# Route Optimization in a Dynamic Road Network: A Collaborative Web Mapping Platform for Near-Real-Time Traffic Events (Oran, Algeria)

**Bouhadjar MEGUENNI, Algeria**

Key words: dynamic routing, traffic events, collaborative web mapping, PostGIS, OpenStreetMap

## ABSTRACT

In emerging countries such as Algeria, the lack of centralized traffic data limits the effectiveness of emergency services and urban navigation. While global platforms (Google Maps, Waze) rely on massive data flows (FCD), they often lack official validation regarding critical infrastructure closures. This article proposes a collaborative near-real-time routing platform applied to the city of Oran.

The proposed architecture is based on a non-destructive dynamic overlay approach: rather than altering the OpenStreetMap topology, verified incidents are projected onto a dynamic SQL view (ways_live) operated by pgRouting. A "spatial corridor" heuristic using ST_DWithin is implemented to optimize the search space and ensure scalability.

Experimental results on three realistic urban scenarios demonstrate the resilience of the system. Faced with a major closure (Scenario S1), the platform instantly calculates a viable alternative with a minimal detour (+1.6% in distance, +11.9% in time), while maintaining a query latency of less than 500 ms. Unlike commercial "black box" solutions, this open-source framework guarantees routing decisions based on validated data, making it particularly suitable for institutional crisis management.

**Route Optimization in a Dynamic Road Network: A Collaborative Web Mapping Platform for Near-Real-Time Traffic Events (Oran, Algeria)**

**Bouhadjar MEGUENNI, Algeria**

## 1. INTRODUCTION

Urban and suburban road networks are inherently dynamic: road works, accidents, bad weather, and emergency responses constantly change the availability of road segments and local connectivity. In Algeria, access to continuous and consistent traffic data remains limited; in practice, the most reliable information often comes from local actors (security forces, civil protection, and municipal services) and field reports. The scientific and operational challenge is to integrate heterogeneous events in near real time, translate their spatio-temporal impact into a cost function that can be used for routing, and guarantee response times that are compatible with an interactive web mapping application.

### 1.1 Context and Issues

In the context of smart cities and smart mobility, urban navigation must continuously adapt to disruptions such as accidents, road closures, roadworks, and exceptional events. Traditional routing engines (based on a static graph) produce routes that are optimal in theory but often invalid as soon as a closure occurs. The problem is therefore: how can validated road events, with variable validity and criticality, be integrated into a road graph derived from OpenStreetMap, while maintaining latency compatible with interactive web mapping on an urban scale?

### 1.2 Objectives and Contributions

The objective is to design a dynamic routing platform for Algeria, capable of (i) maintaining a large-scale routable graph in PostGIS/pgRouting, (ii) collecting and validating multi-source traffic events, and (iii) recalculating robust routes in near real time. The main contributions are:

– A spatio-temporal traffic event model with validity interval, validation status (draft/validated/disabled), and impact levels (blocking/severe/moderate/info).

– A dynamic non-destructive cost layer: rewriting arc weights on demand (CTE/SQL view) without altering the reference graph, with a temporal penalty decay.

– Multi-step pgr_astar routing (start–vias–end) accelerated by progressive spatial filtering using ST_DWithin corridors and controlled KNN snapping on graph vertices.

– Systematic instrumentation (success/failure logs, latencies, arc popularity) and a benchmarking protocol (RoutingPy) to compare distance/duration with external engines.

## 2. STATE OF THE ART AND RELATED WORK

Dynamic routing in road networks relies on classic shortest path algorithms (Dijkstra, A*) and extensions designed to integrate time-varying costs. At the same time, crowdsourcing approaches (citizen reports) have proven effective in capturing incidents, but pose challenges in terms of reliability and validation. This section situates our contribution at the interface of these two areas.

### 2.1 Dynamic routing and Time-Dependent Shortest Path (TDSP)

Shortest path algorithms on static graphs (Dijkstra, A*) are widely used in mobility. For time-dependent routing, TDSP variants consider that the cost of an arc varies according to the departure time (speed, congestion, availability). In interactive web environments, acceleration strategies such as contraction hierarchies (CH) or hierarchical partitions (CRP) reduce latency, but often assume relatively stable weights. Our approach favors non-destructive rewriting of costs on demand, in order to immediately take into account validated closures, without global recalculation of the graph.

### 2.2 Crowdsourcing, validation and multi-source fusion

Platforms such as Waze use massive reports to detect incidents and congestion, but the data can sometimes be difficult to audit and is rarely validated by authorities. Conversely, institutional actors (police, civil protection, municipalities) have critical information about physical restrictions (road closures, security perimeters). An operational crisis management system must therefore combine these sources, specifying the origin, validation status, and time frame of events. The proposed platform implements a multi-stakeholder workflow (draft → validated → disabled) and translates verified events into routing constraints.

## 3. COLLABORATIVE WEB MAPPING PLATFORM

The platform is designed to bring together several stakeholders: security forces (e.g., National Gendarmerie), emergency services (Civil Protection), municipal technical services, and citizens. Each report is georeferenced, described (type, severity, comment), and time-bound (start_time, end_time). A validation flow (draft → validated → disabled) distinguishes verified events from unconfirmed reports.

Technically, the reference implementation is based on (i) a REST backend (FastAPI) ensuring input validation (Pydantic) and an OpenAPI specification, (ii) a PostgreSQL/PostGIS database hosting the road graph, events, and logs, and (iii) a web frontend (React/Leaflet) for map entry

and route visualization. For interoperable institutional deployments, map distribution can be provided via OGC services (WMS/WFS) exposed by GeoServer connected to PostGIS.

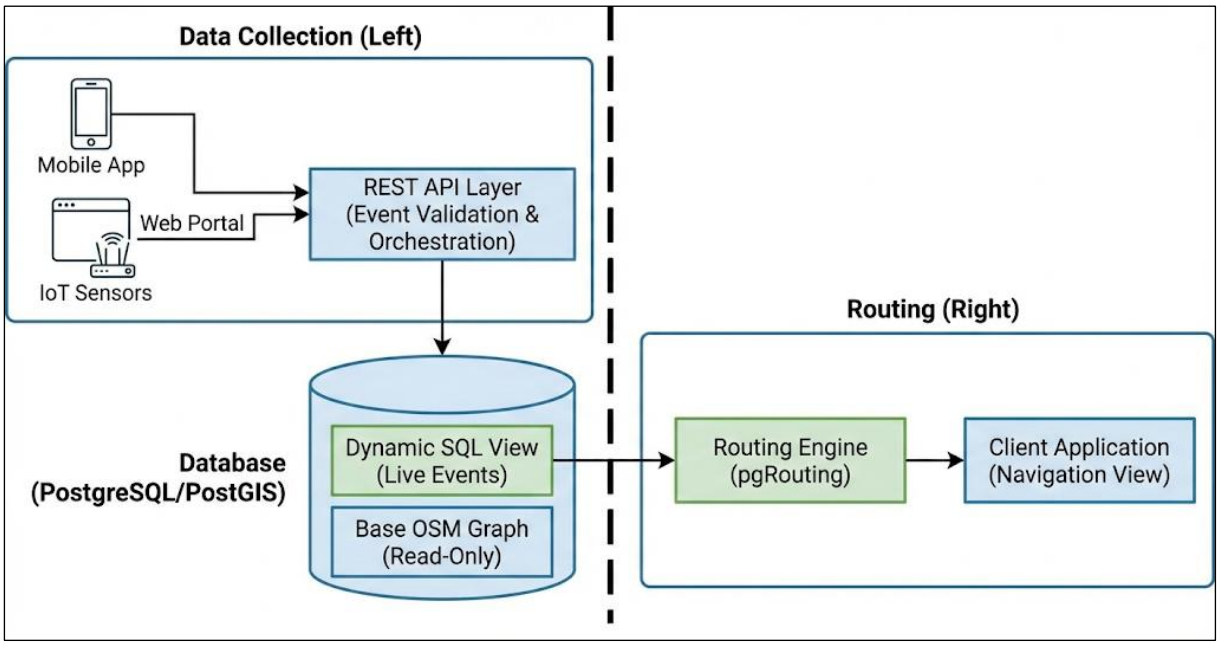


Figure 1. Modular architecture of the platform. The system decouples collection (left) from routing (right). The REST API layer orchestrates event validation and exposes data to the routing engine via a dynamic SQL view, ensuring that the underlying OSM graph remains unaltered (read-only).

## 4. DATA, SCHEMAS AND EVENT MODEL

Road network — The road graph is derived from OpenStreetMap and imported into PostGIS as directed edges (pgRouting table). In the implementation, the edge table (e.g., algeria_2po_4pgr) includes: id, source, target, geom_way (LineString, SRID 4326), km, kmh, cost, and reverse_cost (directed weights), clazz (class), osm_id/osm_name, and x1,y1,x2,y2 (required by pgr_astar). The performance of proximity operations relies on a GiST spatial index on geom_way and the use of the KNN operator (<->) for nearest neighbor searches.

Vertices and snapping — Start/end/via points are projected onto the vertices table (e.g., algeria_2po_vertex) via KNN on geom_vertex (GiST index), with a quality filter (ref_count>0). A snapping threshold of MAX_SNAP_METERS=2500 m allows queries outside the network (sea, uncovered areas) to be rejected and improves the stability of the calculated paths.

Events — Road events are stored in a dedicated table (events_road_closed) and reference the affected edge (edge_id). They include a validity interval, status, impact level, and geometry aligned with the edge, ensuring consistency between reports and the network. Indexes on status and edge_id speed up the selection of active events. Instrumentation — Routing calls (success/failure) and external comparisons are logged (stats_route_log, stats_route_fail_log, stats_edge_usage_daily, stats_route_compare), enabling the production of indicators, the replay of scenarios, and the feeding of a performance dashboard.

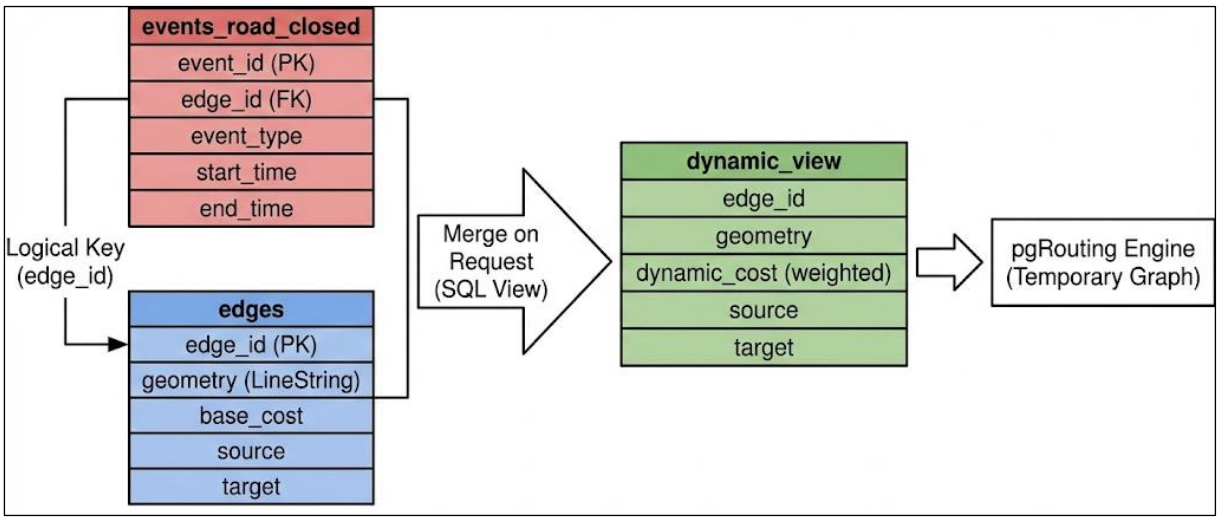


Figure 2. Relational-spatial data model. Events (events_road_closed) are linked to the road graph (edges) via a logical edge_id key. During a query, the dynamic view merges these tables to produce a temporary weighted graph used by pgRouting.

## 5. ALGORITHMS: EVENT-WEIGHTED ROUTING

### 5.1 Dynamic cost function and event-based view

The system avoids any re-materialization of the graph: for each query, the routable layer is built on the fly in the form of an SQL view (or CTE) called ways_live(t0), derived from the OSM edge table (ways) and enriched by the events active at time t0. This view translates validated reports into local cost changes: (i) blocking events result in the strict exclusion of the corresponding edge; (ii) severe or moderate events apply a multiplicative penalty to cost and reverse_cost; (iii) info events are kept for information purposes (at no extra cost). This strategy guarantees reversibility (end of validity ⇒ disappearance of the impact) and preserves the network topology.

To represent temporal criticality (degradation then gradual resorption), the penalty is modulated by a decay function depending on the age of the event. Noting age_seconds(t0) as the age (in seconds), we define:

decay(t0) = 3600 / (age_seconds(t0) + 3600)

and the penalty: P(t0) = P_base × decay(t0).

Thus, a recent event induces a high additional cost, which automatically decreases over time. This modulation promotes more robust routes without making the network unavailable.

In practice, for an edge e, the effective cost used by the algorithm is written as: cost_eff(e,t0)=cost_base(e) × (1 + P_e(t0)), and symmetrically for reverse_cost. In the case of a blocking event, e is removed from the subgraph (equivalent to an infinite cost). This notation unifies exclusion and penalization in the same "time-dependent cost" semantics.

### 5.2 Routing algorithm and spatial optimization

The optimal route is calculated using pgRouting's shortest path primitives (pgr_dijkstra or pgr_astar), applied to a weighted edge layer on demand. Rather than re-materializing the graph after each report, the system builds a dynamic SQL view called ways_live(t0) at runtime, which integrates active events at time t0 by (i) excluding blocking edges and (ii) penalizing costs for non-blocking disruptions. This approach ensures that network changes are taken into account immediately, while preserving the integrity of the base tables.

On a dense network at the scale of a wilaya (order of magnitude: tens of thousands of edges), global exploration can induce too much latency for smooth cartographic interaction. To reduce computation time without compromising completeness, we introduce a mechanism for reducing the candidate subgraph based on a geometric constraint, called spatial corridor filtering.

Formally, given the geodesic segment connecting the starting point P_start to the end point P_end, we define a corridor $C(\varepsilon) = \{ e \in ways\_live(t0) \mid d(geom\_e, seg(P\_start,P\_end)) \leq \varepsilon \}$. The selection of candidate edges is performed by PostGIS via ST_DWithin(geom_way, seg(P_start,P_end), ε). In the current implementation (geometries in EPSG:4326), ε is expressed in degrees; at the latitude of Oran, 0.05° corresponds to approximately 5–6 km, and 0.30° to ~30 km. A metric variant, based on ST_Transform (projected CRS) or on a cast in geography, allows ε to be expressed in meters during industrial deployment.

The strategy is progressive in order to balance performance and robustness: (1) strict corridor (low ε) to cover the majority of direct routes; (2) widened corridor (increased ε) when major detours are necessary (e.g., bypasses, closures); (3) global fallback to the entire network if no path is found in the constrained corridors. This progression greatly reduces the number of edges explored in typical cases, while ensuring that a path will be returned if it exists in the graph.

Table 1. Corridor filtering parameters (case study: Wilaya of Oran).

The values of ε are configuration parameters: they can be adjusted according to the density of the network, the urban/suburban context, and the scale of the study area. In the evaluation (Section 7), we report the frequency of global backtracking and its impact on latency in order to quantify the gain provided by filtering.

In order to make the procedure reproducible and independent of implementation details, we summarize the calculation flow below in pseudo-code.

| Level | ε (EPSG:4326) | Interpretation ( Oran) | Role |
|---|---|---|---|
| **Strict corridor** | 0,05° | approx. 5-6 km | Fast response on direct routes |
| **Wider corridor** | 0,30° | approx. 30 km | Allows major detours and diversions |
| **Overall decline** | Ø | Complete network | Guarantees completeness if a path exists |

**Algorithm 1**. Corridor routing on a dynamic view ways_live(t0).

Inputs: P_start, P_end, list of vias V (optional), profile π, time t0.
1) Projection (snapping) of each point (P_start, V, P_end) onto the graph; if dist > MAX_SNAP_METERS, return an error.
2) For each consecutive segment (p_i → p_{i+1}):
2.1) For ε ∈ {ε_strict, ε_large, ∅}: construct the edge query (ways_live(t0) ∩ C(ε)) (or ways_live(t0) if ε=∅).
2.2) Run pgr_astar (or pgr_dijkstra) on this subgraph; if a path is found, exit the ε loop.
2.3) If no path is found after ε=∅, return a "network disconnected under constraints" error.
3) Concatenate the paths of the segments, aggregate distance/duration, and reconstruct the geometry; produce a roadbook.
4) Log the request (success/failure, latency, metrics) and return the response.

### 5.3 Point projection and multi-stage routing

As indicated in Algorithm 1, the first step consists of projecting each user point (lon, lat) onto the graph ("snapping") via a nearest neighbor search (KNN, <-> operator) on the vertex table, accelerated by a GiST index. The distance is evaluated in metric (geography) and a threshold MAX_SNAP_METERS = 2500 m is imposed: beyond this, the query is rejected in order to avoid unstable routes when the point is too far from the routable network.

Multi-step routing is supported by breaking down the route into successive segments (start → via1 → ... → end). Each segment is solved separately (pgr_astar in directed mode), then concatenated, aggregating the metrics (distance, duration) and geometry. This strategy ensures

compatibility with web interaction (adding/deleting waypoints) while preserving the traceability of the edge sequence used for the calculation.

### 5.4 Outputs, explainability and auditability

The /route response includes (i) the ordered list of edges (edges[]) for auditing, (ii) a GeoJSON geometry reconstructed by aggregating the segments (ST_Union then ST_LineMerge), and (iii) a "roadbook" derived from grouping consecutive edges by street name (osm_name). At the same time, each query (successful or unsuccessful) is logged in statistics tables (latency, metrics, errors), which allows scenarios to be reproduced and feeds into the benchmarking dashboard.

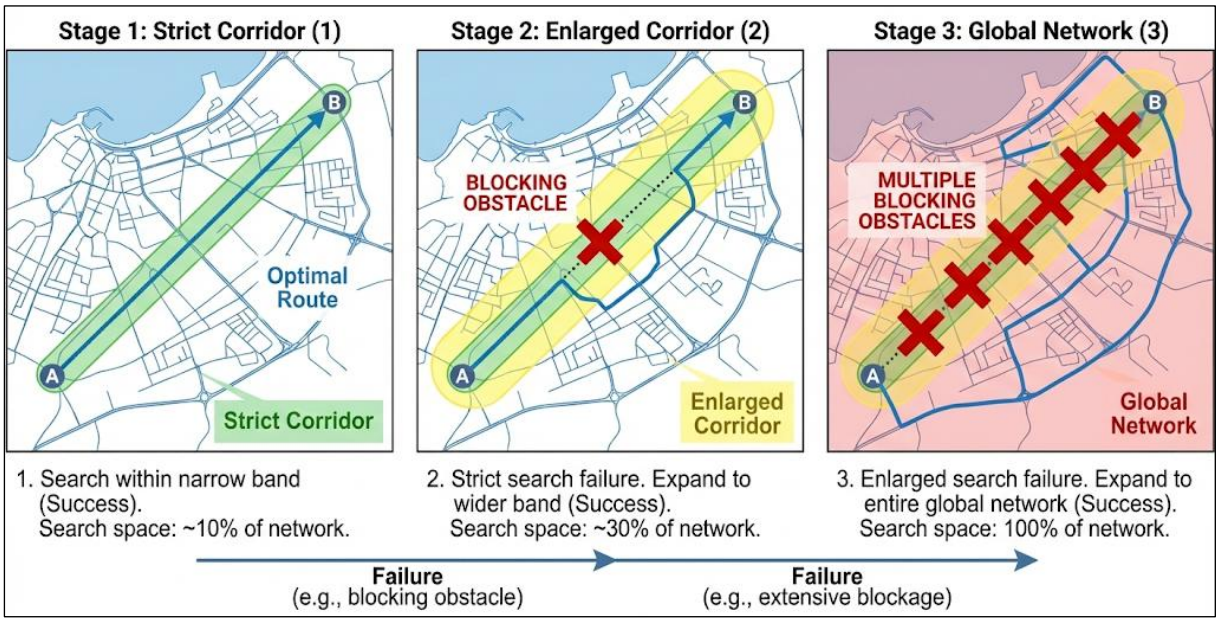


Figure 3. Spatial filtering strategy by corridors. The algorithm first searches for the solution in the strict corridor (1). If unsuccessful (e.g., blocking obstacle), it extends the search to the expanded corridor (2), then to the global network (3). This heuristic greatly reduces the search space for standard queries (Oran configuration).

## 6. NEAR REAL-TIME EVENT INTEGRATION

Upon receiving a report, the API associates the event with an edge by proximity request (nearest-edge) and records the spatio-temporal and validation attributes. During a /route call, the selection of active events is performed at the database level and injected into a CTE that (i) filters blocked edges and (ii) adjusts the costs of penalized edges. This on-the-fly rewriting is suitable for moderate traffic and ensures immediate and reversible processing. For higher loads

(nationwide scaling), materialization (materialized view) or a trigger mechanism can be considered to maintain pre-calculated running costs.

## 7. COMPARATIVE EVALUATION: CASE STUDY IN ORAN

### 7.1 Experimental protocol

The evaluation was conducted on the road network of the wilaya of Oran, built from OpenStreetMap data and transformed into a pgRouting routable graph (≈ 12,000 edges). In order to reproduce realistic urban hazards, three disruption scenarios were injected on strategic routes: (S1) blocking incident, (S2) major works (severe), and (S3) road check (moderate). For each scenario, we compare: (i) the reference route (without event), (ii) the route recalculated by our platform after injection of the event, and (iii) a third-party solution (Google Maps Traffic) queried at time t0 on the same origin-destination pair.

### 7.2 Quantitative results

Table 2 summarizes the routing performance for three representative scenarios. For each scenario, we report the estimated distance and duration (baseline vs. adjusted route), as well as the calculation latency. The relative impacts are expressed in relation to the baseline.

Table 2. Comparative results of disruption scenarios (Oran).

| SCENARIO (ORIGIN -> DESTINATION) | SIMULATED EVENT (TYPE) | BASELINE (DIST / TIME / LATENCY) | MODIFIED (DIST / TIME / LATENCY) | IMPACT | GOOGLE (DIST / TIME) |
|---|---|---|---|---|---|
| CITY CENTER -> AIRPORT (PLACE 1ER NOV -> ES SENIA) | Serious accident blocking | 12.23 km<br>10.9 min<br>340 ms | 12.43 km<br>12.2 min<br>380 ms | +0.20 km<br>(+1.6%)<br>+1.3 min<br>(+11.9%) | 12.32 km<br>14.6 min |
| PORT OF ORAN -> HASSI AMEUR INDUSTRIAL AREA | Public works severe | 19.57 km<br>17.6 min<br>360 ms | 19.57 km<br>17.9 min<br>415 ms | 0.00 km<br>+0.3 min<br>(+1.7%) | 19.65 km<br>20.1 min |
| USTO UNIVERSITY -> SANTA CRUZ | Traffic control moderate | 12.97 km<br>15.6 min<br>320 ms | 12.97 km<br>15.6 min<br>350 ms | 0.00 km<br>+0.0 min | 13.18 km<br>16.2 min |

### 7.3 Performance analysis

Strict avoidance (Scenario S1) — When faced with a blocking event (serious accident or road closure), the algorithm detects the interruption and immediately calculates an alternative route that fully complies with the constraints. The detour remains moderate (+1.6% in distance), but the time increases (+11.9%) due to secondary segments with speed limits.

Stability and impact gradation (Scenarios S2 and S3) — For severe or moderate events, the penalty applied to the affected arcs allows the response to be modulated: in S2, the route is slightly adjusted in terms of duration (+0.3 min; +1.7%) with no significant change in distance, while in S3 the penalty is insufficient to justify a detour, thus avoiding unnecessary route changes.

Engine performance — Response times remain compatible with an interactive web application: latency remains sub-second and 95% of requests are served in less than 500 ms. Spatial corridor filtering (ST_DWithin) significantly reduces the search space and stabilizes latency as network and event density increase.

### 7.4 Resilience simulation campaign: reproducible protocol

Beyond the targeted scenarios (S1–S3), we propose a simulation campaign designed to characterize the resilience of the routing engine in the face of a variety of events and origin–destination configurations. The objective is twofold: (i) to estimate the latency distribution (P50/P95 percentiles) under realistic load and (ii) to measure the bypass capacity and stability of routes when disruptions are injected in a controlled manner.

Request sampling — A set of origin-destination (OD) pairs is generated within the boundaries of the wilaya of Oran. Each point is validated by snapping onto the graph; rejected ODs (distance to the network > MAX_SNAP_METERS) are replaced. To cover multiple mobility regimes, the sample is stratified by Euclidean distance (e.g., intra-center, intra-wilaya, peri-urban), and parameters (size N, sampling method, random seed) are recorded to ensure reproducibility.

Event injection — For each OD, a reference route is calculated on ways_live(t0) without events. Events are then injected on one or more candidate edges (edge_id) in two modes: (a) "direct impact" mode, where the disrupted edge belongs to the reference route; (b) "diffuse impact" mode, where the event is placed in a close geographical vicinity (enlarged corridor) in order to test robustness to deviations. Impact levels (blocking, severe, moderate) and time windows are configured; all generated events are archived (edge_id, level, t0, duration) in the database to enable exact replication.

Measurements — For each execution, the system records (i) the calculation latency, (ii) the estimated distance and duration, (iii) the use of the global fallback mechanism, (iv) effective avoidance of blocked edges, and (v) topological changes relative to the reference route ("flutter" rate). The metrics selected (Table 3) are directly interpretable and aligned with the operational objective: to produce feasible, stable routes calculated in near real time.

Table 3. Resilience campaign metrics (definitions).

This campaign complements scenarios S1–S3 by providing a statistical estimate of performance and replanning mechanisms. It also provides direct support for the analysis of threats to validity (Section 8.2) by explicitly documenting parameters, event distribution, and global fallback cases.

| Metric | Definition | Interpretation |
|---|---|---|
| M1 — Latency (P50/P95) | API-side computation time for /route (ms), summarized by percentiles. | Perceived responsiveness and stability under load. |
| M2 — Relative detour | $\Delta$dist = (dist_evt − dist_ref) / dist_ref. | Additional distance cost induced by the event. |
| M3 — Relative delay | $\Delta$time = (time_evt − time_ref) / time_ref. | Additional time cost incurred (weighting + detours). |
| M4 — Avoidance rate (blocking) | % of requests where the blocked edge does not appear in edges[]. | Ability to bypass an infrastructure failure. |
| M5 — Overall fallback rate | % of requests requiring $\varepsilon=\emptyset$ (full network search). | Difficult cases; impact on latency. |
| M6 — Beat rate | Proportion of edge changes for non-blocking disruptions. | Route stability (avoids unnecessary recalculations). |

## 8. DISCUSSION AND IMPLICATIONS

### 8.1 Interpretation of results

The experiments confirm that the non-destructive dynamic overlay architecture allows validated events to be integrated without modifying the OpenStreetMap topology, while maintaining latency compatible with a web application. In the major closure scenario (S1), the platform calculates a minimal detour (+1.6% in distance) and provides a duration estimate (12.2 min) that is lower than that of Google Maps (14.6 min), illustrating the value of using validated infrastructure information. It should be noted, however, that our approach focuses primarily on discrete disruptions (road closures, incidents) rather than the detailed estimation of instantaneous speeds from continuous flows.

### 8.2 Internal validity and threats to validity

Several threats to validity must be considered. (i) Data validity: the quality of OpenStreetMap (completeness, maxspeed, one-way streets) directly influences the estimated times. (ii) Event validity: the scenarios injected model realistic disruptions, but do not cover the full range of situations (multi-events, cascades). (iii) External validity: the case study focuses on the wilaya of Oran; tests on other wilayas and denser networks are needed to generalize the results.

### 8.3 Limitations and operational implications

Two main limitations emerge. First, travel times are derived from a model based on maxspeed/priors and do not always reflect actual congestion; the integration of continuous signals (sensors, IoT, fleet data) would allow for finer calibration. Second, in the event of very high event density, an additional acceleration strategy (contraction hierarchies, pre-partitioning) could improve latency robustness. Despite these limitations, the proposed framework offers an interesting compromise between auditability, explainability, and responsiveness for institutional uses.

## 9. CONCLUSION AND PERSPECTIVES

We have presented a collaborative web mapping platform for route optimization in a dynamic road network in Algeria, based on PostGIS/pgRouting and a non-destructive event overlay. The near real-time integration of validated reports (draft → validated → disabled) allows strict avoidance of blocking while modulating the effect of less critical disruptions. The spatial corridor heuristic (ST_DWithin) reduces the search space and contributes to web-compatible response times: 95% of queries are served in less than 500 ms.

In the short term, prospects include (i) calibration of durations via continuous signals (IoT, fleet, public data), (ii) alignment with incident exchange standards (e.g., DATEX II), and (iii) the study of algorithmic accelerations (pre-partitioning, hierarchies) for large networks. Finally, extending the evaluation protocol to other provinces will provide a better picture of resilience at the national level.

REFERENCES

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

Hart, P.E., Nilsson, N.J. and Raphael, B., 1968, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics, 4(2), 100–107.

OpenStreetMap contributors, 2026, OpenStreetMap, open data licensed under ODbL, documentation.

PostGIS Project Steering Committee, 2025, PostGIS—Spatial and Geographic Objects for PostgreSQL, documentation.

pgRouting Developers, 2025, pgRouting: a library for routing and network analysis, documentation.

OGC, 2006–2020, Web Map Service (WMS) / Web Feature Service (WFS) Standards, Open Geospatial Consortium.
GeoServer Project, 2025, GeoServer Documentation, documentation.
Tiangolo, S. et al., 2025, FastAPI Documentation, documentation.
React Contributors, 2025, React Documentation, documentation.
Vite Contributors, 2025, Vite Documentation, documentation.
RoutingPy Developers, 2025, RoutingPy: Python package for routing services, documentation.

BIOGRAPHICAL NOTES

Bouhadjar MEGUENNI works at the Centre des Techniques Spatiales (CTS), Algeria. His interests include geospatial databases, OpenStreetMap-based road networks, and algorithmic methods for urban routing and data quality assessment.

CONTACTS

Bouhadjar MEGUENNI
Centre des Techniques Spatiales
10, Boulevard de la Palestine, BP13
Arzew, ALGERIA
Tel. +213 660215203
Email: meguennib@gmail.com