Integrazione di Geometrie Reali e Fattori di Rischio Dinamici in Grafi di Trasporto Pubblico: Un'Analisi Comparativa

Mattia Isidoro
Dipartimento di Ingegneria, Università di Modena e Reggio Emilia
299018@studenti.unimore.it

28 novembre 2025

Sommario

Il presente lavoro propone una metodologia avanzata per l'arricchimento dei grafi di trasporto pubblico basati su specifiche GTFS, integrando dati geometrici ad alta precisione e mappe di rischio incidentale tempo-dipendenti. L'obiettivo è superare le limitazioni dei tradizionali algoritmi di routing basati esclusivamente su orari e sequenze logiche di fermate. Vengono presentati e confrontati due distinti approcci implementativi su database a grafo Neo4j: un approccio a *Geometria Completa*, che modella fisicamente ogni punto del tracciato, e un approccio *Semplificato Orientato al Routing*, che aggrega le proprietà geometriche e di rischio sulle relazioni logiche. I risultati, applicati al caso studio della rete urbana di Modena, dimostrano come l'approccio semplificato offra un bilanciamento ottimale tra accuratezza informativa ed efficienza computazionale per il calcolo di matrici di tempi di percorrenza tra quartieri.

1 Introduzione

La pianificazione della mobilità urbana moderna richiede strumenti di routing che vadano oltre la semplice minimizzazione del tempo di viaggio. Fattori come la sicurezza del percorso e l'aderenza alla topologia reale della rete stradale sono diventati cruciali.

I sistemi standard basati su GTFS (General Transit Feed Specification) modellano la rete come una sequenza discreta di fermate (*Stop*) collegate da segmenti logici. Tuttavia, questo modello astrae eccessivamente la realtà fisica: ignora la curvatura delle strade e, soprattutto, l'esposizione al rischio di incidenti, che non è uniforme nello spazio né nel tempo.

Questo studio mira ad integrare nel grafo di routing:

- La geometria esatta delle linee (file `shapes.txt`).
- La probabilità di incidente variabile per fascia oraria (Morning, Afternoon, Night), derivata da file raster.

2 Definizione del Problema

Il problema principale risiede nella discrepanza tra il modello logico del GTFS e la realtà fisica. Nel GTFS, un viaggio (*Trip*) è definito da una sequenza di orari (*StopTimes*). Il percorso tra due fermate consecutive è spesso approssimato come una linea retta o un arco privo di attributi spaziali intermedi.

Tuttavia, ai fini della sicurezza, ciò che accade *tra* due fermate è fondamentale. Un segmento di strada potrebbe attraversare un incrocio ad alta incidentalità. Se l'algoritmo di routing ignora questo dato, potrebbe suggerire percorsi veloci ma potenzialmente pericolosi.

La sfida tecnica consiste nel mappare dati raster continui (rischio) su una struttura a grafo discreta, mantenendo prestazioni di query accettabili per il calcolo dei percorsi tra i centroidi dei quartieri urbani.

3 Dati e Strumenti

Per lo studio, sono stati utilizzati i seguenti dataset relativi alla città di Modena:

1. **GTFS Statico**: Agenzia, Linee, Trip, Stop e Orari.
2. **Shapes**: Coordinate GPS dettagliate dei percorsi, cioe in grado di descrivere il percorso esatto effettuato da una trip
3. **Raster di Rischio**: Tre file GeoTIFF rappresentanti la probabilità di incidente (normalizzata [0-1]) per fasce orarie.
4. **GeoJSON Quartieri**: Confini amministrativi per l'individuazione dei centroidi di origine/destinazione.

L'architettura software si basa su **Python** (Pandas, Geopandas, Rasterio) per l'elaborazione e **Neo4j** per la persistenza del grafo e l'esecuzione degli algoritmi di cammino minimo (Dijkstra).

4 Approccio 1: Modellazione a Geometria Completa

Il primo approccio metodologico esplorato in questo studio mira alla massima fedeltà nella rappresentazione topologica della rete di trasporto. A differenza dei modelli GTFS standard, che approssimano i percorsi tra le fermate come archi logici privi di forma, questo modello ricostruisce fisicamente la geometria del tracciato all'interno del grafo.

L'obiettivo è creare un substrato fisico ("Physical Layer"), basato sulle coordinate conetnute nel file shapes.txt, su cui mappare i dati di rischio con una risoluzione spaziale metrica, permettendo analisi di sicurezza granulari.

4.1 Struttura del Grafo e Nodi ShapePoint

La struttura del grafo viene estesa introducendo una nuova classe di nodi derivata dal file `shapes.txt` del feed GTFS.

Sia $P = \{p_1, p_2, ..., p_n\}$ la sequenza ordinata di coordinate che definisce la forma di una specifica linea (Shape). Nel grafo, ogni punto p_i viene istanziato come un nodo **ShapePoint**.

Definiamo formalmente le entità coinvolte:

- **Nodo ShapePoint (SP)**: Rappresenta una singola coordinata GPS lungo il percorso.

$$SP_i = \{\text{id}, \text{lat}, \text{lon}, \text{sequence}, \text{shape\_id}\}$$

- **Relazione FOLLOWS**: Collega sequenzialmente i punti fisici, ricostruendo la polilinea stradale.

$$(SP_i) \xrightarrow{\text{FOLLOWS}} (SP_{i+1})$$

La relazione `FOLLOWS` è pesata con la distanza geodetica reale $d(SP_i, SP_{i+1})$, calcolata tramite la formula dell'Haversine. Questo permette di conoscere la lunghezza esatta di ogni micro-segmento stradale, che spesso varia tra i 10 e i 50 metri, sotto l'ipotesi che questi segmenti siano appunto in linea retta.

4.2 Integrazione dei Raster di Rischio

La creazione dei nodi fisici abilita un'integrazione dei dati di rischio incidente ad alta precisione. Poiché la probabilità di incidente è fornita tramite file Raster (GeoTIFF) continui, è necessario discretizzare questa informazione sul grafo.

Il processo di mappatura avviene in due fasi:

4.2.1 Campionamento sui Segmenti Fisici

Per ogni arco (SP_i, SP_{i+1}), l'algoritmo calcola il punto medio geografico m_i e interroga i dataset raster temporali R_t (dove $t \in \{\text{Morning, Afternoon, Night}\}$).

Il valore di rischio estratto viene salvato direttamente come proprietà dell'arco fisico:

$$\text{FOLLOWS}_i.\text{risk}_t = \text{Raster}_t(m_i.\text{lat}, m_i.\text{lon}) \qquad (1)$$

In questo modo, il grafo possiede una "mappa di calore" intrinseca: è possibile interrogare il database per individuare non solo quali linee sono pericolose, ma quali specifici segmenti di strada (es. un particolare incrocio o curva) presentano il rischio maggiore.

4.2.2 Propagazione al Livello Logico (Trip)

Sebbene i nodi `ShapePoint` rappresentino la realtà fisica, l'algoritmo di routing opera sul livello logico dei servizi (`Trip` e `Stoptime`). È quindi necessario trasferire l'informazione di rischio dal livello fisico a quello di servizio.

Definiamo il rischio di un arco logico `PRECEDES` (che collega due fermate A e B) come la media ponderata del rischio di tutti i segmenti `FOLLOWS` compresi tra A e B:

$$\text{Risk}(A, B)_t = \frac{1}{N} \sum_{k=1}^{N} \text{FOLLOWS}_k.\text{risk}_t \qquad (2)$$

dove la sequenza di archi k rappresenta l'insieme ordinato di segmenti fisici che costituiscono il tracciato tra le due fermate.

Questa operazione di aggregazione viene eseguita tramite una procedura Cypher ottimizzata (`apoc.periodic.iterate`) che percorre la catena di `ShapePoint` associata a ciascun `Trip` e aggiorna le proprietà di costo sulle relazioni di routing.

4.3 Vantaggi e Limitazioni

L'approccio a Geometria Completa offre il massimo dettaglio informativo. Permette di visualizzare il percorso esatto su strumenti GIS e di condurre analisi di sicurezza puntuali (es. identificazione di "Black Spots" lungo le linee).

Tuttavia, questo modello comporta un significativo aumento della cardinalità del grafo. Per una rete urbana come quella di Modena, l'introduzione di centinaia di migliaia di nodi `ShapePoint` e delle relative relazioni impatta sui tempi di caricamento e sulla memoria necessaria per le proiezioni GDS, rendendo questo approccio più indicato per analisi offline piuttosto che per il routing in tempo reale su dispositivi con risorse limitate.

4.4 Struttura del Grafo

Oltre ai nodi standard del GTFS, vengono introdotti i nodi `ShapePoint`. La struttura prevede:

- Nodi: `ShapePoint {lat, lon, sequence}`
- Relazioni: `(:ShapePoint)-[:FOLLOWS]->(:ShapePoint)`

Ogni relazione `FOLLOWS` rappresenta un micro-segmento stradale reale (pochi metri). L'intera struttura del grafo e visibile in 1 La principale sfida tecnica in questa fase risiede nella disconnessione strutturale nativa dello standard GTFS: le fermate (`Stop`) sono punti logici individuali, mentre i tracciati (`Shapes`) sono sequenze di coordinate fisiche, senza un legame esplicito che indichi quale segmento di strada unisca due fermate specifiche. La risoluzione di questa ambiguità avviene mediante un'associazione relazionale basata su un identificativo univoco. Durante la fase di propagazione del rischio, il sistema esegue un join efficiente sfruttando la chiave esterna shape-id presente sia nei nodi Trip (livello di servizio) sia nella catena di nodi ShapePoint (livello infrastrutturale). Questo permette di aggregare i profili di rischio della geometria stradale e

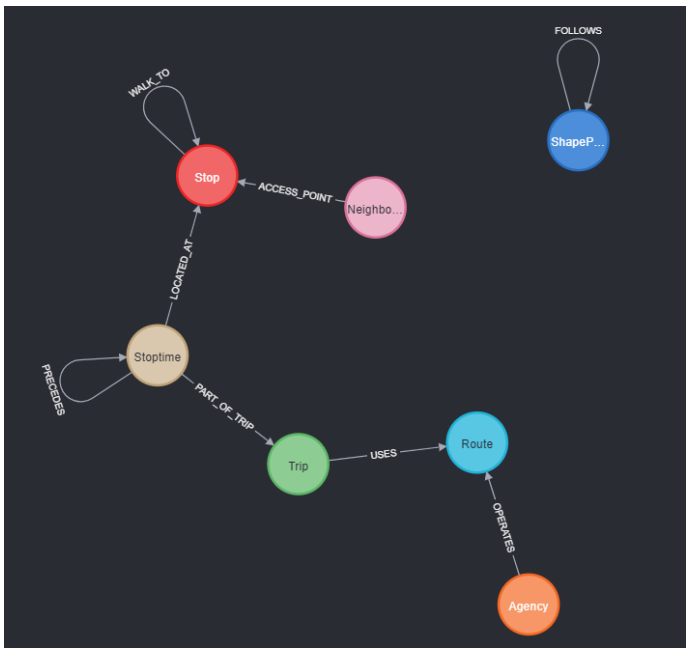

Figura 1: struttura grafo complete shapes

trasferirli al viaggio corrispondente senza la necessità di istanziare onerosi archi topologici di proiezione per ogni singola fermata. Questa operazione trasforma il grafo in una struttura *Dual-Layer*: il routing non avviene più solo tra nodi logici, ma attraverso un percorso ibrido che transita dal nodo di servizio al nodo fisico, percorre la catena di `ShapePoint` (su cui è mappato il rischio puntuale) tramite relazioni `FOLLOWS`, e risale infine alla fermata di destinazione.

4.5 Analisi Critica

Questo approccio offre una granularità eccellente per analisi spaziali (es. hotspot di incidenti). Tuttavia, comporta un'esplosione combinatoria delle dimensioni del grafo. Per una rete urbana, il numero di nodi `ShapePoint` può essere di ordini di grandezza superiore ai nodi `Stop`, rallentando significativamente le proiezioni in memoria (GDS) e le query di manutenzione.

5 Approccio 2: Modellazione Semplificata (Aggregata)

Il secondo approccio, sviluppato per ottimizzare le performance di routing su Neo4j Desktop, sposta la complessità geometrica dal grafo alla fase di pre-elaborazione.

5.1 Metodologia

Invece di creare nodi per ogni punto della forma, la geometria viene gestita interamente in Python prima del caricamento. L'algoritmo procede come segue:

1. **Filtraggio**: Selezione delle sole linee urbane (estratte dal sito ufficiale della SETA `https://www.setaweb.it/mo/` per ridurre il rumore dei dati.
2. **Segmentazione**: Per ogni coppia di fermate consecutive ($Stop_A \rightarrow Stop_B$), si estrae la porzione corrispondente di coordinate dal file `shapes.txt`.
3. **Calcolo WKT e Rischio**: La sequenza di punti viene convertita in una stringa `LineString` (WKT). Contemporaneamente, si calcola la media dei valori raster lungo questi punti.

5.2 Struttura del Grafo

Il grafo risultante è molto più leggero. Non esistono nodi `ShapePoint`. Le informazioni geometriche e di rischio sono salvate direttamente come proprietà della relazione `PRECEDES` tra gli `Stoptime`.

Esempio di proprietà sulla relazione `PRECEDES`:

```
{
  "geometry": "LINESTRING(10.9 44.6, ...)",
  "risk_morning": 0.45,
  "risk_afternoon": 0.20062471429506937,
  "risk_night": 0.11005776623884837,
  "waiting_time": 120
}
```

5.3 Definizione dei Nodi e Archi

Nell'approccio semplificato proposto, la topologia del grafo è ottimizzata per ridurre la cardinalità dei nodi senza perdere informazione spaziale. Il grafo risultante $G = (V, E)$ è strutturato su tre livelli logici interconnessi: livello territoriale, livello infrastrutturale e livello di servizio.

Le entità principali (Nodi V) e le loro interconnessioni (Archi E) sono definite come segue:

- **Neighborhood** (N): Rappresenta il centroide di un'area urbana (quartiere). È il punto di generazione o attrazione della domanda di mobilità. Estratto e creato dal file QuartieriModena.geoson
- **Stop** (S): Rappresenta l'infrastruttura fisica della fermata del trasporto pubblico.
- **Stoptime** (ST): Rappresenta un evento temporale specifico (es. l'arrivo di un veicolo specifico ad una fermata specifica).

Le relazioni tra queste entità codificano la logica di movimento:

1. **ACCESS_POINT** ($N \rightarrow S$): Collega un quartiere alla fermata più vicina. Il peso dell'arco rappresenta il tempo di camminata (distanza euclidea / velocità pedonale).
2. **LOCATED_AT** ($ST \rightarrow S$): Relazione strutturale che collega l'evento temporale al luogo fisico. Permette all'algoritmo di routing di modellare l'interscambio (scendere da un bus e attenderne un altro alla stessa fermata).
3. **PRECEDES** ($ST_i \rightarrow ST_{i+1}$): È l'arco fondamentale che rappresenta il viaggio a bordo del veicolo tra due fermate consecutive.

Innovazione del Modello: A differenza dei grafi GTFS standard, questo arco è arricchito con proprietà complesse pre-calcolate:

- `geometry`: Una stringa WKT (Well-Known Text) che descrive l'esatta polilinea stradale percorsa.
- `risk_factor`: Un valore scalare $[0, 1]$, per 3 diverse fasce orarie, che aggrega la probabilità di incidente lungo l'intero segmento stradale, derivato dai dati raster.

5.4 Rappresentazione Grafica dello Schema

La Figura 2 illustra la struttura locale del grafo e il flusso delle relazioni utilizzate dall'algoritmo di routing.

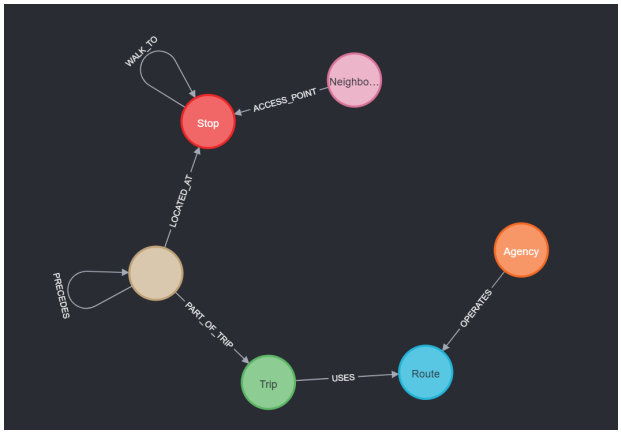

Figura 2: Struttura grafo

5.5 Vantaggi

Questo metodo riduce drasticamente il numero di nodi e relazioni, rendendo l'algoritmo di routing estremamente reattivo, pur mantenendo l'informazione sul "costo di rischio" totale del segmento.

6 Evoluzione dell'algoritmo di Routing: Stima Tempi + rischio incidenti

Per stimare i tempi di percorrenza tra i quartieri, è stato implementato un algoritmo di routing multimodale. Partendo dall'algoritmo di routing iniziale e modificandolo in modo corretto ne abbiamo ottenuto una versione migliorata in grado di tener conto delle shapes reali e soprattuto del rischio di incidenti in questi tragitti.Lo sviluppo del sistema di routing proposto è nato dall'analisi dei limiti intrinseci agli algoritmi standard per il trasporto pubblico. In questa sezione descriviamo il percorso evolutivo che ha trasformato un classico router basato su orari in un sistema multimodale sensibile al rischio.

6.1 Punto di Partenza: Il Routing GTFS Standard

Inizialmente, il sistema si basava su un'implementazione classica di routing tempo-dipendente (Time-Dependent Shortest Path). Il grafo era costruito proiettando esclusivamente le entità logiche del feed GTFS:

- **Nodi:** `Stop` (fermate fisiche) e `Stoptime` (eventi temporali).
- **Archi:** `PRECEDES` (viaggio in bus) e `WAITING` (attesa alla fermata).

L'algoritmo di base, tipicamente un Dijkstra o A*, minimizzava una funzione di costo C strettamente temporale:

$$C_{base} = T_{attesa} + T_{viaggio} + T_{cammino} \tag{3}$$

Sebbene efficace per calcolare l'orario di arrivo stimato (ETA), questo approccio presentava due criticità fondamentali per il nostro caso d'uso:

1. **Cecità Topologica:** Il percorso tra due fermate veniva modellato come un arco astratto. L'algoritmo ignorava se il bus percorresse un'autostrada sicura o un incrocio urbano pericoloso.
2. **Neutralità al Rischio:** Il sistema considerava equivalenti due percorsi di uguale durata, anche se uno dei due attraversava zone ad altissima incidentalità (identificate dai nostri raster).

6.2 Adattamento e Modifica del Modello

Per integrare i requisiti di sicurezza senza compromettere le prestazioni computazionali, abbiamo radicalmente modificato la logica di costruzione del grafo e la funzione di peso.

6.2.1 Arricchimento Topologico degli Archi

Invece di espandere il grafo inserendo migliaia di nodi intermedi per rappresentare la geometria della strada (approccio *ShapePoint*), che avrebbe aumentato esponenzialmente i tempi di calcolo, abbiamo optato per un approccio di **"Arricchimento degli Attributi"**.

Abbiamo modificato le fasi di Extract, Transform e Load per intersecare i dati geometrici (`shapes.txt`) con i segmenti logici del GTFS.

- **Prima:** L'arco `PRECEDES` tra la fermata A e B aveva solo la proprietà `waiting`$_{t}ime$.`Dopo`:$Lostessoarcocontieneoralaproprietgeometry(W$
Questo ci ha permesso di mantenere la complessità del grafo ($O(|V|+|E|)$) invariata rispetto al modello standard, pur iniettando l'informazione geografica reale.

6.2.2 2. Ridefinizione della Funzione di Costo

La modifica più significativa riguarda la funzione obiettivo dell'algoritmo di routing. Abbiamo abbandonato la minimizzazione del solo tempo a favore di un **Costo Generalizzato** che penalizza l'esposizione al rischio.

La nuova funzione di peso W per un arco e (segmento di viaggio) è definita come:

$$W(e) = T_{viaggio}(e) + (\text{Risk}(e,t) \times \alpha) \quad (4)$$

Dove:

- $\text{Risk}(e,t)$ è la probabilità di incidente media sul segmento e nella fascia oraria t, estratta dai dati raster.
- α è un fattore di penalità. Un valore di $\alpha = 1000$ implica che un rischio del 100% viene "pagato" dall'utente come 1000 secondi di ritardo "virtuale2.

L'approccio di base e stato quindi quello di rappresentare il rischio di incidente come un ritardo "extra"; più alto è il rischio, più alto è il ritardo.

6.3 Risultato dell'Evoluzione

Il passaggio dal routing iniziale a quello modificato ci ha permesso di trasformare un sistema di pianificazione oraria in uno strumento di supporto alle decisioni per la sicurezza urbana. L'algoritmo non cerca più "il percorso più veloce", ma "il miglior compromesso tra efficienza e sicurezza", sfruttando la topologia reale della rete modenese.

6.4 Funzione di Costo

L'algoritmo di Dijkstra non minimizza solo il tempo, ma una funzione di costo generalizzata C che penalizza i percorsi a rischio:

$$C_{totale} = T_{viaggio} + (P_{rischio} \times \alpha) \quad (5)$$

Dove:

- $T_{viaggio}$: Tempo di percorrenza o attesa (secondi).
- $P_{rischio}$: Probabilità di incidente normalizzata [0-1] per la fascia oraria specifica.
- α: Fattore di penalità (es. 1000). Un rischio alto viene percepito dall'algoritmo come un "ritardo virtuale", spingendo la scelta verso percorsi più sicuri.

6.5 Esecuzione

I centroidi dei quartieri vengono caricati come nodi `Neighborhood` e collegati alla fermata più vicina. Utilizzando la libreria *Graph Data Science* (GDS), viene calcolato il cammino minimo per tutte le coppie OD (Origine-Destinazione), restituendo una matrice di tempi stimati sensibile alla sicurezza. Quello che otteniamo e una matrice risultante del genere

6.6 Analisi Dettagliata degli Algoritmi di Routing

In questa sezione esaminiamo il funzionamento dell'algoritmo di ricerca del cammino minimo (Shortest Path) applicato ai due modelli di grafo proposti. Sebbene in entrambi i casi il motore algoritmico sottostante sia l'algoritmo di Dijkstra (implementato tramite la libreria Neo4j GDS), la topologia del grafo e la definizione della funzione di costo differiscono sostanzialmente, influenzando le prestazioni e la semantica del calcolo.

	Da	A	Score_Totale
0	Q_0	Q_1	343.271176
1	Q_0	Q_2	231.056525
2	Q_0	Q_3	390.362279
3	Q_0	Q_4	566.713884
4	Q_0	Q_5	792.254697
5	Q_0	Q_6	486.854154
6	Q_0	Q_7	286.908827
7	Q_0	Q_8	582.562537
8	Q_0	Q_9	908.931593
9	Q_1	Q_2	345.132004
10	Q_1	Q_3	584.038452
11	Q_1	Q_4	831.603682
12	Q_1	Q_5	1107.683793
13	Q_1	Q_6	798.730986
14	Q_1	Q_7	616.550043
15	Q_1	Q_8	852.361765
16	Q_1	Q_9	1128.164272
17	Q_2	Q_3	261.945216
18	Q_2	Q_4	501.554443
19	Q_2	Q_5	811.033354

Figura 3: results

6.6.1 Routing nell'Approccio a Geometria Completa (Shape-Based)

In questo modello, il grafo è un sistema ibrido composto da un livello logico (servizio) e un livello fisico (infrastruttura). Lo spazio di ricerca $\mathcal{G}_{full}$ è definito dall'unione dei nodi di servizio V_{srv} (Stop, Stoptime) e dei nodi fisici V_{phy} (ShapePoint). Tuttavia, eseguire un algoritmo di Dijkstra puro che attraversi sequenzialmente ogni singolo nodo fisico ($p_1 \to p_2 \to ... \to p_n$) per ogni viaggio sarebbe computazionalmente troppo costoso.

Pertanto, il routing avviene operativamente sul livello logico, ma il costo di ogni arco è derivato dinamicamente o pre-aggregato dal livello fisico sottostante.

6.6.2 Definizione del Peso (Weight Function)

Sia e_{log} un arco logico `PRECEDES` che collega due istanti temporali t_A e t_B. Il peso $W(e_{log})$ non è una proprietà statica, ma la somma dei costi del sottografo fisico corrispondente.

Se $\Phi(e_{log}) = \{s_1, s_2, ..., s_k\}$ è l'insieme ordinato di segmenti fisici (`FOLLOWS`) che compongono il tragitto tra le due fermate, il peso è calcolato come:

$$W(e_{log}) = \Delta T + \alpha \cdot \left(\frac{1}{L} \sum_{i=1}^{k} \text{Risk}(s_i) \cdot \text{len}(s_i) \right) \quad (6)$$

Dove:

- ΔT: Tempo di percorrenza schedulato (orario).
- α: Fattore di penalità di rischio.
- $\text{Risk}(s_i)$: Valore del raster sul singolo micro-segmento fisico.
- L: Lunghezza totale del tratto.

6.6.3 Complessità Operativa & risultato finale

Sebbene questo approccio sia semanticamente il più ricco, presenta una criticità in fase di *Graph Projection*. Per costruire il grafo in memoria su cui far girare Dijkstra, il database deve accedere a milioni di nodi o mantenere aggiornata la propagazione dei costi. Ogni modifica ai dati raster richiede un ricalcolo massivo su tutta la catena fisica. L'output finale, come per l'altro approccio è una matrice, rappresentante il ritardo totale in relazione al rischio per ogni coppia di quartieri,come mostrato in 4; con valori leggermente diversi dovuti alla maggiore precisione di questo approccio.

6.7 Routing nell'Approccio Semplificato (Edge-Based)

L'approccio semplificato riduce la complessità topologica "collassando" l'informazione geometrica e di rischio direttamente sugli archi del grafo di servizio. Lo spazio di ricerca $\mathcal{G}_{simple}$ è un sottoinsieme stretto di $\mathcal{G}_{full}$, contenente esclusivamente i nodi rilevanti per la decisione di viaggio:

$$V_{simple} = \{\text{Neighborhood}, \text{Stop}, \text{Stoptime}\}$$

Non esistendo nodi intermedi tra due fermate, il grafo è molto sparso e leggero. La transizione di stato da $Stop_A$ a $Stop_B$ è semplicemente un singolo salto.

6.7.1 Meccanismo di Calcolo del Costo

La peculiarità di questo approccio è che non serve attraversare nodi figli per conoscere il rischio; il rischio è una proprietà numerica pre-calcolata dell'arco stesso. La funzione di costo utilizzata dall'algoritmo di Dijkstra diventa un'operazione aritmetica diretta $O(1)$ per ogni arco visitato:

$$W(e) = \underbrace{e.\text{waiting\_time}}_{\text{Tempo GTFS}} + \underbrace{(e.\text{risk\_factor} \times \text{Penalty})}_{\text{Componente Sicurezza}} \quad (7)$$

Durante l'esplorazione del grafo (Traversal), quando l'algoritmo valuta l'arco `PRECEDES`, accede alla proprietà `risk_factor` (che in questo modello è la media pre-elaborata dei valori raster) e applica la penalità istantaneamente.

	Da	A	Score
0	Q_0	Q_1	343.271176
1	Q_0	Q_2	261.646273
2	Q_0	Q_3	468.853444
3	Q_0	Q_4	738.943917
4	Q_0	Q_5	792.254697
5	Q_0	Q_6	750.354206
6	Q_0	Q_7	565.431984
7	Q_0	Q_8	594.510715
8	Q_0	Q_9	910.709506
9	Q_1	Q_2	336.426499
10	Q_1	Q_3	575.332947
11	Q_1	Q_4	830.034069
12	Q_1	Q_5	1108.428187
13	Q_1	Q_6	798.730986
14	Q_1	Q_7	629.073951
15	Q_1	Q_8	871.862152
16	Q_1	Q_9	1166.907674
17	Q_2	Q_3	261.945216
18	Q_2	Q_4	532.035689
19	Q_2	Q_5	828.872890

Figura 4: Result complete geometry approch

6.7.2 Vantaggi Prestazionali e Funzionali

1. **Velocità di Proiezione:** Il caricamento del grafo nella memoria di lavoro (GDS Graph Projection) è estremamente rapido poiché coinvolge un numero di entità inferiore di ordini di grandezza (10^4 nodi invece di 10^6).

2. **Preservazione dell'Informazione:** Nonostante la semplificazione topologica, l'informazione geometrica non è persa. La proprietà `geometry` (stringa WKT) sull'arco permette, una volta calcolato il percorso ottimo (la sequenza di archi), di ricostruire e visualizzare la linea esatta sulla mappa, offrendo all'utente la stessa esperienza visiva del modello a geometria completa.

6.8 Confronto Sintetico

In sintesi, mentre l'approccio a geometria completa risolve il costo attraversando fisicamente lo spazio (o aggregandolo dinamicamente), l'approccio semplificato risolve il costo accedendo a metadati pre-elaborati. Dal punto di vista dell'algoritmo di routing, l'approccio semplificato è prefe-

ribile in quanto riduce lo spazio di ricerca senza sacrificare l'accuratezza della funzione di costo, permettendo il calcolo di matrici OD (Origine-Destinazione) su scala urbana in tempi compatibili con applicazioni real-time. In Tabella 1 è presente un confronto sintetico tra i due approcci.

7 Conclusioni

Lo studio ha dimostrato che l'integrazione di dati eterogenei (GTFS, Shapes, Raster) è fattibile e arricchisce significativamente le capacità di analisi della mobilità.

Sebbene l'Approccio 1 (Geometria Completa) sia superiore per visualizzazioni di dettaglio, l'Approccio 2 (Semplificato) si è rivelato la scelta ottimale per il calcolo di matrici OD su scala urbana. Esso garantisce la correttezza topologica del percorso e l'inclusione dei fattori di rischio, mantenendo al contempo il database leggero e le performance di calcolo compatibili con l'uso su workstation standard (Neo4j Desktop).

Sviluppi futuri potrebbero includere la ponderazione dinamica del fattore α in base alla tipologia di utente (es. anziani o studenti), personalizzando ulteriormente il routing sicuro.

Tabella 1: sintesi confornto: Full Geometry vs. Modello semplificato

Feature	**Approccio 1 (Full Geometry)**	**Approccio 2 (semplificato)**
Struttura	Nodi fisici (`ShapePoint`) + Nodi logici	Solo nodi logici (`Stop`, `Stoptime`)
Ipleentazione rischio	Micro-segmenti (point-based)	Prorpietà di edge logici (media)
Graph Complexity	High (Node explosion)	Low (Optimized topology)
Costo computazionale	Aggregazione dinamica ($O(N)$)	Ricerca diretta ($O(1)$)
Uso ideale	Analisi Offline	Real-time Routing, Matrici Urbane