

# Analisi parco divertimenti Islands of Adventure

Andrea De Filippis

andrea.defilippis@students.uniroma2.eu

Università di Roma Tor Vergata

Matricola - 0333703

Emanuele Valzano

emanuele.valzano@students.uniroma2.eu

Università di Roma Tor Vergata

Matricola - 0341634

Simone Nicosanti

simone.nicosanti@students.uniroma2.eu

Università di Roma Tor Vergata

Matricola - 0334319

26 agosto 2024

## Indice

<b>1</b>	<b>Introduzione</b>	<b>4</b>
1.1	Descrizione del contesto e Problematiche . . . . .	4
1.2	Obiettivi . . . . .	4
<b>2</b>	<b>Modello Concettuale</b>	<b>5</b>
2.1	Descrizione degli Utenti . . . . .	5
2.2	Centri del sistema e loro caratteristiche . . . . .	5
2.3	Flusso . . . . .	6
2.4	Politiche di Scheduling . . . . .	6
2.5	Stati del Sistema . . . . .	6
2.6	Eventi ed Evoluzione del Sistema . . . . .	6
<b>3</b>	<b>Fun Index</b>	<b>7</b>
<b>4</b>	<b>Modello delle Specifiche</b>	<b>8</b>
4.1	Nota sulle unità di misura . . . . .	8
4.2	Dati trovati . . . . .	8
4.3	Modellazione dei Gruppi . . . . .	8
4.4	Modellazione dei Centri . . . . .	9
4.5	Modellazione del Routing . . . . .	10
4.6	Scaling dei Dati . . . . .	11
4.7	Fasce Orarie . . . . .	11
4.8	Matrice di Routing . . . . .	12
<b>5</b>	<b>Modello Computazionale</b>	<b>13</b>
5.1	Linguaggio, Librerie e Approccio . . . . .	13
5.2	Legenda delle Classi . . . . .	13
5.3	Gestione degli Eventi e della Simulazione . . . . .	14
5.4	Raccolta delle Statistiche . . . . .	16
5.5	Gestione delle Code . . . . .	18
5.6	Gestione dei Valori Randomici . . . . .	18

<b>6 Verifica</b>	<b>19</b>
6.1 Risoluzione del Sistema . . . . .	19
6.2 Configurazione di Verifica . . . . .	20
6.3 Analisi Analitica per tipo di Centro . . . . .	20
6.3.1 Ingresso . . . . .	20
6.3.2 Ristorante . . . . .	20
6.3.3 Attrazione . . . . .	21
6.4 Risultati della Verifica con Simulatore . . . . .	21
<b>7 Validazione</b>	<b>25</b>
7.1 Calibrazione del Modello . . . . .	25
7.2 Confronto con Valori Reali . . . . .	26
7.3 Controlli di Consistenza . . . . .	26
<b>8 Esperimenti</b>	<b>29</b>
8.1 Analisi del FunIndex . . . . .	29
8.1.1 Considerazioni . . . . .	29
8.1.2 Esperimento . . . . .	29
8.1.3 Risultati . . . . .	29
8.2 Analisi dei tempi di coda . . . . .	29
8.2.1 Risultati . . . . .	30
8.3 Analisi del numero di utenti prioritari gestibili . . . . .	30
8.3.1 Risultati . . . . .	30
8.4 Analisi dei tempi di coda per intervallo . . . . .	30
8.4.1 Risultati . . . . .	30
8.5 Analisi del Rho . . . . .	34
8.5.1 Risultati . . . . .	34
<b>9 Modello Migliorativo</b>	<b>35</b>
9.1 Considerazioni . . . . .	35
9.2 Limitazione del modello di base . . . . .	35
9.3 Obiettivi . . . . .	35
9.4 Politica di Scheduling Migliorativa . . . . .	35
9.5 Esito del miglioramento . . . . .	35
<b>10 Modello Concettuale</b>	<b>36</b>
10.1 Small Rider Groups . . . . .	36
<b>11 Modello delle Specifiche</b>	<b>36</b>
11.0.1 Probabilità di Routing per Small Rider Groups . . . . .	36
11.0.2 Estrazione della Coda . . . . .	36
<b>12 Modello Computazionale</b>	<b>38</b>
12.1 Passaggio al modello Improved . . . . .	38
<b>13 Verifica</b>	<b>38</b>
<b>14 Validazione</b>	<b>38</b>
14.1 Controlli di Consistenza . . . . .	38
<b>15 Esperimenti</b>	<b>40</b>
15.1 Analisi dei tempi di coda per Intervallo . . . . .	40
15.1.1 Risultati . . . . .	40
15.1.2 Variazione del numero di posti riservati . . . . .	41
15.2 Analisi dei tempi di coda . . . . .	42
15.2.1 Risultati . . . . .	42
15.3 Analisi del Rho . . . . .	45
15.3.1 Risultati . . . . .	45
15.4 Analisi del FunIndex . . . . .	46
15.4.1 Risultati . . . . .	46
<b>16 Conclusioni</b>	<b>48</b>

<b>17 Dati</b>	<b>49</b>
<b>18 Links</b>	<b>51</b>

# 1 Introduzione

Islands of Adventure è un parco situato negli USA, più precisamente ad Orlando in Florida. Il parco riceve ogni anno un enorme numero di visitatori e le code alle giostre devono essere accuratamente gestite, perciò è un perfetto caso di studio. Lo scopo di questa analisi è di studiare il parco per individuare criticità e possibili miglioramenti per la soddisfazione dei visitatori.

## 1.1 Descrizione del contesto e Problematiche

Il parco è composto da numerose attrazioni, ognuna delle quali richiede ai visitatori di mettersi in coda prima di poterla utilizzare. Al momento sono presenti due tipologie di biglietti: basic oppure express. Il biglietto basic non dà alcun tipo di vantaggio, mentre l'express permette di accedere a file prioritarie e, quindi, di saltare completamente o in parte coloro che sono in coda nella fila normale.

Nelle giornate di punta, come durante le festività, il parco raggiunge un livello di affollamento molto elevato e ciò può portare ad un aumento non indifferente dei tempi di attesa, soprattutto per utenti non express. Data l'impossibilità di eliminare le code all'interno di un contesto del genere, risulta vitale gestirle in modo efficiente ed intelligente.

## 1.2 Obiettivi

Lo studio condotto sul sistema è volto al raggiungimento dei seguenti obiettivi:

- Mantenere basso il tempo di attesa di utenti prioritari che, avendo pagato un biglietto più costoso, si aspettano un trattamento preferenziale: in particolare si vuole avere un tempo di attesa inferiore ai 30 minuti.
- Studiare la soddisfazione degli utenti (definita in [sezione 3](#)) e cercare di aumentarla rispetto allo stato di base in modo da aumentare l'affluenza al parco.

## 2 Modello Concettuale

In questa fase dell'analisi, le caratteristiche del sistema vengono definite a un livello astratto. L'obiettivo principale è esaminare gli stati e i possibili eventi all'interno del sistema, valutando come questi siano correlati. In Figura 1 è riportata una rappresentazione del sistema come rete di code.

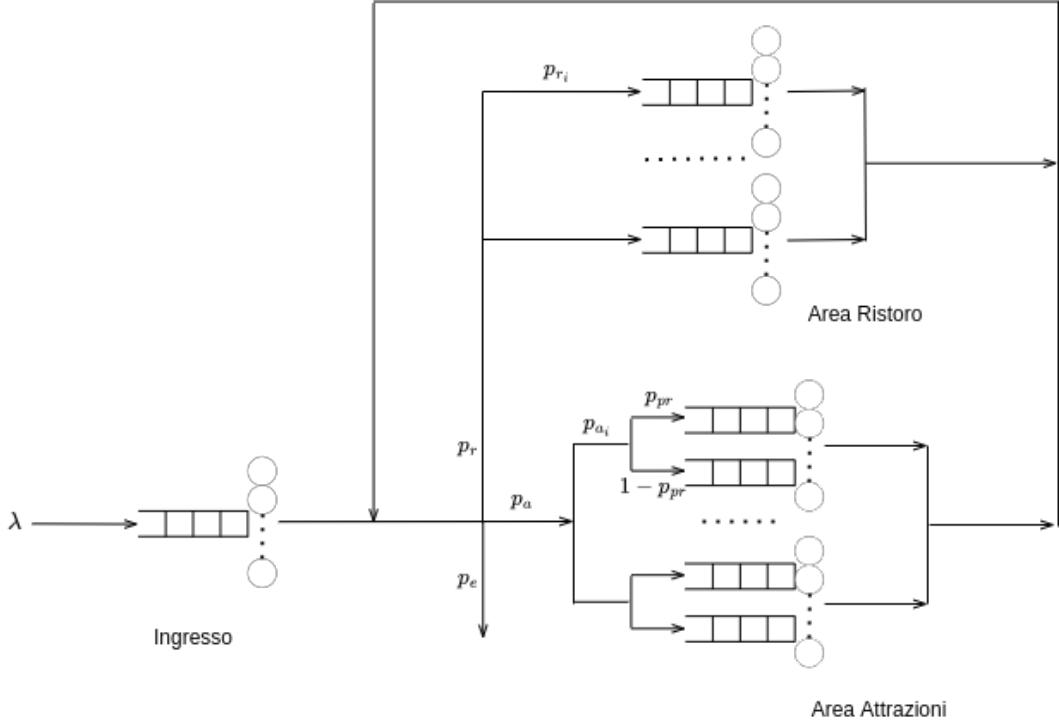


Figura 1: Diagramma Modello Concettuale

Nel diagramma soprastante è presente anche un'area ristoro. Tale area è stata aggiunta per rappresentare meglio la realtà. In questo modo è possibile rappresentare una diversa affluenza alle aree del parco a seconda dell'ora del giorno: e.g. durante l'orario del pranzo le persone saranno molto più inclini a spostarsi nell'area ristoro rispetto al resto della giornata, diminuendo il carico sulle attrazioni.

### 2.1 Descrizione degli Utenti

Gli utenti del sistema possono essere di due tipologie:

- **High Priority Riders:** utenti che hanno acquistato il biglietto express, che fornisce l'accesso a code prioritarie per le giostre.
- **Basic Riders:** utenti che hanno acquistato il biglietto base per l'ingresso al parco.

Quando gli utenti vanno ad un parco divertimenti, tendono ad organizzarsi in gruppi che difficilmente si separano per andare in zone diverse: per modellare questo aspetto è stato supposto che un gruppo sia un'unità inseparabile caratterizzata dalla sua dimensione e in cui tutti i membri hanno lo stesso tipo di biglietto.

### 2.2 Centri del sistema e loro caratteristiche

Alla luce del contesto, le macro aree che caratterizzano il sistema sono suddivise come segue:

- **Ingresso:** Rappresenta la parte del sistema in cui avviene il controllo del biglietto che si suppone precedentemente acquistato. In questo caso il numero di serventi rappresenta il numero di persone fisiche e/o tornelli dedicati al controllo dei biglietti di coloro che entrano nel parco.

- **Area Attrazione:** Le attrazioni sono distribuite in centri separati, ciascuno dedicato a una specifica attrazione, e i clienti con accesso prioritario usufruiscono di una corsia preferenziale, diversa per ogni attrazione. In questo caso il numero di serventi rappresenta il numero totale di posti a sedere di cui la giostra dispone. Non viene considerato il numero di veicoli delle attrazioni.
- **Area Ristoro:** Area dedicata alla ristorazione, accessibile agli utenti in ogni momento, con ogni ristorante che serve le richieste indipendentemente dagli altri. In questo caso il numero di serventi rappresenta il numero di "risorse umane" dedicate ad ogni gruppo all'interno del ristorante, assumendo che ogni gruppo abbia un responsabile dedicato.

### 2.3 Flusso

Alla luce di quanto detto in precedenza, diamo quindi qui una breve descrizione della dinamica del sistema in analisi. Come si evince dalla [Figura 1](#), passato l'ingresso del parco, un gruppo può decidere di spostarsi verso le attrazioni oppure verso l'area ristoro, mentre una percentuale tendenzialmente bassa di utenti può scegliere di uscire immediatamente (ad esempio per qualche tipo di imprevisto). Se il gruppo sceglie di spostarsi verso le attrazioni, dovrà scegliere una tra le diverse che il parco offre: in questo caso la scelta è tendenzialmente influenzata da tre aspetti principali, ovvero la popolarità della giostra, la lunghezza della coda in quella giostra e il numero di visite che sono state già eventualmente fatte a quella giostra. Allo stesso modo se decide di andare all'area ristoro, dovrà scegliere uno dei punti ristoro che il parco mette a disposizione. Terminato il servizio presso il centro che si è scelto (appunto un ristorante o un'attrazione), le scelte precedenti si ripetono fino a quando il gruppo non sceglie di uscire dal sistema oppure fino alla chiusura del parco, in cui tutti i gruppi ancora presenti sono obbligati ad uscire. Durante la chiusura è permesso a tutti centri di terminare il servizio corrente.

Una cosa significativa da tenere in considerazione è che l'affluenza alle due aree e all'uscita varierà a seconda dell'orario del giorno: ad esempio nella fascia oraria del pranzo vi sarà un'affluenza maggiore all'area ristoro piuttosto che a quella delle attrazioni, mentre nella fascia oraria della sera i visitatori saranno più propensi ad uscire.

### 2.4 Politiche di Scheduling

Per quanto riguarda i ristoranti e l'ingresso, abbiamo una sola coda con disciplina FIFO. Nel caso delle giostre, invece, abbiamo due code, entrambe a disciplina FIFO, di cui una è prioritaria per coloro che possiedono il biglietto express, mentre l'altra è per coloro che hanno il biglietto normale. Difficilmente in un parco divertimenti si ha una politica prioritaria stretta, in cui si prendono utenti appartenenti alla coda non prioritaria solo in assenza di utenti di utenti prioritari, ma è invece più comune che si prendano alcuni utenti dalla coda prioritaria e altri dalla coda non prioritaria per poi mandarli in servizio tutti insieme al prossimo giro della giostra. In questo modo si evita di avere una eccessiva ingiustizia tra le due code in caso di un grande numero di visitatori prioritari.

### 2.5 Stati del Sistema

Per quanto riguarda le code, in ogni istante, lo stato di una coda è rappresentato dai gruppi che vi sono accodati.

Nei serventi, invece, lo stato è rappresentato dall'essere occupato o meno. Notiamo che mentre l'ingresso e i ristoranti sono dei centri conservativi, cioè in cui se la coda è non vuota, tutti i serventi sono occupati, questo non vale nel caso delle attrazioni in cui la presenza di un gruppo di dimensioni maggiori rispetto ai posti disponibili potrebbe portare la giostra a partire con un numero di posti occupati minore rispetto al numero totale di posti nella giostra. Il centro ha, inoltre, un ulteriore stato, ovvero quello di chiusura: quando il centro è chiuso non accetta nuovi arrivi e coloro che vi arrivano sono ridiretti verso un altro centro. Il centro passa nello stato *chiuso* quando il parco arriva all'orario di chiusura.

### 2.6 Eventi ed Evoluzione del Sistema

A supporto della descrizione degli eventi nel sistema, si introduce la seguente terminologia:

- Si identifica con il termine *gruppo* o *job* un gruppo di persone che arriva al parco.
- Per *sistema* si intende l'insieme dei suoi centri, ognuno dei quali assume un ruolo ben definito.

Gli eventi considerati sono quelli di arrivo ed uscita per ogni centro. Al livello del *sistema* i possibili eventi sono i seguenti:

- L'arrivo di un *job* presso il sistema. Tale evento è un evento di arrivo al centro di Entrata.
- L'uscita di un *job* dal sistema. Questo può avvenire quando il job finisce il servizio in uno dei centri del sistema.

La gestione degli eventi di arrivo e di uscita nei centri varia in base al tipo di centro:

- **Entrata:**

- **arrivo:** un job in arrivo all'entrata viene messo in coda oppure, se è presente un servente libero, entra immediatamente in servizio.
- **uscita:** l'evento di uscita dal centro si ha quando termina il controllo biglietti di un gruppo. Questo evento porta alla generazione di un arrivo ad uno degli altri centri del sistema, sia esso un'attrazione o un ristorante, e al passaggio allo stato *libero* del servente. Se vi sono altri gruppi da servire nella coda, viene estratto il prossimo gruppo in attesa e fatto partire il suo servizio, facendo passare il servente di nuovo allo stato *occupato*.

- **Attrazioni:**

- **arrivo:** un job in arrivo ad un'attrazione viene messo in coda oppure, se il centro non è in servizio, entra immediatamente in servizio.
- **uscita:** l'evento di uscita di un job dall'attrazione si ha quando termina il servizio dell'attrazione, momento in cui viene schedulato l'evento di uscita per tutti i job correntemente in servizio. L'uscita di un job porta alla generazione di un arrivo ad uno dei centri nelle aree di ristoro oppure delle attrazioni, oppure all'uscita dal parco del job. Se vi sono altri gruppi da servire nella coda, viene fatto partire il successivo servizio.

- **Ristoranti:**

- **arrivo:** un job in arrivo ad un centro dell'area ristoro viene messo in coda oppure, se è presente un servente libero, entra immediatamente in servizio.
- **uscita:** l'evento di uscita da un ristorante si ha quando termina il servizio di uno dei gruppi in servizio. Ciò porta alla generazione di un arrivo ad uno dei centri nelle aree di ristoro oppure delle attrazioni, oppure all'uscita dal parco del job. Se vi sono altri gruppi da servire nella coda, viene estratto il prossimo gruppo in attesa e fatto partire il suo servizio, facendo passare il servente di nuovo allo stato *occupato*.

È importante sottolineare che la gestione delle attrazioni è tale per cui una volta partito il servizio di un insieme di gruppi, anche se vi sono dei posti vuoti, l'arrivo di un altro gruppo all'attrazione non comporta l'inizio di un nuovo servizio, bensì l'attesa di terminazione del servizio correntemente in corso: questa è sicuramente una particolarità rispetto a modelli di centro più classici.

Abbiamo poi che, quando il parco arriva all'orario di chiusura, in ogni centro:

- coloro che sono in coda ad un centro, a prescindere da quale esso sia, vengono fatti uscire dalla coda;
- non si accettano più nuovi arrivi al centro;
- si terminano i servizi attualmente in esecuzione;
- tutti coloro che sono ancora nel parco si avviano verso l'uscita.

### 3 Fun Index

In questo contesto, è stato definito il divertimento di una persona, chiamato *FunIndex*, che ricordiamo essere oggetto di uno degli obiettivi dello studio, come riportato in [Equazione 1](#):

$$FunIndex = \frac{RidingTime}{AttractionQueueTime + 1} \quad (1)$$

Questa definizione consente di ottenere una metrica che si migliora durante il tempo trascorso sulle giostre e peggiora durante l'attesa in coda, rendendola quindi un buon indicatore per stimare la soddisfazione dei visitatori all'interno del parco. Il +1 al denominatore è stato aggiunto per evitare possibili divisioni per zero.

## 4 Modello delle Specifiche

Riconducendoci al diagramma in [Figura 1](#), possiamo scrivere le equazioni di traffico riportate in [Equazione 2](#):

$$\begin{cases} \lambda_a = (\lambda + \lambda_a + \lambda_r) \cdot p_a \\ \lambda_r = (\lambda + \lambda_r + \lambda_a) \cdot p_r \\ \lambda_e = (\lambda + \lambda_a + \lambda_r) \cdot p_e \end{cases} \quad (2)$$

Notiamo che ogni volta che gli utenti terminano una visita a una delle aree del parco, si trovano ancora nel sistema, creando un punto di confluenza del traffico. Di conseguenza, i vari centri del parco devono gestire sia il flusso delle nuove richieste provenienti dall'ingresso che quelle generate dal feedback delle diverse aree di servizio.

### 4.1 Nota sulle unità di misura

Da questo punto in poi per tutto il resto del documento, per non appesantire le notazioni, si considerano le seguenti unità di misura:

- Tempi:  $\text{min}$
- Arrivi:  $\frac{\text{persone}}{\text{min}}$

### 4.2 Dati trovati

Attraverso alcune ricerche online è stato possibile trovare alcuni dati relativi a:

- Numero totale di persone che visitano il parco in un anno (<https://queue-times.com/parks/64/attendances>)
- Tempi di corsa e numero di posti a sedere di alcune giostre (pagine Wikipedia delle giostre)
- Tempi medi di coda giornalieri per singola giorstra (<https://queue-times.com/parks/64/stats/2022>)

Al contrario però non sono stati trovati dati relativi ai tempi di servizio dell'ingresso o al numero o tempi di servizio dei ristoranti. Inoltre, alla luce dei dati trovati, si è deciso di ridurre lo studio alle attrazioni di cui si conoscevano tutti i dati, sette nello specifico, e di considerare quattro ristoranti ed un singolo centro di ingresso. Questo chiaramente ha portato a scalare il dato trovato sul numero di arrivi alla riduzione che abbiamo fatto, come vedremo nella successiva [sottosezione 4.6](#).

### 4.3 Modellazione dei Gruppi

Come accennato in precedenza, i gruppi sono considerati come insieme inseparabili che si muovono insieme in ogni circostanza e i cui membri hanno tutti quanti la stessa priorità. Per quanto riguarda la dimensione del gruppo, essa è stata modellata con la distribuzione in [Equazione 3](#)

$$N = 1 + \text{Poisson}(\lambda = 3) \quad (3)$$

Una modellazione di questo tipo permette di avere:

- Gruppi di dimensione discreta, come effettivamente è nella realtà, evitando le complicazioni portate da distribuzioni continue.
- Si tratta di una distribuzione che dà un peso maggiore a sinistra della distribuzione, rendendo più probabili gruppi di piccole dimensioni (tra 1 e 6) persone e meno probabili gruppi di dimensioni maggiori, come spesso accade nella realtà dei fatti.

Ricordando che per la distribuzione di Poisson vale:

$$P(\text{Poiss}_\lambda \leq x) = \sum_{k=0}^x \frac{\lambda^k e^{-\lambda}}{k!}$$

Troviamo che:

$$P(N \leq x) = P(\text{Pois}_\lambda \leq x - 1) = \sum_{k=0}^{x-1} \frac{\lambda^k e^{-\lambda}}{k!}$$

Da cui abbiamo che:

$$\begin{aligned}
 P(N \leq 6) &= 0.91608 = 91.608\% \\
 P(N = 1) &= 0.04979 = 4.979\% \\
 P(N = 2) &= 0.14936 = 14.936\% \\
 P(N = 3) &= 0.22404 = 22.404\% \\
 P(N = 4) &= 0.22404 = 22.404\% \\
 P(N = 5) &= 0.16803 = 16.803\% \\
 P(N = 6) &= 0.10082 = 10.082\%
 \end{aligned}$$

$$E[N] = 1 + E[Pois_\lambda] = 4$$

Ovvero la maggior parte dei gruppi è effettivamente di dimensione minore di sei, che potrebbe essere considerata la dimensione di una famiglia piuttosto numerosa, mentre ad essere più probabili sono gruppi di tre e quattro persone che sono le dimensioni medie di famiglie. Per quanto riguarda invece eventuali gruppi di amici che arrivano al parco i medesimi ragionamenti si possono applicare: gruppi numerosi sono rari ma, ammesso che ci possano essere, questi potrebbero essere considerati come più gruppi meno numerosi.

Per quanto riguarda invece la priorità, dato il costo del biglietto e l'esperienza personale di alcuni membri del gruppo, si è considerata una probabilità di essere un gruppo prioritario pari a  $p_{prior} = 0.1$ .

#### 4.4 Modellazione dei Centri

Dopo attenta analisi e in funzione dei dati trovati i centri sono stati modellati come segue:

- Ingresso
  - Arrivi: Markoviani
  - Servizio: k-erlang in cui il valore di k dipende dalla dimensione del gruppo. Questa scelta modellistica è stata dettata dal fatto che c'è bisogno di un certo tempo (per quanto piccolo) per controllare il biglietto di ogni membro del gruppo; allo stesso tempo il controllo del biglietto di un singolo membro potrebbe dare problemi anche se questa possibilità è rara: per questo motivo si è scelto di modellare il servizio per il singolo membro con un'esponenziale e quindi, complessivamente, il servizio per il gruppo sarà dato da una k-erlang con k dimensione del gruppo (si sta assumendo, come detto in precedenza, che i membri del gruppo si muovano tutti insieme, quindi una volta che un membro ha passato il controllo aspetterà gli altri).
  - Classi: Una classe di priorità con coda a disciplina FIFO.
- Attrazioni
  - Arrivi: Non Markoviani
  - Servizio: Il tempo di servizio della singola giostra è stato modellato con una variabile aleatoria uniforme della seguente forma

$$U[tempoGiostra - 0.5; tempoGiostra + 0.5]$$

Questa scelta modellistica è stata dovuta al fatto che la durata della corsa di una giostra è un aspetto controllato in modo molto restrittivo e automatizzato perché da essa dipendono anche significativi aspetti di sicurezza; ciononostante non è stato ritenuto opportuno modellarla con un tempo deterministico in quanto vi possono essere aspetti aleatori ad influenzare la durata, come un attrito variabile sui binari della giostra oppure leggeri ritardi degli operatori delle giostre

- Classi: Due classi di priorità ognuna a disciplina FIFO.
- Estrazione dalla coda. Analizziamo in questa sede come viene fatta l'estrazione dalla coda:
  1. Una percentuale dei posti, abbiamo supposto il 40%, viene riservata ad appartenenti alla coda prioritaria.

2. Il resto dei posti, viene usato dai gruppi non prioritari.
3. Se vi sono posti avanzati, questi vengono ceduti nuovamente ai prioritari

Inoltre, per evitare problemi di starvation dei gruppi perché risultano troppo grandi per il numero di posti a loro assegnati, ad esempio un gruppo prioritario di dimensione maggiore del 40% dei posti oppure un gruppo basic di dimensione maggiore del restante 60%, vengono aggiunti i seguenti meccanismi prima di eseguire i passi sopra citati:

1. Viene estratto un gruppo non prioritario se non ne è stato estratto nessuno nelle ultime  $N$  estrazioni (abbiamo supposto  $N = 3$ ).
  2. Se possibile, viene estratto un gruppo prioritario.
- Numero di posti. All'interno di una giostra possiamo distinguere solitamente in *veicoli e posti a sedere* di ciascun veicolo: nel nostro contesto supponiamo che la giostra sia formata da un unico grande veicolo con un numero di posti totale dato dal prodotto. Quest'assunzione non ha ripercussioni sulla validità della simulazione in quanto è ragionevole assumere che gruppi che vanno su un'attrazione che non entrano su un unico veicolo si dividano tra diversi e inoltre veicoli diversi partono solitamente allo stesso istante di tempo.

- Ristoranti

- Arrivi: Non Markoviani
- Servizio. k-erlang in cui il valore di  $k$  dipende dalla dimensione del gruppo. Questa scelta modellistica è stata dettata dal fatto che c'è bisogno di un certo tempo per preparare l'ordine di ogni membro del gruppo; allo stesso tempo il tempo di preparazione del piatto per il singolo membro potrebbe dare problemi o richiedere una preparazione più lunga del normale anche se questa possibilità è rara: per questo motivo si è scelto di modellare il servizio per il singolo membro con un'esponenziale e quindi, complessivamente, il servizio per il gruppo sarà dato da una k-erlang con  $k$  dimensione del gruppo.
- Classi: Una classe di priorità con coda a disciplina FIFO.

## 4.5 Modellazione del Routing

I routing tra i diversi centri del parco sono stati modellati come segue:

- Attrazioni. Per un gruppo che va ad un parco divertimenti, la probabilità che un gruppo  $j$  ha di andare ad una giostra è solitamente influenzata dalla popolarità di quella giostra, dal numero di persone in coda a quella giostra e dal numero di volte in cui la giostra è stata fatta da quel gruppo; possiamo quindi assegnare uno *score* ad ogni attrazione per il job  $j$ :

$$s_{j \rightarrow a_i} = \alpha \cdot f(\text{pop}_i) + \beta \cdot g(v_{ji}) + \gamma \cdot h(N_{q_i})$$

dove  $\alpha, \beta, \gamma$  rappresentano i pesi dati a ciascun termine. In particolare è stato scelto di modellare le funzioni nel seguente modo:

$$\begin{aligned} f(\text{pop}_i) &= \frac{\text{pop}_i}{\sum_{k=1}^{n_a} \text{pop}_k} \\ g(v_{ji}) &= \left(1 - \frac{v_{ji}}{\max_k v_{jk}}\right) \\ h(N_{q_i}) &= \left(1 - \frac{N_{q_i}}{\max_k N_{q_k}}\right) \end{aligned}$$

dove:

- $f(\text{pop}_i)$  rappresenta il termine relativo alla popolarità ed è stato modellato come una softmax delle popolarità di tutte le attrazioni.
- $g(v_{ji})$  è il termine che tiene conto della coda della giostra e permette di assegnare un bonus se una giostra ha una coda inferiore rispetto alle altre.
- $h(N_{q_i})$  considera il numero di corse già fatte dal gruppo ad una determinata giostra e assegna un bonus alle giostre con meno visite.

Calcolato il valore  $s_{j \rightarrow a_i}$  si procede al calcolo delle probabilità con una funzione *softmax* nel seguente modo:

$$\hat{p}_{j \rightarrow a_i} = \frac{e^{s_{j \rightarrow a_i}}}{\sum_{k=1}^{n_a} e^{s_{j \rightarrow a_k}}} \quad (4)$$

Notiamo che, definendola in questo modo, il vettore delle probabilità di andare ai diversi centri somma ad 1, come effettivamente deve essere. Per quanto riguarda la  $N_{q_i}$  un gruppo prioritario guarda la lunghezza della sua coda, mentre se non prioritario guarda sia alla lunghezza della coda non prioritaria sia a quella della coda prioritaria.

Potrebbe esserci una probabilità, anche se bassa, che un gruppo sia troppo grande per i posti dell'attrazione: poiché i membri del gruppo vanno sull'attrazione tutti insieme, questo creerebbe un accodamento infinito nella giostra in quanto la disciplina delle code è FIFO. Per evitare questa condizione si impone, supponendo che un gruppo possa conoscere in anticipo la dimensione della giostra a cui sta andando, che in questo caso la probabilità di andare a quella giostra sia nulla; di conseguenza abbiamo:

$$\bar{p}_{j \rightarrow a_i} = \begin{cases} 0 & \text{se } groupSize > postiGiostra \\ \hat{p}_{j \rightarrow a_i} & \text{altro} \end{cases}$$

A questo punto il vettore non è più normalizzato, quindi la probabilità sarà data dalla normalizzazione dei valori così ottenuti. In conclusione quindi:

$$p_{j \rightarrow a_i} = \frac{\bar{p}_{j \rightarrow a_i}}{\sum_{k=1}^{n_a} \bar{p}_{j \rightarrow a_k}}$$

- Ristoranti. La probabilità di andare ai diversi ristoranti è stata modellata in maniera uniforme.
- Aree. La probabilità di andare alle diverse aree è stata modellata con una probabilità dipendente dalla fascia oraria.

## 4.6 Scaling dei Dati

Partendo dalla modellazione che abbiamo fatto dei gruppi e dallo loro dimensione media, possiamo fare uno scaling dei dati sugli arrivi trovati: nel riscalare il dato è stata supposta la possibilità di scalare linearmente sui giorni dell'anno.

$$11025000 \frac{\text{persone}}{\text{anno}} = 30205 \frac{\text{persone}}{\text{giorno}}$$

Questo per il parco complessivo che è formato da 14 giostre; riscalando su 7 giostre, supponendo la possibilità di scalare linearmente, abbiamo trovato circa  $15100 \frac{\text{persone}}{\text{giorno}}$  i cui arrivi sono da ridistribuire tra le diverse fasce orarie come spiegato nella sezione che segue.

## 4.7 Fasce Orarie

Da dati trovati online, è risultato che il parco è aperto per dodici ore, dalle nove del mattino alle nove di sera; tuttavia non sono stati trovati dati rispetto alla ripartizione degli arrivi tra le diverse fasce o alle tendenze di spostamento delle persone nei diversi slot. Seguendo una logica di buon senso e basandosi su esperienze personali si è scelto di configurare le fasce orarie come riportato in [Tabella 1](#). Si è assunto infatti che la maggior parte delle persone arrivino in mattinata o nella fascia

Fascia Oraria	Arrivi $\lambda[\frac{\text{per}}{\text{min}}]$	Probabilità Attrazioni $p_a$	Probabilità Ristoro $p_r$	Probabilità Uscita $p_e$
9:00 → 12:00	38	0.9	0.05	0.05
12:00 → 14:00	26	0.55	0.45	0.05
14:00 → 19:00	6	0.9	0.05	0.05
19:00 → 21:00	0.8	0.6	0.2	0.2
21:00 →	0	0	0	1.0

Tabella 1: Fasce Orarie

oraria del pranzo, per poi avere una diminuzione dell'affluenza al parco nelle fasce pomeridiane e serali.

Notiamo quindi che quando si arriva alla fascia oraria di chiusura, tutti coloro che escono da un centro, terminando il loro servizio, vengono ridiretti verso l'uscita.

## 4.8 Matrice di Routing

Riportiamo in matrice [Tabella 2](#) la matrice di routing del sistema.

	<b>Attr<sub>1</sub></b>	...	<b>Attr<sub>n_a</sub></b>	<b>Rest<sub>1</sub></b>	...	<b>Rest<sub>n_r</sub></b>	<b>Exit</b>
<b>Entr</b>	$p_a p_{a_1}$	...	$p_a p_{a_{n_a}}$	$p_r p_{r_1}$	...	$p_r p_{r_{n_r}}$	$p_e$
<b>Attr<sub>1</sub></b>	$p_a p_{a_1}$	...	$p_a p_{a_{n_a}}$	$p_r p_{r_1}$	...	$p_r p_{r_{n_r}}$	$p_e$
...	...	...	...	...	...	...	...
<b>Attr<sub>n_a</sub></b>	$p_a p_{a_1}$	...	$p_a p_{a_{n_a}}$	$p_r p_{r_1}$	...	$p_r p_{r_{n_r}}$	$p_e$
<b>Rest<sub>1</sub></b>	$p_a p_{a_1}$	...	$p_a p_{a_{n_a}}$	$p_r p_{r_1}$	...	$p_r p_{r_{n_r}}$	$p_e$
...	...	...	...	...	...	...	...
<b>Rest<sub>n_r</sub></b>	$p_a p_{a_1}$	...	$p_a p_{a_{n_a}}$	$p_r p_{r_1}$	...	$p_r p_{r_{n_r}}$	$p_e$

Tabella 2: Matrice di Routing

Ricordiamo che le probabilità  $p_a, p_r, p_e$  variano con la fascia oraria e facciamo inoltre notare che, data una riga, le sue probabilità sommano sempre ad 1; infatti, ricordando il modo in cui sono state definite  $p_{a_i}$  e  $p_{r_i}$ , troviamo:

$$\sum_{k=1}^{n_a} p_a p_{a_k} + \sum_{k=1}^{n_r} p_r p_{r_k} + p_e = p_a \sum_{k=1}^{n_a} p_{a_k} + p_r \sum_{k=1}^{n_r} p_{r_k} + p_e = p_a + p_r + p_e = 1$$

## 5 Modello Computazionale

### 5.1 Linguaggio, Librerie e Approccio

Come linguaggio di programmazione è stato scelto Java in quanto la strutturazione in classi e la presenza del polimorfismo sono stati considerati aspetti fondamentali per la realizzazione di un simulatore flessibile, configurabile e semplice da modificare.

A supporto dell'implementazione sono state usate alcune librerie per scrittura/lettura di file *.csv* o *.json* per gestire facilmente la scrittura dei risultati e la lettura delle configurazioni.

Come approccio di simulazione si è adottato un approccio di tipo *Next Event Simulation*.

### 5.2 Legenda delle Classi

Diamo qui una panoramica generale sulle diversi classi che formano il simulatore:

- Classi Controller: Contengono i Main del simulatore e sono usate per i diversi tipi di simulazione.
- Classe **Simulation**: Permette di eseguire una simulazione a orizzonte finito o ad orizzonte infinito del sistema, a seconda delle necessità della simulazione.
- Classi Factory
  - **CenterFactory.java**: responsabile della costruzione dei centri partendo da dei file *.csv* di configurazione.
  - **EventBuilder.java**: responsabile della costruzione delle istanze di eventi che avvengono nel sistema.
  - **NetworkBuilder.java**: responsabile della creazione di tutti i centri del sistema e delle istanze di **RoutingNode** ad essi associate.
  - **ParametersParser.java**: responsabile della lettura dei parametri della simulazione da dei file di *config*.
  - **SimulationBuilder.java**: contiene dei metodi che tornano valori diversi a seconda del tipo di simulazione che stiamo eseguendo e del contesto della simulazione stessa (e.g. se siamo in verifica o in contesto normale).
- Classi **QueueManager**. Si tratta di classi associate a diversi tipi di centri che si occupano di implementare la politica di estrazione dalle code associate ai centri stessi.
- Classi Singleton
  - **ClockHandler.java** : singleton responsabile della gestione e dell'incapsulamento del clock di simulazione.
  - **ConfigHandler.java**: singleton responsabile dell'incapsulamento della configurazione corrente della simulazione (e.g. fasce orarie e loro parametri).
  - **EventsPool.java**: singleton responsabile della gestione delle liste dei prossimi eventi che si verificano all'interno del sistema.
  - **RandomHandler.java**: singleton usato per incapsulare l'accesso da diversi punti alla generazione di numeri casuali, l'assegnazione degli stream di numeri casuali alle classi che le generano ecc.
- Classi Event
  - **EventsPoolId.java**: rappresenta la coppia (*Centro, Tipo di Evento*) associata ad un evento, ovvero dove avviene l'evento e il suo tipo.
  - **EventType.java**: enumeratore che rappresenta il tipo di evento che avviene all'interno del sistema.
  - **SystemEvent.java**: classe che rappresenta un evento che avviene all'interno del sistema, con associato il suo tipo e il centro in cui avviene.
- Classi Job
  - **GroupPriority.java**: enumeratore che rappresenta le classi di priorità a cui un gruppo può appartenere.

- `RiderGroup.java`: rappresenta la classe Job nel suo complesso, ovvero il gruppo di persone che si muove all'interno del sistema.
- Classi Queue
  - `FifoQueue.java`: rappresenta una coda che implementa una disciplina FIFO.
  - `QueuePriority.java`: enumeratore della priorità associata ad una coda.
- Classi RoutingNode: classi che implementano la logica di selezione del prossimo centro da parte di un gruppo; sfruttano le classi Probabilities.
- Classi Probabilities: classi che calcolano per le diverse aree la probabilità di andare in un centro associate ad ogni centro di quell'area.
- Classi Server
  - `AbstractCenter.java`: rappresenta un centro astratto; contiene i metodi comuni alla gestione di tutti i tipi di centri.
  - `MultiServer.java`: rappresenta un centro astratto di tipo multi (o single) server.
  - `Attraction.java`, `Restaurant.java`, `Entrance.java`: rappresentano i centri concreti del sistema.
  - `StatsCenter.java`: classe che embedda un centro standard e che implementa il pattern decorator per raccogliere le statistiche in modo trasparente ai centri.
  - `ExitCenter.java`: centro che rappresenta l'uscita; definito per avere maggiore flessibilità nel routing.
- Classi Stats
  - `AreaStats.java`: classe per la raccolta delle statistiche con il metodo delle aree.
  - `BatchStats.java`: classe per la raccolta delle statistiche con il metodo dei batch means.
  - `CenterStatistics.java`: classe che racchiude le diverse statistiche raccolte per il singolo centro.
  - `GroupStats.java`: classe che racchiude le statistiche relative al singolo gruppo.
  - `IntervalStatsManager.java`: classe ausiliaria per la raccolta delle statistiche relative al singolo intervallo.
- Classi Utils. Si tratta di classi usate principalmente per il calcolo di statistiche o di valori teorici associati alla fase di verifica.
- Classi Random. Si tratta di classi o esempi per la generazione di numeri casuali, il calcolo dell'autocorrelazione e degli intervalli di confidenza.
- Classi Writers. Si tratta di classi usate per la scrittura su file dei risultati delle simulazioni, delle statistiche, dei valori teorici e in generale degli output prodotti dal simulatore.
- Altre Classi
  - `Parameters.java`: rappresenta i parametri relativi ad una fascia oraria.
  - `Interval.java`: rappresenta una specifica fascia oraria.
  - Constants.

### 5.3 Gestione degli Eventi e della Simulazione

In [Figura 2](#) è mostrato il class diagram relativo alla gestione degli eventi e della simulazione. Abbiamo diverse istanze che implementano l'interfaccia `Controller` a seconda del tipo di simulazione; il controller, in base a quello che deve fare eseguirà il metodo `batchSimulation` per simulazioni a orizzonte infinito o il metodo `simulateOnce` per simulazioni a orizzonte finito. All'interno della simulazione si procede per prima cosa invocando il metodo `buildNetwork` di `NetwrokBuilder` il quale:

- Costruisce i diversi centri del sistema
- Costruisce i `RoutingNode` del sistema

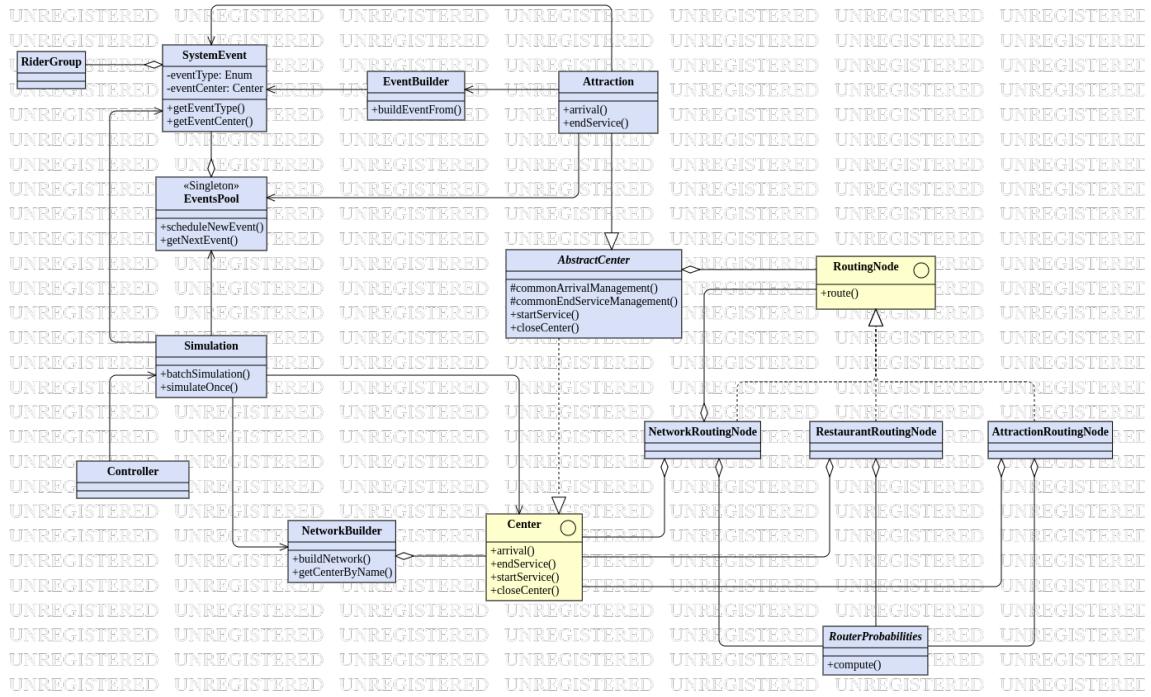


Figura 2: Class Diagram per Gestione degli Eventi e della Simulazione

- Associa ad ogni centro il suo *RoutingNode*

Questo permette di istanziare tutti gli elementi componenti il sistema.

Fatto questo si procede con un approccio di tipo *Next Event*: viene schedulato il primo evento di arrivo al sistema inserendolo nel singleton *EventsPool* e da lì in poi si continuano ad estrarre elementi fino alla condizione di terminazione della simulazione che sono le seguenti:

- Orizzonte finito: arrivati alla chiusura del centro
- Orizzonte infinito: riempiti tutti i batch delle statistiche

Ad ogni giro del ciclo *while* quindi viene invocato il metodo *getNextEvent* che estrae il prossimo evento dall'*EventsPool*; l'*EventsPool* contiene una mappa che associa ad un poolId, formato da nome del centro e tipo di evento, una lista ordinata di eventi. Per quanto riguarda la classe *SystemEvent* questa contiene il suo tipo, il nome del centro associato e il suo *RiderGroup*. Si noti che è stato necessario mantenere la classe *RiderGroup* e non semplicemente un contatore di occupazione dei centri per i seguenti motivi:

- Bisogna raccogliere statistiche relative al gruppo, come il numero di corse fatte su ogni giostra (per il *routingNode* delle attrazioni) oppure il tempo di servizio e coda nelle attrazioni per il calcolo del *FunIndex*.
- Bisogna tenere traccia della dimensione del gruppo sia per il servizio nel ristorante (modellato, come ricordiamo, come una k-erlang con k dimensione del gruppo) sia per la gestione dei numeri di posti a sedere nella giostra.
- Bisogna distinguere i gruppi in base alla loro priorità

Ottenuo il prossimo evento, se ne prende il tipo, il job coinvolto e il centro associato (tramite la *getCenterByName* di *NetworkBuilder*) e, a seconda del tipo di evento, si chiamano diverse operazioni dell'interfaccia *Center*. Consideriamo in questo caso, un evento di *END\_SERVICE*, per spiegare anche come funzionano i *RoutingNode*.

Quando viene invocata la *endService* del centro, ad esempio di un'attrazione, per un certo Job, il Job viene rimosso da una lista di job attualmente in servizio, per poi invocare il metodo *route* del *RoutingNode* associato al centro. A seconda del centro, questo avrà un tipo diverso di *RoutingNode* il quale a sua volta avrà associato un diverso tipo di *RouterProbabilities*: il metodo *compute* di *RouterProbabilities* viene usato per calcolare le probabilità associate ad ognuno dei prossimi centri; fatto questo il *RoutingNode* estrae un numero casuale in base al quale viene scelto il prossimo

centro a cui andare. Si noti che il **RoutingNode** aggrega tanto dei **Center** quanto dei **RoutingNode** e questo rende la struttura estremamente flessibile: per aggiungere altre possibili direzioni una volta usciti da un centro, sarebbe sufficiente aggiungere centri o routing node e modificare la classe di probabilità associata. Trovato il prossimo centro, viene usata la classe **EventBuilder** per costruire il prossimo evento, il quale sarà poi schedulato nell'**EventsPool** con la **scheduleNextEvent**.

Facciamo qui notare come nel caso di un'attrazione, venga fatto partire un nuovo servizio solo quando non ci sono più job in servizio nel sistema: questo equivale a far partire una nuova corsa solo quando tutti i gruppi sono scesi dalla giostra a terminazione della loro corsa.

Per quanto riguarda invece l'evento di **ARRIVAL** al centro, prima di tutto si controlla se il centro può entrare immediatamente in servizio su quel job; in caso affermativo si fa partire immediatamente il servizio tramite la **startService**, mentre, in caso contrario, il job viene accodato in attesa del suo turno. Quando viene invocata la **startService**, che sia perché è arrivato un job al sistema che può entrare subito in servizio o perché è terminato un servizio precedente, viene estratto un nuovo tempo di servizio per il prossimo servizio e schedulato un evento di **END\_SERVICE** per quel job. Si noti inoltre che nel caso delle attrazioni la gestione è leggermente diversa rispetto a quella di un servente classico: in questo caso il prossimo servizio deve essere fatto partire solo quando tutti coloro che sono correntemente in servizio sono stati rimossi dalla lista dei **currentServing**. Per quanto riguarda la simulazione ad orizzonte finito, ad ogni avanzata del tempo si verifica se è cambiata o meno la fascia oraria: in caso affermativo, si modificano i **currentParameter** nel singleton **ConfigHandler**. Qualora si sia superato l'ultimo intervallo temporale, vengono presi tutti quanti i centri ed invocata la loro **closeCenter**: questa chiamata provoca lo svuotamento delle code del centro, ridirezionando tutti i Job che si trovano nel centro usando il **RoutingNode**, e il passaggio del centro allo stato **closed**; quando il centro passa a questo stato, tutti gli arrivi a quel centro non sono aggiunti in coda, ma anche essi ridirezionati. Si fa qui notare che l'uso del meccanismo dei routing node rende il sistema molto più flessibile in quanto, sebbene in questo contesto non sia stato fatto perché non pertinente, si potrebbe modellare facilmente la chiusura dei centri in orari diversi. In questo caso invece, poiché quando si supera l'ultima fascia oraria la probabilità di uscita diventa  $p_e = 1$  mentre  $p_a = p_r = 0$ , tutti coloro che escono dal centro vengono ridiretti verso il centro di uscita.

Quando un evento viene generato e aggiunto all'**EventsPool**, questo viene associato al suo **EventsPoolId**, formato da  $(centerName, eventType)$  e aggiunto alla lista associata; una volta aggiunto, la lista associata a questo *id* viene ordinata secondo un ordinamento fatto primariamente per **eventTime** associato all'evento e secondariamente, a parità di **eventTime**, per gruppo, in modo da garantire un ordinamento totale degli eventi. L'ordinamento non viene fatto manualmente ma facendo implementare alla classe **SystemEvent** l'interfaccia **Comparable**.

## 5.4 Raccolta delle Statistiche

Un class diagram che mostra i collegamenti tra classi per la raccolta delle statistiche è mostrato in [Figura 3](#). Come accennato la raccolta delle statistiche viene gestita tramite l'uso di un pattern

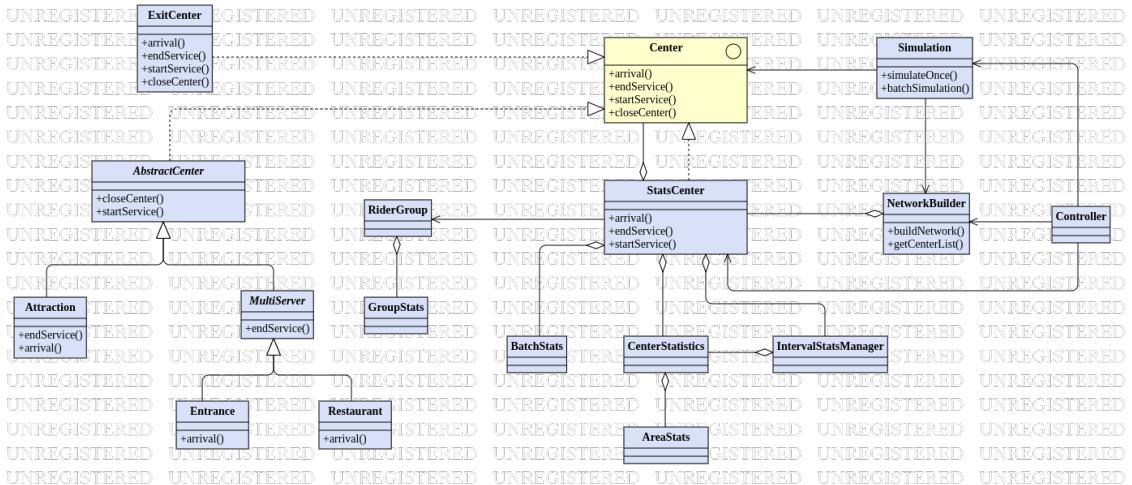


Figura 3: Class Diagram per Raccolta delle Statistiche

*Decorator*, in cui:

- **StatsCenter** rappresenta la classe *decorator*
- **Center** rappresenta la classe *component*
- **Attraction**, **Entrance**, **Restaurant** rappresentano le *concrete component*

L'uso di questo pattern permette di raccogliere le statistiche relative ai diversi centri in modo trasparente rispetto alla logica di gestione dei centri stessi; in questo modo, la logica di raccolta delle statistiche è completamente disaccoppiata dalla logica di funzionamento dei centri, garantendo che modifiche di una delle due logiche non impattino sull'altra e viceversa. Si noti inoltre che qualora esista un centro per il quale non siamo interessati, per qualsiasi motivo, alla raccolta di statistiche, il sistema continua a funzionare anche se ci sono alcuni centri che sono **StatsCenter** e altri che invece non lo sono.

Questo pattern ci permette di "intercettare" le chiamate ai metodi **arrival**, **startService** e **endService** dando la possibilità di aggiungere operazioni prima o dopo tali metodi. In questo modo lo **StatsCenter** può catturare dati per il calcolo delle statistiche in modo completamente trasparente alla logica dei singoli centri.

Essendo lo **StatsCenter** un'istanza di **Center** ed essendo concreto, chiaramente deve implementare tutti i metodi che sono implementati da classi concrete che implementano quest'interfaccia: nella maggior parte dei casi, se l'operazione non è significativa per la raccolta delle statistiche, si procede semplicemente al reindirizzamento della chiamata all'istanza di **Center** decorata. Lo **StatsCenter** mantiene un insieme di attributi per tenere traccia:

- Dell'istante in cui un job è entrato in coda
- Dell'istante in cui un job ha preso servizio
- Della priorità con cui un job è stato inserito in coda
- Dei batch associati al centro
- Delle statistiche per giorno associate al centro
- Delle statistiche per intervallo associate al centro

Analizziamo come lo **StatsCenter** si comporta alla ricezione della chiamata:

- **arrival**. Vengono raccolti dati relativi ai tempi di arrivo in coda del job, tenendo in considerazione anche la priorità della coda. Tali tempi sono usati per tenere traccia dei tempi di coda del job.
- **startService**. Quando inizia l'esecuzione del servizio di un centro, vengono raccolti dati sui job relativi ai tempi di coda e i tempi di inizio servizio, usati in seguito per il calcolo dei tempi di servizio.
- **endService**. Alla fine dell'esecuzione del job, viene raccolto il suo tempo di servizio.

Analizziamo quindi le statistiche calcolate. All'interno di **CenterStatistics** troviamo sia le statistiche per gruppo sia le statistiche per persona; inoltre queste possono essere relative alla giornata complessiva oppure alla singola fascia oraria e in quest'ultimo caso viene usata la classe **IntervalStatsManager** che associa ogni fascia alle sue statistiche.

Consideriamo come esempio di statistica time averaged il numero di job in coda  $q(t)$ ; allora abbiamo che, per definizione dell'integrale di Lebesgue:

$$\int_{[a,b]} q(t) d\mu$$

Definiamo la funzione  $\psi_i(t)$ , indicato con  $s_i$  e  $e_i$  istanti di ingresso e uscita dalla coda, come:

$$\psi_i(t) = \begin{cases} 1 & \text{se } t \in [s_i, e_i] \\ 0 & \text{altro} \end{cases}$$

Allora troviamo che:

$$q(t) = \sum_{i=1}^n \psi_i(t) \quad t \in [a, b]$$

$$\int_{[a,b]} q(t) d\mu = \int_{[a,b]} \sum_{i=1}^n \psi_i(t) d\mu = \sum_{i=1}^n \int_{[a,b]} \psi_i(t) d\mu = \sum_{i=1}^n |[a,b] \cap [s_i, e_i]|$$

Ovvero il contributo dato al valore dell'integrale da un job per l'intervallo è dato dalla quantità di tempo che il job passa in coda in quell'intervallo.

Da ciò quindi possiamo trovare, per ogni intervallo sia statistiche time-averaged sia number-averaged.

Chiaramente se l'intervallo è tutto il giorno troviamo il caso base:

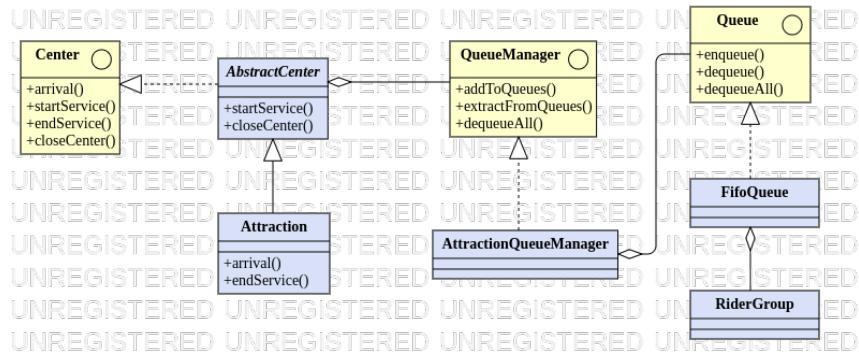
$$\int_0^\tau q(t) dt = \sum_{i=1}^n d_i$$

Il calcolo delle statistiche viene quindi fatto tenendo traccia dell'evoluzione di questo integrale sia per singolo intervallo sia per giornata intera. La classe `IntervalStatsManager` si occupa proprio di fare questo: per ogni intervallo trova l'intersezione dell'intervallo di tempo di coda (o servizio) con l'intervallo della giornata e aggiorna le statistiche in modo coerente.

Per concludere facciamo notare come il valore di  $\rho$  calcolato dalle statistiche per gruppo non sia rappresentativo dell'utilizzazione del centro nel caso di Attrazioni, in quanto il centro serve per persona e non per gruppo, come invece accade per Ristoranti e Attrazioni.

## 5.5 Gestione delle Code

Un class diagram che mostra i collegamenti tra classi per la gestione delle code è mostrato in [Figura 4](#). Per garantire flessibilità al sistema, la gestione degli inserimenti e delle estrazioni dalla



[Figura 4](#): Class Diagram per Queue Management

coda non è delegata al centro, bensì alle classi `QueueManager`. Ogni centro quindi ha il suo `QueueManager` a cui si rivolge per inserire o estrarre dalle code, garantendo trasparenza rispetto al modo in cui le code sono implementate e alle politiche di estrazione.

Il `QueueManager` è quindi responsabile della creazione delle code del centro e della loro gestione; l'interposizione del `QueueManager` garantisce che un cambio nel numero di code o nelle loro politiche di gestione non possa influenzare le implementazioni dei centri

## 5.6 Gestione dei Valori Randomici

La gestione dei valori randomici viene gestita attraverso la classe singleton `RandomHandler`. All'interno della classe troviamo una mappa che va da stringa a valori interi, dove la stringa rappresenta il nome associato al fenomeno randomico, come il tempo di servizio presso un centro oppure il routing ad un nuovo centro. Se il nome non è presente nella mappa, allora viene assegnato un nuovo stream all'evento. Questo permette di avere trasparenza tra chi usa i generatori di numeri randomici e gli stream a loro associati.

Il seed usato per inizializzare il generatore è 4321.

Gli stream utilizzati dai centri del sistema sono indicati [Tabella 3](#). Questi stream vengono assegnati nel caso dei centri e dei routing node all'atto della creazione delle istanze, mentre negli altri casi vengono assegnati staticamente alla creazione del `RandomHandler`.

Stream name	Stream number
ARRIVAL BUILDER - ARRIVAL	0
ARRIVAL BUILDER - PRIORITY	1
GROUP SIZE	2
Restaurant_1	3
Restaurant_2	4
Restaurant_3	5
Restaurant_4	6
Attraction_1	7
Attraction_2	8
Attraction_3	9
Attraction_4	10
Attraction_5	11
Attraction_6	12
Attraction_7	13
Entrance	14
AttractionRoutingNode	15
NetworkRoutingNode	17
RestaurantRoutingNode	16

Tabella 3: Stream usati all'interno della simulazione

## 6 Verifica

Al fine di confrontare il modello computazionale con il modello teorico sono state adottate le seguenti semplificazioni rispetto al modello di base:

- Gruppi formati da una singola persona:
  - Questo fa sì che il servizio k-erlang dei ristoranti e dell'ingresso si riduca ad un servizio esponenziale, cosa che ci permette di analizzare il multiserver con la legge KP.
- Senza priorità, poiché non si hanno a disposizione strumenti analitici per verificare in modo accurato la logica di estrazione introdotta nel sistema.
- Attrazioni senza servizi in burst di gruppi ma con servizi esponenziali per singolo gruppo (vale a dire per singola persona).
- Probabilità di andare alle diverse attrazioni uniforme.
- Probabilità di andare ai diversi ristoranti uniforme.

Inoltre la configurazione del simulatore è stata impostata in modo da avere un sistema stabile, al fine di renderlo confrontabile con il modello teorico.

### 6.1 Risoluzione del Sistema

Tenendo conto del feedback, il sistema di equazioni da risolvere è quello riportato in [Equazione 5](#):

$$\begin{cases} \lambda_a = (\lambda + \lambda_a + \lambda_r) \cdot p_a \\ \lambda_r = (\lambda + \lambda_r + \lambda_a) \cdot p_r \\ \lambda_e = (\lambda + \lambda_a + \lambda_r) \cdot p_e \end{cases} \quad (5)$$

Risolvendo, troviamo le equazioni riportate in [Equazione 6](#):

$$\begin{cases} \lambda_a = \frac{\lambda + \lambda_r}{k_a} \\ \lambda_r = \lambda \cdot \frac{t_a \cdot p_r}{1 - t_a \cdot p_r} \\ \lambda_e = \lambda \\ k_a = \frac{1}{p_a} - 1 \\ t_a = \frac{1}{1 - p_a} \end{cases} \quad (6)$$

Dove  $k_a$  e  $t_a$  sono funzioni della probabilità di andare alle attrazioni  $p_a$  definite per non appesantire eccessivamente la notazione.

## 6.2 Configurazione di Verifica

Partendo da questa risoluzione, è stato trovato un valore di  $\lambda$  che producesse un sistema stabile, e quindi studiabile analiticamente, in tutti i tipi di centri. Considerando le seguenti configurazioni del sistema:

- Ingresso, ( $E[S_i] = 5, m = 20$ )
- Per ogni ristorante, ( $E[S_i] = 16, m = 20$ )
- Per ogni attrazioni, ( $E[S_i] = 2, m = 20$ )
- Probabilità di instradamento ( $p_a = 0.9, p_r = 0.05, p_e = 0.05, p_{a_i} = \frac{1}{7}, p_{r_i} = \frac{1}{4}$ )

Un valore del  $\lambda$  che rendesse stabile il sistema complessivo data questa configurazione è stato trovato in  $\lambda = 3.11$ .

## 6.3 Analisi Analitica per tipo di Centro

Di seguito sono riportati i risultati dell'analisi analitica per tutti i tipi di centro; inoltre, si considerano i tempi presi in minuti.

### 6.3.1 Ingresso

$$\begin{aligned}\lambda &= 3.11 \\ E[S] &= \frac{E[S_i]}{m} = \frac{1}{4} \\ \rho &= \lambda \cdot \frac{E[S_i]}{m} = 0.7775 \\ p(0) &= \left[ \sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!(1-\rho)} \right]^{-1} = [4772715.656 + 1262272.832]^{-1} = 1.675 \cdot 10^{-7} \\ P_Q &= \frac{(m\rho)^m}{m!(1-\rho)} \cdot p(0) = 0.21143 \\ E[T_Q] &= \frac{P_Q \cdot E[S]}{1-\rho} = 0.2375 \\ E[N_Q] &= \lambda \cdot E[T_Q] = 0.73882\end{aligned}$$

### 6.3.2 Ristorante

Di seguito i valori a partire dal  $\rho$  al  $E[N_Q]$  sono relativi al singolo ristorante: avendo tutti i ristoranti la stessa configurazione, i risultati teorici di uno possono estendersi a tutti.

$$\begin{aligned}t_a &= \frac{1}{1-p_a} = 10 \\ \lambda_r &= \lambda \cdot \frac{t_a \cdot p_r}{1-t_a \cdot p_r} = 3.11 \\ \lambda_{r_i} &= \lambda_r \cdot p_{r_i} = 0.775 \\ E[S] &= \frac{E[S_i]}{m} = 0.8 \\ \rho &= \lambda_{r_i} \cdot \frac{E[S_i]}{m} = 0.622 \\ p(0) &= \left[ \sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!(1-\rho)} \right]^{-1} = [245296.4028 + 8566.2588]^{-1} = 3.93914 \cdot 10^{-6} \\ P_Q &= \frac{(m\rho)^m}{m!(1-\rho)} \cdot p(0) = 0.0337 \\ E[T_Q] &= \frac{P_Q \cdot E[S]}{1-\rho} = 0.07141 \\ E[N_Q] &= \lambda_{r_i} \cdot E[T_Q] = 0.0555\end{aligned}$$

### 6.3.3 Attrazione

Di seguito i valori a partire dal  $\rho$  al  $E[N_Q]$  sono relativi alla singola attrazione: avendo tutti le attrazioni la stessa configurazione, i risultati teorici di una possono estendersi a tutte.

$$\begin{aligned}
k_a &= \frac{1}{p_a} - 1 = \frac{1}{9} \\
\lambda_a &= \frac{\lambda + \lambda_r}{k_a} = 55.98 \\
\lambda_{a_i} &= \lambda_a \cdot p_{a_i} = 7.9971 \\
E[S] &= \frac{E[S_i]}{m} = \frac{1}{10} \\
\rho &= \lambda_{a_i} \cdot \frac{E[S_i]}{m} = 0.79971 \\
p(0) &= \left[ \sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!(1-\rho)} \right]^{-1} = [7217730.193 + 2484534.551]^{-1} = 1.03068 \cdot 10^{-7} \\
P_Q &= \frac{(m\rho)^m}{m!(1-\rho)} \cdot p(0) = 0.2561 \\
E[T_Q] &= \frac{P_Q \cdot E[S]}{1-\rho} = 0.128 \\
E[N_Q] &= \lambda_{a_i} \cdot E[T_Q] = 1.0239
\end{aligned}$$

## 6.4 Risultati della Verifica con Simulatore

Per la fase di verifica sono state fatte delle simulazioni ad orizzonte infinito con il metodo del *Batch Means*, con interruzione della simulazione quando i batch sono pieni; la dimensione dei batch è stata presa in modo che il valore di autocorrelazione tra le medie dei batch fosse inferiore a 0.2, seguendo le raccomandazioni riportate nello studio di **Banks et al. [2001]**. In particolare sono stati usati i seguenti parametri:

- Numero di Batch, 200
- Dimensione del Batch, 2048

In [Tabella 4](#) sono riportati i risultati del processo di verifica insieme con i valori di autocorrelazione tra valori medi dei singoli batch. Come si può osservare sia dalla tabella [Tabella 4](#) sia dai grafici forniti nelle sezioni successive, i valori teorici delle diverse metriche analizzate ricadono negli intervalli di confidenza al 99% prodotti: possiamo quindi dire che la verifica ha avuto esito positivo. In particolare:

- Ingresso: In [Figura 5](#) sono riportati i grafici relative alla convergenza delle diverse metriche analizzate, mentre in [Figura 6](#) è fornita una rappresentazione grafica degli intervalli di confidenza.
- Attrazioni: In [Figura 7](#) sono riportati i grafici relative alla convergenza delle diverse metriche analizzate, mentre in [Figura 8](#) è fornita una rappresentazione grafica degli intervalli di confidenza.
- Ristoranti: In [Figura 9](#) sono riportati i grafici relative alla convergenza delle diverse metriche analizzate, mentre in [Figura 10](#) è fornita una rappresentazione grafica degli intervalli di confidenza.

Center Name	Metric Name	Mean Value	Autocorrelation	Interval	Lower Bound	Theory Value	Upper Bound	Inside
Attraction'1	N'Q	1.1152	0.0397	0.2167	0.8985	1.0199	1.3319	True
Attraction'1	QueueTime	0.1352	0.0328	0.0247	0.1104	0.1275	0.1599	True
Attraction'1	Rho	0.7990	0.1595	0.0089	0.7901	0.7997	0.8078	True
Attraction'1	ServiceTime	2.0010	-0.1075	0.0085	1.9925	2.0000	2.0095	True
Attraction'2	N'Q	0.9645	0.0906	0.1434	0.8211	1.0199	1.1080	True
Attraction'2	QueueTime	0.1176	0.0874	0.0165	0.1011	0.1275	0.1341	True
Attraction'2	Rho	0.7981	0.1350	0.0084	0.7898	0.7997	0.8065	True
Attraction'2	ServiceTime	1.9969	0.0947	0.0085	1.9884	2.0000	2.0053	True
Attraction'3	N'Q	1.0198	0.1070	0.1493	0.8705	1.0199	1.1691	True
Attraction'3	QueueTime	0.1245	0.1061	0.0172	0.1073	0.1275	0.1417	True
Attraction'3	Rho	0.8002	0.1034	0.0084	0.7919	0.7997	0.8086	True
Attraction'3	ServiceTime	2.0022	-0.1221	0.0078	1.9943	2.0000	2.0100	True
Attraction'4	N'Q	1.0612	0.0862	0.1700	0.8912	1.0199	1.2312	True
Attraction'4	QueueTime	0.1290	0.0756	0.0195	0.1095	0.1275	0.1485	True
Attraction'4	Rho	0.8002	0.1585	0.0085	0.7918	0.7997	0.8087	True
Attraction'4	ServiceTime	2.0010	0.0159	0.0087	1.9923	2.0000	2.0096	True
Attraction'5	N'Q	1.0186	0.1594	0.1528	0.8658	1.0199	1.1714	True
Attraction'5	QueueTime	0.1237	0.1539	0.0175	0.1062	0.1275	0.1412	True
Attraction'5	Rho	0.8002	0.1687	0.0087	0.7916	0.7997	0.8089	True
Attraction'5	ServiceTime	1.9987	0.0063	0.0077	1.9910	2.0000	2.0065	True
Attraction'6	N'Q	0.9822	0.1210	0.1558	0.8264	1.0199	1.1380	True
Attraction'6	QueueTime	0.1198	0.1141	0.0177	0.1022	0.1275	0.1375	True
Attraction'6	Rho	0.7971	0.1762	0.0085	0.7885	0.7997	0.8056	True
Attraction'6	ServiceTime	1.9980	-0.0444	0.0086	1.9894	2.0000	2.0066	True
Attraction'7	N'Q	1.0465	0.1066	0.1574	0.8892	1.0199	1.2039	True
Attraction'7	QueueTime	0.1273	0.0959	0.0180	0.1093	0.1275	0.1452	True
Attraction'7	Rho	0.8009	0.1487	0.0083	0.7926	0.7997	0.8092	True
Attraction'7	ServiceTime	2.0007	0.0881	0.0087	1.9920	2.0000	2.0093	True
Entrance	N'Q	0.7305	0.0415	0.0874	0.6431	0.7309	0.8179	True
Entrance	QueueTime	0.2336	0.0414	0.0275	0.2061	0.2350	0.2612	True
Entrance	Rho	0.7767	0.0549	0.0046	0.7721	0.7775	0.7814	True
Entrance	ServiceTime	4.9977	0.0427	0.0209	4.9769	5.0000	5.0186	True
Restaurant'1	N'Q	0.0525	0.0405	0.0085	0.0440	0.0555	0.0610	True
Restaurant'1	QueueTime	0.0672	0.0380	0.0109	0.0563	0.0714	0.0781	True
Restaurant'1	Rho	0.6210	0.0807	0.0040	0.6171	0.6220	0.6250	True
Restaurant'1	ServiceTime	15.9867	-0.0003	0.0626	15.9241	16.0000	16.0493	True
Restaurant'2	N'Q	0.0550	-0.0426	0.0090	0.0460	0.0555	0.0639	True
Restaurant'2	QueueTime	0.0699	-0.0440	0.0112	0.0587	0.0714	0.0811	True
Restaurant'2	Rho	0.6221	0.0356	0.0042	0.6180	0.6220	0.6263	True
Restaurant'2	ServiceTime	15.9946	-0.0491	0.0645	15.9301	16.0000	16.0591	True
Restaurant'3	N'Q	0.0578	-0.0506	0.0095	0.0483	0.0555	0.0674	True
Restaurant'3	QueueTime	0.0738	-0.0528	0.0120	0.0618	0.0714	0.0858	True
Restaurant'3	Rho	0.6245	0.0358	0.0040	0.6205	0.6220	0.6285	True
Restaurant'3	ServiceTime	16.0566	-0.0168	0.0674	15.9892	16.0000	16.1240	True
Restaurant'4	N'Q	0.0518	-0.0591	0.0082	0.0436	0.0555	0.0600	True
Restaurant'4	QueueTime	0.0660	-0.0616	0.0102	0.0558	0.0714	0.0761	True
Restaurant'4	Rho	0.6211	-0.0631	0.0038	0.6173	0.6220	0.6249	True
Restaurant'4	ServiceTime	15.9871	-0.0430	0.0570	15.9302	16.0000	16.0441	True

Tabella 4: Risultati della Verifica

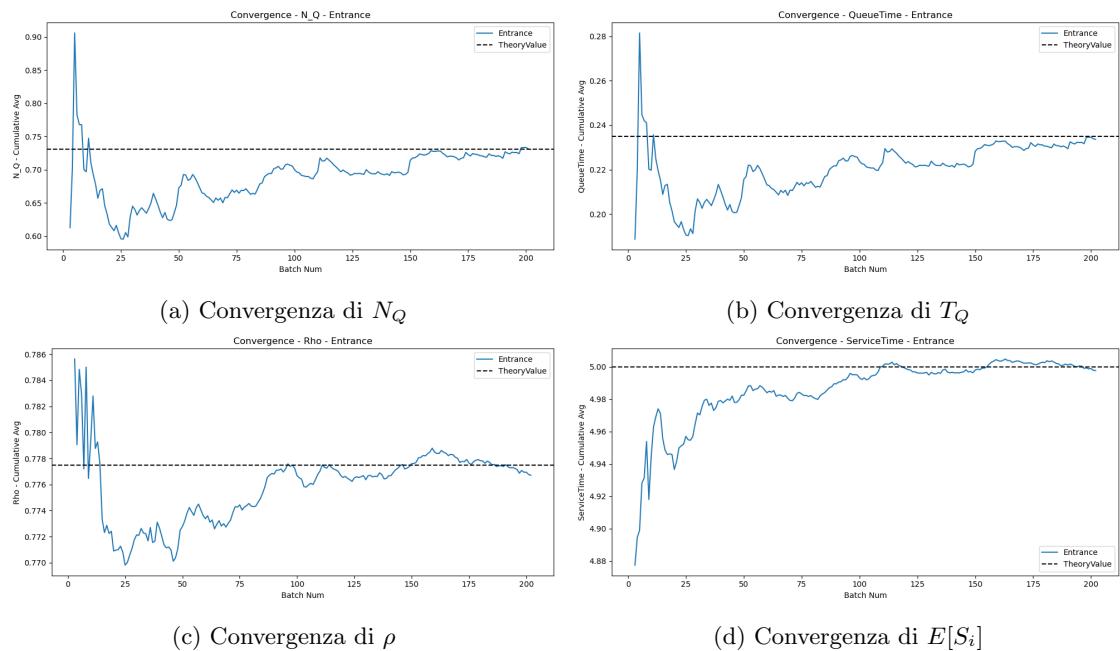


Figura 5: Verifica: Grafici di Convergenza relativi all’Ingresso

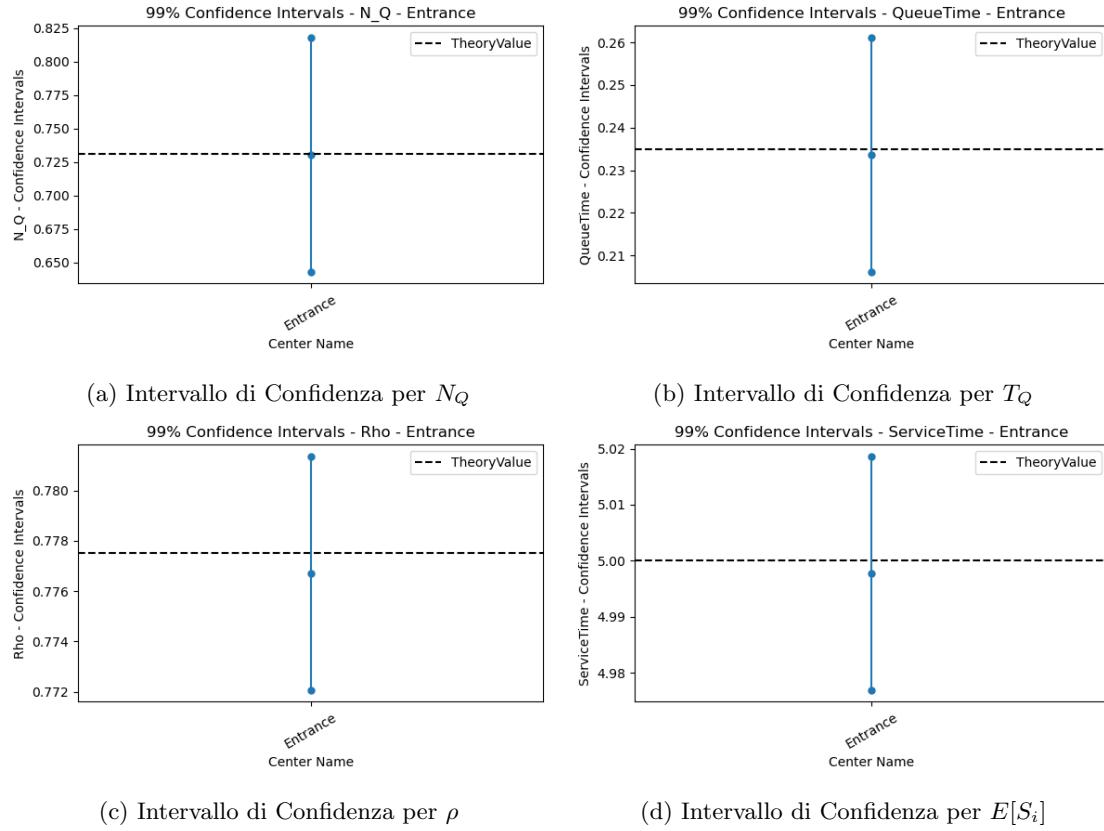


Figura 6: Verifica: Intervalli di confidenza relativi all’Ingresso

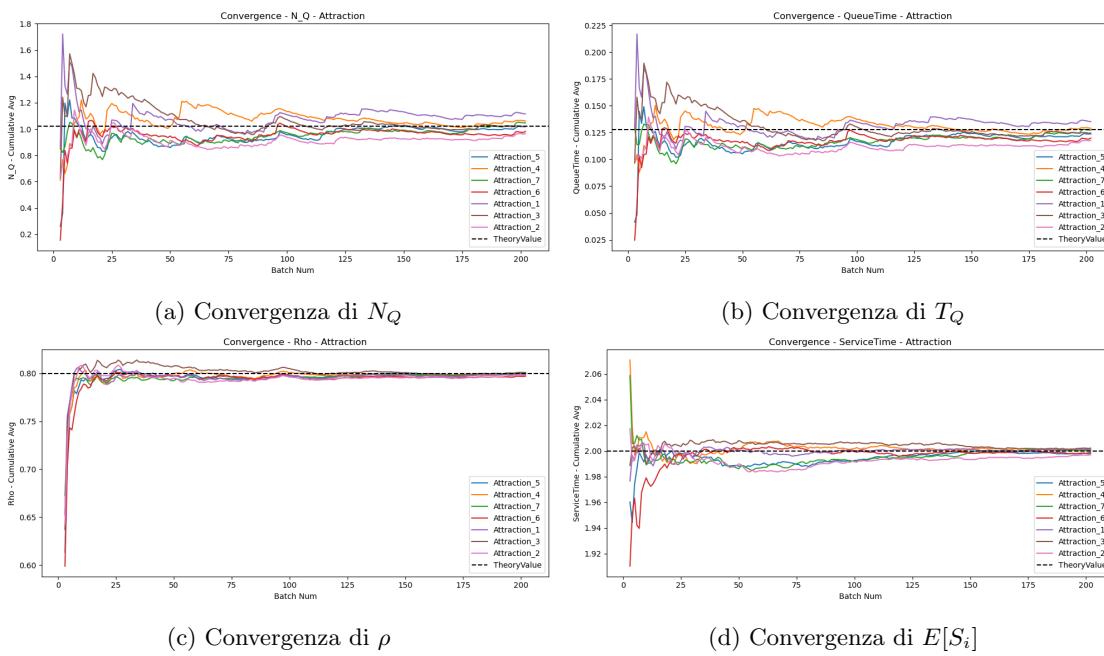


Figura 7: Verifica: Grafici di Convergenza relativi alle Attrazioni

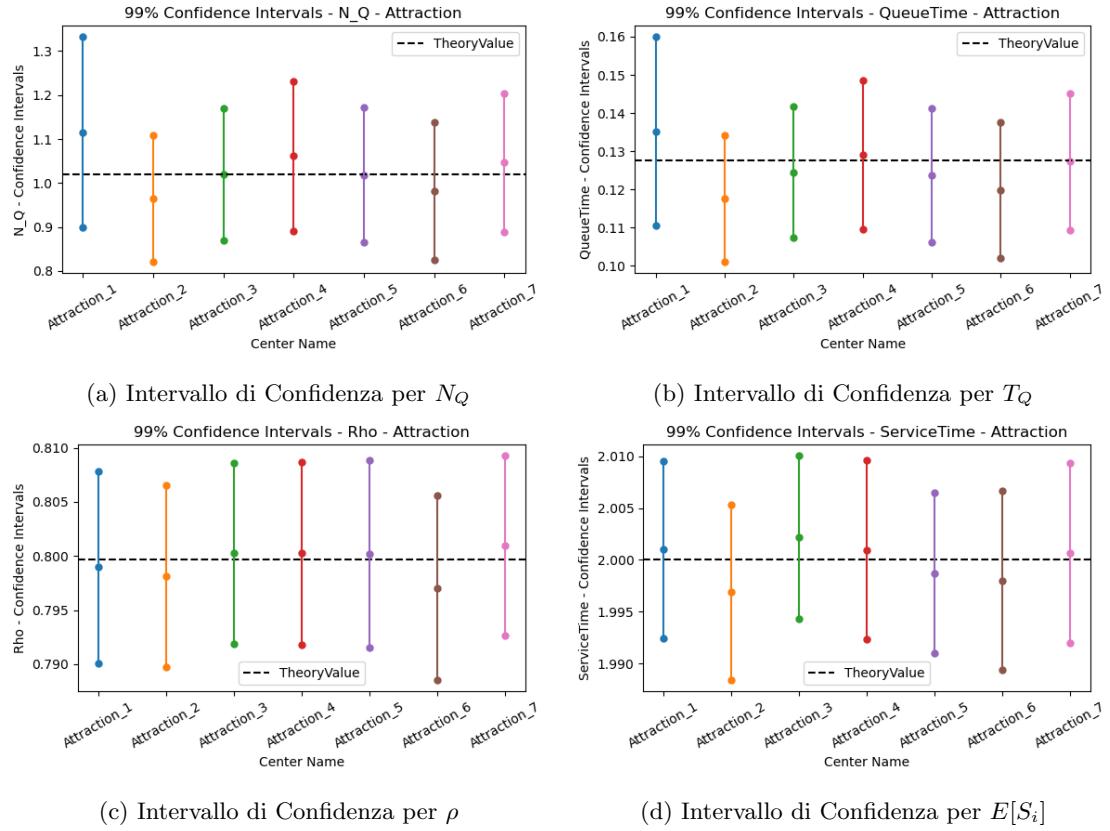


Figura 8: Verifica: Intervalli di confidenza relativi alle Attrazioni

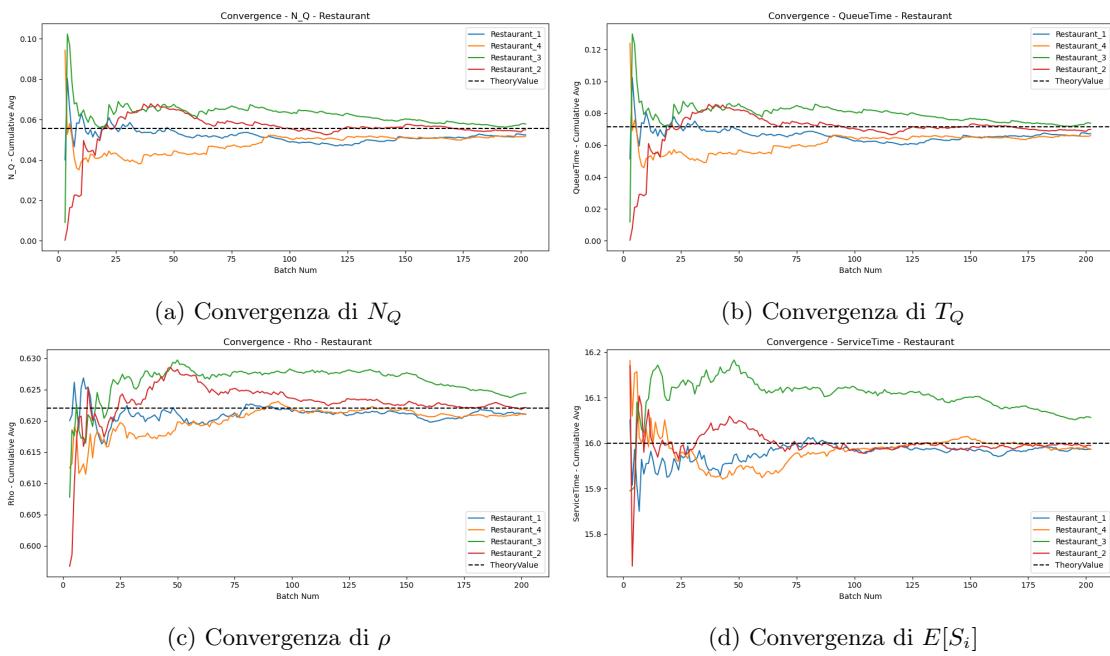


Figura 9: Verifica: Grafici di Convergenza relativi ai Ristoranti

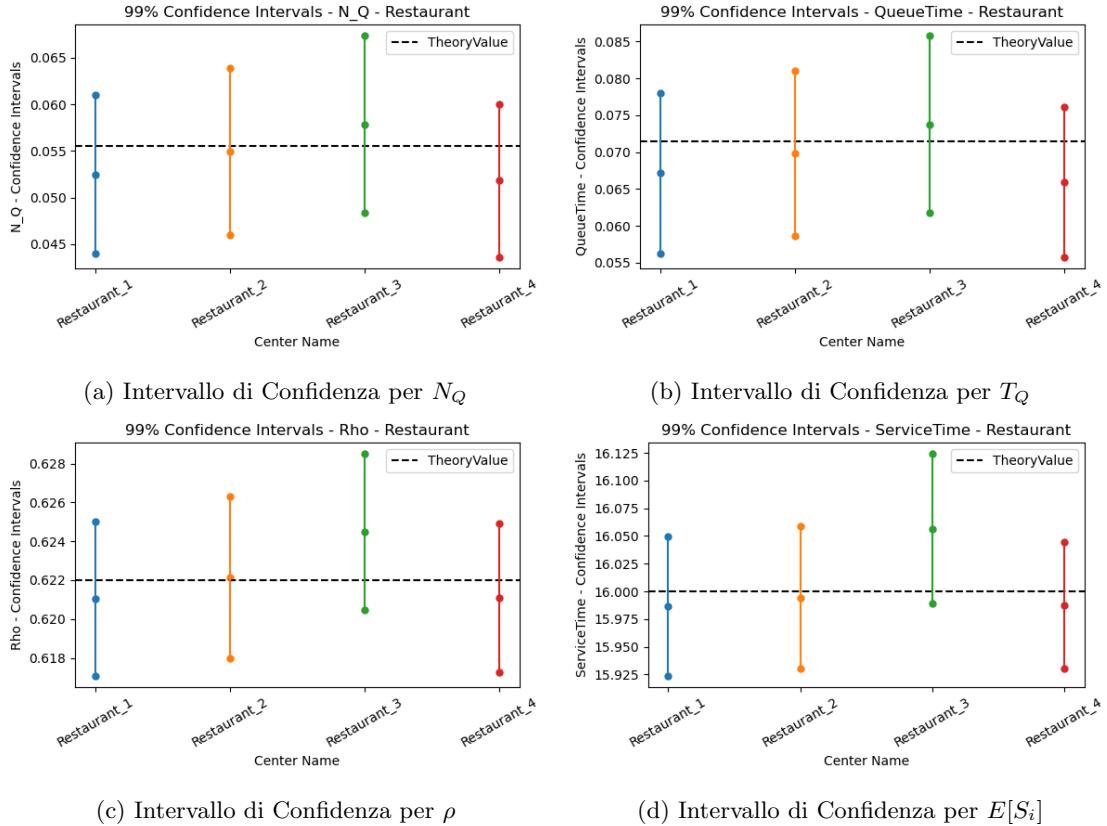


Figura 10: Verifica: Intervalli di confidenza relativi ai Ristoranti

## 7 Validazione

### 7.1 Calibrazione del Modello

Per avere un confronto con i dati reali, è stato necessario associare le giuste popolarità alle attrazioni. Ricordiamo, infatti, che il routing all'interno del parco avviene seguendo l'[Equazione 4](#). Questa calibrazione dei parametri è stata fatta iterativamente effettuando piccole modifiche alle popolarità delle attrazioni e confrontando i risultati con quelli reali riportati in [Tabella 5](#).

Name	$E[Tq]$
Velocicoaster	51
Hippogriff	43
Skull Island	40
Hulk	38
Harry Potter	37
Doom	13
Storm	7

Tabella 5: Tempi di Attesa Medi Totali Reali

Le popolarità trovate da questa fase sono indicate in [Tabella 6](#). Da sottolineare che è presente una attrazione con popolarità negativa, *Flight Hippogriff*: ciò è dovuto al fatto che questa attrazione può reggere una affluenza di visitatori molto bassa perché ha un numero di posti particolarmente limitato. Ricordiamo, inoltre, che l'[Equazione 4](#) utilizza la popolarità come esponente e che, quindi, la probabilità di andare all'attrazione non è mai in nessun modo negativa.

Come pesi nella formula di instradamento alle attrazioni sono stati usati i seguenti pesi:

- $\alpha = 5$
- $\beta = 0.2$
- $\gamma = 0.15$

Name	Popularity	AvgDuration	TotalSeats
Doctor'Doom'Fearfall	4.2	2	32
Flight'Hippogriff	-1.5	2	14
Harry'Potter'Journey	12.2	5	188
Jurassic'World'Velocoaster	11.3	3	96
Skull'Island	2.6	6	66
Storm'Force	3.6	4	60
Incredible'Hulk'Coaster	2.05	3	32

Tabella 6: Dati usati relativi alle attrazioni

## 7.2 Confronto con Valori Reali

Utilizzando le popolarità appena trovate e il metodo delle replicazioni, si hanno i risultati in [Tabella 7](#).

Center Name	E[Tq]	Interval	LowerBound	UpperBound
Skull'Island	39.421	1.769	37.651	41.190
Storm'Force	6.824	0.563	6.261	7.388
Harry'Potter'Journey	37.659	0.788	36.872	38.447
Jurassic'World'Velocoaster	51.861	0.883	50.978	52.743
Incredible'Hulk'Coaster	37.106	1.921	35.185	39.026
Doctor'Doom'Fearfall	11.770	1.216	10.554	12.987
Flight'Hippogriff	40.066	2.641	37.425	42.707

Tabella 7: Tempi di Attesa Medi Totali Simulati

I valori reali ricadono nella maggior parte dei casi negli intervalli di confidenza; i casi in cui questo non succede possono essere imputati al fatto che non sono stati modellati i tempi necessario al movimento dei gruppi da un centro al successivo.

## 7.3 Controlli di Consistenza

I controlli di consistenza sono volti a vedere se modifiche dei parametri del modello portano a cambiamenti verosimili nei risultati. Nel nostro caso è stato incrementato il tasso di arrivo.

I controlli di consistenza sono stati fatti con due simulazione ad orizzonte infinito con il metodo del Batch Means con una configurazione simile a quella della fascia oraria del pranzo, facendo variare il  $\lambda$  da  $\lambda_{pre} = 3$  a  $\lambda_{post} = 4$ . Di seguito sono riportati i risultati relativi alla crescita dei soli tempi di coda. In [Tabella 8](#) e [Tabella 9](#) sono riportati i risultati per i due valori di  $\lambda$ . I parametri del batch usati per le due simulazioni sono:

- Dimensione del Batch, 1500
- Numero di Batch, 250

Center Name	Mean Value	Autocorrelation	Interval	Lower Bound	Upper Bound
Doctor'Doom'Fearfall	4.899	0.053	0.108	4.792	5.007
Entrance	0.000	NaN	0.000	0.000	0.000
Flight'Hippogriff	5.683	-0.017	0.204	5.479	5.887
Harry'Potter'Journey	21.783	0.208	0.446	21.337	22.230
Incredible'Hulk'Coaster	7.037	0.070	0.206	6.830	7.243
Jurassic'World'Velocoaster	21.637	0.177	1.015	20.622	22.652
Restaurant'1	3783.655	0.986	374.105	3409.550	4157.760
Restaurant'2	4054.902	0.987	393.709	3661.193	4448.611
Restaurant'3	4517.200	0.987	410.057	4107.143	4927.257
Restaurant'4	3396.098	0.987	348.921	3047.177	3745.018
Skull'Island	9.754	0.007	0.199	9.555	9.953
Storm'Force	6.194	0.042	0.065	6.128	6.259

Tabella 8: Valori di  $E[T_q]$  per  $\lambda_{pre}$

In [Figura 11](#) e [Figura 12](#) sono mostrati gli andamenti dei tempi di coda al variare del  $\lambda$ .

Center Name	Mean Value	Autocorrelation	Interval	Lower Bound	Upper Bound
Doctor'Doom'Fearfall	5.987	0.033	0.153	5.834	6.140
Entrance	0.000	NaN	0.000	0.000	0.000
Flight'Hippogriff	9.022	0.115	0.504	8.517	9.526
Harry'Potter'Journey	31.793	0.688	2.143	29.650	33.936
Incredible'Hulk'Coaster	10.108	0.066	0.440	9.669	10.548
Jurassic'World'VelocoCoaster	42.855	0.690	3.782	39.073	46.637
Restaurant'1	11314.206	0.987	1098.670	10215.536	12412.876
Restaurant'2	11919.919	0.987	1131.705	10788.214	13051.623
Restaurant'3	11934.460	0.986	1121.059	10813.401	13055.520
Restaurant'4	10916.682	0.986	1057.456	9859.226	11974.138
Skull'Island	13.055	0.162	0.504	12.551	13.560
Storm'Force	6.957	0.081	0.095	6.862	7.052

Tabella 9: Valori di  $E[T_q]$  per  $\lambda_{post}$

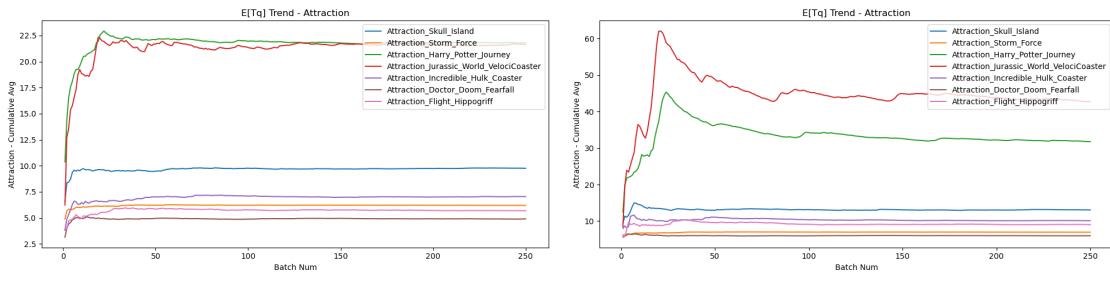


Figura 11: Andamenti di  $E[T_q]$  delle attrazioni al variare di  $\lambda$

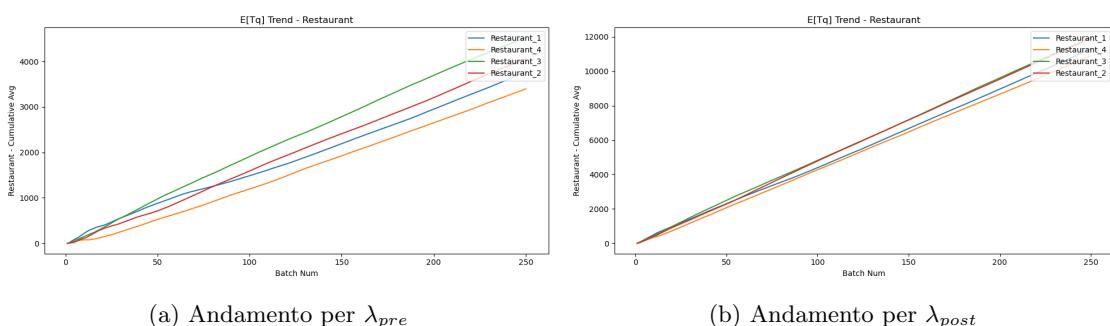


Figura 12: Andamenti di  $E[T_q]$  dei ristoranti al variare di  $\lambda$

Come si vede confrontando le due tabelle e i grafici, il valore dei tempi medi di coda cresce con l'aumento del parametro  $\lambda$ . Notiamo che nel caso dei ristoranti il  $\lambda$  scelto comporta la divergenza del tempo di coda: questo è dovuto al fatto che nella configurazione considerata gli arrivi sono alti e, essendo la simulazione ad orizzonte infinito e la probabilità  $p_r$  di andare ai ristoranti alta, si crea un accodamento che i ristoranti non sono in grado di smaltire.

Notiamo che nel caso del centro *Entrance* abbiamo un *NaN* relativo all'autocorrelazione: questo è dovuto al fatto che avendo fatto il controllo di consistenza con un numero di arrivi relativamente basso non si crea coda nel centro.

## 8 Esperimenti

Per gli esperimenti successivi fatti con il metodo delle replicazioni il numero di replicazioni considerato è 75.

### 8.1 Analisi del FunIndex

Consideriamo la definizione di *FunIndex* data nella [sezione 3](#), ovvero come:

$$\text{FunIndex} = \frac{\text{RidingTime}}{\text{AttractionQueueTime} + 1}$$

#### 8.1.1 Considerazioni

Notiamo che questa metrica può essere influenzata attraverso:

- l'aumento del numero di corse;
- l'aumento della durata delle singole corse;
- la riduzione dei tempi in coda alle singole attrazioni.

Per quanto riguarda il primo aspetto, questo può essere ottenuto banalmente aumentando il numero di attrazioni oppure il numero di posti all'interno della singola attrazione. C'è da specificare, però, che il numero di posti all'interno di un'attrazione può essere limitato superiormente da alcuni aspetti di sicurezza che a noi sono ignoti. L'aumento di posti comunque è obbligatoriamente discretizzato da un aumento del numero di veicoli.

Gli stessi aspetti di sicurezza che coinvolgono il numero di posti può in realtà anche influenzare la durata delle giostre; non avendo informazioni sugli aspetti che influenzano la durata delle giostre, questo non è considerato un parametro influenzabile in nessun modo.

Per quanto riguarda invece i tempi di coda nelle attrazioni questo è chiaramente influenzato dalla politica di estrazione dalla coda e dalla gestione delle medesime. Ricordiamo che un parametro significativo in questo senso è la percentuale di posti riservati agli utenti prioritari, con il limite per il quale almeno un gruppo prioritario viene sempre preso. Possiamo, inoltre, supporre che la percentuale di posti riservata ai prioritari sia una politica del parco e che quindi in tutte le attrazioni venga usata la stessa percentuale.

#### 8.1.2 Esperimento

Lo scopo di questo esperimento è analizzare come varia la metrica in funzione della percentuale di posti riservati ai visitatori con biglietto express. In questo modo, è possibile individuare il valore ottimale per massimizzare l'[Equazione 1](#). Le percentuali di posti riservati agli utenti prioritari considerate vanno da 0.0 a 1.0, con un incremento di 0.1.

Per evitare che la simulazione diventi troppo complessa e prolungata, la percentuale di utenti prioritari è stata mantenuta costante al 10%, valore medio rilevato in alcuni parchi divertimento simili a quello preso in esame.

#### 8.1.3 Risultati

I risultati dell'esperimento sono illustrati in [Figura 13](#). Come previsto, il FunIndex degli utenti prioritari migliora con l'aumento dei posti a loro riservati, mentre si osserva un peggioramento per gli utenti non prioritari. La tendenza si stabilizza oltre lo 0,5; tuttavia, è importante mantenere un certo livello di equità tra i gruppi prioritari e quelli normali. Pertanto, valori superiori a 0,5 possono essere esclusi.

### 8.2 Analisi dei tempi di coda

In questo secondo esperimento l'obiettivo è quello di analizzare i tempi di coda al variare del numero dei visitatori prioritari e del numero dei posti a loro riservati, in modo da poter analizzare quanti posti assegnare agli utenti prioritari per non violare i QoS per essi. Si suppone che il numero di visitatori con biglietto express sia all'incirca il 10%.

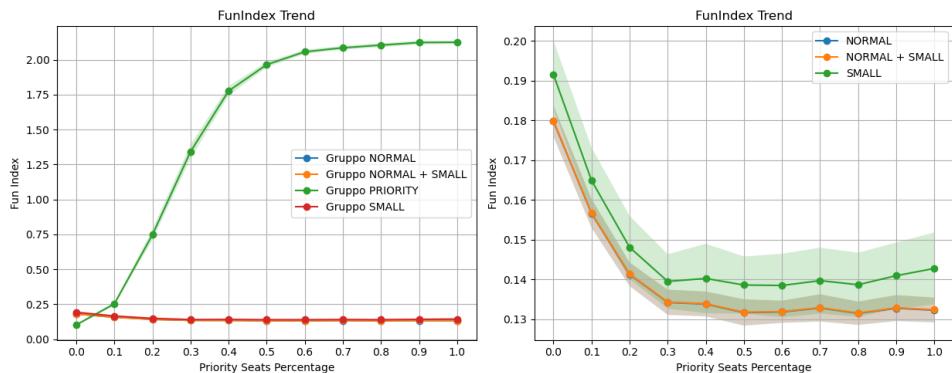


Figura 13: Andamento del Fun Index al variare della percentuale di posti prioritari

### 8.2.1 Risultati

I risultati dell'analisi dei tempi di coda sono in [Figura 14](#). Riservando almeno il 20% dei posti per i visitatori prioritari, tutte le giostre riescono a rispettare il QoS. Tenendo conto anche dei risultati dell'esperimento relativo al *FunIndex* presentati in [sottosezione 8.1](#), i valori ritenuti più significativi sono 0,3, 0,4 e 0,5. Questi saranno utilizzati nei prossimi esperimenti.

## 8.3 Analisi del numero di utenti prioritari gestibili

I risultati del precedente esperimento sono validi solo in condizioni di una giornata media, ossia quando in numero di biglietti prioritari venduti si aggira intorno al 10%. Tuttavia, nel caso in cui si decida di aumentare questo numero, potrebbe non essere più possibile garantire il QoS. Per questo motivo, in questo esperimento intendiamo analizzare quale percentuale di biglietti express può essere venduta senza compromettere il QoS.

Per raggiungere questo obiettivo, abbiamo eseguito una simulazione in cui il numero di biglietti express venduti varia tra il 10%, 20% e 30% del totale dei biglietti. Inoltre, abbiamo fatto variare anche la percentuale di posti a loro riservati tra il 30%, 40% e 50%.

### 8.3.1 Risultati

I risultati dell'esperimento sono mostrati in [Figura 15](#). Assegnare il 30% dei posti non consente di rispettare il QoS in una giornata con un elevato numero di biglietti express venduti, mentre riservarne il 40% o il 50% lo permette. Quindi, poiché un aumento eccessivo dei posti riservati penalizzerebbe allo stesso tempo i visitatori non prioritari, si è deciso di assegnare il 40% dei posti ai visitatori con biglietto express.

## 8.4 Analisi dei tempi di coda per intervallo

Anche se nel precedente esperimento i QoS sono stati sempre ampiamente rispettati, può essere utile effettuare un'analisi a grana più fine, analizzando i tempi di attesa in ogni intervallo della giornata. Infatti, tale QoS deve essere rispettato durante tutti gli intervalli. In questa simulazione è stato considerato un numero dei visitatori express pari a 10% dei totali e riservando loro il 40% dei posti.

### 8.4.1 Risultati

I risultati di questo esperimento sono mostrati in [Figura 16](#).

Il QoS è ampiamente rispettato da tutte le attrazioni in tutti gli intervalli, perciò si è scelto di analizzare i casi limite visti nella [sottosottosezione 8.3.1](#), dove si è scelto di assegnare il 40% dei posti ai visitatori con biglietto express. In questo caso, considerando che il 30% dei biglietti venduti siano prioritari, le attrazioni di *Jurassic World VelociCoaster*, *Skull Island* e *Harry Potter Journey* sono state quelle più vicine a sfornare il QoS.

I risultati di questo secondo esperimento sono in [Figura 17](#).

Dei tre casi di studio, due non rispettano il QoS durante la fascia oraria del pomeriggio, ovvero durante il periodo della giornata in cui sono presenti più persone all'interno del parco. Considerando che questo è un caso estremamente pessimistico rispetto al caso base e che i QoS non sono rispettati

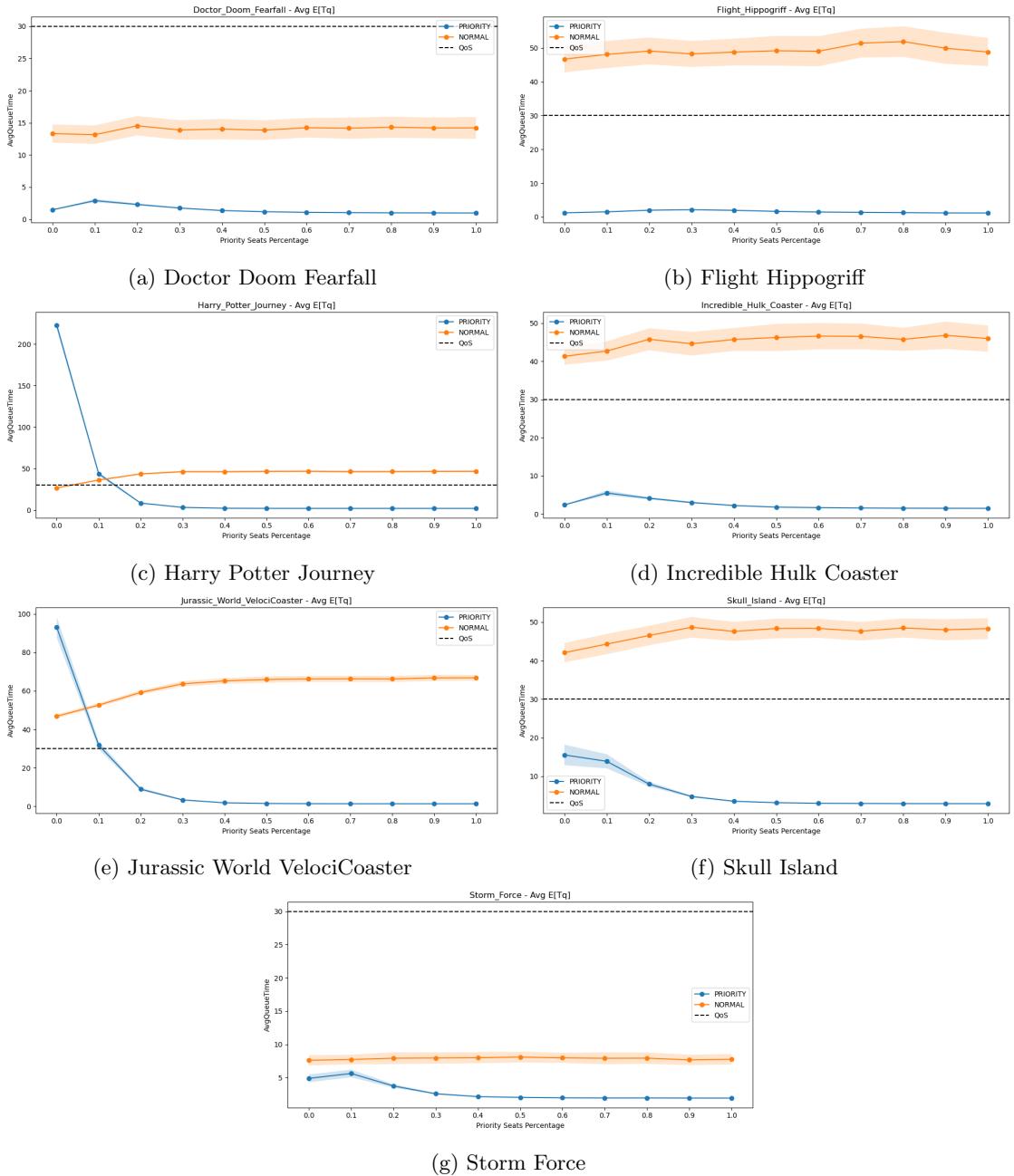


Figura 14: Tempi di coda medi giornalieri

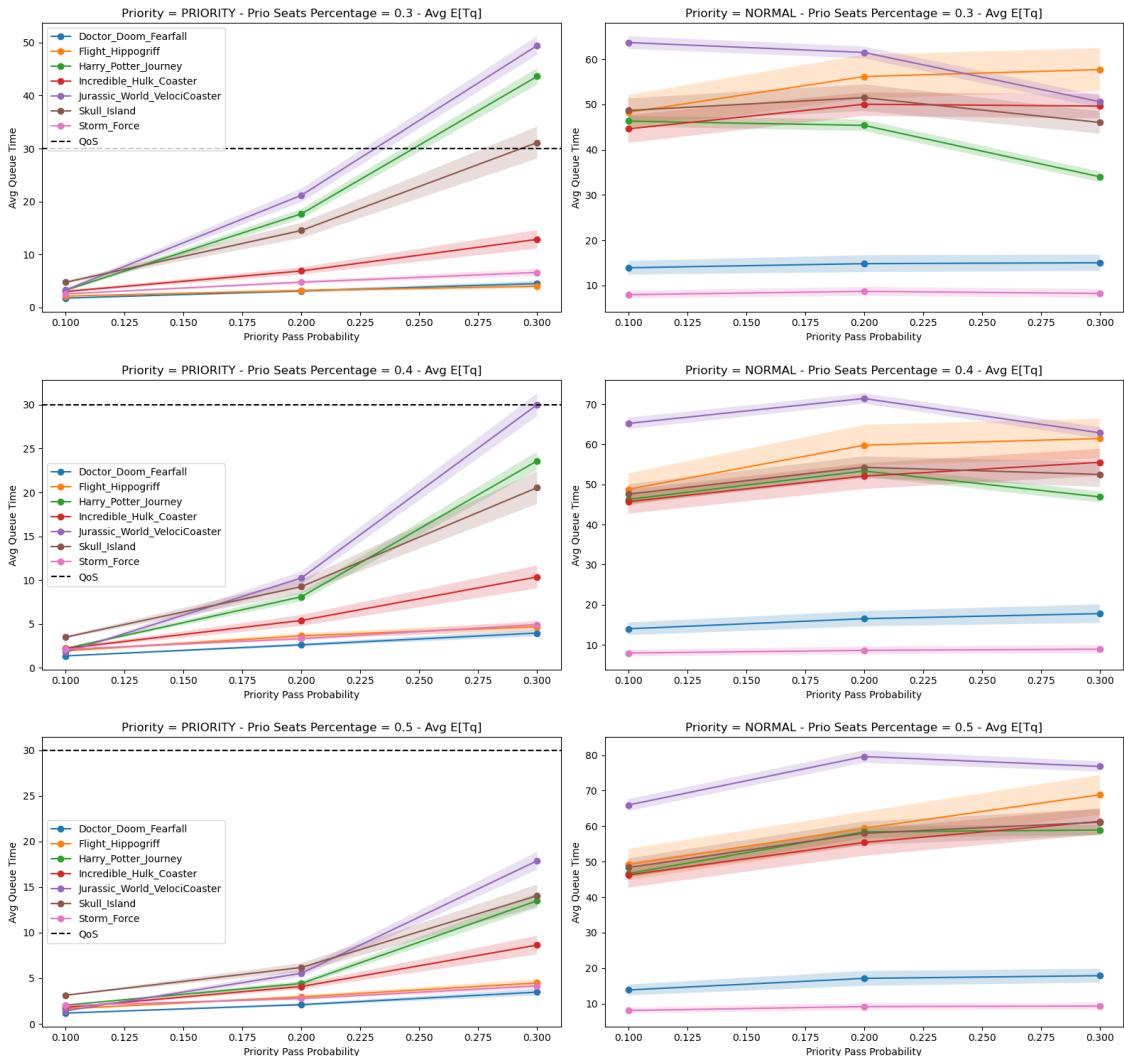


Figura 15: Tempi di coda facendo variare la percentuale di prioritari

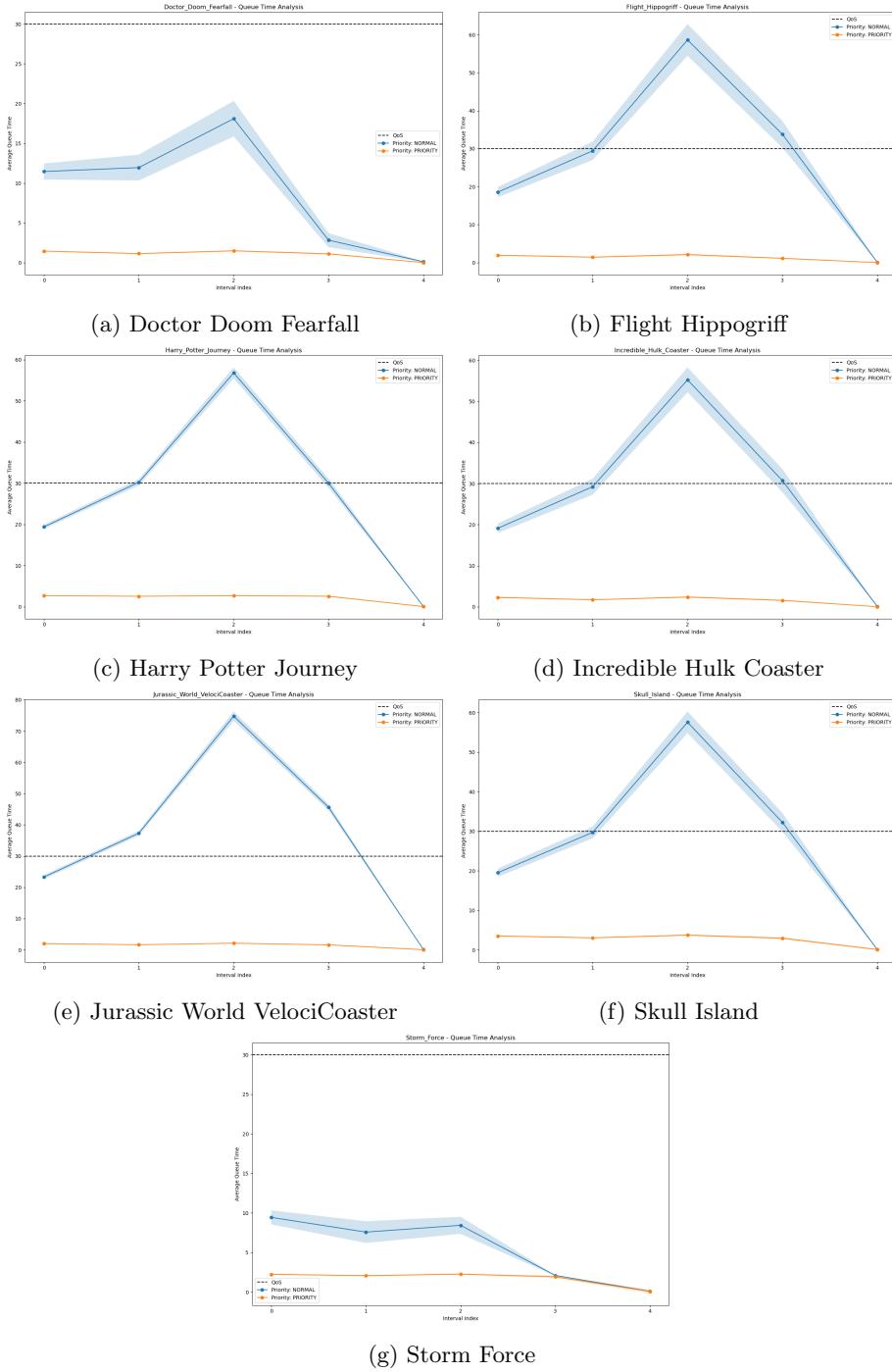


Figura 16: Tempi di coda medi durante la giornata

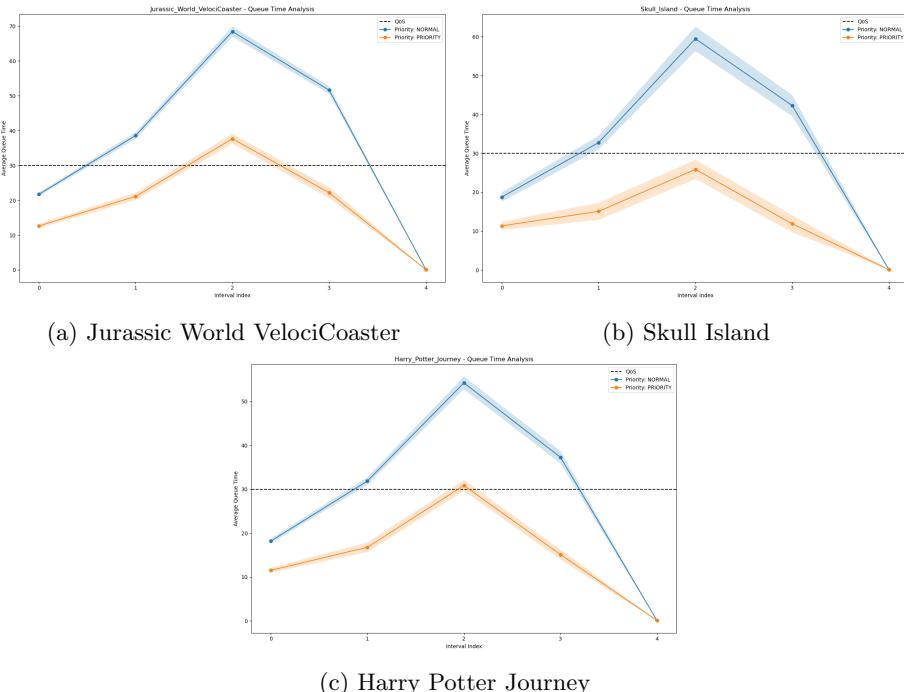


Figura 17: Tempi di coda medi durante la giornata

per pochi minuti, i risultati sono molto positivi e possiamo considerare la nostra configurazione abbastanza robusta da gestire anche casi limite come questi.

## 8.5 Analisi del Rho

Prima di pensare a possibili miglioramenti, è sicuramente utile effettuare una analisi delle utilizzazioni delle attrazioni del sistema, in modo da poter capire in quali aree bisogna intervenire. Per la simulazione è stata considerata una grandezza media dei gruppi nel parco pari a 4 e una percentuale di gruppi prioritari pari a 0.4.

### 8.5.1 Risultati

Per avere una buona ottimizzazione del parco, dobbiamo avere un  $\rho$  che sia molto vicino ad 1, in modo tale che il numero di posti sia quasi sempre utilizzato al massimo. Ovviamente, è impossibile raggiungere l'ottimo considerando che lo stato iniziale del parco è vuoto.

I risultati della simulazione sono mostrati in Tabella 10.

CenterName	MultipliedRho	Interval	Rho
Doctor'Doom'Fearfall	27.82597507718175	0.20333639233797507	0.8695617211619296
Flight'Hippogriff	11.944329376573872	0.03551101248272718	0.8531663840409909
Harry'Potter'Journey	182.85303641688586	0.11332331696633656	0.9726225341323715
Incredible'Hulk'Coaster	29.642805081569737	0.05599661161028809	0.9263376587990543
Jurassic'World'VelocoCoaster	92.88210639163485	0.06800872537066951	0.9675219415795296
Skull'Island	62.744963214776654	0.12140336731625218	0.9506812608299493
Storm'Force	51.27321790982876	0.3755120823423217	0.8545536318304793

Tabella 10: Risultati dell'analisi delle utilizzazioni

I valori di  $\rho$  sono generalmente molto elevati, con solo tre delle sette giostre che mostrano un tasso di utilizzazione inferiore al 90%. La differenza nella percentuale di utilizzazione tra le giostre è probabilmente dovuta al fatto che alcune attrazioni sono molto più popolari e, di conseguenza, attirano un maggior numero di visitatori. Inoltre, la calibrazione eseguita durante la fase di validazione, descritta in [sottosezione 7.1](#), da cui sono state derivate le popolarità delle giostre, si basava sui tempi di attesa reali che abbiamo riscontrato. Pertanto, anche i valori di  $\rho$  ottenuti possono essere considerati un risultato diretto di questa calibrazione.

## 9 Modello Migliorativo

### 9.1 Considerazioni

I risultati ottenuti dal modello di base sono stati generalmente positivi. Tuttavia, per introdurre ulteriori miglioramenti, è essenziale analizzare il sistema in modo approfondito, individuando le aree con margine di ottimizzazione. Poiché la simulazione si è concentrata sull'area delle attrazioni, l'idea è di intervenire sulle politiche di scheduling di quest'ultima, poiché rappresentano componenti sotto il pieno controllo gestionale del parco.

### 9.2 Limitazione del modello di base

Attualmente, il modello di base presenta una limitazione che impedisce di ottimizzare l'utilizzo dei serventi per ogni corsa. Questo problema è principalmente legato all'indivisibilità dei gruppi che, però, è una caratteristica del modello che non può essere modificata. Pertanto, quando i serventi sono parzialmente occupati, può accadere che un'attrazione non disponga di posti sufficienti per accogliere il gruppo successivo in coda, costringendola ad avviare il servizio con alcuni posti vacanti. Ciò è visibile anche dai risultati della [sottosezione 8.5](#), che mostra buoni valori ma sicuramente migliorabili su alcune attrazioni.

### 9.3 Obiettivi

L'obiettivo è risolvere la limitazione sopra descritta, al fine di migliorare l'utilizzazione delle attrazioni e ridurre i tempi di attesa per alcuni utenti, incrementando così la qualità del servizio. Questo miglioramento avrà, di conseguenza, un impatto positivo anche sul *FunIndex*. Pertanto, la nuova politica di scheduling deve essere progettata per ottimizzare queste metriche.

Nell'introdurre questo miglioramento è fondamentale calibrare la percentuale di estrazione dalla coda dei gruppi piccoli per mantenere la fairness nei confronti dei gruppi normali più grandi. Un altro obiettivo, quindi, è anche di mantenere pressoché invariati i tempi di attesa per i gruppi di dimensioni normali, migliorando così l'equità e l'efficacia del sistema di scheduling.

### 9.4 Politica di Scheduling Migliorativa

Per affrontare la problematica descritta, si propone l'introduzione di una coda dedicata ai gruppi di piccole dimensioni (solitamente composti da una o due persone). In questo modo, possiamo prelevare gruppi da questa nuova coda che, avendo gruppi di dimensione piccola, permette di occupare eventuali serventi non utilizzati.

Questa modifica comporta che tutti i gruppi composti da un numero di persone pari a quello dei gruppi di piccola dimensione (definito come parametro nella configurazione della simulazione) siano assegnati a una coda dedicata. Una parte dei posti dell'attrazione saranno dedicati ai gruppi di tale coda, per non rallentarli eccessivamente nel caso in cui siano molti. Quindi, oltre al miglioramento dell'utilizzazione delle attrazioni, ci si aspetta anche che questo meccanismo riduca i tempi medi di attesa per questi gruppi. Di conseguenza, i gruppi più piccoli saranno favoriti rispetto a quelli più grandi, il che è ragionevole poiché, generalmente, non è piacevole visitare un parco divertimenti da soli. Questa preferenza potrebbe incentivare maggiormente la partecipazione dei gruppi piccoli, con un potenziale incremento dei profitti complessivi del parco.

### 9.5 Esito del miglioramento

Dopo alcune simulazioni con le modifiche introdotte, si è osservato un miglioramento generale delle metriche di interesse. I risultati ottenuti saranno discussi nel dettaglio nella sezione dedicata agli esperimenti condotti sul modello migliorato.

## 10 Modello Concettuale

In Figura 18 è riportata una rappresentazione del sistema migliorativo come rete di code.

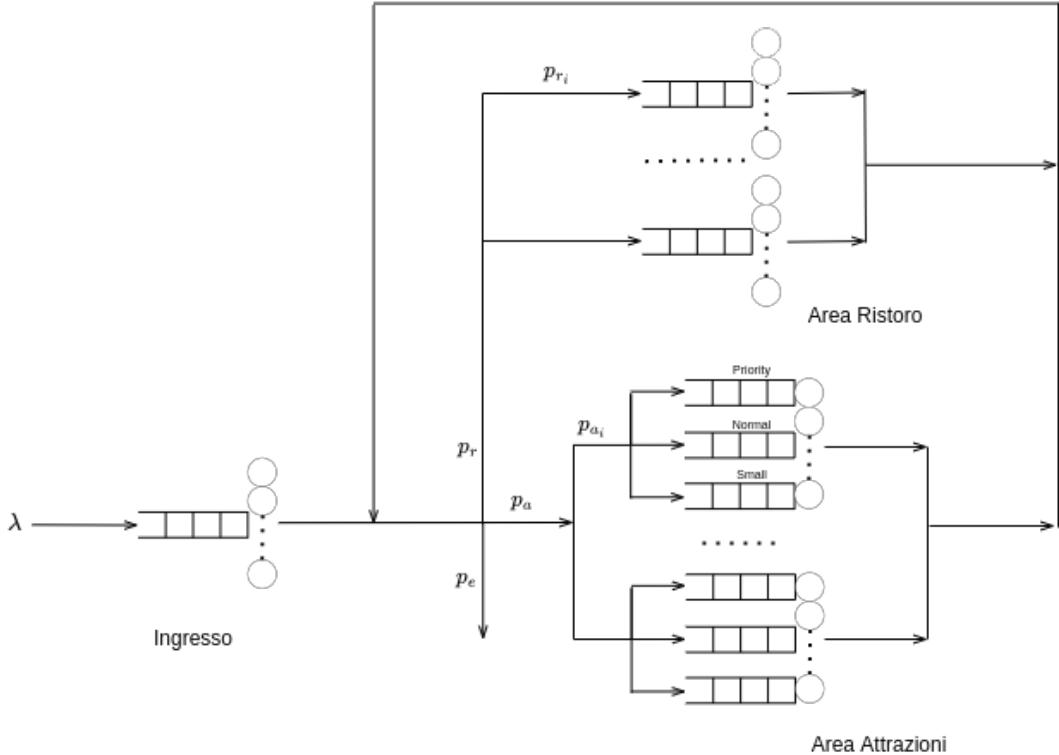


Figura 18: Diagramma Modello Concettuale

Nel diagramma sopra riportato, si noti l'aggiunta, nell'area attrazioni, di una nuova coda riservata ai gruppi di piccole dimensioni che acquistano un biglietto normale.

### 10.1 Small Rider Groups

Per un sottoinsieme di Basic Riders, nel contesto del modello migliorativo, è necessario introdurre un'altra tipologia di utenze; gli **Small Rider Groups**. Questi rappresentano gruppi di piccole dimensioni che hanno acquistato il biglietto normale.

## 11 Modello delle Specifiche

In generale, non ci sono particolari differenze nel modello delle specifiche, rispetto al modello di base.

### 11.0.1 Probabilità di Routing per Small Rider Groups

Poiché il miglioramento introdotto ha comportato l'aggiunta di una nuova coda nei centri dell'area attrazioni, il sistema deve ora gestire una nuova categoria di utenti. La probabilità di routing relativa agli *small rider groups* è influenzata dal numero di gruppi presenti nella loro coda e non dal numero di persone presenti nelle altre code: questo è principalmente dovuto al fatto che uno small group può essere preso a prescindere dalla quantità di persone presente nelle altre code proprio perché la sua utilità è quella di riempire posti vuoti nell'attrazione.

### 11.0.2 Estrazione della Coda

Di seguito, vengono descritte le fasi di estrazione dalla coda degli *small rider groups*:

1. Vengono prima estratti eventuali gruppi bloccanti dalle code normali
2. Viene estratto un gruppo prioritario

3. Viene allocata una percentuale di posti per i gruppi prioritari, estraendoli
4. Viene allocata una percentuale di posti per gli small groups, aspettando ad estrarli
5. Vengono estratti gruppi non prioritari
6. Vengono estratti gli small group considerando sia i posti avanzati sia quelli allocati in precedenza
7. Si prova a riestrarre dai prioritari in caso ci siano dei posti avanzati
8. Si prova a riestrarre dai normali in caso ci siano dei posti avanzati

## 12 Modello Computazionale

Grazie alla modularità di Java e alla flessibilità del modello di base, è stato sufficiente introdurre una nuova classe per gestire la logica di estrazione delle code, riutilizzando gran parte del codice esistente. In particolare, è stata creata la classe *ImprovedAttractionQueueManager.java*. Mentre l'estrazione dei gruppi normali più grandi e di quelli prioritari rimane invariata, è stato implementato un nuovo meccanismo per la gestione delle code dedicate ai gruppi più piccoli. A differenza del modello di base, durante l'estrazione dei gruppi normali, il numero di posti riservati agli *small rider groups* viene sottratto dai posti liberi delle attrazioni.

I parametri di configurazione per la gestione degli *small rider groups* sono i seguenti:

- *SMALL\_GROUP\_PERCENTAGE\_PER\_RIDE*: rappresenta la percentuale di *small riders* da estrarre dalla coda.
- *SMALL\_GROUP\_LIMIT\_SIZE*: rappresenta il limite superiore che identifica gli *small riders*; tutti i gruppi con una dimensione minore o uguale a questa variabile vengono considerati *small rider groups*.

Entrambi i parametri sono stati definiti nella classe *Constants*.

### 12.1 Passaggio al modello Improved

Infatti, attraverso il flag **IMPROVED\_MODEL**, sarà possibile eseguire simulazioni utilizzando la nuova logica introdotta.

## 13 Verifica

Sono state adottate le stesse semplificazioni utilizzate durante la verifica del modello di base. Come nel caso precedente, non sono disponibili strumenti analitici per verificare con precisione la logica di estrazione dalle code.

Poiché l'unica modifica apportata al modello precedente è stata l'introduzione del nuovo *QueueManager*, l'obiettivo principale è verificare il corretto funzionamento di quest'ultimo. Il tipo di simulazione e i parametri di configurazione sono rimasti invariati rispetto al modello di base; l'unica differenza è l'istanza del *QueueManager* utilizzata.

Durante la verifica, poiché i gruppi sono composti da un singolo rider, le uniche code gestite dal *QueueManager* saranno quelle relative agli *small rider groups*, diversamente dalla verifica del modello precedente. Se l'implementazione del modello migliorativo è corretta, ci si aspetta di ottenere gli stessi risultati del modello precedente, poiché la presenza esclusiva di gruppi composti da un singolo rider non dovrebbe introdurre cambiamenti significativi; la differenza è principalmente concettuale, con l'uso delle code associate agli *small rider groups*.

L'esecuzione delle simulazioni ha prodotto risultati positivi, poiché le statistiche ottenute sono risultate del tutto invariate rispetto alla [sezione 6](#). Alla luce di ciò, il nuovo modello può essere considerato verificato.

## 14 Validazione

Per la validazione del modello di consistenza sono stati eseguiti dei controlli di consistenza sui tempi di coda.

### 14.1 Controlli di Consistenza

I risultati relativi alla crescita dei tempi di coda, a seguito dell'esecuzione delle due simulazioni, all'aumentare del tasso di arrivo, vengono riportati di seguito.

Confrontando i risultati delle tabelle [Tabella 11](#) e [Tabella 12](#), si nota che anche nel seguente modello i tempi medi di coda aumentano con l'incremento del parametro  $\lambda$ .

In [Figura 19](#) e [Figura 20](#) sono mostrati gli andamenti dei tempi di coda al variare del  $\lambda$ . Possiamo notare già qui come vi sia un abbassamento dei tempi di coda rispetto al modello di base. Per quanto riguarda i parametri di configurazione si è supposta una percentuale di posti allocati agli small groups dello 0%; per il resto la configurazione è la stessa del modello di base.

Center Name	Mean Value	Autocorrelation	Interval	Lower Bound	Upper Bound
Doctor'Doom'Fearfall	4.708	0.101	0.090	4.618	4.798
Entrance	0.000	NaN	0.000	0.000	0.000
Flight'Hippogriff	5.243	-0.020	0.190	5.053	5.433
Harry'Potter'Journey	21.430	0.166	0.382	21.048	21.812
Incredible'Hulk'Coaster	6.697	0.039	0.192	6.505	6.889
Jurassic'World'VelocoCoaster	19.919	0.151	0.879	19.040	20.798
Restaurant'1	3749.693	0.985	364.710	3384.983	4114.403
Restaurant'2	4404.383	0.986	410.241	3994.142	4814.625
Restaurant'3	4260.640	0.987	398.179	3862.461	4658.819
Restaurant'4	3530.298	0.988	350.412	3179.886	3880.710
Skull'Island	9.503	0.050	0.184	9.319	9.687
Storm'Force	6.148	0.005	0.057	6.091	6.204

Tabella 11: Valori di  $E[T_q]$  per  $\lambda_{pre}$

Center Name	Mean Value	Autocorrelation	Interval	Lower Bound	Upper Bound
Doctor'Doom'Fearfall	5.987	0.033	0.153	5.834	6.140
Entrance	0.000	NaN	0.000	0.000	0.000
Flight'Hippogriff	9.022	0.115	0.504	8.517	9.526
Harry'Potter'Journey	31.793	0.688	2.143	29.650	33.936
Incredible'Hulk'Coaster	10.108	0.066	0.440	9.669	10.548
Jurassic'World'VelocoCoaster	42.855	0.690	3.782	39.073	46.637
Restaurant'1	11314.206	0.987	1098.670	10215.536	12412.876
Restaurant'2	11919.919	0.987	1131.705	10788.214	13051.623
Restaurant'3	11934.460	0.986	1121.059	10813.401	13055.520
Restaurant'4	10916.682	0.986	1057.456	9859.226	11974.138
Skull'Island	13.055	0.162	0.504	12.551	13.560
Storm'Force	6.957	0.081	0.095	6.862	7.052

Tabella 12: Valori di  $E[T_q]$  per  $\lambda_{post}$

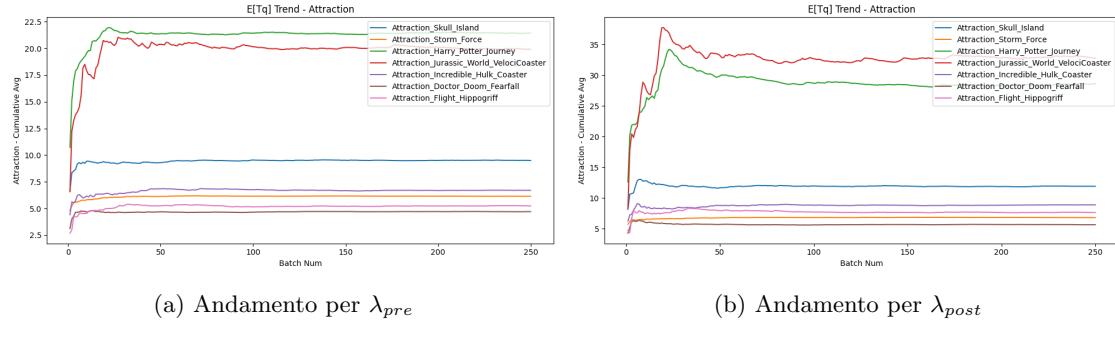


Figura 19: Andamenti di  $E[T_q]$  delle attrazioni al variare di  $\lambda$

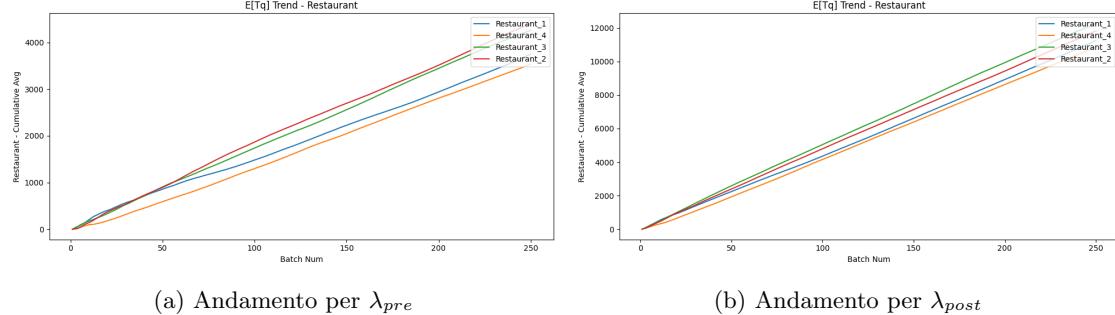


Figura 20: Andamenti di  $E[T_q]$  dei ristoranti al variare di  $\lambda$

# 15 Esperimenti

Gli esperimenti condotti sul modello migliorativo hanno un duplice obiettivo:

- Dimostrare i miglioramenti ottenuti rispetto al modello di base.
- Analizzare le metriche di interesse, variando alcuni parametri di configurazione, al fine di massimizzare la metrica in esame.

Per il momento, ci si concentrerà sulla valutazione delle varie configurazioni degli esperimenti per individuare quella ottimale. I miglioramenti rispetto al modello di base verranno discussi nella conclusione.

## 15.1 Analisi dei tempi di coda per Intervallo

A seguito del cambiamento introdotto, gli esperimenti iniziali si concentreranno sull'analisi dei tempi di coda attraverso simulazioni eseguite nell'arco della giornata. In particolare, verranno osservate le statistiche durante diversi intervalli temporali, con l'obiettivo di identificare i valori ottimali dei tempi di attesa in relazione a specifici parametri di configurazione della simulazione. I parametri coinvolti sono i seguenti:

- **smallGroupSize**: rappresenta il limite superiore per categorizzare un gruppo come *small rider group*. Ad esempio, se il parametro è impostato a 2, allora i gruppi composti da uno o due riders saranno classificati come *small rider groups*.
- **smallPercSeats**: indica la percentuale di *small rider groups* da estrarre dalla coda, introdotta nel modello migliorativo.

Per quanto riguarda la percentuale di posti prioritari da estrarre, si fa riferimento agli esperimenti condotti durante l'analisi del modello di base, decidendo quindi di utilizzare il valore fisso di 0.4.

- *smallGroupSize* è stato variato tra 1 e 2 per valutare in quali condizioni la coda degli *small rider groups* offrisse le migliori prestazioni. Non si è superato il valore di 2, poiché altrimenti la nuova coda introdotta diverrebbe troppo simile a quella degli altri utenti non prioritari con più persone.
- Inizialmente, il parametro *smallPercSeats* è stato impostato a 0 per osservare come i *single rider groups* si comportassero nell'occupare esclusivamente i posti vacanti.

### 15.1.1 Risultati

- I primi risultati ottenuti assegnando **1** alla variabile **smallGroupSize**, sono presentati in [Figura 21](#). Sono riportati i dati relativi a due delle sette attrazioni, in quanto rappresentano i casi più significativi.

Il primo grafico mostra i tempi di attesa per *Incredible Hulk Coaster* e illustra un caso in cui l'introduzione degli small rider groups risulta chiaramente vantaggiosa, poiché i loro tempi di coda sono comparabili a quelli dei possessori di biglietti express. Tutte le altre attrazioni si comportano nel medesimo modo e perciò sono state omesse, ad eccezione di *Harry Potter Journey* che viene analizzato nel grafico successivo.

Il secondo grafico, invece, rappresenta i tempi di attesa per *Harry Potter Journey* e dimostra un caso in cui l'introduzione degli small rider groups apporta vantaggi minimi o inesistenti, come evidenziato dai tempi di attesa di questi gruppi. Questo si verifica perché la giostra ha una capacità molto elevata, con 188 posti disponibili, il che fa sì che il numero di small rider groups che possono accedervi sia sproporzionalmente basso. Poiché questi gruppi non hanno posti riservati, quando l'attrazione accumula una fila considerevole, vengono utilizzati solo per riempire gli ultimi posti vacanti, se i gruppi normali e prioritari successivi sono troppo numerosi. Di conseguenza, vengono inseriti uno o due small rider groups per ciclo di servizio, ma questo non è sufficiente per un'attrazione ad alta affluenza come *Harry Potter Journey*.

- I risultati ottenuti assegnando il valore **2** a **smallGroupSize** sono illustrati in [Figura 22](#). Come nel caso precedente, vengono riportati solo i dati delle attrazioni più significative, le stesse considerate in precedenza. Come previsto, il comportamento è simile al caso precedente. Tuttavia, i tempi di attesa degli *small rider groups* sono aumentati considerevolmente, poiché il numero di gruppi di dimensione 2 è decisamente più elevato, causando una maggiore

affluenza alla coda degli *small rider groups*. Nonostante ora i gruppi di 2 persone siano direzionati verso la coda degli *small rider groups*, non si osserva un miglioramento significativo nei tempi di attesa dei gruppi normali. Questi risultati suggeriscono una preferenza per la configurazione precedente. Tuttavia, è importante notare che in queste configurazioni non sono stati riservati posti agli *small rider groups*, il che complica lo smaltimento della coda, soprattutto per *Harry Potter Journey*.

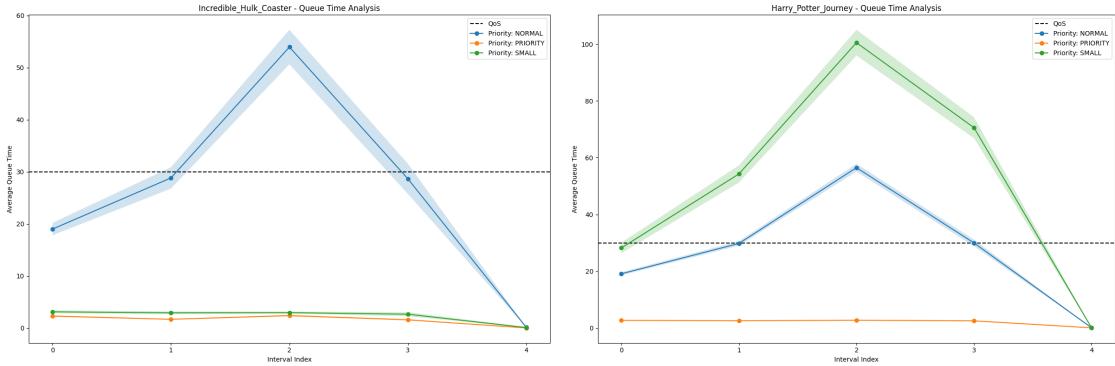


Figura 21: Tempi di attesa medi durante gli intervalli della giornata per  $\text{smallGroupSize} = 1$

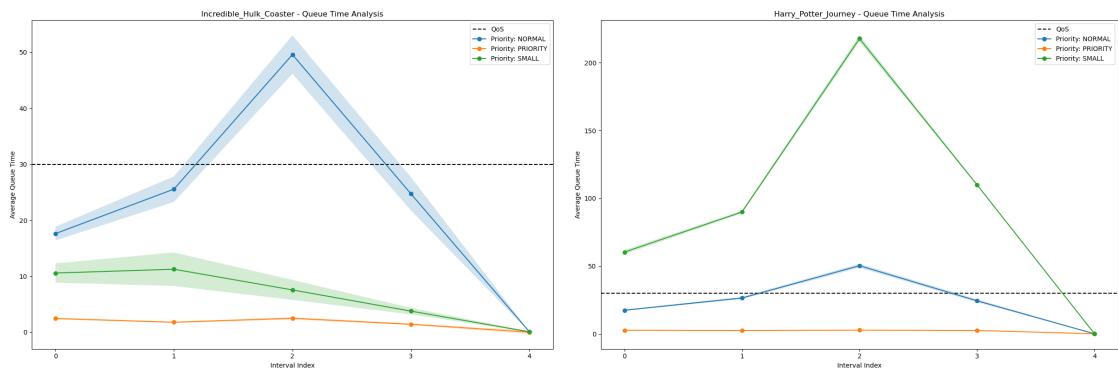


Figura 22: Tempi di attesa medi durante gli intervalli della giornata per  $\text{smallGroupSize} = 2$

### 15.1.2 Variazione del numero di posti riservati

Alla luce di quanto descritto, per migliorare i risultati è opportuno variare il numero di posti riservati agli *small rider groups*, in particolare per evitare comportamenti come quelli osservati in *Harry Potter Journey*. La coda destinata ai gruppi più piccoli deve risultare vantaggiosa per loro, altrimenti non ne usufruirebbero. Per questo motivo, si propone di riservare alcuni posti agli *small rider groups* per smaltire più rapidamente il loro traffico. Questo approccio favorirà anche una maggiore distribuzione degli *small rider groups* tra le attrazioni, aumentando la capacità complessiva di riempimento.

Variando il numero minimo di posti riservati agli *small rider groups* per entrambe le configurazioni, si è osservato che il numero ottimale è pari a tre per la prima configurazione e a 18 per la seconda. Questo risultato è ragionevole, considerando che la percentuale di gruppi da 2 è significativamente più alta rispetto a quelli da 1. Di conseguenza, la percentuale di estrazione dalla coda che ottimizza i tempi di attesa è risultata essere del 1,7% per la prima configurazione e del 10% per la seconda. I miglioramenti descritti sono riportati in [Figura 23](#).

Questi risultati sono considerati i più adatti poiché, nelle altre configurazioni, i miglioramenti nei tempi di coda sono marginali e comportano un peggioramento per i gruppi normali. La configurazione con  $\text{smallGroupSize}$  pari a 1 sembra offrire le prestazioni migliori, con tempi di coda significativamente ridotti e tempi di attesa per i gruppi normali pressoché invariati. Tuttavia, prima di trarre conclusioni definitive, è opportuno esaminare anche le altre metriche.

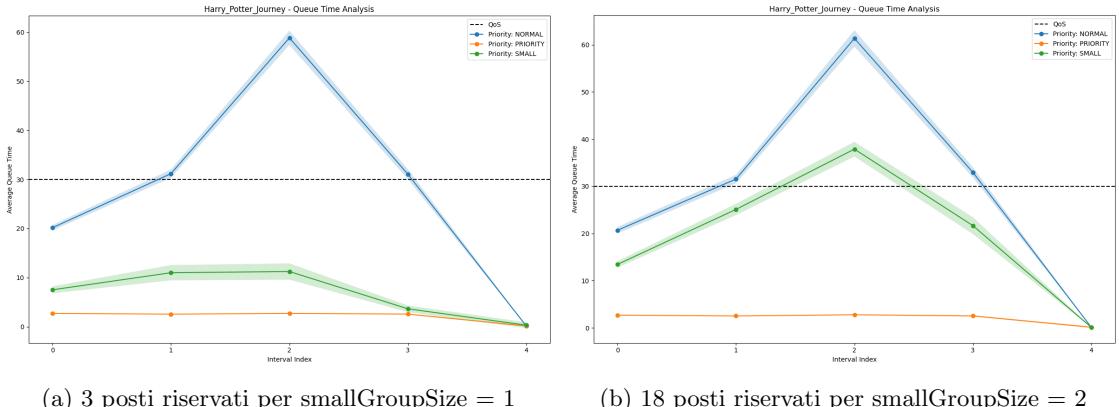


Figura 23: Tempi di attesa medi durante gli intervalli della giornata

## 15.2 Analisi dei tempi di coda

Risulta interessante analizzare anche i tempi medi di coda sull'intera giornata, soprattutto alla luce dei risultati ottenuti dall'esperimento precedente. In particolare, è possibile studiare l'andamento dei tempi medi di coda durante tutta la giornata variando il parametro **poissonLambda**, ovvero il parametro della distribuzione di Poisson che indica il numero medio di riders che compongono un gruppo in arrivo al sistema. La sua funzione è quella di modulare la probabilità che un evento di arrivo sia costituito da small rider groups, consentendo un'analisi più approfondita dell'impatto di questi gruppi sulla modifica introdotta.

Quindi, per i parametri di interesse, l'esperimento è stato condotto con i seguenti valori:

- *poissonLambda* è stato fatto variare tra 1 e 3 con un passo di 0.5. Di conseguenza, nelle prime iterazioni, la simulazione genererà diversi eventi con gruppi di piccole dimensioni.
- *smallGroupSize* è stato fatto variare tra 1 e 2, coerentemente con l'esperimento precedente.
- *smallPercSeats* è stato fissato al 1.7% per *groupSize* = 1, e al 10% per *groupSize* = 2, grazie ai risultati ottenuti nell'esperimento precedente.

### 15.2.1 Risultati

Dai risultati, mostrati in [Figura 24](#) per *smallGroupSize* pari a 1 e in [Figura 25](#) per *smallGroupSize* pari a 2, emerge che:

- Con *smallGroupSize* pari a 1, è possibile tollerare un valore di *poissonLambda* compreso tra 2.5 e 3, mantenendo i tempi di coda medi giornalieri inferiori rispetto a quelli dei gruppi normali. L'unica eccezione è rappresentata da Storm Force, che mostra un lieve aumento dei tempi di attesa. Tuttavia, questi incrementi sono marginali e limitati a questa specifica attrazione, pertanto possono essere considerati trascurabili.
- Con *smallGroupSize* pari a 2, si ottengono risultati analoghi. Anche in questo caso, valori di *poissonLambda* tra 2.5 e 3 rimangono accettabili, benché un numero maggiore di attrazioni mostri tempi di coda medi leggermente più alti rispetto ai gruppi normali. Questo comportamento è comprensibile, poiché l'aumento del numero di *small rider groups* rende la differenza con i tempi di attesa dei gruppi normali sempre più sottile.

Valori di  $\lambda$  inferiori a 2.5 portano a tempi di coda troppo elevati, a causa della generazione di un numero significativamente maggiore di gruppi di piccole dimensioni.

Alla luce di questi risultati, il valore 1 per il parametro *smallGroupSize* appare sempre più come la scelta ottimale, confermando quanto emerso nell'esperimento precedente.

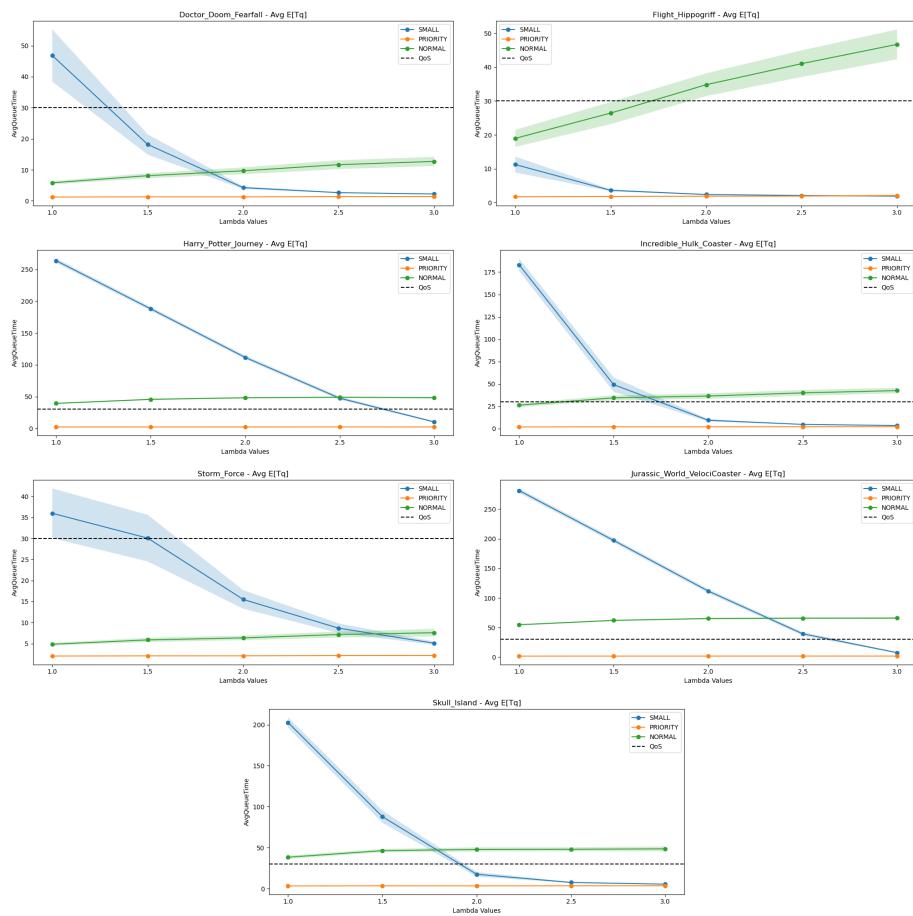


Figura 24: Tempi di attesa medi durante sull’intera giornata per smallGroupSize = 1

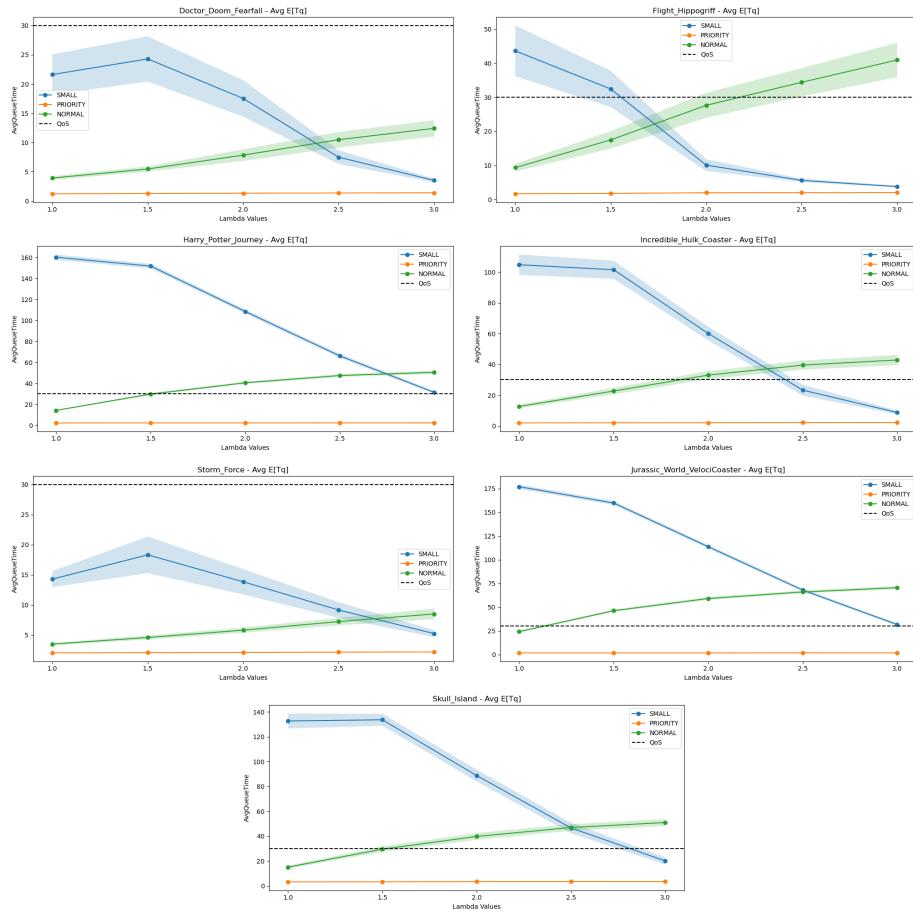


Figura 25: Tempi di attesa medi durante sull'intera giornata per smallGroupSize = 2

### 15.3 Analisi del Rho

Lo scopo principale del presente esperimento è valutare i miglioramenti nell'utilizzazione rispetto al caso base, dimostrando così l'efficacia dell'introduzione della nuova coda per coprire i serventi vacanti. Poiché l'ultima sezione è dedicata specificamente al confronto tra i risultati dei due modelli, in questa fase ci si concentra solo sulla configurazione dell'esperimento e sulla discussione dei risultati ottenuti. Inoltre, è anche interessante confrontare i risultati dell'utilizzazione tra l'impiego di *smallGroupSize* pari a 1 e 2. Anche in questo esperimento vengono utilizzati i risultati migliori ottenuti nel primo esperimento, che consentono di selezionare i valori ottimali per il numero di posti da riservare agli *small rider groups*.

- Per *smallGroupSize* = 1 sono stati analizzati i valori di rho per una percentuale di posti da riservare pari al 1.7% (come definito nel primo esperimento) e, per osservare come il sistema si comporta quando lo scheduling agisce solo per coprire eventuali posti avanzati, lo 0%.
- Similmente, per *smallGroupSize* = 2, sono stati riservati il 10% e lo 0%.

#### 15.3.1 Risultati

Come si evince dalle tabelle in figura Tabella 13, è possibile trarre le seguenti conclusioni:

- **Confronto per *smallGroupSize* = 1 con *smallPercSeats* = 0 e *smallPercSeats* = 0.017:**  
Le differenze tra i due casi non sono particolarmente significative. I valori medi variano leggermente, ma considerando gli intervalli di confidenza, queste differenze risultano trascurabili.
- **Confronto per *smallGroupSize* = 2 con *smallPercSeats* = 0 e *smallPercSeats* = 0.1:**  
Anche in questo caso, i risultati mostrano piccole differenze nei valori medi, che possono essere considerate trascurabili se si tengono in conto gli intervalli di confidenza.
- **Confronto tra *smallGroupSize* = 1 e *smallGroupSize* = 2 alle rispettive percentuali ottimali:**  
Le differenze tra le configurazioni con *smallGroupSize* = 1 e *smallGroupSize* = 2, ciascuna alle rispettive percentuali ottimali, sono anch'esse trascurabili, confermando la consistenza dei risultati ottenuti.

SmallGroupSize	SmallPercentageSize	CenterName	MultipliedRho	Interval	Rho
1	0.0	Doctor'Doom'Fearfall	28.085695039935715	0.1837941333785139	0.8776779699979911
1	0.0	Flight'Hippogriff	12.071584669279357	0.04075526161651221	0.8622560478056683
1	0.0	Harry'Potter'Journey	184.49741987170378	0.11216256243460897	0.9813692546367222
1	0.0	Incredible'Hulk'Coaster	29.91236587971179	0.08727046959993595	0.9347614337409934
1	0.0	Jurassic'World'VelocoCoaster	94.03451210374678	0.0821441748128505	0.9795261677473622
1	0.0	Skull'Island	63.45393141340765	0.10234833503007745	0.9614232032334493
1	0.0	Storm Force	51.78265254568725	0.35711402803035414	0.8630442090947875
2	0.0	Doctor'Doom'Fearfall	28.11714102708601	0.1838238132623537	0.8786606570964378
2	0.0	Flight'Hippogriff	12.130663327027838	0.042968752993829795	0.8664759519305598
2	0.0	Harry'Potter'Journey	184.17869823890703	0.13015312319172068	0.9796739268026969
2	0.0	Incredible'Hulk'Coaster	30.16401268807829	0.09575239590852516	0.9426253965024466
2	0.0	Jurassic'World'VelocoCoaster	94.1867701838452	0.05692537979125588	0.9811121894150542
2	0.0	Skull'Island	63.97792878667158	0.13253585154283767	0.9693625573738118
2	0.0	Storm Force	51.793340610007206	0.359842570445007	0.8632223435001201
1	0.017	Doctor'Doom'Fearfall	28.144497689826427	0.17762079035209585	0.8795155528070758
1	0.017	Flight'Hippogriff	12.15168924601431	0.04707133704368268	0.8679778032867365
1	0.017	Harry'Potter'Journey	183.98319073514205	0.13569860298614692	0.9786339932720322
1	0.017	Incredible'Hulk'Coaster	30.143177157986635	0.07115158255062601	0.9419742861870823
1	0.017	Jurassic'World'VelocoCoaster	94.0137660021434	0.08326641436112368	0.9793100625223271
1	0.017	Skull'Island	63.3719814313275	0.09794731435059183	0.9601815368382954
1	0.017	Storm Force	51.783543418893046	0.3767623862997739	0.8630590569815507
2	0.1	Doctor'Doom'Fearfall	28.28971970805101	0.21093350903176644	0.884053740876594
2	0.1	Flight'Hippogriff	12.398550504793118	0.07158688007643381	0.8856107503423656
2	0.1	Harry'Potter'Journey	184.04439256869225	0.11201475683309035	0.9789595349398523
2	0.1	Incredible'Hulk'Coaster	30.382524420603318	0.11077013526964755	0.9494538881438537
2	0.1	Jurassic'World'VelocoCoaster	94.12756911064743	0.0537784108615011	0.9804955115692441
2	0.1	Skull'Island	63.87548297482859	0.09485068212854807	0.9678103481034636
2	0.1	Storm Force	51.909254156490164	0.37774226184468984	0.8651542359415028

Tabella 13: Valori di *Rho*

In sintesi, non si evidenziano miglioramenti rilevanti tra le diverse configurazioni.

## 15.4 Analisi del FunIndex

Poiché il *FunIndex* risente negativamente di tempi di coda elevati e di una bassa utilizzazione, si procederà a valutare questo indice utilizzando le migliori configurazioni identificate finora. L'obiettivo è confrontare le opzioni con *smallGroupSize* pari a 1 e 2, al fine di determinare quale offre il valore migliore. I risultati preliminari suggeriscono che la configurazione con *smallGroupSize* pari a 1 potrebbe essere la più vantaggiosa. Inoltre, si vuole dimostrare il miglioramento rispetto al caso di base.

L'esperimento è stato condotto con i seguenti parametri:

- *poissonLambda* è stato variato tra 1 e 3, con un incremento di 0.5, per valutare quale calibrazione della generazione dei gruppi produca un *FunIndex* accettabile.
- Per *smallGroupSize* = 1, sono stati riservati prima lo *0%* dei posti disponibili, poi l'*1.7%*.
- Per *smallGroupSize* = 2, sono stati riservati lo *0%* e successivamente il *10%* dei posti disponibili.

### 15.4.1 Risultati

I risultati ottenuti sono illustrati in [Figura 26](#) per *smallGroupSize* = 1 e in [Figura 27](#) per *smallGroupSize* = 2. I grafici sulla destra si concentrano sul *FunIndex* escludendo i gruppi prioritari, poiché, indipendentemente dalla configurazione, per loro non si osservano variazioni significative: lo scheduler li estrae sempre per primi, seguendo la loro percentuale. Di conseguenza, è più interessante analizzare il *FunIndex* degli utenti normali e degli *small rider groups*. La curva "NORMAL + SMALL" rappresenta il *FunIndex* medio ponderato dei gruppi normali e degli *small rider groups* in base al numero di persone. L'obiettivo è individuare la configurazione ottimale che massimizzi il *FunIndex* sia per gli *small rider groups* che per la media di entrambi i tipi di gruppi.

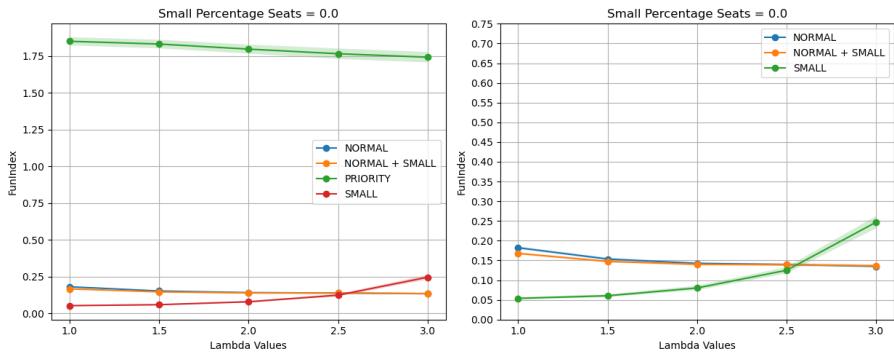
Dai grafici, emerge quanto segue:

Per entrambi i valori di *smallGroupSize*, i valori accettabili di *poissonLambda* si trovano tra 2.5 e 3, poiché al di fuori di questo intervallo, il *FunIndex* degli *small rider groups* è inferiore alla media con i gruppi normali.

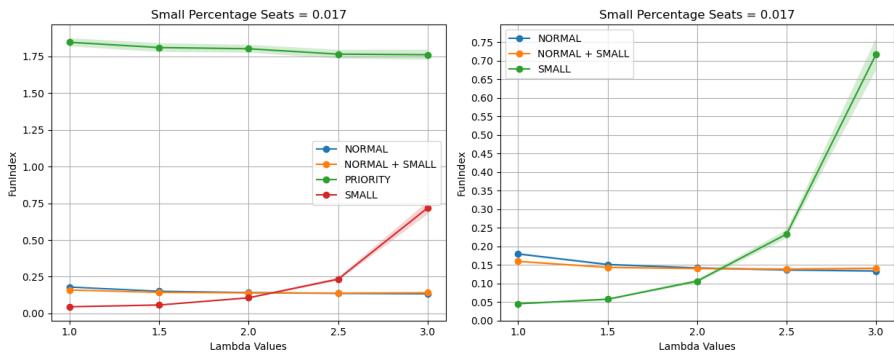
- Per *smallGroupSize* = 1, la configurazione che riserva l'1.7% dei posti agli *small rider groups* è la migliore. Sebbene la curva media con i gruppi normali rimanga pressoché invariata rispetto al caso dello 0%, gli *small rider groups* mostrano un incremento del *FunIndex*.
- Per *smallGroupSize* = 2, anche la configurazione che riserva il 10% dei posti agli *small rider groups* risulta ottimale. In questo caso, l'aumento del *FunIndex* per gli *small rider groups* è evidente, mentre la media con i gruppi normali rimane sostanzialmente invariata.

Confrontando le configurazioni ottimali per i due diversi *smallGroupSize*, quella con *smallGroupSize* pari a 1 risulta la più efficace nel massimizzare il *FunIndex*, soprattutto per i gruppi di piccole dimensioni. Anche il valore medio tra gruppi normali e *small* migliora leggermente. Questa differenza, seppur minima, si deve al minor numero di persone che compongono gli *small groups*, il cui peso nel calcolo della media è ridotto, ma comunque positivo, incentivando i gruppi più piccoli, come previsto dall'obiettivo.

Alla luce di questi risultati e considerando gli altri esperimenti, la configurazione che offre le migliori prestazioni è quella con *smallGroupSize* = 1 e una percentuale di posti riservati pari al **1.7%**. Il valore di *poissonLambda* ottimale, compreso tra 2.5 e 3, rappresenta il valore di default per le simulazioni ed è particolarmente adatto a modellare la realtà.

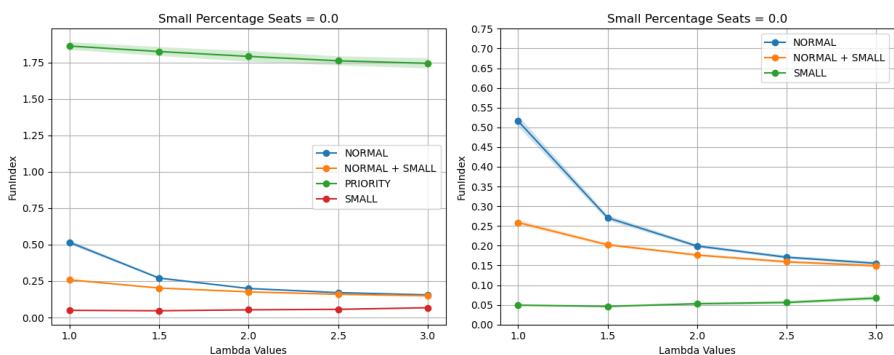


(a) 0% di posti riservati

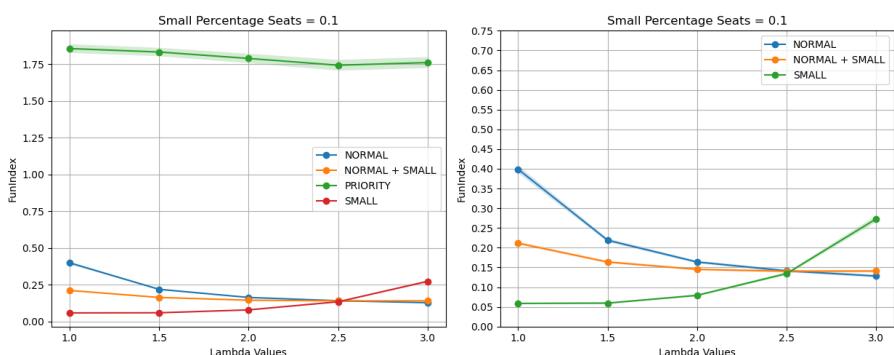


(b) 1.7% di posti riservati

Figura 26: FunIndex per smallGroupSize = 1



(a) 0% di posti riservati



(b) 10% di posti riservati

Figura 27: FunIndex per smallGroupSize = 2

## 16 Conclusioni

L'analisi condotta sul sistema di code ha mostrato miglioramenti nei tempi di coda, nell'utilizzo del sistema, e nel *FunIndex*. La configurazione con `smallGroupSize` pari a 1 e una riserva del 1.7% dei posti si è dimostrata la più efficiente ed è stata utilizzata per molti degli esperimenti fatti, permettendo di ridurre significativamente i tempi di attesa degli small group. Questo ha permesso di mantenere i tempi di coda giornalieri al di sotto dei livelli osservati nei gruppi normali, con eccezioni marginali e trascurabili.

Il valore di  $\rho$ , indicativo del grado di utilizzo del sistema, ha mostrato un miglioramento, confermando l'efficacia della nuova politica di scheduling introdotta. Questa ha permesso di coprire più efficacemente i serventi vacanti, portando l'utilizzazione da 91.35% al 92.44% nel modello migliorativo. L'attrazione che, invece, ha avuto il miglior incremento, è stata Incredible Hulk Coaster con un miglioramento dell'1.6%. Anche se questi incrementi possono sembrare molto bassi, durante l'intero arco della giornata possono avere una influenza maggiore.

Infine, il *FunIndex*, utilizzato per misurare l'efficacia complessiva delle modifiche introdotte, ha registrato un miglioramento medio del 5%, mentre per i gruppi di piccole dimensioni arriva al **410.2%**. In questo modo, è auspicabile aspettarsi un incremento della presenza di questi gruppi, contribuendo a migliorare la popolarità e l'affluenza complessiva del parco.

## 17 Dati

Di seguito sono riportati alcuni dati relativi al *FunIndex*, utilizzati per la generazione dei grafici, a conferma del miglioramento ottenuto.

PrioSeatsPercentage	Priority	FunIndex	ConfInterval
0.0	SMALL	0.1915230789368322	0.008524558764734461
0.0	NORMAL + SMALL	0.17991853937721197	0.0038696663103294808
0.0	PRIORITY	0.10171210820995408	0.006128763580729082
0.0	NORMAL	0.1797714145763046	0.003870119087229674
0.1	SMALL	0.1649000320940159	0.008236440214890593
0.1	NORMAL + SMALL	0.15670102658048868	0.003623345332866723
0.1	PRIORITY	0.2519552006290479	0.01178449647112124
0.1	NORMAL	0.15659732185984196	0.003606881334884428
0.2	SMALL	0.1480887413025385	0.007901499076289367
0.2	NORMAL + SMALL	0.1413736741213156	0.0029438664874383426
0.2	PRIORITY	0.7462546285314801	0.036654785472605304
0.2	NORMAL	0.14129369462511052	0.002947029038068052
0.30000000000000004	SMALL	0.1395295996649249	0.0068180912250018445
0.30000000000000004	NORMAL + SMALL	0.13432829713212702	0.0031394755378244426
0.30000000000000004	PRIORITY	1.3414113899747422	0.049144749502446
0.30000000000000004	NORMAL	0.1342603921439931	0.003158924230124435
0.4	SMALL	0.14026844612832612	0.00871939860459649
0.4	NORMAL + SMALL	0.13391630058434006	0.003112244522813704
0.4	PRIORITY	1.7760566576915509	0.0337773497369321
0.4	NORMAL	0.1338382067309424	0.00312670719105027
0.5	SMALL	0.13862476825039463	0.007165755801192327
0.5	NORMAL + SMALL	0.13178079529795414	0.003297956400587778
0.5	PRIORITY	1.9641402785246298	0.01873482992465265
0.5	NORMAL	0.13169736969278226	0.0033078513551611944
0.6	SMALL	0.13850968585270268	0.007957476305609704
0.6	NORMAL + SMALL	0.1319596911424615	0.0027816909268961938
0.6	PRIORITY	2.0562749628564303	0.014795827617396748
0.6	NORMAL	0.13187218382942503	0.0027749675484392542
0.7	SMALL	0.13972295159267434	0.008277710686278349
0.7	NORMAL + SMALL	0.13293446609709345	0.003418787985954344
0.7	PRIORITY	2.084964632791977	0.013631297070104449
0.7	NORMAL	0.1328469848781336	0.0034150959666574307
0.7999999999999999	SMALL	0.13869522779016677	0.00804846195336231
0.7999999999999999	NORMAL + SMALL	0.13158892501200734	0.0029135530380426347
0.7999999999999999	PRIORITY	2.1038546239193496	0.014127575752105227
0.7999999999999999	NORMAL	0.1314994085705867	0.0029196110628122145
0.8999999999999999	SMALL	0.14097218463608857	0.008329980349461986
0.8999999999999999	NORMAL + SMALL	0.13290161663008485	0.003249405199483756
0.8999999999999999	PRIORITY	2.1224239872179136	0.013726026089981117
0.8999999999999999	NORMAL	0.13279738608467884	0.0032622433166965645
0.9999999999999999	SMALL	0.14279014489300326	0.009032975060191484
0.9999999999999999	NORMAL + SMALL	0.132425692355305	0.0030994118194189245
0.9999999999999999	PRIORITY	2.12425548647368	0.012520553258516923
0.9999999999999999	NORMAL	0.13229932490992402	0.003099774402118355

Tabella 14: FunIndex modello base

SmallSeatsPercentage	PoissonParam	Priority	FunIndex	ConfInterval
0.017	1.0	SMALL	0.045339112916257265	0.002359366470484324
0.017	1.0	NORMAL + SMALL	0.15946636055005786	0.0026880641230211644
0.017	1.0	PRIORITY	1.8456818581001475	0.02792011343259577
0.017	1.0	NORMAL	0.1795705162386575	0.003193901310369352
0.0	1.0	SMALL	0.053764795929468374	0.0031784323480700647
0.0	1.0	NORMAL + SMALL	0.16768684378929277	0.002955492557958059
0.0	1.0	PRIORITY	1.8504530125962124	0.027835054543785282
0.0	1.0	NORMAL	0.18212310696101439	0.0033880765111760427
0.017	1.5	SMALL	0.05749080422408985	0.0026790680872845426
0.017	1.5	NORMAL + SMALL	0.1433073049728429	0.0028735079549426557
0.017	1.5	PRIORITY	1.8098185351959402	0.0301151502327562
0.017	1.5	NORMAL	0.15111378986213886	0.0031167614485201754
0.0	1.5	SMALL	0.06036758989252293	0.0035443628747460158
0.0	1.5	NORMAL + SMALL	0.14740774470069148	0.003067119136657887
0.0	1.5	PRIORITY	1.8315941250594525	0.029143717987963443
0.0	1.5	NORMAL	0.15352984610909945	0.0032449147425232306
0.017	2.0	SMALL	0.10637245301653069	0.004641266281500325
0.017	2.0	NORMAL + SMALL	0.14018993018356307	0.002738333013427043
0.017	2.0	PRIORITY	1.801582120080135	0.026777328138487634
0.017	2.0	NORMAL	0.1417694618432734	0.002887055028670049
0.0	2.0	SMALL	0.08017230761635212	0.005654929749190844
0.0	2.0	NORMAL + SMALL	0.14001754013182352	0.0029173892893346333
0.0	2.0	PRIORITY	1.797159777078408	0.03053120900086571
0.0	2.0	NORMAL	0.14249085324207955	0.0030457003561776127
0.017	2.5	SMALL	0.23323387107193838	0.011315615279981893
0.017	2.5	NORMAL + SMALL	0.13871668497550943	0.002889257683207533
0.017	2.5	PRIORITY	1.7649916590389136	0.030103297167990477
0.017	2.5	NORMAL	0.1364557540920854	0.0029438775703449797
0.0	2.5	SMALL	0.12506180810306336	0.008053497418175418
0.0	2.5	NORMAL + SMALL	0.13914412456313302	0.0031552914312453838
0.0	2.5	PRIORITY	1.765829604283964	0.035598198326421186
0.0	2.5	NORMAL	0.13949129927721798	0.00322780076943535
0.017	3.0	SMALL	0.7184187963896406	0.042954620809213896
0.017	3.0	NORMAL + SMALL	0.14086972593186656	0.0035329886272095355
0.017	3.0	PRIORITY	1.7607956206210151	0.034355337682260144
0.017	3.0	NORMAL	0.1336126314323319	0.003374611612192891
0.0	3.0	SMALL	0.2470033257107976	0.015566165345483609
0.0	3.0	NORMAL + SMALL	0.13649383791265285	0.0031781259146531826
0.0	3.0	PRIORITY	1.7421017314907863	0.035579623133484624
0.0	3.0	NORMAL	0.13512629125536804	0.0031915395356421346

Tabella 15: FunIndex modello migliorativo con smallGroupSize = 1

## 18 Links

- source code: [https://github.com/SimoneNicosanti/PMCSN\\_Project](https://github.com/SimoneNicosanti/PMCSN_Project)
- <https://queue-times.com/parks/64/attendances>
- <https://queue-times.com/parks/64/stats/2022>