TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Car Sharing GoFun

Sviluppo di una simulazione del servizio di car sharing e proposta di un modello ottimizzato per massimizzare il guadagno atteso

Andrea Andreoli *0350012*

Massimo Buniy *0350022*

Lorenzo Grande *0350212*

Roma, gennaio 2025

Abstract

Lo studio si concentra sulla modellazione e simulazione di un sistema di car sharing GoFun, applicando i principi della teoria delle code, con l'obiettivo di determinare il numero ottimale di veicoli disponibili all'interno del servizio, ovvero il numero di veicoli messi a disposizione degli utenti, per massimizzare il profitto dell'azienda.

Indice

1

Introduzione

I sistemi di car sharing stanno emergendo come una soluzione innovativa e pratica per affrontare le sfide della mobilità urbana moderna. Offrendo un'alternativa al possesso di un'auto privata, questi servizi permettono agli utenti di accedere a veicoli solo quando necessario, eliminando le responsabilità e i costi associati alla proprietà di un'auto. Grazie alla loro flessibilità, i sistemi di car sharing possono contribuire a ridurre il numero complessivo di veicoli in circolazione, alleviando così la congestione del traffico e diminuendo l'inquinamento atmosferico. Inoltre, possono ottimizzare l'uso degli spazi di parcheggio e delle risorse territoriali, migliorando la vivibilità e la sostenibilità urbana.

In questo studio, si tratta il caso GoFun, una società di car sharing fondata in Cina, che offre un servizio di noleggio auto a breve termine. Attraverso la sua app mobile, GoFun permette agli utenti di prenotare, sbloccare e utilizzare veicoli, pagando solo per il tempo di utilizzo e la distanza percorsa. Un aspetto distintivo di GoFun è il suo impegno per la sostenibilità ambientale, con una flotta composta interamente da auto elettriche per minimizzare le emissioni di CO2. Utilizzando la teoria delle code e la simulazione, il nostro studio analizza il modello operativo di GoFun, esplorando l'efficacia delle sue strategie di gestione e l'impatto ambientale del servizio.

2

Oggetto di studio

Prima di descrivere nel dettaglio l'oggetto di studio, è fondamentale chiarire il funzionamento generale del sistema. Si considera un sistema di car sharing composto complessivamente da T veicoli e N centri, ciascuna con una propria capacità di parcheggio K_i e una capacità di ricarica J_i, dove i indica un centro appartenente all'insieme dei centri N. Ogni centro opera in modo autonomo, pur appartenendo alla stessa rete di car sharing.

Nel sistema, oltre ai veicoli, sono presenti gli utenti, ossia coloro che noleggiano le automobili. Per avviare un noleggio, l'utente deve recarsi fisicamente presso un centro i per prelevare un'automobile disponibile. Al termine del noleggio, l'utente può restituire l'auto presso lo stesso centro di partenza o presso un altro centro della stessa rete di car sharing, compatibilmente con le capacità di parcheggio e di ricarica dei diversi centri.

In questo studio ci si è concentrati su un singolo centro, trattandolo come un'entità autonoma dal punto di vista gestionale. In particolare, si sono analizzati i profitti generati da questo centro in funzione dei costi operativi e dei ricavi derivanti dall'attività di noleggio delle automobili.

I costi operativi del centro sono composti da:

- Il costo di manutenzione delle automobili, che include interventi periodici e straordinari necessari per mantenere i veicoli in condizioni ottimali.
- Il costo per mantenere operativo il parcheggio, che dipende dalla gestione dello spazio, dal personale e dalle spese amministrative.
- Il costo di ricarica dei veicoli, relativo all'energia elettrica utilizzata per ricaricare le automobili elettriche disponibili nel centro.

Per quanto riguarda i ricavi, essi derivano principalmente da due fonti:

- Il profitto generato dal noleggio di un'automobile, calcolato come il prezzo pagato dall'utente per il servizio di noleggio, in base alla durata o alla distanza percorsa.
- Il profitto generato dai chilometri percorsi dalle automobili noleggiate, che può includere tariffe aggiuntive per l'utilizzo prolungato del veicolo.

3

Obiettivo dell'analisi

L'obiettivo principale di questo studio è massimizzare i profitti dell'azienda attraverso una gestione ottimale delle automobili e delle infrastrutture di supporto. In particolare, si intende determinare il numero ideale di veicoli da includere, al fine di soddisfare la domanda mantenendo i costi sotto controllo. Parallelamente, è necessario ottimizzare la capacità dei parcheggi e il numero delle postazioni di ricarica in ciascun centro di servizio, per garantire un'efficiente operatività delle auto a disposizione.

4

Modello concettuale

Quando si parla di modello concettuale ci si sta riferendo ad una rappresentazione astratta del sistema utilizzata per analizzare e comprendere il comportamento dei processi al suo interno.

Il servizio di car sharing GoFun è stato modellato come in figura 4.1.

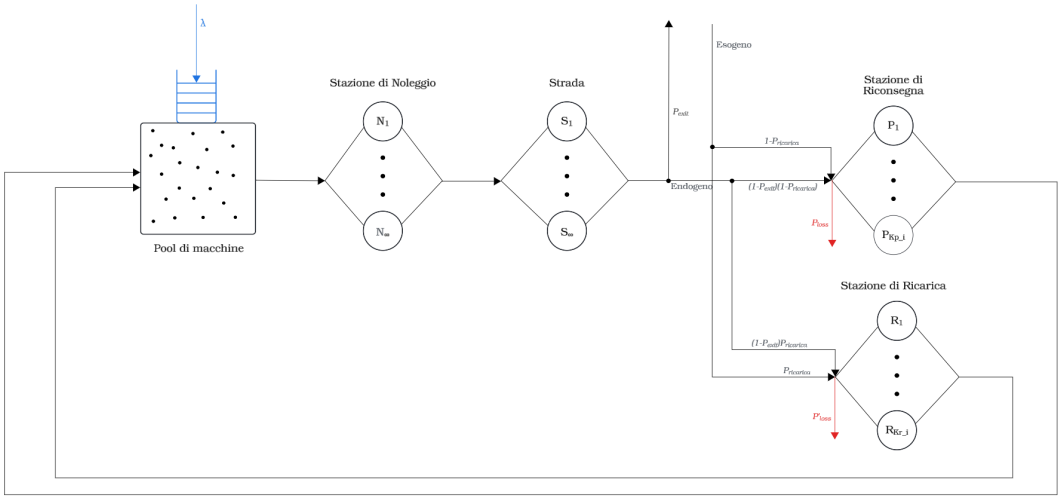


Figura 4.1: *Modello concettuale del servizio di car sharing*

Le figure principali di questo sistema sono:

- **Utente**: colui che desidera noleggiare un'auto.
- **Auto**: il veicolo che può essere noleggiato da un utente nel caso in cui sia disponibile, ovvero non sia già noleggiato da un altro utente.

In particolare si ha:

- **Stazione di Noleggio**: in quest'area viene resa disponibile l'operazione di noleggio di un'auto per un nuovo utente.
- **Strada**: i centri Strada servono a gestire il tempo in cui un'automobile viene noleggiata, garantendo così un guadagno effettivo per l'azienda.

- **Stazione di Riconsegna**: in quest'area è disponibile l'operazione di riconsegna dell'auto noleggiata. L'auto verrà parcheggiata e solo successivamente verrà resa nuovamente disponibile per il noleggio. Nel mentre l'auto attende di venire noleggiata, questa resta all'interno del parcheggio lasciandolo in uno stato occupato.
- **Stazione di Ricarica**: nel caso in cui l'auto restituita abbia la batteria scarica, non verrà inviata direttamente alla Stazione di Riconsegna, ma sarà parcheggiata presso una stazione di ricarica. Una volta completata la ricarica, tornerà disponibile per il noleggio. È importante notare che non tutti i posti auto del noleggio dispongono di una colonnina di ricarica, quindi la capacità di questi sarà inferiore rispetto ai normali posti auto.

Nel caso di studio viene associato al servizio GoFun il concetto di *sistema*, all'interno del quale è possibile identificare vari centri, corrispondenti alle diverse situazioni che si possono verificare durante il noleggio dell'auto.

Nel *sistema* qui descritto, il *job* è definito come la coppia composta dall'utente che desidera noleggiare l'auto e l'auto stessa. Si utilizza la definizione di *job* come coppia poiché è possibile che ci siano auto senza utenti interessati al noleggio, o utenti in coda in attesa che un'auto si liberi.

In ogni istante, un utente che desidera noleggiare un'auto può trovarsi in coda oppure essere accoppiato ad un'auto, rappresentando un servizio 'virtuale'. Inoltre, poiché possiamo associare al concetto di servente la persona che noleggia l'auto, in un dato momento un servente (utente che entra nel sistema) potrà fornire servizio a una sola auto tra quelle disponibili.

4.1 Stato del sistema

Ad ogni istante, lo stato del sistema è determinato da:

- Il numero di utenti in coda in attesa di noleggiare un'auto (λ).
- Il numero di auto disponibili per il noleggio (rappresentato dal pool di macchine nella figura 4.1).
- Il numero di auto attualmente in fase di noleggio da parte degli utenti (rappresentato dai server della Stazione di Noleggio nella figura 4.1).
- Il numero di auto in circolazione sulla strada (rappresentato dai server della Strada nella figura 4.1).
- Il numero di auto in fase di ricarica presso il centro di noleggio (rappresentato dai server della Stazione di Ricarica nella figura 4.1).
- Il numero di auto in procedura di riconsegna presso il centro di noleggio (rappresentato dai server della Stazione di Riconsegna nella figura 4.1).

4.2 Descrizione degli eventi

Come già enunciato, per sistema, si intende l'insieme dei centri presenti nella figura 4.1.

Per il **sistema**:

- Un evento di riferimento è l'accoppiamento tra un utente che desidera noleggiare un'auto e la disponibilità di un'auto presso la Stazione di Noleggio (λ).
- È possibile che si verifichi un evento di uscita dal sistema con probabilità P_{exit}, che rappresenta la situazione in cui un utente decide di dirigersi verso un'altra stazione. Inoltre, un utente può abbandonare la stazione con probabilità P_{loss} nel caso in cui non trovi un posto libero per accedere alla Stazione di Riconsegna. Un evento simile può verificarsi con probabilità P'_{loss} nel caso della Stazione di Ricarica.
- Un evento di arrivo *esogeno* rappresenta l'arrivo di auto noleggiate presso il parcheggio della stazione o presso la stazione di ricarica, provenienti da una stazione di noleggio esterna rispetto a quella analizzata nello studio.

Per il centro **Stazione di Noleggio**, si considerano:

- Due diversi eventi di arrivo:
  - Il primo coincide con l'arrivo di un utente che desidera noleggiare un'auto.
  - Il secondo indica un'auto tornata disponibile per il noleggio dopo essere stata parcheggiata o dopo aver ricaricato la propria batteria.
- Un evento di uscita si verifica quando un'auto viene noleggiata da un utente, ossia, l'utente ha completato il servizio "virtuale" associato a una delle auto disponibili per il noleggio.

Per il centro **Strada**, si considerano:

- Un evento di arrivo al centro coincide con l'uscita di un job dalla *Stazione di Noleggio*, ovvero quando l'utente ha completato la procedura di noleggio e si trova sulla strada con l'auto noleggiata.
- Un evento di uscita si verifica quando il job è stato completato, ossia quando l'utente decide di terminare il noleggio e si accinge a restituire l'auto presso una stazione di noleggio.

Per il centro **Stazione di Riconsegna**, si considerano:

- Un evento di arrivo al centro coincide con l'uscita del job dal centro *Strada*, ovvero quando l'utente ha terminato il noleggio e decide di parcheggiare l'auto presso la stazione. In particolare, l'evento di arrivo *endogeno* rappresenta l'arrivo delle auto noleggiate al parcheggio della stazione o alla stazione di ricarica, provenienti dalla stessa stazione considerata in questo studio.

- Un evento di uscita si verifica quando il job è stato completato, cioè quando l'utente ha concluso la manovra di parcheggio, rendendo l'auto nuovamente disponibile per il noleggio.

Per il centro **Stazione di Ricarica**, si considerano:

- Un evento di arrivo al centro coincide con l'uscita del job dal centro *Strada*, ovvero quando l'utente ha terminato il noleggio dell'auto, ma il veicolo necessita di una ricarica e viene quindi parcheggiato presso la stazione di ricarica.
- Un evento di uscita si verifica quando il job è stato completato, cioè l'utente ha concluso la manovra di parcheggio e il processo di ricarica della batteria è stato portato a termine, rendendo l'auto nuovamente disponibile per il noleggio.

5

Modello delle specifiche

Con l'introduzione del modello delle specifiche si fa riferimento a una rappresentazione formale del sistema descritto in modo astratto nel capitolo 4.

Questo tipo di modellazione permette di effettuare un'analisi matematica del comportamento del sistema, individuando valori come il tempo di attesa, il tempo di risposta e altre metriche utili per valutare le prestazioni del sistema.

La coda di arrivi λ (utenti che desiderano noleggiare un'auto) presente nel sistema è di tipo **FIFO** (First In First Out) con **scheduling astratto senza prelazione**, quindi, nessun utente può essere interrotto durante la richiesta di noleggio dell'auto.

Inoltre si tiene conto del principio di *memoryless* secondo cui gli arrivi futuri non sono influenzati dall'andamento degli arrivi avvenuti in passato. Gli arrivi sono *indipendenti* tra loro.

5.1 Equazioni di traffico

A seguito della risoluzione della rete di figura 4.1 sono state formulate le seguenti equazioni di traffico:

$$\begin{cases} \lambda_N = \lambda + \lambda_P + \lambda_R \\ \lambda_S = \lambda_N \\ \lambda_P = (\lambda_E(1 - P_{\text{ricarica}}) + \lambda_S(1 - P_{\text{exit}})(1 - P_{\text{ricarica}}))\,(1 - P_{\text{loss}}) \\ \lambda_R = (\lambda_E P_{\text{ricarica}} + \lambda_S(1 - P_{\text{exit}})P_{\text{ricarica}})\,(1 - P'_{\text{loss}}) \end{cases} \tag{5.1}$$

5.2 Matrice di routing

La matrice di routing relativa alla rete di figura 4.1 è presente in tabella 5.1.
I componenti della matrice di routing sono stati scelti come segue:

- P_{ricarica} è la probabilità che, al termine del periodo di noleggio, il veicolo abbia una percentuale di carica inferiore al livello minimo di soglia, risultando quindi scarico.

	Esterno	Stazione di Noleggio	Strada	Stazione di Riconsegna	Stazione di Ricarica
Esterno	0	1	0	$(1 - P_{\text{ricarica}})(1 - P_{\text{loss}})$	$P_{\text{ricarica}}(1 - P'_{\text{loss}})$
Stazione di Noleggio	0	0	1	0	0
Strada	P_{exit}	0	0	$(1 - P_{\text{exit}})(1 - P_{\text{ricarica}})(1 - P_{\text{loss}})$	$(1 - P_{\text{exit}})P_{\text{ricarica}}(1 - P'_{\text{loss}})$
Stazione di Riconsegna	0	1	0	0	0
Stazione di ricarica	0	1	0	0	0

Tabella 5.1: *Matrice di routing*

- La variabile P_{exit} rappresenta la probabilità che un veicolo noleggiato non faccia ritorno al centro da cui è partito. Ciò può dipendere sia dalla saturazione del centro di partenza, sia dalla convenienza, durante il tragitto, di scegliere un centro di destinazione diverso.
- Le probabilità P_{loss} e P'_{loss} rappresentano rispettivamente le probabilità di perdita per la Stazione di Riconsegna e per la Stazione di Ricarica nel caso in cui tutti i server siano occupati.

5.3 Modellazione centri

Stazione di Noleggio $M/M/\infty$

La Stazione di Noleggio è modellata secondo un sistema $M/M/\infty$, caratterizzato da un numero infinito di server e da una coda di arrivi infinita associata a un pool di auto. In particolare:

- **Server multipli:** la Stazione di Noleggio dispone di server multipli, il che significa diversi utenti possono noleggiare contemporaneamente diverse macchine.
- **Coda di arrivi + pool di macchine**: esiste una coda di arrivi, ossia l'arrivo di una persona desiderosa di noleggiare un'auto. Nonostante la presenza di server infiniti (che teoricamente non dovrebbe comportare attese), l'utente potrà trovarsi comunque in coda in base alla disponibilità delle auto nel pool. Se un'auto è disponibile, verrà associata all'utente, che potrà procedere con il noleggio. Altrimenti, dovrà attendere in coda fino a quando una macchina non sarà disponibile per il noleggio.

Strada $M/M/\infty$

La Strada è modellata secondo un sistema $M/M/\infty$, caratterizzato da un numero infinito di server e assenza di code. In particolare:

- **Server multipli:** il sistema stradale è modellato con server multipli, poiché, una volta noleggiato, un veicolo può circolare indipendentemente dagli altri.
- **Assenza di code:** una volta che un utente ha noleggiato un veicolo, inizia a pagare il servizio per una durata pari a $\frac{1}{\mu_S}$. Non c'è quindi alcuna attesa prima di iniziare a usufruire del servizio, motivo per cui la coda non è rappresentata.

Nel sistema stradale, il tempo di servizio è pari a $\frac{1}{\mu_S}$, il quale rappresenta il tempo medio di utilizzo di un veicolo da parte di un utente. Questo tempo include anche la pro-

babilità di incontrare rallentamenti dovuti al traffico durante il tragitto per riportare il veicolo a una stazione.

Stazione di Riconsegna $M/M/K_{p_i}$

La Stazione di Riconsegna è modellata secondo un sistema $M/M/K_{p_i}$, caratterizzato da un numero K_{p_i} di server e assenza di code. In particolare:

- **Server multipli:** la Stazione di Riconsegna dispone di K_{p_i} posti auto, il che significa che possono essere parcheggiati fino a K_{p_i} veicoli contemporaneamente.
- **Assenza di code:** non c'è alcuna attesa prima di parcheggiare l'auto, poiché l'utente che deve restituirla può trovare un parcheggio libero all'interno della stesso centro in cui ha noleggiato il veicolo, oppure può cercare un'altra centro GoFun dove parcheggiare l'auto.

Il tempo di servizio nella Stazione di Riconsegna è pari a $\frac{1}{\mu_P}$ e tiene conto del tempo medio necessario per parcheggiare un veicolo. Quando un veicolo entra in servizio, si intende che è entrato nella Stazione di Riconsegna; solo al termine del servizio si può affermare che il veicolo ha trovato un posto e ha parcheggiato.

Stazione di Ricarica $M/M/K_{r_i}$

La Stazione di Ricarica è modellata secondo un sistema $M/M/K_{r_i}$, caratterizzato da un numero K_{r_i} di server e assenza di code. In particolare:

- **Server multipli:** la Stazione di Ricarica dispone di K_{r_i} posti auto con possibilità di ricarica, il che significa che possono essere parcheggiati fino a K_{r_i} veicoli contemporaneamente per la ricarica.
- **Assenza di code:** non c'è alcuna attesa prima di parcheggiare l'auto per effettuare la ricarica della batteria, poiché l'utente che deve restituirla può trovare un parcheggio con annessa colonnina di ricarica libera all'interno dello stesso centro in cui ha noleggiato il veicolo, oppure può cercare un altro centro GoFun dove parcheggiare l'auto e metterla in carica.

Il tempo di servizio nella Stazione di Ricarica è pari a $\frac{1}{\mu_R}$ e comprende il tempo medio necessario per parcheggiare un veicolo e il tempo medio richiesto per completarne la ricarica. Un veicolo entra in servizio nel momento in cui viene parcheggiato nella Stazione di Ricarica, e il servizio si considera concluso solo quando il veicolo è completamente ricaricato ed è pronto per essere noleggiato nuovamente.

Nota

Come precedentemente visto nella spiegazione della Stazione di Riconsegna, Stazione di Ricarica e Stazione di Noleggio, questi tre modelli rappresentano, in realtà, lo stesso centro, ma in tre situazioni differenti. Infatti, a causa della complessità del sistema, è necessario suddividere il centro in tre parti più semplici:

- La **Stazione di Riconsegna** rappresenta la possibilità per un utente di poter riconsegnare l'auto noleggiata. Qualora non fosse possibile l'utente dovrà recarsi presso un altro centro (comporta una diminuzione della soddisfazione del cliente).
- La **Stazione di Ricarica** modella la stessa situazione, ma nel caso in cui l'auto precedentemente noleggiata risulti scarica e necessiti quindi di un tempo di ricarica. Per semplicità, si assume che un'auto vada alla Stazione di Ricarica quando la sua percentuale di carica (φ) è inferiore alla soglia $\varphi_{\text{thr}} = 10\%$, e sia nuovamente disponibile per il noleggio quando raggiunge il valore di $\varphi_{\text{max}} = 100\%$.
- La **Stazione di Noleggio** rappresenta la situazione in cui un auto è stata rilasciata ed è pronta per essere noleggiata da un nuovo utente.

Infine, per quanto riguarda la **Strada**, essa rappresenta un fattore esterno che determina solo per quanto tempo un'auto sta generando profitto senza creare insoddisfazione nel cliente (a differenza di quanto accade nell'attesa per la riconsegna del veicolo, qui il cliente sta pagando per ricevere un servizio effettivo).

6

Modello computazionale

Per modello computazionale si intende una rappresentazione matematica e algoritmica utilizzata per simulare e analizzare il comportamento del sistema di figura 4.1.
Si è deciso di effettuare la simulazione mediante l'utilizzo del linguaggio Java, in quanto questo linguaggio offre un'ampia gamma di librerie e framework, garantendo al contempo portabilità e robustezza.
Il modello di simulazione adottato è di tipo **next-event**. Con questo tipo di simulazione, viene introdotto il concetto di tempo; in particolare, la simulazione next-event gestisce il tempo e gli eventi che si verificano progressivamente con il passare del tempo.
Gli elementi chiave sono:

- **Stato del sistema**: Descrive lo stato del sistema in un determinato istante di tempo. Il significato dello stato del sistema varia a seconda del livello considerato:
  - **Modello concettuale**: Lo stato è una collezione astratta di elementi e descrive come questi evolvono nel tempo.
  - **Livello delle specifiche**: Lo stato è rappresentato da un insieme di variabili matematiche e dalle equazioni che le relazionano.
  - **Livello di computazione**: Lo stato è un insieme di variabili di programma che vengono aggiornate man mano che la simulazione avanza.
- **Eventi**: Un evento è un accadimento che può modificare lo stato del sistema. Per definizione, lo stato può cambiare solo in presenza di eventi. Ogni evento è caratterizzato da un tipo specifico. Inoltre, è possibile definire eventi artificiali che non modificano lo stato del sistema, ma che sono utili per campionare statisticamente lo stato del sistema e/o pianificare eventi in momenti prestabiliti.
- **Clock di simulazione**: Rappresenta il valore corrente del tempo simulato.
- **Schedulazione degli eventi**: Meccanismo di avanzamento temporale che garantisce che gli eventi si verifichino nell'ordine corretto.
- **Lista degli eventi**: Struttura che contiene il tempo di occorrenza di ciascun evento.

6.1 Organizzazione del codice

Il codice della simulazione è strutturato secondo il pattern architetturale **MVC** (Model-View-Controller), molto diffuso nello sviluppo di sistemi software, in particolare nell'ambito della programmazione orientata agli oggetti. Si ha quindi:

- Il package **Controller**: Contiene la logica relativa a ciascun nodo del sistema. In particolare, all'interno del package `controller`, troviamo il codice relativo alle simulazioni (sia a orizzonte finito che infinito).
- Il package **Model**: Contiene il codice che gestisce la lista degli eventi e la loro gestione. Al suo interno, infatti, troviamo le classi `MsqEvent`, `MsqSum` e `MsqT`, le quali verranno approfondite in seguito.

Oltre a questi, abbiamo anche i seguenti package:

- **Utils**: Contiene le classi relative alla gestione delle statistiche e il calcolo dei profitti e dei costi del nostro sistema.
- **Libs**: Contiene il codice di base per la simulazione realizzata, implementato da **Steve Park** e **Deve Geyer**, il quale implementa codice per la generazioni di distribuzioni, la generazione di seed e molte altre cose utili.

6.2 Gestione degli eventi

6.2.1 MsqEvent

La classe **MsqEvent** modella gli eventi che avvengono nei nodi (centri di servizio) del sistema, come gli arrivi o le partenze dei job. Un evento è definito da tre attributi principali:

```
1    private double t;
2    private int x;
3    private boolean fromParking;
```

In particolare:

- **t**: Rappresenta il tempo in cui l'evento si verifica.
- **x**: Indica lo stato dell'evento, dove:
  - **0**: L'evento non è attivo.
  - **1**: L'evento è attivo.
- **fromParking**: Attributo opzionale che indica se un evento proviene dalla stazione di ricarica o dalla Stazione di Riconsegna. Tale attributo è fondamentale in Noleggio per capire da dove arriva una nuova macchina disponibile per essere noleggiata mentre negli altri nodi questo valore al massimo viene impostato ma non letto.

Tra i metodi principali di MsqEvent troviamo:

- **getNextEvent**: Tale metodo scorre la lista degli eventi del chiamante e identifica quello più imminente (ossia l'evento con il tempo minore t che risulta attivo). E' fondamentale per la simulazione ad eventi discreti, poiché permette di determinare quale evento processare successivamente, in particolare:
  - Se ci sono eventi attivi, ritorna quello con il tempo più basso;
  - Se non ci sono eventi attivi, restituisce -1, indicando che non ci sono ulteriori eventi da processare.
- **findOne**: Tale metodo identifica il primo server disponibile (ossia con $x = 0$) per un centro, in particolare restituisce quello che è stato inattivo per più tempo.
- **findActiveServers**: Tale metodo conta il numero di server attivi, cioè quelli che hanno $x = 1$. Questo è utile per monitorare il numero di server busy per uno specifico centro in un dato momento.

6.2.2 *MsqSum*

La classe **MsqSum** è responsabile per l'accumulo delle statistiche relative al tempo di servizio e al numero di job serviti. Questi dati vengono usati per calcolare le prestazioni del sistema alla fine della simulazione. Gli attributi principali di tale classe sono:

```
1         private double service;
2         private long served;
```

In particolare:

- **Service**: Tiene traccia del tempo totale che i job hanno trascorso in servizio.
- **Served**: Indica il numero di job che sono stati serviti fino a un determinato punto della simulazione.

6.2.3 *MsqT*

La classe **MsqT** modella il clock di sistema, un componente centrale nella simulazione a eventi discreti. Essa tiene traccia del tempo corrente e del tempo del prossimo evento. Gli attributi principali sono:

```
1         private double current;
2         private double next;
3         private double batchTimer;
```

In particolare:

- **current**: Il tempo attuale del sistema. Questo valore viene aggiornato man mano che gli eventi vengono processati.
- **next**: Il tempo del prossimo evento, che viene determinato dal metodo `getNextEvent` della classe *MsqEvent*.

- **batchTimer**: Un timer opzionale per la gestione di batch di eventi o per tenere traccia di particolari intervalli temporali(ad esempio, per la simulazione a orizzonte infinito)

L'aggiornamneto continuo di **current** e **next** permette di avanzare la simulazione nel tempo, garantendo che gli eventi vengono processati nell'ordine corretto e che il tempo avanzi in modo discreto, ovvero saltando da un evento all'altro.

6.2.4 Event Handler

Per la gestione degli eventi è stato utilizzato l'approccio proposto dal libro Discrete Event Simulation, nella quale si mantiene una entry per ciascun server, più una per gli ingressi. Quest'ultimi possono essere di due tipo: interni ed esterni. Per tenere traccia degli eventi che occorrono è stata utilizzata una lista degli eventi con la seguente struttura:

```
1    private final List<MsqEvent> serverList = new ArrayList<>(2 +
         SERVER_NUMBER);
```

6.3 Utils

Nel pacchetto `Utils` sono raggruppati un insieme di strumenti e funzioni di supporto per il progetto. La struttura del package include diverse componenti:

- **Constants**: Definisce le costanti utilizzate nel progetto, come parametri di tempo, probabilità e valori di configurazioni.
- **Distribution**: Fornisce metodi per gestire la generazioni di variabili casuali e i processi di arrivo e servizio. Contiene metodi fondamentali come:
  - **getArrival**: Genera tempi di arrivo esponenziali per i vari centri;
  - **getService**: Genera tempi di servizio esponenziale per i diversi centri.
- **RentalProfit**: Definisce le variabili e i metodi relativi ai guadagni e ai costi del centro di Noleggio.
- Ulteriori metodi per la scrittura delle statistiche di interesse su file CSV.

6.4 Controller

Il package `Controller` contiene la logica principale della simulazione, con classi specifiche per la gestione degli eventi associati ai diversi centri simulati (noleggio, parcheggio, ricarica e strada). Ogni classe controller rappresenta un centro specifico e, pur condividendo una struttura di base comune per la gestione degli eventi, implementa comportamenti distinti in base alle caratteristiche del centro che modella. L'inizializzazione di ciascun controller segue uno schema standard, in cui tutti i server vengono

inizialmente impostati come inattivi (idle), ovvero senza alcun evento in corso all'istante zero. Successivamente, viene generato il primo evento, che sarà necessariamente un arrivo, utilizzando la distribuzione specifica degli arrivi del centro corrispondente attraverso il metodo `getArrival(int arrivalType)` descritto nella sezione 6.3.
Dopo l'inizializzazione attraverso il costruttore di default, la simulazione avrà inizio. La classe `Sistema` verificherà quale sia il prossimo evento da processare (ossia l'evento con il tempo minimo tra quelli pianificati per tutti i centri) e passerà il controllo al centro associato a tale evento.

6.4.1 Noleggio

Dalla figura 4.1 si evince che la Stazione di Noleggio è un infinite server. Per questo motivo, la sua lista degli eventi è inizialmente composta da soli due elementi: uno per gli arrivi interni (ovvero una nuova macchina disponibile per essere noleggiata) e uno per gli arrivi esterni (un passeggero che desidera noleggiare una macchina). La lista ha quindi la seguente struttura:

$$\begin{pmatrix} \lambda_{\text{ext}} & \lambda_{\text{int}} \end{pmatrix}$$

```
1    private List<MsqEvent> serverList = new ArrayList<>(2);
```

Quando la classe `Sistema` passa il controllo alla classe Noleggio, quest'ultimo esegue le seguenti operazioni:

1. **Identifica l'evento da gestire**: Verifica quale evento è più prossimo (quello con il tempo più basso) e aggiorna le variabili temporali del sistema.
2. **Gestione dell'evento di arrivo**: Se l'evento corrisponde a un arrivo, il sistema verifica se l'arrivo è interno o esterno. Successivamente, controlla la disponibilità delle macchine, distinguendo tra quelle presenti nel parcheggio e quelle in ricarica. Se non ci sono macchine disponibili, l'evento viene rimandato. Se ci sono macchine sia nel parcheggio che in ricarica, viene scelta quella che ha atteso più a lungo.

```
1    if (sP == -1 && sR == -1) {      /* No car available */
2        ...
3    }
4
5    if (sP != -1 && sR != -1) {
6        /* Both Parcheggio and Ricarica have cars available */
7        ...
8    } else if (sP != -1) /* Available cars only in Parcheggio */
9        ...
10   else /* Available cars only in Ricarica */
11       ...
```

3. **Entrata in servizio**: Se ci sono macchine disponibili, l'utente può entrare in servizio. Viene calcolato il tempo di servizio tramite il metodo `getService(int`

`serviceType`), e, se necessario, viene attivato un nuovo server per gestire la richiesta.

4. **Gestione dell'evento di partenza**: Se l'evento corrisponde a una partenza, viene gestito il routing dal centro Noleggio al centro Strada. L'evento viene trasferito alla lista degli eventi di Strada, e il sistema viene aggiornato con il nuovo evento.
5. **Aggiornamento della lista degli eventi di sistema**: Infine, il sistema aggiorna la lista centrale degli eventi inserendo l'evento più imminente del centro Noleggio, garantendo che la simulazione proceda in modo corretto.

6.4.2 Riconsegna

Nella Stazione di Riconsegna, il flusso degli eventi viene gestito in modo tale da garantire che le auto possano entrare e uscire dal parcheggio aggiornando gli stati dei server. La classe Parcheggio modella un sistema con capacità finita, dove i server rappresentano le aree di parcheggio disponibili. La lista degli eventi, in questo caso, include gli arrivi esterni (una macchina che cerca un parcheggio), gli eventi interni (una macchina che ha terminato la riconsegna del veicolo ed è pronta per essere noleggiata) e i K_p server di Parcheggio. La lista ha quindi la seguente struttura:

$$\begin{pmatrix} \lambda_{\text{ext}} & \lambda_{\text{int}} & s_1 & s_2 & \dots & s_n \end{pmatrix}$$

```
1    private final List<MsqEvent> serviceList = new ArrayList<>(2 +
        PARCHEGGIO_SERVER);
```

Quando la classe `Sistema` passa il controllo alla classe Parcheggio, vengono eseguite le seguenti operazioni:

1. **Identificazione dell'evento da gestire**: Verifica quale evento è più prossimo (quello con il tempo più basso) e aggiorna le variabili temporali del sistema.
2. **Gestione dell'evento di arrivo**: Quando si verifica un arrivo, sia esso esterno o interno , viene verificata la disponibilità dei server (posti liberi nel parcheggio). Se esistono server disponibili, il veicolo viene assegnato al posto libero, e viene calcolato il tempo di fine servizio per indicare quando l'auto sarà pronta per essere noleggiata. Se tutti i server sono occupati, l'evento di arrivo viene considerato una perdita, e viene applicata una penalità per mancato servizio.
3. **Gestione dell'evento di fine servizio**: Quando l'evento corrisponde alla conclusione del servizio, il veicolo viene reso disponibile per essere noleggiato. Per indicare ciò, lo stato del server (ossia del posto di parcheggio) viene aggiornato per indicare che il veicolo è pronto per essere noleggiato (ma il server è ancora busy, e resterà tale finché un utente non noleggerà la macchina).

```
1    eventList.get(e).setX(2); /* Current server is no more
        usable (e = 2 car is ready to be rented) */
```

4. **Aggiornamento della lista degli eventi di sistema**: Infine, il sistema aggiorna la lista centrale degli eventi inserendo l'evento più imminente della classe Parcheggio, garantendo che la simulazione proceda in modo corretto.

6.4.3 Ricarica

La classe Ricarica segue la stessa logica operativa della classe Parcheggio. Le differenze principali risiedono nei valori delle costanti specifiche, quali il numero di server disponibili e il tasso di servizio medio.

6.4.4 Strada

La classe Strada modella un infinite server e si distingue dagli altri centri per l'assenza di arrivi esterni. Gli eventi sono generati esclusivamente dai job che transitano attraverso il sistema. La logica del controller di Strada è la seguente:

1. **Identificazione dell'evento da gestire**: Verifica quale evento è più prossimo (quello con il tempo più basso) e aggiorna le variabili temporali del sistema.
2. **Gestione dell'evento di arrivo**: Se l'evento corrisponde a un arrivo, viene calcolato il tempo di servizio ed in seguito viene verificato se ci sono server disponibili. In caso affermativo, viene attivato un server esistente, altrimenti viene creato un nuovo server per gestire il job in arrivo.
3. **Gestione dell'evento di partenza**: Se l'evento corrisponde a una partenza dobbiamo gestire il routing dell'evento:
   - **Uscita dal sistema**: con probabilità **P_EXIT**, il job lascia definitivamente il sistema.
   - **Permanenza nel sistema**: se il job non lascia il sistema, può essere instradato verso il centro Ricarica o Parcheggio. La scelta avviene tramite un ulteriore controllo probabilistico:
     - Con probabilità **P_RICARICA**, l'evento viene inviato al centro Ricarica e inserito nella lista degli eventi di tale centro.
     - Altrimenti, l'evento viene inviato al centro Parcheggio e aggiornato nella lista degli eventi corrispondente.

```
1  double pLoss = rngs.random();
2  if (pLoss < P_EXIT) { /* Job exit from this system */
3      ...
4  } else { /* Job stays in this system */
5      double pRicarica = rngs.random();
6      if (pRicarica < P_RICARICA) { /* Event sent to Ricarica
           */
7          ...
8      } else { /* Event sent to Parcheggio */
9          ...
10     }
```

```
11      }
```

4. **Aggiornamento della lista degli eventi di sistema**: Infine, il sistema aggiorna la lista centrale degli eventi inserendo l'evento più imminente del centro Strada, garantendo che la simulazione proceda in modo corretto.

6.5 Libreria Rngs

La classe `Rngs` nel pacchetto `Libs` è progettata per la generazione di numeri casuali multi-stream, offrendo una soluzione più robusta rispetto alla classe `Random` di Java, particolarmente utile in simulazioni dove la qualità statistica è cruciale. Utilizza un generatore di tipo Lehmer e il multi-stream consente di generare numeri casuali indipendenti e di impostare lo stato di tutti gli stream con una sola chiamata a `rngs.plantSeed(long x)`, dove `x` è il valore iniziale del seed. I numeri generati determinano i tempi di arrivo e di servizio, garantendo l'indipendenza tra i valori e evitando correlazioni indesiderate nel modello di simulazione.

7

Fase di verifica

La fase di verifica di un modello di simulazione si svolge con il fine ultimo di poter garantire che il modello computazionale sia effettivamente implementato in modo corretto.

Viene quindi effettuato un confronto tra i risultati ottenuti tramite simulazione, utilizzando come seed *123456789*, e quelli analitici, tenendo conto di un *intervallo di confidenza*[1] del 95%.

Il confronto sarà sviluppato considerando unicamente il caso stazionario.

7.1 Assunzioni

Per poter analizzare il modello nel contesto dello studio analitico trattato a lezione, è stato necessario apportare alcune semplificazioni.

Il modello iniziale, pur essendo più dettagliato e realistico, non era direttamente compatibile con gli strumenti e le metodologie analitiche impiegate. Per questa ragione, abbiamo introdotto alcune assunzioni che ci hanno permesso di ridurre la complessità del sistema senza compromettere troppo la rappresentatività dei risultati.

In particolare, queste semplificazioni includono:

- **Rimozione della pool di macchine**: Si assume che il numero di veicoli disponibili per il noleggio sia infinito, eliminando così qualsiasi vincolo legato alla disponibilità effettiva di auto. In questo contesto, il tasso di arrivo degli utenti al sistema (λ) coincide con il tasso di uscita dal centro di noleggio, poiché ogni richiesta di veicolo può essere soddisfatta immediatamente. Di conseguenza, non è necessaria la gestione della pool di macchine né la rappresentazione esplicita della Stazione di Noleggio. Viene considerato soltanto un tasso di arrivo che riflette il flusso degli utenti nel sistema (λ').
- **Eliminazione del feedback**: A seguito dell'eliminazione della pool di veicoli, è stata rimossa anche la logica di feedback relativa ai veicoli che, dopo aver

[1] Un intervallo di confidenza rappresenta un insieme di valori plausibili per il parametro in questione, associato alla stima che viene fatta del parametro stesso

completato il servizio nella Stazione di Riconsegna o nella Stazione di Ricarica, tornavano all'interno della pool. Ora i veicoli escono definitivamente dal sistema.

Il modello risultante a seguito dell'applicazione delle diverse ipotesi precedentemente discusse è rappresentato dalla figura 7.1.

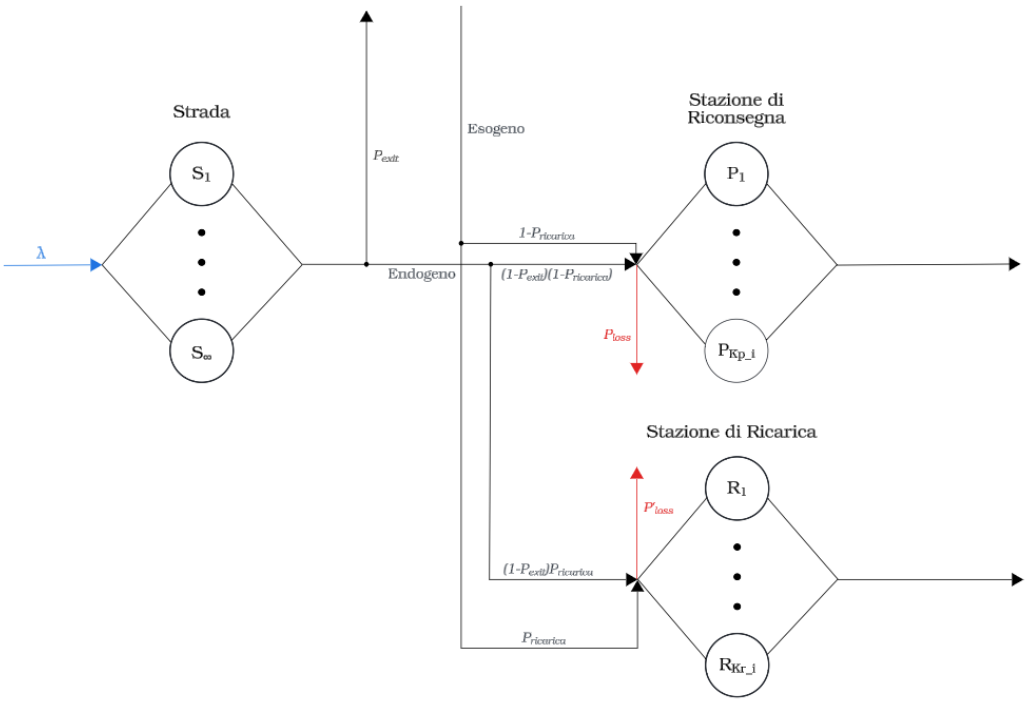


Figura 7.1: *Modello concettuale derivato dalle assunzioni*

7.2 Strada $M/M/\infty$

I risultati ottenuti analiticamente sono i seguenti:

- $\lambda = 0,2$ job/min
- $\mu_S = 0,03333$ job/min
- $E(S) = \lim_{m\to\infty} \frac{1}{m\mu} = 0$ min
- $E(S_i) = \frac{1}{\mu} = 30$ min
- $E(T_Q) = 0$ min
- $E(N_Q) = 0$
- $E(T_S) = E(T_Q) + E(S_i) = E(S_i) = 30$ min
- $E(N_S) = \lambda \cdot E(T_S) = 6$ job

Nota: nel contesto di sistema $M/M/\infty$, non è possibile definire l'utilizzazione come nei sistemi con un numero finito di server, perché ci sono infiniti server, quindi non esiste un concetto di "server totale" o "server occupati" in percentuale fissa.

L'equivalente dell'utilizzazione in un sistema $M/M/\infty$ è **la media del numero di server occupati** in ogni istante di tempo, che è data dalla seguente relazione:

$$L = \frac{\lambda}{\mu} = 6 = E(N_S)$$

Dove L indica il numero medio di server occupati (job nel sistema).

I risultati ottenuti dalla simulazione:

```
Strada

Critical endpoints E[T_S] =  30.039612129838453 +/- 0.12725686934151534
Critical endpoints E[N_S] =  6.0080825093001105 +/- 0.03391418861562531
```

7.3 Riconsegna $M/M/60$

I risultati ottenuti analiticamente sono i seguenti:

- $\lambda_P = 0,189$ job/min
- $\mu_P = 0,625$ job/min
- $E(S) = \frac{1}{m\mu_P}\Big|_{m=60} = 0,02667$ min
- $E(S_i) = \frac{1}{\mu_P} = 1,6$ min
- $\rho = \frac{\lambda_P}{m\mu_P}\Big|_{m=60} = 0,00504$
- $E(T_Q) = 0$ min
- $E(N_Q) = 0$
- $E(T_S) = E(T_Q) + E(S_i) = E(S_i) = 1,6$ min
- $E(N_S) = \lambda_P \cdot E(T_S) = 0,3024$ job

Nota: il modello simulato non riesce a coincidere completamente con quello analitico. Questo è dovuto al comportamento atipico dei server, i quali non si liberano automaticamente dopo aver terminato un servizio. Invece, devono attendere che un nuovo cliente arrivi a noleggiare una macchina prima di poter essere nuovamente operativi. Questa dipendenza tra la Stazione di Noleggio e la Stazione di Riconsegna introduce una complessità che non è completamente riproducibile tramite un approccio analitico. Per verificare questa ipotesi, sono stati scambiati i valori tra la Stazione di Riconsegna e Ricarica, simulando così un minor afflusso di macchine alla Stazione di Riconsegna e un tempo di servizio più elevato. Con questa modifica, il sistema ha mostrato una corrispondenza tra i risultati analitici e quelli simulati.

In ogni caso vengono allegati i risultati ottenuti dalla simulazione (notare che il tempo di risposta coincide):

```
Riconsegna

Critical endpoints E[T_S] =  1.6002146241382966 +/- 0.00729328373076518
Critical endpoints E[N_S] =  0.28710548536978 +/- 0.0018362248083599124
Critical endpoints rho =  0.004785086829221967 +/- 3.0591298727512066E-5
```

7.4 Ricarica $M/M/15$

I risultati ottenuti analiticamente sono i seguenti:

- $\lambda_R = 0,021$ job/min
- $\mu_R = 0,02217$ job/min
- $E(S) = \frac{1}{m\mu_R}\Big|_{m=15} = 3,00707$ min
- $E(S_i) = \frac{1}{\mu_R} = 45,10599$ min
- $\rho = \frac{\lambda_R}{m\mu_R}\Big|_{m=15} = 0,06315$
- $E(T_Q) = 0$ min
- $E(N_Q) = 0$
- $E(T_S) = E(T_Q) + E(S_i) = E(S_i) = 45,10599$ min
- $E(N_S) = \lambda_R \cdot E(T_S) = 0,94723$ job

Nota: la verifica tra il modello simulato e quello analitico ha avuto esito positivo. Ciò è dovuto al basso afflusso di macchine e ai tempi di servizio lunghi, che permettono una gestione più fluida delle richieste. Questo flusso meno intenso consente all'analisi analitica di fornire stime che coincidono con i risultati della simulazione, rendendo il modello analitico affidabile per la Stazione di Ricarica.

I risultati ottenuti dalla simulazione:

```
Ricarica

Critical endpoints E[T_S] =  45.13114876775385 +/- 0.1979159096408257
Critical endpoints E[N_S] =  0.9418499736790178 +/- 0.0057772514800139094
Critical endpoints rho =  0.06279155036457143 +/- 3.862104738868165E-4
```

7.5 Controlli di consistenza

Dopo aver verificato che i risultati ottenuti dalla simulazione coincidessero con quelli derivati dall'analisi analitica, è stato essenziale eseguire ulteriori controlli. Per garantire una validità ancora più solida ai risultati, è stato utile esaminare attentamente se fossero rispettati anche alcuni criteri di consistenza.

In particolare:

$$E(T_S) = E(T_Q) + E(S_i)$$
$$E(N_S) = E(N_Q) + m\rho$$
$$0 < \rho < 1$$

7.6 Tabelle riassuntive dei risultati

A seguire, vengono presentate delle tabelle che riassumono i risultati ottenuti rispettivamente nelle Stazioni di Riconsegna e Ricarica, e Strada.

Indice	Valore analitico	Valore simulazione	Intervallo di confidenza
$E(T_Q)$	$0,00000$	$0,00000$	$\pm\, 0,00000$
$E(N_Q)$	$0,00000$	$0,00000$	$\pm\, 0,00000$
$E(T_S)$	$1,60000$	$1,60021$	$\pm\, 0,00729$
$E(N_S)$	$0,30240$	$0,28711$	$\pm\, 0,00184$
ρ	$0,00504$	$0,00479$	$\pm\, 3,05913 \cdot 10^{-5}$

Tabella 7.1: *Tabella dei risultati (minuti) della Stazione di Riconsegna*

Indice	Valore analitico	Valore simulazione	Intervallo di confidenza
$E(T_Q)$	$0,00000$	$0,00000$	$\pm\, 0,00000$
$E(N_Q)$	$0,00000$	$0,00000$	$\pm\, 0,00000$
$E(T_S)$	$45,10599$	$45,13115$	$\pm\, 0,19792$
$E(N_S)$	$0,94723$	$0,94185$	$\pm\, 0,00578$
ρ	$0,06315$	$0,06279$	$\pm\, 3,86210 \cdot 10^{-4}$

Tabella 7.2: *Tabella dei risultati (minuti) della Stazione di Ricarica*

Indice	Valore analitico	Valore simulazione	Intervallo di confidenza
$E(T_Q)$	$0,00000$	$0,00000$	$\pm\, 0,00000$
$E(N_Q)$	$0,00000$	$0,00000$	$\pm\, 0,00000$
$E(T_S)$	$30,00000$	$30,03961$	$\pm\, 0,12726$
$E(N_S)$	$6,00000$	$6,00808$	$\pm\, 0,03391$

Tabella 7.3: *Tabella dei risultati (minuti) di Strada*

8

Validazione

La validazione di un modello è una fase essenziale per garantire che i risultati della simulazione rappresentino fedelmente il comportamento del sistema reale. In questa fase, i risultati ottenuti dalla simulazione vengono confrontati con dati sperimentali o analitici, verificando che il modello sia in grado di riprodurre accuratamente le dinamiche del sistema sotto studio. L'obiettivo è assicurarsi che il modello non solo funzioni correttamente dal punto di vista computazionale, ma che rifletta con precisione i processi reali che si intende modellare.

Nel nostro caso, il sistema in analisi è caratterizzato dall'afflusso di circa 12 utenti all'ora, secondo le statistiche fornite dall'articolo di riferimento. Questo implica che, ipotizzando che ci siano sempre macchine disponibili per essere noleggiate ci aspetteremmo di gestire circa 288 utenti in un'intera giornata. Per validare il modello, abbiamo impostato la simulazione con la massima disponibilità di auto, aumentando così la probabilità che ogni richiesta di noleggio da parte di un utente venga soddisfatta. In tali condizioni, il valore di utenti serviti $\in [244; 341]$ con un valore medio pari a $277,375$, avvicinandosi al valore teorico calcolato.

Un ulteriore aspetto della validazione del modello consiste nell'analisi dei flussi di veicoli nelle Stazioni di Riconsegna e Ricarica. Dalla simulazione, emerge che il numero di veicoli che accede alla Stazione di Ricarica è significativamente inferiore rispetto a quelli che affluiscono alla Stazione di Riconsegna. Questo comportamento è coerente con le probabilità assegnate alle due stazioni, che riflettono la maggiore frequenza con cui i veicoli tendono ad essere riconsegnati e parcheggiati piuttosto che a fermarsi per la ricarica. Tali risultati confermano che il modello simula correttamente il flusso dei veicoli tra le stazioni, rispettando le probabilità impostate per i diversi percorsi.

9

Design degli esperimenti

Gli esperimenti di simulazione sono stati organizzati in due sezioni principali:

- Simulazione ad orizzonte finito.
- Simulazione ad orizzonte infinito.

In questo capitolo si farà frequentemente riferimento all'intervallo di confidenza, già introdotto nel Capitolo 7. Risulta quindi utile una breve digressione per approfondire questo concetto. In particolare, l'intervallo di confidenza è definito come segue:

$$\text{Intervallo di Confidenza} = t^* \cdot \left(\frac{s}{\sqrt{n-1}} \right)$$

dove s rappresenta la deviazione standard, $n-1$ indica il numero di ripetizioni considerate, e t^* è il valore critico ottenuto dalla distribuzione t di *Student*. Notare che t^* è stato calcolato utilizzando la funzione `idfStudent(·)`, parte della classe `Rvms.java`.

9.1 Condizioni iniziali del sistema

Per la simulazione sono stati utilizzati i valori di default proposti nell'articolo di riferimento. In particolare:

- Il numero di parcheggi disponibili è pari a 75, di cui 15 dedicati alla ricarica delle auto elettriche.
- Il numero di macchine iniziali nel noleggio è 15, simulando così un contesto in cui la flotta di veicoli cresce nel tempo in base alla domanda.

Questi valori di default rappresentano il punto di partenza per analizzare l'evoluzione del sistema e testare diverse configurazioni.

9.2 Analisi dei profitti

Come già analizzato, il profitto della Stazione di Noleggio è il risultato di costi operativi e ricavi derivanti dall'attività di noleggio delle automobili.

A un certo istante di tempo, il numero di parcheggi disponibili potrebbe azzerarsi, o potrebbero non esserci macchine disponibili per il noleggio. Per modellare la realtà, in cui nessuno sarebbe disposto ad aspettare per noleggiare un'auto o per parcheggiarla al termine del noleggio, viene introdotta una penalità per tali situazioni. Questa penalità rappresenta il costo associato a una gestione inefficiente del centro, che non è in grado di soddisfare la domanda dei clienti. L'introduzione di tale costo ci consente di simulare scenari realistici, evidenziando quanto sia cruciale garantire una gestione ottimale delle risorse per evitare perdite di profitto. Esiste quindi una correlazione diretta fra il profitto del centro e il numero di macchine e parcheggi totali. In particolare:

- Al diminuire del numero di parcheggi totali, i costi operativi del centro si riducono, ma allo stesso tempo si rischia di incorrere in una riduzione dei profitti. Ciò accade perché, se i clienti non trovano parcheggi disponibili per terminare il noleggio, viene applicata una penalità che penalizza l'efficienza operativa del sistema.
- Al diminuire del numero di macchine disponibili per il noleggio, anche in questo caso i costi del centro diminuiscono, ma si potrebbe incorrere in una perdita di profitti. Se non ci sono auto da noleggiare quando richieste, il sistema si troverà a dover pagare una penalità per la mancata disponibilità, riducendo così la redditività complessiva.

In un contesto in cui diversi parametri sono strettamente correlati tra loro, la simulazione diventa uno strumento fondamentale per analizzare l'impatto di ciascuna variabile sul profitto complessivo del centro. Grazie alla simulazione, è possibile osservare come il profitto vari al modificarsi di parametri chiave come il numero di parcheggi e il numero di macchine disponibili, fornendo una visione più chiara delle dinamiche operative e dei trade-off coinvolti.

9.3 Simulazione ad orizzonte finito

La simulazione ad **orizzonte finito** si riferisce a un tipo di simulazione che analizza il comportamento di un sistema di code entro un intervallo di tempo limitato. In questo progetto, la simulazione ad orizzonte finito è stata implementata per un periodo di un giorno (86400 secondi).

Per migliorare la robustezza e l'affidabilità dei risultati, è stata adottata la tecnica delle **replicazioni**, che consiste nel ripetere più volte la stessa simulazione variando unicamente il seed iniziale per ciascuna esecuzione (assicurandosi che i seed non si sovrappongano). Ogni esecuzione del programma simulato è definita replicazione.

Sono state effettuate complessivamente 64 replicazioni ed i risultati ottenuti sono i seguenti:

Centro	Indice	Valore simulazione	Intervallo di confidenza
Noleggio	$E(T_Q)$	$6,55821$	$\pm 5,205723$
Noleggio	$E(N_Q)$	$1,22092$	$\pm 0,92514$
Noleggio	$E(T_S)$	$11,65244$	$\pm 5,20809$
Noleggio	$E(N_S)$	$2,22071$	$\pm 0,92195$
Noleggio	ρ	$0,15223$	$\pm 0,00496$
Ricarica	$E(T_Q)$	$0,00000$	$\pm 0,00000$
Ricarica	$E(N_Q)$	$0,00000$	$\pm 0,00000$
Ricarica	$E(T_S)$	$45,60268$	$\pm 1,91821$
Ricarica	$E(N_S)$	$0,88804$	$\pm 0,05718$
Ricarica	ρ	$0,06100$	$\pm 0,00394$
Parcheggio	$E(T_Q)$	$0,00000$	$\pm 0,00000$
Parcheggio	$E(N_Q)$	$0,00000$	$\pm 0,00000$
Parcheggio	$E(T_S)$	$1,58856$	$\pm 0,02326$
Parcheggio	$E(N_S)$	$0,29030$	$\pm 0,00707$
Parcheggio	ρ	$0,00484$	$\pm 1,18525 \cdot 10^{-4}$
Strada	$E(T_Q)$	$0,00000$	$\pm 0,00000$
Strada	$E(N_Q)$	$0,00000$	$\pm 0,00000$
Strada	$E(T_S)$	$30,26321$	$\pm 0,43514$
Strada	$E(N_S)$	$5,81610$	$\pm 0,12109$
Strada	ρ	$0,41586$	$\pm 0,01042$

Tabella 9.2: *Tabella dei risultati (minuti), ad orizzonte finito, per i vari centri*

Può essere utile visualizzare graficamente l'andamento transitorio del tempo di risposta di ciascun centro su 5 seed differenti:

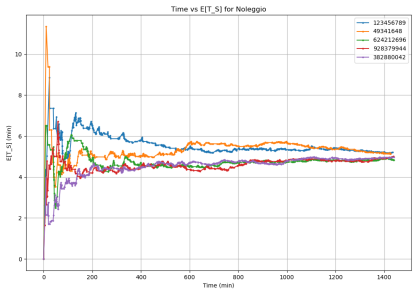


Figura 9.1: *Stazione di Noleggio (minuti)*

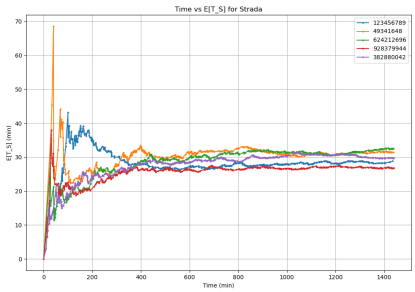


Figura 9.2: *Strada (minuti)*

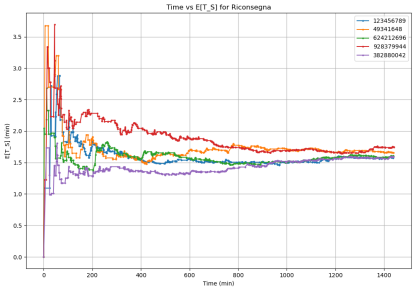


Figura 9.3: *Stazione di Riconsegna (minuti)*

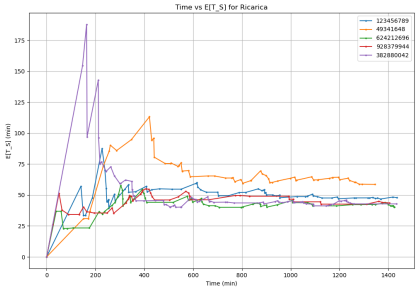


Figura 9.4: *Stazione di Ricarica (minuti)*

Con il passare del tempo, i tempi di risposta tendono a convergere verso uno stato preciso.

È necessario fare un'osservazione riguardo ai risultati ottenuti per la Stazione di Noleggio. In particolare, osservando la Figura 9.1, si nota che i seed tendono a convergere verso un valore intorno ai 5 secondi. Tuttavia, dalla Tabella 9.4, emerge che il valore medio calcolato è pari a 11,65244 secondi. Questa discrepanza è dovuta alla presenza di alcuni seed che divergono durante la simulazione, influenzando il valore medio complessivo.

9.3.1 *Analisi del profitto giornaliero*

Studiando il sistema su un intervallo finito di un giorno, si sono ottenuti i seguenti risultati:

- Entrate medie: ¥23.324,2472
- Costo medio: ¥16.792,1203
- Profitto medio: ¥5.933,6406

9.4 *Simulazione ad orizzonte infinito*

La simulazione ad **orizzonte infinito** si riferisce a una simulazione che mira a rappresentare il comportamento di un sistema in un tempo teoricamente illimitato, studiando le performance a regime. In questa tipologia di simulazione, si vuole esaminare il sistema quando ha raggiunto uno stato stazionario, ignorando i dati transitori.

Per ottenere risultati stabili e ridurre la varianza nelle stime delle metriche, è stata adottata la tecnica **batch means**, che suddivide la lunga esecuzione della simulazione in blocchi temporali di dimensioni uguali. Nel nostro caso, ciascun batch ha una dimensione di $\mathcal{B} = 1512$ osservazioni, e il numero totale di batch utilizzati per le stime è $\mathcal{K} = 128$. Ogni batch rappresenta un insieme indipendente di osservazioni statistiche che permette di stimare la media e la varianza delle misure di performance. Questo approccio consente di calcolare intervalli di confidenza attendibili per le stime, riducendo l'influenza delle fluttuazioni casuali tra i vari batch. I risultati della simulazione sono i seguenti:

Centro	Indice	Valore simulazione	Intervallo di confidenza
Noleggio	$E(T_Q)$	$0,00000$	$\pm\, 0,00000$
Noleggio	$E(N_Q)$	$0,00000$	$\pm\, 0,00000$
Noleggio	$E(T_S)$	$4,99576$	$\pm\, 0,02178$
Noleggio	$E(N_S)$	$0,99985$	$\pm\, 0,00614$
Noleggio	ρ	$0,11228$	$\pm\, 9,06617 \cdot 10^{-4}$
Ricarica	$E(T_Q)$	$0,00000$	$\pm\, 0,00000$
Ricarica	$E(N_Q)$	$0,00000$	$\pm\, 0,00000$
Ricarica	$E(T_S)$	$45,07377$	$\pm\, 0,18691$
Ricarica	$E(N_S)$	$0,94432$	$\pm\, 0,00576$
Ricarica	ρ	$0,06295$	$\pm\, 3,85456 \cdot 10^{-4}$
Parcheggio	$E(T_Q)$	$0,00000$	$\pm\, 0,00000$
Parcheggio	$E(N_Q)$	$0,00000$	$\pm\, 0,00000$
Parcheggio	$E(T_S)$	$1,59983$	$\pm\, 0,00786$
Parcheggio	$E(N_S)$	$0,28697$	$\pm\, 0,00191$
Parcheggio	ρ	$0,00478$	$\pm\, 3,17846 \cdot 10^{-5}$
Strada	$E(T_Q)$	$0,00000$	$\pm\, 0,00000$
Strada	$E(N_Q)$	$0,00000$	$\pm\, 0,00000$
Strada	$E(T_S)$	$30,03523$	$\pm\, 0,13659$
Strada	$E(N_S)$	$6,01101$	$\pm\, 0,03719$
Strada	ρ	$0,29244$	$\pm\, 0,00288$

Tabella 9.4: *Tabella dei risultati (minuti), ad orizzonte infinito, per i vari centri*

Visualizzando graficamente i risultati ottenuti è possibile notare come tutti i centri convergono ad uno stato stazionario.

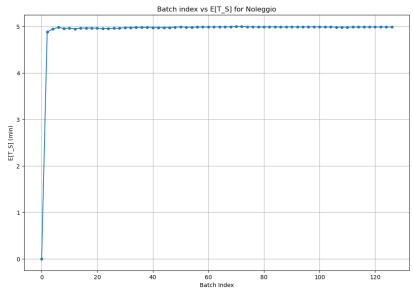


Figura 9.5: *Stazione di Noleggio (minuti)*

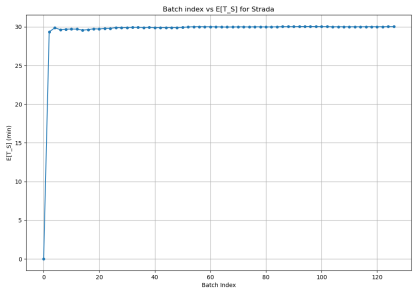


Figura 9.6: *Strada (minuti)*

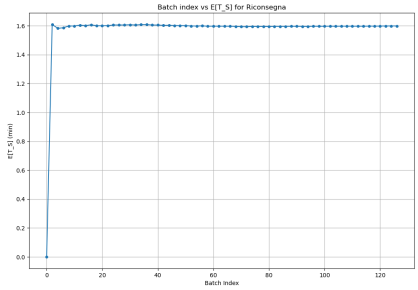


Figura 9.7: *Stazione di Riconsegna* (*minuti*)

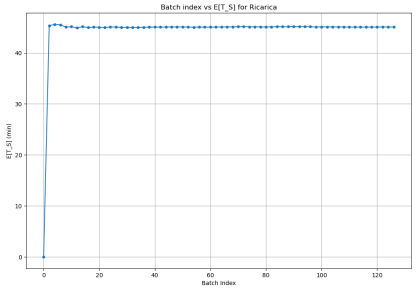


Figura 9.8: *Stazione di Ricarica* (*minuti*)

L'utilizzo di una scala logaritmica per la visualizzazione dei grafici può rivelarsi utile per analizzare meglio l'andamento dei valori, soprattutto quando questi risultano molto simili tra loro:

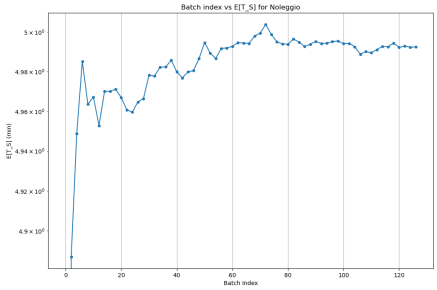


Figura 9.9: *Stazione di Noleggio* (*minuti*)

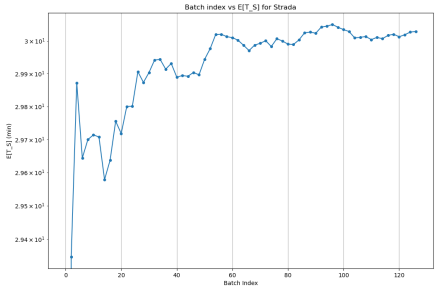


Figura 9.10: *Strada* (*minuti*)

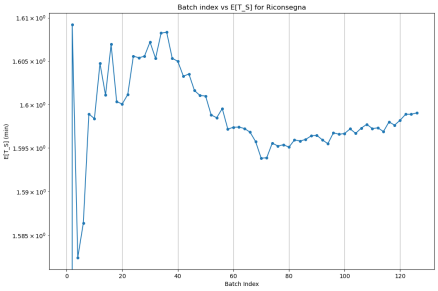


Figura 9.11: *Stazione di Riconsegna* (*minuti*)

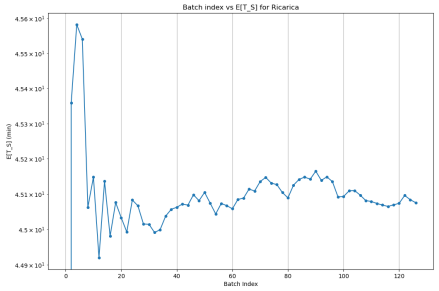


Figura 9.12: *Stazione di Ricarica* (*minuti*)

9.4.1 Analisi e ottimizzazione dei profitti

L'obiettivo principale dell'analisi è massimizzare il profitto del centro. Questo implica l'identificazione delle variabili ottimali ($k_i^*, j_i^*, initial\_cars^*$) (numero di posti di parcheggio, posti di ricarica, numero di veicoli iniziali) che massimizzano la funzione di profitto. Il vantaggio della simulazione risiede nella possibilità di evitare lo studio analitico del modello matematico, permettendoci invece di condurre numerose simulazioni per identificare i parametri che ottimizzano il profitto.

Al fine di trovare valori ottimali per la funzione di costo, è stata adottata una strategia di ottimizzazione iterativa, fissando due delle tre variabili in gioco e osservando come variava la funzione al variare della terza variabile. Questo approccio permette di esplorare il comportamento della funzione in modo isolato per ogni variabile, concentrandosi su una alla volta. Il procedimento è stato ripetuto in modo sistematico su tutte e tre le variabili, con l'obiettivo di identificare una terna di valori che potesse avvicinarsi il più possibile al minimo della funzione.

Procedura dettagliata

1. **Prima fase**: Si è fissata una coppia di variabili su valori iniziali predefiniti, variando solo la terza variabile. È stata eseguita una ricerca binaria per trovare il miglior valore della variabile non fissata. Questo processo ha comportato:

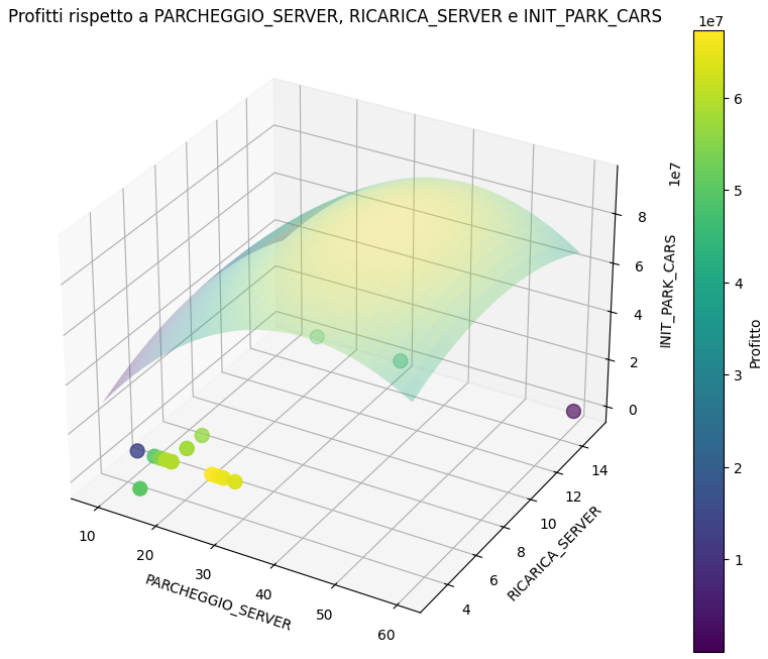
    - Un primo test su un valore iniziale.
    - Un dimezzamento progressivo dell'intervallo di ricerca (dividendo per 2 i valori testati) fino a individuare un valore ottimale o vicino all'ottimo per la variabile considerata.
    - Se un valore migliorava la funzione rispetto al precedente, si continuava a restringere l'intervallo.

2. **Seconda fase**: Una volta trovato un valore soddisfacente per la prima variabile, si è fissato questo valore insieme a una delle altre due variabili. Il processo di ricerca binaria è stato ripetuto per la seconda variabile libera, applicando la stessa logica:

    - Variare la seconda variabile all'interno di un intervallo inizialmente ampio.
    - Suddividere l'intervallo e testare valori intermedi, finché non si trova un valore ottimale.

3. **Terza fase**: Infine, con i valori delle prime due variabili fissati, è stata ripetuta la ricerca binaria sulla terza variabile, cercando di individuare un valore che potesse minimizzare ulteriormente la funzione di costo.

A partire dalle terne di valori simulati, abbiamo condotto un'analisi per approssimare i punti in modo da ottenere una rappresentazione dell'andamento della funzione di costo. La funzione di costo risultante rappresenta una superficie tridimensionale che esprime i profitti in funzione delle tre variabili:



In particolare, grazie all'analisi condotta sulle diverse simulazioni, siamo riusciti a migliorare significativamente il profitto. A partire da un profitto iniziale di ¥6.996.456 e, attraverso l'ottimizzazione delle variabili operative, si è potuto raggiungere un profitto di ¥$6,727589 \cdot 10^7$.

Questo valore è stato ottenuto con la configurazione ottimale rappresentata dalla terna di valori $(22, 5, 9)$, dove:

- 22 rappresenta il numero di posti di parcheggio.
- 5 indica il numero di posti di ricarica.
- 9 corrisponde al numero iniziale di auto parcheggiate.

Questi risultati evidenziano l'importanza di un'accurata gestione delle risorse e dell'ottimizzazione delle configurazioni operative per massimizzare i profitti del centro di noleggio in questione.

10

Studio della distribuzione degli inter-arrivi su Strada

L'analisi dei tempi di inter-arrivo è un elemento fondamentale nello studio dei processi stocastici, in particolare nei sistemi di coda o nei modelli di arrivo di eventi casuali, come nei sistemi di traffico, nelle reti di comunicazione e in altre applicazioni ingegneristiche. Questo studio si concentra sull'analisi di un dataset di tempi di inter-arrivo, ovvero l'intervallo di tempo tra arrivi consecutivi di eventi, e sul confronto di diverse distribuzioni teoriche per descrivere adeguatamente questi dati per *Strada*.

L'obiettivo di questo capitolo è confrontare diverse distribuzioni di probabilità con i dati sperimentali e determinare quale distribuzione descriva meglio il comportamento dei tempi di inter-arrivo su Strada. Le distribuzioni considerate includono la distribuzione esponenziale, la distribuzione di Pareto, la distribuzione lognormale, la distribuzione di Weibull e la distribuzione beta. Dopo un'analisi dettagliata delle proprietà di ciascuna distribuzione e del loro adattamento ai dati empirici, si giungerà a una conclusione sulla distribuzione più appropriata.

10.1 Descrizione dei Dati

I dati presi in considerazione consistono nei tempi di inter-arrivo misurati su Strada, ottenuti dalla simulazione a orizzonte finito.
I parametri su cui è stato poi eseguito lo studio della distribuzione sono stati:

```
START = 0.0;
STOP_FIN = 21600; // Ossia la simulazione ad orizzonte finito
            // è stata eseguita su un intervallo di 6 ore
PARCHEGGIO_MAX_QUEUE = 0;
RICARICA_MAX_QUEUE = 0;

INIT_PARK_CARS = 0; // Il numero di macchine iniziali presenti
              // all'interno della stazione di parcheggio è pari a 0
```

All'interno del dataset sono stati registrati sia il seed sia l'istante di tempo di un arrivo su Strada. Dal dataset si è poi calcolato il tempo tra due arrivi consecutivi come differenza tra i loro relativi tempi, arrivando quindi a definire il tempo degli inter-arrivi. Questi presentano una distribuzione altamente asimmetrica, con un picco significativo verso i valori più piccoli e una coda lunga verso i valori maggiori.

L'analisi di questi dati è stata condotta attraverso la costruzione di un istogramma (figura 10.1) per visualizzare la distribuzione empirica dei tempi di inter-arrivo. Successivamente, sono state considerate diverse distribuzioni teoriche, le cui curve di densità sono state sovrapposte all'istogramma per confrontare visivamente e quantitativamente l'aderenza di ciascuna distribuzione ai dati.

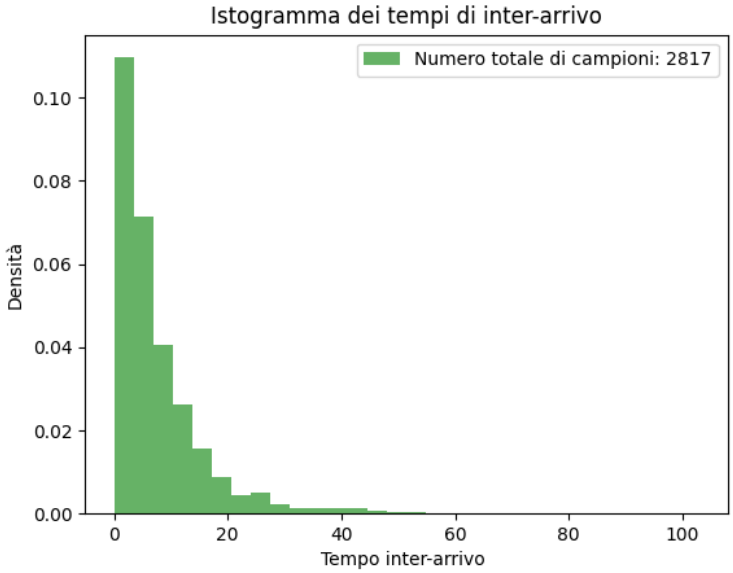


Figura 10.1

Per cercare di avere una distribuzione quanto più indipendente dal seed di riferimento, si è eseguito il campionamento selezionando i seguenti seed ed incociando di relativi dati:

```
selected_seeds = [123456789, 49341648, 624212696, 928379944, 382880042]  # 5 seeds
```

10.2 Distribuzioni considerate

Per decidere quali distribuzioni considerare, si è utilizzata una libreria Python chiamata **Fitter**. Questa libreria è progettata appositamente per aiutare a identificare quale distribuzione statistica si adatta meglio a un insieme di dati.

Il risultato della libreria è il seguente:

```
Fitted anglit distribution with error=0,052981)
Fitted alpha distribution with error=0,005118)
Fitted cosine distribution with error=0,049233)
Fitted arcsine distribution with error=0,044417)
Fitted dgamma distribution with error=0,015565)
Fitted argus distribution with error=0,060561)
Fitted crystalball distribution with error=0,024921)
Fitted expon distribution with error=0,000509)
Fitted bradford distribution with error=0,046208)
Fitted dweibull distribution with error=0,01596)
Fitted cauchy distribution with error=0,0137)
Fitted betaprime distribution with error=0,000387)
Fitted fatiguelife distribution with error=0,001254)
Fitted beta distribution with error=0,000467)
Fitted erlang distribution with error=0,000442)
Fitted exponnorm distribution with error=0,000528)
Fitted foldcauchy distribution with error=0,001715)
Fitted chi distribution with error=0,00414)
Fitted chi2 distribution with error=0,000542)
Fitted burr12 distribution with error=0,000377)
Fitted foldnorm distribution with error=0,009795)
Fitted fisk distribution with error=0,002413)
Fitted f distribution with error=0,000424)
Fitted gamma distribution with error=0,053479)
Fitted gennorm distribution with error=0,013937)
Fitted exponpow distribution with error=0,004555)
Fitted genhalflogistic distribution with error=0,003893)
Fitted gumbel_r distribution with error=0,011873)
Fitted gumbel_l distribution with error=0,042106)
Fitted halfcauchy distribution with error=0,001329)
Fitted halflogistic distribution with error=0,003848)
Fitted genlogistic distribution with error=0,011872)
Fitted halfnorm distribution with error=0,009794)
Fitted genextreme distribution with error=0,004068)
Fitted burr distribution with error=0,000656)
Fitted invgauss distribution with error=0,001991)
Fitted exponweib distribution with error=0,000455)
Fitted invgamma distribution with error=0,003684)
Fitted genpareto distribution with error=0,000387)
Fitted genexpon distribution with error=0,000509)
Fitted ksone distribution with error=0,058909)
Fitted gompertz distribution with error=0,000369)
Fitted invweibull distribution with error=0,004068)
Fitted halfgennorm distribution with error=0,000479)
Fitted kappa3 distribution with error=0,000959)
Fitted gengamma distribution with error=0,000459)
Fitted johnsonsu distribution with error=0,001873)
Fitted johnsonsb distribution with error=0,001313)
Fitted kappa4 distribution with error=0,003499)
Fitted geninvgauss distribution with error=0,000444)
Fitted gausshyper distribution with error=0,008561)
```

Tabella 10.1: *Output di Fitter eseguito su tutte quante le possibili distribuzioni*

Dall'output della tabella 10.1 il dato di interesse per valutare quale distribuzione prendere è `error`. Infatti, quanto più è basso l'errore quanto più la distribuzione in esame si adatta ai dati forniti. Per questa ragione, nella presente analisi, sono state prese in considerazione queste distribuzioni teoriche:

Beta (*beta*),
Beta Prima (*betaprime*),
Burr (*burr*),
Burr XII (*burr12*),
Chi Quadrato (*chi2*),
Erlang (*erlang*),
Esponenziale (*expon*),
Esponenziale modificata (*exponnorm*),
Esponenziale Weibull (*exponweib*),
Fisher-Snedecor (*f*),
Generalized Expon (*genexpon*),
Generalized Gamma (*gengamma*),
Generalized Half-normale (*halfgennorm*),
Generalized Inverse Gaussian (*geninvgauss*),
Generalized Pareto (*genpareto*),
Gompertz (*gompertz*).

La libreria Fitter tuttavia non è sufficiente da sola a verificare l'adattamento della distribuzione ai dati. Per questa ragione si è utilizzato anche il test di **Kolmogorov-Smirnov**. Questo è un metodo statistico non parametrico utilizzato per confrontare una distribuzione campionaria con una distribuzione teorica.
Il test calcola la massima differenza assoluta tra la *funzione di distribuzione cumulativa* (CDF) empirica dei dati e la CDF della distribuzione teorica. L'ipotesi nulla del

test afferma che i dati provengono dalla distribuzione specificata. Un *p-value* basso, inferiore a un certo livello di significatività (tipicamente 0.05), indica evidenza sufficiente per rifiutare l'ipotesi nulla, ossia non vi è un effetto significativo, una differenza o una relazione tra la distribuzioni considerata e la distribuzione dei dati reali, ossia la distribuzione ipotizzata non è una buona descrizione dei dati.
Inoltre, il test di Kolmogorov-Smirnov è utile poiché non richiede assunzioni specifiche sulla forma della distribuzione, rendendolo applicabile a una vasta gamma di situazioni. Tuttavia, è importante considerare anche le limitazioni del test, come la sensibilità ai grandi campioni, dove anche piccole differenze possono risultare significative, portando a un potenziale rifiuto dell'ipotesi nulla.
In combinazione con la libreria Fitter, il test di Kolmogorov-Smirnov fornisce un approccio robusto per valutare l'adattamento delle distribuzioni ai dati, contribuendo a una comprensione più approfondita delle caratteristiche dei dati analizzati.
Di seguito sono riportati i risultati del test K-S, già ordinati per p-value:

```
Test K-S: [gengamma   ], p-value = 0,28340642327562116
Test K-S: [burr12     ], p-value = 0,2804364375840269
Test K-S: [betaprime  ], p-value = 0,2362886521051195
Test K-S: [genpareto  ], p-value = 0,22566134056696285
Test K-S: [exponweib  ], p-value = 0,16100381576251843
Test K-S: [halfgennorm], p-value = 0,14825206923159195
Test K-S: [f          ], p-value = 0,11193802967341926
Test K-S: [chi2       ], p-value = 0,10445115450868203
Test K-S: [geninvgauss], p-value = 0,0941944213829291
Test K-S: [gompertz   ], p-value = 0,08640708367478511
Test K-S: [beta       ], p-value = 0,07623011741660668
Test K-S: [burr       ], p-value = 0,04942023998653122
Test K-S: [expon      ], p-value = 0,012633344287863509
Test K-S: [genexpon   ], p-value = 0,012629574799076662
Test K-S: [exponnorm  ], p-value = 0,008300474208301057
Test K-S: [erlang     ], p-value = 1,5676626485877767e-08
```

Dalle distribuzioni così testate si è quindi scelto di approfondire l'analisi tra le prime cinque di queste, le quali hanno tutte p-value > 0.05, ossia, per nessuna di queste è possibile rifiutare l'ipotesi nulla.

10.2.1 Distribuzione Generalized Gamma (gengamma)

La distribuzione Generalized Gamma (gengamma) è una distribuzione flessibile utilizzata per modellare una vasta gamma di fenomeni. La sua funzione di densità di probabilità è data da:

$$f(x; a, d, p) = \frac{p}{a^d \Gamma(d/p)} x^{d-1} e^{-(x/a)^p}$$

dove a, d e p sono i parametri della distribuzione, con:

- $a > 0$ che controlla la scala.
- $d > 0$ il parametro di forma della funzione.
- $p > 0$ il parametro di potenza.

I parametri che sono stati utilizzati per graficare la curva in figura sono stati:

- $a = 6,249940066444157$
- $d = 0,8697691506328915$
- $p = 1,1227287779568138$

L'adattamento della distribuzione Generalized Gamma ai dati empirici è stato buono nella parte iniziale dell'istogramma (figura 10.2), ovvero per i valori di inter-arrivo più piccoli. Eseguendo il test di Kolmogorov-Smirnov si ottiene il seguente risultato:

$$\text{p-value} = 0.28340642327562116$$

Si è analizzata la distribuzione Generalized Gamma e non la distribuzione Gamma poiché la prima risulta essere una generalizzazione più flessibile della seconda. Infatti la seconda non era in grado di catturare appieno il comportamento dei dati.

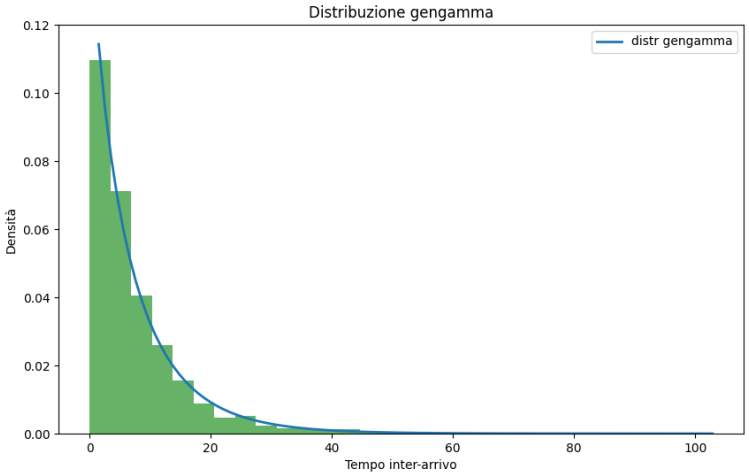


Figura 10.2: *Distribuzione gengamma*

10.2.2 Distribuzione Burr XII

La famiglia di distribuzioni di Burr è composta da 12 tipi di distribuzioni, indicati con numeri romani (Burr I, II, ..., XII). Questi tipi hanno formule diverse per la funzione di densità di probabilità (PDF), ma condividono una struttura parametrica comune che consente di modellare dati con caratteristiche diverse, come simmetria, asimmetria e variazioni nella coda.
La Burr Type XII (o Burr XII) è la più comune e viene spesso chiamata semplicemente "Burr" (quando non ci sono ambiguità), ma è tecnicamente un caso specifico.

È una distribuzione flessibile utilizzata in vari ambiti, come l'economia e l'affidabilità, per modellare dati che presentano una coda lunga o pesante. La sua funzione di densità di probabilità è data da:

$$f(x; c, k) = ck \frac{x^{c-1}}{(1 + x^c)^{k+1}}$$

dove $c > 0$ e $k > 0$ sono i parametri della distribuzione:

- c controlla la forma della distribuzione.
- k regola la coda della distribuzione.

I parametri che sono stati utilizzati per graficare la curva in figura sono stati:

- $c = 1,0091995857382052$
- $k = 8,025305788452137$

L'adattamento della distribuzione Burr12 ai dati empirici è stato buono sia nella parte iniziale dell'istogramma sia nelle code (figura 10.3), suggerendo che sia in grado di catturare sia i valori più piccoli che quelli estremi. Eseguendo il test di Kolmogorov-Smirnov, si ottiene il seguente risultato:

$$\text{p-value} = 0.2804364375840269$$

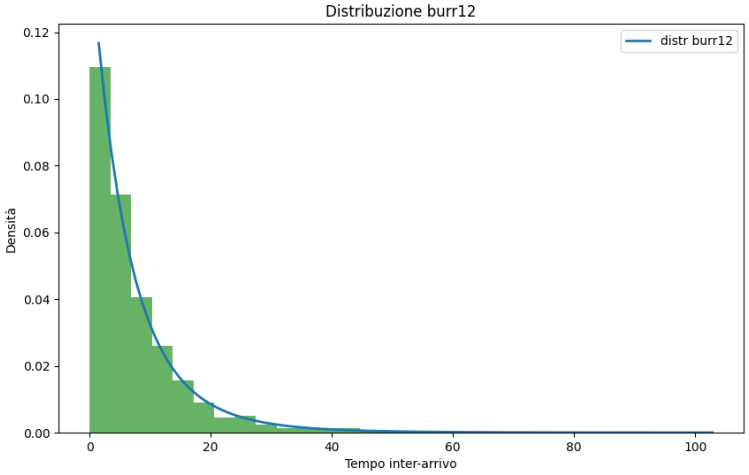


Figura 10.3: *Distribuzione burr12*

10.2.3 Distribuzione Beta prime

La distribuzione Beta Prime (o Beta inversa) è una distribuzione continua utilizzata in vari contesti, come l'analisi di sopravvivenza, la teoria della probabilità e l'economia. La sua funzione di densità di probabilità è data da:

$$f(x; \alpha, \beta) = \frac{x^{\alpha-1}(1 + x)^{-\alpha-\beta}}{B(\alpha, \beta)}$$

dove $B(\alpha, \beta)$ è la funzione Beta, definita come:

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}\, dt$$

e i parametri $\alpha > 0$ e $\beta > 0$ determinano rispettivamente la forma della distribuzione e il comportamento delle code.
I parametri che sono stati utilizzati per graficare la curva in figura sono stati:

- $\alpha = 0,9953192767312321$
- $\beta = 8,940034928431354$

Mentre la distribuzione Beta supporta i valori di $x \in (0, 1)$, la distribuzione Beta prime ha un supporto a $x \in (0, \infty)$.
L'adattamento della distribuzione Beta Prime ai dati empirici è stato soddisfacente (figura 10.4), indicando che riesce a catturare sia i valori centrali che quelli estremi. Eseguendo il test di Kolmogorov-Smirnov, si ottiene il seguente risultato:

$$\text{p-value} = 0.2362886521051195$$

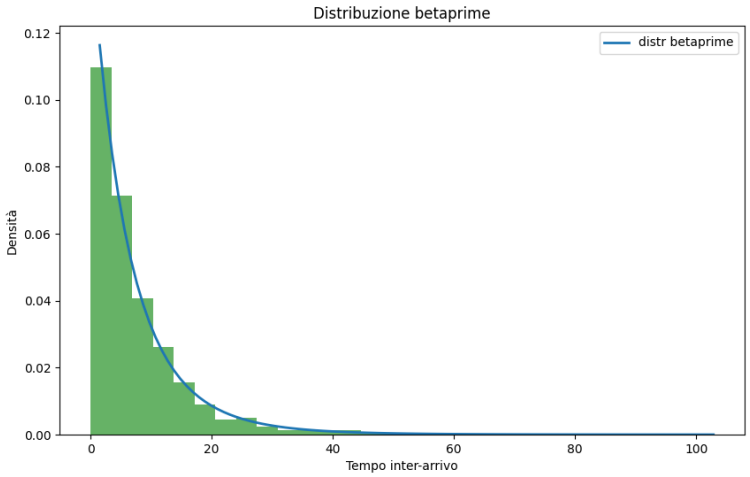


Figura 10.4: *Distribuzione betaprime*

10.2.4 Distribuzione Generalized Pareto

La distribuzione Generalized Pareto è una distribuzione continua che viene spesso utilizzata per modellare i valori estremi e le code di distribuzioni di probabilità. È particolarmente utile nell'analisi di fenomeni legati ai massimi o ai minimi estremi, come le inondazioni, i guasti meccanici o i picchi di traffico. La sua funzione di densità di probabilità è data da:

$$f(x; \xi, \sigma, \theta) = \frac{1}{\sigma}\left(1 + \xi\frac{x-\mu}{\sigma}\right)^{-\frac{1}{\xi}-1}$$

dove:

- ξ è il parametro di forma.

- σ è il parametro di scala.
- μ è il parametro di posizione.

La distribuzione Generalized Pareto è una generalizzazione della distribuzione Pareto ed è ampiamente utilizzata per modellare eventi estremi, specialmente nelle applicazioni di analisi di valori estremi.
I parametri che sono stati utilizzati per graficare la curva in figura sono stati:

- $\xi = 0,11484414547813493$
- $\sigma = 6,706135926087843$
- $\mu = 0,0006240783791455567$

Essendo che la distribuzione Generalized Pareto serve anche ad adattarsi a fenomeni con code più leggere o pesanti a seconda del valore di ξ, possiamo determinare che il sistema analizzato ha delle code "pesanti" (ossia che i valori estremi hanno una probabilità maggiore di verificarsi rispetto ad una distribuzione esponenziale) poiché $\xi > 0$.
L'adattamento della distribuzione Generalized Pareto ai dati empirici è stato soddisfacente (figura 10.5), suggerendo che la distribuzione riesce a catturare i picchi e le code osservate nei dati. Eseguendo il test di Kolmogorov-Smirnov, si ottiene il seguente risultato:

$$\text{p-value} = 0.22566134056696285$$

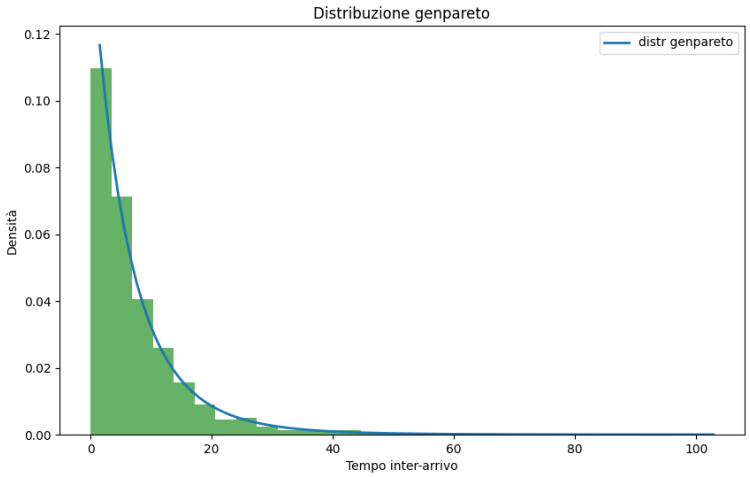


Figura 10.5: *Distribuzione betaprime*

10.2.5 Distribuzione Exponentiated Weibull

La distribuzione Exponentiated Weibull (ExponWeib) è una distribuzione continua che estende la distribuzione Weibull introducendo un parametro aggiuntivo che consente una maggiore flessibilità nella modellazione di dati, in particolare per fenomeni

con distribuzioni asimmetriche o con code più pesanti o leggere. La sua funzione di densità di probabilità è data da:

$$f(x;k,\lambda,\alpha) = \alpha \frac{k}{\lambda} \left(\frac{x}{\lambda}\right)^{\kappa-1} \left(1 - e^{(-x/\lambda)^{\kappa}}\right)^{\alpha-1} e^{-(x/\lambda)^{k}}$$

dove:

- $\lambda > 0$ è il parametro di scala.
- $k > 0$ è il parametro di forma della distribuzione Weibull.
- $\alpha > 0$ è il parametro di forma aggiuntivo che controlla l'ampiezza delle code.

I parametri che sono stati utilizzati per graficare la curva in figura sono stati:

- $\lambda = 5,825907495819535$
- $k = 0,8237221929141614$
- $\alpha = 1,2585538466059571$

La distribuzione Exponentiated Weibull permette di modellare una vasta gamma di fenomeni grazie alla sua capacità di adattarsi a code sia leggere che pesanti. In particolare, il parametro k influisce sulle code della distribuzione. Se $k < 1$ si osservano code più pesanti.
Questa affermazione va in accordo con quanto fatto per la distribuzione Generalized Pareto.
L'adattamento della distribuzione Exponentiated Weibull ai dati empirici è stato soddisfacente (figura 10.6), indicando che riesce a catturare sia i valori centrali che quelli estremi. Eseguendo il test di Kolmogorov-Smirnov, si ottiene il seguente risultato:

$$\text{p-value} = 0.16100381576251843$$

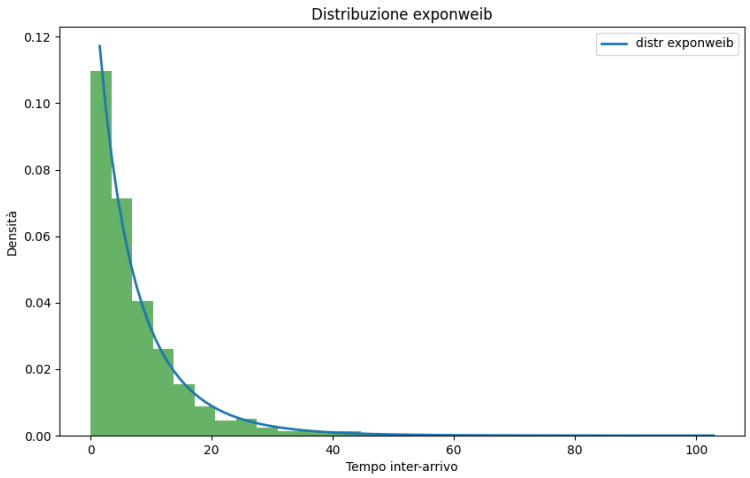


Figura 10.6: *Distribuzione Exponentiated Weibull*

10.3 Confronto tra le Distribuzioni

Il confronto tra le distribuzioni teoriche e i dati empirici è stato condotto attraverso un'analisi grafica e quantitativa (figura 10.7). La sovrapposizione delle curve di densità delle distribuzioni sull'istogramma fornisce indicazioni visive sull'adattamento di ciascuna distribuzione ai dati.

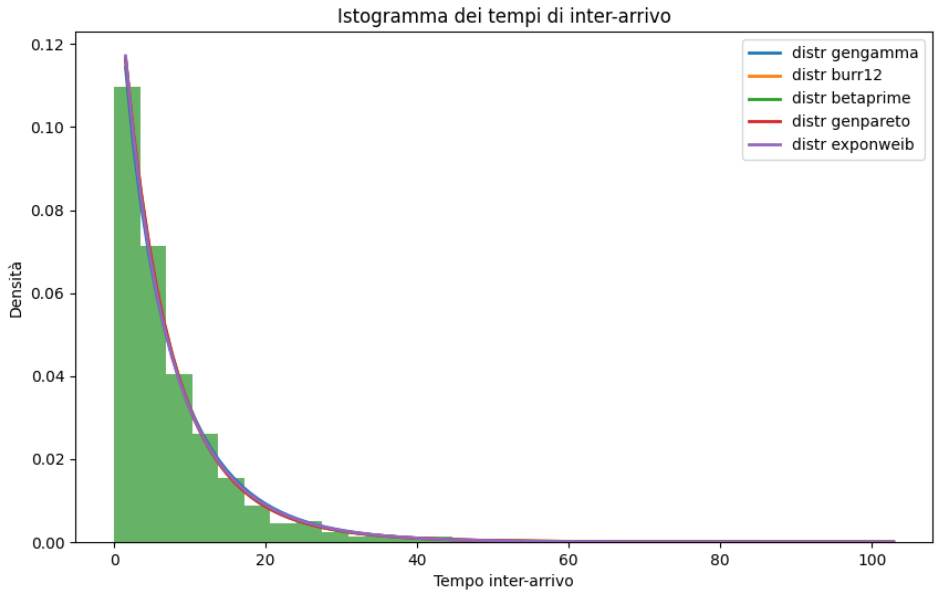


Figura 10.7: *Confronto delle distribuzioni prese in esame*

10.4 Conclusioni

Dal confronto non emerge alcuna distribuzione che segue meglio di un'altra la distribuzione dei dati simulati. Tuttavia è possibile concludere che la distribuzione dei dati simulati non sia una distribuzione esponenziale (su questa simulazione ristretta di 6 ore) poiché il p-value della distribuzione esponenziale è 0,012633344287863509.

11

Conclusioni

Lo studio condotto evidenzia l'importanza di ottimizzare le configurazioni operative in un sistema di car-sharing per migliorare la redditività del servizio. L'approccio simulativo adottato ha permesso di calcolare i profitti in relazione a diverse variabili, come il numero di veicoli disponibili, la capacità di parcheggio e le strutture di ricarica. Questi calcoli hanno fornito una base solida per identificare le configurazioni che portano a una performance economica ottimale.

I risultati ottenuti sono significativi: il profitto medio, inizialmente pari a ¥6.996.456, è aumentato fino a ¥67.275.890, con un incremento del 863%. Questo dimostra chiaramente come la gestione efficiente delle risorse, tramite l'ottimizzazione delle variabili operative, possa tradursi in un significativo aumento della redditività.

L'approccio basato sulla simulazione ha inoltre permesso di esplorare in modo rapido e sicuro una serie di possibili configurazioni, senza il rischio di errori derivanti da calcoli manuali complessi.

In sintesi, l'ottimizzazione delle configurazioni operative in un sistema di car-sharing non solo favorisce una maggiore efficienza, ma costituisce una leva fondamentale per migliorare i profitti e garantire la sostenibilità economica del servizio a lungo termine. I risultati di questo studio confermano l'importanza di approcci quantitativi e simulativi per supportare decisioni strategiche in contesti complessi e dinamici.