Università degli studi di Pavia
Dipartimento di Matematica
Corso di Laurea Magistrale in Matematica

UNIVERSITÀ
DI PAVIA

Cammini minimi multiobiettivo in reti di trasporto pubblico: Modelli e Algoritmi

Tesi di Laurea Magistrale in Matematica

Relatore:
Prof. Stefano Gualandi
Correlatore:
Gabor Riccardi

Tesi di Laurea di:
Giulia Urso
Matricola 557495

Anno Accademico 2025-2026

Abstract

Developing indices to measure transportation poverty in public transit networks requires algorithms capable of handling massive queries and non-linear fare rules. This thesis addresses these challenges by extending the Raptor algorithm with two key advancements.

Firstly, to reduce the computational costs of repeated recalculations for the Earliest Arrival Problem, a warm-start strategy is introduced. The model reuses pre-calculated optimal journeys by applying spatio-temporal translation logic and a topological lower bound to prune the search space. The implementation on the Milan transit network achieved an average computational saving of over 23%.

Secondly, the research tackles the price-optimal routing problem by adopting Conditional Fare Networks (CFNs), a mathematical framework capable of representing complex real-world fare rules. The architecture was validated by modeling Milan's STIBM fare system using the McRAP algorithm. By subsequently integrating the Tight-BMRAP and Range-McRAP extensions, the model effectively manages the combinatorial explosion, guaranteeing exact Pareto-optimal solutions while significantly reducing running times by filtering redundant alternatives.

Finally, we extend our results on translation and reuse criteria to complex fare networks, formally demonstrating how the warm-start approach can be generalized to compute optimal paths within non-additive domains.

Sommario

Lo sviluppo di indici per misurare la *transportation poverty* nelle reti di trasporto pubblico richiede algoritmi capaci di gestire interrogazioni massive e regole tariffarie non lineari. Questa tesi affronta tali sfide estendendo l'algoritmo Raptor attraverso due contributi principali.

In primo luogo, per abbattere i costi computazionali del ricalcolo continuo per l'*Earliest Arrival Problem*, viene introdotta una strategia di *warm-start*. Il modello riutilizza i viaggi ottimi pre-calcolati sfruttando logiche di traslazione spazio-temporale e un *limite inferiore* topologico per potare lo spazio di ricerca. L'implementazione sulla rete di Milano ha registrato un risparmio computazionale medio superiore al 23%.

In secondo luogo, la ricerca affronta il problema dell'instradamento a prezzo ottimo adottando le *Conditional Fare Networks* (CFN), un framework matematico in grado di rappresentare le complesse regole tariffarie del mondo reale. L'architettura è stata validata modellando il sistema tariffario STIBM di Milano tramite l'algoritmo McRAP. Integrando successivamente le estensioni Tight-BMRAP e Range-McRAP, il modello gestisce efficacemente l'esplosione combinatoria, garantendo soluzioni Pareto-ottime esatte e riducendo significativamente i tempi di esecuzione tramite il filtraggio delle alternative ridondanti.

Infine, estendiamo i nostri risultati sui criteri di traslazione e riuso alle reti tariffarie complesse, dimostrando formalmente come l'approccio *warm-start* possa essere generalizzato per il calcolo dei percorsi ottimi all'interno di domini non additivi.

Indice

Capitolo 1

Introduzione

Negli ultimi decenni, la transizione verso modelli di mobilità urbana più sostenibili è diventata una priorità per le grandi metropoli europee. In questo scenario, l'efficienza del trasporto pubblico non si misura più esclusivamente attraverso la velocità dei collegamenti, ma anche attraverso l'accessibilità economica per l'utente finale. Per un utente, la scelta di un itinerario è un processo multi-obiettivo: il costo del viaggio è spesso un fattore decisionale tanto critico quanto l'orario di arrivo o il numero di cambi necessari. Tuttavia, integrare le strutture tariffarie del mondo reale negli algoritmi di routing rappresenta una sfida computazionale complessa. A differenza delle reti stradali [1], dove il costo è generalmente additivo, i sistemi tariffari pubblici sono caratterizzati da regole non lineari: biglietti a zone, validità temporali limitate e vincoli sui trasferimenti. Tali caratteristiche rendono il problema della ricerca del cammino ottimo un problema di tipo NP-hard [2]. Gli approcci tradizionali basati su variazioni dell'algoritmo di Dijkstra, largamente utilizzati per le reti stradali, mostrano forti limiti nel contesto del trasporto pubblico multicriterio. Per gestire la dimensione temporale e tariffaria, i modelli basati su Dijkstra richiedono la costruzione di grafi complessi (come i grafi *time-expanded* o *time-dependent*). In questi contesti, l'introduzione di vincoli tariffari non lineari e criteri multipli genera un'esplosione combinatoria degli stati del grafo, rendendo i tempi di calcolo proibitivi per applicazioni in tempo reale. Per superare tali limitazioni, la ricerca recente si è orientata verso algoritmi strutturati sugli orari (*timetable-based*) che non necessitano della costruzione di un grafo. Il capostipite di questo paradigma è il framework Round-bAsed Public Transit Optimized Router (RAPTOR), introdotto da Delling, Pajor e Werneck [3], il quale opera attraverso round discreti associati al numero di cambi di mezzo. Questo approccio permette una gestione estremamente efficiente della continuità delle linee e della navigazione degli orari. Successivamente, questo modello è stato esteso per supportare scenari multicriterio (Multi-Criteria RAPTOR (McRAPTOR)) integrando la complessità dei sistemi di bigliettazione reali, prendendo il nome di Multi-Criteria Round-Based Public Transit Routing (McRAP) [4].

Contributi della tesi

Il presente lavoro si inserisce nel filone di ricerca del calcolo dei percorsi ottimi, ponendosi un duplice obiettivo: da un lato, abbattere i costi computazionali associati alle interrogazioni massive (*one-to-all*) tramite logiche algoritmiche di ottimizzazione; dall'altro, sviluppare un sistema capace di navigare la complessità tariffaria reale del trasporto metropolitano, identificando la frontiera di Pareto tra tempo e costo.

Il contributo metodologico principale di questa ricerca risiede nello sviluppo e nella validazione di tecniche di ottimizzazione applicate al framework standard RAPTOR. Al fine di superare il collo di bottiglia generato dal ricalcolo iterativo degli alberi dei cammini minimi, il lavoro formalizza una strategia basata sulla strategia del *warm-start*. Sfruttando un limite inferiore, l'algoritmo è in grado di inibire a priori l'esplorazione di direttrici ridondanti, validando il riuso di percorsi pre-calcolati.

Successivamente, ponendosi l'obiettivo di sviluppare e implementare un sistema di pianificazione dei viaggi capace di navigare la complessità tariffaria reale e di identificare la frontiera di Pareto tra tempo e costo, sono stati studiati algoritmi multi-obiettivo. Per modellare correttamente le tariffe, la tesi adotta il formalismo matematico delle Conditional Fare Networks (CFNs) introdotto in [4]. In questo framework, la struttura tariffaria è rappresentata come un "Grafo dei Biglietti", dove ogni nodo corrisponde a un titolo di viaggio con la relativa tariffa e gli archi rappresentano le transizioni possibili innescate da eventi specifici, come il passaggio di un confine zonale o il superamento della durata del biglietto. La validità del modello è stata testata sulla rete di trasporto della Città Metropolitana di Milano, utilizzando i dataset ufficiali in formato General Transit Feed Specification (GTFS). L'implementazione software, sviluppata in linguaggio Python, integra le regole del Sistema Tariffario Integrato del Bacino di Mobilità (STIBM) [5], gestendo un catalogo di 28 biglietti differenti e mappando oltre 5.000 fermate nelle relative zone tariffarie (da Mi1 a Mi9).

Nello specifico, il lavoro apporta i seguenti contributi logici e sperimentali:

- **Ottimizzazione e Riuso Algoritmico (su base Raptor):** Esplorazione e formalizzazione di quattro logiche di potatura spaziale e temporale (traslazione spaziale all'indietro, traslazione temporale in avanti con assorbimento del ritardo, ottimalità dei sottopercorsi e attesa alla sorgente). L'efficacia di questi criteri è stata validata empiricamente per abbattere i tempi di esecuzione.

- **Modellazione Tariffaria Multi-Obiettivo:** Sviluppo del motore di calcolo McRAP su architettura CFN per il bacino di Milano.

- **Integrazione di Estensioni Algoritmiche:** Viene implementata e valutata la variante Tight Bounded Multi-Criteria RAPTOR (Tight-BMRAP), che introduce un parametro di *slack time* per filtrare soluzioni computazionalmente ridondanti [3, 4]. Inoltre, il sistema viene esteso con la logica Range RAPTOR (rRAPTOR) (dando vita a Range Multi-Criteria

Round-Based Public Transit Routing (Range-McRAP)) per l'elaborazione di viaggi Pareto-ottimi su intervalli temporali continui.

- **Analisi di Scalabilità e Stress Test:** Viene effettuata una valutazione approfondita di come il tempo di calcolo vari all'aumentare del numero di sorgenti simultanee, identificando sperimentalmente i colli di bottiglia computazionali del sistema McRAPquando applicato a reti metropolitane estese.

- **Estensione Teorica al Dominio Tariffario:** Come sviluppo conclusivo, viene tracciato un ponte teorico tra le logiche di potatura e il mondo multi-obiettivo. Si definisce un framework formale per estendere le ottimizzazioni spaziali al dominio dei prezzi.

Struttura della tesi

Il presente elaborato è organizzato come segue:

Il **Capitolo 2** (Revisione della Letteratura) traccia lo stato dell'arte relativo al calcolo dei percorsi ottimi nelle reti di trasporto. Analizza l'evoluzione storica dagli algoritmi tradizionali su grafo (come Dijkstra), ai modelli basati sugli orari per il trasporto pubblico, fino alle moderne sfide dell'ottimizzazione multi-obiettivo e del routing tariffario tramite Conditional Fare Networks.

Il **Capitolo 3** (Algoritmi basati su RAPTOR) descrive nel dettaglio il funzionamento del framework RAPTOR e le sue principali tecniche di potatura (route marking, local pruning, target pruning). Viene inoltre esplorata la gestione delle preferenze di trasferimento, l'integrazione dei costi tariffari (McRAPTOR) e vengono presentati approcci alternativi recenti come ULTRA-Multi-Criteria Trip-Based Routing (McTB).

Il **Capitolo 4** (Modelli di Ottimizzazione e Sviluppo Algoritmico) costituisce il nucleo algoritmico originale della ricerca. Formalizza quattro logiche inedite di potatura per evitare il ricalcolo iterativo dei cammini: la traslazione spaziale all'indietro, la traslazione temporale in avanti con assorbimento del ritardo, l'ottimalità dei sottopercorsi e la traslazione all'indietro con attesa.

Il **Capitolo 5** (Modellazione delle Reti Tariffarie Condizionali ed Estensioni Avanzate del Framework RAPTOR) descrive la fase implementativa. Dettaglia l'architettura computazionale dell'algoritmo McRAP per la gestione delle CFN e descrive l'estensione rRAPTOR per le interrogazioni su intervalli temporali continui, culminando nell'integrazione del modello Range-McRAP e della logica Tight-BMRAP. Inoltre si definisce un metodo per estendere le ottimizzazioni al dominio dei prezzi.

Il **Capitolo 6** (Risultati) presenta la validazione empirica e l'analisi sperimentale del lavoro. Include i test delle logiche di ottimizzazione (prima su una rete fittizia di sviluppo e poi sulla rete reale metropolitana) e analizza le prestazioni dei modelli multi-obiettivo applicati allo STIBM di Milano, valutandone l'efficienza tramite analisi di scalabilità spaziale (scaling test) e stress test di profondità sui round.

Il **Capitolo 7** (Conclusioni) riassume infine i principali risultati ottenuti e delinea le possibili direzioni per i futuri sviluppi della ricerca.

Capitolo 2

Revisione della Letteratura

La letteratura presenta una varietà di metodologie per affrontare il problema del calcolo dei percorsi ottimi nelle reti di trasporto. Nel corso degli anni, la ricerca ha vissuto un'evoluzione dettata dalla necessità di gestire modelli sempre più complessi: partendo dai classici algoritmi su grafi stradali, si è passati alla modellazione degli orari dei mezzi pubblici, fino ad arrivare alle moderne sfide dell'ottimizzazione multi-obiettivo e dell'integrazione delle regole tariffarie reali.

2.1 Il problema dello Shortest Path sulle reti stradali

Il problema del cammino minimo (*shortest path problem*) consiste nell'individuare, all'interno di un grafo pesato, un percorso tra un nodo origine e un nodo destinazione tale da minimizzare la somma dei pesi degli archi che lo compongono. Formalmente, dato un grafo $G = (V, E)$ in cui a ciascun arco è associato un costo o peso non negativo, l'obiettivo è trovare una sequenza di vertici contigui che congiunga due nodi minimizzando il costo complessivo di attraversamento.

I primi algoritmi affrontavano il calcolo dei percorsi modellandolo puramente come un problema di ricerca del cammino minimo su grafi, risolvendolo tramite varianti dell'algoritmo di Dijkstra. Dato un grafo $G = (V, E)$, tale algoritmo opera associando a ogni nodo $v \in V$ un'etichetta $d(v)$, che rappresenta la distanza minima provvisoria dal nodo sorgente s. Inizialmente si pone $d(s) = 0$ e $d(v) = \infty$ per tutti gli altri nodi. A ogni iterazione, l'algoritmo estrae il nodo u non ancora visitato che possiede l'etichetta di costo minore, rendendo tale valore definitivo. Successivamente, esegue l'operazione di rilassamento (*relaxation*) su tutti gli archi uscenti da u: per ogni nodo adiacente v, se $d(u) + w(u, v) < d(v)$ (dove $w(u, v)$ è il peso dell'arco), l'etichetta $d(v)$ viene aggiornata con il nuovo valore minimo.

Dal punto di vista geometrico, l'esplorazione procede in modo radiale e "cieco" attorno alla sorgente. Per ridurre gli elevati tempi di interrogazione su reti estese, questi metodi si sono affidati a tecniche di accelerazione (*speed-up techniques*) calcolate in una fase di pre-elaborazione. Tra le più celebri figurano l'algoritmo A* [6, 7], che introduce una funzione euristica (come, ad esempio, la stima della distanza euclidea) per guidare la ricerca in modo mirato verso la destinazione, e le Contraction Hierarchies (CH) [8], che ordinano i vertici

in base alla loro importanza e bypassano quelli meno rilevanti tramite archi fittizi (*shortcuts*), i quali preservano le distanze minime originali consentendo interrogazioni quasi istantanee. Benché queste metodologie siano ampiamente consolidate ed estremamente efficienti per il routing sulle reti stradali [9], esse si sono rivelate inadeguate per i sistemi di trasporto pubblico.

2.2 Lo Shortest Path nel trasporto pubblico

A differenza delle reti stradali standard, dove l'attraversamento di un nodo (un incrocio) è generalmente privo di penalità, nel trasporto pubblico un algoritmo deve saper distinguere tra un passeggero che rimane a bordo dello stesso veicolo e uno che effettua un trasferimento, evento che comporta una penalità inevitabile in termini di tempo. Il limite intrinseco degli algoritmi tradizionali su grafo è che non possiedono il concetto di "linea": valutano ogni singola fermata come un nodo isolato o un nuovo bivio decisionale, ignorando la continuità del viaggio. Per forzare un grafo a catturare queste dinamiche, i modelli storici si basavano su reti espanse nel tempo (*Time-Expanded*) o dipendenti dal tempo (*Time-Dependent*). In un approccio espanso nel tempo, ogni singolo evento di arrivo e partenza in una stazione deve essere istanziato come un nodo separato, interconnesso da archi di trasferimento artificiali. Questo porta a una massiccia esplosione dimensionale. In alternativa, il modello dipendente dal tempo raggruppa tutte le corse della giornata all'interno di un unico arco tra due fermate consecutive, gestendole attraverso funzioni di peso variabili. In questo approccio, il costo temporale dell'arco dipende dall'esatto istante in cui l'utente raggiunge il nodo, poiché la funzione calcola dinamicamente l'attesa per la vettura successiva. Pur riducendo le dimensioni assolute del grafo, questa soluzione complica gravemente l'esecuzione delle interrogazioni, in quanto costringe l'algoritmo a risolvere complesse funzioni matematiche a ogni avanzamento anziché eseguire semplici somme. [3].

A fronte di queste limitazioni strutturali, la comunità scientifica ha operato un netto cambio di paradigma introducendo gli algoritmi *timetable-based* (basati sugli orari). Tra i primi approcci di successo figura il Connection Scan Algorithm (CSA) [10]. Abbandonando la complessa rappresentazione topologica a grafo, il CSA organizza l'intero orario come un singolo array di collegamenti elementari (definiti da fermata di partenza, fermata di arrivo e rispettivi orari) ordinati cronologicamente. L'algoritmo risolve il problema scansionando questo array in modo lineare una sola volta, aggiornando progressivamente i tempi minimi di arrivo per ciascuna fermata. Questo approccio si è dimostrato estremamente rapido ed efficiente per le interrogazioni a singolo criterio (come il calcolo dell'orario di arrivo minimo).

2.3 L'ottimizzazione multi-obiettivo

Nel contesto del trasporto pubblico, l'ottimizzazione del solo tempo di percorrenza risulta limitante. I passeggeri valutano le opzioni di viaggio soppesando simultaneamente molteplici criteri in competizione tra loro, come la durata

complessiva, il numero di interscambi e il costo del biglietto. Risolvere questo *trade-off* richiede il calcolo di un insieme di soluzioni Pareto-ottime, ovvero alternative in cui nessun criterio può essere migliorato senza peggiorarne un altro. Tuttavia, l'adattamento degli algoritmi a singolo criterio a scenari multi-obiettivo comporta sfide computazionali severe, che si manifestano in modo diverso a seconda dell'architettura di base (grafo o array di orari) ma che portano, in entrambi i casi, a un drastico calo delle prestazioni. Per quanto riguarda i modelli basati su grafo (reti espanse o dipendenti dal tempo), l'estensione canonica è rappresentata dall'approccio Multi-Label-Correcting (MLC). A differenza dell'algoritmo di Dijkstra, dove a ogni nodo u è associato un unico valore scalare (il costo minimo), nel MLC ogni vertice deve mantenere una "borsa" (*bag*) B_u contenente un intero insieme di etichette Pareto-ottime. La debolezza strutturale di questo approccio risiede nella propagazione delle soluzioni: ogni volta che l'algoritmo scopre un nuovo percorso non dominato verso un nodo, quest'ultimo deve essere reinserito all'interno di una complessa coda di priorità. In reti di grandi dimensioni, ciò innesca un effetto a cascata che costringe l'algoritmo a riestrarre e riesaminare più e più volte gli stessi vertici, rendendo i tempi di esecuzione computazionalmente proibitivi. Sul versante opposto, gli approcci *timetable-based* (basati sugli orari) tentano di aggirare i problemi topologici scansionando direttamente i dati delle corse. La variante multicriterio del CSA, nota come Multi-Criteria Connection Scan Algorithm (MCSA), estende la scansione lineare dei collegamenti. Tuttavia, per garantire la correttezza del fronte di Pareto, MCSA non può limitarsi a memorizzare il tempo di arrivo minimo. Al contrario, è costretto a costruire e aggiornare costantemente delle etichette per ogni singola fermata e per ogni specifico istante temporale. Ad ogni connessione scansionata, l'algoritmo deve confrontare le nuove soluzioni con interi insiemi di etichette preesistenti. Questo provoca un sovraccarico massiccio sia in termini di memoria allocata che di operazioni di calcolo. In sintesi, sebbene i due approcci differiscano radicalmente nella loro meccanica (l'esplorazione radiale con code di priorità per il MLC, contro la scansione lineare di array per il MCSA) entrambi soccombono alla stessa criticità: all'aumentare dei parametri di ottimizzazione, il mantenimento e il continuo aggiornamento delle etichette non dominate generano una mole di calcoli insostenibile per applicazioni in tempo reale. Per superare questo ostacolo, Delling et al. [3] hanno introdotto il framework RAPTOR. A differenza del CSA, RAPTOR non scansiona singoli collegamenti elementari, ma elabora direttamente intere tratte (linee). L'algoritmo valuta le soluzioni procedendo per round discreti: ogni round k calcola i viaggi ottimi ottenibili con al massimo k cambi. Questa intuizione ha permesso di gestire la continuità delle linee e le penalità di interscambio in modo nativo ed efficiente, rivelandosi una base altamente scalabile per le estensioni tariffarie.

2.4 Il routing tariffario e le Conditional Fare Networks

Se il passaggio ai modelli basati su round ha risolto efficacemente il routing multi-obiettivo per tempo di viaggio e numero di cambi, l'integrazione delle

strutture tariffarie ha rappresentato a lungo una lacuna critica. Storicamente, il costo del biglietto veniva trattato in modo fortemente semplificato: si assumeva un costo puramente additivo e proporzionale alla distanza chilometrica, oppure si applicava una tariffa fissa per ogni interscambio.

Tali approssimazioni falliscono nell'interfacciarsi con i sistemi di bigliettazione reali, che sono governati da regole non lineari. I sistemi metropolitani moderni includono, ad esempio, biglietti a zone, vincoli di validità temporale (es. 90 minuti dal primo tracciamento), restrizioni direzionali e limiti sui mezzi utilizzabili. In questo contesto, le metriche di costo tradizionali sono inefficaci poiché la validità di un titolo di viaggio dipende dallo storico dell'intero itinerario e non dal singolo arco appena attraversato.

Per colmare questo divario, la ricerca più recente ha formalizzato modelli dedicati, culminati nell'introduzione delle *Conditional Fare Networks* [4]. Questo modello matematico, successivamente integrato nelle estensioni multicriterio come McRAP, permette di svincolare le regole tariffarie dalla topologia della rete stradale, rappresentandole invece attraverso grafi di transizione di stato dei titoli di viaggio. Oggi, l'adozione delle CFN costituisce il punto di riferimento assoluto in letteratura per la risoluzione esatta del problema di routing tariffario reale. Poiché tale architettura rappresenta il nucleo metodologico su cui si fonda parte di questo lavoro di tesi, la sua formalizzazione matematica e il suo funzionamento algoritmico verranno trattati in modo estensivo e dettagliato nel Capitolo 5.

Capitolo 3

Algoritmi basati su RAPTOR

Il presente capitolo introduce definizioni e concetti base al fine di descrivere algoritmi basati su RAPTOR. A differenza dei sistemi stradali, in cui la topologia della rete è statica, il trasporto pubblico è intrinsecamente governato dalla dimensione temporale. Pertanto, la rete di trasporto pubblico non viene modellata semplicemente come un grafo orientato, bensì come una tabella oraria.

Definizione 3.1 (Tabella Oraria)**.** Una rete di trasporto pubblico viene definita come una tupla:

$$\mathcal{N} = (\Pi, V, T, R, F)$$

dove:

- $\Pi \subset \mathbb{N}_0$ rappresenta il **periodo di operatività** (da intendersi, ad esempio, come i secondi che compongono una giornata).
- V è l'insieme delle **fermate** (*stops*). Ciascuna fermata in V corrisponde a una distinta posizione fisica all'interno della rete in cui è possibile salire a bordo o scendere da un veicolo (autobus, tram, treno, ecc.); esempi tipici sono le fermate dell'autobus e i binari ferroviari.
- T è l'insieme delle **corse** (*trips*). Ogni corsa $t \in T$ rappresenta una sequenza di fermate visitate da uno specifico veicolo (ad esempio treno, autobus, metropolitana) lungo una linea. Presso ciascuna fermata della sequenza, il veicolo può far scendere o salire i passeggeri. Inoltre, a ogni fermata p all'interno di una corsa t sono associati gli orari di arrivo e di partenza, indicati rispettivamente con $\tau_{arr}(t,p), \tau_{dep}(t,p) \in \Pi$, con la condizione che $\tau_{arr}(t,p) \leq \tau_{dep}(t,p)$. La prima e l'ultima fermata di una corsa hanno, rispettivamente, l'orario di arrivo e l'orario di partenza non definiti.
- R rappresenta l'insieme delle **linee** (*routes*). Le corse in T sono partizionate in linee: ogni linea in R è costituita dalle corse che condividono la medesima sequenza di fermate e che non si superano. Tipicamente, il numero di corse è nettamente superiore al numero di linee.

- F indica l'insieme dei **percorsi pedonali** (*foot-paths* o trasferimenti). I percorsi pedonali in F modellano i collegamenti a piedi tra le fermate. Ciascun trasferimento è costituito da due fermate p_1 e p_2 a cui è associato un tempo di camminata costante $l(p_1, p_2)$. Si noti che F gode della proprietà transitiva: se p_1 e p_2 sono collegati indirettamente da percorsi pedonali, anche la coppia (p_1, p_2) è contenuta in F.

Definizione 3.2 (Viaggio)**.** L'output prodotto da un qualsiasi algoritmo di pianificazione dei viaggi su una tabella oraria è un insieme di **viaggi** $\mathcal{J}$. Un viaggio $J \in \mathcal{J}$ composto da k corse è definito come una sequenza alternata di fermate, corse e percorsi pedonali:

$$J = (p_1, t_1, p'_1, f_1, p_2, t_2, p'_2, \ldots, f_{k-1}, p_k, t_k, p'_k)$$

dove per ogni indice i:

- $t_i \in T$ rappresenta l'i-esima corsa utilizzata nel viaggio;
- $p_i, p'_i \in V(t_i)$ sono le fermate in cui l'utente rispettivamente sale a bordo (*pick-up*) e scende (*drop-off*), ove $V(t_i)$ denota l'insieme ordinato delle fermate visitate dalla corsa t_i;
- $f_i \in F$ rappresenta il percorso pedonale che collega la fermata di discesa p'_i alla successiva fermata di salita p_{i+1}. Tale definizione include il caso di interscambio all'interno della medesima stazione, modellato come un percorso pedonale $f_i = (p'_i, p'_i)$ con peso $l(f_i)$ pari al tempo tecnico di trasferimento.

Affinché la sequenza rappresenti un itinerario fisicamente e cronologicamente ammissibile, devono essere rispettati i seguenti vincoli temporali:

1. **Validità della corsa:** L'orario di arrivo presso la fermata di discesa deve essere strettamente successivo all'orario di partenza dalla fermata di salita:
$$\tau_{arr}(t_i, p'_i) > \tau_{dep}(t_i, p_i) \quad \forall i \in \{1, \ldots, k\}$$
2. **Validità del trasferimento pedonale:** Per ogni interscambio, l'utente deve avere tempo sufficiente per percorrere il tratto a piedi f_i prima della partenza della corsa successiva:
$$\tau_{arr}(t_i, p'_i) + l(f_i) \leq \tau_{dep}(t_{i+1}, p_{i+1}) \quad \forall i \in \{1, \ldots, k-1\}$$

Si noti che, per costruzione, un viaggio contenente k corse presenta esattamente $k - 1$ trasferimenti pedonali intermedi.

I viaggi generati sono associati a differenti criteri di ottimizzazione. Per gestire il confronto e la selezione tra le soluzioni intermedie e finali, si introduce il concetto di dominanza e si definisce l'insieme dei percorsi ottimi.

Definizione 3.3 (**Dominanza e Insieme di Pareto**)**.**

Dati due viaggi J_1 e J_2 valutati secondo un insieme di n criteri da minimizzare (ad esempio, orario di arrivo e numero di cambi), si dice che il viaggio J_1 **domina** J_2 (indicato con $J_1 \leq J_2$) se J_1 ottiene un risultato minore o uguale a J_2 per ogni singolo criterio considerato.

Formalmente, indicando con $c_i(J)$ il valore dell'i-esimo parametro di costo per il viaggio J, la dominanza si esprime come:

$$J_1 \leq J_2 \iff c_i(J_1) \leq c_i(J_2), \quad \forall i \in \{1, \ldots, n\}.$$

In altre parole, J_1 rappresenta un'alternativa pari o superiore a J_2 sotto ogni aspetto valutato, rendendo J_2 ridondante. Un insieme di viaggi reciprocamente non dominanti costituisce un insieme di Pareto. All'interno degli algoritmi, tali viaggi sono rappresentati tramite etichette associate alle fermate.

Siano $p_s \in V$ una fermata sorgente e $p_t \in V$ una fermata di destinazione. Si definisce con P_{st} l'insieme di tutti i possibili percorsi ammissibili che collegano p_s a p_t. L'obiettivo della pianificazione dei viaggi multi-criterio consiste nel determinare **l'insieme Pareto-ottimo**, ovvero la frontiera globale contenente tutti e soli i viaggi non dominati rispetto all'intero spazio delle soluzioni, denotato come $P^*_{st} \subseteq P_{st}$.

L'applicazione della dominanza appena definita per filtrare le soluzioni, comporta lo scarto di alternative identiche sotto il profilo dei costi. Tuttavia, quando si includono preferenze soggettive dell'utente (come la scelta di effettuare interscambi in località più confortevoli o preferite), l'uso di questa dominanza per potare lo spazio di ricerca risulta un'assunzione troppo aggressiva: essa rischierebbe infatti di scartare soluzioni qualitativamente differenti, sebbene presentino valori identici per i criteri di tempo e numero di cambi [3]. Per ovviare a questo limite e preservare una maggiore varietà topologica di alternative all'interno della frontiera ottimale, la letteratura adotta il criterio della dominanza stretta.

Definizione 3.4 (**Dominanza Stretta**)**.**

Siano J_1 e J_2 due distinti viaggi. Si dice che il viaggio J_1 **domina strettamente** il viaggio J_2 (indicato con $J_1 < J_2$) se e solo se J_1 risulta strettamente migliore in *almeno un* criterio considerato rispetto a J_2.

Considerando l'orario di arrivo τ_{arr} e il numero di cambi k come criteri da minimizzare, J_1 domina strettamente J_2 se:

$$(\tau_{arr}(J_1) < \tau_{arr}(J_2) \wedge k(J_1) \leq k(J_2)) \vee (\tau_{arr}(J_1) \leq \tau_{arr}(J_2) \wedge k(J_1) < k(J_2)).$$

Dal punto di vista logico, la dominanza stretta è concettualmente più restrittiva nella sua applicazione rispetto a quella standard: richiedendo un miglioramento netto in almeno un criterio per poter scartare un'alternativa, essa elimina un minor numero di soluzioni. Di conseguenza, la dominanza stretta genera insiemi di Pareto più ampi e inclusivi.

La motivazione principale dietro l'adozione della dominanza stretta risiede nella necessità di non perdere in partenza itinerari che utilizzano hub di scambio preferiti dall'utente; tali alternative vengono mantenute nella frontiera allargata per essere poi valutate e filtrate in una fase successiva di post-elaborazione.

Introduciamo inoltre il concetto di Viaggio concatenato che verrà utilizzato più volte nel corso della tesi. Essendo un viaggio strutturato matematicamente come una sequenza alternata e ordinata di fermate e corse, è possibile unire due itinerari contigui mediante semplice giustapposizione.

Definizione 3.5 (Viaggio Concatenato).
Siano J_1 e J_2 due viaggi

$$J_1 = (p_1, t_1, p'_1, f_1, \ldots, p_n, t_n, p'_n), \quad J_2 = (q_1, u_1, q'_1, g_1, \ldots, q_m, u_m, q'_m).$$

Si supponga che i due viaggi condividano un nodo di aggancio p, tale per cui la fermata di discesa finale di J_1 coincida con la fermata di salita iniziale di J_2 ($p'_n = q_1 = p$).

Per congiungere i due itinerari è necessario introdurre nella sequenza il percorso pedonale di interscambio $f_c = (p, p) \in F$, il cui peso $l(f_c) = \Delta_{trans}(p)$ modella il tempo tecnico di trasferimento in stazione. La concatenazione è definita ammissibile se rispetta il vincolo:

$$\tau_{arr}(t_n, p) + l(f_c) \leq \tau_{dep}(u_1, p).$$

Se la condizione temporale è verificata, si definisce **viaggio concatenato** $J_{concat} = J_1 J_2$ l'itinerario complessivo di $n + m$ corse ottenuto fondendo le due sequenze tramite il trasferimento f_c:

$$J_{concat} = (p_1, t_1, p'_1, f_1, \ldots, p_n, t_n, p, f_c, p, u_1, q'_1, g_1, \ldots, q_m, u_m, q'_m). \tag{3.1}$$

Qualora la corsa di discesa di J_1 sia identica alla corsa di salita di J_2 ($t_n = u_1 = t$), l'utente non effettua un trasbordo. Il nodo p diviene un semplice punto di transito, la penalità di interscambio si annulla e la sequenza si riduce a un'unica corsa continua:

$$J_{concat} = (p_1, t_1, p'_1, f_1, \ldots, p_n, \mathbf{t}, q'_1, g_1, \ldots, q_m, u_m, q'_m). \tag{3.2}$$

3.1 L'algoritmo RAPTOR

Per superare i limiti strutturali degli approcci basati su grafi classici, Delling et al. [3] hanno formalizzato il framework RAPTOR. A differenza delle varianti di Dijkstra, questo modello non fa uso di code di priorità e non elabora la rete come un reticolo di nodi indipendenti. Il sistema opera direttamente sulle tabelle orarie, sfruttando la continuità logica delle linee di trasporto e procedendo per iterazioni (o *round*) successive, durante le quali la frontiera delle soluzioni ottime viene estesa progressivamente.

3.1.1 L'algoritmo base

Per calcolare il viaggio ottimo da una sorgente $p_s \in V$ a una destinazione $p_t \in V$ con partenza all'istante $\tau \in \Pi$, il framework procede iterativamente. L'obiettivo di una singola iterazione k è individuare l'orario di arrivo minimo per ogni nodo della rete impiegando al massimo k corse (ovvero $k-1$ interscambi) [3].

A livello di strutture dati, ogni fermata p è dotata di un vettore temporale $(\tau_0(p), \tau_1(p), \ldots, \tau_K(p))$, dove ciascun elemento $\tau_i(p)$ memorizza l'arrivo più rapido noto utilizzando esattamente fino a i mezzi. La configurazione di partenza prevede l'inizializzazione di tutti i valori a ∞, configurando unicamente l'origine con $\tau_0(p_s) = \tau$ ([3]).

La logica di RAPTOR si fonda su una specifica invariante: all'avvio del round k (con $k \geq 1$), tutti i tempi calcolati fino alla fase $k-1$ sono da considerarsi definitivi e corretti. La costruzione delle etichette $\tau_k(p)$ si articola quindi in tre passaggi operativi [3]:

1. **Inizializzazione:** Il sistema copia i valori della fase precedente ponendo $\tau_k(p) = \tau_{k-1}(p)$ per l'intero dominio spaziale. L'assunto logico alla base è che concedere al passeggero l'utilizzo di un veicolo addizionale non deve mai produrre un orario di destinazione peggiore rispetto a un itinerario con un numero inferiore di cambi [3].

2. **Attraversamento delle Linee (Route Traversal):** Il modello valuta singolarmente ogni linea $r \in R$. Considerando l'insieme cronologico delle corse $T(r)$ assegnate a tale direttrice, l'algoritmo ricerca il primo veicolo utile per l'imbarco presso una data fermata p_i. Tale vettura, definita tramite la funzione $et(r, p_i)$, corrisponde alla corsa più anticipata t in grado di soddisfare il vincolo cronologico $\tau_{dep}(t, p_i) \geq \tau_{k-1}(p_i)$.

   Si noti che questa disequazione non integra alcuna tolleranza intrinseca per i tempi fisici di trasferimento. Infatti, ogni interscambio (incluso quello tecnico tra i binari della stessa struttura) è delegato unicamente ai percorsi pedonali: si modella un arco ricorsivo $(p_i, p_i) \in F$ il cui peso assorbe il tempo di trasbordo ([3]). Tale penalità verrà applicata solo nel terzo passaggio dell'algoritmo.

   Individuata la fermata di imbarco p_i, il sistema avanza lungo la sequenza delle fermate p_j della linea r, registrando l'orario di arrivo se la vettura t offre un miglioramento ($\tau_k(p_j) = \min\{\tau_k(p_j), \tau_{arr}(t, p_j)\}$). Per garantire la successiva ricostruzione dell'itinerario a ritroso, viene salvato un puntatore di derivazione spaziale (*parent pointer*) che indica il punto esatto di salita p_i [3].

   Durante il tragitto, il software monitora costantemente la possibilità di anticipare il viaggio. Se nei round passati era emersa un'opzione più veloce per raggiungere un nodo intermedio ($\tau_{k-1}(p_i) < \tau_{arr}(t, p_i)$), la corsa corrente viene ricalcolata: l'algoritmo aggiorna la funzione di imbarco $et(r, p_i)$ facendola ripartire dal nuovo e migliore orario di arrivo ($\tau_{k-1}(p_i)$). Questo passaggio permette di verificare se, trovandosi in banchina prima

del previsto, è possibile intercettare un veicolo precedente della stessa linea r, massimizzando l'efficienza complessiva dell'instradamento [3].

3. **Percorsi pedonali (Foot-paths):** L'ultima fase processa gli spostamenti a piedi tra stazioni diverse o interne allo stesso nodo. Per ogni arco $(p_i, p_j) \in F$, l'etichetta di destinazione viene minimizzata confrontandola con il tempo di arrivo maggiorato della camminata: $\tau_k(p_j) = \min\{\tau_k(p_j), \tau_k(p_i) + l(p_i, p_j)\}$. La natura transitiva dell'insieme F assicura in un solo passaggio l'individuazione dell'attraversamento pedonale più efficiente [3].

La procedura globale termina non appena si conclude un round k privo di miglioramenti paretiani su qualsiasi etichetta. Poiché ogni direttrice e corsa viene analizzata al massimo una singola volta per iterazione, il costo di esecuzione scala in modo puramente lineare ed è limitato asintoticamente da $O(K(\sum_{r \in R} |r| + |T| + |F|))$, dove K rappresenta il numero massimo di round, $|r|$ indica il numero di fermate servite dalla linea r, mentre $|T|$ e $|F|$ denotano rispettivamente il numero totale di corse e di percorsi pedonali presenti nella rete. Questa fluidità computazionale, dovuta all'assenza di complesse code di priorità, garantisce prestazioni nettamente superiori rispetto alle metodologie basate sui grafi [3].

3.1.2 Ottimizzazioni algoritmiche

L'esecuzione iterativa del framework su reti estese richiede l'adozione di tecniche di potatura (*pruning*) per circoscrivere efficacemente lo spazio di ricerca. Il modello base viene pertanto esteso dagli autori tramite tre meccanismi fondamentali [3]:

- **Marcatura delle Linee (*Route Marking*):** L'esplorazione di una direttrice r durante il round k si innesca esclusivamente se tale linea attraversa almeno un nodo p_i il cui tempo di arrivo sia stato ottimizzato nel round precedente (ossia quando $\tau_{k-1}(p_i) < \tau_{k-2}(p_i)$). In assenza di questo miglioramento stringente, la valutazione riprodurrebbe cammini già consolidati aggiungendovi un interscambio fittizio, in netta violazione dei criteri di dominanza. Operativamente, l'algoritmo contrassegna le fermate aggiornate nel round $k-1$; all'avvio dell'iterazione k, aggrega queste ultime per isolare l'insieme Q delle rotte attive. L'attraversamento di una linea in Q inizierà rigorosamente dalla prima fermata marcata utile lungo la sequenza.

- **Potatura Locale (*Local Pruning*):** L'algoritmo mantiene il parametro $\tau^*(p)$, incaricato di storicizzare il tempo di arrivo minimo assoluto per la fermata p, indipendentemente dall'iterazione corrente. L'aggiornamento e la conseguente marcatura di un nodo si attivano unicamente se la nuova stima è strettamente inferiore a $\tau^*(p)$. Tale logica rende superflua la procedura di copia incondizionata di τ_{k-1} in τ_k all'inizio di ogni round.

- **Potatura della Destinazione (*Target Pruning*):** Parallelamente, il sistema traccia il miglior orario di arrivo $\tau^*(p_t)$ relativo al traguardo finale. Qualsiasi fermata intermedia p che registri un tempo parziale già eccedente la soglia $\tau^*(p_t)$ viene esclusa dalla marcatura, inibendo l'espansione di direttrici palesemente sub-ottimali.

3.1.3 Preferenze di trasferimento

L'integrazione delle preferenze soggettive dell'utente (come la priorità verso specifici hub di interscambio) richiede l'impiego della dominanza stretta (seguendo la Definizione 3.4), la quale fisiologicamente accresce le dimensioni dell'insieme di Pareto preservando varianti topologiche a parità di costo temporale.

Per supportare tale funzione senza saturare la memoria, il framework associa al *parent pointer* dell'etichetta $\tau_k(p)$ non un nodo d'imbarco casuale, ma la fermata che massimizza il punteggio di preferenza tra quelle compatibili con la corsa t. Qualora sia richiesto un tracciamento esaustivo rigoroso, l'algoritmo sostituisce il singolo puntatore con una lista di riferimenti a tutti i nodi di origine p' in cui il trasbordo risulta cronologicamente fattibile, verificando costantemente la condizione $\tau_{k-1}(p') \leq \tau_{dep}(t, p')$.

3.1.4 Prestazioni ed estensioni algoritmiche

La struttura modulare del calcolo per round agevola lo sviluppo di espansioni avanzate, primariamente rRaptor per le *range queries* e McRAPTOR per l'instradamento multicriterio.

rRaptor.

Concepito per processare interrogazioni su una finestra temporale continua $\Delta \subseteq \Pi$, l'algoritmo estrae preventivamente l'insieme Ψ degli orari di partenza disponibili dall'origine p_s. Il problema ha due obiettivi: minimizzare il tempo di arrivo τ_{arr} massimizzando simultaneamente il momento della partenza τ_{dep}.

Definendo il vettore di costo $c(J) = (-\tau_{dep}(J), \tau_{arr}(J))$ e seguendo la Definizione 3.3, un itinerario J_1 rende un itinerario J_2 dominato se e solo se garantisce una partenza ritardata o identica ($-\tau_{dep}(J_1) \leq -\tau_{dep}(J_2)$) congiunta a un arrivo anticipato o coincidente ($\tau_{arr}(J_1) \leq \tau_{arr}(J_2)$).

Per sopprimere le computazioni ridondanti, rRaptor processa l'insieme Ψ applicando un ordinamento decrescente (*Latest-to-Earliest*). L'efficienza strutturale risiede nel mantenimento persistente della matrice delle etichette $\tau_k(p)$ tra un'esecuzione e l'altra: non subendo reinizializzazioni a ∞, i valori in memoria rappresentano i tempi minimi assicurati dai viaggi avviati negli istanti futuri.

Questa logica altera l'invariante di base del framework. Nel Raptor classico, l'apertura del round k impone una sovrascrittura incondizionata $\tau_k(p) = \tau_{k-1}(p)$ valida per ogni $p \in V$. In rRaptor, tale assegnazione distruggerebbe i record storici; pertanto, l'aggiornamento viene eseguito unicamente a fronte di un effettivo miglioramento ($\tau_{k-1}(p) < \tau_k(p)$).

Durante la scansione si configurano quindi due scenari:

- **Miglioramento Accertato:** Se $\tau_{k-1}(p) < \tau_k(p)$, la partenza anticipata corrente offre un reale guadagno temporale rispetto ai viaggi futuri. Il sistema sovrascrive l'etichetta e marca la fermata per propagare l'esplorazione.

- **Inefficienza e Potatura Automatica:** Se $\tau_{k-1}(p) \geq \tau_k(p)$, il viaggio anticipato risulta obsoleto, poiché una partenza successiva garantisce un arrivo migliore o equivalente. L'algoritmo blocca la sovrascrittura e nega la marcatura al nodo, tranciando istantaneamente le direttrici uscenti tramite un meccanismo di *self-pruning*.

McRAPTOR.

L'architettura multi-obiettivo introdotta da Delling et al. [3] supera il limite del RAPTOR standard, il quale è vincolato a mantenere un singolo orario di arrivo $\tau_k(p)$ per ciascuna fermata. Al fine di ottimizzare criteri multipli, gli autori sostituiscono il singolo valore scalare con un contenitore di etichette (*bag*) $B_k(p)$, deputato a raccogliere tutte le soluzioni mutuamente non dominate individuate durante il round k.

Questa generalizzazione impone una modifica nella procedura di rilassamento delle linee. Quando il sistema elabora una direttrice r, viene inizializzato un contenitore temporaneo B_r, all'interno del quale ogni etichetta L conserva un riferimento alla specifica corsa attiva $t(L)$ su cui il passeggero è a bordo. L'esplorazione procede quindi valutando sequenzialmente le fermate della linea attraverso tre passaggi operativi [3]:

1. I tempi di arrivo delle etichette correntemente in B_r vengono aggiornati in base all'orario in cui la corsa $t(L)$ raggiunge la fermata p, filtrando eventuali soluzioni divenute sub-ottimali.

2. Il contenuto del *route bag* B_r viene integrato nel contenitore della fermata $B_k(p)$, scartando rigorosamente le etichette dominate per preservare l'esattezza della frontiera paretiana.

3. I risultati validi consolidati nel round precedente, presenti in $B_{k-1}(p)$, vengono riversati nel contenitore temporaneo B_r, associando loro le nuove corse disponibili per consentire ulteriori diramazioni del viaggio.

La medesima logica viene applicata all'elaborazione dei percorsi pedonali: per ogni trasferimento (p_i, p_j), l'algoritmo duplica le etichette presenti in $B_k(p_i)$, incrementa i loro tempi di arrivo del valore $l(p_i, p_j)$ e le unisce al *bag* della fermata di destinazione $B_k(p_j)$ [3].

Le strategie di *pruning* spaziale e temporale vengono anch'esse riadattate. Il limite storico assoluto τ^*, tipico della versione base, evolve in un *best bag* $B^*(p)$ che storicizza il profilo di Pareto globale della fermata. Di conseguenza, l'inserimento di una nuova etichetta L in $B_k(p)$ è subordinato a un controllo di dominanza incrociato: la soluzione viene potata se risulta dominata dal profilo storico locale $B^*(p)$ o dal profilo della destinazione finale $B^*(p_t)$. In caso contrario, l'etichetta viene convalidata e il registro $B^*(p)$ viene opportunamente aggiornato [3].

Esempio 3.1 (Zone tariffarie.)**.** Un'applicazione del modello McRAPTOR, discussa dagli stessi autori [3], riguarda l'instradamento basato su bacini tariffari. Poiché il calcolo del prezzo esatto in tempo reale presenta spesso forti inefficienze computazionali, una strategia efficace consiste nell'utilizzare l'insieme delle zone visitate Z come criterio di ottimizzazione intermedio, delegando il calcolo monetario esatto a una successiva e rapida fase di post-elaborazione.

Sotto questo paradigma, la topologia assegna a ciascuna fermata p una o più zone $z(p)$. L'etichetta del viaggio assume la struttura $L = (\tau(L), z(L))$, dove $z(L)$ tiene traccia dell'accumulo zonale. La regola di dominanza si modula di conseguenza: una soluzione L_1 domina L_2 se garantisce un arrivo uguale o anticipato ($\tau(L_1) \leq \tau(L_2)$) avendo attraversato un sottoinsieme di zone minore o uguale ($z(L_1) \subseteq z(L_2)$). In fase di avvio, l'origine viene inizializzata con $(\tau, z(p_s))$ e, procedendo nell'esplorazione, l'insieme $z(L)$ viene espanso unendovi sistematicamente l'identificativo zonale di ogni nuova fermata incontrata lungo il tragitto ($z(L) \cup z(p)$).

3.1.5 Applicazioni

L'applicazione delle architetture algoritmiche alla pianificazione urbana richiede prestazioni computazionali in grado di supportare l'analisi geospaziale in tempo reale. A tale scopo, la ricerca condotta da Conway et al. [11] ha dimostrato la fattibilità dell'impiego del framework RAPTOR all'interno di piattaforme di *sketch planning*. Questa particolare metodologia di progettazione preliminare si basa sulla creazione e sul confronto rapido di molteplici scenari infrastrutturali alternativi. Lo *sketch planning* esige un feedback quantitativo quasi istantaneo, idealmente nell'ordine di pochi secondi o minuti, al fine di consentire ad amministratori, pianificatori e stakeholder di valutare l'impatto delle proprie idee in modalità collaborativa direttamente durante le sessioni di co-progettazione.

Il parametro scelto dagli autori per misurare l'efficacia di una rete è l'accessibilità cumulativa spaziale, un indicatore in grado di mantenere il rigore matematico pur essendo facilmente interpretabile. Nello specifico, il sistema quantifica il volume complessivo di destinazioni utili (quali, ad esempio, i poli occupazionali) a cui un utente può accedere partendo da un determinato nodo e rispettando un vincolo temporale massimo, tipicamente fissato a 45 minuti di tragitto [11].

La manipolazione interattiva del grafo dei trasporti non si esaurisce nell'aumento delle frequenze di passaggio su servizi già esistenti. Il framework consente infatti l'applicazione di un set di trasformazioni topologiche e operative codificate, che comprendono:

- La progettazione di linee nuove sia all'interno del grafo stradale che ferroviario;
- La riassegnazione dei percorsi per deviare le linee attuali verso nuovi bacini di utenza;
- L'alterazione dei parametri cinematici dei mezzi, intervenendo direttamente sulla velocità media e sui tempi di permanenza in stazione;

- La soppressione di corse specifiche o la dismissione di nodi di sosta valutati come non efficienti.

La validazione di tali varianti di rete incontra un ostacolo insito nella natura del trasporto pubblico: la forte variabilità dell'accessibilità rispetto all'istante di partenza dell'utente, fortemente condizionata dall'allineamento delle coincidenze. Al fine di neutralizzare questa dipendenza stocastica dal singolo minuto di indagine, l'architettura sfrutta l'estensione rRaptor per processare finestre di partenza continue. Nell'aggregazione dei risultati temporali, gli autori scartano la media aritmetica in favore del tempo di percorrenza mediano. Tale approccio garantisce la corretta gestione dei viaggi anomali e delle destinazioni irraggiungibili: assegnando a queste ultime valore di durata infinito all'interno dell'orizzonte temporale, il calcolo della mediana non subisce distorsioni, poiché questi valori estremi vengono naturalmente confinati oltre il limite temporale accettabile [11].

Un'ulteriore criticità modellistica sorge dall'eterogeneità delle specifiche di rete: mentre l'infrastruttura consolidata dispone di orari esatti, le direttrici in fase di progettazione vengono tipicamente dimensionate solo in base alla frequenza di passaggio desiderata. Per fondere queste due logiche all'interno di un motore *timetable-based*, gli autori introducono un approccio statistico di tipo simulazione di Monte Carlo. Vengono generati circa 1000 orari randomizzati per le nuove linee che sono distribuiti uniformemente nella finestra temporale di partenza [11].

Considerato l'elevato carico computazionale derivante dall'estrazione di mille varianti topologiche, l'algoritmo ricorre a una tecnica di delimitazione dello spazio di ricerca. L'elaborazione viene scissa in due fasi gerarchiche:

- Inizialmente, rRaptor scansiona unicamente il sottografo contenente le corse a orario fisso, ovvero quello formato dai servizi esistenti. Poiché l'immissione di nuovi servizi può, per definizione matematica, solo abbattere i tempi di viaggio, la soluzione deterministica originaria agisce da limite superiore.

- Tale limite temporale superiore viene successivamente innestato nell'esplorazione del grafo completo (contenente le reti stocastiche), abilitando una drastica potatura dei rami inefficienti e abbattendo i tempi di esecuzione del processore.

- L'output finale, pertanto, non restituisce uno scalare puntuale, bensì una distribuzione probabilistica continua, offrendo agli analisti uno strumento statisticamente solido per discriminare e soppesare i diversi piani di sviluppo urbanistico.

3.2 Integrazione dei costi

Sebbene le estensioni dell'algoritmo Raptor abbiano consentito avanzate analisi di accessibilità temporale e spaziale, la maggior parte dei modelli convenzionali valuta l'accesso basandosi esclusivamente sul tempo di viaggio, ignorando

il costo monetario delle tariffe. Integrare i vincoli tariffari è tuttavia fondamentale per valutare l'equità sociale e l'accessibilità reale. Per risolvere il problema della connessione più economica in modo esatto e simultaneo al tempo di viaggio, Conway e Stewart [12] hanno proposto un metodo di ottimizzazione multi-obiettivo basato sul framework McRAPTOR.

3.2.1 Formalizzazione delle indennità di trasferimento

L'ostacolo computazionale principale nell'instradamento tariffario è che un prefisso di viaggio costoso potrebbe produrre il viaggio complessivo più economico grazie agli incentivi sui cambi. L'indennità di trasferimento rappresenta quindi il risparmio monetario potenziale su una corsa futura garantito dall'aver utilizzato una specifica sequenza di mezzi in precedenza. Siano J_1 e J_2 due prefissi di viaggio (ossia dei viaggi che raggiungono un nodo x), e J_3 un suffisso di viaggio (ovvero un viaggio che collega x alla destinazione finale). L'indennità $a_{J_1 J_3}$ per il prefisso J_1 e il suffisso J_3 è definita dagli autori [12] come:

$$a_{J_1 J_3} = f_{J_1} + f_{J_3} - f_{J_1 J_3},$$

dove f_{J_1} è la tariffa cumulativa del prefisso, f_{J_3} è la tariffa intera del suffisso (senza sconti) e $f_{J_1 J_3}$ è la tariffa effettiva del viaggio combinato. La massima indennità di trasferimento per un prefisso J_1 sull'insieme di tutti i possibili suffissi $\mathcal{U}$ aventi origine in x è definita [12] come:

$$a_{J_1} = \max_{J_3 \in \mathcal{U}} \{a_{J_1 J_3}\}.$$

Questa formulazione presuppone la non negatività delle indennità ($a_{J_1 J_3} \geq 0$), implicando che un utente non venga mai penalizzato per l'utilizzo di un titolo di viaggio di trasferimento rispetto al pagamento di una nuova tariffa intera [12].

3.2.2 Regole di dominanza per la ricerca Pareto-Ottima

All'interno di McRAPTOR, un prefisso J_2 può essere potato (*pruned*) a favore di J_1 solo se J_1 garantisce un costo non superiore per qualsiasi possibile suffisso futuro. Poiché valutare analiticamente ogni suffisso $\mathcal{U}$ è computazionalmente proibitivo, gli autori introducono criteri di dominanza conservativi [12]:

- **Dominanza Tariffaria Forte:** Un prefisso J_1 domina J_2 se la tariffa cumulativa di J_1 è inferiore o uguale alla tariffa di J_2 al netto della sua miglior indennità futura [12]:

$$f_{J_1} \leq f_{J_2} - a_{J_2}.$$

- **Dominanza per Classi di Indennità:** Per sistemi che offrono cambi gratuiti illimitati, i prefissi vengono confrontati all'interno di "classi di indennità". J_1 domina J_2 se $f_{J_1} \leq f_{J_2}$ e se J_1 garantisce un'indennità $a_{J_1 J_3} \geq a_{J_2 J_3}$ per ogni scenario J_3 [12].

Tabella 3.1: Notazione matematica e simboli utilizzati nel modello tariffario di Conway e Stewart [12].

Simbolo	**Definizione**
J_1, J_2	Prefissi di viaggio (ovvero combinazioni di corse di trasporto pubblico) che raggiungono un determinato punto x.
J_3	Suffisso di viaggio (ovvero combinazione di corse di trasporto pubblico) che conduce da x a una qualsiasi altra destinazione nella rete.
J_1J_3, J_2J_3	Viaggi completi dall'origine a una destinazione, formati collegando J_1 o J_2 con J_3.
$\mathcal{U}$	Insieme di tutti i suffissi di viaggio che hanno origine nel punto x.
f_{J_1}, f_{J_2}	Tariffa cumulativa rispettivamente per il prefisso di viaggio J_1 o J_2.
t_{J_1}, t_{J_2}	Orario di arrivo al punto x rispettivamente per il prefisso di viaggio J_1 o J_2.
$f_{J_1J_3}, f_{J_2J_3}$	Tariffa pagata rispettivamente per il prefisso di viaggio J_1 o J_2, seguito dal suffisso di viaggio J_3.
f_{J_3}	Tariffa intera per il suffisso di viaggio J_3, assumendo l'assenza di sconti dovuti a corse precedenti.
$a_{J_1J_3}, a_{J_2J_3}$	Indennità di trasferimento (*transfer allowance*), ovvero lo sconto sulla tariffa intera f_{J_3}, dovuto al fatto di aver precedentemente percorso rispettivamente il prefisso J_1 o J_2.
a_{J_1}, a_{J_2}	Sconto massimo su un qualsiasi suffisso di viaggio che può essere ottenuto avendo percorso rispettivamente il prefisso J_1 o J_2.

3.2.3 Dettagli implementativi: limitazione superiore e campionamento stocastico

Per rendere il sistema McRAPTOR computazionalmente sostenibile senza sacrificare l'accuratezza della frontiera di Pareto e per garantire l'efficienza in scenari di pianificazione interattiva, vengono adottate due strategie fondamentali [12]:

- **Limitazione superiore della validità temporale:** In presenza di tariffe basate su finestre temporali (come lo STIBM di Milano o il sistema Massachusetts Bay Transportation Authority (MBTA) di Boston), la dominanza tra due etichette non dipende solo dal costo sostenuto f, ma anche dal tempo di validità residuo del titolo di viaggio. Se un'etichetta J_1 arriva dopo J_2 ($\tau_{J_1} > \tau_{J_2}$) ma possiede un biglietto con una scadenza temporale molto più estesa, J_1 non può essere dominata.

  Tuttavia, per evitare un'esplosione del numero di etichette non dominate, si introduce una funzione di *capping* $C(t, \Delta)$:

  $$\hat{t}_{res} = \min(t_{res}, T_{max} - \tau),$$

  dove t_{res} è il tempo residuo del biglietto e $(T_{max} - \tau)$ è l'orizzonte temporale rimanente della query. Se il biglietto scade oltre l'orizzonte della query, il tempo in eccesso viene troncato (capping), rendendo i prefissi nuovamente confrontabili e permettendo la potatura di soluzioni ridondanti.

- **Campionamento di Monte Carlo per l'incertezza delle frequenze:** Nelle fasi di pianificazione strategica, non tutti i servizi sono definiti da orari fissi; molti sono caratterizzati solo da una frequenza media. Per modellare l'accessibilità economica in questi contesti, il framework non calcola un unico valore, ma stima una distribuzione di probabilità attraverso un approccio Monte Carlo.

L'efficacia del modello è stata validata sulla rete MBTA di Boston [12], caratterizzata da zone tariffarie complesse . L'analisi ha dimostrato come l'approccio multi-obiettivo sia essenziale per identificare correttamente la frontiera di Pareto: ad esempio, percorsi che richiedono il doppio del tempo (autobus multipli) ma con costi ridotti dell'80% rispetto al treno suburbano [12]. Il sistema ha mostrato elevate capacità di scalabilità, completando analisi regionali su larga scala in tempi compatibili con l'utilizzo in ambito urbanistico e decisionale.

3.3 Un approccio alternativo: ULTRA-McTB

Un'evoluzione recente nel campo del routing multicriterio è rappresentata dall'algoritmo McTB, proposto da Potthoff e Sauer per gestire reti multimodali su scala nazionale [13]. A differenza dei framework basati su RAPTOR, che operano sulle fermate, McTB si fonda sul paradigma del **Trip-Based Routing**, focalizzando il calcolo sugli "eventi di fermata" (scambi tra singole corse pre-calcolate) anziché sulla scansione ciclica delle linee.

L'algoritmo introduce il tempo di cambio (inteso come sforzo fisico e durata del trasferimento) come terzo criterio di ottimizzazione, superando i limiti dei modelli biobiettivo che spesso trascurano il disagio dei trasferimenti pedonali illimitati. Per gestire tale complessità senza l'esplosione computazionale tipica degli insiemi di Pareto ad alta dimensionalità, McTB si appoggia alla tecnica di pre-elaborazione **ULTRA** [14], [15], che riduce la rete dei trasferimenti a un insieme compatto di scorciatoie essenziali [13].

Una caratteristica distintiva di McTB è la capacità di ottimizzare tre criteri senza dover mantenere borse di etichette dinamiche ad ogni nodo, riducendo drasticamente l'overhead delle strutture dati [13]. Per massimizzare le prestazioni in scenari interattivi, gli autori introducono il concetto di Insieme di Pareto Ristretto [16], che filtra le soluzioni ridondanti attraverso margini di tolleranza (*slack*) calcolati rispetto a un viaggio di riferimento, detto "di ancoraggio" [13].

Dal punto di vista algoritmico, McTB impone rigorosi invarianti sul tempo di cambio per potare lo spazio di ricerca, garantendo che ogni evento di fermata mantenga solo il valore ottimo per il terzo criterio [13]. Sebbene questo approccio garantisca velocità di interrogazione fino a 80 volte superiori ai metodi tradizionali su reti molto estese, esso richiede una fase di pre-elaborazione intensiva per la generazione delle scorciatoie.

Il punto di arrivo attuale per la modellazione tariffaria esatta è rappresentato dalle Conditional Fare Networks introdotte in [4]. Questo framework supera i limiti dei modelli precedenti separando la rete stradale dalla logica dei biglietti. Sarà proprio questo il modello matematico adottato come base per questa tesi, al fine di applicarlo alla rete di Milano e, soprattutto, di estenderlo con nuove logiche di ottimizzazione computazionale.

Capitolo 4

Modelli di ottimizzazione e sviluppo algoritmico

4.1 Ottimizzazioni algoritmiche

Come illustrato nei capitoli precedenti, l'implementazione delle architetture basate sul framework RAPTOR ha consentito la risoluzione efficiente del problema di instradamento sulla complessa rete metropolitana di Milano. Tuttavia, l'applicazione di tali algoritmi in scenari analitici su larga scala impone sfide computazionali non trascurabili.

In diversi ambiti della pianificazione dei trasporti, quali la valutazione dell'accessibilità spaziale o il calcolo di indici per la misurazione della *transportation poverty* [17], l'indagine non si esaurisce nella singola interrogazione da un nodo sorgente (*one-to-all*). Tali metodologie richiedono tipicamente l'esecuzione di calcoli intensivi e matriciali di tipo *all-to-all* o *many-to-many*. In questi contesti operativi, il sistema è chiamato a processare in rapida successione moli di query caratterizzate da variazioni marginali, ovvero lievi scostamenti spaziali (fermate limitrofe) o temporali (orari di partenza contigui). Affrontare questo carico computazionale mediante un approccio standard, che impone la reinizializzazione totale dello spazio degli stati e il ricalcolo iterativo per ciascuna interrogazione, risulta gravemente inefficiente e proibitivo.

Per agevolare la comprensione dei modelli e delle formulazioni matematiche presentate nelle sezioni successive, si rimanda alla Tabella 4.1, posta a conclusione del presente capitolo, la quale raccoglie e definisce la notazione formale e i simboli utilizzati.

4.1.1 La strategia di Warm-Start e il limite inferiore topologico

Per superare questo collo di bottiglia strutturale, il presente capitolo formalizza una strategia di ottimizzazione inedita basata sul *warm-start*. Lo scopo di tale tecnica consiste nel riutilizzare come punto di partenza i risultati di un'esplorazione pregressa all'interno dello spazio di ricerca della nuova interrogazione.

Nello specifico, dato un insieme P_b^* di percorsi ottimi in partenza da una fermata p_b, il modello mira a riutilizzare un sottoinsieme di viaggi $P^* \subseteq P_b^*$ già calcolati. Applicando opportune operazioni di traslazione spazio-temporale, l'algoritmo può estrapolare soluzioni valide per le query adiacenti. Questo meccanismo di *warm-start* funge da innesco per logiche di potatura (*pruning*) restrittive, capaci di inibire a priori l'esplorazione ridondante di ampie porzioni della rete.

Tuttavia, affinché l'adozione di un percorso traslato P^* preservi l'esattezza matematica della frontiera di Pareto senza invocare il riutilizzo dell'algoritmo originario, è necessario dimostrare analiticamente l'assenza di alternative dominanti. Tale condizione di ottimalità globale viene certificata mediante l'introduzione di un limite inferiore topologico H. Durante la valutazione di qualsiasi altra corsa o direttrice emergente, il sistema somma l'orario di partenza della nuova alternativa al limite inferiore H per la destinazione richiesta, ottenendo il miglior tempo di arrivo teoricamente possibile. Qualora questo tempo teorico ottimo risulti maggiore o uguale al costo temporale già garantito dalla soluzione in memoria P^*, la corsa emergente viene dominata. Di conseguenza, il ramo di esplorazione viene reciso alla radice (*pruned*), inibendo a priori l'esplorazione ridondante di ampie porzioni della rete e validando definitivamente il percorso traslato.

Al fine di istanziare formalmente tale funzione di limite inferiore H e supportare le successive dimostrazioni di ottimalità, si definiscono preliminarmente le seguenti strutture, a partire dalla modellazione del Grafo dei Tempi Minimi.

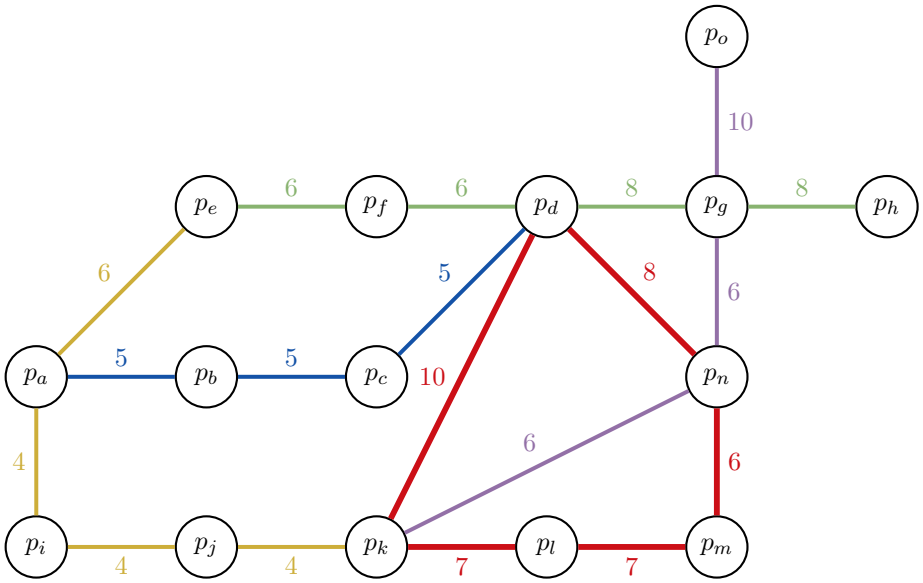

Figura 4.1: Esempio di grafo G_{min} per la validazione dell'algoritmo. I valori numerici sugli archi rappresentano il tempo minimo di percorrenza w_{ij} espresso in minuti. Si assume la simmetria dei pesi $w_{ij} = w_{ji}$ per modellare la bidirezionalità delle tratte.

Definizione 4.1 (**Grafo dei Tempi Minimi** $G_{min} = (V, E_{min})$)**.** Sia V l'insieme delle fermate e $E_{min} \subseteq V \times V$ l'insieme delle coppie di fermate connesse da almeno una tratta diretta. La costruzione di G_{min} avviene mediante una scansione dei dati GTFS:

1. Si identificano tutte le linee (*routes*) operanti nella rete;

2. Per ogni arco $(p_i, p_j) \in E_{min}$, si analizza l'insieme T_{ij} composto da tutte le corse (*trips*) che percorrono tale tratta;

3. Si assegna a ogni arco (p_i, p_j) un peso w_{ij} pari al *limite inferiore* temporale di percorrenza, calcolato come:

$$w_{ij} = \min_{t \in T_{ij}} (\Delta t_t)$$

dove Δt_t rappresenta la durata effettiva della corsa t tra le due fermate.

A partire da G_{min}, è necessario costruire una struttura dati che fornisca un limite temporale inferiore per qualsiasi coppia di nodi nella rete, indipendentemente dall'esistenza di un collegamento diretto.

Definizione 4.2 (**Grafo completo dei Cammini Minimi** H)**.** Sia $G_{min} = (V, E_{min})$ il grafo dei tempi minimi precedentemente definito. Si definisce $H = (V, E_H)$ come un grafo completo pesato, dove l'insieme degli archi E_H comprende tutte le possibili coppie di fermate $(p_u, p_v) \in V \times V$. A ogni arco $(p_u, p_v) \in E_H$ è associato un peso $H(u, v)$ pari alla distanza minima teorica tra i nodi, calcolata come:

$$H(u, v) = \min_{\pi \in \Pi_{uv}} \sum_{(p_i, p_j) \in \pi} w_{ij},$$

dove Π_{uv} è l'insieme di tutti i cammini che connettono p_u a p_v in G_{min}.

Data la struttura di grafo completo, H garantisce la proprietà di *disuguaglianza triangolare* ($\forall p_u, p_v, p_z \in V : H(u, z) \leq H(u, v) + H(v, z)$), rendendo ogni valore $H(u, v)$ un *limite inferiore* non rilassabile: per qualunque strategia di routing che operi su orari reali, il tempo di percorrenza effettivo $\tau_{eff}(p_u, p_v)$ sarà sempre tale che $\tau_{eff}(p_u, p_v) \geq H(u, v)$.

L'integrazione di H permette di trasformare le logiche di potatura da test puramente locali a criteri di dominanza globale.

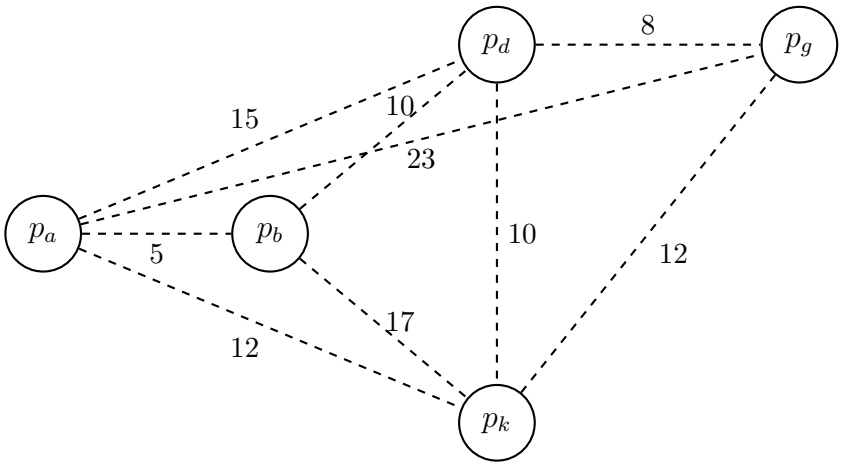

Figura 4.2: Grafo dei tempi minimi H derivato dalla rete esemplificativa presentata in Figura 4.1. Per migliorare la leggibilità, data la natura di grafo completo di H, sono rappresentati solo alcuni nodi e connessioni strategiche. La simmetria dei pesi ($h_{ij} = h_{ji}$) deriva direttamente dalla struttura simmetrica dei pesi w_{ij} in G_{min}.

4.1.2 Caso 1: Traslazione spaziale all'indietro

Il primo livello di ottimizzazione affronta il problema dell'estensione all'indietro di un percorso noto lungo la medesima linea di transito. L'obiettivo è sfruttare una corsa precalcolata, evitando il ricalcolo integrale dell'albero dei cammini minimi, a condizione che non esistano alternative locali dominanti.

Sia $P^*_{b,g}$ la frontiera delle soluzioni ottime originata da una fermata intermedia p_b verso una singola destinazione p_g, e sia $P \in P^*_{b,g}$ un generico viaggio ottimo pre-calcolato, formalmente espresso come la sequenza $P = (p_b, t_1, p_c, \ldots, p_g)$.

Data una nuova interrogazione da una fermata antecedente p_a situata lungo la stessa linea di transito, l'utente può salire a bordo della *medesima corsa* t_1 in anticipo. Si definisce pertanto la tratta diretta di avvicinamento come il viaggio elementare $P_{a,b} = (p_a, t_1, p_b)$.

L'orario richiesto a p_a per intercettare la coincidenza è estratto dalla tabella oraria della linea stessa:

$$\tau_{dep}(p_a) = \tau_{dep}(t_1, p_a).$$

Essendo la corsa di avvicinamento e la prima corsa del viaggio P coincidenti, il vincolo temporale al nodo di aggancio p_b è intrinsecamente rispettato ($\tau_{arr}(t_1, p_b) \leq \tau_{dep}(t_1, p_b)$). Poiché la corsa t_1 non si interrompe, il nodo p_b cessa di essere un punto di interscambio e diviene un punto di transito. Seguendo la Definizione 3.2, l'itinerario si riduce al *viaggio concatenato*:

$$P_{unione} = (p_a, t_1, p_c, \ldots, p_g).$$

Consideriamo l'insieme delle corse alternative.

Definizione 4.3.

Sia $V_a = \{p_s \in V : \Delta_{walk}(p_a, p_s) \leq \Delta_{walk}^{max}\}$ l'insieme delle fermate raggiungibili a piedi dall'origine p_a entro una soglia massima Δ_{walk}^{max}. Si definisce D l'insieme delle corse alternative temporalmente competitive in partenza dal vicinato V_a:

$$D = \Big\{ t' \in \mathcal{P}_s : p_s \in V_a \ \wedge \ \tau_{dep}(p_a) + \Delta_{walk}(p_a, p_s) \leq \tau_{dep}(t', p_s) \\ < \tau_{dep}(p_b) + \Delta t(p_b \to p_a) + \Delta_{walk}(p_a, p_s) \Big\}$$

dove $\mathcal{P}_s$ rappresenta l'insieme delle corse in transito presso la fermata p_s. (Nel caso in cui $p_s = p_a$, il tempo di percorrenza pedonale $\Delta_{walk}(p_a, p_a)$ è nullo).

Una volta definito l'insieme D, definiamo una condizione di ottimalità che servirà a potare lo spazio di ricerca.

Proposizione 4.1 (**Condizione di Ottimalità per traslazione all'indietro**)**.**

Il viaggio concatenato P_{unione} verso la destinazione p_g si definisce ottimo se e solo se $D = \emptyset$, oppure se per ogni corsa $t' \in D$ con primo nodo di svincolo p'_a è verificata la seguente disuguaglianza basata sul limite inferiore topologico H:

$$\tau_{arr}(t', p_{a'}) + H(a', g) \geq \tau_{arr}(P_{unione}, p_g), \tag{4.1}$$

dove $\tau_{arr}(t', p_{a'})$ indica l'orario effettivo di arrivo tabellare della corsa t' al nodo di svincolo p'_a.

Dimostrazione. La dimostrazione procede analizzando l'insieme D:

- **Caso 1 ($D = \emptyset$):** Non esistono corse alternative in partenza dal vicinato di p_a nell'intervallo di tempo utile. Essendo P ottimo da p_b a p_g per ipotesi, la concatenazione P_{unione} conserva intrinsecamente l'ottimalità.

- **Caso 2 ($D \neq \emptyset$):** Esiste almeno una corsa candidata $t' \in D$. Sia $P_{t'}$ un generico itinerario completo da p_a a p_g avente t' come prima tratta. Il tempo minimo teorico stimato per completare tale itinerario è definito come:
$$t_{est}(t') = \tau_{arr}(t', p_{a'}) + H(a', g).$$
Poiché $H(a', g)$ costituisce per definizione un *limite inferiore* assoluto basato sui tempi minimi di rete, il tempo di arrivo effettivo $\tau_{arr}(P_{t'}, p_g)$ soddisfa rigorosamente la disuguaglianza $\tau_{arr}(P_{t'}, p_g) \geq t_{est}(t')$. Assumendo verificata l'ipotesi della proposizione, si ha $t_{est}(t') \geq \tau_{arr}(P_{unione}, p_g)$. Applicando la proprietà transitiva si ottiene:
$$\tau_{arr}(P_{t'}, p_g) \geq \tau_{arr}(P_{unione}, p_g).$$
Ne consegue che ogni possibile itinerario $P_{t'}$ derivato da una qualsiasi corsa $t' \in D$ risulta temporalmente dominato (o al più equivalente) rispetto

a P_{unione}. L'assenza di percorsi strettamente dominanti in D certifica l'ottimalità di P_{unione}.

□

Il fulcro logico della potatura eseguita risiede nel termine $H(a', g)$. Essendo costruito sul grafo dei Cammini Minimi temporali, H agisce all'interno dell'algoritmo come un vero e proprio "orientatore geografico": esso penalizza le corse che si muovono nella direzione sbagliata rispetto alla destinazione, assegnando loro stime temporali (t_{est}) inevitabilmente elevate. Poiché nessun veicolo reale potrà mai coprire tale distanza in un tempo inferiore al minimo teorico, questo filtro permette di scartare a priori intere direttrici di rete senza la necessità di esplorarle integralmente.

Il costrutto *one-to-one* descritto costituisce la base logica dell'ottimizzazione. Tuttavia, in uno scenario di esplorazione one-to-all, l'algoritmo non possiede in memoria un singolo viaggio verso p_g, ma un intero albero dei cammini ottimi P_b^* radicato a p_b, che si dirama verso l'intero dominio V.

Per preservare l'esattezza matematica in tale scenario, la condizione di dominanza non viene più limitata a un solo p_g, ma deve essere validata simultaneamente su tutte le destinazioni.

Corollario 4.1 (**Estensione one-to-all**)**.**

Sia $P_{b,v} \in P_{b,v}^*$ e sia $P_{unione}^{(v)} = P_{a,b}P_{b,v}$ il viaggio concatenato verso una generica destinazione $p_v \in V$, derivato dalla traslazione dell'albero dei cammini minimi pre-calcolato. In uno scenario di esplorazione one-to-all, una corsa candidata $t' \in D$ con primo nodo di svincolo $p_{a'}$ risulta dominata, e conseguentemente potata a priori, se e solo se:

$$\tau_{arr}(t', p_{a'}) + H(a', v) \geq \tau_{arr}(P_{unione}^{(v)}, p_v), \quad \forall p_v \in V. \tag{4.2}$$

L'applicazione di questo corollario garantisce che la valutazione non rilassabile del *limite inferiore* elimini dallo spazio degli stati le direttrici globalmente inefficienti. Estendendo la condizione di ottimalità simultaneamente all'intero dominio V, l'algoritmo bypassa in modo sicuro l'esplorazione di rami che non possiedono le condizioni topologiche e temporali necessarie per migliorare gli orari di arrivo già garantiti dall'albero traslato $P_{unione}^{(v)}$ per alcuna destinazione all'interno della rete.

Esempio 4.1.

Per illustrare il funzionamento del criterio di dominanza basato sul *limite inferiore* H, consideriamo uno scenario operativo sulla rete in Figura 4.1.

Scenario:

Un utente intende raggiungere la fermata p_g partendo dalla fermata p_a. Supponiamo che il sistema abbia in memoria un viaggio ottimo precalcolato $P \in P_{b,g}^*$ (composto dalla tratta linea Blu $p_b \to p_d$ e linea Verde $p_d \to p_g$), che prevede la partenza dalla fermata p_b alle ore 09:05. Il tempo di percorrenza della tratta diretta $\Delta t(p_a \to p_b)$ (linea Blu) è pari a 5 minuti. Il sistema calcola

quindi l'orario di partenza richiesto dall'origine:

$$\tau_{dep}(p_a) = 09:05 - \Delta t(p_a \to p_b) = 09:00.$$

Il viaggio concatenato $P_{unione} = P_{a,b}P$ garantisce l'arrivo a destinazione p_g alle ore 09:26, assumendo $\Delta_{trans}(p_d) = 3$ min.

Analisi delle alternative:

Si ipotizzi la presenza di una corsa candidata t' (linea Gialla), in partenza da p_a alle ore 09:03 ($p_s = p_a, \Delta_{walk} = 0$). Poiché $\tau_{dep}(P_{unione}, p_a) < \tau_{dep}(t', p_a) < \tau_{dep}(P_{unione}, p_a) + \Delta t(p_b \to p_a)$, tale corsa appartiene all'insieme delle alternative da valutare. Dalle tabelle orarie si estrae l'arrivo di t' al nodo di svincolo p_k: $\tau_{arr}(t', p_k) = 09:15$.

Per determinare se t' possa essere competitiva, l'algoritmo applica il test di dominanza basato su H (4.1):

$$\tau_{arr}(t', p_k) + H(k, g) \geq \tau_{arr}(P_{unione}, p_g).$$

Sostituendo i valori estratti dalle strutture dati:

- $\tau_{arr}(t', p_k) = 09:15$
- $H(k, g) = 12$ min (distanza minima teorica)

Il costo totale stimato è dato da $09:15 + 12$ min $= 09:27$.

Conclusione:

Il test di dominanza esegue il confronto:

$$09:27 \geq 09:26 \implies \textbf{Vero}.$$

La condizione risulta vera: il *limite inferiore* calcolato per la corsa t' (09:27) è superiore all'orario di arrivo già garantito da P_{unione} (09:26). Di conseguenza, la corsa t' viene dichiarata dominata. L'algoritmo esclude t' dall'esplorazione, evitando l'esecuzione di una ricerca superflua sulla linea Gialla e preservando le risorse computazionali.

4.1.3 Caso 2: Traslazione temporale in avanti e assorbimento del ritardo

Il secondo caso di ottimizzazione gestisce le interrogazioni traslate temporalmente in avanti, sfruttando i tempi di attesa insiti nei trasferimenti intermodali per evitare il ricalcolo di porzioni di rete già esplorate.

Definizione 4.4 (*Slack Time* e corsa alternativa P').

Sia $P \in P^*_{b,g}$ un viaggio ottimo in memoria originato dalla fermata p_b verso la destinazione p_g, calcolato per un orario di interrogazione t_b. Si assuma che P preveda almeno un nodo di interscambio p_d. Si definisce *slack time* l'attesa in banchina presso il nodo p_d, calcolata come:

$$slack_d = \tau_{dep}(P, p_d) - \tau_{arr}(P, p_d). \tag{4.3}$$

Data una nuova interrogazione per la medesima coppia (p_b, p_g) posticipata a un orario $t'_b = t_b + \Delta t$, si definisce P' il primo viaggio utile che copre la tratta iniziale da p_b a p_d, tale per cui $\tau_{dep}(P', p_b) \geq t'_b$.

Proposizione 4.2 (**Condizione di ammissibilità del trasbordo**)**.**
Il viaggio concatenato $P_{unione} = P'P_{d,g}$ si definisce ammissibile e conserva l'ottimalità del segmento terminale senza richiedere l'esecuzione dell'algoritmo di ricerca se e solo se la corsa alternativa P' soddisfa la seguente disuguaglianza:

$$\tau_{arr}(P', p_d) + \Delta_{trans}(p_d) \leq \tau_{dep}(P, p_d)$$

dove $\Delta_{trans}(p_d)$ rappresenta il tempo tecnico minimo di trasferimento presso il nodo p_d.

Dimostrazione. Sia $P_{d,g}$ la porzione terminale del viaggio ottimo originale P. L'esecuzione garantita di tale segmento richiede la presenza fisica dell'utente al nodo p_d al tempo $\tau_{dep}(P, p_d)$ con idoneità all'imbarco.

L'utente, partendo in ritardo e transitando tramite la corsa alternativa P', giunge fisicamente a p_d al tempo $\tau_{arr}(P', p_d)$. Espletate le operazioni di trasferimento, risulta pronto all'imbarco al tempo $\tau_{arr}(P', p_d)+\Delta_{trans}(p_d)$. Assumendo verificata l'ipotesi della proposizione, si ha:

$$\tau_{arr}(P', p_d) + \Delta_{trans}(p_d) \leq \tau_{dep}(P, p_d).$$

Questo implica che l'utente intercetta in tempo utile la coincidenza originaria. Poiché il segmento $P_{d,g}$ era già stato certificato come ottimo per l'orario di partenza $\tau_{dep}(P, p_d)$, ed essendo quest'ultimo inalterato, l'ottimalità della tratta terminale è preservata. Di conseguenza, il viaggio composto P_{unione} rappresenta una soluzione valida e ottima per la partenza posticipata t'_b. □

Dal punto di vista logico, l'ammissibilità certifica che il ritardo Δt accumulato partendo al tempo t'_b è stato interamente "assorbito" dallo *slack time* in banchina alla fermata p_d.

Corollario 4.2 (**Filtro selettivo per one-to-all**)**.**
In uno scenario di esplorazione su albero globale (one-to-all), la condizione di ammissibilità viene iterata su ogni singolo nodo di interscambio p_{d_i} diramato dalla radice. Il meccanismo agisce come un filtro selettivo sull'albero pre-calcolato:

- *Per i rami in cui $\tau_{arr}(P', p_{d_i}) + \Delta_{trans}(p_{d_i}) \leq \tau_{dep}(P, p_{d_i})$, la coincidenza è garantita: l'algoritmo salva le risorse computazionali riutilizzando i percorsi originari in memoria.*

- *Per i rami in cui la disuguaglianza è violata, la coincidenza è persa: l'algoritmo avvia un ricalcolo mirato esclusivamente per il sotto-albero fallito, preservando l'esattezza complessiva senza sprecare calcoli sulle direttrici intatte.*

Esempio 4.2.

Scenario:

Il sistema ha in memoria un viaggio ottimo $P \in P^*_{b,g}$ con partenza da p_b alle 10:05, la cui tratta di prosecuzione $P_{d,g}$ prevede la partenza dal nodo di svincolo p_d alle ore 10:25. Si consideri un utente che, a causa di un ritardo all'origine p_b, non può intercettare la coincidenza ottimale. L'utente opta per una corsa alternativa P', in partenza da p_b alle ore 10:07, che giunge al nodo di svincolo p_d alle ore 10:17.

Analisi di ammissibilità:

Il sistema deve verificare se P' permette di effettuare il trasbordo verso $P_{d,g}$. Il vincolo di ammissibilità (4.2) richiede che l'orario di arrivo della corsa P' presso il nodo di svincolo p_d, sommato al tempo minimo di interscambio $\Delta_{trans}(p_d)$, sia inferiore o uguale all'orario di partenza della tratta successiva:

$$\tau_{arr}(P', p_d) + \Delta_{trans}(p_d) \leq \tau_{dep}(P_{d,g}, p_d).$$

Sostituendo i valori estratti dalle strutture dati:

- $\tau_{arr}(P', p_d) = 10:17$
- $\Delta_{trans}(p_d) = 3$ min (tempo tecnico di interscambio)
- $\tau_{dep}(P_{d,g}, p_d) = 10:25$

Il tempo di arrivo effettivo al cambio risulta pari a $10:17 + 3 \text{ min} = 10:20$. In questo scenario, poiché $10:20 \leq 10:25$, la coincidenza è tecnicamente fattibile.

Conclusione:

Il ritardo di 2 minuti in partenza è stato totalmente assorbito. Il viaggio unificato $P_{unione} = P'_{b,d} \cup P_{d,g}$ viene dichiarato ammissibile e restituito come percorso ottimo senza riattivare l'algoritmo di ricerca.

4.1.4 Caso 3: Ottimalità dei sottopercorsi

Il terzo livello di ottimizzazione formalizza il riuso logico di un viaggio esplorato, muovendosi in avanti lungo la catena topologica e sfruttando le proprietà strutturali dei cammini minimi.

Sia $P \in P^*_{a,g}$ un viaggio ottimo pre-calcolato originato dalla fermata p_a verso la destinazione p_g. Assumendo che P attraversi un nodo intermedio p_b, il viaggio è scomponibile: $P = P_{a,b}P_{b,g}$.

Data una nuova interrogazione per la tratta (p_b, p_g) con un orario di richiesta t_b, la validità del riuso del segmento terminale $P_{b,g}$ dipende dalla relazione cronologica tra t_b e l'orario di partenza originario della corsa da p_b, definito come $\tau_{dep}(P_{b,g}, p_b)$.

1. Sincronicità esatta

Se l'orario di interrogazione coincide perfettamente con l'orario di transito del viaggio originale, ovvero $t_b = \tau_{dep}(P_{b,g}, p_b)$, l'ottimalità del sottopercorso è garantita dal seguente teorema.

Teorema 4.1 (**Principio di ottimalità di Bellman**).

Dato un percorso ottimo P che congiunge un nodo origine p_a a un nodo destinazione p_g, per qualsiasi nodo intermedio p_b attraversato dal percorso, il segmento $P_{b,g} \subseteq P$ rappresenta inequivocabilmente la soluzione ottima tra p_b e p_g per l'orario di partenza $\tau_{dep}(P_{b,g}, p_b)$.

Dimostrazione. La dimostrazione procede per assurdo. Si supponga che il segmento terminale $P_{b,g}$ non sia il percorso ottimo per congiungere p_b a p_g. Deve dunque esistere un percorso alternativo $P'_{b,g}$, in partenza da p_b al medesimo orario, che risulti strettamente dominante, offrendo ovvero un orario di arrivo antecedente: $\tau_{arr}(P'_{b,g}, p_g) < \tau_{arr}(P_{b,g}, p_g)$.

Se tale percorso esistesse, componendo la tratta iniziale inalterata $P_{a,b}$ con l'alternativa $P'_{b,g}$, si otterrebbe un nuovo viaggio completo $P'_{concat} = P_{a,b} \cup P'_{b,g}$. Tale viaggio risulterebbe strettamente migliore del viaggio originale P. Tuttavia, ciò contraddice palesemente l'ipotesi fondante secondo cui P rappresenta la soluzione ottima globale calcolata dall'algoritmo tra p_a e p_g. L'assurdo dimostra che non può esistere alcun $P'_{b,g}$ dominante; pertanto, il sottopercorso $P_{b,g}$ conserva intrinsecamente la propria ottimalità. □

2. Asincronicità: arrivo in anticipo

Qualora l'orario di query t_b differisca dal transito originale, l'utente giunge al nodo p_b in asincronia.

Il caso del **Ritardo** ($t_b > \tau_{dep}(P_{b,g}, p_b)$) implica la perdita del segmento originale e ricade intrinsecamente nelle logiche di ammissibilità e assorbimento trattate nel Caso 2.

Il caso dell'**Anticipo** ($t_b < \tau_{dep}(P_{b,g}, p_b)$) richiede invece una valutazione specifica per escludere alternative divenute temporaneamente accessibili.

Definiamo l'insieme delle corse alternative che dobbiamo considerare.

Definizione 4.5 (**Finestra di Anticipo e Corse Alternative** D).

Sia $V_b = \{p_s \in V : \Delta_{walk}(p_b, p_s) \leq \Delta_{walk}^{max}\}$ l'insieme delle fermate raggiungibili pedonalmente da p_b. Si definisce $W = [t_b, \tau_{dep}(P_{b,g}, p_b))$ la finestra temporale di anticipo. L'insieme D delle corse alternative temporalmente competitive è definito come:

$$D = \{t' \in \mathcal{P}_s : p_s \in V_b \ \wedge \ t_b \leq \tau_{dep}(t', p_s) - \Delta_{walk}(p_b, p_s) < \tau_{dep}(P_{b,g}, p_b)\}.$$

Ora è necessario definire una condizione di ottimalità per scartare i rami non interessanti.

Proposizione 4.3 (**Condizione di Ottimalità in Anticipo**).

Il sottopercorso originario $P_{b,g}$ conserva la sua ottimalità (e l'utente dovrà attendere l'orario di partenza originario) se e solo se $D = \emptyset$, oppure se per ogni corsa candidata $t' \in D$ con primo nodo di svincolo $p_{b'}$ è verificata la disuguaglianza:

$$\tau_{arr}(t', p_{b'}) + H(b', g) \geq \tau_{arr}(P_{b,g}, p_g).$$

Dimostrazione. Sia $P_{t'}$ un generico itinerario completo da p_b a p_g avente t' come prima tratta. Il tempo minimo teorico stimato per completare tale itinerario è definito come $t_{est}(t') = \tau_{arr}(t', p_{b'}) + H(b', g)$. Poiché $H(b', g)$ costituisce un limite inferiore non rilassabile, il tempo di arrivo effettivo soddisfa rigorosamente $\tau_{arr}(P_{t'}, p_g) \geq t_{est}(t')$. Assumendo verificata l'ipotesi della proposizione, si ottiene $t_{est}(t') \geq \tau_{arr}(P_{b,g}, p_g)$. Per transitività risulta $\tau_{arr}(P_{t'}, p_g) \geq \tau_{arr}(P_{b,g}, p_g)$. Ne consegue che le corse anticipate risultano temporalmente dominate. L'assenza di candidati non dominati nell'insieme D certifica che non vi è alcun vantaggio a sfruttare l'anticipo, confermando l'ottimalità del riuso passivo di $P_{b,g}$. □

Corollario 4.3 (**Estrazione one-to-all**)**.**

Dato l'albero globale dei cammini ottimi P^*_a radicato in p_a, la condizione di sincronicità esatta al nodo p_b ne permette un'estrazione diretta: troncando i segmenti antecedenti, si ottiene istantaneamente l'intero set dei cammini ottimi $P^*_{b,v}$ verso ogni destinazione raggiungibile.

In presenza di anticipo, una generica corsa candidata $t' \in D$ con primo nodo di svincolo $p_{b'}$ viene dominata e preventivamente potata se e solo se:

$$\tau_{arr}(t', p_{b'}) + H(b', v) \geq \tau_{arr}(P^*_{b,v}, p_v), \quad \forall p_v \in V,$$

dove $\tau_{arr}(P^*_{b,v}, p_v)$ rappresenta l'orario di arrivo al nodo p_v garantito dal sottoalbero originario.

L'efficacia di questa formulazione si massimizza negli scenari *one-to-all*. Invece di invalidare e ricalcolare ciecamente l'intera rete a fronte di un anticipo alla sorgente, l'algoritmo filtra le nuove corse disponibili tramite H. Il ricalcolo viene confinato e istanziato in modo mirato solo per quelle direttrici che possiedono un potenziale reale di miglioramento, azzerando il costo computazionale per le restanti porzioni della mappa.

Esempio 4.3 (**Valutazione dell'anticipo tramite limite inferiore**)**.**

Scenario:

Si consideri il viaggio ottimo $P \in P^*_{a,g}$ che parte da p_a alle ore 11:00 (arrivo stimato a p_g alle 11:26). Il sistema ha memorizzato il transito presso il nodo intermedio p_b alle ore 11:05. L'utente giunge a p_b alle ore 11:01, con un anticipo di 4 minuti rispetto al piano originale.

Analisi delle alternative:

L'anticipo permette di esplorare la finestra $W = [11:01, 11:05)$. All'interno di essa si valuta la corsa t' (linea Blu) in partenza da p_b alle ore 11:02. Questa corsa t', precedentemente scartata poiché non consentiva di intercettare l'itinerario delle 11:05, diviene ora un candidato valido per il confronto. Dati di input estratti dalle tabelle orarie:

- $\tau_{arr}(t', p_d) = 11:12$ (arrivo al nodo di svincolo)
- $H(d, g) = 8$ min (distanza minima teorica)

Test di ottimalità:
Il sistema verifica se t' possa migliorare l'arrivo a p_g valutando la condizione di dominanza (4.3):

$$\tau_{arr}(t', p_d) + H(d, g) \geq \tau_{arr}(P^*_{b,g}, p_g).$$

Sostituendo i valori temporali, si ottiene il tempo stimato $t_{est}(t')$:

$$11:12 + 8 \text{ min} = 11:20.$$

Il test esegue il confronto logico:

$$11:20 \geq 11:26 \implies \textbf{Falso}.$$

Conclusione:
La disuguaglianza non è verificata. Il *limite inferiore* teorico della corsa t' (11:20) è strettamente inferiore all'orario di arrivo garantito dal sottopercorso originario (11:26). Poiché t' abilita un itinerario teoricamente più rapido rispetto a quello memorizzato, l'algoritmo **invalida il riuso passivo** del viaggio $P_{b,g}$ e invoca dinamicamente il ricalcolo per esplorare in modo esatto la nuova soluzione offerta da t'.

4.1.5 Caso 4: Traslazione all'indietro con attesa

Il quarto e ultimo scenario di ottimizzazione affronta un disallineamento temporale alla sorgente, combinando la necessità di traslazione spaziale all'indietro con la gestione di un ritardo fisiologico in partenza.

Si supponga di aver calcolato un viaggio ottimo $P \in P^*_{b,g}$. A fronte di una nuova interrogazione da una fermata antecedente p_a (sulla medesima linea) con orario richiesto $t_a = \tau_{dep}(P, p_b)$, l'utente è costretto ad attendere in banchina una corsa successiva P' tale che $\tau_{dep}(P', p_a) \geq t_a$. Assumendo che P preveda un nodo di interscambio p_d, la corsa P' si definisce *ammissibile* in logica *one-to-one* se il ritardo accumulato viene interamente assorbito dallo *slack time* in banchina a p_d, ovvero se:

$$\tau_{arr}(P', p_d) + \Delta_{trans}(p_d) \leq \tau_{dep}(P, p_d).$$

Qualora la condizione sia verificata, si genera il viaggio concatenato $P_{unione} = P'_{a,d} P_{d,g}$.

Definizione 4.6 (**Insieme delle alternative in attesa** D)**.**
Sia V_a l'insieme delle fermate raggiungibili pedonalmente da p_a. Per escludere la presenza di alternative più rapide durante l'attesa alla sorgente, si definisce l'insieme D delle corse in partenza nella finestra temporale antecedente all'imbarco su P':

$$D = \{t' \in \mathcal{P}_s : p_s \in V_a \ \wedge \ t_a \leq \tau_{dep}(t', p_s) - \Delta_{walk}(p_a, p_s) < \tau_{dep}(P', p_a)\}.$$

Proposizione 4.4 (**Condizione di ottimalità globale**).

Il viaggio concatenato ammissibile P_{unione} conserva la sua ottimalità se e solo se $D = \emptyset$, oppure se per ogni corsa candidata $t' \in D$ (avente $p_{a'}$ come primo nodo di svincolo) è verificata la disuguaglianza basata sul limite inferiore topologico H:

$$\tau_{arr}(t', p_{a'}) + H(a', g) \geq \tau_{arr}(P_{unione}, p_g).$$

Dimostrazione. Sia $P_{t'}$ un generico itinerario completo da p_a a p_g originato dalla corsa in attesa t'. Il tempo minimo teorico per completare tale itinerario è definito come:

$$t_{est}(t') = \tau_{arr}(t', p_{a'}) + H(a', g).$$

Poiché $H(a', g)$ costituisce un limite inferiore non rilassabile, il tempo di arrivo effettivo esplorabile soddisfa rigorosamente $\tau_{arr}(P_{t'}, p_g) \geq t_{est}(t')$. Assumendo verificata l'ipotesi della proposizione, si ha $t_{est}(t') \geq \tau_{arr}(P_{unione}, p_g)$. Applicando la proprietà transitiva si ottiene:

$$\tau_{arr}(P_{t'}, p_g) \geq \tau_{arr}(P_{unione}, p_g).$$

Ne consegue che ogni possibile itinerario alternativo nato durante l'attesa risulta temporalmente dominato (o equivalente). L'assenza di candidati strettamente dominanti in D certifica l'ottimalità assoluta di P_{unione}. □

Corollario 4.4 (**Filtro duale per one-to-all**).

In uno scenario di esplorazione su albero globale (one-to-all), la validazione algoritmica si biforca in un sistema a filtro duale:

1. ***Filtro globale alla sorgente (ottimalità):*** *Una generica corsa in attesa $t' \in D$ viene dominata e preventivamente potata se e solo se:*

$$\tau_{arr}(t', p_{a'}) + H(a', v) \geq \tau_{arr}(P^{(v)}_{unione}, p_v), \quad \forall p_v \in V.$$

2. ***Ammissibilità:*** *La condizione di trasbordo $\tau_{arr}(P', p_{d_i}) + \Delta_{trans}(p_{d_i}) \leq \tau_{dep}(P, p_{d_i})$ viene applicata indipendentemente a ciascun nodo di interscambio p_{d_i} dell'albero esplorativo.*

Esempio 4.4 (**Integrazione tra ammissibilità e ottimalità**).

Scenario: Si consideri un utente in partenza dalla fermata p_b alle ore 12:05, con arrivo al nodo di svincolo p_d alle ore 12:15. Il viaggio in memoria P prosegue da p_d verso p_g con partenza alle 12:22 (arrivo a p_g alle 12:30). Un utente invia una query da p_a alle ore 12:05. Il sistema individua la corsa di avvicinamento P' (linea Blu) in partenza da p_a alle 12:07, che giunge al nodo p_d alle 12:17.

Verifica di ammissibilità:

Il sistema controlla se il cambio a p_d è fattibile ($\Delta_{trans}(p_d) = 3$ min):

$$\tau_{arr}(P', p_d) + \Delta_{trans}(p_d) \leq \tau_{dep}(P_{d,g}, p_d) \implies 12:20 \leq 12:22.$$

La condizione è soddisfatta: $P_{unione} = P'_{a,d} \cup P_{d,g}$ è un candidato ammissibile con arrivo a p_g alle 12:30.

Analisi delle alternative (Ottimalità):

Per escludere che l'attesa alla fermata p_a nasconda alternative più rapide, si ispeziona l'insieme D. Si supponga esista una corsa t' (linea Gialla) in partenza da p_a alle ore 12:05 ($12:05 < 12:07$), diretta al nodo di svincolo p_k. Dati di input per t':

- $\tau_{arr}(t', p_k) = 12:17$ (arrivo al nodo di svincolo),
- $H(k, g) = 12$ min.

Conclusione:

L'algoritmo valuta la disuguaglianza di dominanza per verificare se t' può essere scartata (4.4):

$$\tau_{arr}(t', p_k) + H(k, g) \geq \tau_{arr}(P_{unione}, p_g),$$

$$12:17 + 12 \text{ min} \geq 12:30 \implies 12:29 \geq 12:30 \implies \textbf{Falso}.$$

Poiché la disuguaglianza risulta falsa, la potatura fallisce: la corsa t' (con un potenziale teorico di arrivo alle 12:29) è competitiva rispetto alla soluzione assemblata P_{unione} (12:30). Di conseguenza, l'algoritmo **invalida** la certezza di P_{unione} e procede con l'esplorazione del ramo associato alla corsa t'.

4.1.6 Implementazione computazionale e adattamenti euristici

I teoremi e i corollari formalizzati nei paragrafi precedenti delineano un quadro teorico per l'ottimizzazione dei viaggi. Tuttavia, la trasposizione algoritmica di tali regole su un paradigma *one-to-all* all'interno di reti di scala metropolitana (caratterizzate da migliaia di nodi e altissima densità di interscambi) ha fatto emergere alcune criticità strutturali.

L'applicazione letterale dei corollari teorici, pur corretta sulla singola direttrice, ha evidenziato empiricamente la comparsa di differenze nelle soluzioni rispetto alla frontiera esatta del RAPTOR Standard. Per superare questi colli di bottiglia e garantire l'efficienza algoritmica senza compromettere l'esattezza assoluta delle soluzioni, il metodo implementato adotta due specifiche scelte ingegneristiche.

L'ancoraggio alla radice per preservare la memoria

Nei contesti di ritardo (Caso 2 e Caso 4), la formulazione generalizzata del modello a interrogazioni *one-to-all* consentirebbe di rincorrere le coincidenze testando l'assorbimento del ritardo iterativamente su molteplici nodi di interscambio futuri p_{d_i}. A livello computazionale, tuttavia, questa pratica impone una "manutenzione selettiva complessa" della memoria: l'algoritmo dovrebbe invalidare (e ricalcolare) ampie porzioni dello spazio degli stati, preservando solo i sotto-alberi in cui la coincidenza è stata salvata.

Lo svuotamento parziale della memoria inibisce l'efficacia del *limite inferiore* topologico H: privato dei tempi ottimi di riferimento storici per la maggior parte delle destinazioni, il criterio di dominanza globale non è più in grado di

potare i rami sub-ottimali in fase di esplorazione. Di conseguenza, il sistema è costretto a ri-esplorare ciecamente la rete, vanificando i benefici prestazionali del *warm-start*. Peggio ancora, conservare frammenti dell'albero storico rischia di "inquinare" la memoria con soluzioni ibride, mascherando l'esistenza di direttrici totalmente nuove e globalmente più efficienti emerse a causa del ritardo iniziale.

Per prevenire ciò e azzerare il numero di soluzioni diverse, e quindi errate, l'implementazione vincola l'ammissibilità spaziale adottando un'**euristica conservativa di ancoraggio alla radice**. L'assorbimento del ritardo viene testato esclusivamente al nodo origine p_b della memoria pre-calcolata:

$$\tau_{arr}(P', p_b) + \Delta_{trans}(p_b) \leq \tau_{dep}(P^*_b, p_b).$$

Questo adattamento traduce il problema della manutenzione selettiva in un controllo binario di validità:

1. Se il ritardo è assorbibile localmente a p_b, l'intero albero *one-to-all* pre-calcolato rimane invariato al 100%, massimizzando il risparmio computazionale.

2. Se il ritardo eccede lo *slack time* alla radice, l'intera memoria viene invalidata e il ricalcolo viene istanziato *ex-novo* da uno stato pulito.

Tale scelta ingegneristica tutela l'esattezza globale, garantendo esplorazioni sicure a discapito di salvataggi parziali illusori.

Gestione dinamica della memoria: il filtro ibrido (Casi 1, 3 e 4)

La seconda criticità emerge durante le fasi di ricalcolo esplorativo. I corollari teorici prevedono staticamente che una nuova corsa alternativa t' sia valutata confrontandone la stima teorica unicamente contro l'orario garantito dal sotto-albero storico originario ($\tau_{arr}(P^*_{b,v}, p_v)$).

Affidarsi esclusivamente al dato storico genera però un disallineamento durante l'esecuzione del codice. L'algoritmo, avanzando ciclicamente nell'esplorazione, scopre nuove soluzioni locali ottimali non presenti nella memoria originaria. Ignorare queste nuove scoperte porta il sistema a mantenere in vita rami di rete che potrebbero essere scartati, riducendo l'efficienza globale.

Per questo motivo, la validazione algoritmica è stata implementata attraverso un processo a due fasi.

Nella **Fase 1 (Pre-filtraggio Statico)**, l'insieme delle corse candidate D viene costruito applicando rigorosamente la teoria: i veicoli vengono filtrati misurandoli contro l'albero di memoria inattivo.

Quando tuttavia è necessario innescare la ricerca sulla rete, si attiva la **Fase 2 (Esplorazione Dinamica)**. All'interno del ciclo di ricalcolo, il test di dominanza diviene *ibrido*. Il limite inferiore H viene istruito per confrontare le stime di una corsa t' contro uno stato locale, calcolando costantemente il minimo assoluto tra il tempo ereditato dalla memoria e il nuovo miglior tempo

locale (τ^*_{local}) individuato dinamicamente nel rispettivo ciclo di ricerca:

$$\tau_{arr}(t', p_{a'}) + H(a', v) \geq \min(\tau^*_{local}(p_v), \tau_{arr}(P^*_{b,v}, p_v)), \quad \forall p_v \in V.$$

L'adozione dinamica del minimo garantisce che la frontiera di dominanza diventi progressivamente più stringente durante l'esecuzione. Ogni potatura viene eseguita se e solo se la corsa è matematicamente incapace di competere sia con le garanzie storiche, sia con i progressi in tempo reale, garantendo l'assoluta ottimalità dei risultati per ogni fermata della mappa metropolitana.

Unificazione architetturale dei casi operativi

Mentre la formalizzazione matematica ha richiesto di scomporre e analizzare singolarmente i quattro scenari teorici per certificarne la validità analitica, la loro trasposizione algoritmica si fonda su un'architettura integrata. Le casistiche non sono gestite da percorsi separati, bensì convergono in un unico *framework* valutativo che rispetta rigorosamente i vincoli teorici, ottimizzandone l'esecuzione computazionale.

L'architettura di questa unificazione si traduce operativamente in una fase preliminare di selezione dei candidati. Quando una nuova interrogazione prevede una traslazione spaziale (ovvero quando l'origine dell'utente non coincide con la radice dell'albero dei cammini minimi pre-calcolato in memoria), il codice non innesca moduli separati per gestire partenze perfette o ritardi, ma costruisce l'insieme delle corse alternative D incanalando l'intera offerta di transito attraverso una cascata di quattro filtri logici sequenziali:

- **Vicinato pedonale (Filtro spaziale):** Il sistema mappa l'insieme delle fermate s raggiungibili a piedi dal nodo origine a entro un orizzonte temporale massimo prefissato.
- **Ammissibilità in partenza (Filtro cronologico):** Per ogni fermata del vicinato, l'algoritmo isola esclusivamente le corse il cui orario di partenza, decurtato del tempo di camminata Δ_{walk}, risulti maggiore o uguale all'orario richiesto dall'utente t_a.
- **Aggancio alla memoria (Assorbimento del ritardo):** Implementa l'euristica di ancoraggio alla radice (Casi 2 e 4) tramite un controllo binario di validità, decidendo se preservare integralmente o invalidare la cache pre-calcolata.
- **Dominanza topologica (Matrice H):** Le corse candidate sopravvissute vengono infine sottoposte alla condizione di ottimalità globale. Il sistema interroga il limite inferiore H verso tutte le destinazioni: se la stima della nuova corsa risulta peggiore della soluzione originaria pre-calcolata, il ramo viene considerato inefficiente e definitivamente potato.

Parallelamente, quando l'interrogazione prevede una traslazione temporale rispetto all'albero dei cammini minimi che abbiamo in memoria, si verificano due scenari:

- **Ritardo:** L'utente giunge in banchina a un orario successivo, rendendo inevitabile la perdita della coincidenza originaria. Seguendo la formulazione dell'ancoraggio alla radice, azzera preventivamente la memoria e avvia un ricalcolo pulito basato sulla matrice H.
- **Anticipo:** L'utente arriva prima del previsto. In questo scenario il sistema attiva una ricerca mirata all'interno della finestra di anticipo, filtrando le eventuali corse alternative tramite la condizione di dominanza topologica.

Questa architettura a filtri sequenziali permette all'algoritmo di riconoscere organicamente le condizioni al contorno e di auto-instradarsi verso la potatura ottimale, traducendo decine di vincoli teorici in un blocco di calcolo compatto.

Tabella della notazione del Capitolo 4

La seguente tabella riassume i simboli e le notazioni matematiche introdotte in questo capitolo per la modellazione delle ottimizzazioni temporali e spaziali.

Simbolo	**Definizione / Significato**
$G = (V, A)$	Grafo di instradamento del sistema di trasporto (fermate V e archi A).
$G_{min} = (V, E_{min})$	Grafo dei tempi minimi assoluti costruito dai dati GTFS.
$H = (V, E_H)$	Grafo completo dei cammini minimi (limite inferiore globale non rilassabile).
Π_{uv}	Insieme di tutti i cammini che connettono p_u a p_v in G_{min}.
$\tau_{eff}(p_u, p_v)$	Tempo di percorrenza effettivo tra p_u e p_v.
D	Insieme delle corse alternative temporalmente competitive.
$\Delta_{walk}(p_a, p_s)$	Tempo di percorrenza pedonale tra l'origine p_a e la fermata p_s.
Δ_{walk}^{max}	Soglia temporale massima di accettabilità per un percorso pedonale.
V_a	Insieme delle fermate raggiungibili a piedi dall'origine p_a entro la soglia Δ_{walk}^{max}.
$t_{est}(t')$	Tempo minimo teorico stimato per completare un itinerario basato sulla corsa t'.
$\tau_{arr}(P, p_g)$	Orario di arrivo a destinazione p_g garantito dal percorso P.
$\tau_{dep}(t, p_s)$	Orario di partenza della corsa t presso la fermata p_s.
$slack_d$	Tempo di attesa in banchina (*slack time*) presso il nodo di interscambio p_d.
$\Delta_{trans}(p_d)$	Tempo tecnico minimo di trasferimento presso il nodo p_d.
$\tau^*(p_v)$	Orario di arrivo ottimo garantito per la fermata p_v nello scenario *one-to-all*.

Tabella 4.1: Notazione del Capitolo 4.

Capitolo 5

Modellazione delle Reti Tariffarie Condizionali ed estensioni avanzate

Il lavoro presentato in questo capitolo trae ispirazione diretta e si fonda sul framework teorico proposto da Euler, Lindner e Borndörfer nell'articolo *"Price optimal routing in public transportation"* [4]. Gli autori introducono il concetto di **Conditional Fare Networks**, offrendo per la prima volta un formalismo matematico basato sulla teoria dei monoidi capace di catturare la dinamicità dei titoli di viaggio moderni.

Partendo da queste solide basi metodologiche, è stata sviluppata un'implementazione adattata al contesto dello **STIBM**.

Le Reti Tariffarie Condizionali, sono un framework flessibile che modella la struttura tariffaria come un grafo dei biglietti [4]. I parametri tradizionali come il numero di fermate, la distanza percorsa, le zone tariffarie, i cambi e l'attraversamento dei confini cittadini possono essere parzialmente ordinati, sommati lungo un percorso e possiedono un elemento neutro. Pertanto, possono essere modellati come un monoide positivo parzialmente ordinato. Il monoide complessivo integra tutti questi aspetti, rappresentato come $(\mathcal{H}_{dist} \times \mathcal{H}_{stop} \times \mathcal{H}_{tran} \times \mathcal{H}_{zone} \times \prod_{c \in \mathcal{C}} \mathcal{H}_c, +, \leq)$. Tuttavia, il modello puramente monoidale presenta delle limitazioni: richiede che il prezzo e il percorso siano ordinati in modo identico, non può gestire i sovrapprezzi temporali, non riesce a esprimere condizioni logiche complesse e genera alberi di ricerca eccessivamente grandi.

Per superare queste limitazioni, viene utilizzato il concetto di grafo dei biglietti. Nei sistemi reali, i biglietti evolvono lungo il percorso. Un grafo dei biglietti rappresenta ogni tipo di biglietto come un nodo, con archi che rappresentano le possibili transizioni. Le transizioni seguono condizioni precise basate sul peso accumulato e su specifici eventi tariffari, come l'uscita da una zona urbana. Una funzione di transizione Γ prende il biglietto corrente, il peso accumulato e l'evento tariffario per restituire il nuovo biglietto valido. Una CFN completa è definita dal grafo dei biglietti, dalla funzione di transizione, dai pesi, dagli eventi, dagli stati iniziali e da una funzione di prezzo monotona π.

Definizioni di Base

Si riportano di seguito le definizioni fondanti del modello, così come formalizzate originariamente da Euler et al. [4]. Sia $G = (V, A)$ il grafo di instradamento del sistema di trasporto. Sia $(H, +, \leq)$ un monoide positivo parzialmente ordinato. Sia $\mathcal{T} = (\Theta, E)$ il grafo dei biglietti. Sia S l'insieme degli eventi tariffari. Per ogni arco $a \in A$:

- $\omega(a) \in H$ è il peso dell'arco.
- $e(a) \in S$ è l'evento tariffario.

L'aggregazione di tali pesi ed eventi lungo un cammino in G determina il suo stato tariffario [4].

Definizione 5.1 (Stato Tariffario [4])**.** Uno **stato tariffario** $f \in \Phi$ è una coppia composta da un biglietto $\theta(f) \in \Theta$ e un peso accumulato $w(f) \in H$. Lo spazio totale degli stati tariffari è definito come:

$$\Phi := \Theta \times H$$

Ogni vertice $v \in V$ è etichettato con uno stato tariffario iniziale $\mu(v) \in \Phi$. Gli stati tariffari fungono da etichette di percorso per gli algoritmi di calcolo dei cammini minimi, poiché contengono tutte le informazioni necessarie per decidere la dominanza tra i percorsi stessi. A differenza delle comuni applicazioni di cammino minimo (multi-obiettivo), le etichette degli archi definite in $H \times S$ non appartengono allo stesso spazio delle etichette dei percorsi Φ.

Per consentire il tracciamento degli stati tariffari lungo i percorsi in G, il framework teorico formalizza il concetto di funzione di transizione applicata ai biglietti $\theta \in \Theta$. Tale funzione restituisce il nuovo biglietto $\theta_2 \in \Theta$ in cui un biglietto precedente $\theta_1 \in \Theta$ transita, a fronte di un peso accumulato $h \in H$ e di un evento tariffario $s \in S$. I possibili candidati per la transizione includono l'intorno uscente di θ_1 nel grafo dei biglietti, così come θ_1 stesso.

Definizione 5.2 (Funzione di Transizione e Aggiornamento dello Stato [4])**.** La **funzione di transizione** è definita come:

$$\Gamma : \Theta \times H \times S \to \Theta$$

che prende una tupla $(\theta, h, s) \in \Theta \times H \times S$ e restituisce un nuovo biglietto appartenente all'intorno uscente chiuso $\delta^+(\theta) \cup \{\theta\}$.

Per l'**aggiornamento dello stato**, dati uno stato $f \in \Phi$ e un arco $a \in A$, il nuovo stato $g = \mathrm{Up}(f, a)$ generato lungo l'arco è definito da:

$$w(g) := w(f) + w(a), \quad \theta(g) := \Gamma(\theta(f), w(g), e(a))$$

Di conseguenza, per un intero percorso $p = (v_1, \ldots, v_n)$, la sequenza degli stati è ricavata impostando lo stato iniziale $f_1 := \mu(v_1)$ e aggiornando iterativamente $f_i := \mathrm{Up}(f_{i-1}, (v_{i-1}, v_i))$ per $i = 2, \ldots, n$.

Le strutture tariffarie reali non si basano esclusivamente sul percorso effettuato, ma anche su **eventi tariffari** quali cambi tra veicoli, salita su specifici veicoli, o percorsi pedonali. I trasferimenti a piedi tipicamente non alterano lo stato del biglietto, e sono pertanto modellati nel grafo con un peso nullo ($\omega(a) = 0 \in H$) e un evento neutro ($e(a) = s_0 \in S$) tale che $\Gamma(\theta, 0, s_0) = \theta$, senza innescare transizioni [4].

La sintesi degli elementi finora introdotti converge nel concetto centrale di *reti tariffarie condizionali*, le quali permettono di descrivere con precisione una struttura tariffaria [4].

Definizione 5.3 (Rete Tariffaria Condizionale [4])**.** Sia $G = (V, A)$ un grafo di instradamento e dati:

- Un grafo dei biglietti diretto e aciclico $\mathcal{T} = (\Theta, E)$ con funzione di transizione Γ.
- Pesi degli archi $w : A \to H$, presi da un monoide positivo parzialmente ordinato $(H, +, \leq)$.
- Eventi associati agli archi $e : A \to S$.
- Stati tariffari iniziali $\mu : V \to \Phi$.
- Una funzione di prezzo $\pi : \Theta \to \mathbb{Q}^+$ che è monotona non decrescente lungo i percorsi diretti in $\mathcal{T}$, il che significa che se esiste un percorso diretto da θ_1 a θ_2 in $\mathcal{T}$, con $\theta_1, \theta_2 \in \Theta$, allora $\pi(\theta_1) \leq \pi(\theta_2)$.

Tale sestupla $(\mathcal{T}, \Gamma, w, e, \mu, \pi)$ definisce una **Rete Tariffaria Condizionale** $\mathcal{N}$ di G.

Sulla base di tale architettura, gli autori formalizzano il problema del primo arrivo a costo ottimo [4].

Definizione 5.4 (Problema del Primo Arrivo a Prezzo Ottimo (POEAP) [4])**.** Dato un grafo $G = (V, A)$ con una rete tariffaria condizionale $\mathcal{N}$ e una funzione temporale FIFO $c(a) : I \to I$, il problema del Primo Arrivo a Prezzo Ottimo (POEAP) consiste nel determinare l'esatta frontiera Pareto-ottima $P^*_{st} \subseteq P_{st}$ rispetto ai criteri di prezzo π e tempo di arrivo c, tale per cui:

1. Reciproca non-dominazione: Tutti i percorsi in P^*_{st} sono reciprocamente non dominati (nel senso della Definizione 3.3);
2. Copertura completa: Per ogni percorso ammissibile $p \in P_{st}$, esiste almeno un percorso ottimo $p^* \in P^*_{st}$ che lo domina o risulta equivalente ad esso ($\pi(p^*) \leq \pi(p) \land c(p^*) \leq c(p)$).

Classi di biglietti Negli algoritmi di Percorso Minimo Multi-Obiettivo (Multi-Objective Shortest Path (MOSP)), l'ottimalità dei sottopercorsi è essenziale. Poiché l'uso ingenuo del prezzo π per confrontare i percorsi può violare questa proprietà, Euler et al. [4] propongono di partizionare i titoli di viaggio (o stati) Θ in tre classi distinte per gestire correttamente i confronti.

Per formalizzare i criteri di partizione, il modello quantifica l'insieme delle possibili transizioni future in cui un biglietto specifico può evolvere tramite il concetto topologico di *reach* [4].

Definizione 5.5 (Reach [4])**.** Il *reach* (raggiungibilità o copertura) $R(\theta)$ di un vertice $\theta \in \Theta$ è il sottografo indotto da tutti i vertici raggiungibili da θ, ovvero:

$$R(\theta) := \mathcal{T}[\{k \in \Theta \mid \theta \to k\}]$$

L'individuazione del *reach* non è tuttavia sufficiente per garantire la validità dei confronti. Affinché l'ottimalità dei sottopercorsi non venga violata, è fondamentale che le relazioni di dominanza tra due biglietti si mantengano stabili. In altre parole, bisogna impedire che un'etichetta svantaggiosa "sorpassi" un'etichetta migliore a seguito di un aggiornamento tariffario. Tale vincolo è formalizzato dalla proprietà di *no-overtaking* [4].

Definizione 5.6 (Proprietà di no-overtaking [4])**.** Sia $\theta \in \Theta$ un biglietto. Il suo insieme di raggiungibilità $R(\theta)$ soddisfa la **proprietà di no-overtaking** se $\forall k, l \in R(\theta)$ tali che $k \to l$ e per ogni $(h, s) \in H \times S$ vale che:

$$\forall h_1 \in H \text{ tale che } h \leq h_1 \implies \Gamma(k, h, s) \to \Gamma(l, h_1, s).$$

Sfruttando l'ampiezza dell'insieme di raggiungibilità e verificandone le proprietà strutturali, il modello classifica i titoli di viaggio in base al loro grado di comparabilità [4].

Definizione 5.7 (Classi di validità dei titoli di viaggio [4])**.** Sia $G = (V, A)$ una rete di instradamento con una CFN $\mathcal{N} = (\mathcal{T}, \Gamma, w, e, \mu, \pi)$.

- $C_F := \{\theta \in \Theta \mid R(\theta) \text{ è tracciabile e possiede la proprietà di no-overtaking}\}$
- $C_P := \{\theta \in \Theta \setminus C_F \mid \forall k \in R(\theta), \forall s \in S, \forall h_1, h_2 \in H, \Gamma(k, h_1, s) = \Gamma(k, h_2, s)\}$
- $C_N := \Theta \setminus (C_F \cup C_P)$

Affinché $\theta \in \Theta$ appartenga all'insieme C_F, il suo *reach* $R(\theta)$ deve anche essere tracciabile, ovvero contenere un cammino hamiltoniano.

Ordine parziale e monotonia Per ripristinare l'ottimalità dei sottopercorsi, la metodologia introduce un **ordine parziale** $\leq_C$. Siano $\theta_1, \theta_2 \in \Theta$ due biglietti e siano $f_1 = (\theta_1, h_1)$ e $f_2 = (\theta_2, h_2)$ due stati tariffari. Tale ordine parziale tra stati tariffari è definito dagli autori come [4]:

$$f_1 \leq_C f_2 \iff \theta_1 \notin \mathcal{C}_N, \quad h_1 \leq h_2, \text{ e } \begin{cases} \theta_1 = \theta_2, & \text{se } \theta_1 \in \mathcal{C}_P. \\ \exists \theta_1 \to \theta_2 \text{ in } \mathcal{T}, & \text{se } \theta_1 \in \mathcal{C}_F. \end{cases}$$

Questo ordine parziale è strutturato specificamente per escludere confronti "pericolosi", consentendo solo quelli che preservano una coerenza logica tra biglietti e costi. Garantisce che se un percorso è ottimo rispetto a questo ordine, allora ogni suo sottopercorso è ancora valido, ripristinando una forma sufficiente di **ottimalità dei sottopercorsi**.

La **monotonia** assicura che se uno stato tariffario f_1 è meno costoso di f_2, allora questa relazione viene preservata anche dopo un aggiornamento lungo qualsiasi arco $a \in A$:

$$f_1 \leq_C f_2 \implies \mathrm{Up}(f_1, a) \leq_C \mathrm{Up}(f_2, a), \quad \forall a \in A.$$

Ciò significa che la comparabilità tra gli stati si conserva durante l'esecuzione. Grazie a tali proprietà, Euler et al. [4] dimostrano la fattibilità di algoritmi capaci di non generare percorsi sub-ottimali, non scartare percorsi potenzialmente migliori e funzionare in presenza di biglietti dinamici.

La scomposizione dello spazio dei biglietti nelle tre classi C_F, C_P, C_N riveste un ruolo cruciale per le prestazioni dell'algoritmo. Analizzando la struttura a corone concentriche del sistema STIBM di Milano (Mi1-Mi3, Mi4, ecc.), è possibile classificare in modo esatto l'intero catalogo dei titoli di viaggio.

Nel caso specifico dello STIBM, si osserva che l'insieme dei biglietti ricade interamente nella classe di comparabilità completa, ovvero $C_F = \Theta$, mentre $C_P = \emptyset$ e $C_N = \emptyset$. Tale proprietà, analoga a quanto riscontrato in letteratura per altre reti puramente zonali (come la rete MDV [4]), è garantita dal soddisfacimento di due requisiti fondamentali:

1. **Tracciabilità:** Le transizioni tariffarie seguono una gerarchia di espansione chiara (da biglietti urbani verso biglietti extraurbani sempre più ampi).

2. **Proprietà di *no-overtaking*:** Se uno stato tariffario è superiore a un altro, non genererà mai un futuro tariffario peggiore a fronte degli stessi eventi.

Esempio 5.1 (Esempio di comparabilità completa (STIBM):)**.** Siano dati due passeggeri in partenza dalla medesima stazione. Il primo possiede uno stato tariffario f_1 associato a un biglietto "Mi1-Mi3" (costo: 2,20 €), mentre il secondo possiede uno stato f_2 associato a un biglietto "Mi1-Mi4" (costo: 2,60 €). Poiché il primo stato è economicamente più vantaggioso, l'algoritmo stabilisce che $f_1 \leq_C f_2$.

Si supponga che il viaggio prosegua e che entrambi oltrepassino il confine per entrare nella zona Mi4. In base alle funzioni di transizione Γ della *Conditional Fare Network*, lo stato del passeggero 1 subirà un subirà un adeguamento tariffario che porterà il suo costo cumulativo esattamente a 2,60 €. Il passeggero 2, avendo già la corretta copertura zonale, non subirà alcun sovrapprezzo, mantenendo il proprio costo a 2,60 €.

Come si evince dall'esempio, il costo dello stato originariamente più economico (f_1) raggiunge il costo dello stato più oneroso (f_2), ma non lo supera mai. Non esistono, all'interno dello STIBM, "penali asimmetriche" che porterebbero, ad esempio, il passeggero 1 a pagare un totale di 3,50 € a causa della sua

scelta iniziale. Questa impossibilità matematica di subire “sorpassi" di costo (*no-overtaking*) garantisce che un biglietto inizialmente più economico non si trasformerà mai, in futuro, in una scelta globalmente peggiore. Di conseguenza, tutti i biglietti dello STIBM risultano sempre comparabili tra loro ($C_F = \Theta$), autorizzando l'algoritmo a potare in totale sicurezza i rami di ricerca localmente più costosi.

Esempio 5.2 (Esempio di incomparabilità:)**.** Per comprendere l'utilità teorica delle classi di incomparabilità, si consideri una struttura tariffaria fittizia o asimmetrica non presente nello STIBM standard. Si supponga l'esistenza di due titoli di viaggio:

- θ_A: Un “Pass Giornaliero Urbano" (costo 7€) che permette viaggi illimitati sui bus, ma vieta esplicitamente l'accesso al treno speciale per l'aeroporto.

- θ_B: Un “Ticket Aeroportuale Singolo" (costo 8€) che permette l'accesso al treno speciale, ma non consente trasferimenti sui bus urbani.

In questo scenario, θ_A e θ_B non sono ordinabili. Nonostante θ_A costi meno di θ_B, possedere θ_A potrebbe rivelarsi uno svantaggio determinante se il prossimo evento del viaggio fosse “salire sul treno aeroportuale" (obbligando all'acquisto di un nuovo biglietto intero). Viceversa, possedere θ_B sarebbe inutile se il viaggio proseguisse in bus. Le funzioni di transizione Γ di questi due biglietti divergono in base agli eventi futuri, violando la proprietà di *no-overtaking*. Di conseguenza, l'algoritmo non può scartare a priori nessuno dei due stati e deve classificarli come incomparabili ($\theta_A, \theta_B \in C_N$), costringendo il motore di calcolo a preservarli entrambi all'interno dei *bag* $B_k(p)$ per esplorare parallelamente le due diramazioni. L'assenza di tali anomalie nel grafo dei biglietti milanese giustifica le elevate prestazioni computazionali registrate durante i test, in quanto l'algoritmo McRAP non è mai costretto a biforcare lo spazio di ricerca per preservare stati incomparabili.

McRAP

Sebbene in letteratura il termine McRAPTOR indichi genericamente le estensioni multi-obiettivo del framework originale di Delling et al. [3], nel contesto del presente lavoro, riprendendo la denominazione specifica introdotta nell'articolo di riferimento di Euler et al. [4], faremo riferimento con il nome di McRAP alla specifica implementazione e architettura software sviluppata per integrare le Conditional Fare Networks e le regole dello STIBM milanese. La soluzione proposta adatta l'algoritmo McRAP, una versione multi-obiettivo di RAPTOR. A differenza del RAPTOR classico che mantiene una sola etichetta per fermata, McRAP mantiene un insieme Pareto-efficiente di etichette non dominate che rappresentano diverse combinazioni di tempo e prezzo accumulato. Un'etichetta ne domina un'altra se arriva prima o allo stesso momento e costa meno o uguale.

Tabella della notazione del Capitolo 5

La seguente tabella riassume i simboli e le notazioni matematiche introdotte in questo capitolo relative alle Conditional Fare Networks e alle estensioni algoritmiche (McRAP, rRaptor, Range-McRAP).

Simbolo	Definizione / Significato
$(H, +, \leq)$	Monoide positivo parzialmente ordinato per la gestione dei pesi.
$\mathcal{T} = (\Theta, E)$	Grafo dei biglietti.
Θ	Insieme dei nodi (biglietti) del grafo $\mathcal{T}$.
S	Insieme degli eventi tariffari.
$\omega(a)$	Peso dell'arco a nel grafo di instradamento G.
$e(a)$	Evento tariffario associato all'arco a.
f	Stato tariffario composto da titolo di viaggio e peso accumulato ($f \in \Theta \times H$).
Φ	Spazio totale degli stati tariffari ($\Phi := \Theta \times H$).
$\mu(v)$	Stato tariffario iniziale associato a ciascun nodo $v \in V$.
Γ	Funzione di transizione dello stato tariffario.
π	Funzione di prezzo monotona non decrescente associata ai biglietti.
$R(\theta)$	Insieme di raggiungibilità (*reach*) di un vertice θ in $\mathcal{T}$.
C_F, C_P, C_N	Classi di validità dei titoli di viaggio.
$\leq_C$	Ordine parziale definito tra stati tariffari per preservare l'ottimalità dei sottopercorsi.
$B_k(p)$	Insieme (*bag*) di etichette Pareto-efficienti e non dominate per la fermata p nel round k.
$L = (\tau, f, \pi_{back})$	Tupla dell'etichetta non dominata (orario, stato tariffario, puntatore al nodo padre).
ϵ	Parametro di tolleranza temporale (*slack time*) per l'algoritmo *Tight-BMRAP*.

Tabella 5.1: Notazione del Capitolo 5.

5.1 Il motore di calcolo: architettura dell'algoritmo McRAP

In questa sezione viene formalizzata l'architettura del nucleo computazionale del progetto. Il modello estende il framework Raptor per gestire la ricerca di percorsi ottimi all'interno di una frontiera di Pareto multi-obiettivo, in cui la valutazione di dominanza si estende oltre l'orario di arrivo e il numero di cambi, includendo il costo monetario e lo stato di validità del titolo di viaggio.

5.1.1 Inizializzazione e struttura dello spazio degli stati

All'avvio della ricerca, l'algoritmo identifica lo stato tariffario iniziale basandosi sulla localizzazione geografica (e la conseguente zona tariffaria) della fermata di

origine p_s. Attraverso un'operazione insiemistica, il sistema seleziona all'interno del catalogo tariffario il titolo di viaggio base che garantisca la copertura della zona di partenza minimizzando il costo iniziale, generando uno stato temporale nullo.

Lo spazio delle soluzioni è strutturato per round successivi $k \in \{1, \dots, K\}$, dove k rappresenta il numero massimo di corse utilizzate. Per ogni fermata $p \in S$, l'algoritmo mantiene un insieme (o *bag*) $B_k(p)$ di etichette mutuamente non dominate. Ogni etichetta $L \in B_k(p)$ è definita dalla tupla $L = (\tau, f, \pi_{back})$, dove:

- τ: Orario di arrivo calcolato.
- f: Stato tariffario corrente (che include il titolo di viaggio attivo e le relative risorse consumate, come il tempo di validità residuo).
- π_{back}: Puntatore al nodo padre, necessario per la ricostruzione del percorso.

5.1.2 Dinamica dell'esplorazione: imbarco, transito e trasferimento

La propagazione delle etichette all'interno di ogni round k si articola in tre fasi logiche:

- **Fase di Imbarco e Attesa:** Per ogni linea che interseca l'insieme delle fermate marcate nel round precedente, l'algoritmo identifica la prima corsa cronologicamente utile alla partenza. Un contributo metodologico di questa implementazione risiede nella gestione rigorosa del tempo di attesa in banchina: il differenziale temporale tra l'arrivo in stazione e la partenza del mezzo viene applicato direttamente come funzione di decadimento sullo stato tariffario f. Ciò garantisce che il consumo della validità temporale del biglietto (es. i 90 minuti previsti dallo STIBM) venga calcolato in modo continuo e ininterrotto. Per le infrastrutture metropolitane, il tempo di attesa viene incrementato di una costante di sicurezza temporale per modellare i tempi di discesa ai binari e l'attraversamento dei tornelli.
- **Fase di Transito (Discesa):** Durante l'avanzamento lungo la sequenza di fermate della linea corrente, l'algoritmo calcola le transizioni di stato per ogni singola tappa. Se l'evoluzione temporale e tariffaria genera una nuova etichetta L' che risulta non dominata in senso paretiano rispetto all'insieme storico $B^*(p)$ e all'insieme corrente $B_k(p)$ della fermata di destinazione, tale etichetta viene aggiunta alla borsa e la fermata viene marcata per la propagazione nei round successivi.
- **Fase dei Percorsi Pedonali:** Conclusa la scansione delle linee, il modello valuta gli archi di trasferimento a piedi (*foot-paths*). In questa fase viene processato l'evento di interscambio, applicando i tempi di percorrenza pedonale e aggiornando contestualmente lo stato tariffario per riflettere le specifiche regole di trasferimento intermodale previste dal regolamento tariffario.

5.1.3 Estrazione della soluzione ottima

Terminati i round di esplorazione, l'algoritmo attiva la procedura di estrazione dell'insieme di Pareto definitivo. Partendo dalla fermata di destinazione p_t, il sistema opera un attraversamento a ritroso del grafo logico sfruttando esclusivamente i puntatori π_{back} memorizzati in ciascuna etichetta non dominata. Questa fase di *backtracking* permette di ricostruire in modo esatto l'intera catena di eventi topologici e tariffari (tempi di camminata, attese in banchina, transiti sui mezzi operativi) che hanno condotto allo stato finale, restituendo la sequenza formale del cammino minimo multi-obiettivo.

5.1.4 Ottimizzazione avanzata: la logica Tight-BMRAP

Per garantire la scalabilità del modello su reti di dimensioni metropolitane, come il bacino milanese, l'architettura algoritmica è stata estesa integrando i principi della variante ottimizzata *Tight-BMRAP*. Il nucleo di questa estensione risiede nell'introduzione di un parametro temporale di tolleranza, definito *slack time* (ϵ), che instaura un sofisticato meccanismo di dominanza "rilassata".

L'intuizione logico-matematica alla base di questo approccio interviene sulle inefficienze della dominanza di Pareto standard, la quale preserverebbe un nuovo percorso J_1 qualora offrisse anche un solo minuto di vantaggio (sull'orario di arrivo o sul consumo del biglietto) rispetto a un'alternativa J_2 già esplorata. Introducendo lo *slack time* ϵ, il sistema impone una soglia di tolleranza per il filtraggio: se il percorso J_1 registra un lieve miglioramento rispetto a J_2, ma tale vantaggio temporale è strettamente confinato all'interno del margine ϵ (ovvero $\tau_{arr}(J_2) \leq \tau_{arr}(J_1) + \epsilon$), le due soluzioni vengono considerate temporalmente equivalenti. Di conseguenza, in assenza di un netto e reale vantaggio strategico (come un prezzo strettamente inferiore o l'accesso a una zona tariffaria aggiuntiva), il miglioramento non è più sufficiente per giustificare la conservazione di J_1, il quale viene dominato e scartato a favore di J_2.

L'adozione del filtro logico *Tight-BMRAP* si rivela uno strumento cruciale per bilanciare due esigenze contrapposte: da un lato, garantisce l'efficienza computazionale prevenendo efficacemente l'esplosione combinatoria delle etichette intermedie; dall'altro, preserva il rigore teorico necessario per isolare l'esatta frontiera di Pareto in presenza di sistemi di bigliettazione complessi e non additivi come lo STIBM.

5.2 Estensione Range: L'algoritmo rRaptor

Oltre alla risoluzione del singolo quesito di instradamento, una funzionalità avanzata richiesta dai moderni sistemi di pianificazione è la capacità di fornire una panoramica dei viaggi disponibili in un determinato arco temporale. Per rispondere a questa esigenza, è stata implementata una versione dell'algoritmo rRaptor, ispirata al lavoro di *Delling, Pajor e Werneck* in *"Round-Based Public Transit Routing"* [3].

5.2.1 Logica self-pruning e Range temporali

A differenza del RAPTOR standard, che calcola il tempo minimo di arrivo per una singola partenza puntuale, l'estensione rRAPTOR opera su un intero insieme di orari di partenza Ψ compresi in un intervallo $[\Delta_{start}, \Delta_{end}]$. La caratteristica distintiva di questa architettura è l'elaborazione *Latest-to-Earliest*: gli orari di partenza vengono esaminati rigorosamente in ordine decrescente, dal più tardivo al più anticipato.

Questo approccio permette di sfruttare una potente proprietà di *self-pruning* (auto-potatura): un viaggio che inizia prima è considerato utile e viene esplorato se e solo se garantisce un orario di arrivo strettamente precedente rispetto a tutti i viaggi che partono dopo di esso. A livello logico, i tempi di arrivo ottimi calcolati vengono mantenuti persistenti in memoria tra un'iterazione e l'altra; questo funge da barriera temporale, permettendo all'algoritmo di scartare istantaneamente i rami di esplorazione associati a partenze anticipate che producono soluzioni peggiori o uguali a quelle già assicurate dalle partenze successive.

5.2.2 Filtraggio e costruzione della frontiera di Pareto

Ad ogni iterazione per uno specifico orario $\tau \in \Psi$, l'algoritmo esegue un'esplorazione completa procedendo per K round discreti. Al termine della scansione, le etichette non dominate che raggiungono la fermata di destinazione vengono temporaneamente aggregate in un insieme di soluzioni candidate.

Tuttavia, l'accumulo incondizionato dei risultati genererebbe soluzioni ridondanti per l'utente finale. Per estrarre esclusivamente le alternative di viaggio effettivamente competitive, il sistema applica in fase di post-elaborazione un rigoroso filtro basato sulla dominanza stretta (3.4). Un itinerario viene validato e presentato come soluzione finale solo se il suo orario di arrivo costituisce un miglioramento reale e assoluto rispetto al limite temporale stabilito da tutti i viaggi partiti successivamente.

Questo processo di filtraggio a posteriori garantisce che l'output dell'algoritmo rappresenti l'esatta frontiera di Pareto nel dominio del tempo. Vengono così restituite all'utente unicamente le opzioni "significative", ossia i viaggi per i quali non esiste alcuna alternativa che consenta di partire più tardi e arrivare contemporaneamente o in anticipo, riducendo drasticamente il rumore informativo senza sacrificare la rigorosa ottimalità matematica delle soluzioni.

5.3 Integrazione finale: Range-McRAP

Infine è stata implementata l'integrazione tra le capacità multi-obiettivo del modello McRAP e l'efficienza delle interrogazioni su finestre temporali (*Range Queries*) proprie dell'estensione rRAPTOR. Questa fusione genera un'architettura in grado di calcolare e analizzare l'esatta frontiera di Pareto su un intero intervallo temporale continuo $[\Delta_{start}, \Delta_{end}]$, integrando simultaneamente i criteri di tempo, costo e trasferimenti.

L'algoritmo opera secondo un paradigma logico a due fasi. Inizialmente, il sistema estrae dalla tabella oraria l'insieme discreto Ψ di tutti i possibili istanti

di partenza disponibili presso la stazione di origine all'interno dell'intervallo richiesto. Seguendo la logica *Latest-to-Earliest*, tali orari vengono elaborati rigorosamente in ordine cronologico inverso.

Per ogni singolo istante di partenza $\tau_{dep} \in \Psi$, il sistema istanzia ed esegue un'esplorazione multi-obiettivo completa. La vera innovazione architetturale risiede nel fatto che le etichette non dominate generate in ciascuna di queste esecuzioni non rimangono isolate localmente, ma confluiscono in uno spazio delle soluzioni globale. Questo contenitore aggregato funge da "super-frontiera" paretiana, permettendo all'algoritmo di effettuare confronti incrociati e propagare la dominanza tra alternative di viaggio che originano in orari e iterazioni differenti.

5.3.1 Criteri di dominanza Multi-Obiettivo

Il fulcro di questa integrazione risiede nell'espansione delle regole di dominanza. Mentre nell'algoritmo multi-obiettivo puntuale il confronto si limita alla minimizzazione dell'orario di arrivo e del costo tariffario, nel contesto di un'interrogazione su intervallo si include la massimizzazione dell'orario di partenza (τ_{dep}) come ulteriore criterio decisionale.

Un'etichetta consolidata L_{old} domina una nuova etichetta candidata L_{new} se e solo se verifica la condizione di dominanza (Definizione 3.3) su tutti i criteri valutati:

$$(\tau_{dep,old} \geq \tau_{dep,new}) \wedge (\tau_{arr,old} \leq \tau_{arr,new}) \wedge (\pi_{old} \leq \pi_{new}) \wedge (w_{old} \leq_C w_{new}).$$

In termini pratici, un viaggio candidato viene considerato paretianamente superfluo se all'interno dell'insieme globale esiste già un'alternativa che consenta di partire successivamente (o nel medesimo istante), giungere a destinazione antecedentemente (o in contemporanea), sostenendo un costo economico non superiore e preservando uno stato delle risorse tariffarie altrettanto vantaggioso.

5.3.2 Estrazione e ricostruzione degli itinerari

Una volta che è stato eseguito l'algoritmo, il sistema procede all'estrazione degli itinerari effettivi, analizzando esaustivamente l'insieme risultante e ordinando le opzioni valide.

La criticità logica in questa fase è rappresentata dall'operazione di *backtracking* multi-livello. Sfruttando la ricorsione sui puntatori ai nodi padri memorizzati in ogni etichetta, l'algoritmo ripercorre a ritroso l'intera sequenza di stati fino a raggiungere l'evento di genesi del viaggio. Questa architettura consente di ricostruire l'esatta topologia di itinerari complessi, quantificando esplicitamente:

- **Dinamica Intermodale:** L'identificazione rigorosa delle linee utilizzate e l'orientamento geometrico delle tratte percorse.
- **Scomposizione Temporale:** La differenziazione analitica dei tempi di transito, distinguendo in modo netto i tempi spesi per il trasferimento pedonale dai tempi di attesa in banchina per le coincidenze.

L'output finale costituisce la sintesi metodologica di questa architettura: un modello matematico-computazionale in grado di dominare la complessità e le asimmetrie dei vincoli imposti dallo STIBM di Milano, restituendo all'utente finale un ventaglio di opzioni rigorosamente verificate sulla complessa frontiera di Pareto spazio-temporale e tariffaria.

5.4 Estensione teorica delle ottimizzazioni al dominio tariffario

Nei capitoli precedenti, le logiche di ottimizzazione algoritmica basate sul paradigma del *warm-start* e sulla traslazione spaziale sono state formalizzate e validate nel dominio temporale. L'adozione delle Reti Tariffarie Condizionali porta naturalmente all'estensione dei criteri di potatura (*pruning*) allo spazio degli stati multi-obiettivo, considerando simultaneamente i vincoli temporali e tariffari.

A differenza del tempo, che gode di proprietà strettamente additive, il costo all'interno di una CFN è governato da funzioni di transizione di stato non lineari (es. tariffe a macro-zone, validità temporali massime). Di conseguenza, l'applicazione di un *limite inferiore* geometrico al prezzo risulta mal posta.

Tuttavia, analizzando le proprietà strutturali dei moderni sistemi di bigliettazione urbana (quali lo STIBM di Milano [5]), è possibile dimostrare che l'uniformità del prezzo all'interno dei bacini zonali permette di disaccoppiare parzialmente il costo dal tempo, rendendo le ottimizzazioni spaziali nativamente compatibili con la dominanza paretiana, a patto di introdurre rigorosi controlli sullo stato di validità del titolo.

Al fine di estendere i quattro casi di ottimizzazione al dominio multi-obiettivo, si introducono le seguenti definizioni formali relative ai vincoli tariffari.

Definizione 5.8 (**Macro-Zone e Vincolo Temporale (TTL)**)**.**
Sia $\mu(p_v) \in \Phi$ lo stato tariffario iniziale acquisito presso la fermata $p_v \in V$. Si definiscono due vincoli fondamentali per la validità di tale stato:

1. **Equivalenza Zonale:** Si definisce $Z(p_v)$ la macro-zona tariffaria cui appartiene la fermata p_v. Per ogni coppia di fermate $p_u, p_v \in V$, se $Z(p_u) = Z(p_v)$, il costo di accesso alla rete è invariante, ovvero $\pi(\mu(p_u)) = \pi(\mu(p_v))$.

2. **Time To Live (TTL):** Ogni titolo di viaggio possiede una durata massima di validità Δ_{max}. Un itinerario P originato in p_v al tempo $\tau_{dep}(P, p_v)$ è tariffariamente ammissibile se l'orario di arrivo a destinazione soddisfa il vincolo:
$$\tau_{arr}(P, p_g) - \tau_{dep}(P, p_v) \leq \Delta_{max}.$$

Sulla base di tali invarianti, si procede alla riformulazione analitica delle logiche di traslazione per la frontiera di Pareto.

5.4.1 Caso 1: Traslazione spaziale multi-obiettivo

Nel dominio temporale analizzato nel Capitolo 4, la traslazione spaziale all'indietro si esauriva in una concatenazione, definita da $P_{unione} = P_{a,b}P_{b,g}$. L'unico vincolo imposto era il rispetto cronologico al nodo di aggancio.

Nel contesto multi-obiettivo governato dalle *Conditional Fare Networks*, l'estensione all'indietro del percorso introduce delle complessità: anticipare il punto di imbarco dal nodo p_b al nodo antecedente p_a modifica le condizioni iniziali del viaggio: lo stato tariffario iniziale non è più $\mu(p_b)$, bensì $\mu(p_a) \in \Phi$.

Da un punto di vista algoritmico, valutare il nuovo itinerario P_{unione} richiederebbe la rigorosa propagazione del nuovo stato iniziale $f_{p_a} = \mu(p_a)$ lungo l'intera sequenza di transizioni Γ fino alla destinazione finale p_g, al fine di accertare il costo cumulativo $\pi(f_{p_g}^{unione})$ e verificare l'assenza di dominanza. Tale ricalcolo sistematico vanificherebbe i benefici prestazionali del *warm-start*.

Tuttavia, sfruttando le proprietà strutturali dei titoli di viaggio a macrozone (come l'Equivalenza Zonale e il vincolo temporale TTL), è possibile aggirare la computazione delle transizioni e validare l'ottimalità tariffaria della concatenazione in modo intrinseco, come dimostrato dalla seguente proposizione.

Proposizione 5.1 (**Ottimalità Tariffaria Intrinseca**)**.**

Siano p_a e p_b due fermate appartenenti alla medesima macro-zona tariffaria ($Z(p_a) = Z(p_b)$). Se il viaggio concatenato P_{unione} soddisfa il vincolo temporale TTL, esso preserva il costo originario $\pi(P_{unione}) = \pi(P_{b,g})$. Inoltre, qualsiasi corsa alternativa $t' \in D$ originata dal vicinato V_a risulta dominata sul piano tariffario, riconducendo la condizione di potatura al solo limite inferiore temporale $H(a', g)$.

Dimostrazione. Essendo $Z(a) = Z(b)$, il costo di obliterazione iniziale è identico: $\pi(\mu(p_a)) = \pi(\mu(p_b))$. Poiché il percorso topologico a valle di p_b rimane inalterato, le transizioni di zona Γ del viaggio P_{unione} sono equivalenti a quelle di $P_{b,g}$. Ne consegue che $\pi(P_{unione}) = \pi(P_{b,g})$.

Sia $t' \in D$ una corsa alternativa candidata, originata in $p_s \in V_a$. Essendo p_s nel vicinato pedonale di p_a, si assume $Z(p_s) = Z(p_a)$. Il costo di base per accedere a T è $\pi(\mu(p_s)) = \pi(\mu(p_a)) = \pi_{unione}$. Per la proprietà di **monotonia della funzione di prezzo** π ([4]) definita sulle CFN, il costo finale di un qualsiasi itinerario basato su t' non potrà mai essere inferiore al suo costo corrente: $\pi(f_{t'}) \geq \pi_{unione}$. Poiché la corsa t' non potrà mai offrire un vantaggio economico, essa può dominare P_{unione} *solo* offrendo un vantaggio temporale. Pertanto, t' viene dichiarata dominata e potata a priori se e solo se verifica il test topologico:

$$\tau_{arr}(t', p'_a) + H(a', g) \geq \tau_{arr}(P_{unione}, p_g).$$

Il test di dominanza tariffaria è dunque intrinsecamente soddisfatto dalla topologia a zone. □

5.4.2 Caso 2: Assorbimento del ritardo e scadenza del titolo

Nel dominio puramente temporale analizzato nel Capitolo 4, la gestione di una partenza posticipata si basava sull'assorbimento cronologico del ritardo tramite lo *slack time* disponibile presso il nodo di interscambio. Affinché il riuso del sottopercorso $P_{d,g}$ fosse garantito, era sufficiente che l'utente riuscisse a intercettare fisicamente la coincidenza.

L'integrazione delle Reti Tariffarie Condizionali impone tuttavia di valutare l'impatto strutturale del ritardo sullo stato di validità del titolo di viaggio. In un sistema tariffario reale, posticipare l'inizio del viaggio significa far slittare in avanti l'evento di validazione del biglietto, innescando il vincolo del *Time To Live* (TTL) in un istante cronologicamente successivo.

Si potrebbe ipotizzare che una perturbazione in ritardo sulla tabella di marcia esponga il passeggero al rischio di scadenza del titolo prima del raggiungimento della destinazione finale. Al contrario, qualora il ritardo venga interamente assorbito dai tempi di attesa della rete, si ha che l'orario di partenza avanza, ma l'orario di arrivo a destinazione rimane equivalente a quello del viaggio storico pre-calcolato.

Come illustrato nella seguente proposizione, questa dinamica riduce il tempo di validità consumato dal passeggero, garantendo che il riuso del percorso precalcolato sia sicuro sia dal punto di vista spaziale che economico.

Proposizione 5.2 (**Soddisfacimento del TTL**)**.**
Se il viaggio originale $P_{b,g}$ era tariffariamente ammissibile rispetto al vincolo TTL, e la corsa alternativa posticipata P' assorbe validamente il ritardo presso il nodo d ($\tau_{arr}(P', d) + \Delta_{trans}(d) \leq \tau_{dep}(P_{b,g}, d)$), il nuovo viaggio concatenato $P_{unione} = P'P_{d,g}$ è garantito contro la scadenza del titolo di viaggio.

Dimostrazione. Sia $\Delta T_{old} = \tau_{arr}(P, g) - \tau_{dep}(P, b)$ la durata complessiva del viaggio originario. Essendo P ammissibile, si ha $\Delta T_{old} \leq \Delta_{max}$. Nel nuovo scenario, l'utente posticipa la partenza a $\tau_{dep}(P', b) > \tau_{dep}(P, b)$, ma, assorbendo il ritardo, giunge a destinazione al medesimo orario $\tau_{arr}(P, g)$. La nuova durata del viaggio risulta:

$$\Delta T_{new} = \tau_{arr}(P, g) - \tau_{dep}(P', b).$$

Essendo il minuendo inalterato e il sottraendo strettamente maggiore, si deduce che $\Delta T_{new} < \Delta T_{old} \leq \Delta_{max}$, affermando l'ottimalità della traslazione. □

5.4.3 Caso 3: Ottimalità dei sottopercorsi e ordine parziale

Nel dominio temporale, l'estrazione logica di un sottopercorso $P_{b,g}$ a partire da un nodo intermedio p_b trova il suo fondamento teorico nel Principio di Ottimalità di Bellman: ogni segmento terminale di un cammino ottimo è, a sua volta, il cammino ottimo per la relativa sottomappa. L'unica condizione di validità richiesta era il raggiungimento del nodo di aggancio in sincronia con l'orario previsto.

La transizione al multi-obiettivo governato dalle *Conditional Fare Networks* determina una rottura di questo paradigma classico. Negli algoritmi di ricerca

dei cammini minimi multi-obiettivo, l'uso ingenuo del prezzo per confrontare i percorsi viola la condizione di ottimalità dei sottopercorsi. All'interno di una rete tariffaria, infatti, un percorso localmente dominato potrebbe rivelarsi globalmente ottimo a causa di specifiche e complesse regole di transizione successive.

Come abbiamo visto all'inizio del capitolo, per ripristinare l'ottimalità dei sottopercorsi senza rilassare eccessivamente i criteri di dominanza, il modello CFN definisce uno specifico ordine parziale per gli stati tariffari, denotato con $\leq_C$. Tale ordine si basa sulla partizione dell'insieme dei biglietti in tre gruppi di comparabilità disgiunti: C_F (comparabilità completa), C_P (comparabilità parziale) e C_N (nessuna comparabilità).

La convergenza algoritmica su tali strutture si fonda sulla *weak subpath optimality* (ottimalità debole dei sottopercorsi) [4]. Affinché questa proprietà sia valida, la funzione di aggiornamento tariffario (*update function*) deve risultare monotona lungo tutti gli archi rispetto all'ordine parziale $\leq_C$ [4]. Di seguito si riporta la formalizzazione e la dimostrazione di tale proprietà.

Proposizione 5.3 (**Ottimalità Debole dei Sottopercorsi** [4])**.**
Sia $G = (V, A)$ una rete di instradamento e $(\mathcal{T}, \Gamma, w, e, \mu, \pi)$ la sua rete tariffaria condizionale. Sia $P \in P^*_{a,g}$ un percorso ottimo rispetto allo stato (state-optimal) da p_a a p_g per alcuni $p_a, p_g \in V$. Allora, esiste un percorso $P' = (p_a = v_0, \ldots, v_n = p_g) \in P^*_{a,g}$ con $f(P) = f(P')$, tale che ogni suo sottopercorso $P'' = (v_0, \ldots, v_l)$, con $l < n$, di P' è un percorso ottimo rispetto allo stato da v_0 a v_l.

Dimostrazione. La dimostrazione che segue riprende fedelmente l'impianto logico e i passaggi presentati in [4] (Prop. 4.1).

Sia $P = (p_a = v_0, v_1, \ldots, v_{n-1}, v_n = p_g) \in P^*_{a,g}$ un percorso ottimo rispetto allo stato con stati tariffari $(f_0, \ldots, f_n)$. Si assuma l'esistenza di un percorso $\tilde{P} = (p_a = u_0, u_1, \ldots, u_{l-1} = v_{n-1}, u_l = p_g)$ con stati tariffari $(g_0, g_1, \ldots, g_l)$ e $g_0 = f_0$. Per la proprietà di monotonia, possiamo scegliere $\tilde{P}$ come un percorso semplice.

Sia k il più grande intero tale che $v_{n-k} = u_{l-k}$, ovvero i percorsi $(v_{n-k}, \ldots, v_n)$ e $(u_{l-k}, \ldots, u_l)$ sono identici. Ora, assumiamo $g_{l-k-1} <_C f_{n-k-1}$. Per definizione, si ha:

$$f_{n-k} = \mathrm{Up}(f_{n-k-1}, (v_{n-k-1}, v_{n-k})),$$

$$g_{l-k} = \mathrm{Up}(g_{l-k-1}, (u_{l-k-1}, u_{l-k})).$$

Applicando la proprietà di monotonia si ottiene $g_{l-k} \leq_C f_{n-k}$. Ripetendo il processo per $i \in \{k-1, \ldots, 0\}$, troviamo $g_l \leq_C f_n$. Poiché P era ottimo rispetto allo stato, ne consegue che $g_l = f_n$, e conseguentemente anche $\tilde{P}$ è ottimo rispetto allo stato. Poiché il numero di percorsi in G è finito, si può ripetere questa procedura per trovare il percorso P'. □

La proposizione precedente dimostra che, sebbene non tutti i sottopercorsi di un percorso ottimo siano necessariamente ottimi a loro volta, è possibile scartare le varianti sub-ottime preservando comunque nell'insieme delle soluzioni un

percorso con il medesimo stato tariffario finale. In questo modo, l'applicazione degli algoritmi *label-setting* risulta garantita.

Tuttavia, nel contesto delle ottimizzazioni *warm-start* del Caso 3, l'algoritmo non costruisce il viaggio progressivamente dall'origine in avanti, bensì tenta di riutilizzare direttamente un segmento terminale pre-calcolato $P_{b,g}$. Di conseguenza, un arrivo sincronizzato (o anticipato) al nodo intermedio b non costituisce più una condizione sufficiente per certificare l'ottimalità del riuso passivo. La verifica formale richiede la rigorosa applicazione dell'ordine parziale $\leq_C$ per garantire che il nuovo utente non erediti uno stato tariffario incomparabile o peggiore rispetto all'esplorazione originaria.

Proposizione 5.4 (Riuso tramite Ordine Parziale $\leq_C$).

Sia $f_{old} \in \Phi$ lo stato tariffario ereditato al nodo intermedio b dal viaggio pre-calcolato originario. Sia $f_{new} \in \Phi$ lo stato tariffario dell'utente al momento della nuova interrogazione dal nodo b. Il sottopercorso $P_{b,g}$ conserva la propria ottimalità paretiana e può essere riutilizzato senza ricalcolo se e solo se:

$$f_{new} \leq_C f_{old}$$

dove $\leq_C$ rappresenta l'ordine parziale degli stati definito per preservare l'ottimalità dei sottopercorsi nelle CFN.

La relazione $\leq_C$ garantisce la rigorosa comparabilità degli stati all'interno dei gruppi di validità dei titoli di viaggio. La monotonia della CFN assicura che, a parità di percorso fisico $P_{b,g}$, il costo finale generato dalle transizioni innescate da f_{new} sarà sempre minore o uguale al costo originariamente generato da f_{old}, confermando analiticamente la *weak subpath optimality* dell'itinerario concatenato.

5.4.4 Caso 4: Traslazione all'indietro con attesa multi-obiettivo

Il quarto e ultimo scenario operativo estende la traslazione spaziale all'indietro introducendo un disallineamento temporale alla sorgente. L'utente, partendo da un nodo antecedente p_a, è costretto ad attendere in banchina una corsa di avvicinamento successiva P'.

Nel framework delle *Conditional Fare Networks*, questo scenario non si limita ad alterare l'orario di arrivo al nodo di interscambio, ma sposta in avanti l'istante di innesco dello stato tariffario. L'obliterazione del titolo di viaggio, e il conseguente consumo del *Time To Live* (TTL), hanno inizio esattamente al tempo di partenza $\tau_{dep}(P', p_a)$. Affinché l'assemblaggio del viaggio concatenato $P_{unione} = P'_{a,d}P_{d,g}$ (dove p_d è il nodo di interscambio) conservi la sua ottimalità paretiana senza richiedere il ricalcolo dell'algoritmo, è necessario definire una condizione di validità globale che fonda i vincoli topologici, temporali ed economici.

Si definisca, come nel dominio temporale, l'insieme D delle corse alternative in partenza dal vicinato V_a durante l'attesa alla sorgente.

Proposizione 5.5 (**Condizione di Ottimalità Globale Multi-Obiettivo**).

Siano p_a e le fermate alternative $p_s \in V_a$ appartenenti alla medesima macrozona tariffaria ($Z(p_a) = Z(p_s)$). Il viaggio concatenato P_{unione} preserva la sua ottimalità assoluta se e solo se supera positivamente il seguente sistema di validazione a triplo filtro:

1. ***Assorbimento:*** *La corsa in ritardo P' deve intercettare fisicamente la coincidenza pre-calcolata al nodo p_d:*

$$\tau_{arr}(P', p_d) + \Delta_{trans}(p_d) \leq \tau_{dep}(P_{d,g}, p_d).$$

2. ***Ammissibilità Tariffaria (TTL):*** *Il tempo di percorrenza effettivo non deve eccedere la durata massima di validità del biglietto Δ_{max}:*

$$\tau_{arr}(P_{unione}, p_g) - \tau_{dep}(P', p_a) \leq \Delta_{max}.$$

3. ***Ottimalità Euristica (Dominanza):*** *Nessuna corsa candidata $t' \in D$ (con primo nodo di svincolo p'_a) deve poter dominare temporalmente l'orario di arrivo garantito da P_{unione} tramite il limite inferiore $H(a', g)$:*

$$\tau_{arr}(t', p'_a) + H(a', g) \geq \tau_{arr}(P_{unione}, p_g).$$

Dimostrazione. La dimostrazione procede analizzando i tre filtri. Il soddisfacimento del Filtro 1 garantisce, come dimostrato nel Caso 2, l'esistenza fisica del percorso P_{unione} bypassando la necessità di esplorare le coincidenze successive. Il Filtro 2 certifica che l'evoluzione dello stato tariffario originato al tempo $\tau_{dep}(P', p_a)$ rimanga valida per l'intera estensione del grafo fino a p_g, scongiurando l'esigenza di calcolare transizioni tariffarie aggiuntive (Γ) per titoli scaduti.

Infine, per il Filtro 3, si assuma l'esistenza di una corsa candidata $t' \in D$. Essendo $Z(p_a) = Z(p_s)$, il costo di obliterazione iniziale è invariante: $\pi(\mu(p_a)) = \pi(\mu(p_s)) = \pi_{unione}$. Per la proprietà di monotonia della funzione di prezzo sulle CFN [4], il costo finale di qualsiasi itinerario $P_{t'}$ basato su t' soddisfa $\pi(P_{t'}) \geq \pi(f_{t'}) \geq \pi_{unione}$. Poiché t' non potrà mai offrire un costo strettamente inferiore a P_{unione}, la sua unica possibilità di dominanza risiede nel criterio temporale. Il *limite inferiore* non rilassabile $H(a', g)$ stima il limite inferiore del tempo di arrivo per t'. Essendo verificata per ipotesi la disuguaglianza del Filtro 3, si ha analiticamente che $\tau_{arr}(P_{t'}, p_g) \geq \tau_{arr}(P_{unione}, p_g)$. Nessuna corsa $t' \in D$ può dunque risultare strettamente dominante rispetto a P_{unione}, confermando l'ottimalità paretiana del viaggio concatenato. □

Conclusione della Sezione

La formalizzazione matematica di queste quattro logiche di traslazione dimostra che, sebbene le *Conditional Fare Networks* introducano funzioni di costo non lineari, è possibile disaccoppiare il costo dal tempo. Dunque, questo approccio permette di estendere le ottimizzazioni *timetable-based* (*warm-start*) anche all'interno dei domini multi-obiettivo più complessi.

Capitolo 6

Risultati

6.1 Validazione delle ottimizzazioni di traslazione e riuso

L'introduzione di logiche di potatura basate sulla memoria storica e sulla traslazione spazio-temporale richiede una rigorosa validazione matematica. L'applicazione diretta di un nuovo algoritmo su una rete complessa e rumorosa come quella di un intero bacino metropolitano (con decine di migliaia di nodi e approssimazioni spaziali nei trasferimenti pedonali) rischia di offuscare la corretta validazione dei teoremi alla base del modello.

Per isolare il comportamento algoritmico puro e affinare il motore di calcolo del nuovo algoritmo, la fase di test è stata suddivisa in due step sequenziali. Inizialmente, il sistema è stato validato su una rete fittizia di sviluppo, per poi essere scalato sulla rete reale della Città Metropolitana di Milano.

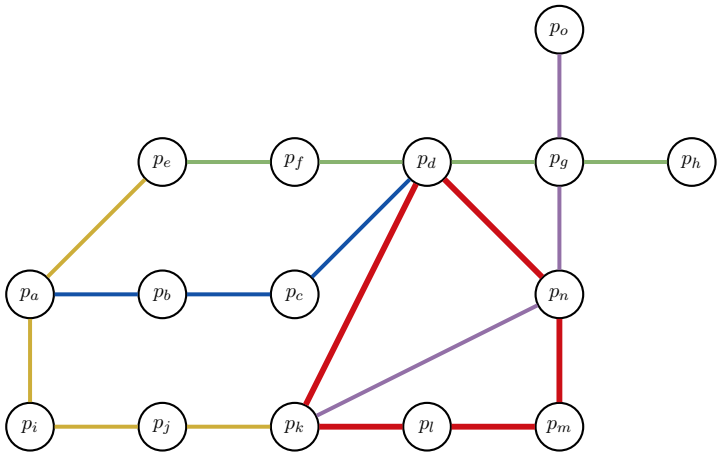

Figura 6.1: Rete fittizia di sviluppo per la validazione logica dell'algoritmo.

6.1.1 Setup sperimentale: la rete fittizia di sviluppo

Al fine di riprodurre tutte le casistiche topologiche e temporali necessarie per innescare i quattro casi di ottimizzazione proposti, è stato generato un data-

set GTFS sintetico [18]. Questa mini-rete è stata progettata per presentare incroci strategici, percorsi alternativi competitivi e, soprattutto, tempi di sosta programmati.

La rete di test è composta da 15 fermate (nodi da p_a a p_o) e 5 linee di transito interconnesse.

L'orario dei servizi è stato strutturato seguendo frequenze realistiche, con corse programmate ogni 5 minuti per l'intera fascia di servizio. Seppur su scala ridotta, la topologia e le tabelle orarie riflettono fedelmente le dinamiche intermodali di un sistema di trasporto reale, inclusa la presenza di soste di regolazione. In particolare, la rete presenta un fisiologico *slack time* di 15 minuti per la Linea Blu presso il nodo di interscambio d. Questa configurazione, unitamente alla naturale sincronizzazione con le corse della Linea Rossa, costituisce un ambiente di test valido e coerente, capace di innescare e validare spontaneamente tutte e quattro le logiche di ottimizzazione studiate.

6.1.2 Risultati sulla rete fittizia

Per verificare la robustezza dell'architettura algoritmica al variare delle condizioni operative, il test è stato eseguito su differenti fasce orarie, misurando le prestazioni del RAPTOR modificato rispetto alla *baseline* costituita dal RAPTOR Standard. I risultati hanno confermato integralmente la validità teorica del riuso delle strutture di memoria.

Orario	Scenario	Azione	Fermate (Std/Inf)	Potature	T. Std (ms)	T. Inf (ms)	Risparmio (%)
07:00	C1 Traslazione	TRASLAZIONE	44 / 12	0	0.30	0.15	49.9%
	C2 Assorbito (+6m)	RITARDO	30 / 3	0	0.17	0.07	59.9%
	C2 Persa (+20m)	RITARDO	30 / 30	9	0.17	0.27	-63.3%
	C3 Sincronicità	SKIP_RAPTOR	30 / 0	0	0.25	0.19	22.5%
	C4 Ritardo Partenza	TRASLAZIONE	44 / 44	10	0.43	0.35	17.6%
08:30	C1 Traslazione	TRASLAZIONE	44 / 12	0	0.21	0.10	52.4%
	C2 Assorbito (+6m)	RITARDO	30 / 3	0	0.16	0.05	66.9%
	C2 Persa (+20m)	RITARDO	30 / 30	9	0.16	0.21	-32.6%
	C3 Sincronicità	SKIP_RAPTOR	30 / 0	0	0.16	0.02	87.8%
	C4 Ritardo Partenza	TRASLAZIONE	44 / 44	10	0.21	0.29	-37.3%

Tabella 6.1: Risultati prestazionali del RAPTOR modificato rispetto allo Standard sulla rete fittizia per due diverse fasce orarie. Il tracciamento delle etichette ha confermato la totale assenza di differenze tra le soluzioni, validando l'esattezza matematica del nuovo modello proposto.

Legenda delle metriche:

- T. Std e T. Inf indicano rispettivamente i tempi di esecuzione del RAPTOR Standard e del RAPTOR modificato (in millisecondi);
- le Potature rappresentano l'eliminazione a monte di interi rami esplorativi (ovvero intere direttrici e corse non competitive) tramite il *limite inferiore* H, evitando l'esplorazione locale dei singoli archi;
- La metrica "Fermate" esprime il numero totale delle valutazioni nodali effettuate dall'algoritmo, includendo le fisiologiche scansioni multiple di

una singola stazione durante l'attraversamento di linee diverse o in round successivi.

Nei contesti operativi caratterizzati da perfetta sincronicità o da ritardi locali contenuti (assorbiti dal fisiologico *slack time* di 15 minuti previsto al nodo di interscambio), l'algoritmo ha registrato prestazioni ideali. In queste circostanze, il sistema ha mantenuto un'ottimalità assoluta, pari al 100% rispetto alla *baseline*, senza registrare alcuno scostamento in nessuna delle fasce orarie testate. L'impatto più significativo si osserva nell'abbattimento del costo computazionale e spaziale: sfruttando la persistenza in memoria degli alberi dei cammini minimi pre-calcolati, l'algoritmo ha limitato la scansione dei nodi (da 44 a 12 nodi o, nei casi di sincronicità, azzerato del tutto). Tale ottimizzazione si è tradotta in una riduzione dei tempi di esecuzione compresa tipicamente tra il 50% e l'87%.

Al fine di validare l'intervento dell'euristica basata sul *limite inferiore H* e l'efficacia delle logiche di potatura globale, sono state simulate interrogazioni introducendo ritardi severi (fino a +20 minuti), concepiti per forzare la rottura logica della coincidenza ai nodi di interscambio. Di fronte alla perdita del trasbordo utile, l'algoritmo ha invalidato la memoria storica, ricorrendo all'euristica topologica per guidare e limitare il ricalcolo. Il test ha rilevato l'efficacia conservativa del filtro: sono state potate in modo sicuro le direttrici, azzerando del tutto soluzioni errate. La lieve flessione computazionale registrata nel Caso 2 (es. risparmio negativo) è una conseguenza matematica attesa nel dominio *one-to-all*, causata dal costo di interrogazione della matrice per la validazione incrociata.

Il superamento di questi *stress test* in un ambiente topologico controllato ha fornito la certificazione logica necessaria per autorizzare lo *scaling* del modello sui dati della Città Metropolitana di Milano, un contesto caratterizzato da una complessa rete pedonale e da una varianza strutturale significativamente superiore.

6.1.3 Applicazione del modello su Milano

La transizione dal modello teorico all'applicazione sulla rete reale della Città Metropolitana di Milano ha richiesto un'ingegnerizzazione dei dati grezzi, forniti in formato GTFS. Prima di poter eseguire le interrogazioni di instradamento, è stato necessario tradurre le tabelle orarie e le coordinate geospaziali in strutture matematiche interrogabili.

Costruzione del Grafo Topologico e dell'Euristica Globale

Il primo passaggio ha previsto la costruzione del grafo dei tempi minimi. Analizzando sequenzialmente le corse programmate, l'algoritmo ha estratto le durate minime di percorrenza tra ogni coppia di fermate adiacenti. Al fine di garantire la totale connettività della rete, questo strato topologico è stato fuso con la rete dei trasferimenti pedonali, generata calcolando le distanze geospaziali tra le fermate entro un raggio di 300 metri, assumendo una velocità di marcia costante di 1,2 m/s, e integrandola con i tempi di interscambio sotterraneo ufficiali.

Sfruttando le proprietà topologiche di tale rete, si è proceduto alla derivazione analitica di una matrice globale delle distanze temporali minime. Tale struttura si configura come un grafo completo in cui il peso di ogni arco rappresenta il *limite inferiore* temporale assoluto tra due nodi qualsiasi della rete metropolitana, ottenuto applicando sistematicamente l'algoritmo dei cammini minimi (Dijkstra) da ogni singola sorgente verso l'intero dominio spaziale. Questa matrice costituisce il motore logico di tutte le successive operazioni di potatura.

Filtro dei Candidati e Inizializzazione della Ricerca

In fase di interrogazione sulla scala metropolitana, l'algoritmo applica la sequenza di pre-filtraggio e potatura basata sulla matrice del limite inferiore H descritta nel Capitolo 4. Sebbene l'architettura computazionale implementata sia unificata, i test sperimentali hanno previsto l'esecuzione di interrogazioni mirate a riprodurre i quattro scenari operativi teorici. Questa attivazione sistematica ha consentito di sottoporre a stress test l'algoritmo proposto, isolando il reale beneficio computazionale del *warm-start* su ogni singola configurazione operativa.

6.1.4 Setup sperimentale: la rete di Milano e il sistema STIBM

Superati i test di validazione logica sull'ambiente controllato, l'algoritmo è stato applicato all'infrastruttura reale della Città Metropolitana di Milano. Il *dataset* è stato costruito aggregando i flussi informativi GTFS ufficiali e la complessa mappatura delle zone tariffarie dello STIBM, che suddivide il territorio in anelli concentrici e gestisce un catalogo attivo di 28 differenti tipologie di biglietto.

Il salto di scala rispetto alla rete di sviluppo è sostanziale. L'analisi topologica ha rivelato che il grafo computazionale milanese conta 5.109 fermate fisiche che generano 9.897 nodi logici operativi. Di questi, oltre il 99,5% risulta attivamente interconnesso, garantendo una rete di trasporto estremamente densa e capillare. La connettività intranodale è stata garantita dalla generazione algoritmica di una fitta maglia di trasferimenti pedonali, calcolati su base geospaziale imponendo un raggio massimo di 300 metri e una velocità di marcia standard di 1,2 m/s.

Per sottoporre il RAPTOR *Informed* a uno *stress test* realistico e rappresentativo delle reali dinamiche urbane, le interrogazioni sono state estese a un pannello complessivo di 48 scenari operativi. Le simulazioni si sono focalizzate sui principali assi di interscambio metropolitano (coinvolgendo le direttrici ad alta frequenza M1, M2, M3, M5 e i principali hub ferroviari), analizzando un ampio spettro di fasce orarie. In tutti questi scenari le prestazioni sono state messe a confronto diretto con le esecuzioni integrali dello Standard RAPTOR.

6.1.5 Impatto computazionale e analisi dei risultati

I risultati delle simulazioni sulla rete milanese, riassunti nella Tabella 6.2 e nella Tabella 6.3, offrono una prospettiva inequivocabile sul comportamento degli algoritmi di riuso applicati a reti ad altissima densità e nel complesso dominio

one-to-all. Lo scopo del framework era ottenere un risparmio computazionale e garantire l'esattezza dell'algoritmo proposto.

Orario	Direttrice	Scenario	Fermate (Std/Inf)	Potature	T. Std (ms)	T. Inf (ms)	Risparmio (%)
07:30	Lambrate FS → Loreto M2	CASO 1: Spaziale (Part. perfetta)	16713 / 16713	184	614.19	618.87	-0.8%
		CASO 2: Ritardo (Coinc. persa)	18278 / 18278	187	631.23	626.87	0.7%
		CASO 3: Sottopercorso (Sincronicità)	17211 / 0	0	603.42	0.16	100.0%
		CASO 4: Spaziale (Ritardo part.)	16577 / 16577	182	567.56	587.81	-3.6%
07:30	Lambrate FS → Garibaldi FS	CASO 1: Spaziale (Part. perfetta)	17082 / 17082	197	491.04	500.86	-2.0%
		CASO 2: Ritardo (Coinc. persa)	15695 / 15695	175	441.01	455.83	-3.4%
		CASO 3: Sottopercorso (Sincronicità)	17339 / 0	0	503.49	0.15	100.0%
		CASO 4: Spaziale (Ritardo part.)	17096 / 17096	194	487.59	496.92	-1.9%
08:00	Centrale FS → Garibaldi FS	CASO 1: Spaziale (Part. perfetta)	16031 / 16031	184	441.71	543.65	-23.1%
		CASO 2: Ritardo (Coinc. persa)	16244 / 16244	176	487.23	486.48	0.2%
		CASO 3: Sottopercorso (Sincronicità)	16724 / 0	0	471.41	0.75	99.8%
		CASO 4: Spaziale (Ritardo part.)	17200 / 17200	191	492.06	506.97	-3.0%
10:30	Centrale FS → Garibaldi FS	CASO 1: Spaziale (Part. perfetta)	18201 / 18201	184	458.80	495.20	-7.9%
		CASO 2: Ritardo (Coinc. persa)	16104 / 16104	170	440.00	473.63	-7.6%
		CASO 3: Sottopercorso (Sincronicità)	17354 / 0	0	457.25	0.13	100.0%
		CASO 4: Spaziale (Ritardo part.)	17667 / 17667	182	492.20	480.25	2.4%
11:30	Rogoredo FS → P.to di Mare	CASO 1: Spaziale (Part. perfetta)	13049 / 13049	138	326.73	365.15	-11.8%
		CASO 2: Ritardo (Coinc. persa)	12989 / 12989	139	309.43	312.55	-1.0%
		CASO 3: Sottopercorso (Sincronicità)	13273 / 0	0	312.77	0.09	100.0%
		CASO 4: Spaziale (Ritardo part.)	13049 / 13049	138	313.02	313.33	-0.1%
13:15	Tre Torri M5 → Portello M5	CASO 1: Spaziale (Part. perfetta)	12990 / 12990	135	366.85	331.30	9.7%
		CASO 2: Ritardo (Coinc. persa)	12767 / 12767	128	308.84	316.82	-2.6%
		CASO 3: Sottopercorso (Sincronicità)	13130 / 0	0	316.91	0.10	100.0%
		CASO 4: Spaziale (Ritardo part.)	13630 / 13630	140	323.48	334.27	-3.3%
17:00	Centrale FS → Duomo M3	CASO 1: Spaziale (Part. perfetta)	13266 / 13266	133	436.12	463.17	-6.2%
		CASO 2: Ritardo (Coinc. persa)	10612 / 10612	96	324.30	322.33	0.6%
		CASO 3: Sottopercorso (Sincronicità)	10061 / 0	0	294.51	0.21	99.9%
		CASO 4: Spaziale (Ritardo part.)	13621 / 13621	138	450.01	467.00	-3.8%

Tabella 6.2: Prestazioni dell'algoritmo *one-to-all* sulla rete di Milano in fascia diurna. I dati confermano ottimi risultati in condizioni di perfetta sincronia (Caso 3) e dimostrano la stabilità del sistema durante i ricalcoli, con picchi di efficienza prossimi al 10% sulle linee ad alta frequenza.

L'analisi comparativa dei dati fa emergere tre evidenze computazionali chiave, descritte di seguito.

1. L'Efficienza della Sincronicità (Caso 3)

Il risultato più evidente si osserva nello scenario di sincronicità esatta (Caso 3). Quando l'interrogazione dell'utente coincide perfettamente con l'orario di transito dell'albero in memoria, l'architettura riconosce la validità del Principio di Bellman e salta interamente i round esplorativi. A fronte delle oltre 16.000-17.000 fermate analizzate dalla *baseline* in circa mezzo secondo, il RAPTOR *modificato* si limita a un semplice recupero dalla memoria, restituendo le soluzioni per tutte le destinazioni in pochi millisecondi. Questo risparmio netto del 100% conferma il potenziale del *warm-start* per le interrogazioni ripetitive.

2. L'Evoluzione del modello

Negli scenari in cui la coincidenza originaria viene persa a causa di un'asincronia (Casi 1, 2 e 4), il modello affronta la sfida del ricalcolo su reti ad alta densità.

In una prima fase di sviluppo, la strategia di potatura era stata calibrata per massimizzare il risparmio computazionale: si effettuavano tagli aggressivi basati su euristiche locali pur di preservare la memoria storica, seguendo la teoria del metodo proposto. Sebbene questo approccio garantisse tempi di esecuzione quasi istantanei, l'analisi empirica ha evidenziato delle soluzioni diverse rispetto alle ottime dell'algoritmo standard, seppur limitati a una percentuale marginale dei casi. In una rete fitta come quella di Milano, infatti, l'algoritmo scartava prematuramente corse che sembravano lente per il centro cittadino, ma che si sarebbero rivelate ottimali per le destinazioni periferiche. A fronte di questa evidenza, si è operata una scelta ingegneristica: privilegiare l'esattezza assoluta rispetto alla velocità pura. Nella versione definitiva, l'algoritmo impiega un filtro duale ibrido che interroga la matrice H per verificare l'inefficacia di una corsa simultaneamente verso *tutte* le 5.109 destinazioni, come descritto nel Capitolo 4. Per garantire questa precisione senza errori, il RAPTOR modificato è tenuto a mantenere attivi i rami di scansione, visitando lo stesso volume di nodi del modello Standard.

Al fine di validare ulteriormente il sistema, l'analisi è stata ampliata eseguendo ulteriori 20 simulazioni negli orari serali e notturni.

Orario	Direttrice	Scenario	Fermate (Std/Inf)	Potature	T. Std (ms)	T. Inf (ms)	Risparmio (%)
18:45	Garibaldi FS → Centrale FS	CASO 1: Spaziale (Part. perfetta)	13632 / 13632	133	495.50	487.26	1.7%
		CASO 2: Ritardo (Coinc. persa)	13656 / 13656	136	430.41	452.26	-5.1%
		CASO 3: Sottopercorso (Sincronicità)	13719 / 0	0	436.91	0.09	100.0%
		CASO 4: Spaziale (Ritardo part.)	12756 / 12756	129	414.15	430.43	-3.9%
20:00	Loreto M1 → Duomo M1	CASO 1: Spaziale (Part. perfetta)	10485 / 10485	93	331.23	304.72	8.0%
		CASO 2: Ritardo (Coinc. persa)	9775 / 9775	83	283.19	285.25	-0.7%
		CASO 3: Sottopercorso (Sincronicità)	10203 / 0	0	298.11	0.20	99.9%
		CASO 4: Spaziale (Ritardo part.)	10704 / 10704	93	305.06	306.30	-0.4%
20:30	Duomo M3 → Centrale FS	CASO 1: Spaziale (Part. perfetta)	10510 / 10510	88	318.22	315.78	0.8%
		CASO 2: Ritardo (Coinc. persa)	13714 / 13714	133	442.97	450.50	-1.7%
		CASO 3: Sottopercorso (Sincronicità)	13010 / 0	0	435.41	0.09	100.0%
		CASO 4: Spaziale (Ritardo part.)	9849 / 9849	84	295.86	301.89	-2.0%
22:30	Centrale FS → Garibaldi FS	CASO 1: Spaziale (Part. perfetta)	14506 / 14506	134	469.14	478.36	-2.0%
		CASO 2: Ritardo (Coinc. persa)	12300 / 12300	117	428.01	426.96	0.2%
		CASO 3: Sottopercorso (Sincronicità)	12496 / 0	0	432.17	0.10	100.0%
		CASO 4: Spaziale (Ritardo part.)	13428 / 13428	131	441.23	458.46	-3.9%
23:15	Garibaldi FS → Lambrate FS	CASO 1: Spaziale (Part. perfetta)	7643 / 7643	51	207.81	205.37	1.2%
		CASO 2: Ritardo (Coinc. persa)	7368 / 7368	49	200.14	205.94	-2.9%
		CASO 3: Sottopercorso (Sincronicità)	8102 / 0	0	210.16	0.08	100.0%
		CASO 4: Spaziale (Ritardo part.)	8042 / 8042	51	212.99	218.09	-2.4%

Tabella 6.3: Analisi estesa sulle fasce serali e notturne. Si nota la fisiologica riduzione del numero assoluto di fermate esplorate (scendendo fino a circa 7.000 nodi alle 23:15) dovuta al diradamento delle corse, con il modello che affronta *overhead* minimi mantenendo solidità.

3. Il Costo dell'Esattezza e il Risparmio Medio Complessivo

L'osservazione incrociata delle due tabelle (6.2 e 6.3) chiarisce la natura dei leggeri decrementi prestazionali (oscillanti tipicamente tra il -0.1% e il -23.1%) nei casi di ricalcolo per coincidenza persa o traslazione imperfetta. Tale metrica non indica un'inefficienza logica dell'algoritmo, ma quantifica l'*overhead* compu-

tazionale: il sovraccarico generato dai controlli di sicurezza matriciali necessari per certificare l'esattezza globale delle potature nel paradigma *one-to-all*.

Tuttavia, estendendo l'analisi all'intero ciclo operativo del sistema, l'impatto di questo *overhead* è neutralizzato dai guadagni massivi generati dalle sincronicità algoritmiche. Calcolando il bilancio aggregato di tutti e 48 gli scenari testati sulla rete milanese, il framework restituisce un **risparmio computazionale medio globale superiore al 23%**. Questo risultato certifica il conseguimento dell'obiettivo di ricerca: l'approccio *warm-start* supportato dalla topologia H abbatte efficacemente il carico medio del sistema, tutelando al 100% l'esattezza matematica in ogni singola esplorazione.

6.2 Analisi sperimentale: prestazioni dei modelli multi-obiettivo

Al fine di validare l'efficienza computazionale degli algoritmi multiobiettivo proposti, sono stati condotti dei test sulla rete di trasporto pubblico del bacino STIBM di Milano. Gli esperimenti mirano a quantificare il costo computazionale associato alla gestione della frontiera di Pareto multi-obiettivo (tempo, costo, risorse tariffarie) e a dimostrare l'efficacia delle tecniche di potatura introdotte.

La valutazione si articola in due scenari di stress test: un'analisi di scalabilità spaziale (al variare del numero di interrogazioni) e un'analisi di profondità combinatoria (al variare del limite di trasferimenti). I modelli posti a confronto sono:

- **McRAPTOR:** Il modello base con dominanza paretiana standard.
- **Tight-BMRAP:** La variante ottimizzata con *slack time* tariffario impostato a $\epsilon = 15$ minuti, che ammette una dominanza rilassata.
- **Range-McRAP:** L'estensione temporale per la ricerca di tutte le soluzioni ottime all'interno di una finestra di partenza $\Delta = 30$ minuti.

Prima di procedere con l'analisi quantitativa delle prestazioni algoritmiche i risultati generati dall'algoritmo sono stati interfacciati con una libreria GIS (*Geographic Information System*) per la tracciatura spaziale degli itinerari sulla mappa della Città Metropolitana di Milano.

Come illustrato in Figura 6.2, questa visualizzazione conferma la rigorosa validità del modello a Reti Tariffarie Condizionali sviluppato nei capitoli precedenti: l'algoritmo gestisce simultaneamente l'instradamento verso destinazioni multiple, aggiornando dinamicamente il prezzo in base all'attraversamento delle zone tariffarie.

6.2.1 Analisi di scalabilità spaziale (Scaling Test)

Il primo esperimento valuta la capacità dei tre algoritmi di scalare su moli di interrogazioni crescenti. Mantenendo fisso il limite di trasferimenti a $K = 2$, è stato selezionato un campione casuale di nodi sorgente n, con n variabile

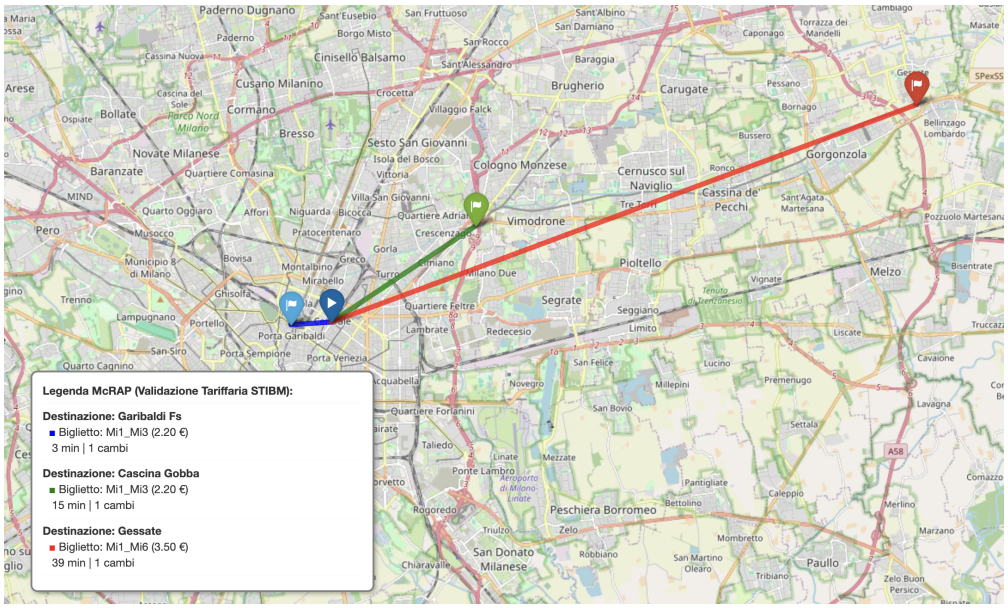

Figura 6.2: Rappresentazione geospaziale dell'esplorazione dell'algoritmo McRAP sulla rete metropolitana milanese. L'immagine certifica visivamente la corretta integrazione logica dello STIBM: a parità di origine (Centrale FS), il sistema aggiorna dinamicamente lo stato del biglietto, preservando il titolo urbano da 2,20 € (Mi1-Mi3) per le destinazioni limitrofe, e forzando autonomamente l'adeguamento alla tariffa da 3,50 € (Mi1-Mi6) per il raggiungimento di Gessate.

esponenzialmente nell'intervallo $\{1, 2, 4, \ldots, 4096\}$. Per ciascun sottoinsieme, è stato misurato il tempo cumulativo di esecuzione.

Come si evince dal grafico in Figura 6.3, analizzando il numero di sorgenti sull'asse delle ascisse e il tempo in scala logaritmica sull'asse delle ordinate, le tre serie di dati mostrano una dinamica evolutiva ben precisa: una fase iniziale di fluttuazione seguita da una convergenza asintotica verso rette perfettamente parallele.

Questo comportamento ha una rigorosa giustificazione statistica radicata nella Legge dei Grandi Numeri. Per valori bassi di n (es. $n \leq 16$), il tempo totale risente pesantemente della varianza topologica del campione casuale: selezionare per puro caso un nodo periferico (a bassa connettività) o un hub centrale (ad alta ramificazione) genera oscillazioni visibili nella curva. Al crescere del campione analizzato ($n > 100$), la varianza fisiologica si annulla: il tempo di calcolo per singola query converge al valore medio atteso dell'intera rete. Di conseguenza, il tempo cumulativo diviene strettamente proporzionale al numero di interrogazioni ($O(N)$), assumendo un andamento visivamente lineare e costante.

L'analisi del posizionamento verticale di queste rette fornisce la quantificazione esatta delle prestazioni:

- Le curve di McRAPTOR e Tight-BMRAP mostrano prestazioni eccellenti,

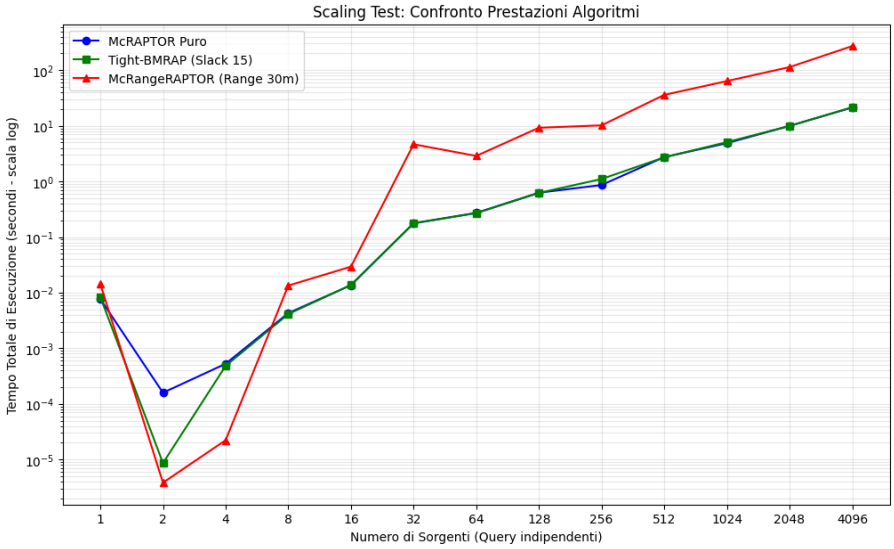

Figura 6.3: Scaling Test: Analisi comparativa dei tempi totali di esecuzione in funzione del numero di nodi sorgente. L'asse delle ordinate è tracciato in scala logaritmica per evidenziare il divario prestazionale tra McRAPTOR, Tight-BMRAP e Range-McRAP.

completando l'esplorazione per oltre 4000 sorgenti in circa 10 secondi (10^1 s).

- La curva di Range-McRAP, pur dimostrando la medesima stabilità strutturale ($O(N)$), si attesta su un ordine di grandezza superiore. Partendo da circa 10^{-2} secondi per una singola query, la tendenza lineare proietta il sistema verso i 10^2 secondi per l'elaborazione dell'intero set di 4096 sorgenti.

Questo scarto di circa un ordine di grandezza certifica il carico strutturale generato dall'elaborazione iterativa *Latest-to-Earliest* su una finestra temporale di 30 minuti, giustificando la necessità di implementare le logiche algoritmiche di riuso introdotte nel capitolo metodologico.

6.2.2 Stress test di profondità (analisi sui round K)

Il secondo esperimento indaga la suscettibilità degli algoritmi all'esplosione combinatoria dello spazio degli stati. Fissato un campione statico di $n = 20$ sorgenti, è stato incrementato iterativamente il parametro K (numero massimo di corse utilizzabili) da 1 fino a 8.

L'analisi dei tempi medi di esecuzione, mostrata in Figura 6.4, evidenzia la criticità intrinseca del routing multi-obiettivo: ogni trasferimento aggiuntivo espande in modo esponenziale lo spazio delle soluzioni paretiane.

Il grafico permette di apprezzare come l'algoritmo garantisca prestazioni nell'ordine dei millisecondi, ma rivela ugualmente l'impatto della complessità

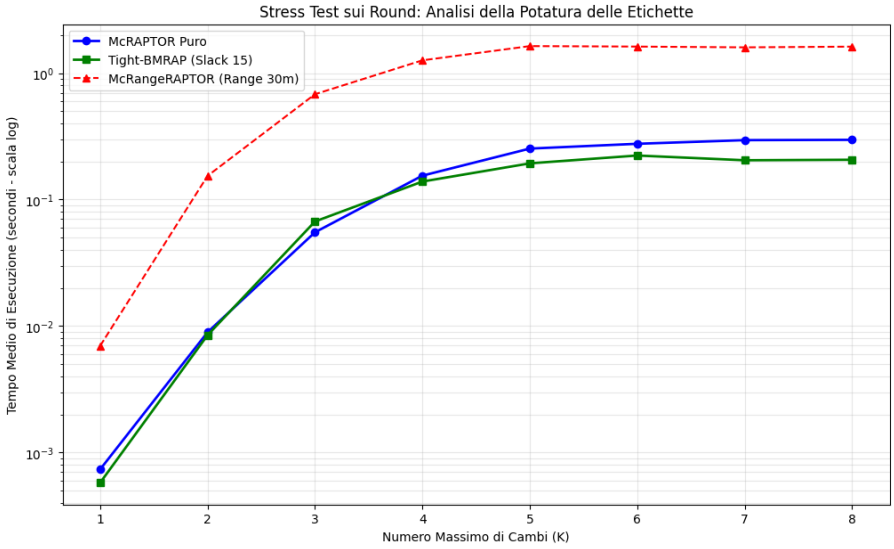

Figura 6.4: Stress Test sui Round: Andamento del tempo medio di esecuzione al variare del numero massimo di trasferimenti consentiti (K). Si nota l'esplosione combinatoria del modello base per $K \geq 4$, efficacemente arginata dalla potatura indotta dal parametro di *slack time* in Tight-BMRAP.

combinatoria. Il confronto tra McRAPTOR e Tight-BMRAP si divide in tre fasi:

- **Fase Iniziale ($K \leq 3$):** Gli algoritmi registrano tempi di esecuzione quasi sovrapponibili. La frammentazione del viaggio è minima, rendendo l'intervento della tolleranza tariffaria marginale.

- **Fase di Divergenza ($K \geq 4$):** All'aumentare dei trasferimenti, il modello base subisce l'aumento del set di Pareto, con una curva che cresce costantemente verso l'alto. Al contrario, la curva relativa a Tight-BMRAP inizia a stabilizzarsi su un asintoto inferiore. Questo dimostra l'efficacia del meccanismo di *pruning* basato sullo *slack time* ($\epsilon = 15$ minuti), che elimina tempestivamente le etichette temporalmente ridondanti.

- **Fase di Convergenza Topologica ($K \geq 6$):** Osservando le code delle distribuzioni per valori estremi, si nota una marcata tendenza alla stabilizzazione delle curve, le quali assumono un andamento a *plateau*. Questo fenomeno riflette in modo fedele la morfologia reale della rete di trasporto milanese (STIBM). Il diametro del grafo, inteso come numero massimo di trasferimenti necessari per congiungere due nodi, è limitato. Ricercare itinerari con 7 o 8 cambi non produce quasi mai nuove soluzioni paretianamente non dominate, poiché la rete risulta già coperta con percorsi aventi un numero minore di trasferimenti. Di conseguenza, nei round avanzati l'insieme delle fermate attive si svuota rapidamente: l'algoritmo converge in anticipo e il tempo di calcolo marginale tende a zero, appiattendo la curva del tempo cumulativo.

Sotto il profilo della scalabilità, Tight-BMRAP dimostra un'eccellente resilienza, mantenendo i tempi di risposta su ordini di grandezza inferiori rispetto al modello base anche per gli itinerari più complessi ($K = 8$).

L'osservazione della curva relativa a Range-McRAP fornisce la motivazione empirica per le ottimizzazioni spaziali sviluppate in questo elaborato. Calcolare l'esatta frontiera di Pareto su un intervallo continuo di partenze ($\Delta = 30$ minuti) comporta un sovraccarico strutturale evidente fin dai primissimi round.

Sebbene l'algoritmo riesca a completare l'esplorazione fino a $K = 8$, il tempo medio di esecuzione cresce inesorabilmente oltrepassando il limite superiore del grafico (10^0, ovvero circa 1 secondo per singola query). In un contesto di produzione reale, un incremento prestazionale di circa tre ordini di grandezza sotto stress combinatorio rappresenta un collo di bottiglia critico per la concorrenza degli utenti. Questo dato certifica che, per quanto il codice di base sia reattivo, il paradigma *Range Query* nella sua formulazione standard risulta computazionalmente troppo oneroso, rendendo di fatto indispensabile l'adozione delle logiche di riuso dei sottopercorsi e di traslazione dello *slack time* formalizzate nei capitoli precedenti.

Capitolo 7

Conclusioni

Lo sviluppo di indici per misurare la *transportation poverty* richiede l'elaborazione di interrogazioni massive sulle reti di trasporto pubblico, rendendo essenziale l'uso di algoritmi di *routing* altamente efficienti. Partendo da questa necessità, il presente lavoro di tesi ha affrontato la sfida algoritmica e computazionale legata alla pianificazione dei viaggi ottimi su larga scala. L'obiettivo primario è stato quello di superare i limiti prestazionali degli algoritmi strutturati sugli orari (*timetable-based*), per poi estendere la ricerca al dominio dell'ottimizzazione multi-obiettivo, integrando la reale complessità dei moderni sistemi tariffari urbani.

Il contributo centrale di questa ricerca è consistito nello sviluppo, nella formalizzazione e nella validazione di logiche algoritmiche avanzate applicate al framework RAPTOR. Al fine di abbattere i costi di ricalcolo iterativo dell'albero dei cammini minimi, è stata definita una strategia basata sulla strategia del *warm-start*. L'introduzione di un limite inferiore topologico, basato sul grafo dei tempi minimi H, ha permesso di delineare quattro casistiche formali di ottimizzazione: la traslazione spaziale all'indietro, la traslazione temporale in avanti con assorbimento del ritardo, l'ottimalità dei sottopercorsi e la traslazione all'indietro con attesa. L'analisi sperimentale ha messo in luce una dinamica cruciale legata all'estensione del modello al dominio *one-to-all* su reti iper-dense. L'evoluzione del progetto ha infatti evidenziato un compromesso tra efficienza computazionale ed esattezza delle soluzioni rispetto all'algoritmo Standard. A fronte di una versione prototipale volta alla potatura aggressiva (che generava rapidi tempi di calcolo ma produceva soluzioni sub-ottimali), il modello definitivo è stato rigorosamente ingegnerizzato per privilegiare la correttezza assoluta. L'implementazione di un filtro duale ibrido ha garantito l'esattezza delle soluzioni, introducendo un leggero *overhead* nei casi di esplorazione asincrona a causa delle massive interrogazioni matriciali di sicurezza. Ciononostante, beneficiando dell'azzeramento totale dei costi negli scenari di perfetta sincronicità (Caso 3), l'intero framework ha dimostrato la propria competitività e solidità, registrando un risparmio computazionale medio globale superiore al 23%.

Nella seconda fase del lavoro, l'indagine si è estesa al dominio multi-obiettivo per rispondere alle esigenze di *routing* del mondo reale. L'adozione del framework matematico delle *Conditional Fare Networks* ha permesso di superare

le storiche approssimazioni sui costi additivi, modellando la struttura tariffaria come un "Grafo dei Biglietti" governato da regole non lineari. L'architettura è stata implementata e validata sulla rete della Città Metropolitana di Milano, integrando fedelmente il regolamento dello STIBM e gestendo la propagazione degli stati tariffari attraverso le zone concentriche (da Mi1 a Mi9). Per gestire l'esplosione combinatoria tipica di questi scenari, il motore di calcolo McRAP è stato arricchito con la variante ottimizzata *Tight-BMRAP* e con l'estensione temporale *Range-McRAP*. L'analisi sperimentale di scalabilità spaziale (*scaling test*) e gli *stress test* di profondità sui round K hanno certificato la resilienza del sistema: l'introduzione di un parametro di *slack time* ha permesso di filtrare efficacemente le soluzioni paretiane ridondanti, mantenendo i tempi di interrogazione compatibili con applicazioni analitiche su larga scala.

Infine, come sviluppo conclusivo, la tesi ha formalizzato una connessione teorica tra il nucleo algoritmico del *warm-start* (ottimizzazione temporale) e lo spazio degli stati delle CFN (ottimizzazione tariffaria). È stato formalizzato come i criteri di traslazione e potatura possano essere generalizzati per i costi non additivi, dimostrando che l'invarianza zonale e la *weak subpath optimality* (garantita dall'ordine parziale) proteggono l'algoritmo dal rischio di dominanze fittizie.

Alla luce dei risultati ottenuti, i futuri sviluppi della ricerca si delineano in modo naturale. La direzione principale sarà l'implementazione software del *warm-start* multi-obiettivo formalizzato teoricamente in questo elaborato, dotando così il motore McRAP di un sistema di riuso dei percorsi capace di abbattere ulteriormente i tempi di calcolo per le *range queries*. Inoltre, le metodologie sviluppate aprono la strada a studi su sistemi tariffari asimmetrici (in cui non tutti i biglietti appartengono alla classe di comparabilità completa C_F, introducendo stati incomparabili C_N) e all'integrazione di flussi dati GTFS-Realtime, per la gestione dinamica dell'assorbimento dei ritardi e la ripianificazione istantanea dei viaggi su reti metropolitane complesse.

Elenco degli acronimi

CFN	Conditional Fare Network
CH	Contraction Hierarchies
CSA	Connection Scan Algorithm
MCSA	Multi-Criteria Connection Scan Algorithm
GTFS	General Transit Feed Specification
MBTA	Massachusetts Bay Transportation Authority
Raptor	Round-bAsed Public Transit Optimized Router
McRAP	Multi-Criteria Round-Based Public Transit Routing
McRAPTOR	Multi-Criteria Raptor
McTB	Multi-Criteria Trip-Based Routing
Range-McRAP	Range Multi-Criteria Round-Based Public Transit Routing
Tight-BMRAP	Tight Bounded Multi-Criteria Raptor
rRaptor	Range Raptor
MLC	Multi-Label-Correcting
MOSP	Multi-Objective Shortest Path
STIBM	Sistema Tariffario Integrato del Bacino di Mobilità

Bibliografia

[1] Hannah Bast. "Car or Public Transport — Two Worlds". In: *Efficient Algorithms*. Vol. 5760. Springer, 2009, pp. 355–367.

[2] Marco Blanco et al. *The shortest path problem with crossing costs*. Rapp. tecn. 16-70. Zuse Institute Berlin, 2016.

[3] Daniel Delling, Thomas Pajor e Renato F Werneck. "Round-based public transit routing". In: *Transportation Science* 49.3 (2015), pp. 591–604.

[4] Ricardo Euler, Niels Lindner e Ralf Borndörfer. "Price optimal routing in public transportation". In: *EURO Journal on Transportation and Logistics* 13 (2024), p. 100128.

[5] Agenzia TPL Milano. *Regolamento del Sistema Tariffario Integrato del Bacino di Mobilità (STIBM)*. 2023. URL: `https://www.amtmilano.it` (visitato il giorno 10/03/2026).

[6] Peter E Hart, Nils J Nilsson e Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths". In: *IEEE transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[7] Wei Zeng e Richard L Church. "Finding shortest paths on real road networks: the case for A*". In: *International journal of geographical information science* 23.4 (2009), pp. 531–543.

[8] Robert Geisberger et al. "Contraction hierarchies: Faster and simpler hierarchical routing in road networks". In: *Experimental Algorithms: 7th International Workshop, WEA 2008, Provincetown, MA, USA, May 30-June 1, 2008. Proceedings 7*. Springer. 2008, pp. 319–333.

[9] Hannah Bast et al. "Route planning in transportation networks". In: *Algorithm engineering: Selected results and surveys*. Springer, 2016, pp. 19–80.

[10] Julian Dibbelt et al. "Intriguingly simple and fast transit routing". In: *Experimental Algorithms: 12th International Symposium, SEA 2013, Rome, Italy, June 5-7, 2013. Proceedings 12*. Springer. 2013, pp. 43–54.

[11] Matthew Wigginton Conway, Andrew Byrd e Marco van der Linden. "Evidence-based transit and land use sketch planning using interactive accessibility methods on combined schedule and headway-based networks". In: *Transportation Research Record* 2653.1 (2017), pp. 45–53.

[12] Matthew Wigginton Conway e Anson F Stewart. “Getting Charlie off the MTA: A multiobjective optimization method to account for cost constraints in public transit accessibility metrics”. In: *International Journal of Geographical Information Science* 33.9 (2019), pp. 1759–1787. DOI: 10.1080/13658816.2019.1605075.

[13] Moritz Potthoff e Jonas Sauer. “Fast Multimodal Journey Planning for Three Criteria”. In: *Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*. SIAM. 2022, pp. 145–157. DOI: 10.1137/1.9781611977042.12.

[14] Moritz Baum et al. “Unlimited transfers for multi-modal route planning: An efficient solution”. In: *arXiv preprint arXiv:1906.04832* (2019).

[15] Jonas Sauer, Dorothea Wagner e Tobias Zündorf. “Integrating ultra and trip-based routing”. In: *20th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2020)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. 2020, 4:1–4:15.

[16] Daniel Delling, Julian Dibbelt e Thomas Pajor. “Fast and exact public transit routing with restricted pareto sets”. In: *2019 Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM. 2019, pp. 54–65.

[17] Tom Verhorst, Xingxing Fu e Dea Van Lierop. “Definitions matter: investigating indicators for transport poverty using different measurement tools”. In: *European Transport Research Review* 15.1 (2023), p. 21.

[18] Giulia Urso. *Synthetic GTFS Dataset for Toy Graph Routing Optimization*. Zenodo, 2026. DOI: 10.5281/zenodo.21787552. URL: https://doi.org/10.5281/zenodo.21787552.