CHECKLIST

Graph Algorithms for Network Path Optimization and Routing Checklist

A practical checklist for evaluating graph-based routing decisions, validating path constraints, and reducing churn before production use.

Graph Algorithms for Network Path Optimization and Routing: Production Readiness Checklist

Use this checklist to decide whether graph algorithms are appropriate for a routing problem and to verify that the resulting path is safe, stable, and operationally useful.

1) Define the routing objective

- [ ] The primary objective is clearly stated: latency, hop count, cost, congestion avoidance, policy compliance, failover behavior, security boundary control, or stability.
- [ ] Secondary objectives are identified and ranked.
- [ ] The team agrees on what ?best path? means in this context.
- [ ] The objective is measurable with available data.
- [ ] The routing goal is specific enough to encode as weights or constraints.

2) Confirm the problem is actually a graph problem

- [ ] The network can be modeled as vertices and edges.

- [ ] Edge attributes are available or can be inferred.
- [ ] Path selection depends on measurable properties, not only connectivity.
- [ ] The problem is not better solved by a simple static route table or rule list.
- [ ] The team understands why ad hoc routing decisions are insufficient.

3) Validate the graph model

- [ ] The graph is directed if links or policies are asymmetric.
- [ ] The graph is undirected only if traversal cost is effectively symmetric.
- [ ] Edge weights represent real network behavior or a documented synthetic score.
- [ ] Forbidden links are excluded or assigned effectively infinite cost.
- [ ] Preferred links are encoded consistently.
- [ ] Policy boundaries, trust zones, and segmentation rules are modeled explicitly.
- [ ] The graph reflects the current topology, not an outdated snapshot.

4) Check the edge weight design

- [ ] Edge weights are non-negative if using Dijkstra-style methods.
- [ ] If weights are dynamic, the update source and frequency are defined.
- [ ] If weights are multi-criteria, the scoring method is documented.
- [ ] Latency, loss, jitter, utilization, risk, and policy penalties are weighted intentionally.
- [ ] The cost function is understandable to operators and auditors.
- [ ] Weight changes are traceable to telemetry or policy state.
- [ ] The model avoids mixing unrelated metrics without justification.

5) Choose the right algorithm family

- [ ] Dijkstra-style shortest path is appropriate for non-negative costs.
- [ ] A heuristic search such as A* is justified by a valid admissible heuristic.
- [ ] A constrained or multi-criteria approach is required if more than one operational goal matters.
- [ ] The algorithm is not being forced into a problem it cannot represent well.
- [ ] The expected path count and search space size are understood.
- [ ] The computation cost is acceptable for the routing cadence.

6) Assess operational stability

- [ ] The path will not change too frequently under normal telemetry variation.
- [ ] Hysteresis or dampening is used to prevent route flapping.
- [ ] Recompute thresholds are defined.
- [ ] Small changes in telemetry do not trigger unnecessary path changes.
- [ ] The stability cost of recomputation is understood.
- [ ] A slightly suboptimal but stable path is acceptable when appropriate.

7) Validate against real network conditions

- [ ] The candidate path is checked against recent telemetry.
- [ ] Current congestion state is considered.
- [ ] The path is verified under expected traffic levels, not only in theory.
- [ ] Known transient failures or degraded segments are accounted for.
- [ ] The model is compared with observed latency and utilization.
- [ ] Stale or missing data is detected and handled safely.

8) Enforce policy and security constraints

- [ ] The path does not cross forbidden or untrusted segments.
- [ ] Security segmentation requirements are preserved.
- [ ] Administrative boundaries are respected.
- [ ] Regulatory or organizational routing rules are encoded in the model.
- [ ] The shortest path is not allowed to override policy.
- [ ] If a path violates policy, it is rejected regardless of cost.

9) Check failover and resilience behavior

- [ ] Alternate paths are available when the primary route degrades.
- [ ] Failover behavior is defined before production use.
- [ ] The model handles link or node failure gracefully.
- [ ] Recovery does not create loops or instability.
- [ ] The system can recompute safely after topology changes.
- [ ] The impact of route churn on service reliability is understood.

10) Review implementation trade-offs

- [ ] The team has documented the accuracy-versus-simplicity trade-off.
- [ ] The model is not overly complex for the operational value it delivers.
- [ ] The routing logic is explainable to operators.
- [ ] The algorithm output can be audited after an incident.
- [ ] Local optimization does not undermine global behavior.
- [ ] The system avoids hidden assumptions about static topology or static costs.

11) Prepare production safeguards

- [ ] A validation loop exists before path selection is applied.

- [ ] Unsafe candidate paths can be rejected automatically.
- [ ] The system supports manual override or rollback.
- [ ] Change management is defined for graph updates and weight changes.
- [ ] Monitoring is in place for latency, congestion, route churn, and policy violations.
- [ ] The control plane overhead of recomputation is acceptable.
- [ ] Operators know what triggers recomputation and why.

12) Final go-live questions

- [ ] Does this path satisfy the objective we actually care about?
- [ ] Does it violate any policy, trust, or segmentation rule?
- [ ] Is it stable enough to use without creating unnecessary churn?
- [ ] Is the graph current enough to trust the result?
- [ ] Can the system explain why this path was chosen?
- [ ] Can the system safely fall back if the chosen path degrades?

Quick pass/fail rule

Approve graph-based routing only if all of the following are true:

- [ ] The objective is measurable and aligned with operations.
- [ ] The graph model reflects real constraints.
- [ ] The algorithm fits the weight structure and search space.
- [ ] Policy and security checks are enforced.
- [ ] Stability controls reduce route churn.
- [ ] The result is validated against current or recent telemetry.
- [ ] There is a safe fallback or rollback path.

Notes

Use graph algorithms as a decision engine, not as an autonomous authority. The best operational route is not always the mathematically shortest one; it is the one that best satisfies performance, policy, and stability requirements at the time of execution.