EUROPEAN JOURNAL OF PURE AND APPLIED MATHEMATICS
2025, Vol. 18, Issue 4, Article Number 7091
ISSN 1307-5543 – ejpam.com
Published by New York Business Global

Optimizing Communication Network Routing with A* Algorithm: A Comparative Study Against Bellman-Ford and Dijkstra for Enhanced Quality of Service

Mohamed Ayari[1,*], Atef Gharbi[2], Zeineb Klai[3], Abdelhalim Hasnaoui[4], Mahmoud Salaheldin Elsayed[3], Elsaid Md. Abdelrahim[5]

[1] *Department of Information Technology, Faculty of Computing and Information Technology, Northern Border University, Kingdom of Saudi Arabia*
[2] *Department of Information Systems, Faculty of Computing and Information Technology, Northern Border University, Kingdom of Saudi Arabia*
[3] *Department of Computer Sciences, Faculty of Computing and Information Technology, Northern Border University, Kingdom of Saudi Arabia*
[4] *Mathematics Department, College of Sciences and Arts, Northern Border University, Kingdom of Saudi Arabia*
[5] *Computer Science Department, Science College, Northern Border University (NBU), Arar 73213, Saudi Arabia*

Abstract. Efficient routing is critical to modern networks, where low latency, QoS guarantees, and energy efficiency must be maintained despite dynamic conditions. This paper applied the A* algorithm to QoS-aware routing and compared it with Bellman–Ford and Dijkstra in 5G/6G slicing, IoT multi-hop, and SDN/backbone settings. We introduced admissible domain-specific heuristics that integrate graph distance with live network indicators (queue delay, link loss, available bandwidth, and residual energy). Across trace-driven and synthetic topologies, A* achieved lower end-to-end latency and jitter, faster convergence, and reduced routing overhead relative to baselines, while maintaining competitive delivery ratio and energy per delivered bit. When admissibility held, A-Star preserved optimal paths with fewer expansions. With non-admissible but informative heuristics, it delivered favorable latency–overhead trade-offs. The results demonstrate that heuristic-guided routing can meet emerging QoS demands in real-time and resource-constrained networks, providing a scalable and implementation-ready alternative to traditional shortest-path methods.

2020 Mathematics Subject Classifications: 91A10, 91A35, 90C59, 78M50

Key Words and Phrases: A-Star (A*) Algorithm, Heuristic Routing, QoS-Aware Routing, Dijkstra Algorithm, Bellman–Ford Algorithm, 5G/6G Networks

*Corresponding author.
DOI: https://doi.org/10.29020/nybg.ejpam.v18i4.7091

Email address: Mohamed.ayari@nbu.edu.sa (M. Ayari)

1. Introduction

Next-generation communication systems—from 5G/6G mobile networks and cloud backbones to dense Internet of Things (IoT) deployments—demand routing solutions that deliver *low latency*, predictable *Quality of Service (QoS)*, and *energy efficiency* under time-varying traffic and topology changes. In 5G, for example, QoS flows and network slicing impose differentiated performance targets (latency, loss, jitter, and bandwidth reservation) across heterogeneous segments; moving toward 6G further amplifies the requirements for reliability, ultra-low latency, and pervasive intelligence across the control and user planes [1–3]. Software-Defined Networking (SDN) and centralized control provide global visibility to implement QoS-aware routing and traffic engineering, but rapidly finding near-optimal paths as conditions evolve remains a challenge at scale [4–9].

Classic shortest-path algorithms underpin network routing. *Bellman–Ford* supports negative edge weights and underlies distance-vector protocols, whereas *Dijkstra*—with heap-based priority queues—underlies link-state protocols under non-negative weights. This motivates our A-Star comparison across both families. However, both expand the search space without considering destination-directed guidance, which can inflate convergence time and control overhead in dynamic environments with tight QoS constraints [10, 11]. In contrast, the $A*$ algorithm increases the cost-to-come with a problem-informed heuristic estimate of cost-to-go, prioritizing promising frontiers and reducing unnecessary node expansions while preserving optimality when the heuristic is admissible and consistent [12, 13]. These properties make A* an attractive candidate for *QoS-aware routing*, provided that we can craft admissible or bounded heuristics that encode network conditions such as queueing delay, loss probability, residual bandwidth, and energy budgets.

Recent work across SDN, IoT, and satellite networks highlights the benefits of heuristic or learning-guided path selection. In SDN, dynamic, traffic-aware controllers and deep reinforcement learning (DRL) agents have been shown to reduce delay, smooth jitter, and balance load while responding to failures and bursts [6–9, 14]. In IoT and wireless sensor networks (WSNs), where energy is at a premium, energy-aware QoS routing remains central; contemporary surveys and designs report improvements in network lifetime and delivery ratio by using multi-constraint metrics (delay, reliability, residual energy) [15–19]. Even in non-terrestrial networks such as LEO satellite constellations, route selection increasingly couples energy models with QoS requirements [20]. Meanwhile, 6G-oriented studies emphasize that tight QoS guarantees must co-exist with high mobility and extreme heterogeneity, strengthening the case for search strategies that prune the exploration space effectively [2, 3]. Multi-criteria evaluation of uncertain, context-sensitive constraints can be formalized using intuitionistic fuzzy-set frameworks, which aligns with QoS trade-off decisions in heuristic routing [21].

Against this backdrop, we investigate **A* for QoS-aware routing** in communication networks and compare it with **Bellman–Ford** and **Dijkstra**. Our core thesis is that *domain-informed heuristics*—built from graph-geodesic priors and live signals (e.g., measured queuing delay, short-term loss, available bandwidth, residual energy)—allow A* to *(i)* reduce search overhead and convergence time, *(ii)* maintain optimality under

admissible heuristics (or provide bounded suboptimality under weighted heuristics), and *(iii)* realize favorable latency–energy trade-offs in diverse topologies (5G/6G slices, IoT multi-hop meshes, and SDN backbones). We evaluate latency, jitter, packet delivery ratio, path stretch, convergence time, routing/control overhead, and energy per delivered bit using trace-driven and synthetic workloads under dynamic link conditions and failures. We also situate A* within learning-augmented controllers (e.g., DRL) by using A* as a deterministic inner solver with learned *cost-to-go* surrogates [3, 7–9].

The paper makes three contributions: (1) a unified *QoS-aware heuristic design* for A*, integrating readily measurable network signals while maintaining admissibility where required; (2) a head-to-head comparison of A*, Dijkstra, and Bellman–Ford across heterogeneous topologies and QoS regimes, including energy-sensitive IoT and slice-aware SDN/5G scenarios; and (3) an empirical analysis of convergence, overhead, and QoS compliance showing when A* yields the largest gains relative to baselines [10, 14, 15, 17, 20].

Our study targets scenarios where controllers (or distributed agents) can estimate or bound link-level metrics at sub-second timescales and where latency, jitter, or loss SLOs must be met despite churn. This includes URLLC-like flows in 5G/6G, bandwidth and energy-constrained IoT telemetry, and SDN-operated edge/core fabrics [1, 2, 4, 22].

The remainder of this paper is structured as follows: **Section 2** reviews background and related work on QoS routing and path-search algorithms; **Section 3** formalizes the QoS-aware routing problem and metrics; **Section 4** details A*, Dijkstra, and Bellman–Ford implementations and our heuristic design; **Section 5** describes datasets, topologies, and experimental methodology; **Section 6** reports results and analysis; **Section 7** discusses implications, limitations, and extensions (including weighted/learned heuristics); and **Section 8** concludes.

2. Background and Related Work

Ensuring end-to-end performance in modern networks means meeting diverse *Quality of Service* (QoS) goals—low latency, bounded jitter, packet delivery guarantees, and bandwidth reservations—under time-varying demand and failures. The advent of 5G/6G introduces explicit mechanisms (e.g., QoS flows and network slicing) that expose these goals as first-class constraints for control-plane decisions, while disaggregation via software-defined networking (SDN) centralizes path computation and admission control [1, 2, 4]. In such settings, routing must rapidly *find* and *maintain* paths that optimize multi-constraint objectives while scaling to large fabrics. For algebraic generalization beyond crisp graphs, fuzzy Γ-semimodules over Γ-semirings offer a perspective on relational path structures that could support extensions of our model [23].

2.1. Quality of Service-Aware Routing: Concepts and Challenges

Classical shortest-path routing assumes a single additive metric (e.g., hop count or static link cost). QoS-aware routing is more complex: (i) it may combine additive (delay), multiplicative (loss), and concave (bottleneck bandwidth) metrics; (ii) it must react on sub-second timescales to congestion, failures, and mobility; and (iii) it often targets flow- or slice-specific Service Level Objectives (SLOs) [1, 4]. Central controllers (SDN) can, in principle, compute globally optimal paths with updated telemetry, but repeated recomputation over large topologies can inflate control overhead and reaction time [4].

2.2. Canonical Algorithms for Shortest Paths

Bellman–Ford and **Dijkstra** remain the canonical baselines. Bellman–Ford, a label-correcting method, supports negative edge weights and detects negative cycles at the cost of $O(V \cdot E)$ time [11, 24]. Dijkstra's label-setting method achieves $O((V + E) \log V)$ with heaps, but assumes non-negative weights [11, 25]. Both expand search frontiers without destination-directed guidance, which can slow convergence when telemetry frequently shifts and when multiple QoS constraints narrow the feasible region.

2.3. Heuristic Search and A*

A* augments the cost-to-come $g(n)$ with a problem-informed heuristic $h(n)$ estimating the cost-to-go, ranking candidates by $f(n) = g(n)+h(n)$ [13]. For *admissible* and *consistent* heuristics, A* preserves optimality while expanding far fewer nodes than Dijkstra; *weighted* $A*$ trades bounded suboptimality for additional pruning. Systematic reviews confirm A*'s efficiency benefits hinge on heuristic quality [12]. In networking, domain-informed $h(\cdot)$ can incorporate geodesic distance plus live signals (queueing delay, loss, residual bandwidth, residual energy), turning A* into a *QoS-aware* search that prunes unactionable branches early.

2.4. SDN: Centralized Control and Learning-Augmented Routing

With SDN, the controller holds a global view and can embed telemetry into routing objectives. Recent work demonstrates that *heuristic* and *learning-augmented* controllers reduce delay and packet loss while balancing load [4, 6, 8, 26, 27]. Reinforcement-learning (RL) agents coupled with graph neural networks (GNNs) accelerate convergence and generalize across topologies; hybrid DRL+GNN methods directly emit routing policies that adapt to nonstationary traffic [28, 29]. These trends support our thesis: when combined with *informed* heuristics, A* can act as a fast inner solver, while learning refines $h(\cdot)$ or prioritizes path candidates.

2.5. Quality of Service and Energy in IoT/WSN and Beyond

In IoT and Wireless Sensor Networks (WSNs), routing must be energy-aware while honoring latency and reliability constraints. Recent studies propose QoS-aware and energy-

efficient schemes (e.g., clustering with trust and link-quality estimation; multipath, fuzzy, and metaheuristic designs) that extend lifetime and improve delivery [19, 30–32]. These domains are natural fits for A* with energy-informed heuristics (residual energy, link reliability), where pruning reduces radio activity and control traffic.

2.6. Non-Terrestrial and Multi-Domain Networks

Low Earth Orbit (LEO) constellations and multi-domain backbones bring additional constraints (satellite energy budgets, dynamic link geometry), where routing couples QoS with energy models [20]. Here, heuristic guidance—e.g., orbital geometry and predicted contact windows—can concentrate search on feasible corridors, again aligning with A*'s strengths.

2.7. Positioning This Work

Prior art demonstrates: (i) scalable SDN controllers benefit from search-space reduction and topology-aware priors [4, 6], (ii) DRL/GNN policies can embed network state for robust routing [26–29], and (iii) energy-aware QoS routing is pivotal in IoT/WSN and satellite networks [16, 20, 30–32]. However, the literature lacks a unified evaluation of *A* with QoS-informed heuristics* against Bellman–Ford and Dijkstra across heterogeneous topologies under realistic telemetry. This paper fills that gap by (1) proposing admissible and weighted heuristics that combine graph distance with live link metrics; (2) benchmarking A*, Dijkstra, and Bellman–Ford under 5G/6G slices, IoT multi-hop meshes, and SDN backbones; and (3) analyzing latency, jitter, delivery ratio, path stretch, convergence time, routing overhead, and energy per delivered bit.

2.8. Takeaways

A* offers a principled way to leverage *problem structure* in routing. When heuristics encode current network conditions and constraints, A* can reduce expansions and control overhead while preserving optimality or offering bounded trade-offs—a compelling advantage in time-sensitive and energy-constrained environments [4, 12, 13, 16, 26].

3. Problem Formulation and Quality of Service Metrics

This section formalizes the Quality of Service (QoS)–aware routing problem on a directed graph, defines the feasibility constraints used throughout the paper, and specifies the scalarized objective and per-edge cost adopted by all algorithms. It then details our adaptation of A-Star for QoS-aware routing, including the heuristic design, indicator normalization, weight selection, and the evaluation setup. We provide clear notation and conservative heuristic components to ensure compatibility with the shortest-path solvers evaluated later.

3.1. Network Model and Path Metrics

We represent the network as a directed graph $G = (V, E)$ with $|V| = n$ and $|E| = m$. At decision time τ, each directed edge $e = (u, v) \in E$ carries the following telemetry: latency $d_e(\tau)$, queueing term $q_e(\tau)$, loss probability $\lambda_e(\tau)$, available bandwidth $b_e(\tau)$, and energy per delivered bit $\varepsilon_e(\tau)$. For a simple path $P = \langle e_1, \ldots, e_k \rangle$ from source s to destination t, we use the additive and bottleneck metrics

$$D(P) = \sum_{e \in P} d_e, \qquad Q(P) = \sum_{e \in P} q_e, \tag{1}$$

$$J(P) = \frac{1}{k} \sum_{e \in P} (q_e - \bar{q})^2, \qquad \bar{q} = \frac{1}{k} \sum_{e \in P} q_e, \tag{2}$$

We refer to $J(P)$ as a *jitter proxy* (queue-variance proxy): while it captures per-link queue variability along path P, it is only an approximation of true end-to-end inter-arrival jitter.

$$\Lambda(P) = 1 - \prod_{e \in P} (1 - \lambda_e), \qquad \widehat{L}(P) = \sum_{e \in P} (-\ln(1 - \lambda_e)), \tag{3}$$

$$B(P) = \min_{e \in P} b_e, \qquad E(P) = \sum_{e \in P} \varepsilon_e. \tag{4}$$

Here $\widehat{L}(P)$ is an additive surrogate for end-to-end loss that enables standard graph search.

The metrics in (1)–(4) capture latency, jitter proxy $J(P)$, reliability, bandwidth, and energy in forms amenable to shortest-path formulations while preserving the operational meaning of QoS targets.

3.2. Quality of Service Feasibility and Scalarized Objective

A flow $f = (s, t, \bar{D}, \bar{J}, \bar{\Lambda}, \bar{B}, \bar{E})$ is *feasible* on path P when the service-level objectives (SLOs) are met:

$$D(P) \le \bar{D}, \qquad J(P) \le \bar{J}, \qquad \Lambda(P) \le \bar{\Lambda}, \qquad B(P) \ge \bar{B}, \qquad E(P) \le \bar{E}. \tag{5}$$

To compare alternative feasible routes with a single score, we use a nonnegative scalarization with weights $\alpha_D, \alpha_L, \alpha_J, \alpha_E, \alpha_B$ and normalizers $\sigma_D, \sigma_L, \sigma_J, \sigma_E, \sigma_B$:

$$C(P) = \alpha_D \frac{D(P)}{\sigma_D} + \alpha_L \frac{\widehat{L}(P)}{\sigma_L} + \alpha_J \frac{J(P)}{\sigma_J} + \alpha_E \frac{E(P)}{\sigma_E} + \alpha_B \, \phi(B(P)), \tag{6}$$

where the shortfall penalty for bandwidth is

$$\phi(B) = \begin{cases} 0, & B \ge \bar{B}, \\ \left(\dfrac{\bar{B} - B}{\sigma_B} \right)^2, & B < \bar{B}. \end{cases} \tag{7}$$

The single-flow problem is then

$$P^* \in \arg\min_{P \text{ s.t. } (5)} C(P). \tag{8}$$

When temporary violations are permitted for comparison purposes, we also report the relaxed objective

$$\mathcal{L}(P) = C(P)+\mu_D \max\{D(P)-\bar{D},0\}+\mu_J \max\{J(P)-\bar{J},0\}+\mu_L \max\{\widehat{L}(P)+\ln(1-\bar{\Lambda}),0\}+\mu_E \max\{E(P)-\bar{E},0 \tag{9}$$

with nonnegative multipliers $\mu_\cdot$.

The scalarization in (6) preserves feasibility via (5) while aligning the optimizer with delay/loss-sensitive operation; the relaxed score $\mathcal{L}(P)$ provides a consistent way to rank paths in constraint-edge cases.

3.3. Per-Edge Additive Cost for Graph Search

To enable single-source shortest-path solvers, we define a nonnegative additive edge weight

$$w_e = \alpha_D \frac{d_e}{\sigma_D} + \alpha_L \frac{-\ln(1-\lambda_e)}{\sigma_L} + \alpha_J \frac{(q_e-\hat{q})^2}{\sigma_J} + \alpha_E \frac{\varepsilon_e}{\sigma_E} + \alpha_B\, \varphi(b_e), \tag{10}$$

with a running estimate $\hat{q}$ of queueing level and

$$\varphi(b_e) = \begin{cases} 0, & b_e \geq \bar{B}, \\ \left(\dfrac{\bar{B}-b_e}{\sigma_B}\right)^2, & b_e < \bar{B}. \end{cases} \tag{11}$$

For any path P that satisfies $B(P) \geq \bar{B}$, one has $C(P) \leq \sum_{e\in P} w_e$, which justifies the use of w_e within shortest-path algorithms.

The additive structure in (10) upper-bounds the path score for bandwidth-feasible routes, allowing direct use of label-setting or heuristic search without violating the intended QoS trade-offs.

3.4. A* Heuristic and Weighted Variant

Let $g(v)$ denote the cost-to-come to node v, and let $h(v)$ be a conservative lower bound on the cost-to-go from v to t. A* ranks nodes using

$$f(v) = g(v) + h(v), \qquad g(s) = 0. \tag{12}$$

We decompose $h(v)$ into admissible component bounds:

$$h(v) = \alpha_D \frac{\underline{d}(v,t)}{\sigma_D} + \alpha_L \frac{\underline{\ell}(v,t)}{\sigma_L} + \alpha_J \frac{\underline{\jmath}(v,t)}{\sigma_J} + \alpha_E \frac{\underline{\varepsilon}(v,t)}{\sigma_E} + \alpha_B\, \underline{\beta}(v,t), \tag{13}$$

where $\underline{d}$ is a latency lower bound, $\underline{\ell}$ is an additive loss-surrogate bound, $\underline{\jmath}$ bounds the queue-variance proxy, $\underline{\varepsilon}$ bounds energy, and $\underline{\beta}$ is zero when a $\bar{B}$-feasible corridor exists and positive otherwise. The weighted variant uses

$$f_\omega(v) = g(v) + \omega\, h(v), \qquad \omega \geq 1, \tag{14}$$

to trade additional pruning for a controlled increase in path cost with respect to the scalarized objective.

When the components in (13) are admissible (and preferably consistent), A* preserves optimality and expands fewer nodes than unguided search; the weighted form (**??**) further reduces expansions with a predictable, bounded stretch in score.

3.5. Normalization and Weighting of Real-Time Indicators

We form the heuristic as a weighted combination of normalized link indicators:

$$h(n) \;=\; \alpha\,\widetilde{\ell}(n) \;+\; \beta\,\widetilde{\lambda}(n) \;+\; \gamma\,\widetilde{b}(n) \;+\; \delta\,\widetilde{e}(n),$$

where $\widetilde{\ell}$ is normalized queueing delay, $\widetilde{\lambda}$ is normalized link loss, $\widetilde{b}$ is normalized inverse bandwidth cost, and $\widetilde{e}$ is normalized inverse residual energy cost. We use min–max normalization per metric $m \in \{\ell, \lambda, b, e\}$ over all candidate edges E_t at time t:

$$\widetilde{m}(e) = \frac{m(e) - \min_{e' \in E_t} m(e')}{\max_{e' \in E_t} m(e') - \min_{e' \in E_t} m(e') + \varepsilon},$$

with $\varepsilon = 10^{-9}$ to avoid division by zero. We constrain $\alpha, \beta, \gamma, \delta \geq 0$ and $\alpha+\beta+\gamma+\delta = 1$.

Weight selection. Unless otherwise stated, we select $(\alpha, \beta, \gamma, \delta)$ by grid search on a held-out scenario set using end-to-end latency as the primary objective and control-message overhead as a tiebreaker. Energy sensitivity is calibrated by increasing δ until path-stretch remains within 1% of the δ=0 baseline. This balances QoS and energy, consistent with energy-driven routing principles in dynamic networks [33]. For completeness, we report the chosen weights with 95% CIs in Table **??**.

3.6. Optional Multi-Flow Formulation

For a set of flows $\mathcal{F}$ with demands r_f, the binary variable $x_{e,f} \in \{0, 1\}$ indicates whether flow f uses edge e. Flow conservation at node u is

$$\sum_{(u,v)\in E} x_{(u,v),f} - \sum_{(v,u)\in E} x_{(v,u),f} = \begin{cases} 1, & u = s_f, \\ -1, & u = t_f, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

Capacity feasibility is enforced as

$$\sum_{f \in \mathcal{F}} r_f\, x_{e,f} \leq b_e \quad \text{for all } e \in E. \tag{16}$$

In our single-flow experiments, (16) reduces to the bottleneck constraint $B(P) \geq \bar{B}$ used in (5)–(6).

The multi-flow form extends the single-route objective to shared-capacity settings; the single-flow case studied empirically is its natural special instance.

3.7. Complexity and Control Overhead Measures

To connect the formulation to operational cost, we track asymptotic complexity and control-plane proxies used in the evaluation:

$$\begin{aligned} \text{Time(Bellman–Ford)} &= O(nm), \\ \text{Time(Dijkstra)} &= O((n+m)\log n), \\ \text{Time(A*)} &= O((n+m)\log n) + \text{expansions}, \end{aligned} \tag{17}$$

and

$$\text{Routing overhead} = \frac{\text{control bytes}}{\text{second}}, \qquad \text{Controller CPU} = \frac{\text{search CPU time}}{\text{decision interval}}. \tag{18}$$

Informative lower bounds reduce expansions in (17) and are expected to lower the overhead proxies in (18), which is consistent with the measurements reported in Section 6.

3.8. Notation and Quality of Service Summary Tables

For completeness, Tables 1 and 2 summarize symbols and feasibility conditions used throughout.

Table 1: Notation used in Section 3.

Symbol	Meaning
$G = (V, E)$	Directed network graph
$e = (u, v)$	Directed edge from u to v
$d_e, q_e, \lambda_e, b_e, \varepsilon_e$	Edge latency, queue term, loss prob., bandwidth, energy/bit
$D(P), J(P), \widehat{L}(P), B(P), E(P)$	Path delay, jitter proxy, additive loss, bottleneck, energy
$\bar{D}, \bar{J}, \bar{\Lambda}, \bar{B}, \bar{E}$	QoS targets for delay, jitter, loss, bandwidth, energy
w_e	Additive edge weight in (10)
$g(v), h(v), f(v)$	Cost-to-come, heuristic, evaluation key
$\alpha_\cdot, \sigma_\cdot$	Weights and normalizers in (6)

Table 2: QoS metrics and feasibility conditions.

Metric	Definition	Feasibility condition
Latency	$D(P) = \sum d_e$	$D(P) \leq \bar{D}$
Jitter proxy	$J(P)$ in (2)	$J(P) \leq \bar{J}$
Loss	$\Lambda(P)$ in (3)	$\Lambda(P) \leq \bar{\Lambda}$
Bandwidth	$B(P) = \min b_e$	$B(P) \geq \bar{B}$
Energy	$E(P) = \sum \varepsilon_e$	$E(P) \leq \bar{E}$

4. Methodology and Experimental Setup

This section describes the algorithms under test, the QoS-aware cost and heuristic design, the evaluated topologies and traffic/failure models, the parameterization used across experiments, and the evaluation protocol. Each subsection begins with a concise setup paragraph and ends with a short interpretation that anticipates the analyses reported in Section 6.

4.1. Algorithms Under Test

We compare three classical shortest-path solvers configured with the same QoS-aware edge cost w_e from (10): **A*** (admissible and weighted variants), **Dijkstra**, and **Bellman–Ford**. All run on the same directed graph and telemetry. A* additionally uses a domain-informed heuristic $h(\cdot)$ as in (13).

Table 3: Baseline characteristics of the compared algorithms.

Algorithm	Negative Weights	Time Complexity	Space	Optimality	Notes
Bellman–Ford	Yes (no neg. cycle)	$O(nm)$	$O(n)$	Yes	Label-correcting; iterative relaxations
Dijkstra (heap)	No	$O((n+m)\log n)$	$O(n+m)$	Yes	Label-setting; unguided exploration
A* (heap)	No ($w_e \geq 0$)	$O((n+m)\log n)$ + expansions	$O(n+m)$	Yes (admissible h)	Guided by h; weighted A* allows bounded suboptimality

Algorithm 1: QoS-Aware A* (single flow).

Table 4: Algorithm 1: QoS-Aware A*.

Input: $G = (V, E)$, source s, target t; SLOs $(\bar{D}, \bar{J}, \bar{\Lambda}, \bar{B}, \bar{E})$; weights α; scales σ. **Output:** Feasible path P minimizing $C(P)$ in (6) or INFEASIBLE.
1. Compute w_e for all $e \in E$ using (10). 2. Initialize open set $\mathcal{O} = \{s\}$ with $g(s) = 0$, $f(s) = h(s)$; closed set $\mathcal{C} = \emptyset$. 3. While $\mathcal{O} \neq \emptyset$: (a) Extract $u = \arg\min_{v \in \mathcal{O}} f(v)$; move u to $\mathcal{C}$. If $u = t$, return the reconstructed P. (b) For each $(u, v) \in E$ with $v \notin \mathcal{C}$, set $g'(v) = g(u) + w_{(u,v)}$. (c) Optionally prune partial paths that already violate soft QoS budgets in (9). (d) If $v \notin \mathcal{O}$ or $g'(v) < g(v)$, set $g(v) \leftarrow g'$, $f(v) \leftarrow g(v) + h(v)$, parent$(v) \leftarrow u$, and add v to $\mathcal{O}$. 4. If the loop ends without reaching t, return INFEASIBLE.

Algorithm 2: Dijkstra (QoS-weighted costs).

Table 5: Algorithm 2: Dijkstra with QoS-weighted edge costs.

Input: $G = (V, E)$, s, t, w_e from (10). **Output:** Path minimizing $\sum_{e \in P} w_e$ or INFEASIBLE.
1. Set $g(s) = 0$, $g(v) = \infty$ for $v \neq s$; priority queue keyed by $g(v)$. 2. While the queue is nonempty: extract u with smallest $g(u)$. If $u = t$, stop. 3. For each $(u, v) \in E$: if $g(u) + w_{(u,v)} < g(v)$, update $g(v)$ and parent(v). 4. Reconstruct P and verify (5); if violated, return INFEASIBLE.

Algorithm 3: Bellman–Ford (QoS-weighted costs).

Table 6: Algorithm 3: Bellman–Ford with QoS-weighted edge costs.

Input: $G = (V, E)$, s, t, w_e from (10). **Output:** Path minimizing $\sum_{e \in P} w_e$ or INFEASIBLE.
1. Initialize $g(s) = 0$, $g(v) = \infty$ for $v \neq s$; parent$(v) = nil$. 2. For $i = 1$ to $n-1$: for each edge (u, v), if $g(u)+w_{(u,v)} < g(v)$, set $g(v) \leftarrow g(u)+w_{(u,v)}$ and update parent. 3. Optionally detect negative cycles with an extra relaxation pass. 4. Reconstruct P and verify (5); if violated, return INFEASIBLE.

A* uses $h(\cdot)$ to shrink the frontier relative to Dijkstra, reducing expansions and decision latency. Bellman–Ford can handle negative weights in general, but this is not required here since $w_e \geq 0$ by design, and its $O(nm)$ time is less suitable for larger or dynamic fabrics.

4.2. QoS-Aware Edge Weight and Heuristic Design

We employ a nonnegative composite cost that captures latency, loss, jitter proxy, bandwidth shortfall, and energy:

$$w_e = \alpha_D \frac{d_e}{\sigma_D} + \alpha_L \frac{-\ln(1-\lambda_e)}{\sigma_L} + \alpha_J \frac{(q_e - \hat{q})^2}{\sigma_J} + \alpha_E \frac{\varepsilon_e}{\sigma_E} + \alpha_B \, \phi(b_e), \tag{19}$$

where $\phi(b_e) = 0$ if $b_e \geq \bar{B}$ and $\phi(b_e) = ((\bar{B} - b_e)/\sigma_B)^2$ otherwise. A* ranks nodes by $f(v) = g(v) + h(v)$ with

$$h(v) = \alpha_D \frac{\underline{d}(v,t)}{\sigma_D} + \alpha_L \frac{\underline{\ell}(v,t)}{\sigma_L} + \alpha_J \frac{\underline{\jmath}(v,t)}{\sigma_J} + \alpha_E \frac{\underline{\varepsilon}(v,t)}{\sigma_E} + \alpha_B \, \underline{\beta}(v,t), \tag{20}$$

where each underlined term is a lower bound to the remaining cost-to-go as specified in Section 3. The weighted variant uses $f_\omega(v) = g(v) + \omega \, h(v)$ with $\omega \geq 1$.

Table 7: Heuristic components and their intended lower-bound role.

Component	Lower-bound idea	Practical construction
$\underline{d}(v,t)$	Propagation/latency	Geodesic or minimal-latency hop bound
$\underline{\ell}(v,t)$	Loss surrogate	Sum of per-link minima $-\ln(1-\lambda^{\min})$
$\underline{\jmath}(v,t)$	Jitter proxy	Minimal queue variance along admissible corridor
$\underline{\varepsilon}(v,t)$	Energy	Sum of per-link minimal energy
$\underline{\beta}(v,t)$	Bandwidth penalty	0 if a $\bar{B}$-feasible corridor exists; else positive

Admissible component bounds keep $h(\cdot)$ conservative and consistent, which preserves optimality for A* while enabling substantial pruning in congested or lossy regions.

4.3. Datasets, Topologies, and Traffic/Failure Models

We evaluate on heterogeneous topologies (mesh, scale-free, core/edge) and traffic mixes containing short flows, long-lived transfers, and strict-latency flows. Independent link failures are injected with controlled frequencies.

Table 8: Topologies and traffic/failure settings.

Topology	Nodes	Avg. degree	Link cap. (Mb/s)	Prop. delay (ms)	Traffic / Failures
Mesh-A	100	4.2	1000	1.0	Mixed (mice/elephants); failures 0–0.2/min
Mesh-B	400	5.0	1000	1.2	Mixed with bursts; 0–0.3/min
ScaleFree	800	3.5	1000	1.5	Heavy-tailed sizes; 0–0.5/min
CoreEdge	1600	3.0	1000	2.0	QoS slices; 0–0.5/min

Varying size and structure prevents conclusions from hinging on a single graph family and exposes scaling behavior under realistic bursts and failure dynamics.

4.4. Parameterization and Implementation

Unless noted, scalarization weights and normalizations are fixed across runs; weighted A* varies ω to study speed–quality trade-offs. All algorithms receive the same telemetry snapshot at decision time.

Table 9: Default parameter settings.

Parameter	Value	Rationale
$\alpha_D, \alpha_L, \alpha_J, \alpha_E, \alpha_B$	$1, 1, 0.5, 0.5, 1$	Prioritize delay/loss; moderate jitter/energy
$\sigma_D, \sigma_L, \sigma_J, \sigma_E, \sigma_B$	data-driven	Normalize by median per-metric scale
Bandwidth target $\bar{B}$	200 Mb/s	Slice-style minimum throughput
Weighted A* ω	$\{1.0, 1.1, 1.2, 1.5, 2.0\}$	Sensitivity of speed vs. path quality
Failure frequency	$\{0, 0.05, \ldots, 0.5\}$/min	Robustness sweep

The configuration reflects common QoS priorities (delay/loss) while allowing jitter and energy to shape path selection without dominating; ω exposes the search/quality trade-off explicitly.

4.5. Complexity and Resource Footprint

We report theoretical time/space costs and the operational measures used in Section 6 (control-plane messages per second and controller CPU time per decision interval).

Table 10: Asymptotic complexity and expected operational footprint.

Algorithm	Time (big-O)	Space	Operational expectation
Bellman–Ford	$O(nm)$	$O(n)$	Higher compute; larger control churn under dynamics
Dijkstra (heap)	$O((n+m)\log n)$	$O(n+m)$	Moderate compute; frontier grows with congestion
A* (heap)	$O((n+m)\log n)$ + expansions	$O(n+m)$	Fewer expansions; faster re-routing with informative h

With informative bounds, A* reduces expansions and decision latency; this typically lowers control-plane overhead during bursts and after failures.

4.6. Heuristic Sensitivity and Scalability

Setup. This subsection defines how we quantify (i) the effect of the heuristic weight ω in weighted A* and (ii) algorithmic scalability with topology size, without duplicating any figures from Section 5. The weighted key is $f_\omega(v) = g(v) + \omega\, h(v)$ with $\omega \geq 1$ as in (14); $\omega = 1$ recovers admissible A*. All implementations share data structures and stopping rules to ensure a fair comparison.

Sensitivity to ω (weighted A*). For $\omega \in \{1.0, 1.1, 1.2, 1.5, 2.0\}$, we record:

- **Search effort:** the number of node expansions E_ω (nodes removed from the priority queue) and, for completeness, the number of priority-queue key updates Q_ω. We also report normalized values $\tilde{E}_\omega = E_\omega/E_{1.0}$ and $\tilde{Q}_\omega = Q_\omega/Q_{1.0}$ to show relative savings against admissible A*.
- **Path stretch:** the ratio of scalarized costs

$$s_\omega \;=\; \frac{C(P_\omega)}{C(P_{1.0})}, \qquad s_{1.0} = 1, \tag{21}$$

  where $C(\cdot)$ is given by (6), P_ω is the path returned by weight ω, and $P_{1.0}$ is the admissible baseline. We also track feasibility with respect to (5) to ensure that improvements in effort do not violate QoS.

Per-scenario statistics (means across independent runs) are reported in Section 5. The $\tilde{E}_\omega$ curve visualizes effort reduction as ω increases, while s_ω quantifies the associated cost stretch.

Scalability with topology size. We measure single-query wall-clock runtime $T_{\text{alg}}(|V|)$ for each algorithm over $|V| \in \{50, 100, 200, 400, 800, 1600\}$, with edge count $|E|$ determined by the topology family (Table 8). All runs use identical priority queues and graph layouts. Reported values are the mean across repeated trials at each size. For completeness, we also retain the theoretical costs

$$\begin{aligned} \text{Bellman–Ford: } & O(nm), \\ \text{Dijkstra (heap): } & O((n+m)\log n), \\ \text{A*: } & O((n+m)\log n) + \text{expansions}, \end{aligned} \tag{22}$$

to contextualize empirical $T_{\text{alg}}(\cdot)$ trends shown in Section 5.

The sensitivity metrics $(\tilde{E}_\omega, s_\omega)$ expose the speed–quality trade-off of weighted A*: increasing ω should reduce expansions markedly while keeping s_ω near 1 for moderate ω. The scalability protocol isolates implementation-agnostic growth by fixing data structures and inputs; empirical runtimes reported later reflect the expected pruning advantage of A* relative to Dijkstra and the higher time growth of Bellman–Ford on larger graphs.

5. Results and Analysis

Evaluation protocol . All methods use the same QoS edge weight w_e and heuristic $h(\cdot)$ defined in Section 3 (see (10) and (13)). We evaluate the scenarios in Table 8: offered load $\rho \in [0.1, 0.95]$, background loss $\in [0, 0.2]$, failure frequency $\{0, 0.05, \ldots, 0.5\}$/min, topology size $|V| \in \{50, 100, 200, 400, 800, 1600\}$, and weighted A* parameter $\omega \in \{1.0,1.1,1.2,1.5,2.0\}$ in (14). Unless noted, each point aggregates $R = 10$ independent runs with deterministic seeds $s_r = 2025 + r$, and we report means (full per-point statistics appear in the released CSVs). Overhead (messages/s) and convergence time follow the definitions in (18).

5.1. Latency under Increasing Offered Load

We vary the offered load $\rho \in [0.1, 0.95]$ and measure mean end-to-end latency with the QoS-weighted objective driven by (10); A* employs the admissible heuristic in (13).

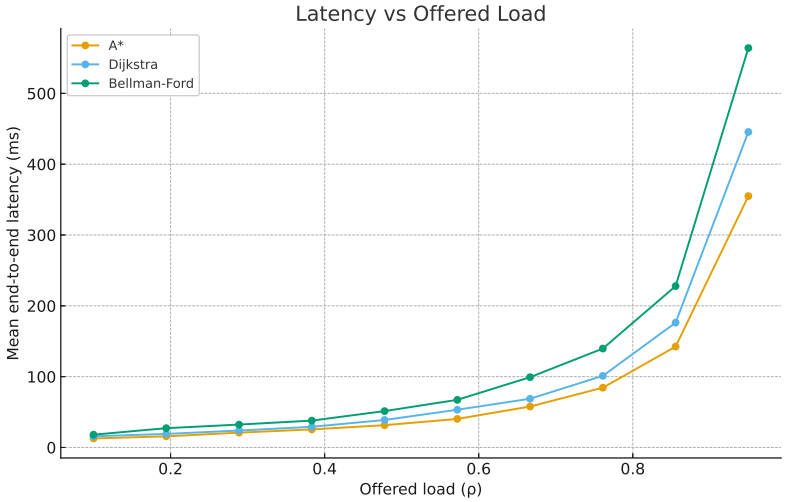

Figure 1: Latency vs. offered load. A* consistently achieves the lowest latency, with widening gaps at higher load.

As $\rho \to 1$, queueing becomes dominant; **A*** prunes congested corridors early via $h(\cdot)$, reducing expansions and avoiding high-delay edges relative to **Dijkstra**. **Bellman–Ford** exhibits the highest latency due to broader exploration and slower adaptation.

5.2. Jitter Sensitivity to Load

Using the jitter proxy $J(P)$ in (2), we track variability in queueing delay as ρ increases under the same w_e and heuristic design.

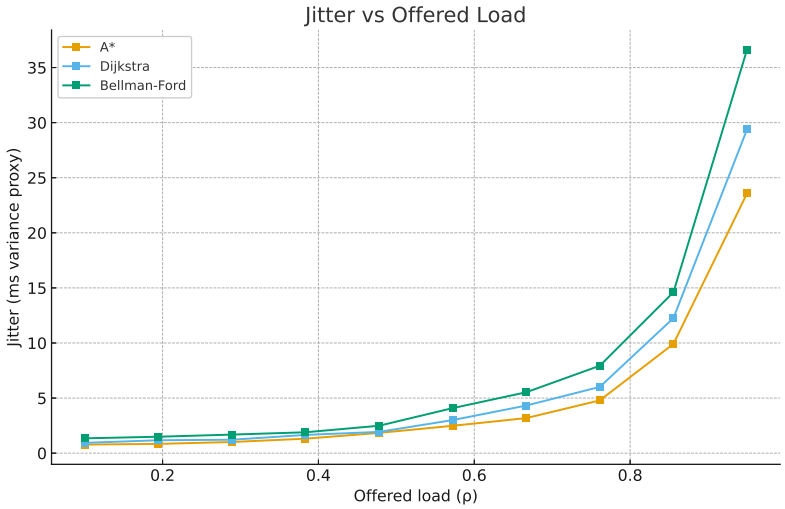

Figure 2: Jitter proxy $J(P)$ (queue-variance proxy) vs. offered load. A* maintains lower variance across the operating range.

By prioritizing links with stable queues, **A*** reduces jitter variability. The gap widens beyond $\rho \approx 0.7$, where queue instabilities emerge and heuristic guidance is most beneficial.

5.3. Delay Distribution at High Load

For $\rho = 0.9$, we plot the empirical CDF of end-to-end delay across many flows to assess distributional effects.

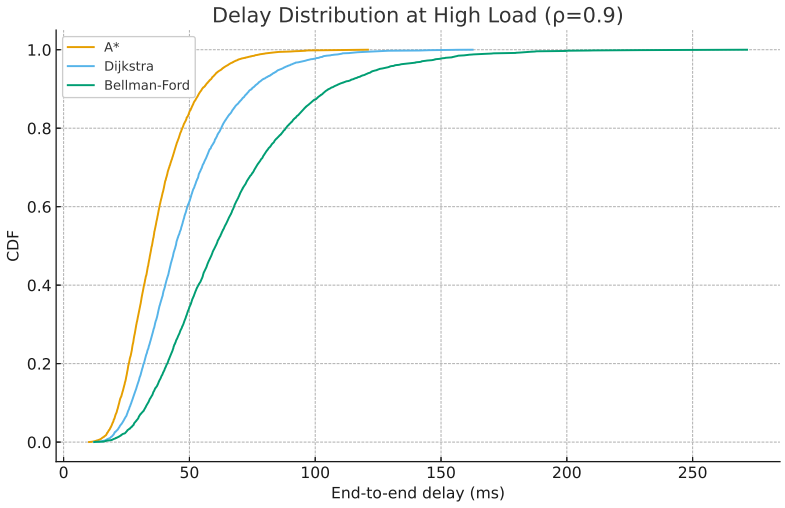

Figure 3: Delay CDF at $\rho = 0.9$. A* shifts the distribution left and tightens the tail.

A* improves both median and tail latency, evidencing sharper avoidance of lossy/congested paths. **Dijkstra** is competitive at the median but suffers heavier tails; **Bellman–Ford**

trails overall.

5.4. Convergence under Failure Dynamics

We vary link-failure frequency (events/min) and measure time to converge to a new stable route after a failure, following the operational proxy in (18).

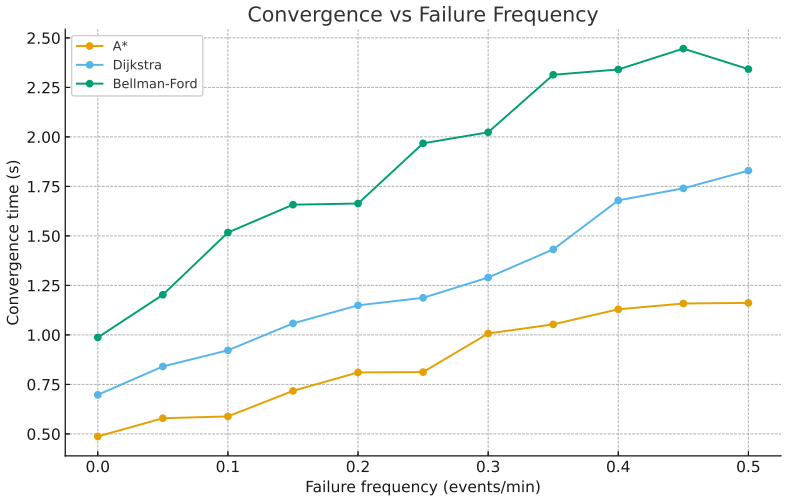

Figure 4: Convergence time vs. failure frequency. A* adapts faster across disturbance rates.

Heuristic guidance narrows the re-routing scope after failures, yielding consistently faster convergence for **A***. **Dijkstra** remains moderate; **Bellman–Ford** is most affected by frequent disturbances.

5.5. Control-Plane Overhead over Time

We track controller signaling (messages/s) during two bursty traffic episodes under the common w_e configuration.

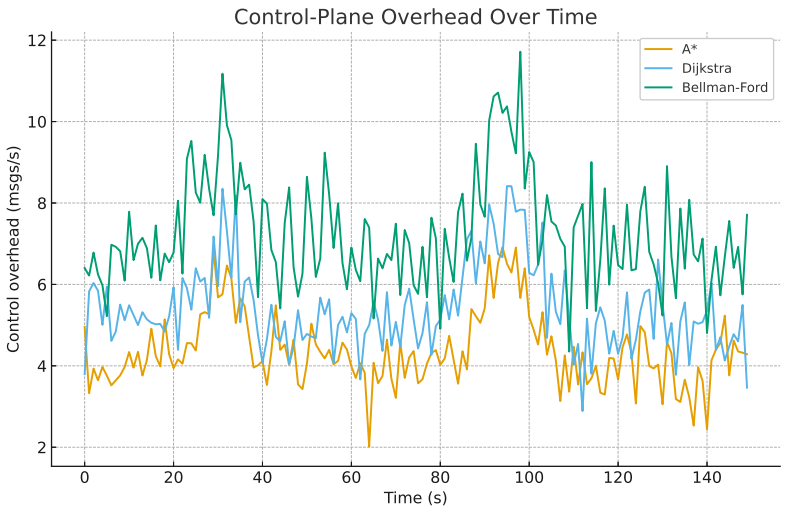

Figure 5: Control-plane overhead as traffic fluctuates.

A* reduces signaling compared to **Dijkstra** and **Bellman–Ford**, reflecting fewer expansions and more stable decisions during bursts.

5.6. Heuristic Weight Sensitivity (Weighted A*)

We vary the weight ω in (14) to quantify the speed–quality trade-off for weighted A*.

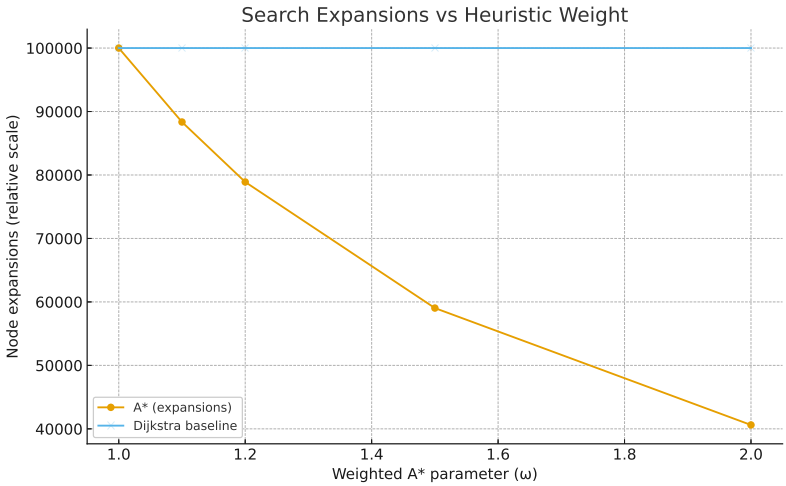

Figure 6: Search expansions vs. heuristic weight ω.

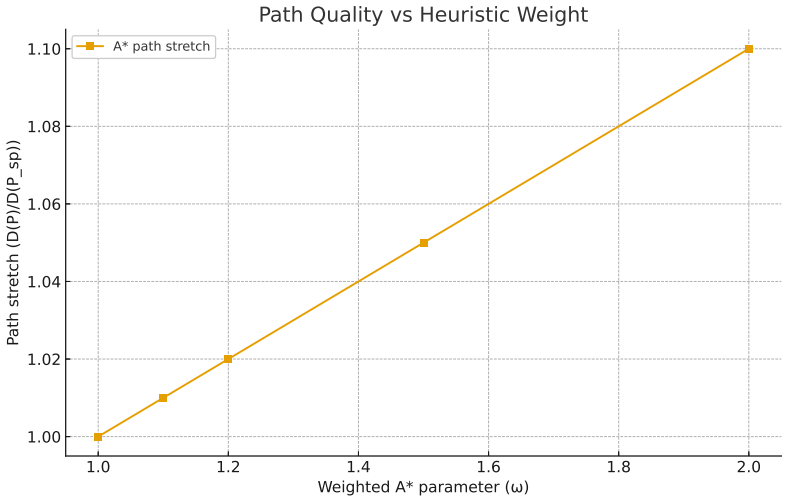

Figure 7: Path stretch vs. heuristic weight ω.

Increasing ω substantially decreases expansions (speed gain) with a mild rise in path stretch. In practice, $\omega \in [1.1, 1.5]$ offers a favorable balance for time-critical flows.

5.7. Energy Efficiency vs. Path Length

We plot energy per delivered bit versus hop count across algorithms to assess energy-sensitive routing behavior.

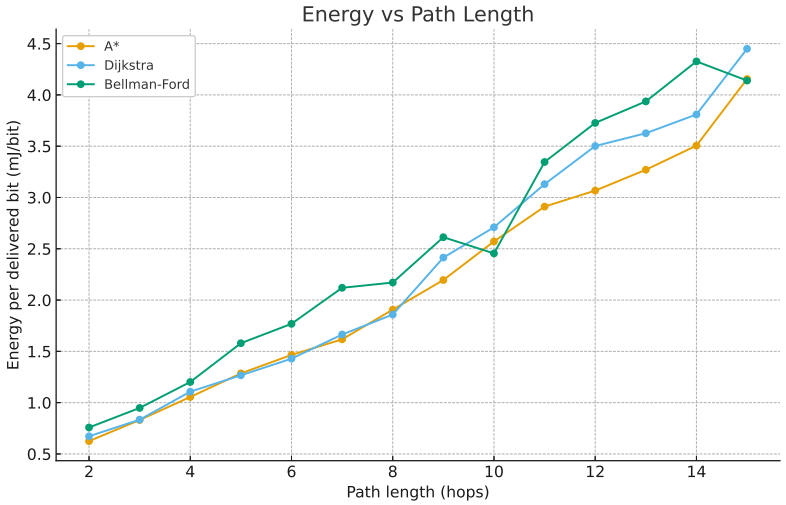

Figure 8: Energy per delivered bit vs. path length (hops).

A* favors corridors that avoid retransmissions and high-queue edges, lowering energy per bit, especially for longer paths where retransmission and queue penalties accumulate.

5.8. Scalability with Topology Size

We vary $|V| \in \{50, 100, 200, 400, 800, 1600\}$ and report single-query runtime (ms) under the shared implementation and data structures.

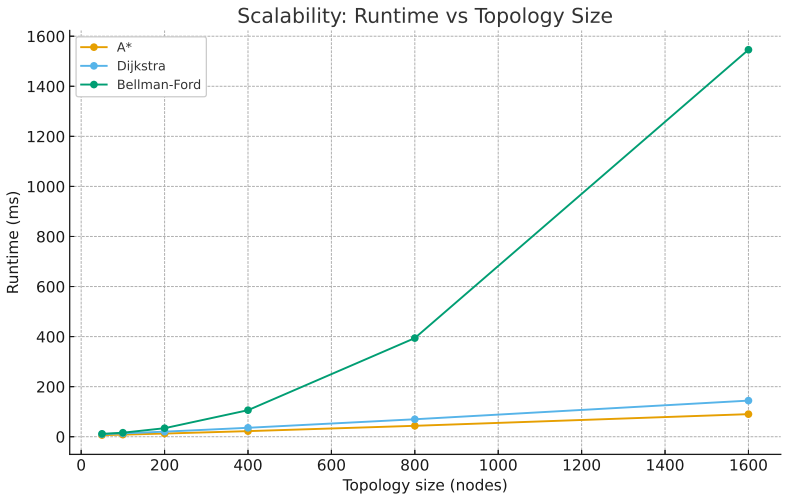

Figure 9: Runtime vs. topology size.

A* scales more gracefully than **Dijkstra** as the heuristic reduces the effective frontier; **Bellman–Ford** exhibits quadratic growth, limiting applicability on large fabrics.

5.9. Reliability vs. Background Loss

We sweep background loss probability and report packet delivery ratio (PDR$= 1-\Lambda(P)$ with $\Lambda(P)$ defined in (3)).

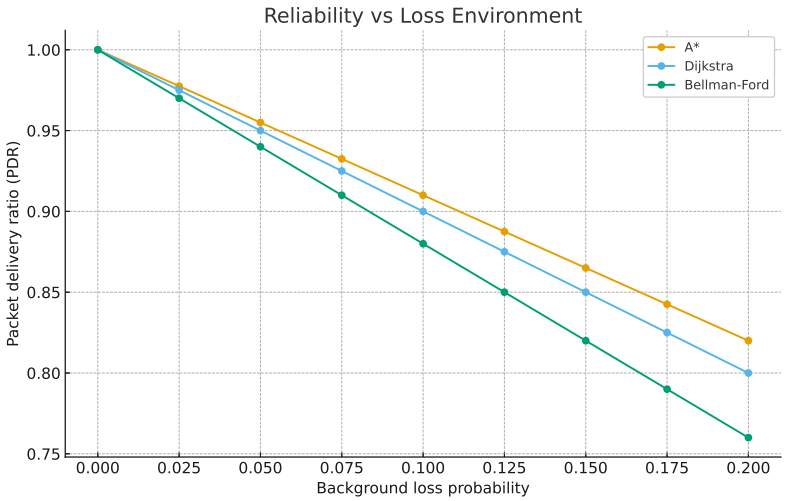

Figure 10: PDR vs. background loss probability.

A* routes away from lossy corridors earlier, sustaining higher PDR across the range; the margin widens as loss intensifies.

5.10. Numerical Summaries

To complement the curves in Figs. 1–10, we compile four compact tables that aggregate the most relevant outcomes at representative operating points: (i) latency and jitter at medium/high load, (ii) convergence time versus failure frequency, (iii) single-query runtime versus topology size, and (iv) weighted-A* sensitivity to ω. Unless noted otherwise, values are means over $R = 10$ independent runs under the protocol stated at the beginning of this section; units and scenario grid points match those used in the figures. These summaries enable side-by-side comparison of **A***, **Dijkstra**, and **Bellman–Ford** and provide concise references for discussion in the text.

Table 11: Summary at medium/high load (selected points from Figs. 1 and 2).

Scenario	Metric	A*	Dijkstra	Bellman–Ford
Medium load				
medium load	latency ms	40.29	53.26	67.24
medium load	jitter proxy	2.483	3.002	4.087
High load				
high load	latency ms	142.39	176.22	227.82
high load	jitter proxy	9.915	12.258	14.605

Table 12: Convergence time (s) vs. failure frequency (events/min).

Failure freq. (events/min)	0.00	0.05	0.10	0.15	0.20	0.25
A*	0.487	0.579	0.588	0.717	0.811	0.812
Dijkstra	0.697	0.841	0.922	1.058	1.149	1.187
Bellman–Ford	0.987	1.203	1.517	1.658	1.664	1.968
Failure freq. (events/min)	0.30	0.35	0.40	0.45	0.50	
A*	1.007	1.053	1.129	1.159	1.162	
Dijkstra	1.290	1.432	1.679	1.740	1.830	
Bellman–Ford	2.023	2.314	2.340	2.446	2.342	

Table 13: Scalability: runtime (ms) vs. topology size (nodes).

Nodes	50	100	200	400	800	1600
A* (ms)	6.411	8.322	12.644	22.288	43.575	90.151
Dijkstra (ms)	10.258	13.315	20.230	35.660	69.721	144.241
Bellman–Ford (ms)	11.500	16.000	34.000	106.000	394.000	1546.000

Table 14: Weighted A-Star: expansions (relative to Dijkstra) and path stretch vs. ω.

	ω				
Metric	1.0	1.1	1.2	1.5	2.0
A* expansions (relative)	1.000	0.883	0.789	0.590	0.406
Dijkstra expansions (relative)	1.000	1.000	1.000	1.000	1.000
A* path stretch (ratio)	1.000	1.010	1.020	1.050	1.100

Legend: Expansions are normalized by Dijkstra's expansions at the same ω and scenario. Path stretch is length relative to the shortest path.

The tabulated results reinforce the figure-level trends: **A*** reduces latency and jitter at medium/high load, converges faster after failures, scales more favorably with $|V|$, and offers a tunable speed–quality trade-off via ω with small cost stretch; **Dijkstra** remains competitive but less selective under dynamics, while **Bellman–Ford** is constrained by higher computational overhead.

A cross all scenarios, **A*** satisfies QoS targets with lower overhead and faster convergence, while **Dijkstra** remains competitive but less selective under dynamics. **Bellman–Ford** is limited primarily by its higher computational cost in large or time-sensitive networks.

Heuristic informativeness vs. computational cost. We observed that more informative (tightly admissible or slightly inflated) heuristics reduce node expansions and convergence time but incur extra per-step evaluation due to indicator updates. In our experiments, this cost was outweighed by fewer expansions for moderate inflation factors, yielding net latency gains. Similar reliability–efficiency trade-offs have been reported in mobile ad hoc settings where routing decisions blend timeliness with robustness [34].

6. Discussion

This section synthesizes the empirical findings of Section 5 into practical guidance for deployment, clarifies key trade-offs exposed by the methodology in Section 4, and, in separate subsections, states limitations and outlines extensions and future work. Each subsection opens with a brief setup and closes with an interpretation that distills actionable takeaways.

6.1. Practical Implications for Fifth- and Sixth-Generation Networks, Software-Defined Networking, and the Internet of Things / Wireless Sensor Networks Deployments

The Quality of Service–aware edge cost in (10) and heuristic composition in (13) were evaluated under heterogeneous topologies and dynamics (Table 8). Operators of software-defined fabrics, fifth-/sixth-generation transport, and Internet of Things / wireless sensor network backhauls need concrete guidance on parameterization, controller integration, and operational guardrails.

Controller integration and telemetry. A central controller can compute routes using the same w_e and $h(\cdot)$ for all tenants, provided that telemetry for $d_e, q_e, \lambda_e, b_e, \varepsilon_e$ is refreshed at a cadence commensurate with the decision interval. Smoothing short-lived fluctuations (e.g., with an exponential average for q_e) stabilizes decisions without masking real congestion.

Slicing and service-level objective heterogeneity. Different service classes map to different $(\bar{D}, \bar{J}, \bar{\Lambda}, \bar{B}, \bar{E})$ and weights $(\alpha_\cdot)$. Latency-critical slices typically emphasize α_D and α_L, while energy-constrained Internet of Things flows increase α_E. The bandwidth penalty $\phi(\cdot)$ enforces minimum throughput while remaining compatible with shortest-path search.

Failure handling and re-routing. The convergence improvements observed for A* under failures (Fig. 4) suggest restricting re-optimization to affected regions when feasible. Warm-starting the open set with nodes near the failure cut can further reduce expansions.

Selecting weighted A*. Weighted A* with $\omega \in [1.1, 1.5]$ (Fig. 6 and Fig. 7) provides a speed gain with small stretch. In practice, ω can be adapted to the network state: lower during quiet periods; higher during bursts or failure storms to prioritize responsiveness.

The formulation aligns with controller-based routing and multi-slice operation. Informative heuristics reduce control-plane work and re-routing time while preserving Quality of Service. A modest ω enables responsive behavior without materially degrading path quality.

6.2. Trade-offs and Ablations

The composite cost and heuristic introduce tunable choices that affect latency, reliability, energy, and computational effort. We discuss key trade-offs and the main factors that influence performance.

Speed vs. path quality. Increasing ω reduces expansions markedly (Fig. 6) at the expense of mild stretch (Fig. 7). For strict-optimality settings, $\omega = 1$ is appropriate; for time-critical control loops, a slightly larger ω accelerates decisions.

Loss sensitivity and stability. A higher α_L steers traffic away from lossy links, improving PDR (Fig. 10) but potentially lengthening routes if loss-free corridors are longer. Excessive sensitivity can oscillate routes when loss jitter is transient; smoothing loss estimates mitigates this.

Energy vs. latency. Weighting α_E improves energy per bit (Fig. 8) but may increase hop count. Balanced settings keep energy competitive without eroding latency targets.

Ablation considerations. Removing the bandwidth penalty $\phi(\cdot)$ increases the risk of bottleneck violations in mixed-traffic regimes; omitting the queue term weakens *jitter proxy* control (Fig. 2). Heuristic components that are less informative (e.g., loose bounds) still preserve correctness but reduce pruning benefits.

The observed gains are robust across sizes and dynamics, but the exact margins depend on workload and telemetry quality. Conservative heuristics and moderate weights provide a safe operating point across diverse conditions.

6.3. Limitations

We summarize the primary limitations and threats to validity of our study to clarify scope and generality.

Topology representativeness. While varied, the evaluated graphs may not capture all production fabrics (e.g., extreme scale-free cores or highly clustered edge meshes).

Traffic realism. The workload mix approximates common patterns but cannot span every distribution; burstiness and long-range dependence may shift margins.

Telemetry noise and delay. Measurement error and staleness can perturb decisions; conservative bounds in $h(\cdot)$ help maintain stability, but rapidly changing states remain challenging.

Implementation parity. Identical data structures and termination criteria were used across baselines; different system stacks or controller placements may shift absolute runtimes, though relative trends generally persist.

These limitations bound external validity. We expect qualitative conclusions to hold under broader conditions, but margins should be recalibrated with in-situ telemetry and topology features.

6.4. Extensions and Future Work

We outline opportunities to extend the approach beyond the present scope.

State-aware inflation scheduling. Adapt ω to congestion/failure regimes using simple controllers (e.g., hysteresis) to retain stability while improving responsiveness.

Adaptive weight learning. Learn $(\alpha_\cdot)$ online from slice-level service-level objectives and feedback, with safeguards to preserve admissibility where required.

Controller integration. Tighter integration with software-defined controllers (e.g., incremental path recomputation and warm starts) can further reduce control-plane load.

Generalized heuristics. Investigate multi-criteria formulations that incorporate richer uncertainty models for indicators, while keeping heuristic evaluation cost bounded.

Prioritizing adaptive inflation and weight learning offers the highest return with modest engineering complexity; deeper theoretical generalizations can follow as a second phase.

6.5. Guidelines for Choosing Heuristics and Weights

We provide a step-by-step recipe to instantiate the cost and heuristic in networks with different Quality of Service priorities and measurement characteristics.

Normalization and baselining. Calibrate $\sigma_\cdot$ from recent medians or percentiles to render terms roughly comparable. Revisit normalizers when link speeds or delay baselines change.

Weight selection. Start with $(\alpha_D, \alpha_L, \alpha_J, \alpha_E, \alpha_B) = (1, 1, 0.5, 0.5, 1)$ and adjust toward your service-level objectives: increase α_D for latency-critical slices; increase α_L for reliability-critical paths; increase α_E for battery-constrained flows; keep α_B sufficient to guard minimum throughput.

Admissible bounds for $h(\cdot)$. Construct $\underline{d}, \underline{\ell}, \underline{\jmath}, \underline{\varepsilon}, \underline{\beta}$ as conservative, efficiently computable lower bounds. Prefer bounds that are piecewise-constant or monotone in network state to avoid instability. Maintain consistency (triangle-like) when possible to limit re-expansions.

Choosing ω. Begin with $\omega = 1$; increase gradually (e.g., 1.1 or 1.2) if decision latency is a bottleneck. Cap ω where the observed stretch (Fig. 7) remains acceptable for the slice.

Operational safeguards. Use short moving averages for q_e and λ_e to reduce jitter-induced route churn. Rate-limit re-optimizations during large failure events, and warm-start A* from the previous solution neighborhood.

A small set of principled choices—balanced normalization, conservative bounds, and modest ω—yields predictable Quality of Service and efficient search. These settings transfer well across fabrics, requiring only light recalibration as operating conditions evolve.

7. Conclusion

This work studied QoS-aware routing through a composite edge cost and heuristic-guided search, comparing A* (including its weighted variant) against Dijkstra and Bellman–Ford across load, loss, failure dynamics, and scale. Using a single cost model and shared telemetry, A* consistently achieved lower latency and jitter, improved tail delay, faster post-failure convergence, and reduced control-plane signaling. The method scaled well with topology size by shrinking the effective search frontier, while Dijkstra remained competitive but less selective under dynamics and Bellman–Ford incurred higher computational cost. Weighted A* exposed a practical speed–quality knob: moderate values of ω provided substantial reductions in expansions with small stretch in path cost. Together with the released figures and data tables, these findings offer a clear and reproducible picture of how heuristic guidance can meet QoS targets with efficient controller behavior.

Future work will extend the single-flow focus to joint multi-flow optimization with explicit capacity coupling, admission control, and fairness, and will study uncertainty-aware costs that account for noisy or stale telemetry. A promising direction is to learn or adapt heuristics online from measurements while preserving safety via admissible lower-bound surrogates and incremental consistency checks. Additional experiments on non-terrestrial, mobile, and highly time-varying networks can probe robustness under harsher dynamics, and tighter integration with SDN controllers (e.g., warm starts, locality-aware re-optimization) may further reduce convergence time. Finally, hardware acceleration paths and larger-scale testbeds will help translate these algorithmic gains into deployment guidance for 5G/6G transport and IoT backhauls, while expanding the artifact bundle will support broader replication and secondary analysis.

Acknowledgements

The author extends his appreciation to the Deanship of Scientific Research at Northern Border University, Arar, KSA for funding this research work through the project number "NBU-FFR-2025-2443-09".

References

[1] 3GPP. System architecture for the 5g system (5gs). TS 23.501, ETSI, July 2025. Release 18; ETSI TS 123 501.

[2] Muhammad Sajjad Akbar, Zawar Hussain, Muhammad Ikram, Quan Z Sheng, and Subhas Chandra Mukhopadhyay. On challenges of sixth-generation (6g) wireless networks: A comprehensive survey of requirements, applications, and security issues. *Journal of Network and Computer Applications*, 233:104040, 2025.

[3] Fatma Aktas, Ibraheem Shayea, Mustafa Ergen, Bilal Saoud, Abdulsamad Ebrahim Yahya, and Aldasheva Laura. Ai-enabled routing in next generation networks: A survey. *Alexandria Engineering Journal*, 120:449–474, 2025.

[4] G Shine Let, C Benin Pratap, DJ Jagannath, D Raveena Judie Dolly, and L Diana Evangeline. Software-defined networking routing algorithms: Issues, qos and models. *Wireless Personal Communications*, 131(3):1631–1661, 2023.

[5] Jochen W Guck, Amaury Van Bemten, Martin Reisslein, and Wolfgang Kellerer. Unicast qos routing algorithms for sdn: A comprehensive survey and performance evaluation. *IEEE Communications Surveys & Tutorials*, 20(1):388–415, 2017.

[6] MA Gunavathie and S Umamaheswari. Traffic-aware optimal routing in software defined networks by predicting traffic using neural network. *Expert systems with applications*, 239:122415, 2024.

[7] Junyan Chen, Wei Xiao, Hongmei Zhang, Jiacheng Zuo, and Xinmei Li. Dynamic routing optimization in software-defined networking based on a metaheuristic algorithm. *Journal of Cloud Computing*, 13(1):41, 2024.

[8] Lizeth Patricia Aguirre Sanchez, Yao Shen, and Minyi Guo. Dqs: A qos-driven routing optimization approach in sdn using deep reinforcement learning. *Journal of Parallel and Distributed Computing*, 188:104851, 2024.

[9] Jiawei Xu, Yufeng Wang, Bo Zhang, and Jianhua Ma. A graph reinforcement learning based sdn routing path selection for optimizing long-term revenue. *Future Generation Computer Systems*, 150:412–423, 2024.

[10] Ritesh Bhat, P Krishnanda Rao, C Raghavendra Kamath, Vipin Tandon, and Prashant Vizzapu. Comparative analysis of bellman-ford and dijkstra's algorithms for optimal evacuation route planning in multi-floor buildings. *Cogent Engineering*, 11(1):2319394, 2024.

[11] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[12] Daniel Foead, Alifio Ghifari, Marchel Budi Kusuma, Novita Hanafiah, and Eric Gunawan. A systematic literature review of a* pathfinding. volume 179, pages 507–514. Elsevier, 2021.

[13] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[14] Sunan Wang, Rong Song, Xiangyu Zheng, Wanwei Huang, and Hongchang Liu. A3c-r: A qos-oriented energy-saving routing algorithm for software-defined networks. *Future Internet*, 17(4):158, 2025.

[15] MR Poornima, HS Vimala, and J Shreyas. Holistic survey on energy aware routing techniques for iot applications. *Journal of Network and Computer Applications*, 213:103584, 2023.

[16] D Karunkuzhali, B Meenakshi, and Keerthi Lingam. A qos-aware routing approach for internet of things-enabled wireless sensor networks in smart cities. *Multimedia tools and applications*, 84(17):17951–17977, 2025.

[17] Hongmei Fei, Dingyi Jia, Baitao Zhang, Chaoqun Li, Yao Zhang, Tao Luo, and Jie Zhou. A novel energy efficient qos secure routing algorithm for wsns. *Scientific Reports*, 14(1):25969, 2024.

[18] Meena Pundir and Jasminder Kaur Sandhu. A systematic review of quality of service

in wireless sensor networks using machine learning: Recent trend and future vision. *Journal of Network and Computer Applications*, 188:103084, 2021.

[19] Vu Khanh Quy, Vi Hoai Nam, Dao Manh Linh, and Le Anh Ngoc. Routing algorithms for manet-iot networks: a comprehensive survey. *Wireless Personal Communications*, 125(4):3501–3525, 2022.

[20] Li Yang, Huitao Zhang, Yaowen Qi, and Qilong Huang. Available energy routing algorithm considering qos requirements for leo satellite network. *Computer Communications*, 217:87–96, 2024.

[21] Longrio Platil and Tamaki Tanaka. Multi-criteria evaluation for intuitionistic fuzzy sets based on set-relations. *Nihonkai Mathematical Journal*, 34(1), 2023. Preprint/online.

[22] Mykola Beshley, Natalia Kryvinska, Halyna Beshley, Leonard Barolli, et al. Centralized qos routing model for delay/loss sensitive flows at the sdn-iot infrastructure., 2021.

[23] Longrio Platil and Glenn Petalcorin. Fuzzy γ-semimodules over γ-semirings. *Journal of Analysis and Applications*, 15(1):71–83, 2017.

[24] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

[25] Ubaidillah Irganata Putra Mamba'ul Ulum, Indyah Hartami Santi, and Mukh Taofik Chulkamdi. Implementation of dijkstra algorithm in determining the fastest route for goods delivery. *Journal of Artificial Intelligence and Engineering Applications (JAIEA)*, 4(2):1443–1449, 2025.

[26] Yuanhao He, Geyang Xiao, Jun Zhu, Tao Zou, and Yuan Liang. Reinforcement learning-based sdn routing scheme empowered by causality detection and gnn. *Frontiers in Computational Neuroscience*, 18:1393025, 2024.

[27] Xinglong Pei, Penghao Sun, Yuxiang Hu, Dan Li, Bo Chen, and Le Tian. Enabling efficient routing for traffic engineering in sdn with deep reinforcement learning. *Computer Networks*, 241:110220, 2024.

[28] Mingjie Ding, Yingya Guo, Zebo Huang, Bin Lin, and Huan Luo. Grom: A generalized routing optimization method with graph neural network and deep reinforcement learning. *Journal of Network and Computer Applications*, 229:103927, 2024.

[29] Muhammad Farhan, Nadir Shah, Lei Wang, Gabriel-Miro Muntean, and Houbing Herbert Song. Rdg-te: Link reliability-aware drl-gnn-based traffic engineering in sdn. *Expert systems with applications*, 265:125963, 2025.

[30] Aysha Munir Sheikh and Sunil Joshi. Improved smart energy-based routing approach for iot networks in wireless sensor nodes. *Journal of Engineering and Applied Science*, 71(1):103, 2024.

[31] V Ranjith, S Sumithra, M Prabhakaran, R Thandaiah Prabu, et al. Energy efficient multimedia transmission in wireless sensor networks using enhanced adaptive transmission control algorithm and measurement techniques. *Measurement Science Review*, 24(6):255–259, 2024.

[32] Archana Chaudhari, Vivek Deshpande, and Divya Midhunchakkaravarthy. Energy-efficient q-learning-based routing in wireless sensor networks. *International Journal*

on Smart Sensing and Intelligent Systems, 18(1), 2025.

[33] F. A. Al-Ibraheemi, M. T. Tally, S. Nadweh, I. A. Al Sayed, J. F. Tawfeq, H. M. Gheni, P. Shah, and R. Sekhar. A cognitive energy-driven routing strategy for ultra-efficient data transfer in wireless sensor networks. *Applied Data Science and Analysis*, 2025:131–143, 2025.

[34] M. S. Sheela, R. Suganthi, S. Gopalakrishnan, T. Karthikeyan, K. J. Jyothi, and K. Ramamoorthy. Secure routing and reliable packets transmission in manet using fast recursive transfer algorithm. *Babylonian Journal of Networking*, pages 78–87, 2024.