Intelligent Systems with Applications 32 (2026) 200741

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications

Digital twin-driven congestion-aware route planning for intelligent transportation systems

Tingjun Lei [a,*], Ketian Ye [a,b], Justin London [a], Sicong Shao [a], Jun Chen [c], Chaomin Luo [d]

[a] School of Electrical Engineering and Computer Science, University of North Dakota, Grand Forks, ND, 58202, USA
[b] Siemens PTI, Siemens Industry, Inc., Alpharetta, GA, 30022, USA
[c] Department of Electrical and Computer Engineering, Oakland University, Rochester, MI, 48309, USA
[d] Department of Electrical and Computer Engineering, Mississippi State University, Starkville, MS, 39762, USA

ARTICLE INFO

Keywords:
Digital twin
Route planning
Traffic management
Bacterial foraging optimization
Chaos theory
Intelligent transportation systems

ABSTRACT

With the rapid advancement of autonomous driving and navigation technologies, an increasing number of vehicles rely on real-time routing services such as Google Maps and Waze. However, these services frequently fail to improve aggregate traffic conditions, as rerouting drivers based on existing congestion amplifies herding behavior and produces new secondary bottlenecks. To address this issue, a digital twin-based dynamic route-planning framework is proposed, in which real-time vehicle detection, surrogate-modeled traffic-flow prediction, and a congestion-aware metaheuristic router are integrated into a closed-loop pipeline. A network of intersection cameras with a YOLOv8 detector supplies real-time inflow and outflow data, which is fed into a Lighthill–Whitham–Richards (LWR)-based simulator incorporating signal, lane, and weather factors. The counting stage is validated at the level of individual crossing events on continuously annotated traffic sequences. To deliver real-time predictions at network scale, a parallel partial Gaussian stochastic process (PP-GaSP) surrogate is adapted to edge-level traffic volumes, providing sub-millisecond predictions with closed-form Student-t uncertainty that the router consumes as a risk-inflated flow estimate. Since the single-vehicle social-cost objective is edge-separable and solved exactly by a modified Dijkstra search, the paper addresses routing where herding actually arises, jointly optimizing a batch of concurrent demand streams whose coupled objective contains a non-separable network-bottleneck term beyond the reach of per-vehicle best-response methods. A new metaheuristic, Adaptive Chaotic Bacterial Foraging Optimization (ACBFO) is proposed to solve the batch problem through congestion-guided chemotaxis, which reroutes demand streams off the current fleet bottleneck within a chaotically scheduled bacterial foraging search. In paired experiments, ACBFO attains the provably optimal single-vehicle solution on every instance and significantly outperforms exact sequential routing, iterated best response, and metaheuristic and learning-based baselines, with the largest gains on the fleet bottleneck ratio, and a multi-window closed-loop replay exercises the prediction, joint-routing, and feedback stages under time-varying boundary-count inputs.

1. Introduction

Traffic congestion has long been a major issue in urban areas, occurring when traffic demand exceeds road capacity. With the rapid advancement of mobile smart devices and wireless communication technologies, drivers now heavily rely on Traveler Information Systems (TIS) such as Google Maps, Apple Maps, and Waze to plan optimal travel routes (Lei, Luo, et al., 2023). These services typically prioritize travel-time reduction, although the initial route planning may incorporate additional factors such as fuel efficiency, toll avoidance, and other economic considerations. Autonomous vehicles, including those manufactured by Tesla and Waymo, also rely on the same class of services to navigate efficiently.

Despite their utility, a significant limitation arises when many vehicles select the same route based on identical real-time information (Chintam et al., 2024; Lei, Sellers, et al., 2023; Sellers et al., 2022; Short et al., 2023; Tao et al., 2018; Wang et al., 2022). Although systems such as Google Maps reroute drivers based on updated traffic conditions, this reactive strategy is unable to mitigate pre-existing congestion and may even contribute to secondary traffic build-ups by redirecting vehicles to alternative but equally congested routes (Jan et al., 2024; Riser

* Corresponding author.
E-mail address: Tingjun.Lei@UND.edu (T. Lei).

https://doi.org/10.1016/j.iswa.2026.200741
Received 10 May 2026; Received in revised form 4 September 2026; Accepted 6 October 2026
Available online 23 October 2026
2667-3053/© 2026 Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

et al., 2024). The fundamental cause is well known in transportation theory (Sheffi, 1985; Wardrop, 1952): minimizing each user's travel time independently drives the system toward a user equilibrium that is generally suboptimal at the network level. To address this challenge, a digital twin is leveraged in this paper to provide real-time road information for dynamic route planning of smart vehicles, with the aim of mitigating congestion proactively and optimizing traffic flow across urban networks.

1.1. Related work

Intelligent transportation systems (ITS) have attracted growing research interest in recent years, since traffic congestion poses serious threats to long-term city planning and growth. A wide range of factors contribute to traffic congestion, including construction zones, adverse weather, special events, and accidents (Gomes et al., 2023). Accordingly, dynamic route planning under real-time traffic conditions requires the joint development of three tightly coupled components, including vehicle detection and counting at intersections, traffic-flow prediction at the network level, and congestion-aware route planning. Each of these components has been extensively studied, but seldom integrated into a unified framework.

Vehicle detection and counting from traffic videos has become a foundational component of ITS (Lin et al., 2022). Two complementary counting paradigms have been developed in the literature. The first is the virtual detection-area paradigm, in which a virtual reference line or region is set up in the camera frame and counts are incremented whenever vehicles cross it. The second is the vehicle-tracking paradigm, which extracts the trajectory of each vehicle by matching detections across consecutive frames within a detection-tracking-counting (DTC) pipeline and counts vehicles based on the resulting trajectories (Lei et al., 2024; Lin et al., 2022; Lu et al., 2021; Sellers et al., 2024). The two paradigms trade off accuracy against computational cost. Virtual detection is highly efficient but error-prone under lane changes and parallel driving, whereas vehicle tracking is more accurate but substantially more expensive. Existing methods generally rely on detectors that identify vehicles based on their appearance and features, which are localized through foreground detection techniques. Background subtraction (Mandellos et al., 2011), frame differencing (Kasturi et al., 2009), and optical flow rely on hand-crafted features and are simple but error-prone. Deep-learning detectors achieve much higher accuracy via automatic feature extraction, at the cost of computation and the need for annotated data (Abdullah & Oothariasamy, 2020; London, Lei, Luo & Yuan, 2025; London, Lei, Yuan, et al., 2025). Object detectors themselves are commonly classified as two-stage detectors such as the R-CNN family, which feed region proposals into a downstream classifier and regressor for higher accuracy at the expense of efficiency, and one-stage detectors such as the YOLO family (Redmon et al., 2016) and SSD, which treat detection as a single regression problem and are therefore far better suited to real-time deployment. Among one-stage detectors, the YOLO family has emerged as the dominant choice for real-time counting due to its competitive accuracy and ability to assign type labels to detected vehicles. Vasu et al. (2021) integrated YOLOv4 with DeepSort (Wojke et al., 2017) and k-means trajectory clustering to improve counting robustness, and Dey and Kundu (2019) demonstrated transfer learning from MobileNet for urban intersection analysis. Adl et al. (2016) further introduced a hybrid scheme that combines zone-based and path-based counting to handle long and broken tracks captured by fisheye cameras. However, vehicle counting still poses challenges due to occlusion, poor illumination, and adverse weather conditions, especially under fisheye distortion that introduces variable orientations and scales (Adl et al., 2016; Gochoo et al., 2023).

Beyond per-intersection counting, network-scale traffic-flow prediction has long been recognized as a critical building block of intelligent transportation. Gomes et al. (2023) provided a comprehensive survey of traffic-flow prediction and classification, highlighting the predominant role of historical data and the rising importance of real-time data integration in modern ITS. Macroscopic traffic-flow models such as the Lighthill–Whitham–Richards (LWR) PDE model (Lighthill & Whitham, 1955; Richards, 1956) provide a principled foundation that captures shock-wave behavior and flow-density coupling. However, the numerical solution of the LWR PDE is too slow for real-time use over a city-scale graph. Data-driven surrogates trained on simulation outputs offer a practical alternative that combines physical realism with millisecond-level inference. Gaussian processes (Rasmussen & Williams, 2006) have shown particular promise in this regard owing to their non-parametric flexibility and predictive uncertainty estimates. Nevertheless, scaling Gaussian processes to city-scale road networks remains a non-trivial challenge, since the cubic complexity of standard inference quickly becomes prohibitive without targeted approximations.

Once predicted edge volumes are available, the routing problem becomes the search for an origin–destination path that minimizes a flow-dependent travel-time function. Classical graph-search methods such as Dijkstra and A* provably yield the shortest path under static edge weights, but they implicitly assume that each user's optimal decision is independent of the system-level consequences. To overcome this limitation, metaheuristic algorithms have been widely explored. Particle swarm Optimization (Kennedy & Eberhart, 1995), Grey Wolf Optimizer (Mirjalili et al., 2014), whale Optimization algorithm (Mirjalili & Lewis, 2016), and arithmetic Optimization (Abualigah et al., 2021) have all been applied to path-planning and related combinatorial problems. Bacterial Foraging Optimization (BFO), originally proposed by Passino (Passino, 2002), mimics the chemotactic behavior of *Escherichia coli* and has shown competitive performance in path planning. Lei et al. (2021) proposed a hybrid bat-pigeon algorithm that combines the global search of pigeon-inspired Optimization with the local search of the bat algorithm, demonstrating that hybridizing complementary search strategies can substantially improve convergence. Ponti et al. (2021) introduced a Wasserstein distance-based multi-objective evolutionary algorithm that further illustrates the value of customized genetic operators for domain-specific Optimization. Despite these advances, most existing metaheuristic routers still optimize only individual travel time and do not explicitly account for the externalities imposed on the network when many vehicles follow the same recommended path. Consequently, the secondary-bottleneck problem at the heart of consumer navigation services remains insufficiently addressed by these individual-optimal formulations.

In parallel with these metaheuristic developments, a complementary line of work has explored deep learning for both traffic-flow prediction and routing. Spatio-temporal graph neural networks such as DCRNN (Li et al., 2018), STGCN (Yu et al., 2018), and Graph WaveNet (Wu et al., 2019) achieve strong predictive accuracy on benchmarks like METR-LA and PeMS-BAY, while deep reinforcement learning has been applied to congestion-aware routing (Koh et al., 2020). Recent network-level route-guidance studies have further used deep reinforcement learning together with realistic compliance models or system-optimal objectives (Hicks et al., 2025; Jiang et al., 2024; Yun et al., 2024). These methods provide an important learning-based comparison point, but their state spaces, demand models, and policy-training assumptions differ from the batch joint-assignment formulation considered here. Compared with these data-driven approaches, the Gaussian-process surrogate adopted here offers two properties that are difficult to obtain from neural surrogates. The first is a closed-form predictive distribution that yields per-edge uncertainty at no additional cost; the second is tractable training from relatively few simulator queries owing to its non-parametric nature. Coupled with a metaheuristic router that does not require offline policy training, the proposed framework remains deployable in cold-start scenarios where labeled trajectory data is scarce.

In summary, although vehicle counting, traffic-flow prediction, and route planning have all been studied extensively in isolation, few existing works integrate them into a closed-loop digital twin that delivers congestion-aware routing in real time. The framework proposed in this paper aims to fill this gap.

1.2. Proposed methods and original contribution

In this paper, a digital twin-driven dynamic route-planning framework is proposed for intelligent transportation systems. A network of strategically placed intersection cameras is deployed to detect entry and exit vehicles using a YOLOv8 detector, providing real-time inflow and outflow measurements to the downstream models. The detected counts are propagated across the network by an LWR-based simulator that incorporates signal, lane, and weather factors, after which a parallel partial Gaussian stochastic process (PP-GaSP) surrogate is trained on the simulation outputs to predict edge-level traffic volumes at network scale within a fraction of a millisecond, together with per-edge predictive uncertainty.

Building on the predicted volumes, route planning is posed at the level at which the herding problem actually arises, a batch of concurrent origin–destination demand streams that must be routed jointly. We first show that the per-vehicle social-cost objective is edge-separable and hence exactly solvable by a modified Dijkstra search. The batch formulation, in contrast, couples all streams through the induced flows and contains a non-separable network-bottleneck term that per-vehicle best-response methods are structurally unable to optimize. An Adaptive Chaotic Bacterial Foraging Optimization (ACBFO) algorithm is developed for the batch problem. Its congestion-guided chemotaxis identifies the current fleet bottleneck edge and reroutes vehicles around it, a chaotic operator schedule driven by a probability-integral-transformed logistic map balances exploration and exploitation, and a clipped Lévy-flight elimination-dispersal stage restores population diversity. The contributions of this paper are summarized as follows:

1. A digital twin-driven dynamic route-planning framework is proposed that integrates real-time vehicle detection, surrogate-modeled traffic-flow prediction, and congestion-aware metaheuristic routing into a unified pipeline. A multi-window closed-loop replay exercises the prediction, joint-routing, and feedback stages under time-varying boundary-count inputs, in which the framework suppresses the cumulative herding divergence exhibited by reactive free-flow routing. The perception module is evaluated separately at both frame and event levels.
2. A YOLOv8-based vehicle detection and counting module is developed, in which intersection cameras are deployed to detect entry and exit vehicles and supply real-time inflow and outflow data. Beyond frame-level detection metrics on real traffic images, the short-horizon association and virtual-line-crossing stages are quantitatively validated on 40 controlled continuous sequences spanning moderate, dense, bidirectional, and mixed-speed traffic, with 3.31% aggregate APE and 98.23% event F1.
3. A PP-GaSP surrogate is adapted to edge-level traffic prediction with a product Matérn kernel, periodic time-of-day encoding, and boundary-count inputs. The surrogate matches the simulator at a 200× lower query latency, and its closed-form Student-t predictive distribution is exploited operationally. Prediction intervals widen automatically under out-of-distribution conditions, and the router consumes a risk-inflated flow estimate that measurably improves realized routing performance, a form of uncertainty quantification that neural surrogates do not provide at comparable cost.
4. Routing is reformulated as batch joint optimization of concurrent demand streams, and ACBFO is designed so that its congestion-guided chemotaxis attacks the non-separable bottleneck term directly. In 30 paired instances, ACBFO attains the provably optimal single-vehicle solution on every instance, significantly outperforms sequential exact routing, iterated best response, six metaheuristic baselines, and a learning-based router on the batch objective, and yields the lowest fleet worst-edge volume-to-capacity ratio, with every mechanism validated by a leave-one-out ablation.

The remainder of this paper is organized as follows. The digital twin architecture and the LWR-based traffic-flow modeling problem with realistic factors are formulated in Section 2. The YOLOv8-based vehicle detection and counting module is presented in Section 3. The PP-GaSP surrogate for real-time, network-scale traffic-volume prediction is developed in Section 4. The batch joint-routing formulation and the proposed ACBFO algorithm are presented in Section 5. Experimental results on vehicle detection, traffic-flow modeling, and route planning are reported in Section 6. Conclusions and future work are summarized in Section 7.

2. Digital twin architecture and traffic flow modeling

In this section, the proposed digital twin architecture is described, and the underlying traffic flow model that drives the digital twin is formulated. The digital twin is constructed as a closed-loop pipeline that mirrors the operating state of the physical road network in real time, predicts how the network will evolve under different routing decisions, and feeds the predictions back to a congestion-aware route planner. The vehicle counting problem and the LWR traffic flow model with realistic factors are introduced as the perceptual and simulation cores of the digital twin, respectively. The numerical solution of the LWR model is then derived to support real-time inference.

2.1. Digital twin system overview

The proposed digital twin couples a physical layer with a virtual layer through a continuous data exchange loop, as illustrated in Fig. 1. In the physical layer, a network of intersection cameras is deployed to monitor inflows and outflows at each road segment of interest. The collected video streams are processed by a YOLOv8 detector in the perception layer, by which entry and exit counts are produced at each node in real time. These counts are treated as the boundary conditions of the virtual layer, which is formed by an LWR-based traffic flow simulator that incorporates signal, lane, and weather factors. The simulator is responsible for propagating intersection-level counts across the entire road graph and predicting edge-level traffic volumes under varying conditions.

To make the digital twin operate at the temporal resolution required by route planning, a PP-GaSP surrogate is trained on simulation outputs and integrated into the virtual layer, as detailed in Section 4. The surrogate is queried by the proposed ACBFO algorithm in Section 5, by which congestion-aware routes are generated and dispatched back to vehicles in the physical layer. As vehicles follow the recommended routes, intersection counts are updated and the loop is closed. Through this closed-loop interaction, the digital twin not only reflects the present state of the network but also anticipates the consequences of routing decisions before they are realized on the physical road. The perception, simulation, and decision components of the digital twin are detailed in Sections Section 3, 2.2, and 5, respectively.

2.2. LWR traffic flow model with realistic factors

With the intersection counts supplied by the vehicle counting module, edge-level traffic states across the entire network must be derived to support route planning. To this end, a macroscopic traffic flow model is adopted as the core of the virtual layer of the digital twin. The LWR model (Lighthill & Whitham, 1955; Richards, 1956) is selected for its well-established capability to capture shock-wave behavior and flow-density coupling, and it has been widely accepted as a foundational tool in traffic flow theory.

In the LWR model, traffic on a road segment is treated as a one-dimensional compressible fluid whose dynamics are governed by the continuity equation

$$\frac{\partial \rho}{\partial t} + \frac{\partial Q}{\partial x} = 0, \tag{1}$$

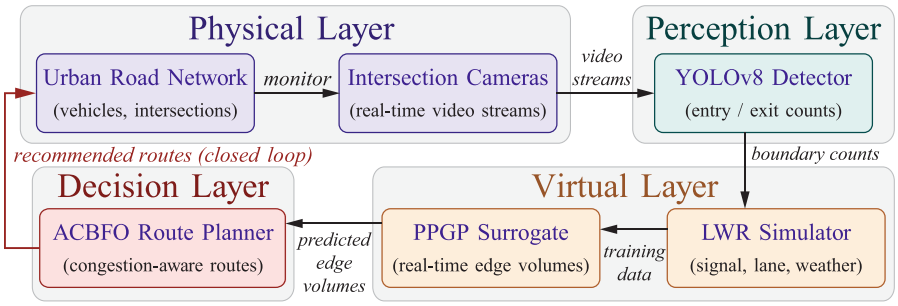


Fig. 1. The proposed digital twin architecture for congestion-aware route planning. Vehicle counts produced by the YOLOv8 detector in the perception layer are fed into the LWR simulator and the PP-GaSP surrogate in the virtual layer, which together predict edge-level traffic volumes in real time. The volumes are then consumed by the ACBFO route planner in the decision layer, whose recommended routes are dispatched back to vehicles in the physical layer to close the loop.

where ρ denotes the traffic density (veh/length) and Q denotes the traffic flow (veh/time). A triangular fundamental diagram is adopted to link flow and density,

$$Q(\rho) = \begin{cases} V_{\max}\,\rho, & 0 \le \rho \le \rho^*, \\ w_b\,(\rho_{\max} - \rho), & \rho^* < \rho \le \rho_{\max}, \end{cases} \tag{2}$$

in which $\rho_{\max} = 1/L_v$ is the jam density determined by the effective vehicle length L_v, w_b is the backward wave speed that propagates congestion upstream (typically 16 to 25 km/h), and the critical density ρ^* is set by the continuity condition $V_{\max}\rho^* = w_b(\rho_{\max} - \rho^*)$, which yields $\rho^* = w_b\rho_{\max}/(V_{\max} + w_b)$ and a capacity of $Q_{\max} = V_{\max}\rho^*$. The maximum velocity is $V_{\max} = \min(V_{\lim}, V_{\text{free}})$. To make the digital twin sensitive to real-world operating conditions, the free-flow speed V_{free} is modulated by three influence factors related to traffic signals, lane configuration, and weather:

$$V_{\text{free}} = F_{\text{sig}} \cdot F_w \cdot F_l \cdot V_0, \tag{3}$$

where V_0 is the optimal free-flow speed of the road segment. The signal factor F_{sig} captures the slowdown effect imposed by the density and timing of traffic signals along the segment and is computed as

$$F_{\text{sig}} = 1 - w_{\text{sig}}(t), \tag{4}$$

$$w_{\text{sig}}(t) = \gamma_1\,\sigma(N_{\text{sig}}) + \gamma_2\,\sigma(L_{\text{sig}}), \tag{5}$$

in which N_{sig} and L_{sig} denote the number of signals and the average red-light duration on the segment, γ_1 and γ_2 are weighting coefficients, and $\sigma(x) = 1/(1+e^{-x})$ is the logistic squashing function. The lane factor F_l captures the impact of lane configuration through

$$F_l = \left(1 - \frac{\alpha_1}{N_l}\right)\left(1 - \frac{\alpha_2}{L_l}\right), \tag{6}$$

where N_l is the number of lanes, L_l is the average lane width, and α_1 and α_2 are tuning parameters. The weather factor F_w is treated as a categorical multiplier whose value is selected from a small set of pre-calibrated coefficients corresponding to sunny, rainy, and snowy conditions. Through this parameterization, a wide range of operating scenarios can be represented by the digital twin with only a small number of physically interpretable parameters, and the model remains analytic enough to support fast numerical solution.

2.3. Numerical solution of the LWR model

For the LWR model to drive the virtual layer of the digital twin in real time, an efficient and stable numerical scheme is required. Space and time are discretized on a uniform grid, and a first-order monotone upwind discretization is used with the differencing direction selected from the sign of the characteristic speed. The spatial derivative $\partial Q/\partial x$ is approximated by first-order finite differences whose direction is selected according to the propagation of information:

$$\frac{\partial Q}{\partial x} \approx \begin{cases} \dfrac{Q_{k,j} - Q_{k-1,j}}{\Delta x}, & \text{free flow}, \\ \dfrac{Q_{k+1,j} - Q_{k,j}}{\Delta x}, & \text{congested flow}, \end{cases} \tag{7}$$

where k and j index spatial cells and time steps, respectively. The temporal derivative is discretized by the explicit Euler scheme, which yields the update rule

$$\rho_{k,j+1} = \rho_{k,j} - \frac{\Delta t}{\Delta x}\left(Q^{(\text{out})}_{\bullet,j} - Q^{(\text{in})}_{\bullet,j}\right), \tag{8}$$

where the difference is taken in the appropriate upwind or downwind direction in accordance with (7). To ensure numerical stability, the time step is constrained by the Courant–Friedrichs–Lewy (CFL) condition

$$\Delta t \le \frac{\Delta x}{\max\left(V_{\max},\, w_b\right)}, \tag{9}$$

in which the characteristic speed $|\partial Q/\partial\rho|$ equals $V_{\max}$ in the free-flow branch and w_b in the congested branch of the triangular fundamental diagram of (2). Under this condition, the explicit scheme is numerically stable for the triangular flux used here. The present study uses it as the physics-based traffic generator rather than claiming independent field validation of the simulator. Although the resulting solver is reasonably efficient, repeated evaluation across all edges and time steps of a city-scale network is still computationally expensive for online use. To overcome this limitation, a surrogate model is developed in Section 4, by which the simulator is replaced with a Gaussian-process predictor whose inference cost is several orders of magnitude lower while closely reproducing the LWR simulator over the evaluated operating range.

3. Vehicle detection and counting at intersections

The fidelity of the digital twin is bounded by the quality of the measurements that close its perception loop, since the LWR simulator developed in Section 2.2 and the surrogate developed in Section 4 both consume the inflow and outflow counts produced at each intersection as their boundary conditions. The perception module of the digital twin is therefore designed around three requirements: it must operate in real time on streaming video from intersection cameras, it must be robust to common adverse conditions such as occlusion, low illumination, and inclement weather, and it must distinguish between vehicle types so that heterogeneous flows can be reasoned about by the downstream layers. A YOLOv8-based perception module is developed to meet these requirements, in which a one-stage detector is paired with a virtual region-of-interest counting paradigm and a type-aware aggregation layer.

3.1. Counting problem and performance metrics

Let the video stream from a single intersection camera be partitioned into non-overlapping counting windows of duration ΔT. For each window, the perception module is required to produce a count vector

$$\mathbf{n}_i(t) = \left[n_i^{(1)}(t),\, n_i^{(2)}(t),\, \ldots,\, n_i^{(C)}(t)\right]^{\top}, \tag{10}$$

in which $n_i^{(c)}(t)$ denotes the number of vehicles of class $c \in \{1, \dots, C\}$ that have crossed the virtual reference line at intersection node i during the window ending at time t. The per-class counts are subsequently summed into per-edge inflow and outflow rates that are passed to the LWR simulator as boundary conditions of the conservation law in Eq. (1).

Two scalar metrics are adopted to evaluate the counting accuracy of the perception module. Let N_{det} and N_{gt} denote the total detected and ground-truth vehicle counts in a given window, respectively. The vehicle counting accuracy (VCA) is defined as

$$\text{VCA} = \frac{N_{\text{det}}}{N_{\text{gt}}} \times 100\%, \tag{11}$$

and the absolute percentage error (APE) is given by

$$\text{APE} = \left| \frac{N_{\text{det}} - N_{\text{gt}}}{N_{\text{gt}}} \right| \times 100\%. \tag{12}$$

For event-level validation, let N_{match} denote one-to-one matched predicted and ground-truth crossing events within a temporal tolerance, N_{miss} the unmatched ground-truth events, N_{fp} the unmatched predicted events, N_{dup} repeated predictions associated with an already counted ground-truth crossing, and N_{dir} matched events with the correct crossing direction. We report

$$R_{\text{miss}} = \frac{N_{\text{miss}}}{N_{\text{gt}}}, \qquad R_{\text{dup}} = \frac{N_{\text{dup}}}{N_{\text{gt}}}, \tag{13}$$

$$A_{\text{dir}} = \frac{N_{\text{dir}}}{N_{\text{match}}}, \qquad F_1^{\text{event}} = \frac{2PR}{P+R}, \tag{14}$$

where $P = N_{\text{match}}/(N_{\text{match}} + N_{\text{fp}})$ and $R = N_{\text{match}}/N_{\text{gt}}$. These metrics separate missed crossings, spurious or duplicate crossings, and direction errors that can be hidden by aggregate count ratios. Because duplicate counts can drive N_{det} above N_{gt}, VCA can exceed 100% and is interpreted as a count ratio rather than an accuracy. It is therefore complemented by the missed-count rate, duplicate-count rate, entry/exit direction accuracy, and event F1 in the continuous-sequence evaluation of Section 6.1. VCA and APE summarize aggregate count agreement, while the event-level metrics above diagnose the failure modes of the association and crossing stages. Achieving both simultaneously is nontrivial in practice because three classes of nuisance factors commonly degrade vehicle counting performance: *occlusion*, in which one vehicle is partly hidden behind another or behind roadside objects; *adverse illumination*, ranging from headlight glare at night to back-lit silhouettes at dusk; and *adverse weather*, in which rain, snow, or fog reduces contrast and introduces motion blur (Lin et al., 2022). The detector architecture and the post-processing pipeline described in the following subsections are designed with these factors in mind.

3.2. Counting paradigms

Two counting paradigms have been studied extensively in the literature, and the design of the perception module of the digital twin draws on both. The first is the *virtual detection-area* paradigm, in which one or more virtual reference lines or regions are drawn on the camera frame, and the count is incremented whenever a detected bounding box crosses the line in a specified direction. This paradigm is highly efficient because no inter-frame correspondence is required, but it is sensitive to lane changes and parallel driving and may miss fast vehicles that traverse the reference line between consecutive frames. To mitigate this, an additional inner contour can be introduced behind the outer reference line, by which the effective detection area is expanded and missed crossings are recovered (Ibryaeva et al., 2019).

The second is the *vehicle-tracking* paradigm, which is commonly realized as a detection-tracking-counting (DTC) pipeline (Lin et al., 2022; Lu et al., 2021). In a DTC pipeline, vehicles are first detected in each frame, then associated across frames into trajectories using tracking algorithms such as DeepSort (Wojke et al., 2017), and finally counted on the basis of the resulting trajectories. A common similarity metric for the association step combines an appearance term, a motion term, and a shape term:

$$d(i,j) = d_{\text{ap}}(i,j) + d_{\text{motion}}(i,j) + d_{\text{shape}}(i,j),$$
$$d_{\text{ap}}(i,j) = 1 - \cos(C_i, C_j), \tag{15}$$

in which C_i, C_j are color histograms of the tracked and detected boxes, and d_{motion} and d_{shape} measure displacement and aspect-ratio change normalized by the bounding-box dimensions (Lu et al., 2021). Although vehicle tracking achieves higher accuracy than virtual detection because identity is preserved across frames, it is substantially more expensive both in feature extraction and in association.

A hybrid strategy is therefore adopted, in which the perception module of the digital twin uses the virtual detection-area paradigm as its default counting mechanism for efficiency, and falls back to a lightweight short-horizon tracking step only when overlapping detections are observed near the reference line. This design preserves the real-time performance required by the closed-loop digital twin while retaining the robustness of trajectory-based counting in difficult cases.

3.3. YOLOv8-based detector

A one-stage detector is preferred over a two-stage detector for the perception module of the digital twin, because the strict latency budget imposed by the closed-loop pipeline favors the single-pass regression formulation of one-stage detectors. Among the available one-stage architectures, YOLOv8 (Jocher et al., 2023) is adopted in this work for three reasons: (i) it provides competitive accuracy on small-object detection in traffic scenes, where vehicles far from the camera occupy only a small fraction of the frame; (ii) it natively supports multi-class classification, which is required for the type-aware counting strategy described below; and (iii) its frame rate on a single consumer GPU comfortably exceeds the temporal resolution at which the digital twin is updated.

A YOLOv8 detector takes a frame $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$ as input and outputs a set of detections

$$\mathcal{D}(\mathbf{I}) = \left\{ (b_k, c_k, p_k) \,\middle|\, p_k \geq \tau_p \right\}_{k=1}^{K}, \tag{16}$$

in which $b_k \in \mathbb{R}^4$ is the predicted bounding box, $c_k \in \{1, \dots, C\}$ is the predicted class label, $p_k \in [0,1]$ is the predicted confidence, and τ_p is a confidence threshold tuned on the validation set. The detection branch internally evaluates a CSP-based backbone with a feature-pyramid neck and a decoupled head that separates the classification and regression tasks. The detector is supervised by a composite loss

$$\mathcal{L} = \mathcal{L}_{\text{box}} + \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{dfl}}, \tag{17}$$

consisting of a complete-IoU box-regression term $\mathcal{L}_{\text{box}}$, a binary cross-entropy classification term $\mathcal{L}_{\text{cls}}$, and a distribution focal-loss term $\mathcal{L}_{\text{dfl}}$ that sharpens the predicted box-distribution. The three terms are summed with default weights and optimized end-to-end during training, by which both localization and classification are jointly improved.

3.4. From per-frame detections to inflow/outflow counts

The detections in Eq. (16) are per-frame observations and must be aggregated into per-window counts before they can be supplied as boundary conditions of the LWR simulator. A four-step counting pipeline is therefore applied to each intersection camera, as illustrated in Fig. 2.

Step 1: Region-of-interest definition. For each camera, a polygonal region of interest (ROI) and a virtual reference line are specified offline based on the road geometry visible in the frame. The ROI excludes parked vehicles, pedestrian areas, and irrelevant background, while the reference line is positioned across the lanes of interest at a location with low occlusion.

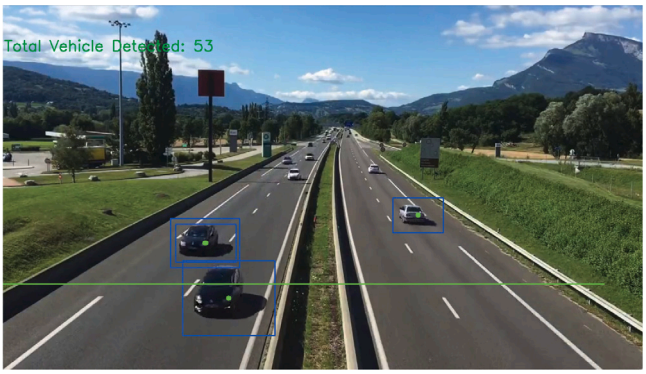


Fig. 2. YOLOv8-based vehicle counting at a virtual reference line. Detected vehicles are enclosed by bounding boxes, their centroids are tracked across frames, and the running total is incremented whenever a centroid crosses the green reference line. The same mechanism is applied at every intersection camera of the digital twin to produce per-class inflow and outflow counts.

Step 2: Per-frame detection. The YOLOv8 detector is applied to every rth frame of the stream, where r is selected such that the inter-frame displacement of a vehicle traveling at the segment's free-flow speed remains smaller than half of its bounding-box length. This sampling rule is intended to reduce the chance that a fast vehicle crosses the reference line entirely between two consecutive analyzed frames.

Step 3: Short-horizon track association. The implementation used in this work deliberately keeps the association stage lightweight. Near-duplicate same-class detections within 10 pixels are suppressed first. Each remaining centroid is greedily associated with the nearest class-consistent track after a constant-velocity prediction, with a 32-pixel base gate that expands modestly when one or two analyzed frames are missed. The track velocity is exponentially updated with weights 0.55 and 0.45 on the previous and new estimates, respectively. Tracks survive at most two missed frames and are confirmed after four associated detections. This geometry-only implementation omits the appearance term in Eq. (15); the equation is retained above to describe the more general DTC association family. The simpler implementation is sufficient for short-horizon line crossing and makes the counting stage inexpensive and reproducible.

Step 4: Type-aware counting. A line-crossing event is declared whenever a track centroid moves from one side of the reference line to the other. To reject one-frame jitter, the track must already be confirmed and must have been observed at least 18 pixels away from the line before the crossing; each track can contribute at most one crossing event. Crossing direction follows the sign of the centroid motion. The class label $c_k \in \{\text{car}, \text{bus}, \text{truck}, \text{motorbike}, \text{person}\}$ predicted at the crossing is used to increment the corresponding entry of the count vector $\mathbf{n}_i(t)$ in Eq. (10). The class breakdown is preserved through to the LWR simulator because heterogeneous vehicle classes exert different demands on capacity. Heavy vehicles such as buses and trucks contribute disproportionately to congestion and are commonly weighted more heavily in passenger-car-equivalent flow calculations (Lin et al., 2022).

3.5. Integration with the digital twin

The class-specific counts produced by the perception module are converted to scalar inflow and outflow rates by a passenger-car equivalence (PCE) weighting,

$$Q_i^{\text{in}}(t) = \frac{1}{\Delta T} \sum_{c=1}^{C} \pi_c \, n_i^{(c)}(t), \tag{18}$$

in which π_c is the PCE coefficient for class c (the `person` class is excluded from the vehicular PCE aggregation and retained only for detection reporting, since pedestrians do not contribute vehicular flow) and the outflow rate is defined analogously. These rates are stamped with the timestamp t, the camera-specific environmental indicators (weather, day of week, time of day), and the corresponding road identifier, and then dispatched to the LWR simulator of Section 2.2 as Dirichlet boundary conditions. The same tuple is also written into the training database that supports the PP-GaSP surrogate of Section 4, which makes the perception module the upstream provider of both the physics-based and the data-driven branches of the virtual layer of the digital twin. Implementation details, datasets, and quantitative evaluation of the perception module are reported in Section 6.1.

4. Surrogate modeling for traffic volumes

The simulator developed in Section 2.2 is physics-based but computationally expensive when evaluated repeatedly over all edges and time steps of a city-scale network. To make the digital twin operate in real time, a parallel partial Gaussian stochastic process (PP-GaSP) surrogate is trained on simulation outputs and integrated into the virtual layer, as the data-driven counterpart of the physics-based simulator. Three properties motivate the choice of PP-GaSP over alternative surrogates: (a) Gaussian processes (GPs) are non-parametric and recover the underlying mapping with very few training samples, which matters because the simulator is itself expensive to query; (b) the parallel partial Gaussian stochastic process (PP-GaSP) formulation introduced by Gu and Berger (2016), implemented in the RobustGaSP framework (Gu et al., 2019), couples per-output mean and variance parameters with shared kernel parameters, which lets statistical strength be borrowed across edges without forcing them to share scale; and (c) the predictive distribution is a closed-form Student-t, so predictive uncertainty estimates are obtained at no additional cost and can be used by the digital twin to flag low-confidence edges to the route planner.

4.1. PP-GaSP formulation for network traffic volumes

Each road edge $\ell \in \{1, \dots, L\}$ is endowed with its own mean parameter μ_ℓ and variance σ_ℓ^2, while the parameters of the covariance kernel are shared across edges so that statistical strength is borrowed from the network as a whole. The traffic volume on edge ℓ at input λ is modeled as

$$Q_\ell(\lambda, t) = \mathcal{F}_\ell(\lambda, t) + \epsilon_\ell, \tag{19}$$

where λ collects environmental and demand inputs such as time of day, day of week, and weather indicators, $\mathcal{F}_\ell$ is a stationary Gaussian process with mean μ_ℓ and covariance $\sigma_\ell^2 \, k(\lambda, \lambda'; \boldsymbol{\gamma})$, and $\epsilon_\ell \sim \mathcal{N}(0, \sigma_\ell^2 \nu)$ is a small Gaussian observation noise whose variance is tied to the GP scale through the nugget parameter ν. Marginalizing the latent function over a set of training inputs $\{\lambda_1, \dots, \lambda_n\}$ yields a multivariate normal (MVN) distribution

$$[Q_\ell(\lambda_1), \dots, Q_\ell(\lambda_n)]^\top \sim \mathcal{N}\big(\mu_\ell \mathbf{1}_n, \; \sigma_\ell^2 (\mathbf{K} + \nu \mathbf{I}_n)\big), \tag{20}$$

where $\mathbf{K}$ is the kernel matrix with entries $\mathbf{K}_{ij} = k(\lambda_i, \lambda_j; \boldsymbol{\gamma})$ and $\mathbf{1}_n$ is the all-ones vector. Stacking the L outputs yields the data matrix $\mathbf{Q}^D \in \mathbb{R}^{n \times L}$ whose columns are the per-edge observations.

4.2. Covariance kernel and prior

The inputs λ are heterogeneous. Time of day is periodic and is therefore encoded by the pair $(\sin 2\pi t/24, \cos 2\pi t/24)$, weather is categorical and enters through one-hot indicators, and all inputs are standardized. Following the standard PP-GaSP construction (Gu & Berger, 2016; Gu et al., 2019), a separable product kernel with dimension-specific ranges is adopted,

$$\sigma_\ell^2 \, k(\lambda, \lambda') = \sigma_\ell^2 \prod_{k=1}^{p} \left(1 + \frac{\sqrt{5} h_k}{\gamma_k} + \frac{5 h_k^2}{3 \gamma_k^2}\right) \exp\left(-\frac{\sqrt{5} h_k}{\gamma_k}\right), \qquad h_k = |\lambda_k - \lambda_k'|, \tag{21}$$

where each input dimension k carries its own range γ_k, so that relevance is learned per dimension rather than imposed by a single isotropic scale.

The Matérn-5/2 kernel is preferred over the squared-exponential kernel because traffic flow has finite smoothness rather than the infinite differentiability that the squared-exponential implies, and over the rougher Matérn-3/2 because shocks are absorbed by the LWR layer rather than appearing directly in the volume signal. The mean and variance parameters are treated as nuisance and are analytically marginalized under the conjugate-style prior

$$\pi(\mu_1,\dots,\mu_L,\sigma_1^2,\dots,\sigma_L^2,\boldsymbol{\gamma},\nu) \propto \frac{\pi(\boldsymbol{\gamma},\nu)}{\prod_{\ell=1}^{L}\sigma_\ell^2}, \tag{22}$$

in which $\pi(\boldsymbol{\gamma},\nu)$ is the prior on the kernel parameters, specified in Section 4.4. Under (22) only $\boldsymbol{\gamma}$ and ν remain to be estimated from data; all other $2L$ per-edge parameters are integrated out in closed form.

4.3. Predictive distribution

Once $\boldsymbol{\gamma}$ and ν are fixed, the predictive distribution at a new test input $\lambda^\star$ for edge ℓ is obtained by integrating μ_ℓ and σ_ℓ^2 out of the joint posterior, which yields a Student-t distribution with $n-1$ degrees of freedom:

$$Q_\ell(\lambda^\star) \mid \mathbf{Q}^{\mathcal{D}},\boldsymbol{\gamma},\nu \sim t\big(\hat{Q}_\ell(\lambda^\star),\ \hat{\sigma}_\ell^2\, c^{**},\ n-1\big). \tag{23}$$

The predictive mean, scale, and degree-of-freedom correction admit the closed-form expressions

$$\hat{Q}_\ell(\lambda^\star) = \hat{\mu}_\ell + \boldsymbol{k}^\top(\lambda^\star)\,\widetilde{\mathbf{K}}^{-1}\big(\mathbf{Q}_\ell^{\mathcal{D}} - \hat{\mu}_\ell \mathbf{1}_n\big), \tag{24}$$

$$\hat{\sigma}_\ell^2 = \frac{1}{n-1}\big(\mathbf{Q}_\ell^{\mathcal{D}} - \hat{\mu}_\ell \mathbf{1}_n\big)^\top \widetilde{\mathbf{K}}^{-1}\big(\mathbf{Q}_\ell^{\mathcal{D}} - \hat{\mu}_\ell \mathbf{1}_n\big), \tag{25}$$

$$c^{**} = 1 + \nu - \boldsymbol{k}^\top(\lambda^\star)\,\widetilde{\mathbf{K}}^{-1}\boldsymbol{k}(\lambda^\star) + \Big(\mathbf{1}_n^\top\widetilde{\mathbf{K}}^{-1}\mathbf{1}_n\Big)^{-1}\Big(1 - \mathbf{1}_n^\top\widetilde{\mathbf{K}}^{-1}\boldsymbol{k}(\lambda^\star)\Big)^2, \tag{26}$$

in which $\widetilde{\mathbf{K}} = \mathbf{K} + \nu\mathbf{I}_n$ enforces numerical stability through the nugget, $\boldsymbol{k}(\lambda^\star) = \big(k(\lambda^\star,\lambda_1;\boldsymbol{\gamma}),\dots,k(\lambda^\star,\lambda_n;\boldsymbol{\gamma})\big)^\top$ is the cross-covariance vector between the test point and the training inputs, and the per-edge mean is the generalized least-squares estimate

$$\hat{\mu}_\ell = \Big(\mathbf{1}_n^\top\widetilde{\mathbf{K}}^{-1}\mathbf{1}_n\Big)^{-1}\mathbf{1}_n^\top\widetilde{\mathbf{K}}^{-1}\mathbf{Q}_\ell^{\mathcal{D}}. \tag{27}$$

The closed-form predictive distribution in (23) is one of the practical advantages of the PP-GaSP surrogate over alternative data-driven models. The route planner of Section 5 consumes both the predictive mean $\hat{Q}_\ell$ and the predictive variance $\hat{\sigma}_\ell^2 c^{**}$, the latter of which serves as a per-edge confidence indicator and can be used to inflate the BPR cost on poorly observed edges.

4.4. Parameter estimation via marginal posterior

The kernel parameters $(\boldsymbol{\gamma},\nu)$ are estimated by maximizing the marginal posterior. Direct maximum likelihood is unstable because the correlation matrix $\mathbf{K}$ becomes nearly singular in small- and large-range regimes, behaving in the limit either as the identity matrix or as the all-ones matrix (Rasmussen & Williams, 2006). To mitigate this, the range is reparameterized through the inverse-range vector $\boldsymbol{\beta} = (1/\gamma_1,\dots,1/\gamma_p)$, and the pair $(\boldsymbol{\beta},\nu)$ is estimated via the maximum marginal posterior estimate

$$(\hat{\boldsymbol{\beta}},\hat{\nu}) = \arg\max_{\boldsymbol{\beta},\nu}\ \log\big(p(\mathbf{Q}^{\mathcal{D}} \mid \boldsymbol{\beta},\nu)\,\pi^{JR}(\boldsymbol{\beta},\nu)\big), \tag{28}$$

where the marginal likelihood, obtained after integrating out the mean and variance parameters, has the closed form

$$p(\mathbf{Q}^{\mathcal{D}} \mid \boldsymbol{\beta},\nu) \propto |\widetilde{\mathbf{K}}|^{-L/2}\,\Big|\mathbf{1}_n^\top\widetilde{\mathbf{K}}^{-1}\mathbf{1}_n\Big|^{-L/2}\prod_{\ell=1}^{L}(R_\ell^2)^{-(n-1)/2}, \tag{29}$$

with $R_\ell^2 = (\mathbf{Q}_\ell^{\mathcal{D}})^\top\mathbf{J}\,\mathbf{Q}_\ell^{\mathcal{D}}$, $\mathbf{J} = \widetilde{\mathbf{K}}^{-1}\mathbf{P}$, and $\mathbf{P} = \mathbf{I}_n - \mathbf{1}_n(\mathbf{1}_n^\top\widetilde{\mathbf{K}}^{-1}\mathbf{1}_n)^{-1}\mathbf{1}_n^\top\widetilde{\mathbf{K}}^{-1}$. The jointly robust prior

$$\pi^{JR}(\boldsymbol{\beta},\nu) \propto (C_\beta+\nu)^a\,\exp\big(-bC_\beta - b\nu\big), \qquad C_\beta = \sum_{k=1}^{p} C_k\beta_k, \tag{30}$$

with hyperparameters $a,\ b > 0$, and positive scale constants C_k, is closely related to the reference prior and provides additional regularization in the corners of the parameter space where the likelihood is flat. The optimization is carried out using the low-storage L-BFGS quasi-Newton method, which is well suited to the smooth, almost convex landscape induced by the Matérn-5/2 kernel together with the jointly robust prior.

4.5. Computational complexity and composite likelihood

A naive evaluation of (29) costs $\mathcal{O}(n^3 + n^2L)$ per optimization step, which is dominated by the Cholesky decomposition of $\widetilde{\mathbf{K}}$ and the per-edge quadratic forms. Across q L-BFGS iterations the total complexity is $\mathcal{O}(q(n^3+n^2L))$, which scales linearly in the number of edges L but cubically in the number of training inputs n. For city-scale networks with thousands of edges this cost is acceptable, but for very long historical windows it becomes prohibitive. To scale the surrogate further, the Independent Composite Marginal Likelihood (ICML) approximation is applied:

$$p_r(\mathbf{Q}^{\mathcal{D}} \mid \boldsymbol{\gamma},\nu) = \prod_{b=1}^{r} p\Big(\mathbf{Q}_{(b)}^{\mathcal{D}} \mid \boldsymbol{\gamma},\nu\Big), \tag{31}$$

in which the full marginal likelihood is replaced by a product of per-batch marginal likelihoods over r disjoint subsets of the training inputs. ICML retains the closed-form structure of (29) within each batch and reduces the per-step cost to $\mathcal{O}\big(r(n/r)^3 + r(n/r)^2L\big) = \mathcal{O}(n^3/r^2 + n^2L/r)$ when the r batches are balanced. The accuracy loss has been shown empirically to be small when the batches preserve temporal locality, which makes the ICML approximation a practical workhorse for offline training on full city-scale road graphs.

The resulting surrogate is the central acceleration component of the digital twin. Node-level vehicle counts produced by the YOLOv8 detector and contextual features such as time of day, weather, and day of week are taken as inputs, and edge-level traffic volumes $\{Q_\ell\}_{\ell=1}^{L}$ are returned in milliseconds. These predicted volumes are passed directly to the route planner introduced in the next section, by which congestion-aware routes are computed without ever invoking the LWR simulator at query time. Quantitative evaluation of the surrogate is reported in Section 6.2.

5. Dynamic route planning via ACBFO

Given the surrogate-predicted edge volumes, the digital twin must determine, for a vehicle that requests guidance from origin o to destination d, which sequence of edges $\mathcal{P} = (e_1,\dots,e_K)$ minimizes travel time while also avoiding the secondary bottlenecks that purely greedy services create when many users follow the same recommendation. The proposed ACBFO algorithm addresses this question by augmenting the standard BFO framework with a chaos- and Lévy-driven search strategy and a fitness function tailored to traffic routing.

5.1. Edge cost and fitness with congestion externality

For every directed edge (u,v) of the road graph $G = (V,E)$, the digital twin maintains four quantities. Free-flow time t_{uv}^0, capacity c_{uv}, background flow $\bar{x}_{uv}$, and current flow x_{uv}. The travel time of a single vehicle on edge (u,v) is given by the standard Bureau of Public Roads (BPR) volume-delay function

$$\tau_{uv}(x_{uv}) = t_{uv}^0\left(1 + \alpha\left(\frac{x_{uv}}{c_{uv}}\right)^\beta\right), \tag{32}$$

with the customary parameters $\alpha = 0.15$ and $\beta = 4$. Eq. (32) is a standard link-level volume-delay relationship, consistent in spirit with the congestion effects captured by the LWR model of Section 2.2 although not derived from it: travel time grows mildly with flow up to capacity ($x_{uv} = c_{uv}$, $v/c = 1$) and explodes thereafter. The travel time of a path $\mathcal{P}$ is given by $\tau(\mathcal{P}) = \sum_{(u,v)\in\mathcal{P}} \tau_{uv}(x_{uv})$.

A naive shortest-path solver minimizes only $\tau(\mathcal{P})$. This is the strategy of single-user navigation services and, as is well known in transportation theory (Sheffi, 1985; Wardrop, 1952), drives the system toward a user equilibrium that is generally suboptimal. To make the route planner aware of the externalities imposed on the network, the objective is augmented with a marginal-social-cost penalty:

$$\mathcal{J}(\mathcal{P}) = \tau(\mathcal{P}) + \lambda \sum_{(u,v)\in\mathcal{P}} \left[\max\left(0,\ \frac{x_{uv}+\delta}{c_{uv}} - \tau_{\text{th}}\right)\right]^2, \tag{33}$$

where δ is the per-vehicle flow contribution, $\tau_{\text{th}} = 0.85$ is the saturation threshold, and λ is a scalar weight. The penalty activates only on edges that the new vehicle would push past 85% of capacity, so paths whose edges have ample slack are unaffected. The weight of this separable saturation term is fixed at $\lambda_{\text{sat}} = 6$ throughout, and a sensitivity analysis is reported in Section 6.5.

5.2. From separable per-vehicle costs to batch joint routing

A structural observation motivates the routing formulation adopted in this paper. For a fixed network state, both terms of Eq. (33) are sums of nonnegative costs over the edges of $\mathcal{P}$. The objective is therefore *edge-separable*, and the single-vehicle problem is solved *exactly* by a Dijkstra search over the modified edge weights $c^{\text{MSC}}_{uv} = \tau_{uv}(x_{uv}) + \lambda_{\text{sat}}[\max(0, (x_{uv}+\delta)/c_{uv} - \tau_{\text{th}})]^2$, referred to as MSC-Dijkstra below. Because an exact polynomial-time solver exists, no metaheuristic is justified for the single-vehicle problem, and MSC-Dijkstra is accordingly used as an exact baseline and sanity check.

The herding phenomenon that motivates this work, however, does not arise one vehicle at a time. A routing service faces a *batch* of concurrent requests within each decision window, and the quality of any one recommendation depends on all others issued in the same window. Each request $i \in \{1, \dots, m\}$ is modeled as an origin–destination demand stream contributing δ veh/h along its assigned path $\mathcal{P}_i$, so that the flow induced on edge (u, v) is $x_{uv}(\mathbf{P}) = \bar{x}_{uv} + \delta\,|\{i : (u,v) \in \mathcal{P}_i\}|$ for the joint assignment $\mathbf{P} = (\mathcal{P}_1, \dots, \mathcal{P}_m)$. The batch objective is

$$\mathcal{F}(\mathbf{P}) = \sum_{i=1}^{m} \sum_{(u,v)\in\mathcal{P}_i} \tau_{uv}\big(x_{uv}(\mathbf{P})\big) \;+\; \lambda_{\max} \max_{(u,v)\in E(\mathbf{P})} \frac{x_{uv}(\mathbf{P})}{c_{uv}} + \lambda_{\text{sat}} \sum_{(u,v)\in E(\mathbf{P})} \left[\max\left(0,\ \frac{x_{uv}(\mathbf{P})}{c_{uv}} - \tau_{\text{th}}\right)\right]^2, \tag{34}$$

where $E(\mathbf{P})$ denotes the set of edges used by the fleet and $\lambda_{\max} = 100$, $\lambda_{\text{sat}} = 6$ unless stated otherwise. Two properties make Eq. (34) fundamentally harder than Eq. (33). The travel-time term couples all streams through the induced flows, and the network-bottleneck term $\max_{(u,v)} x_{uv}/c_{uv}$ is not decomposable over edges or vehicles. As a consequence, neither a single shortest-path computation nor per-vehicle best-response iteration can optimize Eq. (34). Best-response dynamics converge to a local equilibrium of the separable part while leaving the bottleneck term unoptimized, and the population-based joint search is designed to close this gap.

Finally, the predictive uncertainty exposed by the surrogate enters the objective through a risk-inflated flow estimate,

$$\widetilde{x}_{uv} = \hat{x}_{uv} + \kappa\, s_{uv}, \tag{35}$$

where $\hat{x}_{uv}$ and s_{uv} are the Student-t predictive mean and scale of the surrogate and $\kappa \geq 0$ controls risk aversion, with $\kappa = 0$ recovering mean-only routing. Section 6.6 shows that moderate κ improves realized routing performance when the surrogate is queried outside its training distribution.

5.3. Bacterial foraging optimization

BFO (Passino, 2002) mimics the foraging behavior of *Escherichia coli* bacteria and consists of chemotaxis, swarming, reproduction, and elimination-dispersal. In the chemotaxis stage, each bacterium tumbles and swims in response to the chemical concentration gradient of the search space. The location of the ith bacterium at the jth chemotaxis step, the kth reproduction step, and the lth elimination-dispersal step is updated by

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C_i\,\phi^i(j), \quad \phi^i(j) = \frac{\Delta^i}{\sqrt{(\Delta^i)^{\top}\Delta^i}}, \tag{36}$$

where C_i is the swim step length and Δ^i is a random unit vector that defines the tumbling direction. In the swarming stage, bacteria release attractant and repellent chemical signals that allow them to cluster around promising regions while maintaining individual diversity. In the reproduction stage, the population is sorted by health value $J^i_{\text{health}} = \sum_{j=1}^{N_c} J(i,j,k,l)$, and the worse half is replaced by copies of the better half so that the population size is preserved. In the elimination-dispersal stage, each bacterium is replaced by a randomly generated new individual with a fixed probability P_{ed}, by which the population diversity is restored and local optima are escaped.

A weakness of the standard formulation is that the constant elimination probability P_{ed} assigns the same dispersal chance to the best and the worst bacteria, which may eliminate near-optimal individuals and slow convergence (Sahib et al., 2018). Several variants have therefore replaced the constant probability with a non-uniform distribution that retains the elite. The ACBFO described next inherits this insight and goes further by introducing chaos and Lévy-flight perturbations into both the initialization and the chemotaxis stages.

5.4. The ACBFO algorithm

The proposed ACBFO retains the four-stage structure of standard BFO (chemotaxis, swarming, reproduction, elimination-dispersal) and operates on *joint* solutions. Each bacterium encodes the complete batch assignment $\mathbf{P} = (\mathcal{P}_1, \dots, \mathcal{P}_m)$, one loop-free node path per demand stream. All algorithms compared in Section 6 share the same path encoding and the same two generic variation operators, so that observed differences reflect only the search strategy. Segment-Mutate regrows a randomly selected subpath with a goal-biased random walk, and Crossover exchanges the suffix of a path with that of a reference solution at a common node. ACBFO introduces three mechanisms on top of this shared substrate, of which the congestion-guided chemotaxis of mechanism (i) is the primary algorithmic contribution; the ablation study attributes the performance of ACBFO to it in both the warm-started and the cold-started regime.

(i) Congestion-guided chemotaxis. The central mechanism makes the search operators, and not only the fitness, aware of congestion externalities. At a chemotaxis step, the operator identifies the current fleet bottleneck edge $e^\star = \arg\max_{(u,v)\in E(\mathbf{P})} x_{uv}(\mathbf{P})/c_{uv}$, selects at random one stream whose path traverses $e^\star$, and reroutes that stream by a Dijkstra search over the myopic MSC weights with $e^\star$ excluded. This targeted move attacks the non-separable bottleneck term of Eq. (34) directly, a term that per-vehicle best-response methods cannot optimize, at the cost of one shortest-path computation of $\mathcal{O}(|E|\log|V|)$ per application. The wall-clock overhead of the operator is accounted for explicitly in the compute-matched comparison.

(ii) Chaotic scheduling with a probability-integral transform. The logistic map $z_{k+1} = \mu z_k(1-z_k)$ with $\mu = 4$ is ergodic on $(0,1)$, but its invariant density is the arcsine law $\pi^{-1}[z(1-z)]^{-1/2}$, which concentrates mass near 0 and 1 rather than covering the unit interval uniformly. The chaotic driver is therefore passed through the probability-integral transform of the arcsine distribution,

$$u_k = \frac{2}{\pi}\arcsin\!\left(\sqrt{z_k}\right), \tag{37}$$

Algorithm 1: ACBFO for batch joint route planning

Input: Road graph G, demand streams $\{(o_i, d_i)\}_{i=1}^{m}$, population S, evaluation budget B, weights $(\lambda_{\max}, \lambda_{\text{sat}})$, threshold τ_{th}, stream rate δ, risk factor κ, swim cap $N_s^{\max}$, optional warm-start assignment $\mathbf{P}^{w}$.
Output: Best batch assignment $\mathbf{P}^{\star}$.
Initialize population $\{\mathbf{P}_j\}_{j=1}^{S}$ with the chaotic path builder driven by Eq. (37); if provided, insert $\mathbf{P}^{w}$ as one individual;
$\mathbf{P}^{\star} \leftarrow \arg\min_j \mathcal{F}(\mathbf{P}_j)$;
while *evaluations* $< B$ **do**
  $\rho \leftarrow$ consumed budget fraction;
  foreach *bacterium* $\mathbf{P}_j$ **do**
    draw u from Eq. (37);
    if $u < 0.40$ **then**
      $\mathbf{P}' \leftarrow$ Segment-Mutate($\mathbf{P}_j$);
    else if $u < 0.70$ **then**
      $\mathbf{P}' \leftarrow$ congestion-guided reroute of a stream off the fleet bottleneck edge of $\mathbf{P}_j$;
    else
      $\mathbf{P}' \leftarrow$ Crossover($\mathbf{P}_j, \mathbf{P}^{\star}$);
    end
    if $\mathcal{F}(\mathbf{P}') < \mathcal{F}(\mathbf{P}_j)$ **then**
      $\mathbf{P}_j \leftarrow \mathbf{P}'$; swim with the same operator while improving, up to $\lceil N_s^{\max}(1 - 0.7\rho)\rceil$ steps;
    end
    update $\mathbf{P}^{\star}$;
  end
  every N_c sweeps, sort by $\mathcal{F}$ and clone the better half over the worse half (reproduction);
  every $N_c N_{re}$ sweeps, draw Lévy variates and disperse with the clipped probabilities of Eq. (38), keeping $\mathbf{P}^{\star}$;
end
return $\mathbf{P}^{\star}$;

which is exactly uniform on $(0, 1)$ while remaining a deterministic chaotic sequence. The transformed sequence serves two roles. During initialization, successive u_k set the goal-bias probability of the randomized path builder, yielding a reproducible spread from near-uniform random walks to strongly goal-directed paths. During chemotaxis, u_k schedules the operator applied to each bacterium, with Segment-Mutate for $u_k < 0.40$ (exploration), congestion-guided rerouting for $0.40 \le u_k < 0.70$, and Crossover with the best-so-far solution for $u_k \ge 0.70$ (exploitation). Improving moves trigger up to $N_s = \lceil N_s^{\max}(1 - 0.7\rho)\rceil$ additional swims of the same operator, where $\rho \in [0, 1]$ is the fraction of the evaluation budget consumed, so that early iterations explore aggressively and later iterations refine.

(iii) Clipped Lévy elimination-dispersal. In the elimination-dispersal stage, the rank-based non-uniform dispersal probability of Sahib et al. (2018) is perturbed by a Lévy variate ℓ_i with index $\beta_{\text{Lévy}} = 1.5$,

$$P_{ed}^{i} \leftarrow \text{clip}\big(P_{ed}^{i} + 0.3\,\ell_i - 0.15,\ 0,\ 1\big), \tag{38}$$

where the explicit clipping guarantees a valid probability, since large Lévy excursions would otherwise drive P_{ed}^{i} outside $[0, 1]$. Dispersed individuals are re-initialized by the chaotic path builder when the flight is long ($\ell_i > 0.6$), in which case a random subset of streams is re-seeded, and by a perturbed copy of the incumbent best otherwise; the incumbent itself is always retained. The complete procedure is summarized in Algorithm 1.

For completeness, the operator internals are specified as follows. The goal-biased path builder steps toward the destination (by Euclidean distance) with probability p and to a uniformly random unvisited neighbor otherwise; if it dead-ends it restarts, and after 15 failed attempts it falls back to the free-flow shortest path. Segment-Mutate draws two cut positions uniformly at random and regrows the intermediate segment with the builder using $p \sim \mathcal{U}(0.5, 0.95)$. Crossover selects a uniformly random interior node common to both parents and exchanges suffixes, falling back to Segment-Mutate when no such node exists. Every offspring passes a loop-removal repair that keeps the first occurrence of each node, so all encoded paths remain simple origin–destination paths by construction. All algorithmic settings are collected in Table 1.

Table 1
Implementation parameters of ACBFO and the shared metaheuristic framework.

Symbol	Meaning	Value	Symbol/Value
S	Population size	30	$N_s^{\max} = 4$ (swim cap)
B	Evaluation budget	8000	$N_c = \lfloor B/(8S)\rfloor$ sweeps
P_{ed}	Base dispersal prob.	0.25	Dispersal every $2N_c$ sweeps
$\beta_{\text{Lévy}}$	Lévy index	1.5	Schedule thresholds 0.40/0.70
$\lambda_{\max}$	Bottleneck weight	100	Goal bias clipped to $[0.05, 0.95]$
λ_{sat}	Saturation weight	6	Rank dispersal $2P_{ed}$ rank$/(S-1)$
τ_{th}	Saturation threshold	0.85	$\delta = 60$ veh/h, $m = 40$

6. Simulation and comparison studies

In this section, the proposed digital twin framework is evaluated across its three functional layers: vehicle detection and counting at intersections, traffic-flow modeling and network-scale surrogate prediction, and congestion-aware route planning. Each layer is evaluated on data and benchmarks specific to its task, and the experimental settings particular to each layer are introduced together with the corresponding results.

6.1. Vehicle detection and counting results

The perceptual front-end of the digital twin is evaluated on the Traffic Intersection Vehicle Detection dataset (Roboflow, 2023), which is selected as the primary benchmark because its images are captured from intersection cameras at the same vantage point and field of view that the deployed digital twin would observe. The dataset comprises 4526 images at a resolution of 640 × 640, partitioned into 3,162 training images, 902 validation images, and 462 test images. Five vehicle-relevant classes are annotated: car, bus, truck, motorbike, and person, which directly support the type-aware counting strategy. Images from the dataset have weather variation, multiple lanes, and partial occlusion. The detector is trained for 50 epochs with batch size 32, an initial learning rate of 10^{-4}, a final learning-rate fraction of 0.1 relative to the initial rate, a dropout rate of 0.1, and the best-validation weights are retained for evaluation.

After training, the detector reaches a precision of 0.853, a recall of 0.918, an mAP_{50} of 0.935, and an $\text{mAP}_{50\text{–}95}$ of 0.685 on the validation split, with overall fitness 0.709. The training and validation curves are reported in Fig. 3, where the rapid drop of the box, classification, and DFL losses together with the steady rise of precision, recall, and mAP indicate that the detector has converged without overfitting. On the held-out test split the detector generalizes well, with precision 0.899, recall 0.871, mAP_{50} of 0.932, and $\text{mAP}_{50\text{–}95}$ of 0.699. These results establish the frame-level detection performance of the perception front-end; event-level counting reliability is a separate quantity and is treated explicitly below.

The per-class behavior of the detector is examined in Fig. 4. The precision–recall curves of Fig. 4(a) show that the dominant vehicle classes, including car ($\text{AP}_{50} = 0.987$), truck (0.985), and bus (0.984), are detected at near-perfect accuracy, whereas motorbike (0.903) and person (0.815) are noticeably harder. The corresponding F1-confidence curves of Fig. 4(b) reach an aggregate maximum F1 of 0.89 at confidence 0.368, which is therefore adopted as the operating threshold τ_p in Eq. (16). The normalized confusion matrix of Fig. 4(c) corroborates these findings: bus, car, and truck are correctly classified 98%, 98%, and 99% of the time, respectively; the off-diagonal mass

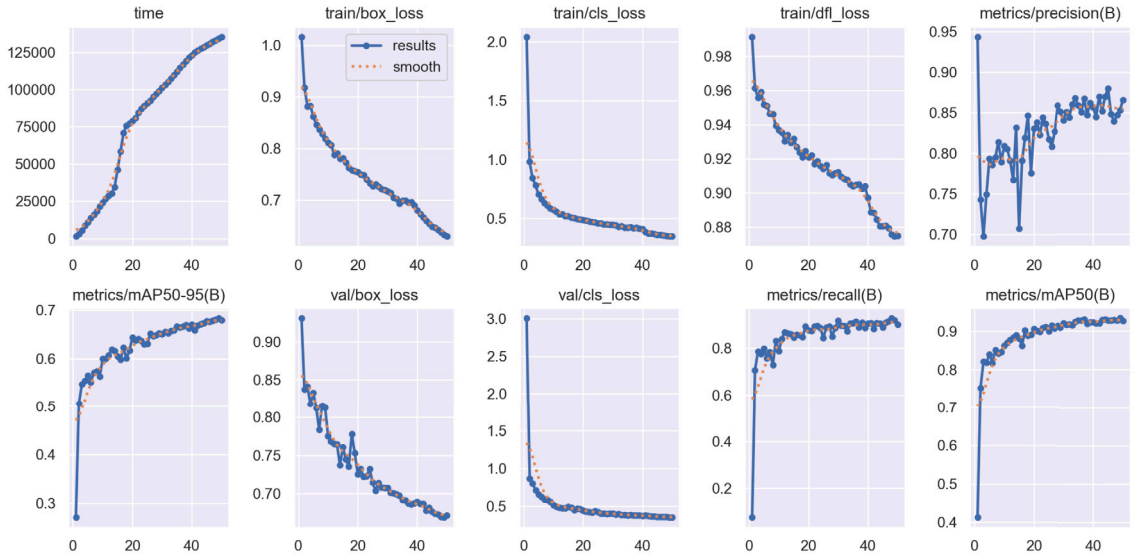


Fig. 3. YOLOv8 training and validation metrics on the Traffic Intersection Vehicle Detection dataset. The losses (box, classification, DFL) decay smoothly while precision, recall, and the two mAP metrics rise to their plateau values, indicating convergence without overfitting.

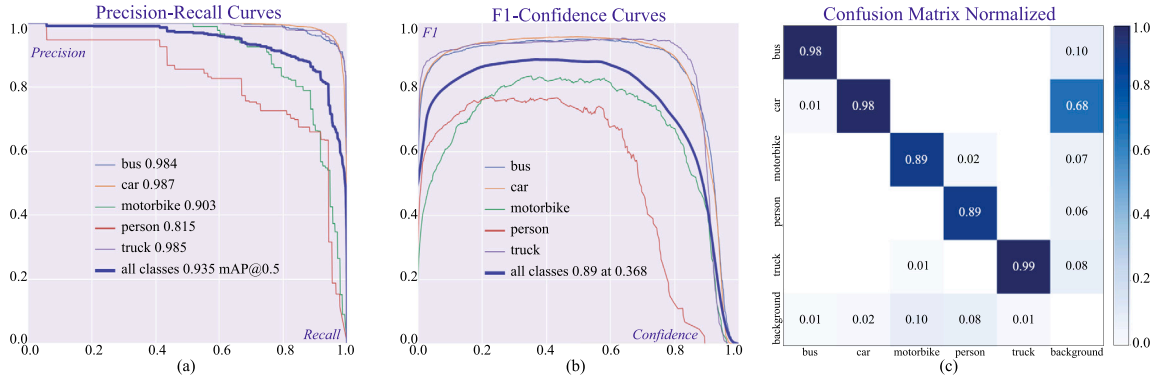


Fig. 4. Per-class diagnostic curves of the YOLOv8 detector on the Traffic Intersection Vehicle Detection dataset. (a) Precision–recall curve. The aggregate mAP_{50} is 0.935, with car, truck, and bus all above 0.98. (b) F1-confidence curve. The aggregate F1 peaks at 0.89 at confidence 0.368, which is adopted as the operating threshold. (c) Normalized confusion matrix. Bus, car, and truck are recovered with ≥ 0.98 accuracy, while a fraction of small cars is absorbed into the background class. The effect of this residual error on aggregated counts is discussed in the text.

is dominated by missed detections of small or occluded objects that fall back to the background class. Although a sizeable fraction of small cars is occasionally absorbed into the background, frame-level confusion alone does not determine the resulting boundary-count error; its effect on event-level counting is quantified in the controlled continuous-sequence benchmark reported below.

A qualitative impression of the detector under deployment conditions is provided by Fig. 5, which shows nine test-set frames captured under varied weather and illumination. In dry conditions, vehicles of all five classes are enclosed by tight bounding boxes with confidences typically above 0.85, including small cars in the far field. In wet conditions the road surface produces specular reflections that cause occasional near-duplicate detections. The qualitative behavior observed across these frames matches the quantitative frame-level metrics reported above.

Event-level validation of association and line crossing. The two image datasets above do not contain continuous vehicle trajectories, so frame-level mAP cannot by itself validate the short-horizon association and virtual-line-crossing stages. To evaluate those stages quantitatively without conflating them with detector retraining, we constructed a controlled, continuously annotated traffic-sequence benchmark. The benchmark contains 40 one-minute sequences at 960 × 540 resolution and 10 analyzed frames per second, with 10 independently seeded sequences in each of four conditions: moderate unidirectional traffic, dense unidirectional traffic, bidirectional traffic, and mixed-speed bidirectional traffic. Cars, trucks, and buses move on fixed lanes with speeds spanning 35 to 95 pixels/s, producing 2419 known ground-truth line crossings in total.

Frame-level boxes are generated from the known trajectories and then perturbed to reproduce the held-out detector operating point reported above. Each true box is retained with probability 0.871, the false-positive rate is selected so that the expected frame-level precision is 0.899, centroid coordinates receive zero-mean Gaussian noise with standard deviation 2.5 pixels, and box dimensions receive 3% relative Gaussian jitter. The association and crossing parameters specified in Section 3 are fixed before the sweep and are not tuned separately for any traffic condition. A predicted crossing is matched to a ground-truth event when it occurs within six analyzed frames and within 55 pixels in the image x coordinate. This protocol evaluates the counting logic under controlled detector-like errors while keeping exact event ground truth.

Table 2 reports the resulting counting metrics. Across all 40 sequences, the predicted-to-ground-truth count ratio is 96.69%, the



Fig. 5. Qualitative test-set inferences of the YOLOv8 detector on the Traffic Intersection Vehicle Detection dataset. Dry and wet conditions, multiple vehicle classes, and partial occlusion are handled with consistent type assignment and tight localization.

Table 2
Event-level validation of the short-horizon association and virtual-line-crossing stages on 40 controlled continuously annotated traffic sequences. All rates are percentages.

Condition	GT crossings	VCA	APE	Missed	Duplicate	Direction acc.	Event F1
Moderate	443	97.97	2.03	2.03	0.00	100.00	98.97
Dense	767	95.96	4.04	4.17	0.13	100.00	97.80
Bidirectional	622	96.14	3.86	3.86	0.00	100.00	98.03
Mixed-speed	587	97.27	2.73	2.90	0.17	100.00	98.45
Overall	2419	96.69	3.31	3.39	0.08	100.00	98.23

aggregate APE is 3.31%, the missed-count rate is 3.39%, and the duplicate-count rate is 0.08%. All matched events have the correct entry/exit direction, and the overall event F1 is 98.23%. Dense traffic is the most difficult condition, with a 4.04% aggregate APE and 97.80% event F1, while performance remains similar under bidirectional and mixed-speed traffic. The mean per-sequence APE across all 40 runs is $3.16 \pm 2.06\%$. The benchmark provides a direct quantitative evaluation of the short-horizon association and virtual-line-crossing stages. Because the sequences are controlled rather than field recordings, the results characterize the counting logic under detector-matched perturbations; a physical camera-in-the-loop deployment is planned to extend this evaluation to field conditions.

To verify that these results are not specific to a single dataset, the YOLOv8 detector is additionally retrained and evaluated on the Top-View Vehicle Detection Image Dataset (Adnekouei, 2024a, 2024b), which contains 536 training images and 90 validation images of single-class vehicles seen from a top-down camera. Under the same training hyperparameters, the detector reaches precision 0.918, recall 0.925, mAP_{50} of 0.974, $mAP_{50–95}$ of 0.730, and fitness 0.755. The metrics on this complementary dataset are even higher than on the intersection dataset, as expected from the simpler single-class setup, and confirm that the perception pipeline transfers gracefully across viewpoints. The complementary dataset therefore supports the transferability of the frame-level detector across viewpoints. Together with the controlled continuous-sequence benchmark in Table 2, the evaluation separates detector generalization from the event-level accuracy of the association and line-crossing stages.

6.2. Traffic flow modeling and PP-GaSP surrogate prediction

The traffic-flow layer of the digital twin is evaluated in two stages. The LWR simulator is first exercised under representative weather and weekday/weekend scenarios to confirm that the parameterized free-flow speed of Section 2.2 captures the qualitative behavior of urban

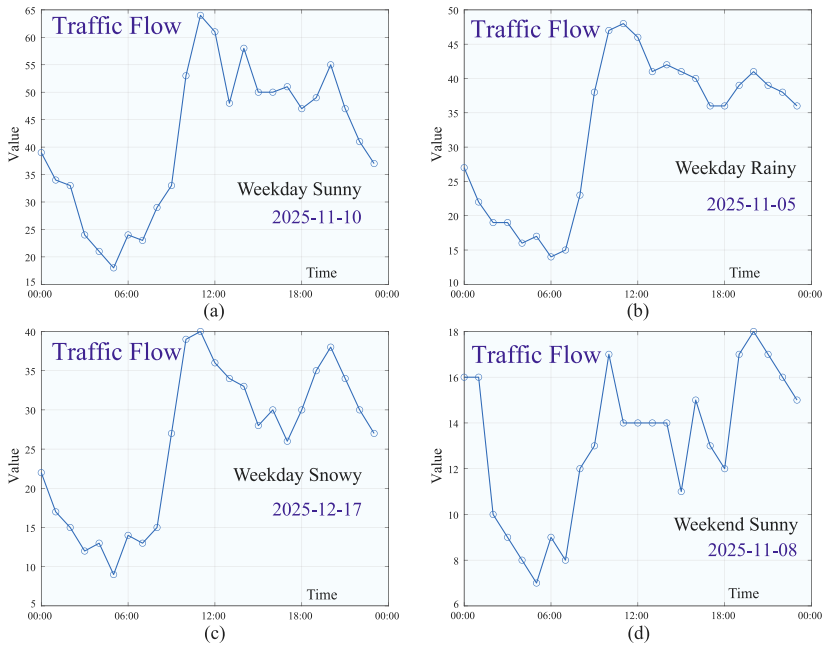


Fig. 6. LWR-based traffic-flow simulation under four representative scenarios. (a) Weekday sunny, (b) Weekday rainy. (c) Weekday snowy. (d) Weekend sunny. The vertical values are traffic flow in veh/h, and the horizontal axis is time of day.

traffic. The PP-GaSP surrogate developed in Section 4 is then trained on the simulator outputs and evaluated against the simulator on accuracy, computational efficiency, and uncertainty calibration.

The node-level counts produced by the YOLOv8 detector are propagated across the network by the LWR simulator described in Section 2.2, with the signal, lane, and weather factors configured from publicly available road characteristics. The simulated traffic-flow patterns under four representative conditions, including weekday sunny, weekday rainy, weekday snowy, and weekend sunny, are illustrated in Fig. 6. On weekdays, two prominent peaks appear: a morning buildup that begins around 7 AM and peaks near 11 AM and an evening rush that spans 6 to 8 PM, separated by an afternoon trough. Weekends exhibit overall lower volume and less pronounced peaks. The weather sweep further confirms that rain reduces flow relative to sunny conditions, and that snow reduces it even further, which is consistent with field observations and indicates that the digital twin captures the qualitative behavior of real urban traffic.

The PP-GaSP surrogate is trained on a subset of simulator outputs sampled across the full input space of λ, which comprises a periodic encoding of time of day, a weekend indicator, one-hot weather indicators, and the aggregate boundary inflow count supplied by the perception layer, so that the measured counts enter the prediction directly, and queried at the held-out time stamps of the weekday sunny scenario. Fig. 7 compares the PP-GaSP-predicted traffic flow with the LWR simulator output. The fitted curve tracks the simulated data accurately across the day, including both peak and trough regions and the sharp transitions between them, which confirms that the surrogate closely reproduces the simulator over the evaluated operating range. The Matérn-5/2 kernel chosen in Section 4.2 is responsible for this behavior. Its finite differentiability allows the GP to follow the abrupt rises and falls of rush-hour transitions without introducing the oscillations that a smoother squared-exponential kernel would produce around discontinuities.

Three additional properties of the PP-GaSP surrogate are critical for closed-loop deployment. *Sample efficiency:* the predictions in Fig. 7 are obtained from only a small subset of the simulator outputs, owing to the non-parametric nature of the GP. This matters because the simulator itself is the expensive upstream component, and matching its accuracy

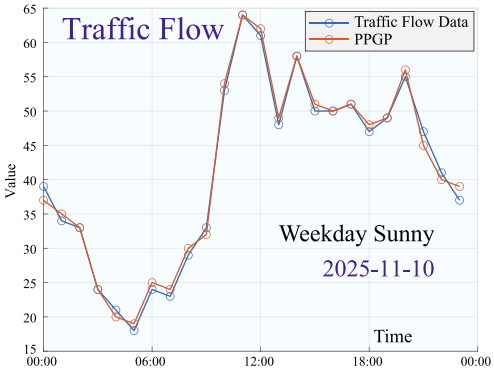


Fig. 7. PP-GaSP surrogate versus LWR simulation on the weekday sunny scenario. The surrogate matches the simulator across the full day while delivering predictions orders of magnitude faster. Traffic flow is reported in veh/h against time of day.

from few queries directly translates into faster offline training. *Inference latency:* once trained, the surrogate produces network-scale predictions in milliseconds via the closed-form predictive mean of (24). By contrast, direct evaluation of the LWR simulator over all edges and time steps of a city-scale graph is too slow to support online routing decisions. The combination of the analytic marginalization of the $2L$ mean and variance parameters in Section 4.1, the closed-form Student-t predictive distribution of (23), and the ICML approximation of (31) keeps both training and inference tractable on full city-scale road graphs. *Predictive uncertainty:* the Student-t predictive distribution returns a per-edge variance $\hat{\sigma}^2_{\ell} c^{**}$ at no additional cost, and the surrogate widens its predicted intervals on edges whose recent inputs lie outside the training range. This predictive uncertainty is exploited by the ACBFO route planner in the next section, which inflates the BPR cost on poorly observed edges so that the proposed routes do not silently rely on low-confidence predictions. Together, these three properties demonstrate that the PP-GaSP surrogate satisfies the design requirements stated at the beginning of Section 4: it closely reproduces the LWR simulator

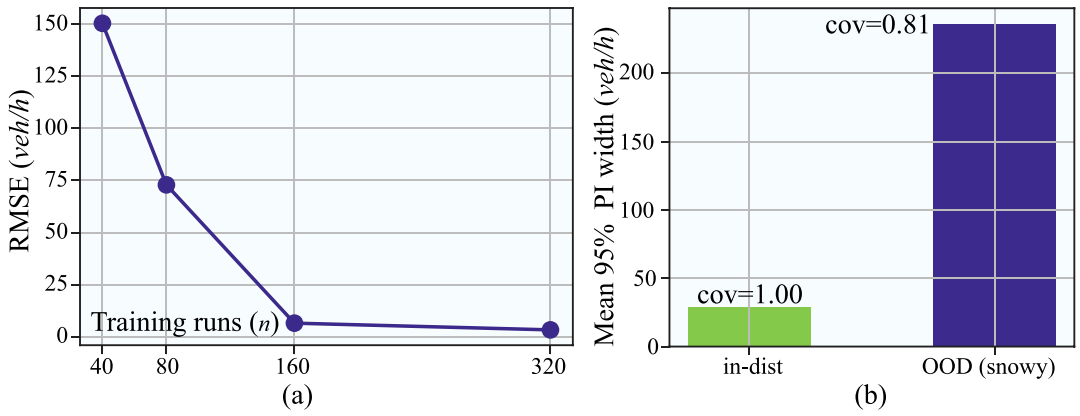


Fig. 8. Quantitative surrogate validation. (a) Learning curve of the PP-GaSP on held-out simulator runs. (b) Mean 95% prediction interval width in distribution versus out of distribution (snowy regime withheld from training). Intervals widen by 8.3× and retain 81% coverage, signaling reduced confidence exactly where the training data are uninformative.

over the evaluated operating range, delivers network-scale predictions at millisecond latency, and exposes per-edge confidence to the downstream route planner. This combination of simulator agreement, speed, and predictive uncertainty is what makes the PP-GaSP surrogate the practical workhorse of the closed-loop digital twin.

Beyond this qualitative agreement, the surrogate is validated quantitatively on held-out simulator runs sampled across the full input space (240 training and 60 test runs, over all edges of the network). The PP-GaSP attains $R^2 > 0.999$ and the lowest errors among all compared predictors (MAE of 1.0 veh/h versus 6.6 for k-NN and 10.0 for an MLP; RMSE of 4.1 versus 24.6 and 26.9 veh/h). The distinguishing property of the PP-GaSP is its closed-form predictive uncertainty obtained at no additional cost. The 95% Student-t prediction intervals achieve 99.8% empirical coverage in distribution with a mean width of 19.9 veh/h (mildly conservative relative to the nominal 95%) and a continuous ranked probability score of 1.4 veh/h, none of which the point-forecast baselines provide. A single network-wide prediction takes 0.02 ms, versus 4.9 ms per simulator query in our implementation, a 200× speed-up that widens further as simulator fidelity grows. The learning curve in Fig. 8(a) confirms the sample efficiency of the nonparametric formulation.

Two further experiments address the relation between the surrogate and the conditions it will face in deployment. In an out-of-distribution (OOD) holdout, the surrogate is trained with the snowy regime withheld entirely and then queried on snowy conditions. RMSE degrades from 10.4 to 283 veh/h, but the prediction intervals widen automatically by a factor of 8.3 (mean width 28.6 to 236 veh/h, Fig. 8(b)) while retaining 81% coverage, so the surrogate signals its reduced confidence exactly where the training data are uninformative. This widening is the signal consumed by the risk-inflated routing of Eq. (35) and evaluated in Section 6.6. As a limitation, surrogate accuracy remains upper-bounded by the fidelity of the simulator it emulates, and residual calibration against field measurements is an important direction for future work.

6.3. Route planning benchmarks and single-vehicle results

6.3.1. Experiment setup

The test environment is a 10 × 10 road network with stochastic geometry. Nodes on a unit grid are perturbed by Gaussian noise with standard deviation 0.15, all four-connected directed edges are added with Euclidean lengths, free-flow speeds and capacities are drawn i.i.d. from $\mathcal{U}(40, 60)$ km/h and $\mathcal{U}(800, 1500)$ veh/h, and background congestion is bimodal (a fraction 0.30 of edges carries $[0.95, 1.45]\times 0.85\times c$ and the remainder $[0.10, 0.60] \times 0.85 \times c$). Reflecting the rush-hour setting in which herding arises, each instance routes $m = 40$ concurrent demand streams of $\delta = 60$ veh/h sampled from a residential block to a central-business-district block on opposite corners of the network.

The proposed ACBFO is benchmarked against four exact or greedy baselines and six metaheuristics. The four baselines are sequential free-flow Dijkstra, which represents consumer navigation without congestion awareness, sequential congested Dijkstra with BPR-adjusted times, sequential MSC-Dijkstra, which routes each stream exactly against the separable objective of Eq. (33) given all previously routed streams, and iterated best response (IBR), which repeatedly re-optimizes each stream's myopic MSC cost given all others until no stream changes, providing a strong per-stream best-response baseline. The metaheuristic baselines are PSO (Kennedy & Eberhart, 1995), GWO (Mirjalili et al., 2014), WOA (Mirjalili & Lewis, 2016), AOA (Abualigah et al., 2021), standard BFO (Passino, 2002), and IBFO (Sahib et al., 2018). All metaheuristics share the same joint-path encoding, variation operators, population size $S = 30$, and evaluation budget of 8,000 objective evaluations, and all receive the same IBR solution injected into their initial population, so that every method starts from the best per-vehicle solution and the comparison isolates the ability of each search strategy to improve beyond the per-vehicle optimum.

Every configuration is run on 30 paired instances with identical networks, demand sets, and warm starts across algorithms. We report mean ± standard deviation together with 95% Student-t confidence intervals for the across-instance mean, compare the algorithm family with a Friedman test, assess pairwise differences against ACBFO with two-sided Wilcoxon signed-rank tests under Holm-Bonferroni correction, and quantify effect sizes with Cliff's δ.

6.3.2. Single-vehicle results and cold-start convergence

Because the single-vehicle objective is edge-separable (Section 5.2), MSC-Dijkstra provides the provably optimal solution and thus the natural yardstick. On 30 instances, the proposed ACBFO attains this optimum on every instance, with mean and maximum optimality gaps of 0.000%, driven by its congestion-guided rerouting operator. The remaining metaheuristics land within 0.5% to 1.5% on average at the same budget, while free-flow and congested Dijkstra incur mean gaps of 8.3% and 2.8%. The single-vehicle problem is therefore reported as a sanity check rather than as evidence of metaheuristic superiority, and the batch problem of Section 6.4 is where joint search becomes necessary.

Cold-start convergence on the batch problem without any warm start is shown in Fig. 9 for 15 instances. ACBFO reaches a batch objective of 784 within 2000 evaluations, which is better than the value any baseline attains with the full 8,000-evaluation budget, and converges

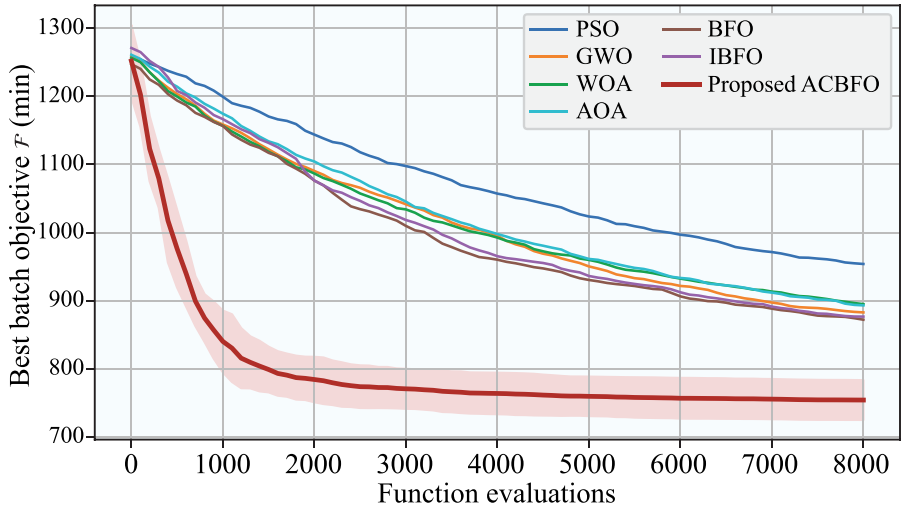


Fig. 9. Cold-start convergence on the batch joint-routing problem (mean over 15 paired instances). ACBFO (red) surpasses within 2000 evaluations the quality that every baseline reaches only with the full 8000-evaluation budget.

Table 3
Batch joint routing of $m = 40$ demand streams on 30 paired instances (mean ± std). p-values are two-sided Wilcoxon signed-rank versus ACBFO on $\mathcal{F}$ with Holm correction, and δ denotes Cliff's delta. Best values in bold.

Algorithm	Mean stream TT (min)	Fleet max v/c	$\mathcal{F}$ (min)	p_{Holm}	δ
Seq. Dijkstra (free-flow)	37.57 ± 16.07	2.773 ± 0.482	1888.3 ± 721.1	$< 10^{-3}$	1.00
Seq. Dijkstra (congested)	15.54 ± 0.61	1.561 ± 0.121	803.4 ± 29.1	$< 10^{-3}$	0.85
Seq. MSC-Dijkstra (exact)	15.39 ± 0.63	1.293 ± 0.039	756.9 ± 27.2	$< 10^{-3}$	0.26
Iterated Best Response	15.30 ± 0.64	1.282 ± 0.036	751.1 ± 27.3	$< 10^{-3}$	0.14
FQI (learning-based)	16.40 ± 0.66	1.481 ± 0.069	835.6 ± 32.1	$< 10^{-3}$	0.99
PSO	15.27 ± 0.64	1.277 ± 0.038	749.7 ± 27.3	$< 10^{-3}$	0.11
GWO	15.25 ± 0.63	1.273 ± 0.037	748.9 ± 27.4	$< 10^{-3}$	0.10
WOA	15.26 ± 0.63	1.273 ± 0.039	749.2 ± 27.2	$< 10^{-3}$	0.11
AOA	15.26 ± 0.63	1.276 ± 0.038	749.6 ± 27.0	$< 10^{-3}$	0.12
BFO	**15.24 ± 0.61**	1.260 ± 0.034	747.2 ± 27.1	0.001	0.03
IBFO	15.23 ± 0.62	1.264 ± 0.036	747.6 ± 26.8	$< 10^{-3}$	0.06
ACBFO (proposed)	15.30 ± 0.63	**1.229 ± 0.039**	**746.0 ± 26.9**	–	–

Table 4
95% Student-t confidence intervals for the across-instance means in Table 3 ($n = 30$). These intervals are computed as $\bar{x} \pm t_{0.975,29}\, s/\sqrt{30}$ from the reported paired-instance summaries.

Algorithm	Mean stream TT (min)	Fleet max v/c	$\mathcal{F}$ (min)
Seq. Dijkstra (free-flow)	[31.57, 43.57]	[2.593, 2.953]	[1619.0, 2157.6]
Seq. Dijkstra (congested)	[15.31, 15.77]	[1.516, 1.606]	[792.5, 814.3]
Seq. MSC-Dijkstra (exact)	[15.15, 15.63]	[1.278, 1.308]	[746.7, 767.1]
Iterated Best Response	[15.06, 15.54]	[1.269, 1.295]	[740.9, 761.3]
FQI (learning-based)	[16.15, 16.65]	[1.455, 1.507]	[823.6, 847.6]
PSO	[15.03, 15.51]	[1.263, 1.291]	[739.5, 759.9]
GWO	[15.01, 15.49]	[1.259, 1.287]	[738.7, 759.1]
WOA	[15.02, 15.50]	[1.258, 1.288]	[739.0, 759.4]
AOA	[15.02, 15.50]	[1.262, 1.290]	[739.5, 759.7]
BFO	[15.01, 15.47]	[1.247, 1.273]	[737.1, 757.3]
IBFO	[15.00, 15.46]	[1.251, 1.277]	[737.6, 757.6]
ACBFO (proposed)	[15.06, 15.54]	[1.214, 1.244]	[736.0, 756.0]

to 754, within 1% of its warm-started result. ACBFO therefore holds a substantial advantage under tight real-time budgets, which is the operationally relevant regime for a routing service that must answer within a fixed decision window.

6.4. Batch multi-stream routing and bottleneck mitigation

Table 3 summarizes the joint routing of $m = 40$ concurrent streams on 30 paired instances. Table 4 reports the corresponding 95% confidence intervals for these results. First, purely reactive free-flow routing is catastrophic under concentrated demand, with a mean stream travel time of 37.6 min and a fleet worst-edge v/c of 2.77, versus 15.2 to 15.4 min and 1.23 to 1.29 for all congestion-aware methods, which quantifies directly the herding failure that motivates this work. Second, all population-based joint methods significantly improve on both sequential exact routing and IBR (Friedman $\chi^2 = 138.0$, $p = 2.7 \times 10^{-27}$, with all pairwise Holm-corrected $p \le 10^{-3}$), confirming that the batch objective rewards joint optimization beyond per-vehicle exact method. Third, ACBFO attains the lowest batch objective (746.0 ± 26.9; the 0.2% margin over BFO is modest in size but consistent across instances, $p = 0.001$) and, more consequentially, the lowest fleet worst-edge v/c ratio (1.229 ± 0.039 versus 1.260 for BFO with Cliff's $\delta = 0.43$ and $p < 10^{-4}$, and versus 1.293 for sequential MSC-Dijkstra, a 5.0% reduction).

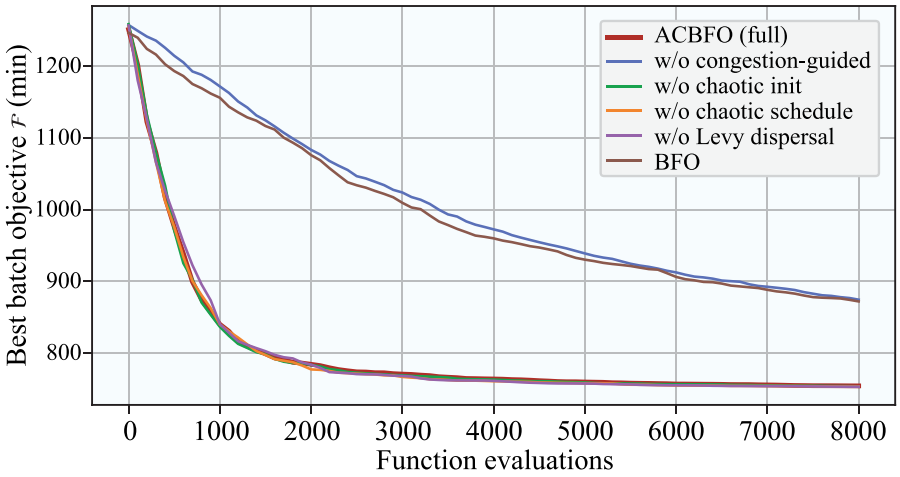


Fig. 10. Cold-start mechanism ablation (mean over 15 paired instances). Removing the congestion-guided operator collapses ACBFO onto standard BFO, while removing any chaotic or Lévy component leaves the trajectory statistically unchanged.

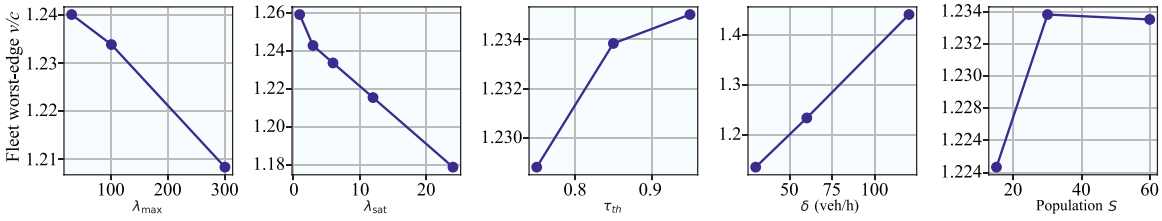


Fig. 11. Parameter sensitivity of ACBFO, with six paired instances per setting evaluated under the common default configuration. (a) Bottleneck weight $\lambda_{\max}$. (b) Saturation weight λ_{sat}. (c) Saturation threshold τ_{th}. (d) Stream rate δ. (e) Population size S.

This improvement has a measurable cost. The mean stream travel time of ACBFO is 0.4% higher than that of BFO (15.30 versus 15.24 min), reflecting the trade-off inherent in social-cost routing, where small individual detours reduce network-level saturation.

Because the congestion-guided operator embeds a shortest-path computation, a compute-matched control was run in which standard BFO receives 13,000 evaluations to equalize wall-clock time with the 8,000 evaluations of ACBFO. ACBFO retains its advantage on the matched instances ($\Delta\mathcal{F} = -1.5$ and $\Delta(v/c)_{\max} = -0.029$), so the gains are attributable to the mechanism rather than to extra computation.

A learning-based baseline is also included, denoted FQI in Table 3: a fitted Q-iteration router that assigns each request sequentially among $K = 6$ shortest-path candidates using a gradient-boosted value function, following the discrete-action formulation of deep reinforcement learning navigation (Koh et al., 2020), trained on 16 held-out instances of the same network family. Recent regional and system-optimal route-guidance studies have also adopted deep and multi-agent reinforcement learning; because their state/action spaces and macroscopic traffic models differ from the edge-level batch assignment used here, FQI is included as a training-dependent learning control rather than presented as a state-of-the-art DRL reproduction. On the unseen test instances it realizes $\mathcal{F} = 835.6 \pm 32.1$, between the free-flow and the congestion-aware exact baselines, which quantifies the transfer gap incurred by learned policies on networks they were not trained on; the search-based methods avoid this gap entirely and require no training data. Stronger graph-learning and multi-agent routers are an interesting complement left to future work.

6.5. Ablation and sensitivity analyses

Two ablations attribute the observed gains, both on the same 30 paired instances. The objective ablation runs ACBFO with the travel-time-only objective and re-evaluates the result under the full metric set. The fleet worst-edge v/c rises from 1.229 to 1.337 while mean travel time falls by only 0.9% (15.31 to 15.17 min), so the externality terms purchase a large bottleneck reduction at a small individual cost. The mechanism ablation removes each ACBFO component in turn. Removing congestion-guided chemotaxis is decisive, with $\Delta\mathcal{F} = +1.70$ and $\Delta(v/c)_{\max} = +0.031$ (both $p < 10^{-4}$), which reverts ACBFO to baseline-BFO behavior. A dedicated cold-start ablation (Fig. 10, 15 paired instances) confirms the attribution. Removing the congestion-guided operator collapses the cold-start trajectory onto that of standard BFO ($\Delta\mathcal{F} = +332$ at 1000 evaluations, $p = 10^{-4}$), while the chaotic and Lévy components act as deterministic, reproducible scheduling mechanisms whose individual removal does not significantly change solution quality in either regime ($p \geq 0.05$).

Fig. 11 sweeps the manually selected parameters on six paired instances per setting, with all results re-evaluated under the common default configuration. The conclusions are insensitive to the bottleneck weight over a tenfold range, with the fleet worst-edge v/c moving monotonically from 1.240 to 1.208 as $\lambda_{\max}$ grows from 30 to 300 at a 0.6% travel-time cost. The saturation weight trades the two metrics smoothly, with the worst-edge ratio falling from 1.259 to 1.179 and the mean travel time rising from 15.57 to 16.01 min as λ_{sat} grows from 1 to 24; the default $\lambda_{\text{sat}} = 6$ sits on the flat middle of this trade-off. They are likewise insensitive to the saturation threshold, for which the ratio varies within 0.006 over $\tau_{\text{th}} \in [0.75, 0.95]$, and to the population size, which changes the ratio by less than 0.01 over $S \in \{15, 30, 60\}$. The stream rate δ is a scenario parameter rather than a tuning parameter, since it controls the congestion regime itself, with worst-edge ratios of 1.135, 1.234, and 1.442 at $\delta = 30$, 60, and 120 veh/h, and all comparative conclusions are drawn at fixed δ.

6.6. Runtime, uncertainty-aware routing, and closed-loop operation

Runtime and scalability. All timings were measured on a single core of an Intel Xeon processor at 2.10 GHz with 4 GB of memory, using a pure Python implementation; compiled implementations

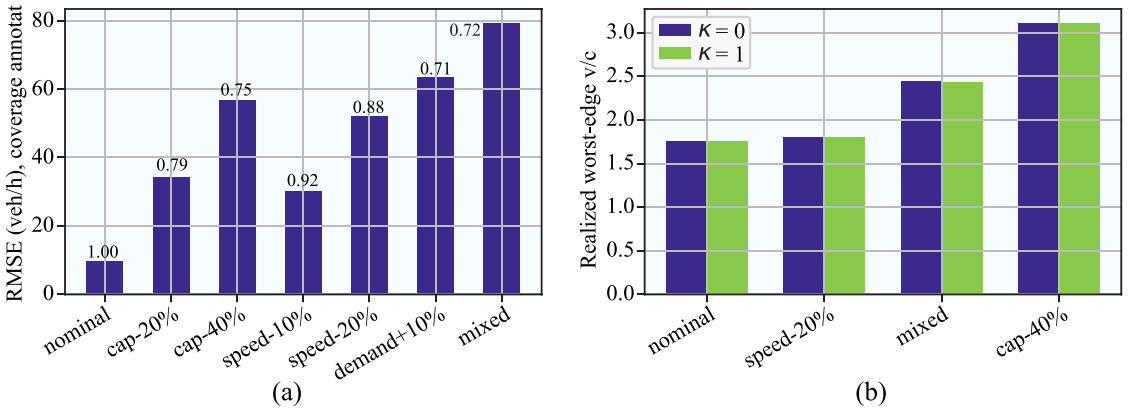


Fig. 12. Simulator-discrepancy stress test, in which the surrogate is trained on nominal dynamics and evaluated against perturbed dynamics. (a) Prediction RMSE for each perturbation, with the empirical 95% coverage annotated above each bar. (b) Realized worst-edge v/c of routing on the mismatched predictions, with and without risk inflation.

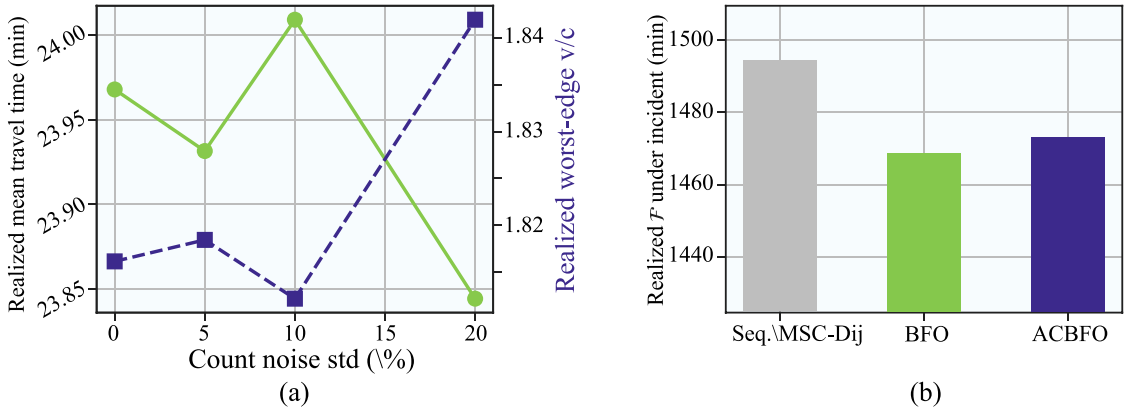


Fig. 13. Robustness tests of the prediction-and-routing pipeline. (a) Realized performance under multiplicative sensor noise on the boundary-count input. (b) Realized objective when a heavily loaded edge unexpectedly loses half its capacity after planning.

and parallel populations would reduce absolute times considerably. The perception and routing experiments were executed in different hardware/software environments, so a single camera-to-route latency was not measured in this study. The timings below therefore quantify the prediction-and-routing stack and should not be interpreted as a complete perception-to-decision latency. For the controlled counting benchmark, the association and line-crossing post-processing alone averages 0.072 ms per analyzed frame on one AMD EPYC 9V74 CPU core. Detector inference was not rerun on this same device, so these timings are reported per stage rather than aggregated. One PP-GaSP network-wide prediction takes 0.02 ms after 16 s of offline fitting on 240 runs. On the 10×10 network with 360 directed edges, sequential MSC-Dijkstra routes the full batch in 0.02 s, IBR converges in 0.11 s, and ACBFO consumes 4.8 s for a 6,000-evaluation budget. On 20×20 and 30×30 networks with 1520 and 3480 edges, the corresponding ACBFO times are 13.8 s and 27.1 s, so growth is approximately linear in network size (Fig. 14); a least-squares fit of runtime against directed-edge count over these three network sizes gives $R^2 = 0.999$. The anytime profile of Fig. 9 allows the budget to be cut to fit any decision-window deadline at a quantified quality cost, and a complete prediction-and-routing cycle of the closed-loop replay below (which routes $m = 25$ streams under a 4000-evaluation budget) completes in under 3 s per window on this hardware, occupying well under one percent of the 15-minute decision interval.

Uncertainty-aware routing under distribution shift. This experiment connects the uncertainty of the surrogate to the router. The PP-GaSP is trained with the snowy regime withheld, as in the OOD holdout of Section 6.2. Twenty-four snowy morning-peak scenarios are then planned on the risk-inflated flow estimate of Eq. (35) and evaluated on the true simulator volumes. Moderate risk inflation leaves the realized objective statistically unchanged (973.2 at $\kappa = 0.5$ versus 975.6 at $\kappa = 0$, $p = 0.21$), while the realized bottleneck falls significantly and monotonically from 1.611 at $\kappa = 0$ to 1.458 at $\kappa = 2$ ($p < 10^{-4}$), at a travel-time premium of 7.4% at the most conservative setting (Fig. 15). The benefit of uncertainty-aware routing is therefore concentrated on the risk dimension, protecting the network against worst-case saturation when the surrogate is queried outside its training distribution, rather than on the aggregate objective.

Closed-loop replay. Finally, the prediction, routing, and feedback stages are exercised jointly over 16 consecutive 15-minute decision windows spanning the morning peak on a single network. Time-varying boundary-count inputs drive the PP-GaSP prediction of the next window's volumes, the batch router assigns 25 new demand streams per window, and the assigned flows are fed back into the network state with persistence, closing the loop. Three policies are compared under identical realized conditions. Open-loop free-flow routing diverges cumulatively, as fleet flows pile onto the same corridors window after window and drive the realized mean travel time to 580 min, illustrating the compounding effect of herding. The reactive policy, a congested Dijkstra on currently observed volumes that idealizes consumer navigation with perfect instantaneous observation, and the proposed closed-loop policy both remain stable. The closed-loop policy matches the reactive

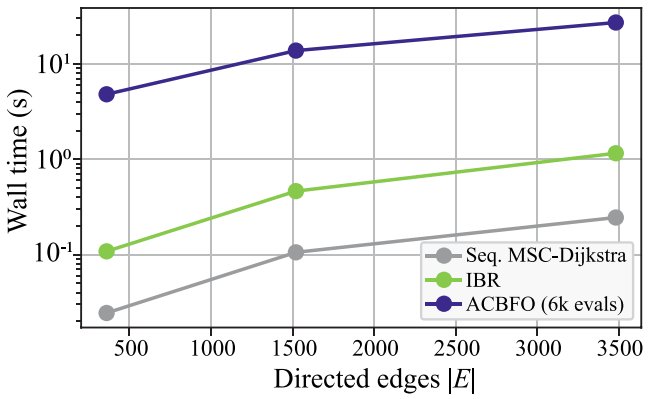


Fig. 14. Wall time versus network size for the routing stack (single-core CPU, logarithmic scale).

policy on realized travel time (30.7 versus 29.7 min) while reducing the share of saturated used edges by 10% (0.518 versus 0.579), with a prediction-and-routing latency below 3 s per window (Fig. 16). The perfect-observation assumption favors the reactive baseline, and the primary role of the case study is to test integrated prediction, routing, and feedback operation, with the saturation-exposure reduction indicating the direction of the closed-loop benefit.

Robustness to simulator discrepancy, sensor noise, and incidents. Three stress tests probe the pipeline outside its nominal operating assumptions (Figs. 12 and 13). First, the surrogate is trained on nominal dynamics while the ground truth is generated by a perturbed simulator. Under a combined perturbation (capacities −20%, speeds −10%, demand +10%), prediction RMSE grows from 9.8 to 79.4 veh/h and interval coverage falls from 1.00 to 0.72, and the realized routing quality degrades gracefully rather than abruptly, with mean travel time rising from 23.9 to 34.8 min and the realized bottleneck from 1.76 to 2.45. Because the well-fitted surrogate produces narrow in-distribution intervals, risk inflation with $\kappa = 1$ changes the realized routing only marginally under model discrepancy (bottleneck differences within 0.01); the substantive hedging benefit arises under distribution shift, where the intervals widen substantially, as shown below. Second, multiplicative sensor noise of 5% to 20% on the boundary-count input leaves realized mean travel time within 0.5% of the noise-free value, with the bottleneck rising by at most 0.03 at the 20% level; the aggregate-count input and the smoothing of the Gaussian process absorb realistic counting errors. Third, in an incident scenario an unexpected 50% capacity loss is applied to a heavily loaded edge after planning. All methods degrade, and the jointly optimized solutions degrade least. ACBFO and BFO are statistically indistinguishable under the incident ($\mathcal{F} = 1473$ and 1469) and both remain ahead of sequential MSC-Dijkstra (1495), with ACBFO retaining the lowest realized bottleneck (2.43);

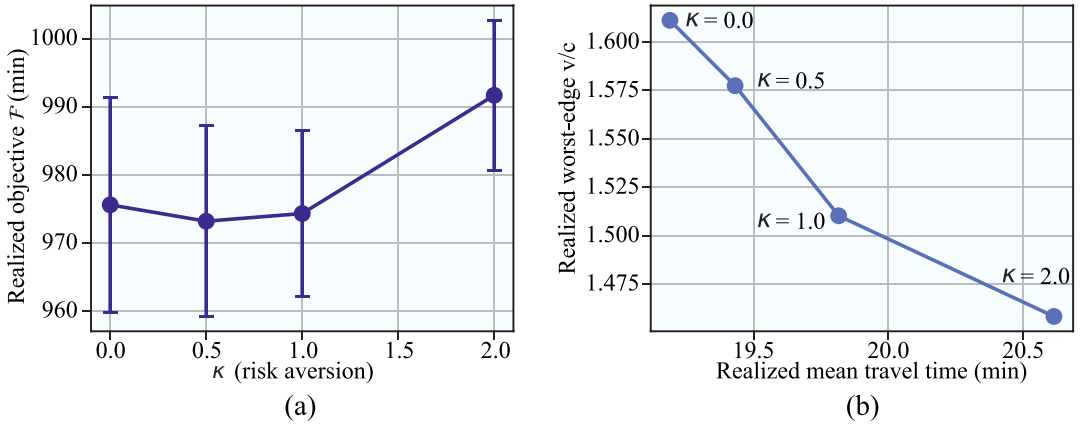


Fig. 15. Uncertainty-aware routing under distribution shift, over 24 snowy scenarios realized on true volumes. (a) Realized batch objective versus the risk factor κ. (b) Risk-return frontier between realized mean travel time and realized worst-edge v/c.

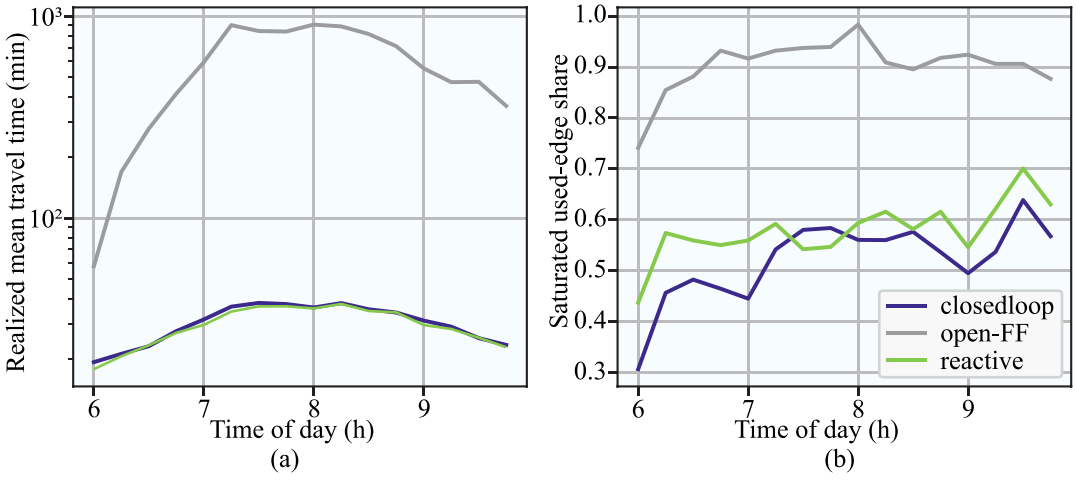


Fig. 16. Closed-loop replay over 16 morning-peak windows. (a) Realized mean travel time on a logarithmic scale; open-loop free-flow routing diverges cumulatively. (b) Saturated used-edge share; the closed-loop policy maintains the lowest saturation exposure.

the slack created by joint bottleneck avoidance improves robustness to unmodeled disturbances.

7. Conclusion

A digital twin-driven framework for dynamic route planning has been proposed to address a long-standing weakness of consumer navigation services, whose reactive rerouting improves individual trips while amplifying congestion at the network level. The framework couples YOLOv8-based perception, an LWR-based simulator, a PP-GaSP surrogate with closed-form predictive uncertainty, and a batch joint-routing layer in a closed loop. The paper shows that the per-vehicle social-cost objective is edge-separable and exactly solvable, and accordingly formulates routing as the joint assignment of concurrent demand streams under a coupled objective with a non-separable bottleneck term, which the proposed ACBFO addresses through congestion-guided chemotaxis. Across 30 paired instances, ACBFO attains the exact single-vehicle optimum on every instance, improves significantly on the exact, best-response, metaheuristic, and learning-based baselines, and delivers the lowest fleet bottleneck, while risk-inflated routing under distribution shift and a multi-window closed-loop replay demonstrate the operational value of the predictive uncertainty and of the integrated pipeline. Future work will focus on field validation with annotated intersection video, residual calibration against real traffic measurements, and game-theoretic modeling of driver compliance.

CRediT authorship contribution statement

Tingjun Lei: Conceptualization, Formal analysis, Methodology, Data curation, Validation, Writing – original draft, Supervision. **Ketian Ye:** Formal analysis, Experiments, Data curation, Validation, Writing – original draft. **Justin London:** Software, Experiments, Data curation, Validation, Writing – original draft. **Sicong Shao:** Data curation, Writing – review & editing. **Jun Chen:** Writing – review & editing, Supervision. **Chaomin Luo:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Tingjun Lei reports financial support was provided by North Dakota NASA EPSCoR. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was partially supported by the North Dakota NASA EPSCoR Research Grant and the University of North Dakota Early Career Scholars Program.

Data availability

Data will be made available on request.

References

Abdullah, A., & Oothariasamy, J. (2020). Vehicle counting using deep learning models: A comparative study. *International Journal of Advanced Computer Science and Applications*, *11*.

Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., & Gandomi, A. H. (2021). The arithmetic optimization algorithm. *Computer Methods in Applied Mechanics and Engineering*, *376*, Article 113609.

Adl, M., Ahmed, R., Vidal, C., & Emadi, A. (2016). Enhanced vehicle movement counting at intersections via a self-learning fisheye camera system. *IEEE Access*.

Adnekouei, F. (2024a). Top-view vehicle detection image dataset. Kaggle dataset.

Adnekouei, F. (2024b). YOLOv8 traffic density estimation. GitHub repository.

Chintam, P., Lei, T., Osmanoglu, B., Wang, Y., & Luo, C. (2024). Informed sampling space driven robot informative path planning. *Robotics and Autonomous Systems*, Article 104656.

Dey, B., & Kundu, M. (2019). Turning video into traffic data: An application to urban intersection analysis using transfer learning. *IET Image Processing*, *13*, 673–679.

Gochoo, M., Otgonbold, M., Ganfold, J., Hsieh, J., Chang, M., Chen, P., Dorj, B., Al Jassmi, H., Batnasan, G., Alnajjar, F., Abuljabbar, M., & Lin, F. (2023). Fisheye8K: A benchmark and dataset for fisheye camera object detection. In *IEEE/CVF conference on computer vision and pattern recognition workshops* (pp. 5305–5315).

Gomes, B., Coelho, J., & Aidos, H. (2023). A survey on traffic flow prediction and classification. *Intelligent Systems with Applications*, *20*, Article 200268. http://dx.doi.org/10.1016/j.iswa.2023.200268.

Gu, M., & Berger, J. O. (2016). Parallel partial Gaussian process emulation for computer models with massive output. *The Annals of Applied Statistics*, *10*, 1317–1347.

Gu, M., Palomo, J., & Berger, J. O. (2019). RobustGaSP: Robust Gaussian stochastic process emulation in R. *The R Journal*, *11*, 112–136.

Hicks, M., Lei, T., Luo, C., Carruth, D. W., & Bi, Z. (2025). A bio-inspired goal-directed cognitive map model for robot navigation and exploration. *IEEE Transactions on Cognitive and Developmental Systems*, *17*(5), 1125–1140.

Ibryaeva, O., Zhulev, A., Shepelev, V., & Burzev, A. (2019). Vehicle counting at road intersections using video data and zone-based method with additional contour. *IOP Conference Series: Materials Science and Engineering*.

Jan, G. E., Lei, T., Sun, C. C., You, Z. Y., & Luo, C. (2024). On the problems of drone formation and light shows. *IEEE Transactions on Consumer Electronics*.

Jiang, S., Tran, C. Q., & Keyvan-Ekbatani, M. (2024). Regional route guidance with realistic compliance patterns: Application of deep reinforcement learning and MPC. *Transportation Research Part C: Emerging Technologies*, *158*, Article 104440. http://dx.doi.org/10.1016/j.trc.2023.104440.

Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLOv8. GitHub repository.

Kasturi, R., Goldgof, D., Soundararagan, P., Manohar, V., & Garofolo, J. (2009). Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*, 319–336.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks* (pp. 1942–1948).

Koh, S., Zhou, B., Fang, H., Yang, P., Yang, Z., Yang, Q., Guan, L., & Ji, Z. (2020). Real-time deep reinforcement learning based vehicle navigation. *Applied Soft Computing*, *96*, Article 106694.

Lei, T., Luo, C., Sellers, T., & Rahimi, S. (2021). A bat-pigeon algorithm to crack detection-enabled autonomous vehicle navigation and mapping. *Intelligent Systems with Applications*, *12*, Article 200053. http://dx.doi.org/10.1016/j.iswa.2021.200053.

Lei, T., Luo, C., Yang, S. X., Carruth, D. W., & Bi, Z. (2023). Bio-inspired intelligence-based multi-agent navigation with safety-aware considerations. *IEEE Transactions on Artificial Intelligence*.

Lei, T., Sellers, T., Luo, C., Cao, L., & Bi, Z. (2024). Digital twin-based multi-objective autonomous vehicle navigation approach as applied in infrastructure construction. *IET Cyber-Systems and Robotics*, *6*(2), Article e12110.

Lei, T., Sellers, T., Luo, C., Carruth, D. W., & Bi, Z. (2023). Graph-based robot optimal path planning with bio-inspired algorithms. *Biomimetic Intelligence and Robotics*, Article 100119.

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2018). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International conference on learning representations*.

Lighthill, M. J., & Whitham, G. B. (1955). On kinematic waves II: A theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London A*, *229*, 317–345.

Lin, H., Yuan, Z., He, B., Kuai, X., Li, X., & Guo, R. (2022). A deep learning framework for video-based vehicle counting. *Frontiers in Physics*, *10*.

London, J., Lei, T., Luo, C., & Yuan, Y. (2025). Human activity recognition and biomedical signal classification using multi-modal deep learning. In *2025 IEEE 68th international midwest symposium on circuits and systems* (pp. 676–680). IEEE.

London, J., Lei, T., Yuan, Y., Shao, S., & Hu, J. (2025). Enhanced biomedical signal processing using time-frequency analysis and wavelet transforms. In *2025 IEEE 68th international midwest symposium on circuits and systems* (pp. 463–467). IEEE.

Lu, J., Xia, M., Gao, X., Yang, X., Tao, T., Meng, H., Zhang, W., Tan, X., Shi, Y., Li, G., & Ding, E. (2021). Robust and online vehicle counting at crowded intersections. In *IEEE/CVF conference on computer vision and pattern recognition workshops* (pp. 1–7).

Mandellos, N., Keramitsoglou, I., & Kiranoudis, C. (2011). A background subtraction algorithm for detecting and tracking vehicles. *Expert Systems with Applications*, *38*, 1619–1631.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51–67.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, *69*, 46–61.

Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, *22*, 52–67.

Ponti, A., Candelieri, A., & Archetti, F. (2021). A wasserstein distance based multiobjective evolutionary algorithm for the risk aware optimization of sensor placement. *Intelligent Systems with Applications*, *10–11*, Article 200047. http://dx.doi.org/10.1016/j.iswa.2021.200047.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *IEEE conference on computer vision and pattern recognition* (pp. 779–788).

Richards, P. I. (1956). Shock waves on the highway. *Operations Research*, *4*, 42–51.

Riser, E., Sellers, T., Lei, T., Pradeep, C., & Luo, C. (2024). Multi-robot navigation using improved rrt*-smart with digital twin technology. In *Autonomous systems: sensors, processing, and security for ground, air, sea, and space vehicles and infrastructure 2024* (pp. 114–125). SPIE.

Roboflow (2023). Traffic intersection vehicle detection. Roboflow Universe dataset.

Sahib, M. A., Abdulnabi, A. R., & Mohammed, M. A. (2018). Improving bacterial foraging algorithm using non-uniform elimination-dispersal probability distribution. *Alexandria Engineering Journal*, *57*, 3341–3349. http://dx.doi.org/10.1016/j.aej.2018.01.022.

Sellers, T., Lei, T., Luo, C., Bi, Z., & Jan, G. E. (2024). Human autonomy teaming-based safety-aware navigation through bio-inspired and graph-based algorithms. *Biomimetic Intelligence and Robotics*, *4*(4), Article 100189.

Sellers, T., Lei, T., Luo, C., Jan, G. E., & Ma, J. (2022). A node selection algorithm to graph-based multi-waypoint optimization navigation and mapping. *Intelligence & Robotics*, *2*(4), 333–354.

Sheffi, Y. (1985). *Urban transportation networks: Equilibrium analysis with mathematical programming methods*. Prentice-Hall.

Short, D., Lei, T., Carruth, D. W., Luo, C., & Bi, Z. (2023). A bio-inspired algorithm in image-based path planning and localization using visual features and maps. *Intelligence & Robotics*, *3*(2), 222–241.

Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H., & Sui, F. (2018). Digital twin-driven product design, manufacturing and service with big data. *International Journal of Advanced Manufacturing Technology*, *94*, 3563–3576.

Vasu, M., Breu, N., Vasquez, R., & Lopez, C. (2021). Vehicle-counting with automatic region-of-interest and driving-trajectory detection. In *Proceedings of the latinX in AI*.

Wang, Z., Gupta, R., Han, K., Wang, H., Ganlath, A., Ammar, N., & Tiwari, P. (2022). Mobility digital twin: Concept, architecture, case study, and future challenges. *IEEE Internet of Things Journal*, *9*, 17452–17467.

Wardrop, J. G. (1952). Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers*, *1*, 325–362.

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *Proceedings of the IEEE international conference on image processing* (pp. 3645–3649).

Wu, Z., Pan, S., Long, G., Jiang, J., & Zhang, C. (2019). Graph WaveNet for deep spatial-temporal graph modeling. In *Proceedings of the 28th international joint conference on artificial intelligence* (pp. 1907–1913).

Yu, B., Yin, H., & Zhu, Z. (2018). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the 27th international joint conference on artificial intelligence* (pp. 3634–3640).

Yun, H., Kim, E.-j., Ham, S. W., & Kim, D.-K. (2024). Navigating the non-compliance effects on system optimal route guidance using reinforcement learning. *Transportation Research Part C: Emerging Technologies*, *165*, Article 104721. http://dx.doi.org/10.1016/j.trc.2024.104721.