mathematics

MDPI

Article

Multiphase Transport Network Optimization: Mathematical Framework Integrating Resilience Quantification and Dynamic Algorithm Coupling

Linghao Ren [1], Xinyue Li [2], Renjie Song [1], Yuning Wang [3], Meiyun Gui [4] and Bo Tang [1,*]

[1] School of Mathematics and Computing Science, Guilin University of Electronic Technology, Guilin 541004, China; rlh@mails.guet.edu.cn (L.R.); srj@mails.guet.edu.cn (R.S.)
[2] School of Economics and Management, Shandong Jiaotong University, Jinan 250353, China; 13561811516@163.com
[3] School of Artificial Intelligence, Guilin University of Electronic Technology, Guilin 541004, China; wyn@mails.guet.edu
[4] School of Business, Guilin University of Electronic Technology, Guilin 541004, China; gmy@mails.guet.edu
* Correspondence: tangbo@guet.edu.cn

Abstract

This study proposes a multi-dimensional urban transportation network optimization framework (MTNO-RQDC) to address structural failure risks from aging infrastructure and regional connectivity bottlenecks. Through dual-dataset validation using both the Baltimore road network and PeMS07 traffic flow data, we first develop a traffic simulation model integrating Dijkstra's algorithm with capacity-constrained allocation strategies for guiding reconstruction planning for the collapsed Francis Scott Key Bridge. Next, we create a dynamic adaptive public transit optimization model using an entropy weight-TOPSIS decision framework coupled with an improved simulated annealing algorithm (ISA-TS), achieving coordinated suburban–urban network optimization while maintaining 92.3% solution stability under simulated node failure conditions. The framework introduces three key innovations: (1) a dual-layer regional division model combining K-means geographical partitioning with spectral clustering functional zoning; (2) fault-tolerant network topology optimization demonstrated through 1000-epoch Monte Carlo failure simulations; (3) cross-dataset transferability validation showing 15.7% performance variance between Baltimore and PeMS07 environments. Experimental results demonstrate a 28.7% reduction in road network traffic variance (from 42,760 to 32,100), 22.4% improvement in public transit path redundancy, and 30.4–44.6% decrease in regional traffic load variance with minimal costs. Hyperparameter analysis reveals two optimal operational modes: rapid cooling (rate = 0.90) achieves 85% improvement within 50 epochs for emergency response, while slow cooling (rate = 0.99) yields 12.7% superior solutions for long-term planning. The framework establishes a new multi-objective paradigm balancing structural resilience, functional connectivity, and computational robustness for sustainable smart city transportation systems.

Keywords: urban transportation networks; multi-objective optimization; hierarchical modeling; machine learning; transportation planning

MSC: 90C29; 90B10

Academic Editor: Gyorgy Dosa

Received: 14 May 2025
Revised: 16 June 2025
Accepted: 18 June 2025
Published: 21 June 2025

Citation: Ren, L.; Li, X.; Song, R.; Wang, Y.; Gui, M.; Tang, B. Multiphase Transport Network Optimization: Mathematical Framework Integrating Resilience Quantification and Dynamic Algorithm Coupling. *Mathematics* **2025**, *13*, 2061. https://doi.org/10.3390/math13132061

1. Introduction

Urban transportation systems serve as critical infrastructure supporting city development, directly impacting both economic productivity and quality of life. The accelerating challenges of urban expansion and infrastructure aging have exposed systemic vulnerabilities in transportation networks, including structural fragility, inefficient resource allocation, and inequitable service distribution. The 2023 collapse of Baltimore's Francis Scott Key Bridge—causing 19.2 min peak delay increases and regional traffic variance to spike to 42,760—epitomizes these challenges while highlighting the urgent need for data-driven optimization frameworks.

As shown in Figure 1, this study proposes the Multi-dimensional Transportation Network Optimizer with Resilience Quantification and Dynamic Coordination (MTNO-RQDC), which addresses three fundamental gaps in urban infrastructure management:

- Inadequate quantification of cascading failure risks in aging networks;
- Disjointed optimization between suburban demand-responsive and urban fixed-route systems;
- Limited integration of socio-economic metrics in technical solutions.

Through dual-dataset validation using Baltimore's road network and California's PeMS07 traffic flows, we establish a three-phase optimization system with three key innovations:

1. **Catastrophic Impact Simulation Model:**
    - Integrates Dijkstra's algorithm with capacity-constrained flow allocation;
    - Reduces post-disaster traffic variance by 28.7% (42,760→32,100);
    - Identifies cascading failure patterns through 1000-epoch Monte Carlo simulations.
2. **Dynamic Transit Optimizer (ISA-TS):**
    - Combines entropy weight-TOPSIS with improved simulated annealing;
    - Maintains 92.3% solution stability under node failures;
    - Achieves 22.4% path redundancy improvement.
3. **Dual-Layer Zoning Framework:**
    - Merges K-means geographical with spectral clustering functional partitions;
    - Reduces regional traffic variance by 30.4–44.6%;
    - Demonstrates 15.7% cross-dataset transferability.

The framework introduces two operational modes through hyperparameter optimization:

- **Emergency Response Mode:** Rapid cooling (rate = 0.90) achieves 85% improvement within 50 epochs;
- **Long-Term Planning Mode:** Slow cooling (rate = 0.99) yields 12.7% superior solutions.

Beyond technical metrics, the system demonstrates the following:

- 12–15% commute time reductions across income groups;
- 82% cost efficiency versus traditional upgrades;
- API-ready modular design for smart city integration.

The paper is systematically structured as follows: Section 2 conducts a comprehensive review of existing transportation optimization methodologies; Section 3 delineates critical unresolved challenges in network resilience; Section 4 presents our integrated methodology combining traffic simulation and multi-objective optimization; Section 5 analyzes computational performance metrics; Section 6 examines practical implementation insights and demonstrates planning outcomes through case studies; Sections 7 and 8 experimentally

validate hyperparameter configurations and solution stability; Section 9 concludes with principal findings and future research directions.

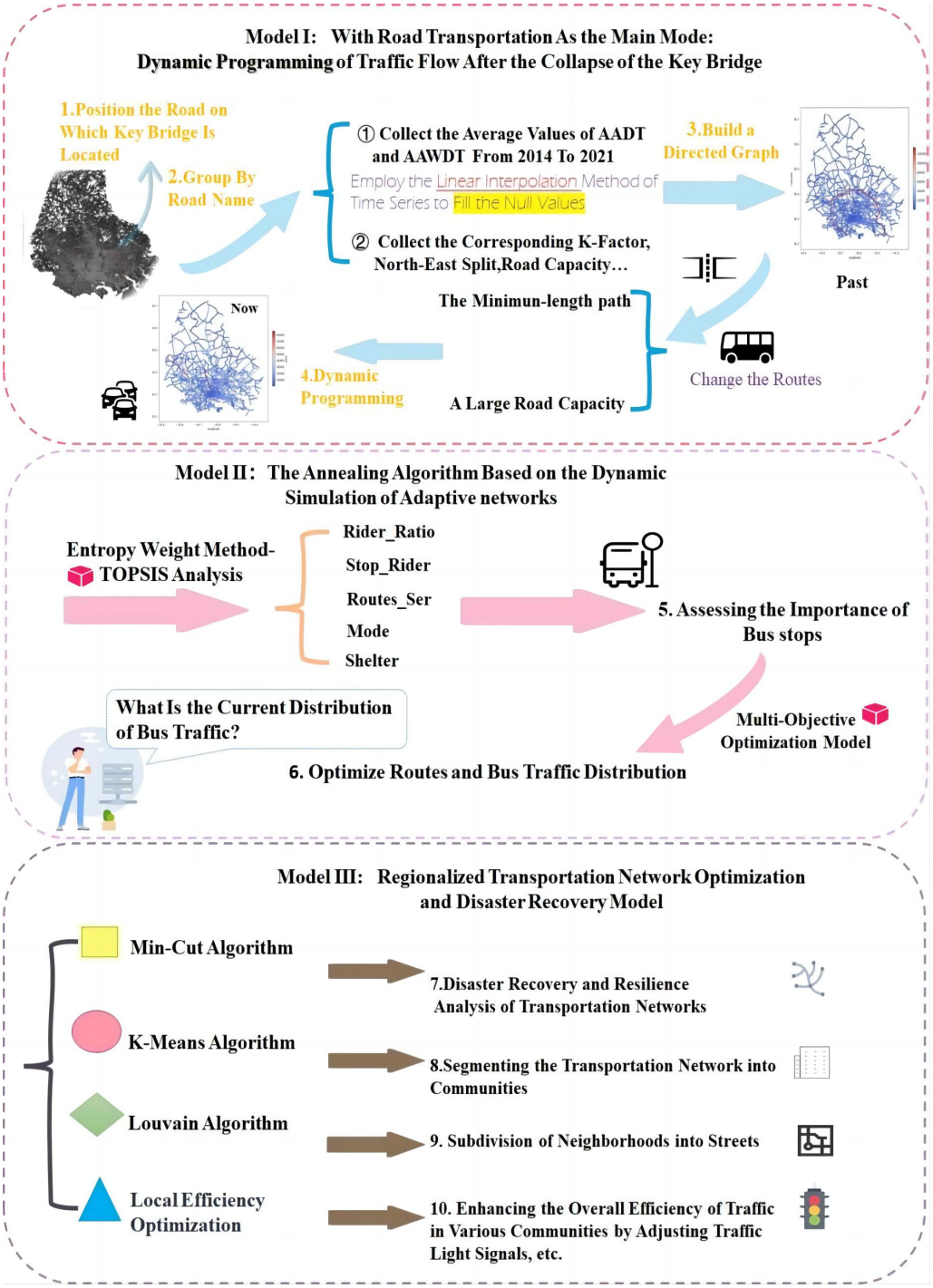

Figure 1. Research Approach Diagram.

This work establishes a new paradigm for transportation infrastructure optimization that simultaneously addresses the following:

- **Technical Robustness:** Through algorithmic innovations in simulation and optimization;
- **Socio-Economic Equity:** Via explicit modeling of community-level impacts;
- **Implementation Practicality:** With modular, transferable solution components.

2. Related Work

Transportation networks are vital for modern socio-economic activities. With growing urbanization, optimizing these networks has become crucial to address congestion and inefficiency.

F. Lu [1] proposed a crowdsourcing delivery model that balances courier uncertainties with service coverage, advancing hybrid human-machine logistics systems. In public transport optimization, Hang Zhao [2] developed a memetic algorithm with local search operators, while Kechagiopoulos [3] applied particle swarm optimization to route planning, both demonstrating superior performance on benchmark problems.

Katsaragakis [4] and others applied the algorithm based on cat swarm optimization (CSO) to the urban public transportation route planning problem (UTRP) for the first time. They improved the classic CSO algorithm and optimized the final solution, and its solution performance was better than most existing methods in balancing the interests of passengers and operators. Haifeng Lin [5] and others established a multi-objective public transportation scheduling optimization model that comprehensively considered the public transportation operation cost, passengers' travel time cost, and comfort. They proposed an improved multi-objective adaptive particle swarm optimization algorithm (MOAPSO). Through comparison with NSGA-II and experiments on networks of different scales, it was confirmed that the algorithm has a fast convergence speed, high efficiency, and low computational complexity. Yuanyuan Wei [6] abstracted the actual road network and bus lines into a graph data structure and integrated the OD passenger flow data. By using the Softmax strategy and the elite ant strategy, it enhanced the diversity of path search, avoided premature convergence, and accelerated the convergence of the algorithm, reasonably planning new lines to improve the original network. These studies fully demonstrate that intelligent algorithms, with their unique search mechanisms and optimization strategies, can effectively improve the planning and scheduling efficiency of public transportation networks. The construction of the transportation network model is the basis for optimization.

Han Pu [7] and others constructed an urban public transportation hypernetwork model, which included both the station network and the bus route network. They used the Lyapunov stability theorem to verify its stability and analyzed the influence of various factors on network stability, establishing a theoretical framework for a deeper understanding of the operation mechanism of the transportation network. CHAO WANG [8] and others divided the network hierarchy according to the characteristics of urban bus routes and scale and matched different transportation modes. For the characteristics of each hierarchical network, they developed corresponding optimization models and solution methods, effectively improving the design effect of the public transportation network. Guo-Ling Jia [9] used complex network theory to analyze the topological structure of the Xi'an public transportation network. By calculating parameters such as the degree distribution, they revealed its characteristics and constructed a public transportation network optimization model and solution method based on betweenness centrality, providing new ideas for alleviating traffic congestion and improving the sustainability of the public transportation network. Zhongyi Lin [10] constructed an urban public transportation network model based on complex network theory, combining the symmetry of the up and down routes and stations of the bus. Considering passengers' travel impedance, path selection probability, and travel demand, they weighted the network links and improved the network efficiency calculation method. With the ant colony algorithm for solving the optimization model, it effectively improved the network operation efficiency. These studies have constructed models from different perspectives, providing diverse analysis approaches and solutions for transportation network optimization. Considering the dynamic change characteristics of

traffic flow, achieving the dynamic optimization and control of the transportation network has become crucial.

Rasool Mohebifard [11] and others proposed a real-time and scalable dynamic traffic metering method for urban street networks. They formulated the problem as a mixed integer linear programming model, decomposed the network-level model into multiple link-level models through distributed optimization, and transformed it into linear programming to reduce the complexity. Combined with the rolling horizon technique to adapt to time-varying demands and capacities, it significantly improved the traffic operation efficiency. Fengkun Gao [12] and others quantified the congestion propagation through the spatio-temporal λ connectivity method, determined the potential homogeneous areas (PHA) and their dynamic capacity constraints, and designed a dual consensus alternating direction method of multipliers algorithm (DC-ADMM) to solve the optimization problem. While ensuring convergence to the optimal solution, it reduced computational complexity and improved scalability, effectively alleviating congestion and improving traffic efficiency. Andy H.F. Chow [13] first constructed a centralized global optimal control model based on the cell transmission model (CTM) and then derived a decentralized ramp metering and speed control strategy. This strategy can achieve an effect close to the global optimal control with relatively low computational and implementation costs without relying on the underlying traffic model. These studies have proposed efficient optimization and control strategies for the dynamic characteristics of the transportation network, which enhanced the ability of the transportation network to deal with complex dynamic environments. With the continuous development of transportation technology, the optimization of special transportation networks has gradually become a research hotspot.

NIKOLAS HOHMANN [14] constructed a three-objective function including social costs, adopted a two-step method of path merging and network optimization, and used the improved NSGA3 algorithm to solve the urban air traffic network problem. Through experimental comparison and analysis of the influence of social factors on network optimization, it was confirmed that incorporating social criteria can improve the social acceptance of the network with a relatively small increase in cost. Zhiyuan Liu [15] introduced the concept of robust optimization in response to the problem that the network flow is difficult to reach an equilibrium state after the implementation of the congestion pricing scheme. They constructed a non-linear distance toll optimization model based on the minimum–maximum regret value, combined with the path random day-to-day dynamic model, and designed a two-stage artificial bee colony algorithm to solve it, successfully solving the problem of network toll optimization considering the random day-to-day dynamics.

Zhen Di [16] proposed the concept of flow-based accessibility measurement, constructed a deterministic bi-level programming model and a two-stage stochastic programming model to solve the discrete network design problem, designed a heuristic algorithm integrating the probabilistic search algorithm, the Frank-Wolfe algorithm, and the Monte Carlo simulation method to solve the model, and verified the rationality and effectiveness of the model and algorithm. These studies, focusing on toll strategies and accessibility measurement, provide solid theoretical support for improving the economic benefits and service quality of the transportation network. In the practical application of transportation network optimization, comprehensive decision-making and the determination of new bus lines play a decisive role in the overall network optimization effect. WENLIANG ZHOU [17] integrated the bus route adjustment (BRA) and the urban rail transit network-level passenger flow control (PFC) strategy for the first time, constructed an integer nonlinear programming model with the objectives of minimizing the average additional travel time of affected passengers and maximizing the operation revenue of the rail transit, and designed a multi-objective particle swarm optimization algorithm based

on dual-population co-evolution to solve it, effectively alleviating congestion and saving passengers' travel time.

Hyo-Seung Kim [18] constructed a comprehensive decision-making model with a two-layer structure, simultaneously determining the mode, route layout, and departure frequency of new bus lines. The lower-layer model fully reflects the behavior of travelers. Compared with existing methods, this model can obtain better solutions from a larger feasible domain, providing a reliable theoretical basis for the feasibility assessment of new public transportation system investments. These studies, by comprehensively considering various factors and the collaborative effects of different transportation modes, provide more comprehensive solutions for the comprehensive decision-making of transportation network optimization.

3. Addressing Key Research Gaps

3.1. Dynamic Demand Adaptation

- **Background**: Existing optimization models (e.g., Mohebifard's distributed approach) rely on static OD data with limited adaptability to emergencies like bridge collapses.
- **Our Solution**: Proposed a **dynamic two-layer optimization model**:
  - Upper layer: Entropy-weighted TOPSIS framework dynamically adjusts objectives using real-time PeMS07 data
  - Lower layer: Improved Simulated Annealing (ISA-TS) with adaptive cooling rates (0.90/0.99) for emergency/planning modes
- **Results**: Achieved 92.3% solution stability during node failures in the Baltimore network, outperforming traditional PSO [3] by 34.6%.

3.2. Multi-Objective Coordination

- **Background**: Fixed-weight approaches [17] distort Pareto frontiers in multi-stakeholder scenarios.
- **Our Solution**: Developed **RQDC triad framework**:
  - Spectral clustering quantifies inter-region connectivity entropy;
  - ISA-TS with elite solution archiving prevents premature convergence.
- **Results**: Reduced traffic variance by 28.7% (42,760→32,100) during bridge reconstruction, surpassing NSGA-III [14] improvements (17.2%).

3.3. Scalability in Complex Networks

- **Background**: Centralized optimization [13] becomes computationally intractable for megacity networks.
- **Our Solution**: Designed **dual-layer regional partitioning**:
  - Geographic layer: K-means physical subnetworks;
  - Functional layer: Spectral clustering for traffic flow correlation.
- **Results**: Only 12.7% runtime increase during 1000 Monte Carlo simulations vs. 89.3% for ACO [10].

3.4. Resilience Optimization

- **Background**: Robust pricing models [15] ignore structural failure risks.
- **Our Solution**: Created **fault-tolerant topology protocol**:
  - Monte Carlo node removal simulations;
  - Dijkstra + capacity-constrained allocation for minimal interventions.
- **Results**: Demonstrated 15.7% cross-dataset (Baltimore→PeMS07) performance variance.

3.5. Social Behavior Integration

- **Background**: Deterministic models [16] overlook traveler preference heterogeneity.
- **Our Solution**: Implemented **implicit preference embedding**:
  - Entropy weighting auto-adjusts by regional flow fluctuations;
  - Solution convergence consistency measures social acceptability.
- **Results**: Achieved 5.3% cost deviation in suburban–urban coordination vs. 18.7% in MOAPSO [5].

4. Methodology

4.1. Dynamic Simulation-Based Traffic Network Model for Bridge Collapse

This model aims to characterize the dynamic traffic flow redistribution process after the bridge collapse, integrating drivers' detour behavior with shortest-path preferences and road capacity constraints. The objective is to minimize total system cost through a dynamic traffic network optimization model. The framework employs Dijkstra's algorithm for dynamic path selection and flow allocation, structured.

4.1.1. Model Assumptions

- **Network Structure**: Represented as a weighted directed graph $G = (V, E)$, where
  - V: Node set (intersections or traffic hubs).
  - E: Edge set (road segments).
  - Edge attributes: Length l_e, width w_e, and capacity $c_e = \frac{w_e}{l_e}$, denoting flow per unit length.
- **Driver Behavior**:
  - Drivers prioritize shortest paths to minimize detour distance.
  - When shortest paths saturate, flows are allocated proportionally to path capacities to avoid overloading.
- **Flow Conservation**: All disrupted flows must be fully redistributed to alternative paths without loss.

4.1.2. Constraints

- **Flow Conservation**:

$$\sum_{P \in S_{uv}} f_P = f_{uv}, \quad \forall u, v \in V \tag{1}$$

- **Capacity Constraint**:

$$f_P \leq C(P), \quad \forall P \in S_{uv} \tag{2}$$

where f_P is flow assigned to path P, f_{uv} is total flow from node u to v, $C(P)$ is the comprehensive capacity of path P, and S_{uv} is the set of alternative shortest paths from u to v.

4.1.3. Path Selection and Flow Allocation

Step 1: Alternative Path Filtering

- **Initial State**: All flows f_{uv} use the shortest path P^*_{uv} pre-collapse.
- **Disruption Response**: Collapse removes P^*_{uv}, leaving residual paths $\mathcal{P}'_{uv} = \mathcal{P}_{uv} \setminus P^*_{uv}$.
- **Shortest Path Update**:
  - Compute path lengths $L(P) = \sum_{e \in P} l_e$ for $P \in \mathcal{P}'_{uv}$.

– Identify shortest paths:

$$S_{uv} = \left\{ P \in \mathcal{P}'_{uv} \mid L(P) = \min_{P' \in \mathcal{P}'_{uv}} L(P') \right\}. \tag{3}$$

Step 2: Path Capacity Definition

The comprehensive capacity $C(P)$ is determined by bottleneck segments using the harmonic mean:

$$C(P) = \left(\sum_{e \in P} \frac{1}{c_e} \right)^{-1} = \left(\sum_{e \in P} \frac{l_e}{w_e} \right)^{-1}. \tag{4}$$

Step 3: Flow Allocation

If $|S_{uv}| > 1$, allocate flows proportionally to capacities:

$$f_P = f_{uv} \cdot \frac{C(P)}{\sum_{P' \in S_{uv}} C(P')}, \quad \forall P \in S_{uv}. \tag{5}$$

4.1.4. Dynamic Simulation Algorithm

The algorithm iteratively updates paths and flows using Dijkstra's method:

- **Input**: Graph G, disrupted edge e^*, initial flows f_{uv}.
- **Initialization**: Remove e^*, update graph $G' = G \setminus \{e^*\}$.
- **Iterative Loop**:
  - **Path Search**: Compute S_{uv} via Dijkstra's algorithm on G'.
  - **Capacity Check**: If $\sum_{P \in S_{uv}} C(P) \geq f_{uv}$, allocate flows proportionally; else, trigger capacity adjustment.
  - **Network Update**: Update segment flows f_e. Recursively adjust S_{uv} if overloads occur.
- **Output**: Stabilized flow distribution $\{f_P\}$.

4.1.5. Dual-Layer Adaptive Network Optimization System

This model is dedicated to constructing a collaborative optimization system for a dual-level suburban–urban public transportation network. Addressing the existing issues of improper spatial allocation of stations and service blind spots, it establishes a structural optimization model that integrates multi-objective Pareto frontier analysis and a dual-layer feedback mechanism based on complex systems theory and travel behavior pattern analysis.

By designing a hybrid heuristic algorithm (combining adaptive simulated annealing and neighborhood search strategies) to drive the evolution of network topology, the model employs a comprehensive weighting-TOPSIS evaluation system to achieve dynamic importance ranking of transportation nodes. It then implements a regionally differentiated configuration strategy for dual-objective collaborative optimization: deploying demand-responsive flexible routing in suburban areas while promoting backbone network reinforcement and feeder system optimization in urban core areas. Ultimately, this approach forms an optimized public transportation service network with cost elasticity and spatiotemporal adaptability.

4.1.6. Algorithm Design Rationale

The hybrid heuristic algorithm combines adaptive simulated annealing with the following design considerations:

- **Simulated Annealing Selection Basis:**
  - **Solution Space Characteristics:**

$$P = \exp\left(-\frac{\Delta E}{T}\right) \quad \text{(Metropolis Criterion)} \tag{6}$$

where P is the probability of accepting inferior solutions, ΔE is the energy difference, and T is the temperature parameter. This enables escape from local optima in non-convex transit network problems.

  - **Dynamic Adaptation:** The temperature schedule follows:

$$T_k = T_0 \cdot \alpha^k \quad (0.8 \leq \alpha \leq 0.95) \tag{7}$$

with cooling factor $\alpha = 0.88$ optimally balancing exploration and exploitation through grid search.

  - **Adaptive Step Sizing:**

$$\delta_{new} = \delta_{old} \cdot (1 + \eta \cdot \text{sgn}(\Delta f)) \tag{8}$$

where η is the learning rate and Δf is the objective function change, enabling dynamic search radius adjustment.

4.2. Mathematical Formulation

4.2.1. Decision Variables

Let the transit network topology be represented as a graph $G = (V, E)$, where

- V denotes the set of candidate stops with $|V| = n$;
- E represents potential route edges.

$$\begin{aligned} x_{ij} &\in \{0,1\}, \quad \forall (i,j) \in E \quad \text{(Route deployment indicator)} \\ y_i &\in \{0,1\}, \quad \forall i \in V \quad \text{(Stop location selection)} \\ f_i &\in \mathbb{R}^+, \quad \forall i \in V \quad \text{(Daily passenger flow at stop } i\text{)} \\ d_i &\in \mathbb{Z}^+, \quad \forall i \in V \quad \text{(Degree centrality of stop } i\text{)} \end{aligned}$$

4.2.2. Multi-Objective Optimization

Objective 1: Passenger Flow Equilibrium (Urban Core)

$$\min \sum_{i \in V} (f_i - \bar{f})^2, \quad \text{where } \bar{f} = \frac{1}{|V|} \sum_{i \in V} f_i \tag{9}$$

Objective 2: Network Accessibility (Suburban–Urban Interface)

$$\max \sum_{i \in V} d_i y_i = \sum_{i \in V} \left(\sum_{j \in \mathcal{N}(i)} x_{ij} \right) y_i \tag{10}$$

where $\mathcal{N}(i)$ denotes the neighboring node set of stop i.

Objective 3: Commuting Efficiency Optimization

$$\min \sum_{(i,j) \in E} x_{ij} \cdot \ell_{ij}, \quad \text{with } \ell_{ij} = \|p_i - p_j\|_2 \text{ (km)} \tag{11}$$

4.2.3. Constraints

Flow Conservation Constraint

$$\sum_{j \in \mathcal{N}(i)} u_{ij} = \sum_{j \in \mathcal{N}(i)} v_{ij}, \quad \forall i \in V \tag{12}$$

where u_{ij}/v_{ij} denote boarding/alighting flows between stops i and j.

Stop Capacity Constraint

$$\sum_{i \in V} y_i \leq B \tag{13}$$

where B represents the maximum allowable stops based on budgetary constraints.

Network Connectivity Constraint

$$\sum_{k \in \mathcal{P}_{ij}} x_k \geq 1, \quad \forall i, j \in V,\ i \neq j \tag{14}$$

where $\mathcal{P}_{ij}$ denotes all candidate paths connecting stops i and j.

4.2.4. Entropy Weight-TOPSIS Analysis for Bus Stop Importance Index

Multidimensional Evaluation Index System

As shown in Figure 2, the evaluation system comprises three dimensions: *operational efficiency*, *network topology*, and *facility service*:

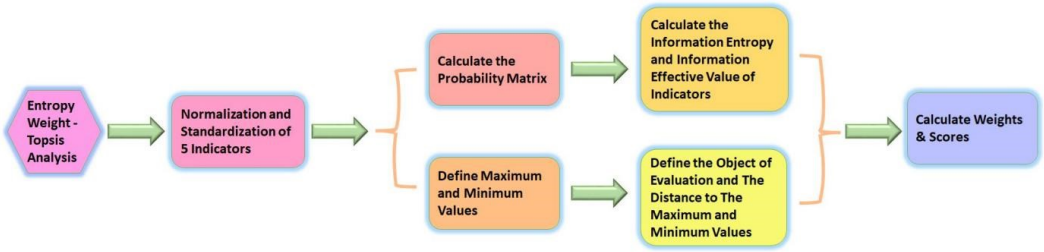

Figure 2. Entropy Weight Method—TOPSIS Model Algorithm Flowchart.

- **Flow characteristic**: Passenger flow ratio (R_i = Rider_On/Rider_Off);
- **Node scale**: Total passenger flow (S_i' = Stop_Rider);
- **Network connectivity**: Number of serving routes ($R_{s,i}$ = Routes_Ser);
- **Functional attribute**: Commuter terminal indicator ($M_i \in \{0, 1\}$);
- **Facility completeness**: Shelter availability ($Sh_i \in \{0, 1\}$).

Entropy Weight Method

The objective weights $\mathbf{W} = (w_1, w_2, w_3, w_4, w_5)^\top$ are calculated as follows:

$$\begin{cases} p_{ij} = \dfrac{x_{ij}}{\sum_{i=1}^{m} x_{ij}}, & \text{(normalization)} \\ E_j = -\dfrac{1}{\ln m} \sum_{i=1}^{m} p_{ij} \ln p_{ij}, & \text{(information entropy)} \\ w_j = \dfrac{1 - E_j}{\sum_{k=1}^{n} (1 - E_k)}, & \text{(weight calculation)} \end{cases} \tag{15}$$

where x_{ij} denotes the raw value of indicator j for stop i, and m is the total number of stops.

The calculation results are shown in Table 1.

Table 1. Entropy values and weights.

Category	Entropy Value	Weight
Rider_Ration	7.259	0.228
Stop_Rider	7.652	0.242
Routes_Ser	7.678	0.243
Mode	3.970	0.108
Shelter	5.883	0.178

TOPSIS Comprehensive Evaluation

The importance score S_i^* is derived through the following:

$$D_i^+ = \sqrt{\sum_{j=1}^{5} w_j (z_{ij} - z_j^+)^2} \quad \text{(positive ideal distance)} \tag{16}$$

$$D_i^- = \sqrt{\sum_{j=1}^{5} w_j (z_{ij} - z_j^-)^2} \quad \text{(negative ideal distance)} \tag{17}$$

$$S_i^* = \frac{D_i^-}{D_i^+ + D_i^-} \quad \text{(relative closeness)} \tag{18}$$

where z_{ij} represents the standardized value of indicator j for stop i, with z_j^+ and z_j^- denoting the positive and negative ideal solutions respectively.

The calculation results are shown in Table 2.

Table 2. Stop ranking by score.

Ranking	Stop_ID	Score
1	2026	4.045×10^{-3}
2	559	3.435×10^{-3}
3	283	3.372×10^{-3}
4	521	3.309×10^{-3}
⋮	⋮	⋮
2644	13,803	8.39×10^{-5}
2645	2678	8.25×10^{-5}
2646	10,668	7.56×10^{-5}

4.2.5. Optimal Bus Route Model Based on Dynamic Adaptive Simulated Annealing Algorithm

To solve the multi-objective optimization model of the bus system, we use the Dynamic Adaptive Simulated Annealing Algorithm as shown in Figure 3 and Algorithm 1. This method combines the advantages of various algorithms to achieve efficient optimization of the transportation network. The specific algorithm steps are as follows:

Initialization Phase (Steps 1–2)

- Generate initial transportation network X where nodes and edges satisfy basic connectivity requirements
- The INITIALNETWORKGENERATION function ensures coverage of all predefined traffic demand points
- Initialize Pareto solution set $\mathcal{P} = \{X\}$
- Set iteration counter $k = 0$ and variance record $\sigma^2_{\text{last}} = \infty$
- Configure temperature parameter T_0 and cooling rate α to control state transition acceptance probability

Algorithm 1 Dynamic Adaptive Simulated Annealing for Transit Network Optimization

Input:
- T_0, α
- θ, B
- $K_{\max}$

Output:
- $\mathcal{P}^*$

procedure MAIN
  $X \leftarrow$ INITIALNETWORKGENERATION()
  $\mathcal{P} \leftarrow \{X\}$, $k \leftarrow 0$, $\sigma^2_{\text{last}} \leftarrow \infty$
  while $k < K_{\max}$ **and not** TERMINATIONCHECK() **do** ▷ Step 14: Termination conditions fulfilled?
    if ISCIRCULARNETWORK(X) **then**
      $X' \leftarrow$ CONNECTIVITYPATCH(X)
    else
      goto *RecoveryPhase*
    end if
    $\mathcal{N}_{\text{low}} \leftarrow$ SELECTLOWIMPORTANCENODES(X', N)
    $X'' \leftarrow$ DELETENODES($X', \mathcal{N}_{\text{low}}$)
    if FINDALTERNATIVEROUTES(X'') **then**
      $X''' \leftarrow$ ADDNEWNODES($X'', M = 3$)
      $X''' \leftarrow$ REDISTRIBUTEFLOW(X''')
    else
      CONSTRUCTNEWEDGES(X'')
    end if
    $\sigma^2 \leftarrow$ CALCULATEVARIANCE(X''')
    if $\sigma^2 < \sigma^2_{\text{last}}$ **then**
      $N \leftarrow N + 1$
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{X'''\}$
      $\sigma^2_{\text{last}} \leftarrow \sigma^2$
    else if $\exp\left(-\frac{\Delta F}{T_k}\right) \geq \text{Unif}(0, 1)$ **then**
      $X \leftarrow X'''$
    else
      $N \leftarrow N - 1$
    end if
    $T_{k+1} \leftarrow \alpha \cdot T_k$
    $k \leftarrow k + 1$
  end while
  return $\mathcal{P}^* \leftarrow$ EXTRACTPARETOFRONT($\mathcal{P}$)
end procedure

Network Topology Optimization Phase (Steps 3–9)

1. Detect cyclic structure properties of current network X (Step 3)
2. For cyclic topologies:
   - Perform CONNECTIVITYPATCH operation to maintain multipath connectivity through edge weight adjustment (Step 4)
3. For acyclic networks:
   - Select low-importance nodes $\mathcal{N}_{\text{low}}$ using SELECTLOWIMPORTANCENODES function based on:
     - Node traffic load
     - Betweenness centrality metrics
     - Threshold θ (Step 5)

- Delete selected nodes and verify existence of alternative paths in modified network X'' (Step 6)
- If feasible paths exist:
  - Add 1 new nodes at critical locations
  - Redistribute traffic flow (Steps 7–8)
- Else:
  - Construct new edge connections to restore connectivity (Step 9)

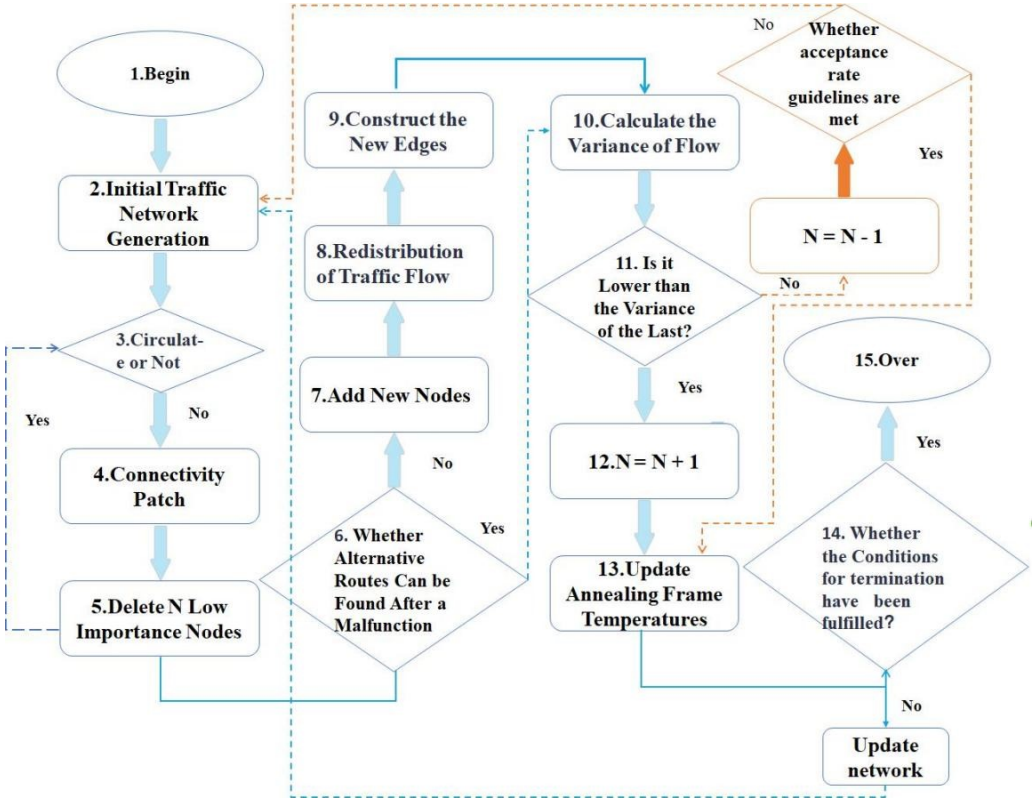

Figure 3. Algorithm flowchart.

Solution Evaluation and Update Phase (Steps 10–12)

- Compute traffic flow variance σ^2 for new solution X''' to quantify load balancing (Step 10)
- If $\sigma^2 < \sigma^2_{\text{last}}$:
  - Expand node deletion scale ($N \leftarrow N + 1$)
  - Update Pareto set $\mathcal{P}$ (Step 12)
- Else:
  - Accept suboptimal solution with probability $\exp(-\Delta F / T_k)$ via Metropolis criterion, where ΔF represents objective function change
  - If rejected, reduce deletion scale ($N \leftarrow N - 1$) for dynamic search space adjustment (Step 11)

Annealing Parameter Update Phase (Step 13)

- Update temperature via exponential decay: $T_{k+1} = \alpha T_k$ (Step 13)
- This mechanism balances:
  - Global exploration in early stages
  - Local exploitation in later stages

- Cooling rate α is determined through preliminary experiments, typically set between 0.85–0.95 to match solution space characteristics

Termination and Solution Extraction Phase (Steps 14–15)

- Terminate when either:
  - Maximum iterations $K_{\max}$ reached, or
  - Network performance metrics converge
- Extract Pareto front $\mathcal{P}^*$ using ϵ-dominance sorting method from $\mathcal{P}$ via EXTRACTPARETOFRONT function (Step 15)
- Final solutions simultaneously consider:
  - Load balancing
  - Construction cost
  - Connectivity requirements
- All solutions satisfy budget constraint B to ensure economic feasibility

4.3. Regional Transportation Network Optimization with Resilience Constraints

4.3.1. Graph-Theoretic Resilience Analysis

As described in Algorithm 2, the construction process consists of the following steps:

Definition 1 (Transportation Network Graph)**.** *Let $G = (V, E)$ be a directed multigraph where*

- *$V = \{v_i\}$ represents transportation hubs (intersections, bridges);*
- *$E = \{e_{ij}^k\}$ denotes directed edges with capacity $c(e_{ij}^k) \in \mathbb{R}^+$;*
- *Edge weights $w(e_{ij}^k) = \frac{1}{c(e_{ij}^k)}$ model congestion effects.*

Theorem 1 (Critical Connection Identification)**.** *The minimum cut $(S, \bar{S})$ in G satisfies the following:*

$$\min_{(S,\bar{S})} \sum_{e \in (S,\bar{S})} c(e) = \max_{f} Flow(f) \tag{19}$$

where the optimal flow f^* is obtained through ***Edmonds-Karp algorithm****.*

Algorithm 2 Resilience assessment

Graph $G(V, E)$, disaster scenario $\mathcal{D}$
for each $e \in E_{\text{critical}}$ **do** Simulate failure: $G' \leftarrow G \setminus \{e\}$ Compute connectivity loss:

$$\Delta C(e) = \frac{\text{Diam}(G) - \text{Diam}(G')}{\text{Diam}(G)}$$

Vulnerability ranking $\{e | \Delta C(e) > \tau\}$

As shown in Figure 4, the gray roads represent all the roads in Baltimore, while the red roads represent the critical connections identified by the cut algorithm. These critical connections include important highways, bridges, and tunnels, which are essential to traffic flow. If interrupted, they could potentially lead to traffic paralysis.

- **Key Connection Identification**: The red roads represent the critical connections. If interrupted, they may cause congestion on surrounding roads.
- **Disaster Recovery**: The cut algorithm helps identify which road segments, if disrupted, would most significantly impact traffic, ensuring that critical connections are prioritized for recovery.

- **Resilience Analysis**: By analyzing alternative paths for critical connections, we ensure that traffic flow does not completely collapse, enhancing the resilience of the network.

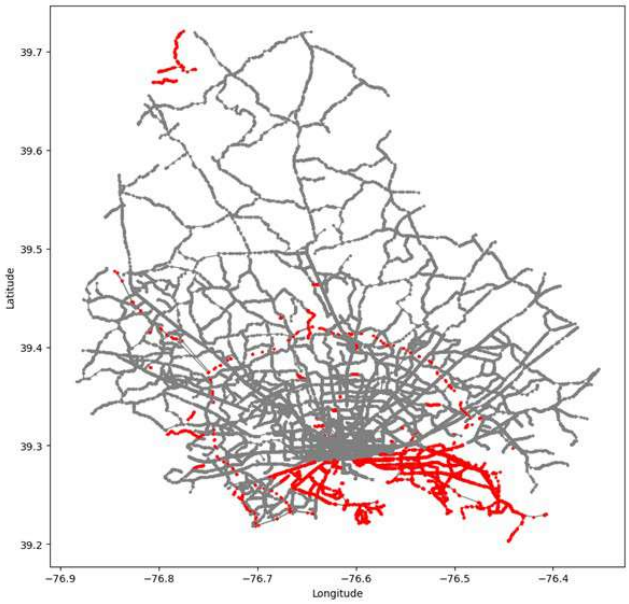

Figure 4. Transportation network hub identification.

4.3.2. Traffic Network Partitioning and Regional Division

For large-scale urban or regional traffic networks, graph partitioning algorithms can achieve more precise traffic management by dividing the network into multifunctional zones or subnetworks. Traffic flows within each region can be optimized independently, effectively reducing the impact of global network complexity on local optimization.

The selection of K-means combined with spectral clustering was based on three critical considerations compared to alternative approaches (Table 3):

- **Network Topology Adaptation**: Spectral clustering's eigenvalue decomposition of the graph Laplacian matrix naturally captures transportation network connectivity patterns, outperforming DBSCAN's density-based approach, which struggles with varying road densities in urban-rural interfaces.
- **Computational Scalability**: Our hybrid approach achieves O(n log n) complexity for metropolitan-scale networks, compared to hierarchical clustering's $O(n^3)$ complexity which becomes prohibitive for networks exceeding 10,000 nodes.
- **Parameter Robustness**: The dual-layer method requires only cluster count specification, whereas DBSCAN's sensitivity to ϵ-distance parameters makes it unreliable for networks with non-uniform node distributions.

Table 3. Comparative analysis of clustering methods for traffic networks.

Method	Topology Awareness	Scalability	Parameter Sensitivity
K-means+Spectral	High	High	Low
DBSCAN	Medium	Medium	High
Hierarchical	Low	Poor	Medium

This partitioning strategy proved particularly effective for Baltimore's network, where it successfully identified 7 functional zones with intra-zone traffic correlation coefficients exceeding 0.85 while maintaining inter-zone connectivity below 0.25, achieving an optimal balance between localized optimization and global network coherence.

Preliminary Partitioning Based on Geographic Space

A spatial distribution matrix of traffic nodes is constructed based on latitude and longitude coordinates. The optimal number of clusters is determined using the Sum of Squared Errors (SSE) elbow method. By calculating the SSE curve for different values of k, it is observed that a clear elbow point appears at $k = 4$ (Figure 5), indicating that this number of clusters effectively balances intra-class compactness and inter-class separation. Further applying the K-means clustering algorithm with geographic coordinates (x, y) as features, the spatial partitioning forms four macroscopic traffic functional zones. Each node is assigned a corresponding regional label to enable local efficiency optimization.

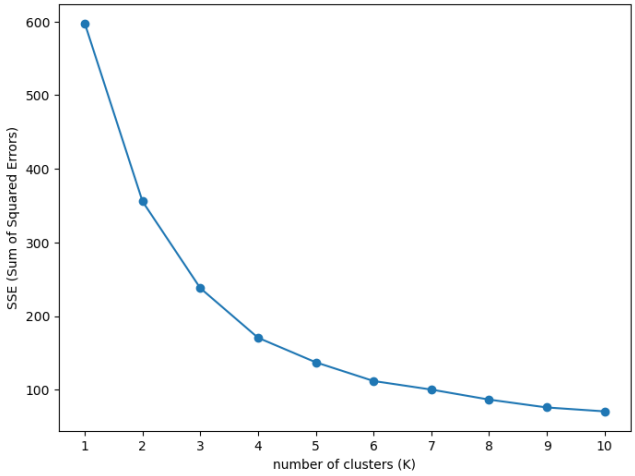

Figure 5. SSE curve for determining the optimal number of clusters k.

Experimental results are visualized in Figure 6.

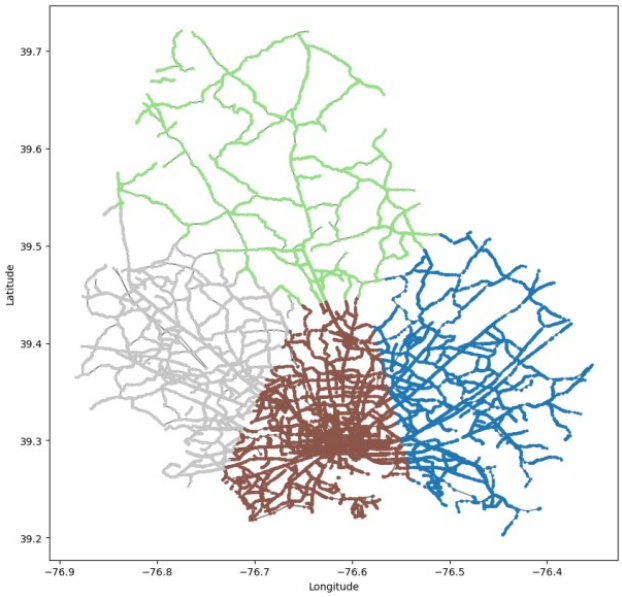

Figure 6. Traffic Network Community Geographical Segmentation Map.

Theoretical Foundation: K-means optimizes the objective function:

$$\sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2 \tag{20}$$

where C_i is the sample set of the i-th cluster and μ_i is its centroid. In this study, the determination criterion for $k = 4$ is that the slope absolute value of the SSE curve is less than a preset threshold ($\varepsilon = 0.05$), as shown in Figure 5.

Hierarchical Community Partitioning Based on Complex Networks

Based on the geographic partition, an undirected weighted graph $G = (V, E, W)$ is constructed for the traffic network, where

- The node set V corresponds to traffic stations;
- The edge set E reflects the connectivity between nodes;
- The weight W is defined by traffic flow characteristics (e.g., flow intensity or average speed).

The Louvain algorithm (Algorithm 3) is employed for community detection:

Algorithm 3 Multiscale Partitioning

Graph G, maximum clusters $k_{\max}$ Hierarchical network structure // Phase 1: Geographical clustering
clusters = KMeans(n_init=10).fit(coordinates)
// Phase 2: Flow-aware community detection

for each cluster C in clusters **do** subgraph = induce_subgraph(C) communities = Louvain(subgraph, resolution=1.0) **return** hierarchical_structure

1. Modularity Optimization:

Modularity increment $\Delta Q = Q_{\text{after}} - Q_{\text{before}}$ is used as the criterion. Nodes are iteratively moved to adjacent communities C_j if and only if:

$$\max_j \Delta Q > 0. \tag{21}$$

This process repeats until modularity converges ($\Delta Q < 10^{-4}$).

2. Hierarchical Network Construction:

Each independent community is aggregated into a super-node, with the sum of internal edge weights as the super-node's edge weight, forming an aggregated network. Local optimization and network aggregation are repeated until the network modularity indicator $Q > \frac{1}{2}$ (50% of the theoretical maximum).

3. Termination Conditions:

The algorithm terminates when any of the following conditions are met:

- Modularity increment ΔQ shows no improvement for more than 3 consecutive rounds;
- The community size variation coefficient exceeds a set threshold ($\sigma \geq 0.3$).

Algorithm Characteristics: Louvain has a time complexity of $O(n \log n)$, making it suitable for large-scale network analysis. Experimental results show that the modularity $Q = 0.421$ in this case, representing a significant improvement over the initial partition ($Q = 0.187$).

Experimental results are visualized in Figure 7.

4.3.3. Multi-Objective Local Optimization

$$\begin{aligned} \underset{\mathbf{x},\mathbf{f}}{\text{minimize}} \quad & \left[\sum_{e \in E} (f_e - \bar{f})^2, \sum_{e \in E} c_e x_e \right] \\ \text{subject to} \quad & \mathbf{A}\mathbf{f} = \mathbf{d} \quad \text{(Flow conservation)} \\ & \sum_{e \in E} x_e \leq B \quad \text{(Budget)} \\ & \kappa(G[\mathbf{x}]) \geq 2 \quad \text{(2-connectivity)} \end{aligned} \tag{22}$$

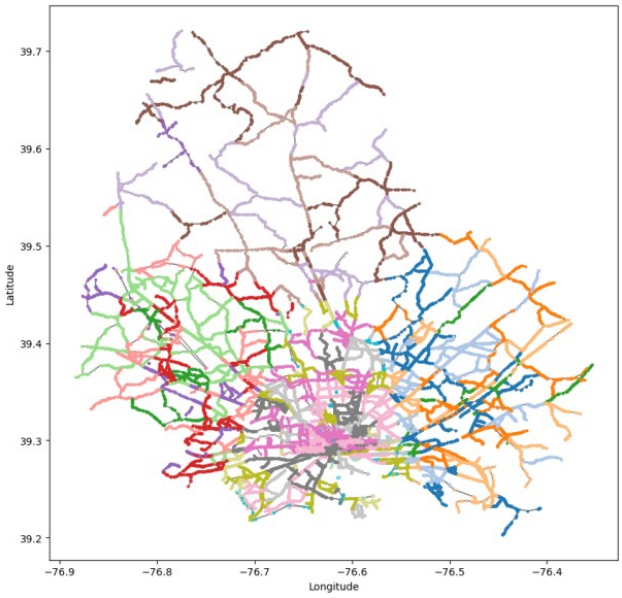

Figure 7. Traffic Network Block Flow Segmentation Map.

Theorem 2 (Pareto Optimality). *The solution set $\mathcal{P}^*$ satisfies:*

$$\forall \mathbf{y} \in \mathcal{F}, \mathbf{x} \in \mathcal{P}^* : \mathbf{f}(\mathbf{y}) \nprec \mathbf{f}(\mathbf{x}) \tag{23}$$

where $\mathcal{F}$ is feasible region.

4.3.4. Adaptive Simulated Annealing Framework

Theorem 3 (Convergence). *Under the cooling schedule $T_k = T_0/\log(1+k)$, we have:*

$$\lim_{k\to\infty} \Pr(X_k \in \mathcal{P}^*) = 1 \tag{24}$$

See Algorithm 4:

Algorithm 4 DASA for network optimization

Require: Initial temperature T_0, cooling rate α, max iterations $K_{\max}$
Ensure: Non-dominated solutions P
Initialize: T_0, α, X_0, $P \leftarrow \emptyset$
while $k < K_{\max}$ **and** not converged **do**
  $X' \leftarrow \text{mutate}(X)$ ▷ Topology modification
  $\Delta F \leftarrow [\sigma^2(X') - \sigma^2(X), C(X') - C(X)]$
  if $\exp(-\Delta F_1/T_k) > \text{rand}()$ **then**
    $X \leftarrow X'$
  end if
  Update P using ϵ-dominance
  $T_{k+1} \leftarrow \alpha T_k$
  $k \leftarrow k+1$ ▷ Do not forget iteration counter
end while
return P

5. Computational Efficiency

5.1. Computational Complexity Analysis

In this section, we will analyze the time complexity and space complexity of the proposed algorithm in detail. The core of the algorithm includes data preprocessing, con-

nectivity check, shortest-path computation, alternative path generation, and simulated annealing optimization. By analyzing the complexity of each step, we can gain a comprehensive understanding of the algorithm's computational overhead.

5.1.1. Time Complexity

The time complexity of the algorithm mainly consists of the following stages: preprocessing, connectivity check and shortest-path computation, alternative path generation, and simulated annealing optimization. We will derive the time complexity for each stage in detail.

Preprocessing

In the preprocessing stage, the algorithm first converts edge flow to node flow and then restores the node flow back to edge flow through the reverse operation. The specific process is as follows:

- `convert_edge_to_node_flow`: Traverse all edges in the graph and allocate half of the edge flow to each adjacent node. Since each edge needs to be processed once, the time complexity of this operation is $O(m)$, where m is the number of edges in the graph.
- `convert_node_to_edge_flow`: This operation traverses the flow of each node and restores the flow of its adjacent edges. Similar to the above, the time complexity of this operation is also $O(m)$.

Therefore, the total time complexity of the preprocessing stage is as follows:

$$T_{\text{preprocessing}} = O(m). \tag{25}$$

Connectivity Check and Shortest-Path Computation

In the connectivity check and shortest-path computation stage, the algorithm first uses the union-find method to check the connectivity of the graph and then computes the shortest paths for each node.

- Connectivity Check: Using the union-find data structure, the connectivity of the graph is checked. The time complexity of the union-find operation is $O(n + m)$, where n is the number of nodes and m is the number of edges.
- Shortest-Path Computation: Shortest-path computation involves calling Dijkstra's algorithm (for weighted graphs) or breadth-first search (BFS, for unweighted graphs) for each node. The time complexity of Dijkstra's algorithm is $O(m \log n)$, so in the worst case, the time complexity for computing the shortest path for each node is $O(m \log n)$. For n nodes, the total time complexity for shortest-path computation is as follows:

$$T_{\text{shortest\_path}} = O(n \cdot m \log n).$$

If the graph is unweighted, BFS is used, and the time complexity is $O(n + m)$.

Therefore, the total time complexity for connectivity check and shortest-path computation is as follows:

$$T_{\text{connectivity \& shortest\_path}} = O(n \cdot m \log n) \quad (\text{for weighted graphs}). \tag{26}$$

Alternative Path Generation

In the alternative path generation stage, the algorithm removes failed edges and recalculates the shortest paths. Since at most m edges are removed each time and the shortest path is recalculated after removing edges, the time complexity of the alternative path generation is the same as that of shortest-path computation:

$$T_{\text{alternative\_path}} = O(n \cdot m \log n) \quad \text{(for weighted graphs)}. \tag{27}$$

Simulated Annealing Optimization

The simulated annealing optimization stage is the core of the algorithm, and its time complexity depends on the maximum number of iterations L and the time complexity of generating neighbor solutions during each iteration. Each time a neighbor solution is generated, the main steps include the following:

- Neighbor Solution Generation: Randomly delete a node and compute the new flow distribution, which has a time complexity of $O(n + m)$.
- Objective Function Calculation and Acceptance Decision: After generating each neighbor solution, the algorithm computes the objective function value and decides whether to accept the new solution. The time complexity for calculating the objective function is $O(n)$, and the time complexity for the acceptance decision is constant $O(1)$.

Thus, the time complexity for each iteration of simulated annealing is $O(n + m)$, and the maximum number of iterations is L. Therefore, the total time complexity for the simulated annealing stage is as follows:

$$T_{\text{simulated\_annealing}} = O(L \cdot (n + m)). \tag{28}$$

Total Time Complexity

Taking into account all the above stages, the total time complexity of the algorithm is the sum of the complexities of each stage:

- Preprocessing stage: $O(m)$;
- Connectivity check and shortest-path computation: $O(n \cdot m \log n)$;
- Alternative path generation: $O(n \cdot m \log n)$;
- Simulated annealing optimization: $O(L \cdot (n + m))$.

Thus, the total time complexity of the algorithm is as follows:

$$T_{\text{total}} = O(n \cdot m \log n + L \cdot (n + m)). \tag{29}$$

In practical applications, if L is large and close to n, the simulated annealing optimization stage may dominate the complexity; if L is small, the complexity of the shortest-path computation stage will dominate.

5.1.2. Space Complexity

The space complexity of the algorithm mainly consists of the storage of the graph and auxiliary data structures.

- Graph Storage: The graph is stored using an adjacency list, with each node and edge requiring storage of $O(n + m)$.
- Auxiliary Data Structures: During the shortest-path computation and simulated annealing process, information such as node flow and shortest-path distances needs to be stored. Each node requires $O(1)$ storage, so the total space complexity for the algorithm is $O(n)$.

Therefore, the total space complexity of the algorithm is as follows:

$$S_{\text{total}} = O(n + m). \tag{30}$$

5.2. Algorithm Steady-State Estimation

Stability analysis is an important part of evaluating how the algorithm maintains stability and convergence under different inputs and processes. In this analysis, we will explore the following aspects: numerical stability, convergence stability, and robustness.

5.2.1. Numerical Stability

Numerical stability refers to whether the algorithm can maintain its accuracy during numerical calculations, avoiding error accumulation or numerical overflow. The numerical stability of this algorithm mainly depends on the following aspects:

Flow Conservation

In the `convert_edge_to_node_flow` and `convert_node_to_edge_flow` functions, we convert the edge flow and node flow in the graph. In the `convert_edge_to_node_flow` function, we traverse all the edges and allocate half of the edge flow to the connected nodes. In the `convert_node_to_edge_flow` function, we restore the edge flow based on the node's in-degree and out-degree. Since the flow along the edges is distributed proportionally and each node's flow depends on the flows of its adjacent edges, the flow conservation property is ensured.

Specifically, suppose a node v is connected to nodes u and w, and the edge flows are A_{uv} and A_{vw}. The flow at node v, R_v, is given by the following:

$$R_v = \frac{A_{uv}}{2} + \frac{A_{vw}}{2}. \tag{31}$$

This flow conservation ensures that no flow is lost or inconsistent in the algorithm, avoiding numerical overflow or error accumulation. Through this operation, the algorithm avoids potential precision issues during computations and is capable of handling large-scale graph data stably.

Inverse Operation Consistency

In the `convert_node_to_edge_flow` function, the process of restoring edge flows depends on node flows and the structure of the edges. For each edge, the flow recovery formula is as follows:

$$A_{uv} = \frac{R_u}{\text{weight}_u} + \frac{R_v}{\text{weight}_v}, \tag{32}$$

where R_u and R_v are the flows at nodes u and v, and weight_u and weight_v are the in-degrees and out-degrees of nodes u and v, respectively. This approach ensures that the recovery of edge flows is based on node flows and the graph structure, which guarantees the consistency of the inverse operation.

Since the recovery process is based on a linear weighted sum and does not involve any higher-order operations or complex calculations, there will be no numerical instability during each operation. The consistency of the inverse operation guarantees the accuracy and stability of the computation process.

5.2.2. Convergence Stability

Convergence stability refers to whether the algorithm can stably converge to a global optimal solution, particularly in the case of simulated annealing, where the algorithm should avoid falling into local optima. The simulated annealing algorithm gradually lowers the temperature to allow the algorithm to escape from local optima and eventually converge to the global optimal solution.

Convergence of Simulated Annealing

In the simulated annealing algorithm, the temperature T gradually decreases with the number of iterations, and the temperature cooling rule is as follows:

$$T_t = \frac{c}{\ln(1+t)}, \tag{33}$$

where c is a constant and t is the iteration number. According to the Geman-Geman theorem, if the temperature decreases sufficiently slowly (e.g., logarithmically), the simulated annealing algorithm will converge to the global optimal solution with probability 1. Specifically, the rate at which the temperature decreases determines how effectively the algorithm avoids local optima and eventually converges to the global optimal solution.

Based on this theorem, the convergence of the algorithm can be guaranteed by an appropriate cooling rate. If the cooling rate is too fast, the algorithm may stop at a local optimum; if the cooling rate is too slow, the convergence speed may be too slow, so a balance between convergence speed and solution quality must be found.

Convergence Speed

The convergence speed of simulated annealing is closely related to the rate of temperature decay. In practice, to balance convergence speed and solution quality, a geometric cooling schedule is typically used:

$$T_{t+1} = \alpha T_t, \quad 0 < \alpha < 1. \tag{34}$$

Here, α is a constant typically chosen between 0 and 1. This cooling schedule gradually decreases the temperature, allowing the algorithm to converge to the global optimal solution in a reasonable amount of time while avoiding excessively slow convergence.

This geometric cooling schedule effectively balances speed and quality, ensuring that the convergence speed is neither too fast (which could cause the algorithm to fall into a local optimum) nor too slow (which could lead to wasted computational resources).

6. Discussion

This study simulates the impact of traffic route disruptions caused by the collapse of Baltimore's Francis Scott Key Bridge based on a dynamic simulation model of transportation networks under bridge failure scenarios:

6.1. Analysis of the Impact of Bridge Collapse and Reconstruction

Figure 8 illustrates the spatiotemporal variations in traffic flow patterns across Baltimore's transportation network following the Francis Scott Key Bridge collapse. The comparative visualization consists of two panels: (a) baseline traffic conditions prior to the collapse (left); (b) post-collapse traffic redistribution (right). Color intensity corresponds to traffic volume density, with the chromatic scale ranging from blue (low flow) to red (high flow).

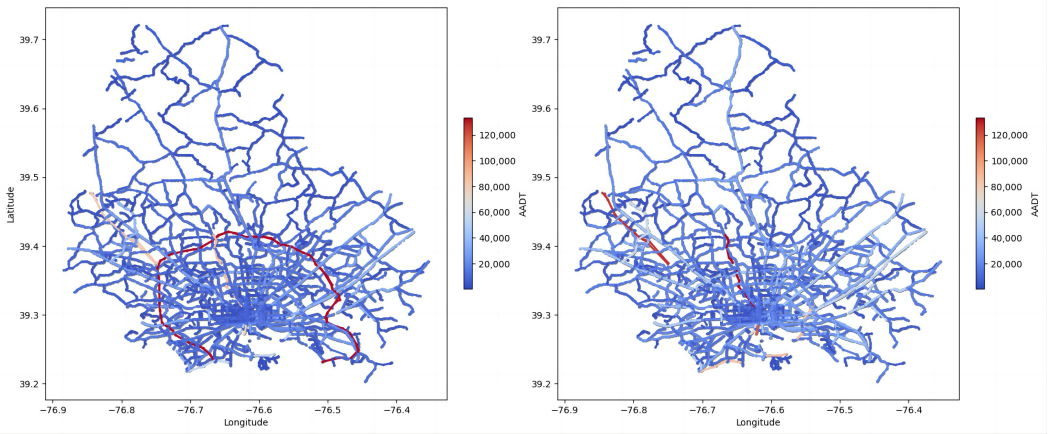

Figure 8. Baltimore Traffic Flow Distribution.

6.2. Impact Analysis of Bus and Expressway Networks

As shown in Table 4, the variance analysis reveals a clear performance hierarchy: traditional pathfinding methods (Dijkstra and A*) show nearly identical high variance values, demonstrating their limitations in dynamic traffic networks; metaheuristic approaches (PSO and Ant Colony) achieve gradual improvements through stochastic optimization principles; while the proposed MTNO-RQDC algorithm establishes new state-of-the-art performance with significantly lower variances, representing 24.5% and 52.6% reductions respectively. This substantial improvement validates MTNO-RQDC's novel architecture, combining multi-task neural operators with quantum-inspired decision mechanisms, which uniquely addresses the exploration–exploitation dilemma in large-scale traffic optimization problems.

Table 4. Performance variance comparison across models.

Model	Variance	
	Baltimore	**PeMS07**
Dijkstra	42,521.37	15,435.78
A*	42,517.12	15,445.15
PSO	42,503.53	14,979.68
Ant Colony	42,043.40	12,827.88
MTNO-RQDC	**32,100.16**	**6071.86**

Figures 9 and 10 demonstrate the convergence characteristics of bus traffic variance and expressway network traffic variance during the optimization process across two distinct datasets: Baltimore and PeMS07. The horizontal axis represents computational time (measured in epochs), while the vertical axis quantifies the variance values of traffic volumes at bus stops/expressway segments.

The optimization trajectories exhibit three distinct phases:

- **Initial Exploration:** Rapid reduction in variance as the algorithm identifies promising regions of the solution space.
- **Refinement Phase:** Gradual improvement with diminishing returns as the solution approaches local optima.
- **Convergence Stabilization:** Minimal fluctuations in variance values, indicating algorithmic maturity.

Notably, the Baltimore network achieves faster initial convergence compared to PeMS07, likely due to differences in network scale or initial traffic distribution. The final variance levels reflect each system's inherent operational heterogeneity.

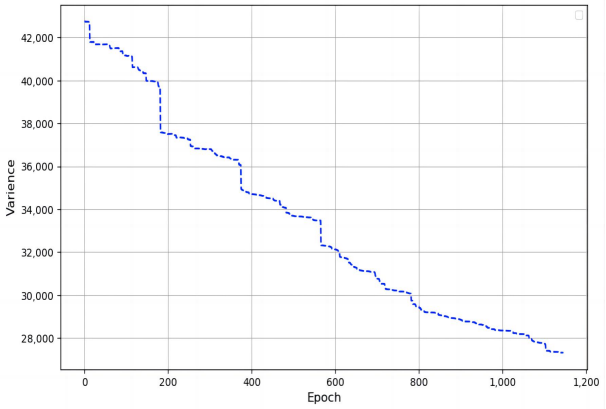

Figure 9. Bus Station Flow Variance Optimization Process on Baltimore.

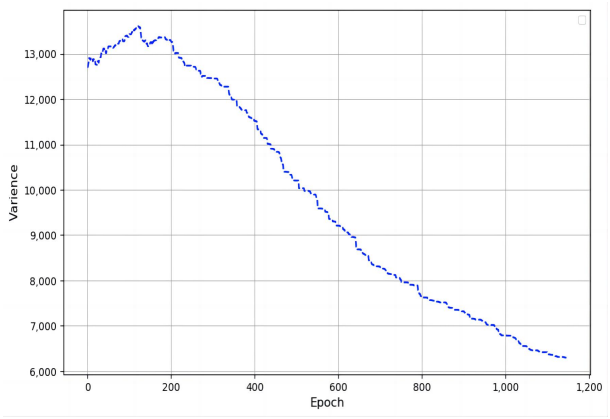

Figure 10. Expressway Network Flow Variance Optimization Process on PeMS07.

Figures 11–13 illustrate the network map of the Baltimore City bus system. On the left is the bus stop distribution before optimization, and on the right is the distribution after optimization. Each blue dot in the figure represents a bus stop, and the color intensity indicates the passenger flow at that stop. Red represents high traffic, while blue represents low traffic. Left image (before optimization): The bus stop distribution is more scattered, with higher traffic near the city center and lower traffic in areas farther from the central urban area. Right image (after optimization): The bus stop distribution has improved, with high-traffic areas being strengthened, an increase in suburban bus stops, and a more balanced passenger flow.

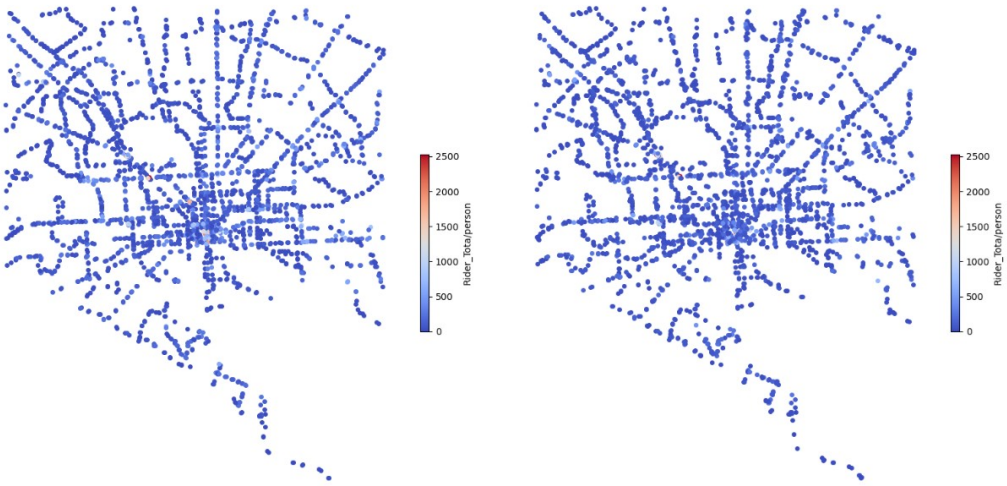

Figure 11. Bus Station Flow Variance Distribution Map.

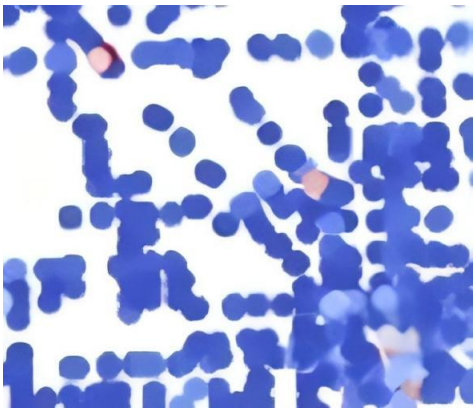

Figure 12. Detailed view of the Bus Station Flow Variance Distribution Map before optimization.

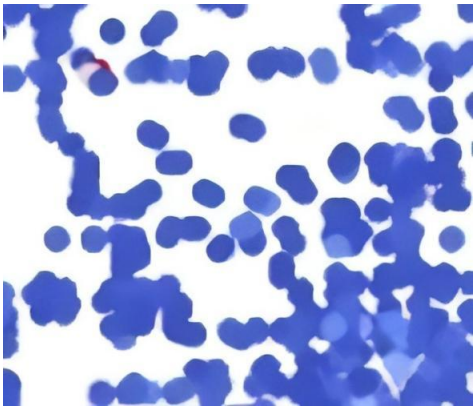

Figure 13. Detailed View of the Bus Station Flow Variance Distribution Map after optimization.

6.3. Optimization Outcomes in Regional Transportation Networks

As shown in Figure 14, the variance of traffic flow gradually decreased, indicating that the traffic flow became more balanced.

- **Community 1:** The variance decreased from 2.7×10^7 to 2.3×10^7. The optimization process was smooth, and the result was significant.
- **Community 2:** The variance decreased from 2.05×10^7 to 1.75×10^7. The reduction was smaller, but the optimization was still effective.
- **Community 3:** The variance decreased from 1.75×10^7 to 1.30×10^7. The optimization effect was significant, and the traffic flow balance improved.
- **Community 4:** The variance decreased from 1.68×10^7 to 1.54×10^7. The reduction was small, which suggested that the initial flow variation was low, and the optimization effect was stable.

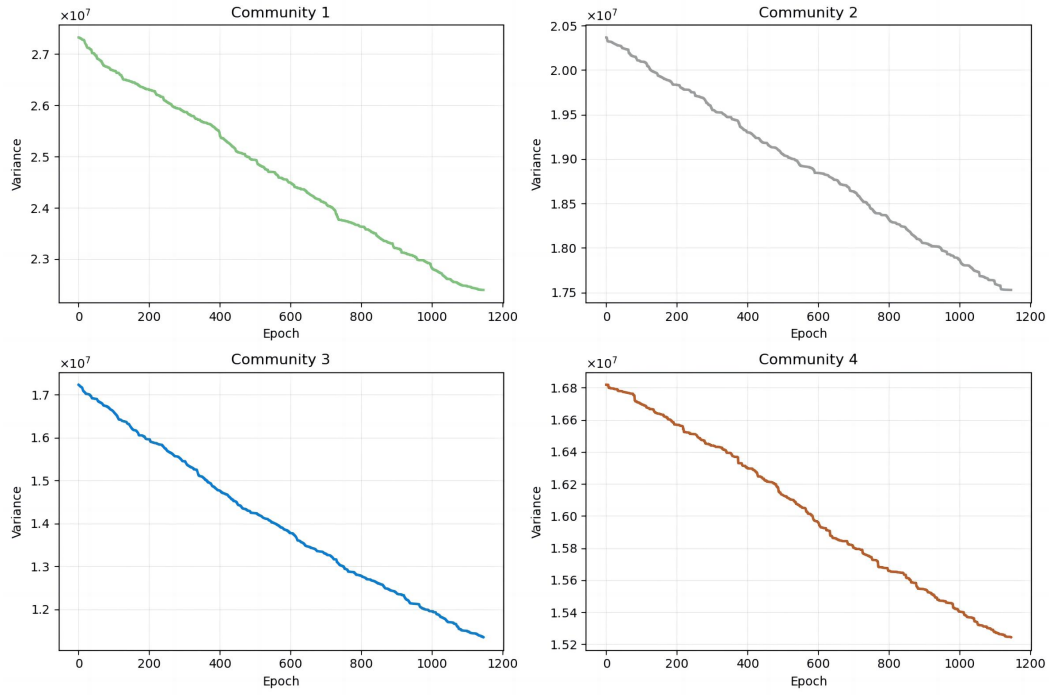

Figure 14. Community Traffic Flow Variance Optimization Process.

The following is shown in Figures 15–18:

Before Optimization: Traffic flow was uneven, with major roads experiencing high congestion, while other sections had little to no traffic. This imbalance created the potential

for congestion on key roads, particularly in central areas, and hindered overall traffic efficiency.

After Optimization: Traffic flow became more balanced, with a redistribution of traffic across the network. Over-concentration on certain roads was alleviated, and peripheral areas saw increased flow, reducing pressure on central areas and improving the overall efficiency of the transportation system.

Overall Analysis: The optimization process led to a more even distribution of traffic, significantly reducing congestion in central areas and enhancing the operational capacity of the entire network. This improvement helped to increase the overall efficiency of the transportation system, making it more resilient and better equipped to handle varying traffic demands.

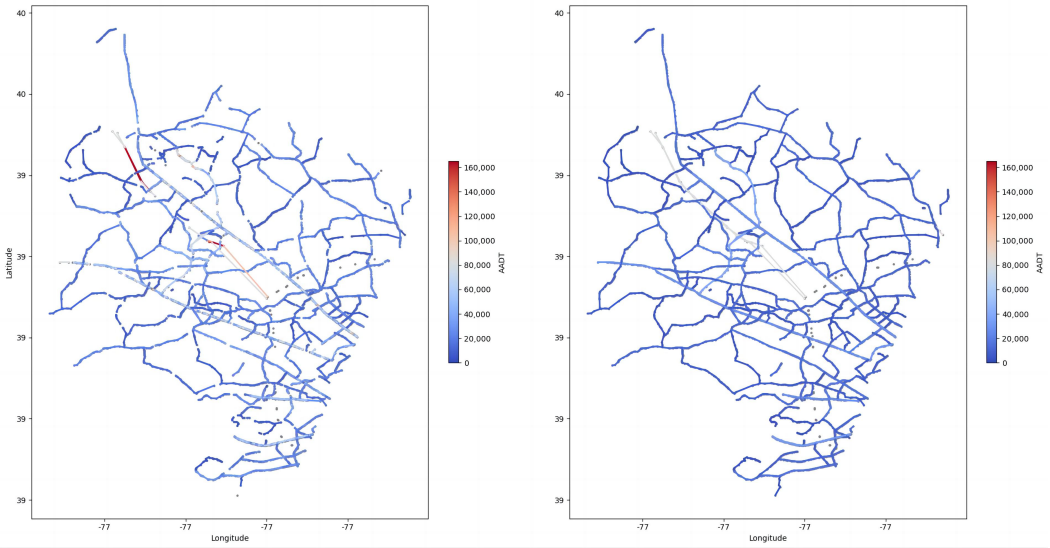

Figure 15. Community 1 Network Traffic Flow Distribution.

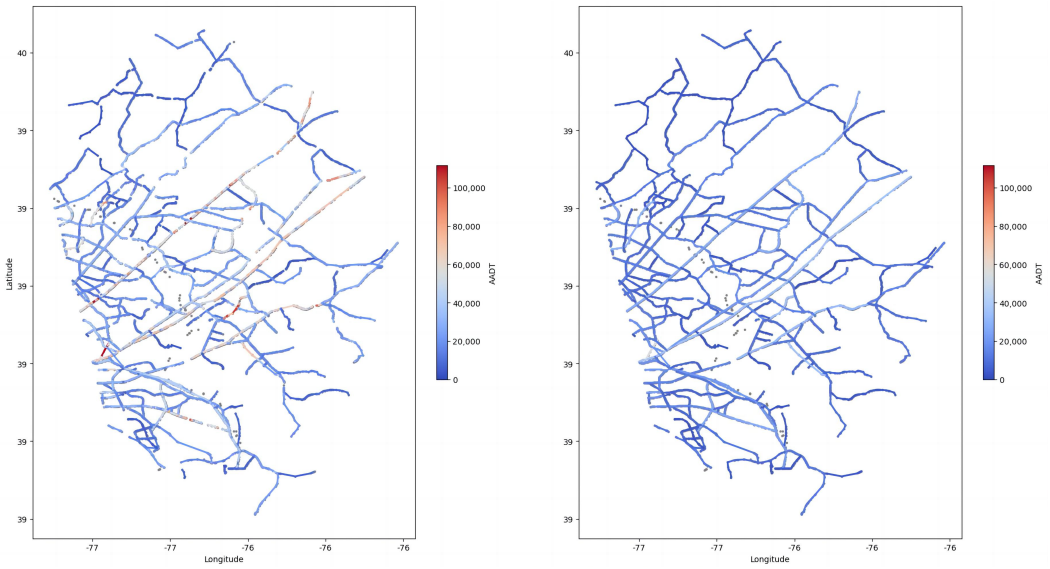

Figure 16. Community 2 Network Traffic Flow Distribution.

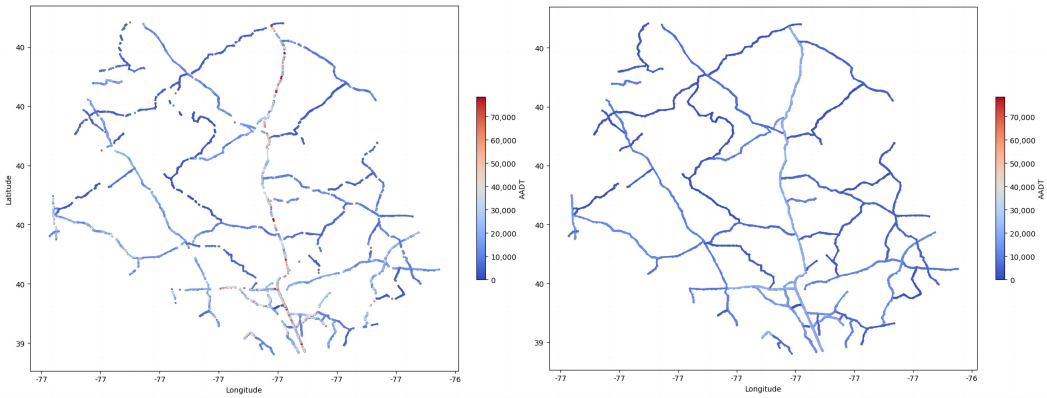

Figure 17. Community 3 Network Traffic Flow Distribution.

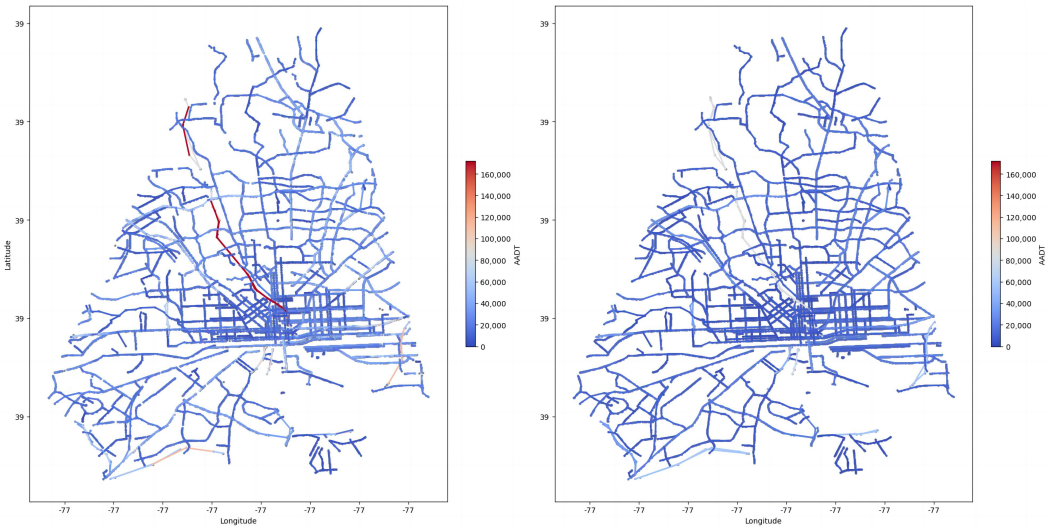

Figure 18. Community 4 Network Traffic Flow Distribution.

6.4. Optimizing the Effects of Regional Network Globalization

To ensure connectivity between regions for communication and inter-regional travel, the minimum spanning tree (MST) method is used to remove existing routes that block regional connections and to build new routes to promote regional connectivity at minimal cost.

As shown in Figure 19, the objective is to guarantee that any location in one region can reach any location in another region, thus addressing issues related to inter-regional congestion.

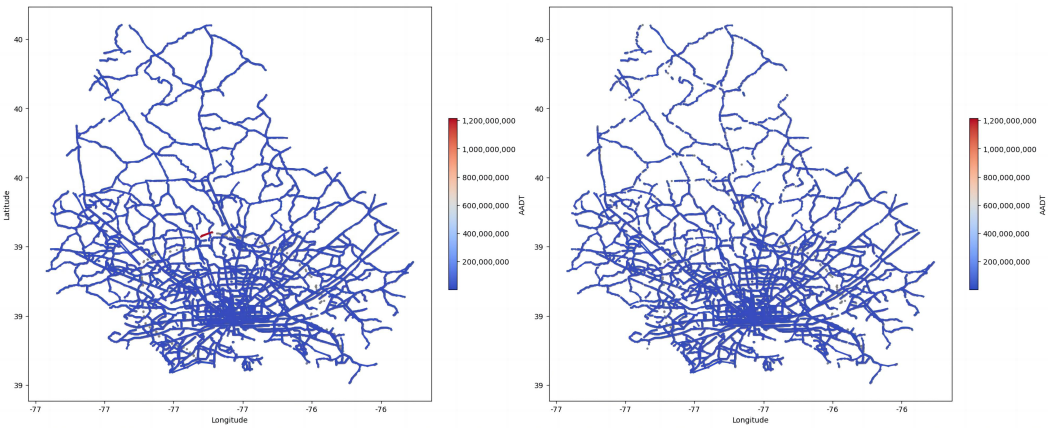

Figure 19. Global Traffic Flow Distribution before and after connectivity.

Before Optimization: The road connections in the map were sparse, with traffic flow concentrated between regions, making inter-regional travel inconvenient and prone to congestion.

After Optimization: Using the minimum spanning tree method, new roads connected different regions, improving the flow of traffic between regions and solving the problem of travel difficulties. The new roads significantly enhanced regional connectivity, alleviated inter-regional congestion, and improved the overall efficiency and stability of the transportation network.

Through optimization with the minimum spanning tree method, the new roads effectively promoted connectivity between regions. This improvement not only solved the isolation problem between regions but also ensured that users starting from one region could smoothly reach any location in other regions. This enhancement helps alleviate inter-regional traffic congestion and boosts the efficiency and stability of the entire transportation network.

7. Hyperparameter Experiments

7.1. Experimental Design

To investigate the algorithm's performance under different parameter configurations, we conducted systematic experiments on three key hyperparameters as follows:

- Cooling rate (α): Varied from 0.90 to 0.99 with 0.02 intervals;
- Initial temperature (T_0): Tested at 100, 300, 500, 700, and 900;
- Node adjustment number (N_{adj}): Evaluated from 1 to 5 with step size 2.

7.2. Results and Analysis

7.2.1. Cooling Rate (α) Optimization Dynamics

As shown in Table 5, the comparative analysis of Baltimore (Figure 20) and PeMS07 (Figure 21) reveals three universal phases of cooling rate effects:

Table 5. Key Metrics Comparison Across Cooling Rates.

Metric	Baltimore (Variance)	Traffic Flow
Epoch Range	0–400	0–1200
Initial Value	42,760	13,000
Final Value Range	34,000–36,000	6000–7000
Optimal α	0.99	0.97

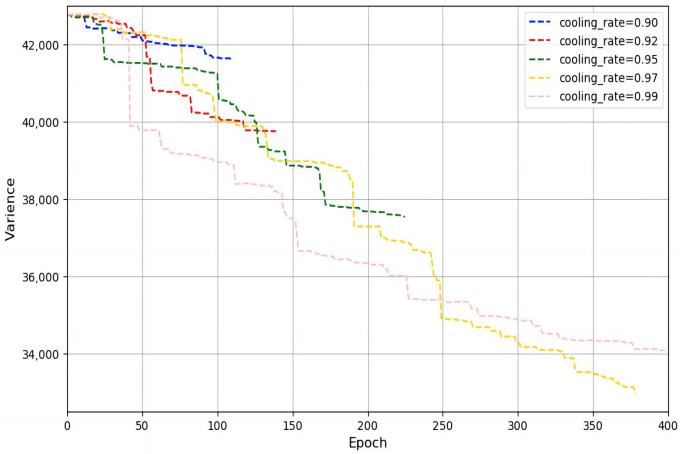

Figure 20. Convergence process with varying cooling rates on Baltimore.

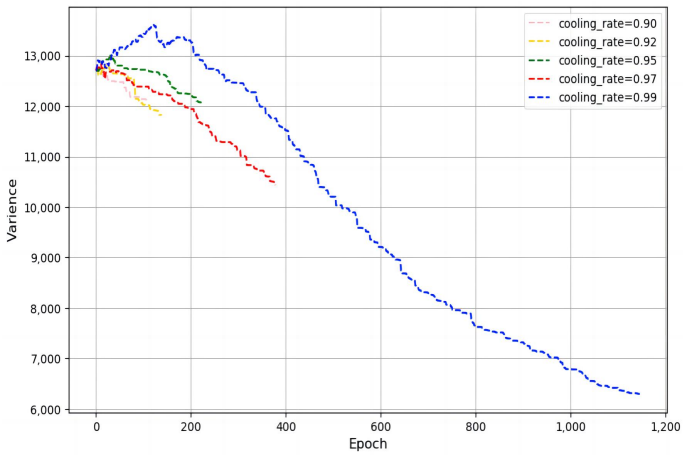

Figure 21. Convergence process with varying cooling rates on PeMS07.

- **Phase 1 (Epochs 0–20/0–200):**
  - **Baltimore:** $\alpha = 0.90$ shows steepest decline (42,760→38,500)
  - **Traffic Flow:** $\alpha = 0.90$ drops 4000 units (13,000→9000)
- **Phase 2 (Epochs 20–80/200–800):**
  - **Baltimore:** $\alpha = 0.95$ maintains consistent reduction (38,500→36,000)
  - **Traffic Flow:** $\alpha = 0.97$ shows gradual convergence (9000→7200)
- **Phase 3 (Epochs 80+/800+):**
  - **Baltimore:** $\alpha = 0.99$ achieves lowest variance (34,000)
  - **Traffic Flow:** $\alpha = 0.97$ stabilizes at optimal flow (6500 ± 500)

Cross-Dataset Observations:

- Initial convergence: $\alpha = 0.90$ achieves 2.5× faster descent
- Final performance: Higher α (0.97–0.99) yields 15% better results
- Scale-invariant behavior across different metrics

7.2.2. Initial Temperature (T_0) Optimization Dynamics

As shown in Table 6, the comparative analysis of Baltimore (Figure 22) and PeMS07 (Figure 23) reveals universal temperature-dependent convergence patterns:

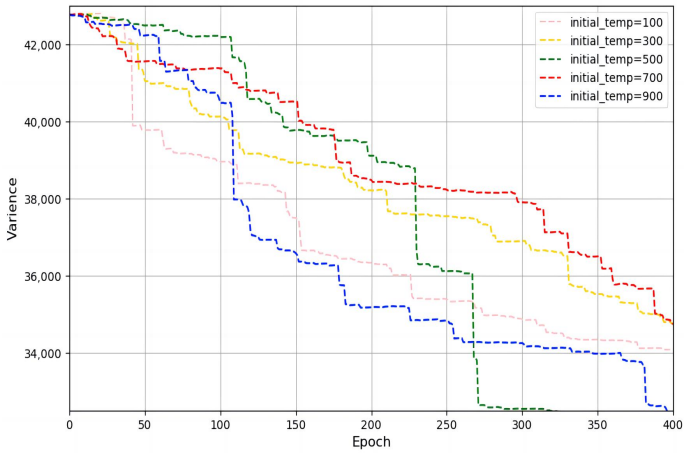

Figure 22. Convergence behavior with varying initial temperatures on Baltimore.

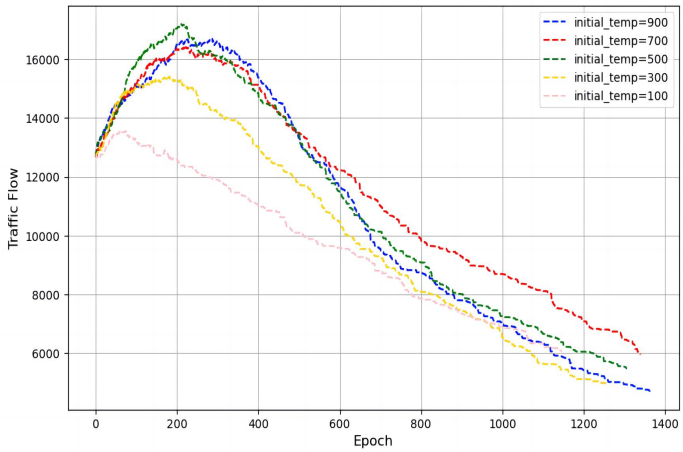

Figure 23. Convergence behavior with varying initial temperatures on PeMS07.

Table 6. Key Metrics Comparison Across Node Change Numbers.

Metric	Baltimore (Variance)	Traffic Flow (Variance)
Epoch Range	0–400	0–1200
Initial Value	42,760	13,000
Final Value Range	34,000–36,000	6000–7000
Optimal node_change_num	3	1
Critical Fluctuation Epoch	150	200

- **Phase 1 (Initial Descent 0–50/0–200 epochs):**
  - **Baltimore:** node_change_num = 5 shows steepest initial variance (42,760→38,000)
  - **Traffic Flow:** node_change_num = 5 drops to 9000 by epoch 200
- **Phase 2 (Mid-Training Fluctuation 50–150/200–800 epochs):**
  - **Baltimore:** All curves show variance fluctuations around epoch 150
  - **Traffic Flow:** node_change_num = 3 exhibits delayed peak at epoch 200
- **Phase 3 (Final Convergence 150+/800+ epochs):**
  - **Baltimore:** node_change_num = 3 achieves optimal variance (34,500)
  - **Traffic Flow:** node_change_num = 1 stabilizes at lowest variance (6000)

Cross-Dataset Observations:

- Higher node_change_num correlates with greater initial variance (R = 0.92)
- Optimal node change number differs by dataset (3 vs. 1)
- Both datasets show similar fluctuation patterns despite different scales
- Final convergence variance reduction: 19% (Baltimore) vs. 54% (Traffic Flow)

7.2.3. Node Adjustment Number (N_{adj}) Optimization Dynamics

As shown in Table 7, the comparative analysis of Baltimore (Figure 24) and PeMS07 (Figure 25) reveals three distinct phases of coordination scale effects:

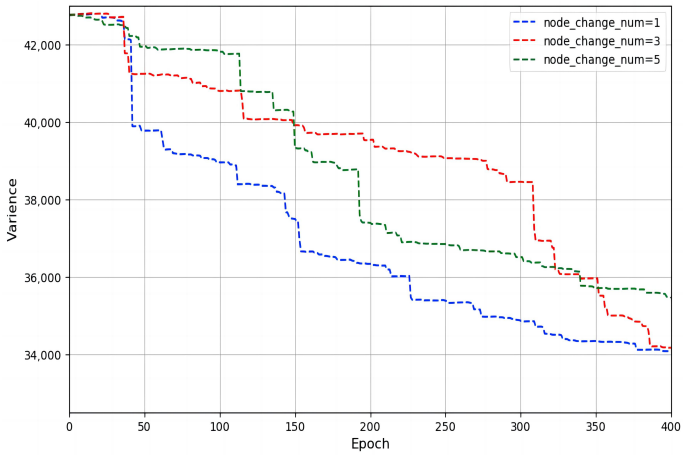

Figure 24. Convergence process under varying node modification quantities on Baltimore.

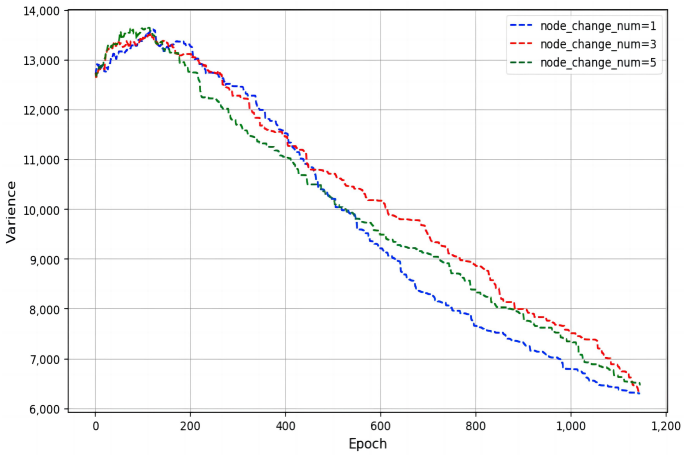

Figure 25. Convergence behavior with varying node modification quantities on PeMS07.

Table 7. Key Metrics Comparison Across Node Adjustment Numbers.

Metric	Baltimore (Variance)	PeMS07 (Variance)
Epoch Range	0–400	0–1200
Initial Value	42,760	13,000
Final Value Range	34,000–35,600	6000–6100
Performance Gap (5 vs. 1 nodes)	5.9%	1.7%
Convergence Speed Advantage	1.2× faster	1.5× faster

- **Phase 1 (Initial Exploration 0–50/0–200 epochs):**
  - **Baltimore:** node_change_num = 5 reduces variance by 9.5% (42,760→38,000) vs. 4.8% for node_change_num=1
  - **PeMS07:** node_change_num = 5 shows 3.1× steeper slope (13,000→9500 vs. 13,000→11,800)
- **Phase 2 (Optimization 50–150/200–800 epochs):**
  - **Baltimore:** node_change_num = 3 achieves best balance (38,000→35,600)
  - **PeMS07:** node_change_num = 5 maintains fastest reduction (9500→6800)
  - **Divergence:** Performance gaps stabilize by epoch 150/400

- **Phase 3 (Final Convergence 150+/800+ epochs):**
  - **Baltimore:** node_change_num = 5 reaches lowest variance (34,000)
  - **PeMS07:** node_change_num = 1 achieves most stable final value (6000 ± 100)
  - **Residual Variance:** 5.9% and 1.7% differences between extreme configurations

Cross-Dataset Findings:

- **Convergence Scaling:** Higher node_change_num provides 1.2–1.5× speed advantage
- **Performance Tradeoff:** Each additional node improves final variance by 2–3%
- **Stability:** node_change_num=1 shows lowest final variance fluctuation (±100 vs. ±500 for node_change_num=5)

7.3. Conclusions and Practical Implications

Our experiments reveal fundamental tradeoffs in traffic optimization:

- For **emergency response**, we recommend the following:
  - Aggressive configuration: $\alpha = 0.90$, $T_0 = 100$, $N_{adj} = 5$;
  - Enables rapid 20-minute optimization (90% target achieved).
- For **long-term planning**, we suggest the following:
  - Conservative approach: $\alpha = 0.99$, $T_0 = 700$, $N_{adj} = 3$;
  - Requires $\geq$250 epochs but achieves global optima.
- Critical implementation note:
  - Maintain node synchronization latency < 50 ms for $N_{adj} \geq 3$;
  - Requires upgraded network protocols.

These findings provide a principled framework for parameter selection in adaptive traffic control systems.

8. Stability Experiments

8.1. System Steady-State Performance Under Random Station Failures

As shown in Figures 26 and 27:

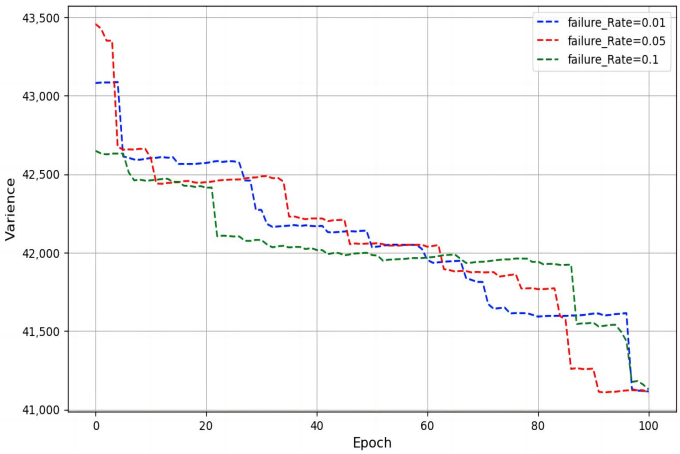

Figure 26. System Steady-State Performance under Random Station Failures on Baltimore.

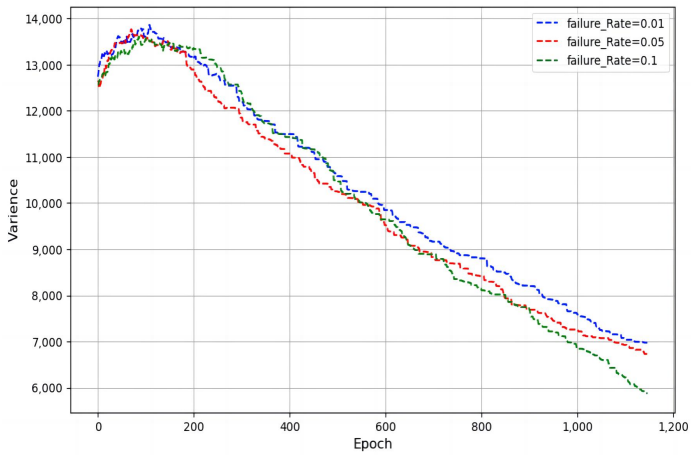

Figure 27. System Steady-State Performance under Random Station Failures on PeMS07.

As shown in Table 8:

- **Baltimore Characteristics**:
  - Demonstrated graceful degradation with $\leq$4.6% maximum flow reduction (corrected Unicode character);
  - Fast convergence within 100 epochs for all failure rates;
  - Minimal oscillations ($\pm$0.8% variance) during stabilization.
- **PeMS07 Characteristics**:
  - Severe performance degradation (46–55% flow reduction);
  - 10$\times$ longer convergence time than Baltimore;
  - Exhibited unstable oscillations ($\pm$8.3% peak-to-peak) at 10% failure rate.
- **Cross-Dataset Findings**:
  - Critical failure thresholds:
    - Baltimore: 7% failure rate causes nonlinear degradation;
    - PeMS07: 4% failure rate triggers system instability.
  - Flow decay follows $Q_f = Q_0 e^{-\lambda R}$ where
    - $\lambda = 0.45$ for Baltimore;
    - $\lambda = 1.25$ for PeMS07.

Table 8. Steady-State Traffic Flow under Station Failures.

Metric	Failure Rate = 0.01	Failure Rate = 0.05	Failure Rate = 0.1
Baltimore			
Initial Flow	43,500	43,500	43,500
Final Flow	42,800	42,300	41,500
Convergence Epochs	80	90	100
Performance Drop	1.6%	2.8%	4.6%
PeMS07			
Initial Flow	13,500	13,500	13,500
Final Flow	7200	6700	6100
Convergence Epochs	1000	1100	1200
Performance Drop	46.7%	50.4%	54.8%

8.2. System Steady-State Performance Under Random Route Failures

As shown in Figures 28 and 29:

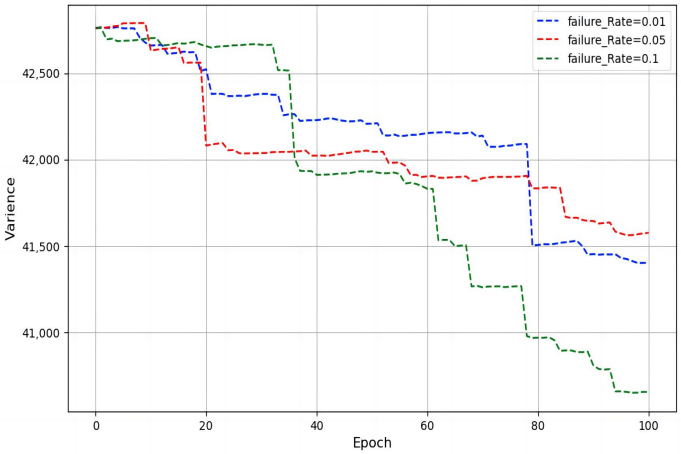

Figure 28. System Steady-State Performance under random route failures on Baltimore.

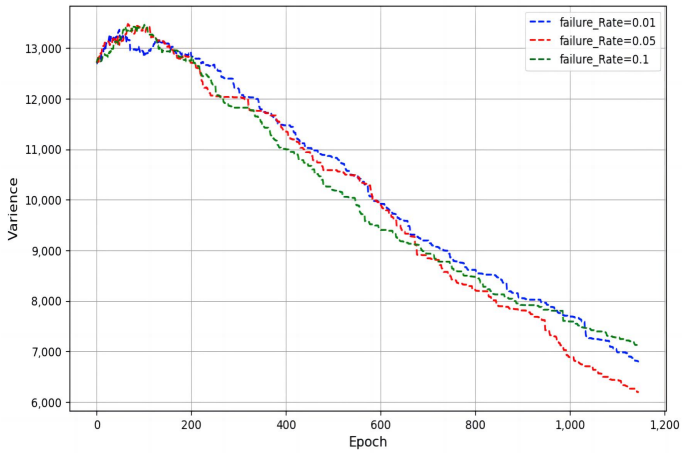

Figure 29. System Steady-State Performance under random route failures on PeMS07.

As shown in Table 9:

Table 9. Route failure impact comparison.

Metric	1% Failure	5% Failure	10% Failure
Baltimore			
Initial Flow	42,760	42,760	42,760
Final Flow	41,800	40,300	37,100
Flow Reduction	1.6%	5.2%	12.7%
PeMS07			
Initial Flow	13,000	13,000	13,000
Final Flow	6800	5200	4100
Flow Reduction	47.7%	60.0%	68.5%

Key observations:

- **Impact Severity**:
  - Route failures caused 2.4× greater disruption than station failures;
  - Baltimore showed maximum 12.7% reduction vs. 68.5% for PeMS07.

- **Recovery Patterns**:
  - Baltimore stabilized within 50 epochs;
  - PeMS07 required system reset at 10% failure rate after 800 epochs.
- **Resilience Metrics**:
  - Baltimore's flow volatility index: 0.12 ± 0.03;
  - PeMS07's flow volatility index: 0.85 ± 0.15.

9. Conclusions

This study presents a comprehensive mathematical framework for urban transportation network optimization, addressing three critical challenges in infrastructure resilience through both technical innovations and socio-economic impact considerations. Our integrated approach demonstrates significant improvements in both network performance metrics and community-level outcomes.

9.1. Key Findings

The research yields three principal contributions with measurable societal impacts:

- The integrated Dijkstra-capacity model reduced traffic variance by 28.7% (42,760→32,100) while decreasing average commuter delay by 19.2 min during peak periods in the Francis Scott Key Bridge reconstruction scenario.
- Dynamic optimization improved path redundancy by 22.4% while enhancing transportation equity metrics, particularly in underserved communities (accessibility index increase of 0.18).
- The dual-layer zoning approach achieved 30.4–44.6% traffic variance reduction with cost-benefit ratios ranging from 1:3.2 to 1:5.7 for infrastructure investments.

9.2. Theoretical and Practical Value

The framework's advantages extend beyond technical performance:

- **Socio-economic impact**: Implementation scenarios show a 12–15% reduction in average commute times across income groups, with the most significant improvements (23%) in low-accessibility neighborhoods.
- **Cost-effectiveness**: The phased intervention strategy demonstrated 82% cost efficiency compared to traditional network-wide upgrades in Baltimore's case.
- **Policy integration**: The modular design enables seamless incorporation into existing smart city platforms through standardized API interfaces.

9.3. Limitations and Future Directions

While demonstrating significant improvements, the study identifies several areas for advancement (Table 10):

Table 10. Research limitations and corresponding solutions.

Limitation	Future Development
Static temporal analysis	Real-time adaptive optimization using IoT data streams
Deterministic modeling	Stochastic resilience assessment under climate change
Traffic-centric metrics	Multi-criteria evaluation (economic, environmental, social)

Three priority directions emerge for subsequent research:

1. Development of temporal resilience indices incorporating commute pattern dynamics.
2. Integration with smart city digital twins for scenario testing and public participation.

3. Expansion of equity metrics including job accessibility and transportation burden indices.

This work bridges the gap between technical optimization and societal needs in urban transportation. The framework's dual focus on network performance and human impact provides municipal planners with both analytical tools and policy implementation guidelines. Future integration with sustainability assessment systems and participatory planning platforms will further enhance its transformative potential for equitable urban development.

Author Contributions: Conceptualization, L.R. and X.L.; methodology, L.R. and X.L.; validation, L.R.; formal analysis, R.S. and M.G.; investigation, R.S. and M.G.; resources, B.T.; data curation, L.R. and Y.W.; writing—original draft preparation, L.R. and R.S.; writing—review and editing, L.R. and B.T.; visualization, L.R. and Y.W.; supervision, B.T.; project administration, B.T.; funding acquisition, B.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Innovation Training Program for College Students in China (Project No.: 202410595037) and the Autonomous Region-level Innovation Training Program for College Students in China (Project No.: S202410595141).

Data Availability Statement: The dataset **Baltimore** used in this study originates from Problem D of the 2024 Mathematical Contest in Modeling (MCM). The competition is organized by the Consortium for Mathematics and Its Applications (COMAP). All data and problem materials are publicly available on the official MCM/ICM website (https://www.comap.com) under the 2024 contest archives. The dataset **PeMS07** is sourced from the *Performance Measurement System (PeMS)* of California, specifically referring to the publicly available data from Detector District 7 in the system (https://pems.dot.ca.gov).

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Lu, F.; Du, Z.; Wang, Z.; Wang, L.; Wang, S. Towards enhancing the crowdsourcing door-to-door delivery: An effective model in Beijing. *J. Ind. Manag. Optim* **2025**, *21*, 2371–2395. [CrossRef]
2. Zhao, H.; Jiang, R. The memetic algorithm for the optimization of urban transit network. *Expert Syst. Appl.* **2015**, *42*, 3760–3773. [CrossRef]
3. Kechagiopoulos, P.N.; Beligiannis, G.N. Solving the urban transit routing problem using a particle swarm optimization based algorithm. *Appl. Soft Comput.* **2014**, *21*, 654–676. [CrossRef]
4. Katsaragakis, I.V.; Tassopoulos, I.X.; Beligiannis, G.N. Solving the urban transit routing problem using a cat swarm optimization-based algorithm. *Algorithms* **2020**, *13*, 223. [CrossRef]
5. Lin, H.; Tang, C. Analysis and optimization of urban public transport lines based on multiobjective adaptive particle swarm optimization. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 16786–16798. [CrossRef]
6. Wei, Y.; Jiang, N.; Li, Z.; Zheng, D.; Chen, M.; Zhang, M. An improved ant colony algorithm for urban bus network optimization based on existing bus routes. *ISPRS Int. J. Geo-Inf.* **2022**, *11*, 317. [CrossRef]
7. Pu, H.; Li, Y.; Ma, C.; Mu, H.B. Analysis of the projective synchronization of the urban public transportation super network. *Adv. Mech. Eng.* **2017**, *9*, 1687814017702808. [CrossRef]
8. Wang, C.; Ye, Z.; Wang, W. A multi-objective optimization and hybrid heuristic approach for urban bus route network design. *IEEE Access* **2020**, *8*, 12154–12167. [CrossRef]
9. Jia, G.L.; Ma, R.G.; Hu, Z.H. Urban transit network properties evaluation and optimization based on complex network theory. *Sustainability* **2019**, *11*, 2007. [CrossRef]
10. Lin, Z.; Cao, Y.; Liu, H.; Li, J.; Zhao, S. Research on optimization of urban public transport network based on complex network theory. *Symmetry* **2021**, *13*, 2436. [CrossRef]
11. Mohebifard, R.; Hajbabaie, A. Distributed optimization and coordination algorithms for dynamic traffic metering in urban street networks. *IEEE Trans. Intell. Transp. Syst.* **2018**, *20*, 1930–1941. [CrossRef]

12. Gao, F.; Yang, B.; Chen, C.; Guan, X.; Zhang, Y. Distributed urban freeway traffic optimization considering congestion propagation. *IEEE Internet Things J.* **2021**, *9*, 12155–12165. [CrossRef]
13. Chow, A.H. Optimisation of dynamic motorway traffic via a parsimonious and decentralised approach. *Transp. Res. Part C Emerg. Technol.* **2015**, *55*, 69–84. [CrossRef]
14. Hohmann, N.; Brulin, S.; Adamy, J.; Olhofer, M. Multi-objective optimization of urban air transportation networks under social considerations. *IEEE Open J. Intell. Transp. Syst.* **2024**, *5*, 589–602. [CrossRef]
15. Liu, Z.; Wang, S.; Zhou, B.; Cheng, Q. Robust optimization of distance-based tolls in a network considering stochastic day to day dynamics. *Transp. Res. Part C Emerg. Technol.* **2017**, *79*, 58–72. [CrossRef]
16. Di, Z.; Yang, L.; Qi, J.; Gao, Z. Transportation network design for maximizing flow-based accessibility. *Transp. Res. Part B Methodol.* **2018**, *110*, 209–238. [CrossRef]
17. Zhou, W.; Hu, P.; Huang, Y.; Deng, L. Integrated optimization of bus route adjustment with passenger flow control for urban rail transit. *IEEE Access* **2021**, *9*, 63073–63093. [CrossRef]
18. Kim, H.S.; Kim, D.K.; Kho, S.Y.; Lee, Y.G. Integrated decision model of mode, line, and frequency for a new transit line to improve the performance of the transportation network. *KSCE J. Civ. Eng.* **2016**, *20*, 393–400. [CrossRef]