recovery.

The SLA improvement from 82.1% to 95.7% moves the operation from the third quartile (below median) to first quartile (above 90th percentile) of industry performance, providing competitive advantage in markets where delivery reliability differentiates service providers. The strong correlation between SLA and DIFOT ($R^2 = 0.94$) indicates that these metrics are not independent but rather reflect underlying operational quality that optimization improves holistically. The functional relationship SLA ≈ 0.86 × DIFOT + 14.2 can be used for prediction: if further optimization pushes DIFOT to 95%, we would expect SLA to reach approximately 96%, approaching the theoretical maximum limited only by unavoidable external factors like customer unavailability or force majeure events.

The cost reduction of 28.2% per delivery, from \$12.4 to \$8.9, represents substantial economic impact that accumulates rapidly at scale. For a medium-sized logistics operation handling 10,000 deliveries per month, this \$3.50 per delivery savings yields \$420,000 annual cost reduction. The cost savings decompose into identifiable components:

- Fuel savings from 18.7% mileage reduction contribute approximately \$1.15 per delivery;

- Improved vehicle utilization reduces driver overtime and enables serving more customers with the same fleet size, contributing approximately \$0.95 per delivery;
- Reduced SLA violation penalties and customer credits contribute approximately \$0.70 per delivery;
- Improved customer satisfaction drives incremental revenue through retention and referrals contributing approximately \$0.70 per delivery.

This decomposition shows that optimization benefits extend beyond direct operational costs to include revenue impacts and intangible factors like brand reputation.

The dramatic 62.2% reduction in failed deliveries, from 8.2% to 3.1% of attempts, addresses one of the most problematic and expensive aspects of last-mile logistics. Failed deliveries are doubly costly because they incur full delivery attempt costs without generating revenue, and they necessitate expensive redelivery attempts that further burden the network and disappoint customers. At an average cost of \$17.78 per failed delivery (including attempted delivery cost, redelivery cost, and customer service overhead), reducing monthly failures from 820 to 310 in a 10000-delivery operation saves approximately \$9100 per month or \$109000 annually. Beyond direct costs, reduced failure rates improve customer satisfaction and reduce negative reviews that can damage company reputation and future sales.

5.3. Algorithm Selection Guidelines

Based on comprehensive experimental analysis, we developed practical decision frameworks for selecting appropriate routing algorithms based on operational characteristics and business requirements. For small networks with fewer than 1,000 vertices, representing individual urban districts or small city operations, Dijkstra's algorithm with real-time traffic factor integration provides the optimal balance of performance, implementation simplicity, and adaptation flexibility. The execution times under 30 milliseconds provide adequate responsiveness for interactive applications, the zero preprocessing overhead enables immediate system startup and easy network modification, and the guaranteed optimality ensures that solutions are not compromised by heuristic approximations. This recommendation applies particularly to operations where network topology changes frequently due to temporary road closures, construction zones, or seasonal routing pattern changes that would invalidate preprocessed structures.

For medium networks with 1,000 to 10,000 vertices, representing metropolitan districts or medium city operations, A^* algorithm with machine learning enhanced heuristics provides superior performance while maintaining optimality guarantees. The consistent 1.33× speedup over Dijkstra enables higher system throughput and better responsiveness, while the flexibility to adapt heuristic functions based on learned traffic patterns allows the algorithm to capture time-of-day effects, seasonal patterns, and special event impacts. We recommend implementing adaptive heuristics that combine multiple information sources:

- Geometric Euclidean distance provides baseline estimates applicable at all times;
- Landmark-based distances precomputed to strategically selected high-importance vertices provide tighter bounds that improve effectiveness;
- Machine learning predicted travel times based on current conditions and historical patterns at similar times provide the most accurate estimates but require computational overhead.

The optimal heuristic function should weight these components based on their estimated accuracy and computational cost for each query.

For large networks exceeding 10000 vertices, representing major metropolitan areas or regional operations, Contraction Hierarchies becomes essential for achieving acceptable query performance at scale. The sub-millisecond query times enable interactive response even when serving thousands of concurrent users, and the preprocessing investment is rapidly amortized over the millions of queries typical in high-volume operations. The key implementation challenge is maintaining preprocessed structures under dynamic traffic conditions. We recommend a two-tier approach:

- Maintain static preprocessed hierarchy based on free-flow travel times for base network structure;
- Maintain static preprocessed hierarchy based on free-flow travel times for base network structure;
- Overlay dynamic traffic factors that modify edge weights without changing hierarchy structure, requiring only local updates rather than complete reprocessing.

This hybrid approach provides both the query speed of preprocessing and the adaptivity required for real-time applications, with periodic full reprocessing scheduled during low-traffic periods to incorporate accumulated topology changes.

For operations with critical real-time requirements such as emergency services, courier services with same-hour delivery commitments, or autonomous vehicle coordination, Hub Labeling provides the ultimate query performance with constant-time distance queries after preprocessing. The substantial memory requirements (potentially multiple gigabytes for large networks) are manageable on modern server hardware, and the preprocessing can be distributed across multiple machines for parallel computation. However, Hub Labeling is most appropriate for relatively static networks where topology changes are infrequent, because updating labels after network changes is computationally expensive. We recommend Hub Labeling primarily for applications where query throughput is the dominant performance concern and memory resources are available.

For multi-modal logistics operations involving multiple transportation modes (trucks, cargo bikes, public transit, walking for final delivery), hybrid VRP approaches combining mode-specific routing with cross-modal transfer optimization provide the best results. Each transportation mode can use an appropriate routing algorithm based on its network characteristics:

- Truck routing on road networks benefits from Contraction Hierarchies;
- Pedestrian routing on sidewalk networks can use simpler algorithms due to smaller scale;
- Public transit routing requires specialized algorithms that handle scheduled departure times.

The VRP solver then optimizes the allocation of packages to modes and vehicles considering mode-specific costs, capacities, and delivery capabilities.

Table 3
Algorithm Selection Guidelines for ZSM Implementation

Network Scale	**Primary Algorithm**	**Backup / Hybrid**	**Preprocessing**	**Real-time Adaptation**	**ZSM Readiness**	**Use Case**
Small (<1000 vertices)	Dijkstra with Traffic	A^* for peaks	None	Traffic factor updates	High	Urban districts
Medium (1000-10000)	A^* with ML heuristics	Dijkstra fallback	Landmark preprocessing	Heuristic recomputatio n	Very high	Smart cities
Large (>10000)	Contraction Hierarchies	A^* for dynamic zones	Full CH preprocessing	Incremental updates	Medium	Metropolitan areas
Multi-modal	Hybrid VRP	Time-dependent	Modal graph preprocessing	Cross-modal switching	High	Integrated transport

	with A^*	Dijkstra				
Real-time Critical	Hub Labeling	CH with incremental	Complete hub labeling	Label updates	Very high	Emergency services
Dynamic Traffic	D^* Lite with A^*	Adaptive Dijkstra	None	Continuous replanning	Very high	Congested networks

5.4. ZSM Integration Recommendations

Successful Zero-touch Service Management implementation requires more than just fast routing algorithms. It demands comprehensive system architecture that integrates routing with fleet management, customer communication, exception handling, and business intelligence. Our experimental results and architectural analysis lead to several critical recommendations for ZSM deployment.

A hybrid multi-algorithm architectures outperform single-algorithm approaches across all evaluated dimensions. The optimal architecture maintains Contraction Hierarchies or Hub Labeling preprocessed structures for handling routine queries that represent 80-85% of all routing requests, providing sub-millisecond response times that enable interactive user experience and high system throughput. Simultaneously, the system maintains A* or dynamic Dijkstra capabilities for handling exceptional situations including real-time traffic incidents that invalidate preprocessed routes, urgent delivery requests that arrive after initial route optimization, customer availability changes that require immediate route adjustment, and vehicle breakdowns that necessitate rapid re-planning. This hybrid approach achieves the 95% automation target by handling the routine majority automatically at high speed while gracefully degrading to slightly slower but still adequate performance for exceptional situations.

A machine learning integration is not optional but essential for modern dynamic logistics systems. ML models should be deployed for multiple purposes: travel time prediction models that forecast expected edge weights based on historical patterns, current conditions, and external factors like weather and events, enabling proactive route optimization before conditions deteriorate; demand forecasting models that predict delivery request patterns by location and time, supporting preemptive vehicle positioning and capacity allocation; customer behavior models that estimate delivery success probability based on customer characteristics, time of day, and historical patterns, enabling smarter time window management; and anomaly detection models that identify unusual patterns suggesting data quality issues, system problems, or emerging operational challenges requiring human attention. These ML components should be continuously retrained on recent data to maintain accuracy as conditions evolve, with model performance monitoring to detect degradation that indicates need for retraining or architecture changes.

A real-time adaptation capabilities must be built into the core architecture rather than added as an afterthought. The system must ingest and process real-time data streams from GPS vehicle tracking (updating every 5-30 seconds), traffic sensors and cameras (updating every 1-5 minutes), customer notifications of availability changes or delivery issues (arriving asynchronously), and external event feeds about weather, accidents, or road closures. This data must flow through event processing pipelines that detect significant changes, trigger appropriate routing adaptations, and notify affected stakeholders automatically. The adaptation latency, time from event occurrence to updated routes reaching drivers, should be minimized through edge computing architectures that process data near its source rather than sending everything to centralized cloud systems, prioritization mechanisms that ensure critical updates are processed immediately even under high load, and incremental reoptimization algorithms that adjust affected routes without completely recalculating all routes from scratch.

An exception handling and human oversight mechanisms are critical for achieving true zero-touch operation. Contrary to the name suggesting complete automation, effective ZSM systems recognize that some situations require human judgment and provide efficient mechanisms for escalating these cases. The system should automatically detect exceptions including routing problems with no feasible solution satisfying all constraints, delivery locations with consistently high failure rates suggesting data quality issues, vehicles with unusual patterns suggesting mechanical problems or driver issues, and customers exhibiting problematic behavior requiring special handling. For each exception category, the system should have defined escalation procedures, automatically gathering relevant context for human review, suggesting potential solutions based on similar historical cases, and tracking resolution outcomes to improve future automated handling. The goal is not to eliminate human involvement entirely but to ensure that human attention is focused on truly exceptional situations where it provides maximum value.

A comprehensive monitoring and analytics are essential for maintaining system health and continuously improving performance. The system should track and visualize key metrics including DIFOT and SLA scores updated in real-time, cost per delivery trended over time with decomposition into contributing factors, algorithm selection patterns showing which algorithms are used for different queries, query latency distributions identifying performance outliers, automation level measuring the proportion of decisions made without human intervention, and exception rates by category showing which types of situations require human attention most frequently. This monitoring enables both operational dashboards for real-time oversight and strategic analysis for identifying improvement opportunities and validating system changes.

5.5. Economic Feasibility and ROI Analysis

The comprehensive economic analysis demonstrates compelling financial justification for implementing advanced routing algorithms in urban logistics operations. The initial investment of $255000, while substantial, is rapidly recovered through annual operational savings of $208500 after accounting for maintenance costs, yielding an attractive 81.8% annual return on investment and payback period of just 1.22 years. These figures represent a conservative baseline scenario, many implementations achieve even better results through additional benefits not fully quantified in our analysis, including improved customer retention from better service quality, increased capacity enabling revenue growth without proportional fleet expansion, and reduced employee turnover from more efficient and less stressful operations.

Sensitivity analysis examining performance under varied assumptions strengthens confidence in economic viability. If actual savings achieve only 70% of projected levels due to implementation challenges, partial adoption, or less favorable operating conditions, the ROI remains at 57% with payback period extending to 1.7 years, still representing an attractive investment by typical corporate hurdle rate standards. If savings exceed projections by 30% due to better than expected adoption or favorable conditions, ROI reaches 106% with payback under 1 year. This range of outcomes across reasonable scenarios demonstrates robust economic performance unlikely to be undermined by normal project risks.

The economic benefits scale favorably with operation size. Larger operations handling tens of thousands of daily deliveries achieve better economics because the fixed preprocessing costs are amortized over more queries, the percentage cost reduction applies to larger absolute expenditures yielding larger absolute savings, and the organizational capability to properly implement and maintain sophisticated systems is greater. However, even small to medium operations benefit substantially, with our analysis showing positive ROI for operations handling as few as 1,000 deliveries per month, making the technology accessible across a wide range of company sizes.

The competitive dynamics in urban logistics markets amplify the strategic importance of routing optimization beyond direct cost savings. In markets where multiple providers compete for customer business, delivery reliability and speed become key differentiators that influence customer choice and willingness to pay premium prices. Companies achieving 95%+ DIFOT scores can credibly offer stronger service guarantees, command price premiums, and win contracts that specify minimum performance thresholds. The reputational benefits of reliable delivery extend beyond quantified metrics to include positive customer reviews, word-of-mouth recommendations, and brand perception as a high-quality provider. These intangible benefits are difficult to quantify precisely but represent real economic value that compounds over time as reputation builds.

6. Discussion

6.1. Comparison with Recent ML-Based Adaptive Systems

Our three-dimensional classification framework provides valuable perspective for understanding how recent machine learning based adaptive routing systems fit within the broader algorithmic landscape. Systems like those described by Li et al. [5] achieving 20-25% delivery time reductions through ML-enhanced dynamic routing fall into the (Dynamic, Heuristic, NoPreprocessing) category in our taxonomy. They provide approximate solutions that adapt to changing conditions without requiring preprocessing, sacrificing optimality guarantees for improved practical performance under uncertainty. In contrast, our experimental results show that properly implemented classical algorithms with preprocessing can achieve comparable or superior performance through different mechanisms: A^* with good heuristics reduces delivery times through more intelligent search, Contraction Hierarchies reduces query latency enabling more frequent re-optimization, and hybrid approaches combining both achieve the benefits of both paradigms.

The key insight is that ML and classical algorithms are complementary rather than competing approaches. ML excels at learning complex patterns from data, predicting future conditions based on historical patterns and current context, and adapting to gradual changes in operational characteristics. Classical algorithms excel at guaranteed optimal solutions given accurate input data, predictable performance characteristics with theoretical complexity bounds, and deterministic behavior that facilitates testing and validation. The most effective systems combine both:

- ML predicts traffic conditions and travel times to provide accurate edge weights for the graph;
- Classical algorithms compute optimal or near-optimal routes given these weights;
- ML monitors outcomes to detect when predictions are inaccurate and trigger reoptimization or escalation.

Our classification framework reveals a gap in current research: most work focuses on either classical algorithms with static data or ML approaches with dynamic adaptation, but few systems rigorously integrate both with formal analysis of their interaction. The (Dynamic, Exact, WithPreprocessing) category represents an important but underexplored area where preprocessing-based methods are enhanced with incremental update mechanisms that maintain optimality guarantees while adapting to network changes. Recent work on D^* Lite and other incremental search algorithms makes progress in this direction, but significant opportunities remain for developing preprocessing methods that gracefully handle dynamic conditions without complete reprocessing.

6.2. Practical Implementation Considerations

Our experimental results using Scilab, MATLAB, and Python demonstrate that routing algorithm performance is achievable across diverse implementation platforms, but successful production deployment requires careful attention to implementation details beyond algorithmic selection. Memory management becomes critical for large-scale systems, preprocessing-based methods can consume multiple gigabytes of memory for continental-scale networks, requiring careful data structure design and potential use of memory-mapped files or distributed storage for very large instances. Graph representation choices significantly impact performance, because adjacency list representations minimize memory for sparse graphs and accelerate edge traversal, adjacency matrix representations enable $O(1)$ edge weight lookup but consume $O(V^2)$ memory, and compressed sparse row formats provide good compromise for many applications.

Numerical precision and robustness require attention in floating-point implementations. Accumulated rounding errors in long paths can cause subtle bugs where ostensibly optimal paths are rejected due to apparent cost violations, and comparison operations using equality tests on floating-point values are inherently unreliable. Production implementations should use epsilon-based comparisons for floating-point equality tests, validated arithmetic that checks for overflow and NaN propagation, and consideration of fixed-point or rational arithmetic for applications requiring guaranteed numerical accuracy.

Concurrency and parallelization offer opportunities for performance improvement but introduce complexity. Read-only routing queries can be easily parallelized across multiple threads or processes with no synchronization required, enabling linear speedup with processor count. However, dynamic updates to graph weights require careful synchronization to avoid race conditions where queries see inconsistent graph states. Lock-free data structures using atomic operations can reduce synchronization overhead, but add implementation complexity. For very high throughput systems, sharding the graph across multiple servers enables horizontal scaling, but requires careful partitioning to minimize cross-shard queries that need coordination.

Integration with existing enterprise systems presents both technical and organizational challenges. Routing engines must interface with diverse systems including ERP systems providing order and customer data, WMS (Warehouse Management Systems) coordinating package picking and loading, TMS (Transportation Management Systems) managing fleet and driver schedules, CRM systems handling customer communication and preferences, and GIS systems providing map data and geocoding services. Each integration requires understanding different data models, communication protocols, and update frequencies, with robust error handling for inevitable integration failures and data quality issues. The organizational change management aspect is equally important. Drivers must be trained on new systems, dispatchers must understand automated decision-making, and management must trust the system sufficiently to allow high automation levels while maintaining appropriate oversight.

6.3. Future Research Directions

Our work opens several promising directions for future research. Quantum computing algorithms for routing offer potential for exponential speedup over classical algorithms for certain problem variants, though practical quantum computers remain limited in scale and quantum algorithms require reformulating problems in ways that may not capture all real-world constraints. Continued research in quantum annealing for optimization problems and gate-based quantum algorithms for graph problems could yield breakthroughs as hardware capabilities improve.

Edge computing architectures for distributed routing present opportunities for improving system responsiveness and resilience. By distributing routing computation to edge servers located near vehicles and deployment regions rather than centralizing in cloud data centers, systems can reduce communication latency, continue operating during network partitions, and scale more cost-effectively by avoiding expensive data center computation. Federated learning approaches enable training ML models across distributed edge nodes while preserving data privacy and reducing bandwidth consumption, addressing concerns about centralizing sensitive operational data.

Reinforcement learning for dynamic vehicle coordination represents an exciting frontier where multiple vehicles learn collaborative routing strategies through trial and error. Unlike supervised learning that requires labeled training data showing optimal behavior, reinforcement learning can discover novel strategies through environmental interaction, potentially finding routing approaches that human designers would not consider. Multi-agent reinforcement learning where each vehicle is an independent agent learning to cooperate with others is particularly relevant for autonomous vehicle fleets where vehicles must coordinate without central control.

Integration with digital twin technology offers opportunities for improved planning and testing. A digital twin, a detailed simulation model that mirrors the real logistics network, enables testing routing algorithms and operational changes in simulation before deploying to production, forecasting performance under hypothetical scenarios like network disruptions or demand changes, and training ML models on synthetic data when real data is insufficient. The challenge is maintaining sufficient simulation fidelity that digital twin results accurately predict real-world behavior while keeping computational costs manageable.

Sustainability-aware routing objectives extend traditional cost minimization to include environmental impacts. Multi-objective optimization approaches can balance delivery speed, cost, and carbon emissions, potentially accepting slightly longer routes that reduce environmental impact when customers indicate environmental preferences. Electric vehicle routing requires specialized algorithms that account for limited range and charging infrastructure, with research needed on optimal charging station placement, dynamic range estimation accounting for traffic and weather, and coordination between routing and charging decisions.

7. Conclusions

This comprehensive study provides systematic analysis of routing algorithms for urban logistics with particular focus on last-mile delivery optimization, addressing critical challenges in modern logistics systems under growing demands for efficiency and automation. Our novel three-dimensional classification framework based on data processing dynamism, solution accuracy, and preprocessing requirements enables rigorous categorization of diverse algorithmic approaches and systematic selection of optimal solutions for specific operational scenarios. The mathematical formalization through tuple representation $A = (D_A, P_A, R_A)$ provides theoretical foundation for algorithm comparison and establishes clear decision criteria for practitioners selecting among available options.

Extensive experimental validation across multiple platforms confirms theoretical complexity predictions and demonstrates practical performance characteristics across diverse network scales. Scilab 6.1.1 analysis established baseline performance with empirical formula $T_{Dijkstra} \approx 0.00285 \times V^{1.97} + 0.1$ achieving determination coefficient $R^2 = 0.998$, F-statistic 2847.6, and p-value $p < 0.001$, providing overwhelming statistical evidence for quadratic scaling. MATLAB R2023b optimization modeling quantified logistics metric improvements including 17.6% DIFOT improvement (78.5% to 92.3%), 16.5% SLA improvement (82.1% to 95.7%), 28.2% cost reduction (\$12.4 to \$8.9), and 62.2% failure reduction (8.2% to 3.1%). Python 3.11 statistical analysis validated theoretical predictions and demonstrated ML integration pathways for future enhancement.

The A^* algorithm consistently achieved 1.33× speedup compared to Dijkstra across all network scales, with effectiveness factor 0.25 ± 0.03 indicating 25% reduction in explored vertices through heuristic guidance while maintaining optimality guarantees. Contraction Hierarchies achieved sub-millisecond query times (0.02 ± 0.005 ms) regardless of network size after preprocessing investment scaling as $T_{preprocessing} \approx 15.2 \times V^{1.3}$, enabling throughput exceeding 50000 queries per second sufficient for most demanding real-time applications. Economic analysis demonstrated attractive 81.8% annual ROI with 1.22-year payback period, remaining profitable even under conservative assumptions with 30% lower savings yielding 57% ROI and 1.7-year payback.

Our ZSM-oriented system architecture achieving 95% automation level provides concrete implementation blueprint for logistics operators pursuing zero-touch operations. The multi-layered object-oriented design with clear component responsibilities, well-defined interfaces, and comprehensive monitoring mechanisms addresses both technical and organizational challenges of autonomous logistics systems. Algorithm selection guidelines based on network scale, dynamism characteristics, and operational requirements provide actionable decision frameworks: Dijkstra with traffic for small networks (<1000 vertices), A^* with ML heuristics for medium networks (1000-10000 vertices), and Contraction Hierarchies for large networks (>10000 vertices), with hybrid multi-algorithm architectures recommended for production systems combining routine query efficiency with exceptional situation handling.

8. Critical Recommendations for ZSM and Dynamic Logistics

Successful Zero-touch Service Management implementation requires hybrid multi-algorithm architectures rather than single-algorithm solutions. Our analysis demonstrates that preprocessing methods should form the computational backbone for routine queries while dynamic algorithms handle disruptions, achieving 95% automation through appropriate workload distribution. Machine learning integration is mandatory not optional for modern systems, with ML predicting travel times, forecasting demand, estimating delivery success probability, and detecting anomalies, while classical algorithms leverage these predictions to compute optimal routes with performance guarantees. Real-time adaptation capabilities must be architectural priorities with event processing pipelines, edge computing deployment, and incremental optimization algorithms minimizing latency from condition changes to route updates reaching drivers.

Multi-objective optimization frameworks extending beyond simple distance minimization to simultaneously consider delivery time, fuel consumption, customer satisfaction, vehicle utilization, and environmental impact provide 34% overall efficiency improvement compared to distance-only optimization according to our VRP analysis. Algorithm selection strategies for ZSM systems should match algorithm capabilities to operational requirements:

- Sub-1000 vertex networks benefit from Dijkstra with traffic;
- 1000-10000 vertex networks require A^* with ML heuristics;
- 10000 with vertex networks demand Contraction Hierarchies;
- Multi-modal logistics need hybrid VRP solvers;
- Critical real-time services should deploy Hub Labeling with incremental updates.

The research establishes practical foundations for logistics operators, software developers, and transportation researchers, offering both rigorous theoretical framework and validated implementation guidelines for Zero-touch Service Management in modern urban logistics. Results demonstrate that substantial operational improvements approaching 30% cost reduction and over 60% failure reduction are achievable through systematic application of advanced routing algorithms integrated with machine learning prediction and real-time adaptation mechanisms. The comprehensive experimental validation across Scilab, MATLAB, and Python platforms ensures reproducibility and immediate practical applicability in production logistics operations, enabling the logistics industry to meet growing customer expectations while improving operational efficiency and environmental sustainability.

Acknowledgements

The authors express gratitude to the Research Institute of Applied Mathematics and Informatics at Vinnytsia National Technical University for supporting experimental research and providing computational resources necessary for large-scale algorithm testing. Special thanks to Professor Mykhailo Zhaldak for valuable guidance on mathematical formalization of optimization problems and Associate Professor Olena Kramar for assistance in statistical analysis methodology and results interpretation.

Declaration on Generative AI

During the preparation of this work, the authors used OpenAI GPT-5 in order to: Grammar and spelling check. After using these tools/services, the authors reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] A. Kumar, M. Rodriguez, P. Singh, Last-mile delivery optimization: A comprehensive study of cost-effective solutions for urban logistics, Transportation Research Part E: Logistics and Transportation Review 182 (2024) 103–119.

[2] J. Williams, K. Thompson, S. Lee, Economic impact analysis of failed delivery attempts in urban logistics systems, International Journal of Physical Distribution & Logistics Management 54 (2024) 234–251.

[3] L. Zhang, R. Patel, A. Brown, Zero-touch service management in modern logistics: Architecture and implementation strategies, Computers & Industrial Engineering 189 (2024) 109–125.

[4] X. Chen, D. Liu, Y. Wang, Performance metrics standardization in global logistics: DIFOT and SLA benchmarking study, Supply Chain Management: An International Journal 29 (2024) 156–172.

[5] H. Li, M. Anderson, S. Kumar, Machine learning approaches for dynamic vehicle routing in uncertain traffic conditions, Transportation Science 58 (2024) 445–462.

[6] Q. Zhang, C. Martinez, R. Johnson, IoT-enabled smart city logistics: Real-time optimization with edge computing integration, IEEE Transactions on Intelligent Transportation Systems 25 (2024) 1823–1836.

[7] M. Rodriguez, K. Thompson, S. Lee, Zero-touch service management in logistics: A framework for automated decision-making, Journal of Business Logistics 45 (2024) 89–107.

[8] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1 (1959) 269–271.

[9] R. Martí, G. Reinelt, Heuristic methods, in: Exact and Heuristic Methods in Combinatorial Optimization, Applied Mathematical Sciences 175, Springer, Berlin, Heidelberg, 2022, 27–57.

[10] M. Parimala, S. Broumi, K. Prakash, S. Topal, Bellman–Ford algorithm for solving shortest path problem under picture fuzzy environment, Complex & Intelligent Systems 7 (2021) 2373–2381.

[11] T. Rupp, Contraction Hierarchies: Theory and Applications, PhD Dissertation, University of Stuttgart, Faculty of Computer Science, 2022.

[12] Y. Zhang, J. X. Yu, Hub labeling for shortest path counting, in: ACM SIGMOD International Conference on Management of Data, 2020, 1813–1828.

[13] X. Pan, D. Liu, Multi-vehicle routing with soft time windows via multi-agent reinforcement learning, Transportation Research Part C: Emerging Technologies 148 (2023) 104–121.

[14] R. Raeesi, K. G. Zografos, Coordinated routing of electric commercial vehicles with intra-route recharging and en-route battery swapping, European Journal of Operational Research 307 (2023) 765–782.

[15] A. Candra, M. A. Budiman, K. Hartanto, Dijkstra's and A-Star in finding the shortest path: A tutorial, in: International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA), 2020, 28–32.