



UNIVERSITÀ DI PISA

UNIVERSITÀ DI PISA

DIPARTIMENTO DI MATEMATICA

Laurea Triennale in Matematica

Vehicle Scheduling in Contesti Multi-Operatore:

**Un Modello di Ottimizzazione Lessicografica
per la Ripartizione del Lavoro**

Relatore:

Prof: Antonio Frangioni

Candidata:

Chiara Bertani

ANNO ACCADEMICO 2025/2026

Abstract

Questo lavoro di tesi, realizzato in collaborazione con Hitachi Rail (già M.A.I.O.R. S.r.l.), affronta il *Vehicle Scheduling Problem* nel trasporto pubblico locale, in cui il servizio e le relative corse devono essere suddivisi tra più operatori. L'obiettivo principale del lavoro è la progettazione e l'implementazione di un modello algoritmico capace di ripartire equamente le risorse operative – misurabili in turni macchina, chilometri in linea o trasferimenti a vuoto – garantendo al contempo la minimizzazione dei costi di esercizio.

Il lavoro adotta un approccio *ex post*: a partire da un sottoinsieme promettente di turni ammissibili generato preventivamente dagli algoritmi di MAIOR senza considerare i vincoli di ripartizione, il problema viene affrontato mediante un modello di ottimizzazione basato su Programmazione Lineare Mista (MIP). L'intera procedura di analisi dei dati e di formulazione è stata sviluppata in linguaggio Julia, avvalendosi del framework JuMP e di solutori *general-purpose* sia open-source (HiGHS) che commerciali (Gurobi). La struttura matematica si basa sul *Set Partitioning* ed implementa un approccio lessicografico a due fasi: la prima fase calcola lo scarto strutturale minimo per definire i limiti di bilanciamento dell'infrastruttura, mentre la seconda minimizza i costi operativi e la flotta entro una soglia di tolleranza predefinita.

Questo impianto metodologico offre il vantaggio di essere potenzialmente integrabile all'interno della suite di ottimizzazione aziendale. Il modello è stato testato su scenari reali di complessità crescente, dimostrando la validità della formulazione e la sua efficacia come strumento decisionale.

Indice

Abstract	1
1 Introduzione e Scopo della Tesi	1
1.1 Il Vehicle Scheduling Problem nel Trasporto Pubblico Locale	1
1.2 Stato dell'Arte e Formulazioni Classiche	1
1.3 Il Contesto Industriale e il Problema del Bilanciamento Multi-Operatore . .	2
1.4 Struttura dell'elaborato	2
2 Contesto Aziendale e Modellizzazione Matematica	4
2.1 Richiami teorici: Programmazione Lineare Intera e Metodi Risolutivi	4
2.1.1 Il Rilassamento Lineare e l'Algoritmo del Simplex	4
2.1.2 L'Esplorazione dello Spazio: Branch & Bound	5
2.1.3 L'applicazione al Set Partitioning	5
2.2 Formulazione Matematica del Modello	6
2.2.1 Approccio Lessicografico a Due Fasi	7
3 Elaborazione dei Dati e Implementazione in Julia	9
3.1 Analisi della struttura dei file operativi MAIOR	9
3.2 Strategie di pre-processing e gestione dei dati in Julia	10
3.3 Definizione delle metriche di valutazione	11
3.4 Costruzione del modello matematico con JuMP e HiGHS	12
3.4.1 Implementazione dell'Approccio Lessicografico a Due Fasi e Tolleranza Dinamica	13
4 Caso di Studio A: Modello a 2 Depositi	15
4.1 Descrizione dello scenario operativo	15
4.2 Analisi computazionale sulle metriche di bilanciamento	15
4.2.1 Risultati sulla metrica Number of Blocks	15
4.2.2 Risultati sulla metrica Chilometri in Linea	17
4.2.3 Risultati sulla metrica Chilometri a Vuoto	17

4.2.4	Risultati sul Chilometraggio Totale	18
4.3	Benchmark Comparativo: HiGHS vs. Gurobi e Analisi Costo-Efficacia . . .	18
4.3.1	Parametrizzazione Algoritmica a Confronto	19
4.3.2	Valutazione Industriale e Prospettive su Larga Scala	19
5	Il Modello Esteso: Caso Studio a 3 Depositi e Gestione della Comples-	21
	sità	
5.1	Il Contesto Commerciale e i Requisiti MAIOR	21
5.1.1	Generalizzazione del Modello per il Bilanciamento Multi-Metrica . .	22
5.1.2	Strategie di Pre-processing e Tuning del Solutore HiGHS	24
5.1.3	Il «Pool Esteso» e la Baseline di Riferimento	25
5.2	Trattabilità Computazionale e Analisi di Sensitività del Parametro ϵ	25
5.2.1	La Soluzione: Regularizzazione e Boost Euristico	25
5.2.2	Analisi di Sensitività Computazionale	26
5.2.3	Discussione dei Risultati e Individuazione dello <i>Sweet Spot</i>	26
5.3	Ottimizzazione a Singola Metrica: La Scoperta della «Rigidità di Rete» . .	27
5.3.1	Risultati Computazionali	27
5.3.2	Analisi e Discussione: La Rigidità Topologica	27
5.4	Ottimizzazione Simultanea Multi-Obiettivo: Scalatura e Trade-Off	28
5.4.1	Lo Scenario a Tolleranza Elastica ($\delta = 2\%$)	29
5.4.2	Lo Scenario a Tolleranza Rigida e la Robustezza della Rete	29
5.4.3	Un'Analisi di Stress con Tolleranza Estrema e Saturazione Temporale	29
5.4.4	Considerazioni Computazionali e Tabella Sinottica	30
5.5	Validazione del Modello: Stress Test e Limiti della Column Generation . . .	31
5.5.1	Fallimento sul Bacino Originale (Infeasible)	31
5.5.2	Successo sul Bacino Esteso «Ad Hoc» e Validazione Finale	31
5.5.3	Discussione Critica e Diagnosi dei Limiti dell'Approccio <i>Ex Post</i> . .	32
5.6	Benchmark Industriale su Larga Scala: HiGHS vs. Gurobi	33
5.6.1	Considerazioni Operative e la Viabilità Pratica dell'Open-Source . .	33
6	Conclusioni e Sviluppi Futuri	35
6.1	Sintesi del Lavoro Svolto	35
6.2	Risultati Conseguiti	35
6.3	Sviluppi Futuri	36

Introduzione e Scopo della Tesi

1.1 Il Vehicle Scheduling Problem nel Trasporto Pubblico Locale

Nel panorama del Trasporto Pubblico Locale (TPL), la qualità del servizio offerto ai passeggeri deve misurarsi quotidianamente con i limiti delle risorse aziendali. Il nucleo algoritmico di questo processo è il *Vehicle Scheduling Problem* (VSP), che consiste nel tradurre una griglia oraria in un'assegnazione operativa concreta. In termini formali, il compito del VSP è raggruppare le singole corse (*journeys*) in turni di lavoro continui (*vehicle blocks*) da assegnare a una flotta reale di autobus.

La costruzione dei turni richiede il rispetto di stringenti vincoli spaziali e operativi, tra cui la gestione dei tempi di trasferimento a vuoto (*deadhead*) per riposizionare i mezzi tra capolinea distanti e il rispetto delle capacità fisiche dei depositi aziendali.

1.2 Stato dell'Arte e Formulazioni Classiche

Il problema, nella sua estensione multi-deposito (*Multi-Depot Vehicle Scheduling Problem*, MDVSP), appartiene alla classe dei problemi *NP-hard* ed è ampiamente trattato nella letteratura di Ricerca Operativa. Storicamente, i metodi risolutivi si dividono in due grandi famiglie algebriche:

- **Modelli a Flusso su Rete Multi-Commodity:** in cui ogni deposito rappresenta una merce distinta all'interno di una rete di connessioni (cfr. Ribeiro e Soumis, 1994; Löbel, 1998).
- **Modelli a Set Partitioning / Set Covering:** in cui si seleziona un sottoinsieme ottimo di turni ammissibili pre-generati (colonne), garantendo la copertura esatta di ogni corsa.

Sebbene i moderni algoritmi di *Column Generation* e *Branch-and-Price* abbiano spinto in avanti i limiti della risoluzione esatta, la letteratura evidenzia una forte rigidità strutturale

quando ai tradizionali obiettivi di minimizzazione dei costi si affiancano vincoli complessi di natura multi-criterio o di equità distributiva tra operatori indipendenti.

1.3 Il Contesto Industriale e il Problema del Bilanciamento Multi-Operatore

Questa rigidità trova riscontro diretto nella realtà industriale gestita da M.A.I.O.R. S.r.l., società focalizzata sullo sviluppo di software per la mobilità, fondata nel 1989 da esperti di Ricerca Operativa dell'Università di Pisa e oggi parte del gruppo Hitachi Rail. Nelle reti urbane gestite in sinergia da più operatori o suddivise in diversi poli logistici il costo operativo totale non è l'unico parametro rilevante. Come emerso recentemente in diverse realtà del mercato sudamericano, esiste una forte necessità operativa di bilanciare i carichi di lavoro assegnando i turni nel rispetto di specifiche percentuali prefissate per ciascun polo logistico. Le metriche richieste comprendono principalmente:

- Il tempo o i chilometri di effettivo servizio passeggeri (*in-service driving time*).
- Il tempo o i chilometri impiegati nei trasferimenti a vuoto (*deadhead driving time*).
- Il numero totale di turni e veicoli assegnati a ciascuna struttura (*number of vehicle blocks*).

I software commerciali attuali presentano limitazioni strutturali nella gestione di queste dinamiche in modo flessibile, limitandosi a imporre un singolo criterio per volta. Per le aziende di trasporto è invece fondamentale poter incrociare più metriche, cercando il miglior compromesso possibile tra equità logistica e risparmio economico.

Difatti, il presente lavoro si inserisce direttamente nella roadmap di sviluppo tecnologico di MAIOR in vista della prototipazione prevista per l'autunno del 2026. L'obiettivo principale è progettare e implementare un modello matematico basato su *Set Partitioning* e risolto tramite un approccio lessicografico a due fasi, sviluppato in linguaggio Julia e interfacciato con i solutori *general-purpose* HiGHS e Gurobi.

Come suggerito da analisi preliminari e confermato dai requisiti aziendali, l'implementazione di una logica di ripartizione nativa nei moduli euristici richiederebbe sforzi di sviluppo ingenti, questo elaborato valuta la validità e i limiti di un'architettura di post-processing *ex post*. In pratica, a partire da un sottoinsieme di turni ammissibili generato a monte senza vincoli di ripartizione, il modello verifica la capacità di un solutore esterno di ripartire equamente le risorse operative, fornendo al contempo un quadro diagnostico fondamentale sui confini strutturali e topologici della rete.

1.4 Struttura dell'elaborato

Questa tesi è organizzata in cinque capitoli principali, strutturati come segue:

- Il **Capitolo 2** fornisce i richiami teorici fondamentali sulla Programmazione Lineare Intera (PLI) e introduce formalmente il modello matematico sviluppato, basato su una formulazione esatta di *Set Partitioning* risolta tramite un approccio lessicografico a due fasi.
- Il **Capitolo 3** descrive le procedure informatiche di elaborazione dei dati. Vengono analizzati la struttura dei file operativi, le tecniche di estrazione in linguaggio Julia per l'ottimizzazione della memoria e la traduzione in codice delle metriche di valutazione richieste.
- Il **Capitolo 4** presenta una prima fase prototipale dello studio, analizzando i risultati su un caso aziendale ridotto a due depositi, essenziale per validare la correttezza logica del bilanciamento e del modello algebrico.
- Il **Capitolo 5** affronta l'estensione del problema all'intera rete operativa a tre depositi. La sezione analizza in primis la trattabilità computazionale del modello su larga scala, risolvendo l'estrema degenerazione primale tramite l'introduzione di un parametro di regolarizzazione (ϵ) e tecniche euristiche avanzate. Successivamente, esplora l'ottimizzazione multi-obiettivo simultanea (bilanciando congiuntamente turni e chilometri), misurando i limiti topologici dell'infrastruttura (rigidità di rete) e il compromesso tra efficienza economica ed equità al variare della tolleranza. Il capitolo si chiude validando il modello attraverso severi *stress test* su target fortemente asimmetrici, che fungono da strumento diagnostico per evidenziare i limiti dell'approccio *ex post* e giustificare futuri sviluppi architetturali basati su *Column Generation*.
- Il **Capitolo 6**, infine, sintetizza i traguardi raggiunti con lo sviluppo del prototipo e traccia le possibili linee di sviluppo future per l'integrazione del modello a livello industriale.

Contesto Aziendale e Modellizzazione Matematica

2.1 Richiami teorici: Programmazione Lineare Intera e Metodi Risolutivi

La Programmazione Lineare Intera (PLI) costituisce uno degli approcci di riferimento nella Ricerca Operativa per la modellazione e risoluzione di problemi di ottimizzazione combinatoria tra cui il *Vehicle Scheduling*. A differenza della Programmazione Lineare (PL) continua, in cui le variabili decisionali possono assumere valori reali, nella PLI si richiede che alcune o tutte le variabili assumano esclusivamente valori interi.

Un problema di PLI può essere definito come la ricerca del minimo (o massimo) di una funzione obiettivo lineare, soggetta a un sistema di vincoli lineari e a condizioni di interezza:

$$\begin{aligned}
 \min \quad & c^T x \\
 \text{s.t.} \quad & Ax \geq b \\
 & x \geq 0 \\
 & x \in \mathbb{Z}^n \quad (\text{oppure } x \in \{0, 1\}^n)
 \end{aligned} \tag{2.1}$$

Mentre i problemi di PL continua possono essere risolti in tempo polinomiale grazie ad algoritmi esatti, l'aggiunta del vincolo di interezza rende la PLI un problema NP-arduo. Per questo motivo, solutori moderni come HiGHS o Gurobi non si affidano a un'enumerazione esaustiva delle combinazioni possibili, ma sfruttano tecniche algoritmiche in grado di esplorare lo spazio delle soluzioni in modo mirato

2.1.1 Il Rilassamento Lineare e l'Algoritmo del Simplex

Il processo risolutivo parte tipicamente dal cosiddetto *Rilassamento Lineare*, che consiste nell'ignorare temporaneamente i vincoli di interezza, ammettendo variabili frazionarie (es. $x_i \in [0, 1]$).

Risolvendo questo problema continuo, si ottiene un lower bound fondamentale: il costo della soluzione ottima del problema continuo sarà sempre minore o uguale a quello della soluzione intera ottima. Per risolvere questo stadio si utilizza tipicamente l'*Algoritmo del Simplex*. Sviluppato da George Dantzig, questo metodo sfrutta il principio geometrico secondo cui la soluzione ottima di un problema lineare, qualora esista e sia finita, si trova necessariamente in corrispondenza di un vertice del poliedro convesso che descrive la regione ammissibile. Il Simplex esplora iterativamente i vertici adiacenti del poliedro muovendosi lungo direzioni di discesa del costo, fino a raggiungere l'ottimo globale.

2.1.2 L'Esplorazione dello Spazio: Branch & Bound

Se la soluzione ottima fornita dal Simplex nel rilassamento lineare presenta valori esclusivamente interi, essa è esattamente l'ottimo del problema originale di PLI e l'algoritmo termina. Tuttavia, nei problemi complessi la soluzione risulta quasi sempre frazionaria. Per individuare la soluzione intera ammissibile si ricorre pertanto all'algoritmo di *Branch & Bound* (B&B).

Il B&B si basa su un approccio *divide et impera*, partizionando iterativamente lo spazio delle soluzioni ammissibili. Tale processo genera una struttura ad albero decisionale: partendo dal nodo radice (che rappresenta il rilassamento lineare dell'intero problema), l'algoritmo seleziona una variabile frazionaria per generare due nuovi sottoproblemi (*Branching*), imponendo vincoli più stringenti (ad esempio $x_i = 0$ in un ramo e $x_i = 1$ nell'altro).

In ciascun nodo figlio viene risolto un nuovo rilassamento lineare. La valutazione di ciascun nodo (*Bounding*) permette di interrompere l'esplorazione dei rami il cui lower bound risulta peggiore della migliore soluzione intera già nota, riducendo drasticamente i tempi di calcolo. L'algoritmo termina quando tutti i nodi sono stati esplorati o scartati.

2.1.3 L'applicazione al Set Partitioning

Nel modello di *Vehicle Scheduling* sviluppato in questa tesi, il problema è formulato interamente come un *Set Partitioning* con variabili decisionali binarie ($x \in \{0, 1\}$). A partire da un insieme di turni ammissibili pre-generati, l'obiettivo è selezionare un sottoinsieme esatto che garantisca la copertura di tutte le corse in orario.

L'imposizione del vincolo di uguaglianza stretta ($= 1$) genera tipicamente un poliedro ammissibile in cui i vertici ottimi del rilassamento continuo risultano fortemente frazionari. Questa debolezza strutturale del *bound* inferiore ostacola il *pruning* dei rami nell'albero di B&B, compromettendo i tempi di convergenza. Per gestire questa complessità computazionale su dataset di grandi dimensioni, il modello matematico è stato affiancato da un'implementazione efficiente. Le logiche di sviluppo in linguaggio Julia, l'ottimizzazione della memoria e la calibrazione del solutore saranno oggetto di approfondimento nei capitoli successivi.

2.2 Formulazione Matematica del Modello

Il problema è stato modellizzato estendendo la classica formulazione di *Set Partitioning*, introducendo vincoli operativi di flotta e un sistema di calcolo generalizzato per le metriche di bilanciamento.

Insiemi e Parametri

Siano definiti i seguenti insiemi e parametri fondamentali forniti in input:

- I : l'insieme dei turni macchina candidati.
- J : l'insieme delle corse (journeys) che compongono il servizio da coprire.
- K : l'insieme dei depositi o operatori.
- V : l'insieme delle tipologie di veicoli disponibili.
- $I(k) \subseteq I$: sottoinsieme dei turni assegnati al deposito k .
- $I(k, v) \subseteq I(k)$: sottoinsieme dei turni del deposito k operati da veicoli di tipo v .
- $a_{ji} \in \{0, 1\}$: parametro binario pari a 1 se il turno i copre la corsa j , 0 altrimenti.
- Cap_k : capacità massima (espressa in numero di turni) del deposito k .
- Veh_{kv} : numero massimo di veicoli di tipo v fisicamente presenti nel deposito k .
- $P_k \in [0, 1]$: quota percentuale target assegnata al deposito k .
- c_i : costo operativo associato al turno i .
- q_i : parametro generalizzato che rappresenta la metrica da equidistribuire (es. numero di blocchi, tempo di guida, chilometri) per il turno i .

Variabili Decisionali

Il modello definisce le seguenti variabili:

- $x_i \in \{0, 1\}$: variabile binaria pari a 1 se il turno i viene selezionato in soluzione, 0 altrimenti.
- $y_k \geq 0, \in \mathbb{Z}$: numero totale di turni assegnati al deposito k .
- $v_k \geq 0$: somma della metrica generalizzata q_i accumulata dal deposito k .
- $V_{TOT} \geq 0$: volume totale della metrica q_i generato sull'intera rete.
- $s_k \geq 0$: errore (scarto) assoluto tra la metrica assegnata al deposito k e il target teorico.

Vincoli Strutturali e di Copertura

La condizione necessaria di ammissibilità, che definisce la struttura di *Set Partitioning*, richiede che ogni corsa venga coperta esattamente una volta:

$$\sum_{i \in I} a_{ji} x_i = 1 \quad \forall j \in J \quad (2.2)$$

I vincoli seguenti definiscono i limiti fisici della rete, garantendo che i turni assegnati non superino la capacità dei piazzali e la reale disponibilità della flotta per ciascun tipo di mezzo:

$$y_k = \sum_{i \in I(k)} x_i \quad \forall k \in K \quad (2.3)$$

$$y_k \leq Cap_k \quad \forall k \in K \quad (2.4)$$

$$\sum_{i \in I(k,v)} x_i \leq Veh_{kv} \quad \forall k \in K, \forall v \in V \quad (2.5)$$

Vincoli di Bilanciamento e Linearizzazione

Per valutare l'equità logistica, il modello calcola i volumi totali e parziali della metrica di interesse (q_i) e ne linearizza lo scarto assoluto $s_k = |v_k - P_k V_{TOT}|$ tramite due disequazioni complementari:

$$V_{TOT} = \sum_{i \in I} q_i x_i \quad (2.6)$$

$$v_k = \sum_{i \in I(k)} q_i x_i \quad \forall k \in K \quad (2.7)$$

$$s_k \geq v_k - (P_k \cdot V_{TOT}) \quad \forall k \in K \quad (2.8)$$

$$s_k \geq (P_k \cdot V_{TOT}) - v_k \quad \forall k \in K \quad (2.9)$$

2.2.1 Approccio Lessicografico a Due Fasi

Per gestire l'ottimizzazione multiobiettivo tra l'equità distributiva e l'efficienza economica, il problema è stato strutturato secondo un approccio lessicografico a due fasi sequenziali.

Fase 1: Minimizzazione dello Scarto Strutturale

Il primo obiettivo è ricercare il massimo bilanciamento possibile, calcolando l'errore minimo strutturale date le risorse a disposizione:

$$\min \sum_{k \in K} s_k \quad (2.10)$$

Sia S^* il valore ottimo numerico restituito dal solutore al termine di questa fase.

Fase 2: Minimizzazione dei Costi con Tolleranza

Nella seconda fase, l'obiettivo primario diventa la minimizzazione dei costi operativi, vincolando l'errore distributivo a non superare il valore ottimo precedentemente individuato, concedendo al massimo una minima tolleranza $\delta \geq 0$:

$$\min \sum_{i \in I} c_i x_i \tag{2.11}$$

$$\sum_{k \in K} s_k \leq S^* + \delta \tag{2.12}$$

Elaborazione dei Dati e Implementazione in Julia

3.1 Analisi della struttura dei file operativi MAIOR

I sistemi proprietari di MAIOR esportano la configurazione della rete in file di testo con estensione `.vbds`. Questi file contengono tutte le informazioni strutturali e di costo necessarie per costruire il modello matematico del VSP. I dati sono organizzati in blocchi gerarchici, delimitati da parentesi graffe e hashtag (`#{ ... #}`).

Per dare un'idea concreta della sintassi, il Listato 3.1 mostra un estratto semplificato delle sezioni lette dall'algoritmo:

```

1  #{
2  DEPOTS 2
3  #{
4  DEPOT DEPOT_A DEPOT_A DEPOT_A 54000 5400 0 0 1 217 ??? ???
5  #}
6  #{
7  DEPOT DEPOT_B DEPOT_B DEPOT_B 54000 1800 0 0 1 95 ??? ???
8  #}
9  #}
10
11 #{
12 VEHICLES_IN_DEPOT 7
13 DEPOT_A MnB_2p_ELE 7
14 DEPOT_A STD_2p_Hyb 68
15 DEPOT_B STD_2p 73
16 #}
17
18 #{
19 VEHICLE_BLOCK 0 ??? DEPOT_B STD_2p_Hyb 177 48044.5 720.845 [...]
20 #{
21 COSTS 6
22 AlgoCost 1.32 C

```

```

23 VLengthCost 0 F
24 #}
25 #{
26 PIECES_LIST 10
27 #{
28 DEADHEAD_PIECE DEPOT_B N24078 55800 55860 1 0
29 #}
30 #{
31 JOURNEY_PIECE N24078 N22430 55860 57660 1743950
32 #}
33 #}
34 #}

```

Listing 3.1: Esempio generico della struttura di un file .vbds

L'estrazione delle informazioni chiave si concentra su tre livelli, come si può vedere:

- **DEPOTS e VEHICLES_IN_DEPOT**: definiscono i limiti fisici del problema. Indicano la capacità massima dei piazzali (es. 217 turni per *DEPOT_A*) e la disponibilità della flotta divisa per tipo di veicolo.
- **VEHICLE_BLOCKS**: contiene l'elenco dei turni candidati, ovvero le nostre variabili decisionali x_i . Dalla riga di intestazione si leggono il deposito di appartenenza, il veicolo usato e i chilometri percorsi.
- **COSTS e PIECES_LIST**: definiscono il costo operativo del turno (*AlgoCost*) e l'elenco delle corse da coprire. Da qui si estraggono gli ID univoci delle corse commerciali (*JOURNEY_PIECE*) per costruire il vincolo di copertura esatta.

3.2 Strategie di pre-processing e gestione dei dati in Julia

Per evitare di saturare la memoria del sistema caricando l'intero file in RAM, la fase di lettura (*parsing*) è stata implementata in Julia utilizzando la funzione nativa `eachline()`, che consente di analizzare il documento sequenzialmente, riga per riga.

Dal punto di vista computazionale, la traduzione in codice dei parametri matematici (come i limiti fisici della flotta Veh_{kv} e le capacità Cap_k) richiede strutture dati capaci di garantire tempi di accesso minimi. A tal fine, i parametri estratti sono stati organizzati in:

- **Dizionari (*Dict*)**: implementati come *hash table* per associare le chiavi testuali univoche (es. «DPPLE_MnB_2p_ELE») alle rispettive capacità. Questa scelta algoritmica garantisce tempi di ricerca costanti ($\mathcal{O}(1)$), rivelandosi cruciale per evitare iterazioni superflue ed esaurienti scansioni di array durante la successiva generazione dei vincoli algebrici.

- **Vettori (*Vector*):** utilizzati per immagazzinare le caratteristiche dei singoli turni (costi, impianto, corse coperte, chilometri), mantenendo i dati strutturalmente allineati con l'indice i della variabile decisionale x_i .

Di seguito è riportato un estratto rappresentativo del blocco logico dedicato all'estrazione dei limiti di flotta e all'identificazione dei turni macchina:

```
1 # Estrazione limiti di flotta
2 if length(elementi) >= 3 && haskey(capacita_depositi, elementi[1])
3     deposito = elementi[1]
4     tipo_veicolo = elementi[2]
5     quantita = parse{Int, elementi[3]}
6     chiave_veicolo = deposito * "_" * tipo_veicolo
7     flotta[chave_veicolo] = quantita
8 end
9
10 # Lettura di un nuovo turno
11 if elementi[1] == "VEHICLE_BLOCK"
12     dentro_blocco_turno = true
13     deposito_del_turno = elementi[4]
14     tipo_veicolo_del_turno = elementi[5]
15
16     # Valutazione delle distanze
17     km_in_linea = parse{Float64, elementi[7]}
18     km_fuori_linea = parse{Float64, elementi[8]}
19
20     push!(depositi_turni, deposito_del_turno)
21     push!(veicoli_turni, tipo_veicolo_del_turno)
22     push!(journeys_turni, String[])
23 end
```

Listing 3.2: Estrazione dei parametri in Julia

3.3 Definizione delle metriche di valutazione

Una delle richieste operative dell'azienda era avere un modello flessibile, capace di bilanciare criteri logistici diversi. Nel codice, questa versatilità è gestita dalla funzione `calcola_qi`, che riceve in input la stringa identificativa della metrica desiderata dall'utente e genera il vettore dei parametri per la funzione obiettivo.

```
1 function calcola_qi(opzione::String, numero_turni::Int, km_in::Vector{
2     Float64}, km_fuori::Vector{Float64}, km_tot::Vector{Float64})
3
4     q_turni = zeros(Float64, numero_turni)
5
6     if opzione == "number_of_blocks"
7         q_turni = ones(Float64, numero_turni)
8     elseif opzione == "distance_total"
9         q_turni = copy(km_tot)
10     end
11 end
```

```

8     elseif opzione == "distance_in_line"
9         q_turni = copy(km_in)
10    elseif opzione == "distance_out_line"
11        q_turni = copy(km_fuori)
12    else
13        q_turni = ones(Float64, numero_turni)
14    end
15
16    return q_turni
17 end

```

Listing 3.3: Definizione dei coefficienti q_i

Come si evince dal codice, nel caso del conteggio dei turni (`number_of_blocks`) a ogni turno viene assegnato il valore unitario, mentre per le metriche spaziali i valori vengono ereditati dai dati grezzi estratti dal file sorgente. Tali metriche verranno impiegate come base di calcolo nei test computazionali discussi nei capitoli successivi.

3.4 Costruzione del modello matematico con JuMP e HiGHS

Con i dati pronti in memoria, il prototipo sfrutta JuMP, un pacchetto per la modellazione algebrica (AML) nativo per Julia, per istanziare le variabili decisionali e i vincoli.

Nelle fasi preliminari dello sviluppo, i test di validazione del modello sono stati condotti avvalendosi di Gurobi, solutore commerciale *state-of-the-art* per problemi di Programmazione Lineare Mista Intera (MIP). Tuttavia, un'attenta analisi comparativa condotta sul caso di studio a due depositi ha dimostrato che HiGHS, un potente solutore *open-source*, era in grado di garantire tempi di convergenza e standard qualitativi del tutto equiparabili alla controparte commerciale. Al fine di mantenere l'intera architettura algoritmica accessibile, facilmente replicabile e indipendente da licenze proprietarie, HiGHS è stato pertanto selezionato come motore risolutivo *backend* definitivo per l'intero lavoro.

In ambito aziendale, ottimizzare reti di grandi dimensioni può richiedere ore. Per avere tempi di risposta compatibili con le necessità pratiche e prevenire colli di bottiglia computazionali, il solutore è stato calibrato impostando un limite di tempo (*time limit*) e una tolleranza sul gap di ottimalità relativo (*MIP gap*) pari all'1%. Tali soglie permettono di arrestare l'algoritmo non appena viene individuata una soluzione di alta qualità matematica, senza disperdere risorse computazionali per cercare affinamenti decimali irrilevanti dal punto di vista operativo.

```

1 # Inizializzazione del modello
2 modello = Model(HiGHS.Optimizer)
3
4 # Impostazione dei limiti per avere tempi di risposta ragionevoli
5 set_optimizer_attribute(modello, "time_limit", 600.0)
6 set_optimizer_attribute(modello, "mip_rel_gap", 0.01)

```

Listing 3.4: Inizializzazione del solutore e limiti di calcolo

La sintassi di JuMP è molto intuitiva, ma la stesura del vincolo di *Set Partitioning* ($\sum_{i \in I} a_{ji} x_i = 1$) ha richiesto una particolare attenzione pratica per l'ottimizzazione dei tempi di calcolo.

Matematicamente, questo vincolo impone che per ogni corsa $j \in J$ venga selezionato esattamente un turno $i \in I$ che la copra. In un'implementazione ingenua, per costruire questo sistema di equazioni si dovrebbe interrogare il set di dati innumerevoli volte, incrociando continuamente l'intero insieme delle corse con l'insieme di tutti i turni candidati.

Per ottimizzare questo processo, l'algoritmo sfrutta un dizionario di appoggio (`turni_per_corsa`). Il grande vantaggio di questa struttura è che viene popolata con un'unica iterazione durante la fase di lettura iniziale del file di testo (*parsing*): man mano che il documento viene elaborato, il dizionario memorizza per ogni singola corsa j la lista esatta dei turni i che la contengono.

Di conseguenza, nel momento in cui JuMP deve generare materialmente le equazioni in memoria, non vi è alcuna necessità di rileggere i dati o di iterare su tutti i turni che non coprono quella specifica corsa (ovvero in corrispondenza dei coefficienti nulli della matrice di incidenza). Il programma interroga semplicemente il dizionario e inserisce nella sommatoria esclusivamente le variabili decisionali x_i associate, costruendo il vincolo in modo diretto e pressoché istantaneo.

3.4.1 Implementazione dell'Approccio Lessicografico a Due Fasi e Tolleranza Dinamica

Il grande vantaggio di usare JuMP per il metodo a due fasi è che permette di risolvere il modello in sequenza, alterando la funzione obiettivo in corsa senza dover distruggere e ricreare le variabili decisionali.

Nella Fase 1, la funzione obiettivo è impostata per minimizzare solo la somma degli scarti strutturali tra gli operatori. Trovata la soluzione ottimale, il valore minimo dell'errore (salvato nella variabile S^*) viene recuperato con il comando `objective_value()`. A questo punto, il modello viene aggiornato aggiungendo un vincolo strutturale, imponendo che l'errore distributivo finale non superi S^* maggiorato di un parametro di tolleranza δ .

Un elemento cruciale dell'implementazione è la definizione logica di questo parametro. Le prime sperimentazioni hanno evidenziato che l'utilizzo di un valore fisso e assoluto restringe eccessivamente lo spazio delle soluzioni ammissibili qualora si valutino metriche caratterizzate da volumi molto ampi (come il chilometraggio totale). Questo rallentava drasticamente i tempi di convergenza dell'algoritmo di *Branch & Bound*, portando frequentemente il solutore a esaurire il tempo limite preimpostato senza riuscire a individuare soluzioni intere valide.

Per garantire la scalabilità del modello su qualsiasi grandezza fisica, il limite è stato implementato in logica dinamica: il parametro di rilassamento δ viene calcolato *a run-time* come percentuale (es. 2%) del volume logistico totale elaborato nella Fase 1. Nel caso specifico del puro conteggio dei turni (`number_of_blocks`), per evitare di imporre tolleranze sub-unitarie, il codice prevede una soglia minima di salvaguardia ($\delta \geq 1.5$): tale valore garantisce al solutore il margine algebrico sufficiente per scostarsi di almeno un veicolo fisico intero, preservando sempre l'ammissibilità del problema.

```
1 # --- FASE 1: MINIMIZZARE L'ERRORE LOGISTICO ---
2 @objective(mio_modello, Min, sum(s[o] for o in lista_operatori))
3 optimize!(mio_modello)
4 S_star = objective_value(mio_modello)
5
6 # --- FASE 2: MINIMIZZARE I COSTI OPERATIVI ---
7 V_TOT_fase1 = value(mio_modello[:V_TOT])
8 percentuale_tolleranza = 0.02
9
10 # Calcolo del delta dinamico con salvaguardia per la metrica a blocchi
11 if metrica_scelta == "number_of_blocks"
12     delta = max(1.5, percentuale_tolleranza * V_TOT_fase1)
13 else
14     delta = percentuale_tolleranza * V_TOT_fase1
15 end
16
17 # Inserimento del vincolo adattivo sull'errore accettato
18 @constraint(mio_modello, vincolo_V9, sum(s[o] for o in lista_operatori) <=
19     S_star + delta)
20
21 # Cambio della funzione obiettivo
22 @objective(mio_modello, Min, sum(costi[i] * x[i] for i in 1:totale_turni))
23 optimize!(mio_modello)
```

Listing 3.5: Calcolo della tolleranza dinamica ed esecuzione di Fase 1 e Fase 2

Questo accorgimento permette di «allargare» proporzionalmente la regione ammissibile per l'algoritmo, agevolando la risoluzione ed evitando l'attivazione ingiustificata di nuovi veicoli. Al termine della Fase 2, se la ricerca ha esito positivo, i turni estratti come soluzione ottima ($x_i = 1$) vengono trascritti e isolati in un nuovo file `.vbds`, pronto per essere reimportato e analizzato dal software dell'azienda.

Caso di Studio A: Modello a 2 Depositi

4.1 Descrizione dello scenario operativo

Per validare computazionalmente il modello algebrico, è stato condotto un primo caso di studio operando su un'istanza infrastrutturale ridotta. Il dataset fornito da MAIOR comprende un totale di 2.339 corse (*journeys*) da coprire, le quali compongono l'orario ufficiale del servizio. Il solutore matematico ha il compito di estrarre il sottoinsieme ottimo selezionando le variabili binarie da uno spazio delle soluzioni composto da circa 120.000 turni macchina candidati.

In questo scenario, la rete fa capo a due soli poli logistici:

- **DPASN**: deposito di dimensioni ridotte, con un limite superiore di capienza fissato a 95 turni.
- **DPPLE**: deposito principale, avente una capacità massima di 217 turni.

L'analisi computazionale presentata in questo capitolo valuta la risposta algoritmica del modello su tutte le metriche di bilanciamento precedentemente definite. I paragrafi seguenti illustrano i risultati ottenuti applicando progressivamente l'ottimizzazione alla cardinalità pura della flotta (`number_of_blocks`), per poi estendere l'analisi alle grandezze spaziali relative alle percorrenze in linea, a vuoto e totali.

4.2 Analisi computazionale sulle metriche di bilanciamento

4.2.1 Risultati sulla metrica Number of Blocks

Il primo ciclo di test analizza la metrica operativa più elementare: il bilanciamento del numero totale di turni (`number_of_blocks`). La Tabella 4.1 riassume i valori ottimi ottenuti con una tolleranza dinamica del 2% (limitata inferiormente a $\delta = 1.5$ per evitare vincoli sub-unitari).

Target (PLE / ASN)	S^*	Costo Totale	Flotta	Turni (LE / ASN)	Distr. Reale	Tempo F1 / F2 (s)	Tempo Tot. (s)
90% / 10%	0.0	111.94	48	43 / 5	89.6% / 10.4%	66.3 / 18.3	84.6
70% / 30%	0.0	108.01	48	33 / 15	68.8% / 31.2%	77.2 / 4.6	81.8
50% / 50%	0.0	107.68	48	24 / 24	50.0% / 50.0%	75.2 / 25.1	100.3
30% / 70%	0.0	133.94	74	23 / 51	31.1% / 68.9%	82.3 / 41.3	123.6
10% / 90%	26.0	152.14	90	23 / 67	25.6% / 74.4%	143.9 / 61.5	205.4

Tabella 4.1: Risultati dell’ottimizzazione basata sul numero di turni macchina (*number_of_blocks*).

Equilibrio e propensione naturale della rete

Imponendo un vincolo di perfetta equità (Target 50% / 50%), il solutore individua facilmente una scomposizione esatta: il modello attiva complessivamente 48 turni divisi simmetricamente, convergendo in 100.3 secondi con un costo operativo di 107.68.

Spostando le percentuali a favore del deposito DPPLE (scenari 70%/30% e 90%/10%), il numero totale dei turni si mantiene stabile sull’ottimo di 48 veicoli, con incrementi del tutto trascurabili sui costi. Dal punto di vista pratico, questo dato matematico ci permette di fare una deduzione molto chiara: anche senza conoscere nel dettaglio la mappa della città, i risultati dimostrano che la distribuzione spaziale e temporale delle corse favorisce naturalmente questo deposito. L’algoritmo riesce infatti a concatenare i viaggi in modo molto più efficiente quando ha margine per appoggiarsi maggiormente a DPPLE.

Il limite fisico e l’esplosione dei costi

L’aspetto più interessante emerge forzando le percentuali nella direzione opposta, ovvero chiedendo al modello di assegnare la maggior parte del lavoro al deposito DPASN (scenari 30%/70% e 10%/90%).

In questo caso, l’approccio a due fasi rivela un limite invalicabile della rete: nella Fase 1, il solutore si ferma su un errore strutturale minimo ineliminabile ($S^* = 26.0$ nello scenario 10%/90%). Questo dato dimostra che è materialmente impossibile coprire tutte le corse in orario attivando meno di 23 macchine del deposito DPPLE. Dal punto di vista pratico, significa che una fetta consistente del servizio è geograficamente «vincolata» a questo impianto, probabilmente perché alcuni capolinea sono semplicemente troppo distanti da DPASN per poter essere serviti in modo sensato.

Non potendo abbassare l’impiego di questo deposito, il solutore cerca comunque di soddisfare la percentuale richiesta accrescendo il denominatore, ovvero espandendo il totale dei veicoli. Negli scenari 30%/70% e 10%/90%, il modello è quindi costretto ad attivare decine di turni frammentati e inefficienti dal deposito DPASN. Il risultato pratico è un’esplosione ingiustificata della flotta (che passa dai 48 ottimali a ben 90 veicoli) e un aumento inaccettabile dei costi operativi, che arrivano a 152.14. Tuttavia, l’uso della

tolleranza dinamica ha mitigato questa inefficienza strutturale: rispetto alle prime calibrazioni a tolleranza fissa (che richiedevano 93 veicoli per lo scenario 10%/90% e 75 per il 30%/70%), l'approccio adattivo ha permesso al solutore di risparmiare rispettivamente 3 e 1 turni macchina.

4.2.2 Risultati sulla metrica Chilometri in Linea

La seconda fase di test ha valutato la distribuzione del carico di lavoro produttivo, ovvero i chilometri percorsi in servizio passeggeri (`distance_in_line`). I risultati (Tabella 4.2) confermano l'asimmetria geografica dell'infrastruttura ed evidenziano il potenziale risolutivo del parametro δ .

Target (DPPLE / DPASN)	S*	Costo Totale	Flotta	Turni (LE / ASN)	Distr. Reale	Tempo F1 / F2 (s)	Tempo Tot. (s)
90% / 10%	2.44	111.17	48	41 / 7	89.2% / 10.8%	600.0* / 3.0	603.0
70% / 30%	0.5	108.14	48	33 / 15	69.5% / 30.5%	600.0* / 5.5	605.5
50% / 50%	0.5	107.71	48	24 / 24	50.8% / 49.2%	600.5* / 8.6	609.1
30% / 70%	9745.08	145.32	67	29 / 38	31.0% / 69.0%	600.5* / 34.7	635.2
10% / 90%	2.6×10^6	147.93	69	29 / 40	30.6% / 69.4%	31.4 / 37.6	69.0

Tabella 4.2: Ottimizzazione basata sui chilometri in linea.

Su un volume di oltre 6,7 milioni di chilometri, un'eventuale tolleranza fissa di 2.500 km generava un *bound* così restrittivo (circa 0,04%) da esaurire il tempo computazionale concesso negli scenari 30%/70% e 10%/90% senza individuare soluzioni ammissibili. Il rilassamento dinamico al 2% ($\delta \approx 134.000$ km) ha invece ampliato lo spazio di ricerca, permettendo all'algoritmo di convergere entro il *time limit* imposto, pur palesando, anche in questo caso, l'inevitabile frammentazione dei turni per forzare i carichi su DPASN.

4.2.3 Risultati sulla metrica Chilometri a Vuoto

L'analisi focalizzata sui trasferimenti a vuoto (`distance_out_line`), ovvero gli spostamenti tra depositi e capolinea, dimostra la netta asimmetria posizionale dell'infrastruttura.

Target (DPPLE / DPASN)	S*	Costo Totale	Flotta	Turni (LE / ASN)	Distr. Reale	Tempo F1 / F2 (s)	Tempo Tot. (s)
90% / 10%	0.04	108.45	49	34 / 15	89.3% / 10.7%	600.0* / 21.7	621.7
70% / 30%	2.66	107.58	48	25 / 23	70.8% / 29.2%	600.1* / 5.4	605.5
50% / 50%	5.78	119.53	53	23 / 30	51.0% / 49.0%	600.0* / 31.8	631.8
30% / 70%	68189.85	162.69	84	24 / 60	34.9% / 65.1%	64.1 / 125.2	189.3
10% / 90%	432584.87	150.20	75	24 / 51	38.1% / 61.9%	34.2 / 36.3	70.5

Tabella 4.3: Ottimizzazione basata sui trasferimenti a vuoto.

I dati certificano che l'impianto DPPLE gode di un vantaggio geografico per la gestione dei rientri. Quando la maggioranza dei trasferimenti è allocata su questo deposito (scenari 70%/30% e 90%/10%), la rete logistica opera nella sua condizione ottimale e fisiologica. Al contrario, a differenza delle altre metriche, l'imposizione di un target equo (50%/50%) causa un forte degrado della funzione obiettivo: imponendo a DPASN di percorrere metà dei chilometri a vuoto totali, il modello alloca 53 macchine, portando i costi operativi a 119.53. L'utilizzo della flessibilità dinamica, pur raggiungendo il 51.0% / 49.0% si è rivelato cruciale per fermare la flotta a 53 unità, evitando l'attivazione forzata del 54esimo veicolo che veniva invece richiesta matematicamente dalle prime implementazioni a tolleranza fissa ($\delta = 500$ km), salvaguardando l'economicità aziendale senza compromettere l'equità.

4.2.4 Risultati sul Chilometraggio Totale

L'integrazione delle componenti lineari e a vuoto all'interno del parametro di chilometraggio totale (`distance_total`) dimostra la robustezza del framework e la sua capacità di calcolare compromessi spaziali complessi.

Target (DPPLE / DPASN)	S*	Costo Totale	Flotta	Turni (LE / ASN)	Distr. Reale	Tempo F1 / F2 (s)	Tempo Tot. (s)
90% / 10%	1.94×10^5	108.06	48	32 / 16	69.5% / 30.5%	56.2 / 12.7	68.9
70% / 30%	2.66	111.18	48	41 / 7	89.5% / 10.5%	600.1* / 5.4	605.5
50% / 50%	11.56	107.88	48	23 / 25	51.0% / 49.0%	600.0* / 19.5	619.5
30% / 70%	1.95×10^5	167.19	78	30 / 48	32.3% / 67.7%	56.2 / 19.7	75.9
10% / 90%	3.41×10^6	149.44	70	30 / 40	33.4% / 66.6%	36.5 / 34.0	70.5

Tabella 4.4: Ottimizzazione basata sui chilometri complessivi percorsi.

Lo scenario 50/50 dimostra la capacità di autocompensazione del solutore: il modello ripristina l'equilibrio (51.0% / 49.0%) e ritorna all'ottimo di 48 veicoli, bilanciando l'inefficienza precedentemente osservata nei trasferimenti a vuoto tramite un'allocazione strategica dei chilometri in linea. Inoltre, il test 90%/10% esplicita il valore economico della flessibilità: concedendo al modello di scostarsi di un marginale 0.5% rispetto alla rigidità teorica, si evita l'attivazione forzata del 49esimo veicolo (riscontrata nei test a tolleranza fissa con $\delta = 3000$ km). Questa libertà algebrica ha abbassato la funzione di costo da 111.48 a 111.17, ottimizzando l'impiego complessivo della flotta.

4.3 Benchmark Comparativo: HiGHS vs. Gurobi e Analisi Costo-Efficacia

Un quesito centrale per l'integrazione del modello nella *suite* commerciale di M.A.I.O.R. riguarda la quantificazione del divario prestazionale tra un solutore commerciale proprietario *state-of-the-art* (Gurobi Optimizer) e un solutore *open-source* (HiGHS). Se da una parte

l'adozione di un motore commerciale comporta oneri di licenza, dall'altra l'impiego di una soluzione aperta deve garantire tempi di risposta compatibili con i cicli di pianificazione.

Per valutare tale *trade-off*, i due solutori sono stati messi a confronto sui due scenari più rilevanti per la rete a due depositi: lo scenario di perfetto equilibrio (Target 50% / 50%) e lo scenario di stress asimmetrico (Target 10% / 90%), valutandoli su tutte e quattro le metriche operative.

4.3.1 Parametrizzazione Algoritmica a Confronto

La differenza architetturale tra i due motori ha richiesto l'adozione di parametri operativi differenti:

- **Gurobi 13.0:** è stato eseguito con parametri puramente di *default*. La convergenza rapida è assicurata dall'attivazione nativa di tecniche avanzate di ricerca euristica e dalla generazione automatica di piani di taglio aggressivi (*Gomory*, *Zero-half*, *Clique*).
- **HiGHS 1.15.0:** l'assenza di tagli proprietari avanzati ha reso necessario un limite temporale di 600 secondi per arginare l'esplorazione dell'albero di *Branch & Bound* in Fase 1, affidandosi maggiormente alla tolleranza dinamica δ nella Fase 2 per compensare il gap prestazionale. Su scenari logistici più estesi si è reso inoltre necessario l'innalzamento del parametro `mip_heuristic_effort` (il cui impatto risolutivo e la relativa trattazione di dettaglio sono rimandati alla Sezione 5.2).

I risultati computazionali aggregati sono suddivisi nelle Tabelle 4.5 e 4.6.

Metrica	Solutore	Tempo Fase 1 (s)	Tempo Fase 2 (s)	Tempo Totale (s)
Number of Blocks	Gurobi	2.1	1.2	3.3
	HiGHS	75.2	25.1	100.3
Distance In Line	Gurobi	140.7	1.0	141.7
	HiGHS	600.5*	8.6	609.1
Distance Out Line	Gurobi	25.9	4.4	30.3
	HiGHS	600.0*	31.8	631.8
Distance Total	Gurobi	19.0	0.7	19.7
	HiGHS	600.0*	19.5	619.5

Tabella 4.5: Confronto prestazionale tra Gurobi 13.0 e HiGHS 1.15.0 nello scenario di equilibrio (Target 50% / 50%). (*Raggiungimento del limite temporale.)

4.3.2 Valutazione Industriale e Prospettive su Larga Scala

L'analisi quantitativa su questa istanza ridotta evidenzia un netto divario di potenza bruta a favore del software proprietario: sfruttando le impostazioni di *default*, Gurobi risulta estremamente più rapido nel certificare l'ottimalità. Risolve infatti in manciate di secondi

4. Caso di Studio A: Modello a 2 Depositi

Metrica	Solutore	Tempo Fase 1 (s)	Tempo Fase 2 (s)	Tempo Totale (s)
Number of Blocks	Gurobi	6.6	2.8	9.4
	HiGHS	143.9	61.5	205.4
Distance In Line	Gurobi	4.1	1.0	5.1
	HiGHS	31.4	37.6	69.0
Distance Out Line	Gurobi	4.0	11.2	15.2
	HiGHS	34.2	36.3	70.5
Distance Total	Gurobi	2.3	11.2	13.5
	HiGHS	36.5	34.0	70.5

Tabella 4.6: Confronto prestazionale tra Gurobi 13.0 e HiGHS 1.15.0 nello scenario di stress asimmetrico (Target 10% / 90%).

istanze su cui HiGHS fatica a chiudere il *MIP Gap*, esaurendo frequentemente i 10 minuti di limite temporale imposti in Fase 1. Questa supremazia si manifesta in modo evidente soprattutto sulle metriche chilometriche dello scenario 50%/50%, dove la complessità algebrica del partizionamento esatto è maggiore.

Tuttavia, il dato industriale più rilevante ai fini aziendali risiede nell’assoluta equivalenza logistica della soluzione: in tutti gli scenari testati, sia in condizioni di equilibrio sia di stress asimmetrico, entrambi i solutori pervengono esattamente alla medesima funzione obiettivo finale e alla medesima allocazione della flotta. HiGHS compensa la maggiore lentezza nella certificazione del *lower bound* fornendo comunque — grazie all’uso della tolleranza dinamica nella Fase 2 — soluzioni ottime in tempi ampiamente compatibili con i cicli di pianificazione aziendale *offline* (entro gli 11 minuti totali).

Se su una rete a due depositi l’azienda si trova a valutare un lineare *trade-off* tra la drastica riduzione dei tempi di calcolo (Gurobi) e l’abbattimento dei costi di licenza (HiGHS), l’esplosione dimensionale del modello permetterà di testare la tenuta di questo equilibrio su scala reale. Come verrà ampiamente discusso nel prossimo capitolo, estendendo l’analisi all’intera rete infrastrutturale a tre depositi e introducendo un’estrema degenerazione primale, la prevedibile superiorità computazionale del software proprietario verrà confermata, ma la solidità operativa del solutore *open-source* non verrà meno. HiGHS si dimostrerà infatti capace di convergere esattamente alle medesime configurazioni di ottimo, confermandosi non come un semplice ripiego economico, ma come una sofisticata e competitiva risorsa algoritmica ai fini pratici aziendali.

Il Modello Esteso: Caso Studio a 3 Depositi e Gestione della Complessità

Dopo aver validato la correttezza strutturale e la coerenza algebrica del modello di partizionamento su un'istanza ridotta a due soli depositi, la sperimentazione è stata estesa alla rete completa del caso studio. Con questo passaggio si passa a uno scenario operativo più realistico, in cui è necessario gestire contemporaneamente tre poli logistici: DEP_001, DEP_002 e DEP_003. In questo contesto, il volume dei dati subisce una forte esplosione dimensionale: il bacino esplorativo di partenza, generato dall'algoritmo euristico aziendale, consta di oltre 52.000 *Vehicle Blocks* candidati.

5.1 Il Contesto Commerciale e i Requisiti MAIOR

L'estensione dell'analisi a una rete complessa a tre depositi risponde direttamente alle specifiche emerse nella roadmap di sviluppo di Hitachi Rail / M.A.I.O.R. S.r.l. Come evidenziato nell'analisi dei requisiti aziendali per il contestuale bilanciamento multi-operatore, la suite commerciale richiede la capacità di gestire richieste complesse in cui le percentuali target si riferiscono simultaneamente a più criteri, quali il numero di turni macchina (*number_of_blocks*) e i chilometri in servizio (*distance_in_line*).

Tuttavia, la documentazione tecnica di MAIOR evidenziava un limite critico: l'implementazione nativa di tali vincoli comportava stime di sviluppo elevate (50-70 giorni per i soli blocchi, fino a 200 giorni per la gestione congiunta) e il forte rischio di un'esplosione esponenziale dei tempi di calcolo a livello di prodotto. Il presente capitolo modella e sperimenta proprio questo scenario critico, valutando l'efficacia dell'approccio *ex post* nel misurare il compromesso (*trade-off*) tra l'efficienza economica e il rigoroso rispetto dei target distribuitivi.

5.1.1 Generalizzazione del Modello per il Bilanciamento Multi-Metrica

Per gestire la complessità strutturale della rete estesa senza incorrere in limitazioni architetture, lo script sviluppato in linguaggio Julia è stato generalizzato rispetto alla formulazione iniziale. Nel prototipo precedente, il calcolo dell'equità era vincolato a una singola metrica prefissata a monte. L'architettura attuale introduce invece una parametrizzazione basata su dizionari di vettori (`dict_array_qi`), consentendo al modello di attivare dinamicamente una o più grandezze operative in parallelo (es. il conteggio dei turni macchina e i chilometri in linea).

Dal punto di vista algebrico, questa flessibilità si riflette sull'estensione delle variabili decisionali:

- **Gestione parametrica delle metriche:** Nel prototipo a due depositi, lo script era strutturato per analizzare una sola grandezza alla volta, definita tramite una variabile singola. Nel nuovo modello, il parametro è stato sostituito da un vettore dinamico (`lista_metriche`) che permette di attivare e calcolare simultaneamente più grandezze.
- **Indicizzazione multidimensionale degli scarti:** Le variabili di accumulo della metrica per operatore (v_{op}) e le relative variabili di scarto assoluto (s) non sono più monodimensionali, ma vengono istanziate come matrici indicizzate sull'insieme delle metriche attive m e degli operatori o ($v_{op}[m, o]$ e $s[m, o]$).

L'implementazione in JuMP di queste variabili multidimensionali sfrutta la seguente sintassi per permettere al modello di tracciare lo scarto per ogni metrica e operatore all'interno dello stesso ciclo di esecuzione:

```
1
2 # Vettore dinamico delle metriche attivate dall'utente
3
4 lista_metriche = ["number_of_blocks", "distance_in_line"]
5
6 # Variabili decisionali per il bilanciamento: indicizzate su metrica e
  operatore
7
8 @variable(mio_modello, v_op[m in lista_metriche, o in lista_operator]
  >= 0)
9
10 @variable(mio_modello, V_TOT[m in lista_metriche] >= 0)
11
12 # Variabile di scarto (errore assoluto), calcolata simultaneamente per
  ogni metrica
13
14 @variable(mio_modello, s[m in lista_metriche, o in lista_operator] >=
  0)
```

Listing 5.1: Dichiarazione dinamica delle variabili multidimensionali in JuMP

- **Parametro di Rilassamento:** Coerentemente con questa generalizzazione multidimensionale, anche il vincolo strutturale che regola la transizione alla Fase 2 è stato adattato. Il parametro di rilassamento dinamico (δ) viene calcolato moltiplicando la percentuale di tolleranza desiderata per il volume complessivo di tutte le metriche scalate ottenute in Fase 1, definendo così il limite superiore (*upper bound*) applicato alla somma dell'intera matrice degli scarti.

Di seguito si riporta l'estratto del codice JuMP reale che gestisce questa transizione all'interno dello script:

```

1
2 # --- FASE 2: MINIMIZZARE I COSTI CON TOLLERANZA ---
3
4 percentuale_tolleranza = 0.02
5
6 # Delta dinamico basato sul volume totale di TUTTE le metriche scalate
7
8 volume_totale_fase1 = sum(value(mio_modello[:V_TOT][m]) for m in
    lista_metriche)
9 delta = percentuale_tolleranza * volume_totale_fase1
10
11 @constraint(mio_modello, vincolo_V9, sum(s[m, o] for m in
    lista_metriche, o in lista_operatori) <= S_star + delta)
12
13 # Minimizzazione dei costi operativi
14
15 @objective(mio_modello, Min, sum(costi[i] * x[i] for i in 1:
    totale_turni))
16 optimize!(mio_modello)

```

Listing 5.2: Transizione alla Fase 2: applicazione del vincolo di tolleranza congiunto

- **Funzione obiettivo congiunta:** L'approccio lessicografico a due fasi è stato adeguato di conseguenza. La minimizzazione dell'errore strutturale nella Fase 1 viene estesa minimizzando la somma congiunta degli errori su tutte le metriche simultaneamente:

$$\min \sum_{m \in M} \sum_{o \in O} s[m, o] + \epsilon \sum_{i \in I} c_i x_i \quad (5.1)$$

dove il secondo termine introduce una perturbazione controllata ($\epsilon = 10^{-4}$) legata ai costi operativi per garantire la rigorosa rottura della simmetria tra soluzioni equivalenti e il perfetto condizionamento numerico del solutore. Gli impatti computazionali di questa scelta e la relativa analisi di sensitività sul parametro saranno discussi in dettaglio nella sezione 5.2.

5.1.2 Strategie di Pre-processing e Tuning del Solutore HiGHS

L'elaborazione di un bacino così esteso richiede l'adozione di rigorose tecniche di gestione della memoria e di calibrazione del solutore backend (HiGHS):

1. **Parsing sequenziale a basso impatto di memoria:** L'acquisizione dei dati strutturali avviene riga per riga tramite la funzione nativa `eachline()`, evitando la saturazione della RAM e popolando contestualmente dizionari di supporto ($O(1)$) per il rapido reperimento delle capacità dei depositi e della flotta disponibile.
2. **Normalizzazione delle metriche spaziali:** Per prevenire il malcondizionamento della matrice a causa della forte eterogeneità di scala tra variabili discrete (i turni) e grandezze continue ampie (i chilometri), viene applicata una normalizzazione preventiva basata sulla divisione per i volumi complessivi di riferimento (`valore_storico_turni`, `valore_storico_km_in`, ecc.). Tali parametri di *baseline* sono stati estratti direttamente dal file di *output* relativo alla soluzione ottima a 163 turni fornita da M.A.I.O.R. S.r.l. Questa operazione adimensionale riconduce tutte le metriche a frazioni omogenee, stabilizzando i coefficienti della matrice dei vincoli. La logica è gestita dalla seguente funzione ausiliaria:

```

1  # =====
2  # PARAMETRI SOLUZIONE STORICA MAIOR
3  # =====
4  const valore_storico_turni = 163.0
5  const valore_storico_km_in = 3.46249498e7
6  const valore_storico_km_fuori = 2.40848386000000003e6
7  const valore_storico_km_tot = 3.703343366e7
8
9  # =====
10 # FUNZIONE AUSILIARIA PER LE METRICHE (q_i)
11 # =====
12 function calcola_qi(opzione::String, numero_turni::Int, km_in::Vector{
    Float64}, km_fuori::Vector{Float64}, km_tot::Vector{Float64})
13     if opzione == "number_of_blocks"
14         return ones(Float64, numero_turni) ./ valore_storico_turni
15     elseif opzione == "distance_total"
16         return copy(km_tot) ./ valore_storico_km_tot
17     elseif opzione == "distance_in_line"
18         return copy(km_in) ./ valore_storico_km_in
19     elseif opzione == "distance_out_line"
20         return copy(km_fuori) ./ valore_storico_km_fuori
21     else
22         return ones(Float64, numero_turni) ./ valore_storico_turni
23     end
24 end

```

Listing 5.3: Funzione ausiliaria per la normalizzazione parametrica delle metriche basata sui volumi storici

3. **Rottura della simmetria e tolleranza assoluta:** L'impostazione di un *MIP Absolute Gap* pari a 0.05 impone l'arresto immediato non appena la distanza dall'ottimo intero scende sotto la soglia di un singolo turno.

5.1.3 Il «Pool Esteso» e la Baseline di Riferimento

Prima di introdurre i vincoli stringenti di equità ed effettuare le analisi su tre depositi, si è resa necessaria un'operazione preliminare sul bacino esplorativo. Il *pool* originario fornito dall'algoritmo euristico aziendale (composto da 52.212 turni candidati) risultava infatti strutturalmente carente per garantire una copertura ottima. Essendo basato su un'euristica orientata al puro contenimento dei costi, l'algoritmo originario non ha generato al suo interno le combinazioni necessarie a rispettare vincoli di equità su più metriche contemporaneamente.

Per ovviare a questa limitazione e condurre i test in condizioni ideali, il *dataset* di partenza è stato arricchito tramite l'iniezione diretta di 163 turni *ottimi*. Questo nuovo spazio di ricerca ha permesso al modello di convergere verso la *baseline* non vincolata. Tale configurazione, che prevede l'attivazione asimmetrica dei 163 turni (46.6% assegnato al DEP_001, 26.4% al DEP_002 e 27.0% al DEP_003), costituisce il *benchmark* fondamentale ed è stata utilizzata come base di calcolo per tutti i successivi test di ottimizzazione a metrica singola e multi-obiettivo (Paragrafi 5.2 e 5.3).

5.2 Trattabilità Computazionale e Analisi di Sensitività del Parametro ϵ

L'applicazione del modello sull'infrastruttura completa a tre depositi ha permesso di condurre un'analisi approfondita sui limiti algoritmici dell'approccio lessicografico puro. L'architettura originaria prevedeva, nella Fase 1, una funzione obiettivo focalizzata esclusivamente sulla minimizzazione degli scarti percentuali tra gli operatori.

Tuttavia, i test sul bacino esteso da oltre 52.000 turni hanno dimostrato che l'approccio puramente lessicografico ($\epsilon = 0$) risulta computazionalmente intrattabile su scala reale. Il motivo tecnico risiede nell'estrema degenerazione primale e duale del modello matematico: in assenza di un gradiente economico, la funzione obiettivo basata sui soli scarti risulta «piatta». Il solutore, non rilevando alcuna pendenza differenziale tra milioni di combinazioni frazionarie equivalenti, entra in stallo al nodo radice dell'albero di *Branch & Bound*, esaurendo il limite temporale imposto senza innescare le euristiche di ricerca.

5.2.1 La Soluzione: Regularizzazione e Boost Euristico

Per garantire la convergenza senza invalidare la priorità logica della Fase 1, la strategia risolutiva si è basata su due leve congiunte:

1. **Regularizzazione (ϵ):** Introduzione di un termine infinitesimale di costo (ϵ) nella funzione obiettivo della Fase 1 per inclinare leggermente il piano di costo, fornendo

5. Il Modello Esteso: Caso Studio a 3 Depositi e Gestione della Complessità

una direzione al solutore senza alterare la priorità lessicografica degli scarti:

$$\min \sum_{m \in M} \sum_{o \in O} s[m, o] + \epsilon \sum_{i \in I} c_i x_i$$

2. **Boost Euristico:** A differenza del caso prototipale a due depositi, in cui i parametri di *default* risultavano sufficienti, per gestire la complessità della rete estesa si è reso necessario l'innalzamento del parametro `mip_heuristic_effort` del solutore HiGHS a 0.5. Questa impostazione costringe l'algoritmo ad attivare massicciamente euristiche pesanti di arrotondamento e ricerca locale (quali la *Feasibility Pump* e le *Sub-MIP*), rivelatesi indispensabili per sbloccare lo stallo e individuare la prima soluzione intera ammissibile.

5.2.2 Analisi di Sensitività Computazionale

Al fine di calibrare rigorosamente la scelta del parametro, è stata condotta un'analisi di sensitività mirata a individuare il valore minimo di regolarizzazione in grado di rompere la degenerazione, salvaguardando al contempo la purezza teorica della Fase 1. La Tabella 5.1 riassume i risultati dei test computazionali condotti sul bacino esteso.

Parametro (ϵ)	Stato Fase 1	Nodi B&B	Iterazioni LP	Tempo Fase 1	Comportamento Algoritmico
0 (Puro)	<i>Time Limit</i>	0	$\sim 2.45 \times 10^6$	> 3600 s	Stallo per degenerazione. Obiettivo piatto, euristiche inibite.
10^{-5}	<i>Time Limit</i>	0	$\sim 2.27 \times 10^6$	> 3600 s	Gradiente debole. Pendenza insufficiente per innescare le Sub-MIP.
10^{-4}	Ottimo Trovato	1	$\sim 3.58 \times 10^5$	~ 812 s	Sweet Spot. Il gradiente guida l'euristica, rottura della degenerazione.
10^{-3}	<i>Ottimo Trovato</i>	Var.	$\sim 6.00 \times 10^5$	~ 600 s	Convergenza rapida, ma perturbazione teorica meno rigorosa.

Tabella 5.1: Analisi di sensitività sul parametro di regolarizzazione (ϵ) con `mip_heuristic_effort` = 0.5 sul bacino esteso a tre depositi.

5.2.3 Discussione dei Risultati e Individuazione dello *Sweet Spot*

Come si evince dai dati sinottici, impostare un peso nullo ($\epsilon = 0$) o eccessivamente debole ($\epsilon = 10^{-5}$) causa un collasso prestazionale: il solutore esegue oltre 2,2 milioni di iterazioni del rilassamento continuo girando a vuoto al Nodo 0, poiché l'assenza di un gradiente economico inibisce l'attivazione delle euristiche.

Al contrario, la configurazione $\epsilon = 10^{-4}$ rappresenta il perfetto *sweet spot* computazionale. Grazie a questa pendenza infinitesimale, al secondo 812 l'euristica *Sub-MIP* locale riesce a sbloccare lo stallo, individuando la prima soluzione intera e chiudendo la Fase 1 praticamente al primo nodo dell'albero di ricerca con appena 358.000 iterazioni *LP*. Valori superiori ($\epsilon = 10^{-3}$), pur garantendo una convergenza numerica rapida, introdu-

rebbero una distorsione economica eccessiva, compromettendo il rigore della separazione lessicografica tra equità e costi.

5.3 Ottimizzazione a Singola Metrica: La Scoperta della «Rigidità di Rete»

Il primo test sul bacino completo a tre depositi è stato focalizzato sull'ottimizzazione di una singola grandezza operativa, con l'obiettivo di valutare la sensibilità della rete alle forzature percentuali. Come evidenziato nello scenario di *baseline* non vincolato, la propensione geografica naturale dell'infrastruttura distribuisce i turni macchina con proporzioni approssimabili al 47%, 26% e 27%.

In accordo con le richieste aziendali, si è tentato di uniformare le altre metriche operative a questa specifica distribuzione. L'esperimento ha quindi imposto al solutore di utilizzare unicamente la metrica dei chilometri percorsi in servizio passeggeri (`distance_in_line`), costringendo il modello ad allineare il chilometraggio di ciascun deposito ai target naturali dei turni.

La configurazione è stata eseguita imponendo una tolleranza rigida ($\delta = 0.001$) nella transizione alla Fase 2, stringendo severamente il margine di scostamento consentito rispetto all'ottimo di Fase 1.

5.3.1 Risultati Computazionali

Dal punto di vista prestazionale, l'efficacia della normalizzazione spaziale basata sui volumi storici è apparsa subito evidente. La Fase 1 ha individuato lo scarto minimo strutturale normalizzato ($S^* \approx 0.04286$) in appena 131 secondi, mentre la Fase 2 ha completato l'ottimizzazione al primo nodo in 14.14 secondi, per un tempo totale di elaborazione inferiore ai tre minuti.

Confrontando l'output del modello di *Set Partitioning* con la soluzione originaria non vincolata, i dati estratti dimostrano una straordinaria stabilità strutturale:

- **Stabilità della flotta:** Il solutore ha attivato esattamente 163 veicoli (78 al DEP_001, 41 al DEP_002 e 44 al DEP_003), con un costo operativo invariato di 177.867 e garantendo il 100% di copertura del servizio.
- **Distribuzione Chilometrica:** Le quote raggiunte si sono attestate al 46.4% (`Operatore_1`), 24.4% (`Operatore_2`) e 29.1% (`Operatore_3`). Ben 161 *Vehicle Blocks* sono risultati identici all'incastro di base, limitando le variazioni allo spostamento marginale di pochissimi turni.

5.3.2 Analisi e Discussione: La Rigidità Topologica

Da un lato, questo risultato rappresenta una forte validazione dell'implementazione informatica, confermando che i vincoli fisici e strutturali codificati nello script Julia riproducono

fedelmente la logica operativa aziendale. Dall'altro, la minima variazione della soluzione ha permesso di isolare una dinamica infrastrutturale definibile come «rigidità di rete». L'ipotesi analitica è che la lunghezza fisica delle corse commerciali e le distanze topografiche tra i capolinea rappresentino un limite rigido: per forzare il modello a raggiungere l'esatto 27% sui chilometri in linea per il DEP_003, il solutore avrebbe dovuto scambiare massicciamente le percorrenze tra i poli. Tuttavia, la topologia della rete inibisce queste operazioni, poiché assegnare corse lunghe a depositi distanti innesca l'esplosione dei trasferimenti a vuoto (*deadhead*) o la violazione dei vincoli di capacità. Trovandosi vincolato da margini di tolleranza stringenti ($\delta = 0.001$), il modello dimostra una convergenza eccellente, assorbendo perfettamente lo sforzo di equità senza alcuna penalizzazione economica e ancorando la flotta al minimo assoluto di 163 turni.

Questo esperimento ha fornito due chiavi di lettura fondamentali per il prosieguo dello studio:

1. La formulazione algebrica normalizzata è in grado di gestire tolleranze rigidissime senza incorrere in crisi combinatorie, garantendo una stabilità computazionale ideale per l'analisi multi-obiettivo (dinamica che verrà esplorata nel Paragrafo 5.4).
2. Per testare la reale flessibilità e i limiti combinatori del *software*, l'imposizione di proporzioni vicine a quelle naturali evidenzia l'ottimalità del bacino; si rende pertanto necessario valutare scenari con target asimmetrici ed estremi (come dimostrato negli *stress test* del Paragrafo 5.5).

5.4 Ottimizzazione Simultanea Multi-Obiettivo: Scalatura e Trade-Off

Dopo aver analizzato il comportamento della rete su una singola grandezza, lo studio si è concentrato sul nucleo operativo del progetto: l'ottimizzazione congiunta. L'obiettivo è richiedere al modello di bilanciare simultaneamente sia il numero di turni macchina (`number_of_blocks`) sia i chilometri in linea (`distance_in_line`), forzandoli verso i medesimi target di propensione naturale (47%, 26%, 27%).

Tentando di risolvere questo scenario sul *pool* originario aziendale (52.212 turni), si ripresenta il limite strutturale del *software* commerciale privo di colonne idonee a soddisfare vincoli multipli di equità simultanea, restituendo uno stato di *Infeasible*. Di conseguenza, l'analisi multi-obiettivo è stata condotta sul bacino esteso standard introdotto nel Paragrafo 5.1.3, garantendo la piena ammissibilità del problema.

Sfruttando l'architettura generalizzata basata su variabili multidimensionali (Paragrafo 5.2.1) e la normalizzazione spaziale preventiva, l'algoritmo ha gestito con successo la complessità congiunta. L'analisi ha osservato la risposta strutturale della rete al variare della tolleranza δ concessa sull'errore congiunto, mettendo a confronto uno scenario elastico con uno rigidamente vincolato.

5.4.1 Lo Scenario a Tolleranza Elastica ($\delta = 2\%$)

La prima configurazione è stata eseguita imponendo una tolleranza elastica ($\delta = 2\%$) nella transizione alla Fase 2. Dopo aver determinato lo scarto strutturale congiunto minimo in Fase 1 ($S^* \approx 0.0476$), l'algoritmo ha completato l'ottimizzazione in tempi estremamente ridotti, chiudendo la Fase 2 in appena 21.8 secondi.

- **Efficienza Economica e Dimensionale:** Il sistema preserva la flotta minima assoluta di 163 veicoli (77 al DEP_001, 42 al DEP_002 e 44 al DEP_003), fissando il costo operativo a 177.867 a fronte di una copertura integrale del servizio.
- **Fedeltà ai Target:** I turni macchina centrano con precisione millimetrica la distribuzione attesa (47.2%, 25.8%, 27.0%), mentre la percorrenza chilometrica si allinea in modo eccellente (45.8%, 25.0%, 29.1%).

5.4.2 Lo Scenario a Tolleranza Rigida e la Robustezza della Rete

La seconda configurazione ha imposto una tolleranza rigida ($\delta = 0.001$), stringendo severamente il margine di scostamento consentito rispetto all'ottimo di Fase 1. In questo scenario, l'esplorazione dello spazio congiunto richiede un maggiore sforzo computazionale nella Fase 1 (846 secondi per convergere allo scarto $S^* \approx 0.0476$), evidenziando la complessità algebrica del soddisfacimento simultaneo dei vincoli.

Tuttavia, grazie al perfetto condizionamento della matrice normalizzata, la Fase 2 si è conclusa al Nodo 1 in appena 25 secondi, senza alcuna esplorazione combinatoria distruttiva. Il riscontro più rilevante dal punto di vista analitico risiede nell'invarianza della soluzione: stringere la tolleranza al millesimo non altera in alcun modo i parametri economici e dimensionali. Il modello converge esattamente alla medesima configurazione dello scenario elastico (163 veicoli e costo pari a 177.867). Questo comportamento dimostra che l'incastro individuato rappresenta il limite strutturale e geometrico intrinseco della rete: la disposizione dei depositi e la topologia dei collegamenti non ammettono configurazioni fisicamente superiori, sancendo la totale robustezza dell'approccio proposto.

5.4.3 Un'Analisi di Stress con Tolleranza Estrema e Saturazione Temporale

Per vagliare i confini operativi limite dell'architettura multi-obiettivo congiunta, è stata condotta un'ulteriore sessione di calcolo imponendo un vincolo di tolleranza estremamente restrittivo ($\delta \approx 0.0002$) nella transizione alla Fase 2.

A differenza dei casi precedenti in cui la convergenza avveniva rapidamente, questo scenario spinge il solutore a un'esplorazione profonda dello spazio combinatorio, esaurendo il limite temporale impostato di 3600 secondi (1 ora). Nonostante la severità del vincolo, l'architettura dimostra una straordinaria resilienza numerica: l'algoritmo naviga l'albero di *Branch & Bound* esplorando 698 nodi ed eseguendo oltre 3,7 milioni di iterazioni di

5. Il Modello Esteso: Caso Studio a 3 Depositi e Gestione della Complessità

semplesso, giungendo a chiudere l'ottimizzazione con un *Gap* residuo estremamente ridotto del 3.04%.

L'analisi dei valori restituiti evidenzia il compromesso economico-distributivo quando il sistema è forzato a una precisione rigorosa:

- **Dimensionamento della Flotta e Costi:** Per soddisfare contemporaneamente la stretta tolleranza congiunta su entrambe le metriche, il modello attiva una flotta complessiva di 169 veicoli (84 al DEP_001, 43 al DEP_002 e 42 al DEP_003), a fronte di un costo operativo totale pari a 186.999.
- **Precisione di Bilanciamento Congiunto:** Il compromesso raggiunto modula i carichi con accuratezza, centrando sia i target sui turni macchina (49.7%, 25.4% e 24.9%) sia la distribuzione dei chilometri in linea (48.4%, 22.7% e 28.9%).

Questo test stress-computazionale conferma la robustezza dell'approccio a due fasi: anche quando le richieste di equità raggiungono livelli di ristrettezza tali da saturare la finestra temporale di calcolo, il modello non diverge né restituisce soluzioni inammissibili, ma garantisce una transizione controllata verso configurazioni operative pienamente sfruttabili.

5.4.4 Considerazioni Computazionali e Tabella Sinottica

Il comportamento riscontrato nell'intera sperimentazione simultanea dimostra che l'architettura a due fasi generalizzata possiede un'elevatissima robustezza numerica. La normalizzazione adimensionale previene qualsiasi degrado anomalo, consentendo all'analista di governare il *trade-off* tra costi, flotta e precisione di equità.

La Tabella 5.2 sintetizza l'evoluzione dei risultati al variare della severità del vincolo di tolleranza, mettendo in evidenza la stabilità strutturale della rete e l'impatto economico delle configurazioni estreme.

Configurazione	Tolleranza (δ)	Costo Operativo	Flotta	Bilanciamento Turni (%)	Bilanciamento Km (%)	Tempo F1 / F2 (s)	Tempo Tot. (s)
Propensione Naturale	-	177.867	163	47.0 - 26.0 - 27.0	47.0 - 26.0 - 27.0	-	-
Scenario Elastico	2%	177.867	163	47.2 - 25.8 - 27.0	45.8 - 25.0 - 29.1	812.7 / 21.8	834.5
Scenario Rigido	0.001	177.867	163	47.2 - 25.8 - 27.0	45.8 - 25.0 - 29.1	846.0 / 25.0	871.0
Stress Estremo	0.0001	186.999	169	49.7 - 25.4 - 24.9	48.4 - 22.7 - 28.9	846.0 / 3600*	4446

Tabella 5.2: Confronto sinottico dell'ottimizzazione simultanea multi-obiettivo: stabilità, costi e tempi computazionali al variare della tolleranza. (*Raggiungimento del limite temporale impostato.)

5.5 Validazione del Modello: Stress Test e Limiti della Column Generation

A conclusione dell'analisi, per testare la reale flessibilità del sistema logistico, il solutore è stato sottoposto a uno *stress test* totalmente slegato dalla propensione geografica naturale della rete. Il modello è stato forzato a raggiungere un target fortemente asimmetrico basato sul conteggio dei turni macchina (`number_of_blocks`): 55% per il DEP_001, 20% per il DEP_002 e 25% per il DEP_003.

5.5.1 Fallimento sul Bacino Originale (Infeasible)

Il primo test è stato eseguito elaborando il *pool* puro di partenza fornito dall'azienda (52.212 turni). Come ampiamente prevedibile alla luce delle analisi precedenti, sottoposto alla forzatura di target asimmetrici estremi, il solutore si arresta già durante la Fase 1, restituendo lo stato di *Infeasible*.

Questo esito conferma il limite strutturale del bacino di partenza: nel file di partenza mancano materialmente i turni necessari per forzare la rete contro la sua reale geografia. Diventa quindi evidente che non si può fare a meno di estendere lo spazio di ricerca.

5.5.2 Successo sul Bacino Esteso «Ad Hoc» e Validazione Finale

Per sbloccare il sistema e dimostrare la validità dell'architettura in Julia, è stata utilizzata una seconda e specifica estensione dello spazio di ricerca. Il bacino originale è stato ripreso e arricchito tramite l'iniezione di 163 nuovi turni forniti direttamente dall'azienda (M.A.I.O.R. S.r.l.).

Questo set di colonne, generato *ad hoc* scavalcando l'euristica standard, è stato inserito per fornire al solutore il materiale algebrico necessario ad assecondare la forte asimmetria richiesta (55-20-25). Replicando il test su questo bacino esteso (con tolleranza rigida $\delta = 0.001$), l'algoritmo converge rapidamente verso una soluzione ottima e ammissibile. I parametri di output dimostrano l'assoluto successo dell'operazione:

- **Tempi di esecuzione:** La convergenza risulta estremamente rapida. La Fase 1 individua l'ottimo globale al nodo radice in 51.1 secondi, con un errore strutturale minimo ridotto a $S^* \approx 0.008$. La Fase 2 completa l'ottimizzazione in 27.1 secondi.
- **Flotta Attivata e Costi:** Il modello preserva l'efficienza assoluta garantendo la copertura del servizio al 100% con la flotta ottima minima di 163 veicoli, a fronte di un costo operativo pari a 179.995.
- **Bilanciamento Raggiunto:** I turni vengono plasmati sui target asimmetrici imposti:
 - DEP_001: 89 turni (pari al 54.6%, contro il target del 55%).
 - DEP_002: 33 turni (pari al 20.2%, contro il target del 20%).

- DEP.003: 41 turni (pari al 25.2%, contro il target del 25%).

5.5.3 Discussione Critica e Diagnosi dei Limiti dell'Approccio *Ex Post*

L'analisi condotta attraverso gli stress test su distribuzioni asimmetrico-estreme (55%-20%-25%) offre una chiave di lettura fondamentale per comprendere i confini operativi dell'intero lavoro. Il riscontro dello stato di *Infeasible* sul bacino originario non deve essere interpretato come un limite computazionale del solutore, bensì come una diagnosi strutturale della natura stessa dell'approccio *ex post*. Poiché l'algoritmo euristico aziendale opera a monte con il solo obiettivo di minimizzare i costi globali, esso riduce e seleziona lo spazio di ricerca escludendo a priori proprio quelle combinazioni geometriche che risulterebbero necessarie per sostenere un forte sbilanciamento tra gli operatori.

Tuttavia, quando il modello di *Set Partitioning* viene alimentato con un bacino arricchito di colonne *ad hoc*, la formulazione algebrica dimostra la sua piena validità. Il sistema dimostra di saper ricombinare i turni alla perfezione, centrando i target asimmetrici e mantenendo intatta la flotta minima a quota 163 veicoli. Tale evidenza conferma che il vincolo restrittivo non risiede nell'efficacia del modello di ottimizzazione, ma nella povertà strutturale del *pool* iniziale generato in modo isolato.

In quest'ottica, il prototipo sviluppato si trasforma da semplice strumento di calcolo a un vero e proprio ambiente diagnostico per l'azienda:

- **Quantificare il prezzo dell'equità:** Il modello consente di misurare con precisione il *trade-off* economico tra il contenimento dei costi e il rispetto delle quote percentuali, quantificando l'assorbimento addizionale di risorse richiesto per alterare la distribuzione del servizio.
- **Isolare le rigidità topologiche:** I risultati dimostrano che molte disparità operative non derivano da inefficienze procedurali, ma dalla conformazione fisica della rete e dalla dislocazione geografica dei depositi.
- **Comprendere i limiti dell'approccio *ex post*:** Emerge con chiarezza che un'ottimizzazione condotta unicamente a valle risulta penalizzata se i dati in partenza ignorano le logiche di bilanciamento, giustificando la necessità di integrare i vincoli direttamente nei generatori di colonne (*Pricing Problem*) nelle future evoluzioni architetturali.

In definitiva, questo lavoro non si limita a risolvere istanze di test isolate, ma consegna a MAIOR un ambiente di simulazione e stress test solido e affidabile, ponendosi come banco di prova ideale per valutare la resilienza sui bacini reali e tracciare la rotta verso la futura *suite* logistica.

5.6 Benchmark Industriale su Larga Scala: HiGHS vs. Gurobi

A completamento dell'analisi condotta nel Capitolo 4 sull'istanza ridotta, il confronto prestazionale tra il solutore *open-source* HiGHS (1.15.0) e il motore commerciale Gurobi (13.0) è stato esteso alla rete completa a tre depositi. L'obiettivo è quantificare il divario computazionale su un'istanza reale di grandi dimensioni (oltre 52.000 turni candidati), caratterizzata da forte degenerazione e multi-obiettività.

I test sono stati condotti applicando a Gurobi i medesimi parametri di arresto (*Time Limit* di 900 secondi per la Fase 1 e *MIP Gap* dell'1%) e le medesime leve architetturali sviluppate per HiGHS. Al fine di garantire un confronto rigoroso, il modello su Gurobi ha mantenuto la medesima perturbazione algoritmica sui costi ($\epsilon = 10^{-4}$) per rompere lo stallo da degenerazione primale, unita al forzamento esplicito di tecniche euristiche avanzate come la *Feasibility Pump* e *RINS*:

```

1 # FASE 1: Obiettivo regolarizzato per rompere la degenerazione (epsilon = 10^-4)
2 @objective(mio_modello, Min, scarti_totali + (0.0001 * costi_totali))
3
4 # Inizializzazione del modello con tuning euristico aggressivo per Gurobi
5 mio_modello = Model(optimizer_with_attributes(Gurobi.Optimizer,
6     "TimeLimit" => 900.0,
7     "MIPGap" => 0.01,
8     "MIPFocus" => 1,
9     "Heuristics" => 0.5,
10    "PumpPasses" => 10, # Forza i passaggi della Feasibility Pump
11    "RINS" => 10       # Applica RINS ogni 10 nodi
12 ))

```

La Tabella 5.3 riassume i tempi di esecuzione, i *gap* e i costi finali sui due scenari più complessi analizzati nel presente capitolo: l'ottimizzazione simultanea rigida ($\delta = 0.001\%$) e lo *stress test* asimmetrico sul bacino esteso *ad hoc* (55%-20%-25%).

Scenario (3 Depositi)	Solutore	Tempo F1 (s)	Gap F1 (%)	Tempo F2 (s)	Costo Finale
Simultaneo Rigido (47%-26%-27%)	Gurobi	900.8*	52.3	5.4	177.867
	HiGHS	846.9*	71.4	25.0	177.867
Stress Test Asimmetrico (55%-20%-25%)	Gurobi	4.3	0.0	3.2	179.995
	HiGHS	48.0	0.0	30.0	179.995

Tabella 5.3: Confronto prestazionale su larga scala (oltre 52.000 turni) tra Gurobi 13.0 e HiGHS 1.15.0. (*Raggiungimento del limite temporale impostato a 15 minuti.)

5.6.1 Considerazioni Operative e la Viabilità Pratica dell'Open-Source

I dati emersi dal confronto su larga scala ristabiliscono le gerarchie computazionali attese tra un motore commerciale *state-of-the-art* e un solutore *open-source*, ma offrono al contempo un responso industriale estremamente positivo per HiGHS.

5. Il Modello Esteso: Caso Studio a 3 Depositi e Gestione della Complessità

Analizzando lo scenario di stress asimmetrico (55%-20%-25%), Gurobi dimostra una netta superiorità di potenza bruta e di potatura dell'albero di *Branch & Bound*: il solutore chiude la Fase 1 in appena 4.33 secondi e completa l'ottimizzazione in Fase 2 in ulteriori 3.17 secondi. La controparte *open-source*, pur difendendosi egregiamente e arrivando a certificare l'ottimo, richiede tempi fisiologicamente maggiori (circa 48 secondi per la prima fase e 30 per la seconda).

Nello scenario simultaneo a tolleranza rigida, l'estrema degenerazione primale porta entrambi i solutori a esaurire il *Time Limit* di 900 secondi imposto nella Fase 1. Tuttavia, a parità di tempo limite, il solutore commerciale sfrutta la propria efficacia euristica per individuare un *Upper Bound* nettamente migliore, riducendo il *MIP Gap* residuo al 52.3% (contro il 71.4% registrato da HiGHS). Questa robustezza algoritmica si riflette nella successiva Fase 2, che viene risolta da Gurobi in soli 5.39 secondi, contro i 25 secondi necessari al motore *open-source*.

Tuttavia, nonostante l'evidente e prevedibile scarto prestazionale in termini di velocità pura sui *bound* intermedi, il dato logistico di maggior rilievo ai fini industriali risiede nell'assoluta equivalenza della soluzione finale. In entrambi gli scenari, HiGHS perviene esattamente alla medesima configurazione ottima: un costo operativo pari a 177.867 per il bilanciamento naturale e 179.995 nello stress test asimmetrico.

Questa evidenza permette di concludere che, ai fini pratici aziendali, HiGHS si comporta in modo del tutto equivalente a Gurobi se il parametro decisionale primario è la qualità del risultato al termine della Fase 2. Pur non potendo eguagliare la rapidità di calcolo del software proprietario, l'approccio *open-source* certifica le medesime soluzioni ottimali in tempi — nell'ordine dei pochi minuti — ampiamente compatibili con i cicli di pianificazione logistica *offline*, confermandosi una validissima alternativa strategica per l'industria.

Conclusioni e Sviluppi Futuri

6.1 Sintesi del Lavoro Svolto

Questo lavoro di tesi è nato dall'esigenza concreta, emersa in recenti contesti operativi di M.A.I.O.R. S.r.l., di estendere le logiche tradizionali del *Vehicle Scheduling Problem*. Superando la pura minimizzazione dei costi, l'obiettivo è stato quello di sviluppare un modello matematico in grado di garantire una ripartizione equa dei carichi di lavoro tra diversi poli logistici, bilanciando simultaneamente metriche eterogenee come il numero di veicoli, i chilometri in linea e i trasferimenti a vuoto.

Per affrontare questa sfida, è stato progettato e implementato un *framework* algoritmico in linguaggio Julia, interfacciato tramite JuMP con il solutore HiGHS. L'architettura sviluppata si basa su un modello di *Set Partitioning* risolto tramite un approccio lessicografico a due fasi: una prima fase dedicata alla minimizzazione dello scarto strutturale rispetto ai target aziendali, e una seconda fase volta a ripristinare l'efficienza economica entro un margine di tolleranza dinamico.

6.2 Risultati Conseguiti

La sperimentazione condotta su dataset reali di complessità crescente ha dimostrato la validità e la robustezza dell'approccio. Sul caso di studio iniziale a due depositi, il modello ha bilanciato efficacemente le metriche, evidenziando fin da subito la dipendenza dei costi dalla propensione geografica naturale della rete.

Estendendo l'analisi all'intera rete a tre depositi, l'ottimizzazione multi-obiettivo ha permesso di quantificare matematicamente il *trade-off* tra risparmio economico ed equità logistica. I dati dimostrano che l'architettura normalizzata implementata garantisce un'eccezionale stabilità: l'imposizione di una tolleranza rigidamente serrata non comporta alcun degrado economico o dimensionale, preservando la flotta minima a quota 163 veicoli e confermando la totale robustezza numerica del sistema.

L'analisi comparativa su larga scala condotta tra il solutore commerciale Gurobi e l'alternativa *open-source* HiGHS ha permesso di validare la viabilità pratica di soluzioni

non proprietarie in contesti industriali. Pur confermando la prevedibile superiorità computazionale del motore commerciale — in grado di esplorare l'albero di ricerca e garantire *bound* migliori in tempi più rapidi, anche grazie al forzamento di euristiche avanzate come *Feasibility Pump* e *RINS* — HiGHS ha dimostrato una notevole solidità operativa. Su istanze complesse e fortemente deformate da una spiccata degenerazione, il solutore *open-source* è infatti riuscito a convergere esattamente alla medesima configurazione ottima finale. Questo risultato offre all'azienda un'alternativa non solo economicamente vantaggiosa, ma tecnicamente pienamente adeguata ai tempi richiesti dai cicli di pianificazione logistica *offline*.

Infine, l'analisi condotta sulla rete a tre depositi e confermata dai successivi stress test ha messo in luce i limiti strutturali dell'approccio *ex post*. Già in fase di baseline, infatti, si è evidenziato come il bacino esplorativo originario, generato da un'euristica orientata al solo abbattimento dei costi, non possieda nativamente le combinazioni geometriche necessarie per gestire vincoli di equità multi-metrica, rendendo indispensabile l'iniezione di nuovi turni per garantire l'ammissibilità. L'aver individuato e gestito questo confine strutturale rappresenta un successo diagnostico fondamentale, in quanto fornisce all'azienda la prova matematica necessaria per orientare gli imminenti sviluppi tecnologici.

6.3 Sviluppi Futuri

Il prototipo sviluppato in questa tesi non chiude un percorso, ma costituisce il fondamento esplorativo per la realizzazione di un modulo commerciale completo, previsto nella *roadmap* aziendale di Hitachi Rail per l'autunno del 2026. Alla luce delle evidenze emerse, si delineano due direttrici principali per gli sviluppi futuri:

1. **Integrazione nel Pricing Problem (*Column Generation*):** Sarà necessario integrare i vincoli di bilanciamento direttamente a monte, all'interno dell'algoritmo di *Column Generation*. Modificando la funzione di costo ridotto nel sottoproblema (*Pricing*), il solutore potrà generare nativamente turni sub-ottimali dal punto di vista economico, ma geometricamente perfetti per soddisfare le quote di equità richieste, eliminando in toto la necessità di iniezioni esterne di colonne.
2. **Sviluppo dell'Interfaccia Industriale (UI):** L'attuale *script* in Julia, concepito come solido motore algoritmico di *backend*, dovrà evolversi in un vero e proprio *Decision Support System* integrato nell'ecosistema *software* di M.A.I.O.R.. Tramite un'apposita interfaccia utente, gli analisti logistici potranno selezionare dinamicamente le metriche e calibrare le soglie di tolleranza interattivamente, esplorando il compromesso tra equità e costi direttamente sul campo.

In conclusione, questo lavoro ha dimostrato che bilanciare equamente i carichi operativi nel TPL è un obiettivo matematicamente e operativamente fattibile. Il modello sviluppato consegna quindi all'azienda una solida base algoritmica e una diagnosi chiara da cui partire per dominare i futuri scenari multi-operatore.

Bibliografia

- [1] Ribeiro, C. C., & Soumis, F. (1994). *A column generation approach to the multiple-depot vehicle scheduling problem*. Operations Research, 42(1), 41–52.
- [2] Löbel, A. (1998). *Vehicle scheduling in public transit and lagrangean pricing*. Management Science, 44(12-part-2), 1637–1648.
- [3] Hillier, F. S., & Lieberman, G. J. (2015). *Introduction to Operations Research* (10th ed.). McGraw-Hill Education.
- [4] Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). *Julia: A fresh approach to numerical computing*. SIAM Review, 59(1), 65–98.
- [5] Dunning, I., Huchette, J., & Lubin, M. (2017). *JuMP: A modeling language for mathematical optimization*. SIAM Review, 59(2), 295–320.
- [6] Huangfu, Q., & Hall, J. A. (2018). *Parallelizing the dual revised simplex method*. Mathematical Programming Computation, 10(1), 119–142.
- [7] Gurobi Optimization, LLC (2026). *Gurobi Optimizer Reference Manual*. Disponibile online: <https://www.gurobi.com>