Contents

Progetto PMCSN
Gestione dei mezzi in un Polo impiantistico AMA

Simone Festa 0320408

Michele Tosi 0327862

Università degli studi di Roma Tor Vergata

Abstract

Il presente studio vuole analizzare la gestione operativa di un Polo impiantistico AMA, composto da una sezione dedita allo smaltimento dei rifiuti e una sezione per la riparazione dei mezzi circolanti. Lo studio è assecondato da **dati reali** del sistema, usati nella fase simulativa per analizzarne l'attuale comportamento. Si vuole inoltre fornire una strategia di miglioramenti che porteranno a una gestione più efficiente dei mezzi, permettendo quindi una maggiore disponibilità di questi per la raccolta dei rifiuti (rispettando criteri reali), oltre che a un maggior decoro per la città di Roma.

1 Introduzione

Il sistema analizzato è un Polo impiantistico AMA. L'azienda gestisce una flotta di automezzi suddivisi in base alla loro capacità. Dopo aver completato un turno di lavoro, i mezzi impiegati nella raccolta rientrano al centro, dove i rifiuti raccolti vengono differenziali e smaltiti.

Ogni mezzo, al termine del proprio servizio, entra nel centro. All'ingresso, se il mezzo non presenta problemi, viene indirizzato direttamente allo smaltimento. Nel caso in cui riporti qualche problema tecnico, il quale può aver impedito il completamento della raccolta, il mezzo passa prima per l'accettazione, dove viene diagnosticato il problema, per poi essere reindirizzato, in base alla diagnosi, presso l'autofficina competente.

Dopo la fase di smaltimento, se necessario, i mezzi vengono sottoposti a sanitizzazione o rifornimento, per poi uscire dal sistema in una fase denominata check-out; altrimenti, escono direttamente.

2 Problematiche del sistema

L'azienda è tenuta ad assicurare un numero minimo di mezzi per ogni tipologia al di fuori del sistema nell'arco della giornata. Tuttavia, spesso ciò non si verifica poiché i mezzi che entrano nel centro per essere riparati affrontano notevoli tempi di attesa, anche per riparazioni minori. Questo accade in quanto devono attendere che vengano serviti tutti i mezzi arrivati precedentemente.

La criticità risiede nella mancanza di una gestione ottimale delle code dei mezzi. Attualmente, la gestione si basa solo sull'ordine di arrivo, senza considerare la necessità di avere un certo tipo di mezzo fuori dal sistema il prima possibile per rispettare le richieste sul numero di mezzi.

L'ignorare questi aspetti può causare malfunzionamenti nel servizio di raccolta, con ripercussioni sia sulla comunità che sul decoro pubblico, comportando penalità per l'azienda.

3 Scopi e Obiettivi

Lo studio condotto sul sistema si propone di raggiungere come obiettivo quello di minimiz-

zare il numero di mezzi all'interno del sistema per ogni categoria in modo da rispettare per complementarità i QoS richiesti (meno mezzi si hanno nel sistema, più mezzi sono effettivamente in uso per raccogliere rifiuti).

- Il numero di mezzi di tipo CSL2 (veicoli *"piccoli"*) al di fuori del sistema deve superare il 27%.
- Il numero di mezzi di tipo CSL3 (veicoli *"grandi"*) al di fuori del sistema deve superare il 64%.

Dato che il Polo impiantistico dispone di un numero limitato di mezzi (182 in totale, di cui 103 di piccole dimensioni e 79 di grandi dimensioni, per coprire un'ampia area di Roma) e considerando possibili tempi lunghi nelle riparazioni, l'obiettivo principale è **minimizzare le situazioni in cui non si riesce a garantire il rispetto di tali standard di servizio (quality of service)**. Il miglioramento dovrà essere implementato senza apportare modifiche all'organizzazione del polo, mantenendo invariato sia il numero di mezzi che le tempistiche delle riparazioni.

4 Modello concettuale

In questa fase della trattazione, le caratteristiche del sistema sono definite solo a livello *astratto*, e il nostro focus verte sull'interrogarci in merito agli stati ed eventi nel sistema (discriminando quelli rilevanti da quelli non rilevanti), e alla loro interrelazione.

Per iniziare, identifichiamo gli utenti del sistema in:

- **CSL2**: Mezzi leggeri o piccoli, con una capienza di 50 tonnellate.
- **CSL3**: Mezzi pesanti o grandi, con una capienza di 100 tonnellate.

Il sistema è suddivisibile nelle seguenti macroaree:

- **Accettazione**: Permette la diagnosi dei mezzi guasti rientranti nel sistema. In quest'area si identifica il tipo di guasto, per indirizzare il mezzo presso l'officina competente.
- **Officina**: Zona dedita alla riparazione dei mezzi per un particolare tipo di guasto. Questa area si compone di 5 officine specializzate:
  - *Gommista*: Questa officina si occupa della sostituzione delle ruote per i veicoli presenti nel sistema. L'operazione richiede l'uso di strumenti specifici, come i *cric* appositi per sollevare il mezzo. La maggior parte delle situazioni che richiedono la sostituzione delle ruote è causata dall'*usura*, spesso portando alla sostituzione di più ruote contemporaneamente.
  - *Carrozzeria*: Gestisce aspetti maggiormente estetici che, tuttavia, potrebbero influenzare la funzionalità del veicolo.
  - *Elettrauto*: L'officina di elettrauto si occupa della parte elettronica del veicolo, comprendente batteria, motorino di avviamento, alternatore, fusibili, cavi e connettori.
  - *Carpenteria*: Gestisce problemi legati alle parte anteriore e posteriore del veicolo, come *braccia per il sollevamento dei cassonetti* e *compattatori dei rifiuti*.
  - *Meccanica*: L'officina meccanica si occupa di componenti come motori e freni, oltre agli aspetti interni del veicolo e quelli legati alla parte dedicata allo scarico. Le riparazioni di questa natura sono essenziali per il corretto funzionamento del veicolo e richiedono un considerevole tempo di intervento.
- **Smaltimento**: Zona adibita allo scarico dei rifiuti, permette lo scarico di 5 mezzi contemporaneamente.
- **Checkout**: Un mezzo uscente dal sistema potrebbe richiedere servizi di sanitizzazione o rifornimento.

Un operatore AMA, al rientro dal servizio, può presentare o meno un guasto al mezzo, non attualmente identificato.

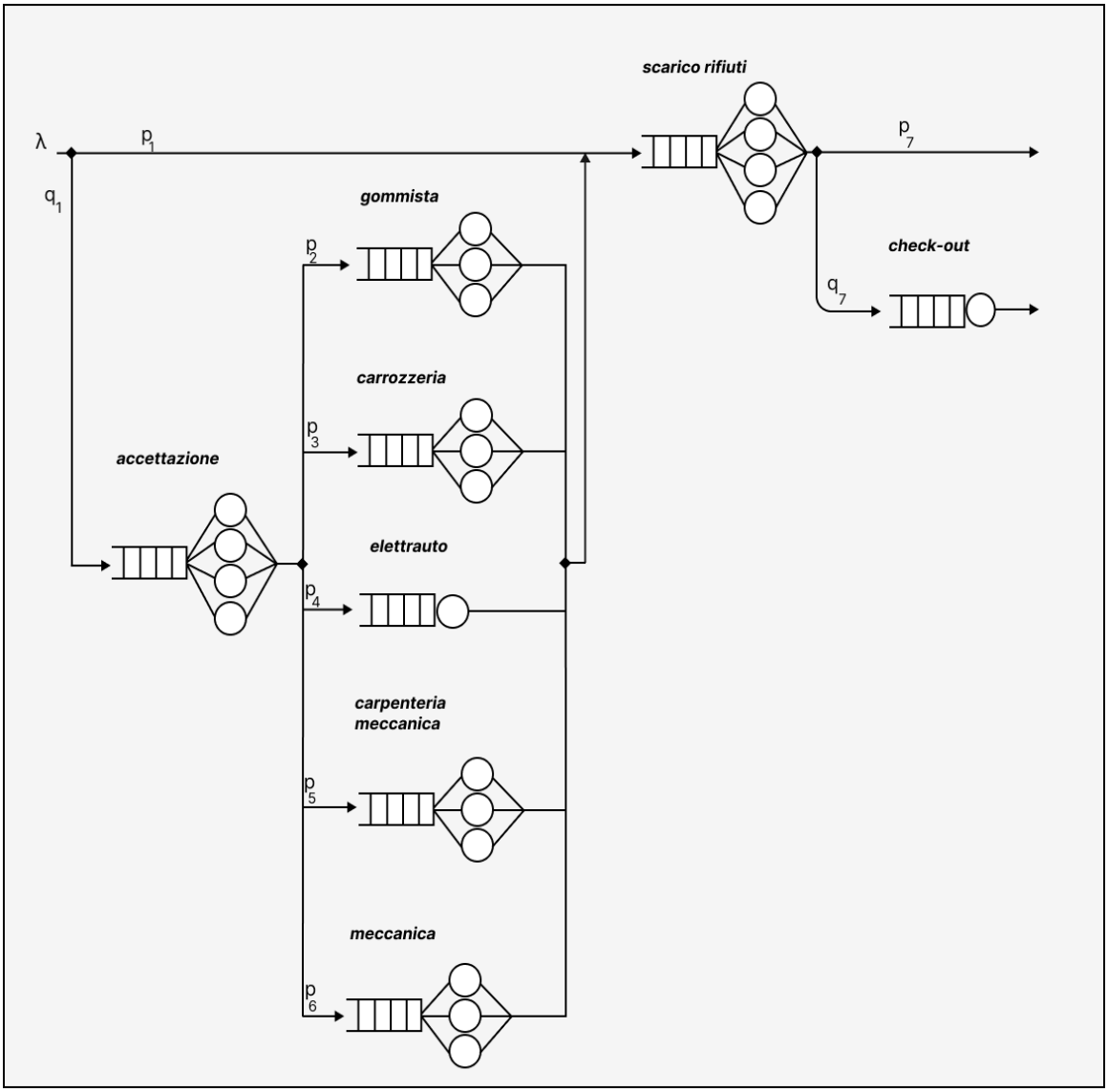
- Se il mezzo è integro, l'operatore si reca presso la zona **smaltimento**, nella quale attende il suo turno per smaltire i rifiuti.

- Se il mezzo presenta anomalie, viene reindirizzato presso **l'accettazione**, che identifica il tipo di problema. Tipicamente, il mezzo stesso presenta un sistema di diagnostica per facilitare l'operazione, e ciò si traduce in tempi rapidi per l'identificazione dell'officina di competenza. Solo dopo questa fase, un mezzo precedentemente guasto può eseguire lo scarico. E' normale prassi, infatti, che un mezzo guasto non venga svuotato prima della riparazione, ma solo successivamente, in quanto lo scarico dei rifiuti non è un'operazione meccanicamente banale, e la sua attuazione in condizioni non idonee potrebbe portare a guasti di maggior entità. Questa teoria è rafforzata anche dai report, che spesso, insieme alla riparazione, specificano la necessità di scaricare il mezzo:

Descrizione RdI
MEZZO DA SCARICARE

- Terminato lo scarico, il mezzo può uscire direttamente dal sistema, oppure passare per il **checkout**, per ricevere servizi di sanitizzazione o rifornimento.

Quanto descritto finora nel modello concettuale si può schematizzare nell'immagine che segue.



Eventi

In questa trattazione, identifichiamo con il termine *mezzo* (inteso come camion adibito al trasporto di rifiuti) un *job* che arriva nel sistema e richiede uno o più servizi. Inoltre, discriminiamo il termine *sistema* (inteso come insieme dei centri presenti nella figura di sopra) da i suoi *centri* (interni al sistema, adibiti ad uno specifico scopo).

Per il **sistema**:

- ❒ Un evento considerabile di riferimento è l'*arrivo* di un job.
- ❒ Un job può *uscire* dal sistema solo uscendo direttamente dal centro *scarico* o indirettamente dal *check-out*.

Per il **centro accettazione**:

- ❒ Un evento considerabile è l'*arrivo* di un job, che coincide con l'arrivo di un nuovo job nel sistema.
- ❒ Un evento di *uscita* per il centro (ma non per il sistema) è dato dal processamento del servizio del job, ovvero l'identificazione di un guasto.

Per i **centri "Gommista", "Carrozzeria", "Elettrauto", "Carpenteria", "Meccanica"** consideriamo:

- ❒ Un evento di *arrivo* in un centro, il quale coincide con l'evento di uscita di questo stesso job dal centro *accettazione*. Tale evento non coincide con l'arrivo di un nuovo job nel **sistema.**
- ❒ Un evento di *uscita* per il centro (ma non per il sistema) è dato dal processamento del servizio del job, ovvero la riparazione del difetto riscontrato in fase di accettazione.

Per il centro **Scarico rifiuti** consideriamo:

- ❒ Un evento di *arrivo* di un job, che può coincidere con l'evento di arrivo di un job nel sistema (nel caso il mezzo non sia guasto), o meno (nel caso in cui il job provenga da uno dei centri di riparazione).
- ❒ Un evento di *uscita* per il centro è dato dal processamento di tale job, ovvero lo scarico dei rifiuti raccolti; che può coincidere con l'uscita di un job dal sistema (nel caso in cui tale job non passi per il centro *check-out*) o meno.

Per il centro **Checkout** consideriamo:

- ❒ Un evento di *arrivo* di un job, coincidente con l'uscita di tale job dal centro *scarico rifiuti*, e non coincide mai con l'evento di arrivo di un nuovo job nel sistema.
- ❒ Un evento di *uscita* per il centro è dato dal processamento di tale job, ovvero la sanitizzazione e/o rifornimento del mezzo, che coincide sempre con un evento di uscita di un job per il sistema.

Descrizione degli eventi

Un mezzo entrante nel sistema e **non guasto**, si reca presso l'area conforme allo scarico dei rifiuti. Questa prevede 5 zone per lo scarico. Un mezzo occupa una sola "zona scarico". Se tutte le zone sono occupate, il mezzo viene messo in coda, secondo una politica *FIFO astratta*, ma non può abbandonare il sistema, in quanto non agibile per la raccolta di nuovi rifiuti.

Un mezzo entrante nel sistema e **guasto**, viene portato nell'area **accettazione**. Qui si esegue un lavoro di tipo *dispatcher*, in quanto il mezzo viene indirizzato verso il centro di riparazione più congeniale. Sono presenti 4 serventi dediti a tale fase. Se tutti i serventi sono occupati, il mezzo rimane in attesa secondo una politica *FIFO astratta*.

Successivamente a questa fase, il mezzo guasto viene preposto per la riparazione, classificata per la tipologia di guasto, a cui corrisponde una determinata officina. A seconda dell'officina, possono esserci più o meno serventi, rappresentati nella figura precedentemente esposta. Tutti i mezzi in attesa di riparazione non possono lasciare la coda nè il sistema. Tutti i mezzi riparati sono reintroducibili nella coda principale per lo scarico dei rifiuti, di cui abbiamo già esaminato le caratteristiche.

Infine, tutti i mezzi che hanno espletato l'operazione di scarico dei rifiuti possono uscire dal sistema, o rimanervi in attesa della sanitizzazione e rifornimento.

Nello stato attuale dell'analisi, tutte le code

sono definite con priorità FIFO astratte. L'unica differenziazione può avvenire nel numero di serventi, che confluiscono nell'uso di single-server o multi-server.

5 Obiettivo e profitto del sistema

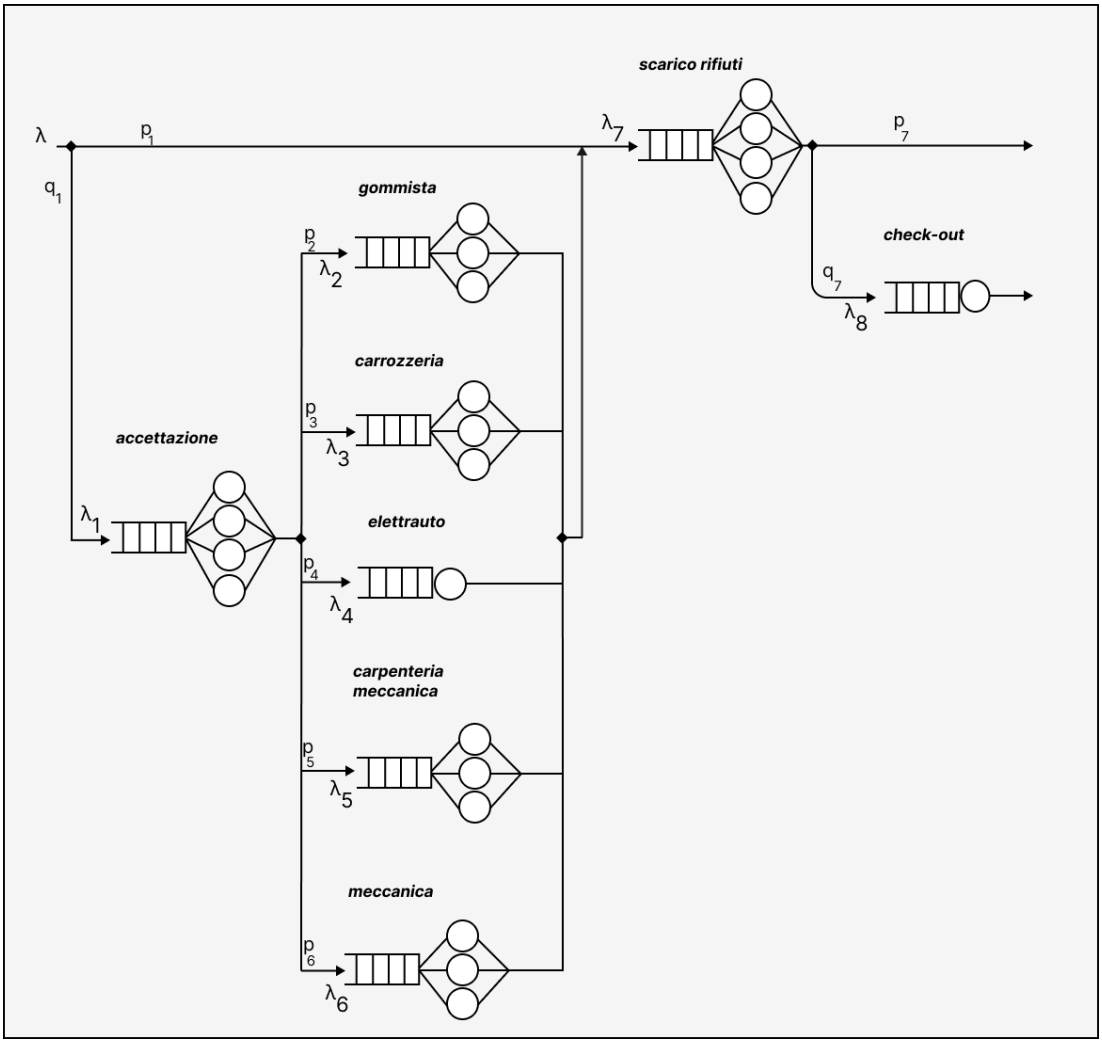
In quanto azienda pubblica, è inappropriato utilizzare parametri finanziari come indicatori di performance. Tuttavia, è essenziale considerare l'impatto culturale del decoro urbano sul turismo e sulla percezione del territorio. L'analisi si concentra sulla qualità del servizio offerto, correlata al numero di mezzi fuori servizio. Un veicolo fermo non partecipa alle operazioni di pulizia delle strade, causando potenziale insoddisfazione tra i cittadini. Questi aspetti sono sostenuti da limiti minimi relativi al numero di mezzi in circolazione. L'obiettivo primario è ridurre al minimo le situazioni in cui non si riesce a rispettare il requisito di disponibilità minima, che coincide con l'incapacità di garantire un numero prefissato di mezzi (di entrambe le tipologie) al di fuori del sistema.

6 Modello delle specifiche

A tale livello, lo stato del sistema esiste come collezione di variabili di stato (di tipo matematico) corroborate da equazioni aventi lo scopo di descrivere la loro interrelazione. Per rendere il più possibile realistica la modellazione del sistema, abbiamo chiesto ed ottenuto dei dati direttamente dall'AMA, come il numero di mezzi totali e quelli fermi in attesa di riparazione. Sono stati ottenuti anche report su alcune tempistiche di riparazione, oltre che dati sui tempi richiesti per lo smaltimento di un mezzo.

Per facilitare la trattazione, presentiamo nuovamente la rappresentazione del sistema, supportata questa volta dalle probabilità di entrare in un centro, fornendo quindi le **equazioni di traffico** la **matrice di routing**.

Equazioni di traffico



$$\lambda_1 = (1 - p_1) \cdot \lambda$$
$$\lambda_2 = p_2 \cdot \lambda_1$$
$$\lambda_3 = p_3 \cdot \lambda_1$$
$$\lambda_4 = p_4 \cdot \lambda_1$$
$$\lambda_5 = p_5 \cdot \lambda_1$$
$$\lambda_6 = p_6 \cdot \lambda_1$$
$$\lambda_7 = \lambda$$
$$\lambda_8 = \lambda_7 \cdot (1 - p_7)$$

Di seguito, viene invece illustrata la matrice di routing.

Matrice di routing

	Esterno	Accettazione	Gommista	Carrozzeria	Elettrauti	Carpenteria	Meccanica	Scarico	Checkout
Esterno	0	$1 - p_1$	0	0	0	0	0	p_1	0
Accettazione	0	0	p_2	p_3	p_4	p_5	p_6	0	0
Gommista	0	0	0	0	0	0	0	1	0
Carrozzeria	0	0	0	0	0	0	0	1	0
Elettrauti	0	0	0	0	0	0	0	1	0
Carpenteria	0	0	0	0	0	0	0	1	0
Meccanica	0	0	0	0	0	0	0	1	0
Scarico	p_7	0	0	0	0	0	0	0	$1 - p_7$
Checkout	1	0	0	0	0	0	0	0	0

Modellazione dei centri

Teorema: se il numero di arrivi durante un qualsiasi intervallo segue una distribuzione di Poisson, allora i tempi di interarrivo sono distribuiti Esponenzialmente. Altri aspetti di cui terremo conto, in quanto attinenti al sistema in esame:

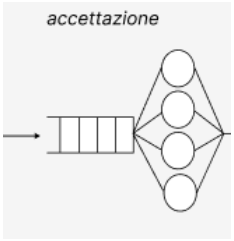
- ❒ Indipendenza per gli arrivi: l'arrivo di un mezzo nello stabilimento non influenza altri arrivi.
- ❒ Memoryless: Un arrivo futuro non è influenzato da arrivi passati.
- ❒ Arrivi Poissoniani: Trattasi di eventi casuali ed indipendenti, con un tasso di arrivo medio.

Disclaimer: La distribuzione per i tempi di servizio utilizzata è una *Gaussiana troncata*. Questa scelta non ci permetterebbe di parlare in modo appropriato di code $M\backslash M\backslash x$, in quanto queste richiederebbero servizi esponenziali. Come vedremo in seguito, sono state condotte delle simulazioni anche con i tempi di servizio esponenziali, e proprio per questo, nella seguente sezione, abuseremo della notazione $M\backslash M\backslash x$.

Centro Accettazione

Tale centro viene modellato con un **Multi-server con coda M/M/4**.

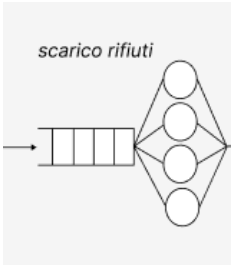
- ❒ La distribuzione dei tempi di interrarivo è **Esponenziale** di parametro $\frac{1}{\lambda_1}$.
- ❒ La distribuzione dei servizi è rappresentata da una **Gaussiana troncata**, avente come tempo di servizio medio, per il singolo servente, $E[S_i] = 600$ secondi (pari a 10 minuti). Questo tempo di servizio può oscillare tra un minimo di 5 minuti ed un massimo di 15 minuti.
- ❒ Non vi è alcuna possibilità di abbandono per un job, in quanto, come già disquisito in precedenza, un mezzo guasto può unicamente aspettare nel sistema.



Scarico rifiuti

Tale centro viene modellato con un **Multi-server con coda M/M/4**.

- ❒ La distribuzione dei tempi di interrarivo è **Esponenziale** di parametro $\frac{1}{\lambda}$.
- ❒ La distribuzione dei servizi è rappresentata da una **Gaussiana troncata**, avente come tempo di servizio medio, per il singolo servente, $E[S_i] = 600$ secondi (pari a 10 minuti). Questo tempo di servizio può oscillare tra un minimo di 8 minuti ed un massimo di 15 minuti.
- ❒ Non vi è alcuna possibilità di abbandono per un job, in quanto un mezzo in attesa di essere servito, non è operativo fintantochè non viene scarico dei suoi rifiuti.

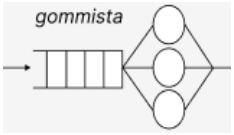


Autofficina

Gommista

Tale centro viene modellato con un **Multi-server con coda M/M/3**.

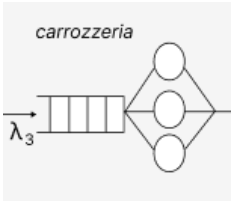
- ❒ La distribuzione dei tempi di interrarivo è **Esponenziale** di parametro $\frac{1}{\lambda_2}$.
- ❒ La distribuzione dei servizi è rappresentata da una **Gaussiana troncata**, avente come tempo di servizio medio, per il singolo servente, $E[S_i] = 3600$ secondi (pari a un'ora). Questo tempo di servizio può oscillare tra un minimo di 1800 secondi (pari a mezz'ora) ed un massimo di 5400 secondi (pari a un'ora e mezza).
- ❒ Nessun job può lasciare la coda di riparazione.



Carrozzeria

Tale centro viene modellato con un **Multi-server con coda M/M/3**.

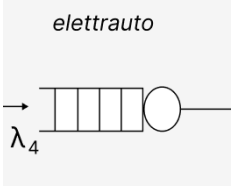
- La distribuzione dei tempi di interrarivo è **Esponenziale** di parametro $\frac{1}{\lambda_3}$.
- La distribuzione dei servizi è rappresentata da una **Gaussiana troncata**, avente come tempo di servizio medio, per il singolo servente, $E[S_i] = 5400$ secondi (pari a due ora). Questo tempo di servizio può oscillare tra un minimo di 3600 secondi (pari a un'ora) ed un massimo di 7200 secondi (pari a due ore).
- Nessun job può lasciare la coda di riparazione.



Elettrauto

Tale centro viene modellato con un **Multi-server con coda M/M/1**.

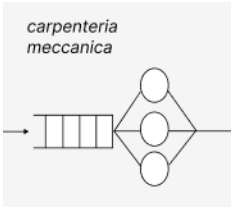
- La distribuzione dei tempi di interrarivo è **Esponenziale** di parametro $\frac{1}{\lambda_4}$.
- La distribuzione dei servizi è rappresentata da una **Gaussiana troncata**, avente come tempo di servizio medio, per il singolo servente, $E[S_i] = 5400$ secondi (pari a due ora). Questo tempo di servizio può oscillare tra un minimo di 3600 secondi (pari a un'ora) ed un massimo di 7200 secondi (pari a due ore).
- Nessun job può lasciare la coda di riparazione.



Carpenteria Meccanica

Tale centro viene modellato con un **Multi-server con coda M/M/3**.

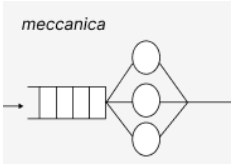
- La distribuzione dei tempi di interrarivo è **Esponenziale** di parametro $\frac{1}{\lambda_5}$.
- La distribuzione dei servizi è rappresentata da una **Gaussiana troncata**, avente come tempo di servizio medio, per il singolo servente, $E[S_i] = 5400$ secondi (pari a due ora). Questo tempo di servizio può oscillare tra un minimo di 3600 secondi (pari a un'ora) ed un massimo di 7200 secondi (pari a due ore).
- Nessun job può lasciare la coda di riparazione.



Meccanica

Tale centro viene modellato con un **Multi-server con coda M/M/3**.

- La distribuzione dei tempi di interrarivo è **Esponenziale** di parametro $\frac{1}{\lambda_6}$.
- La distribuzione dei servizi è rappresentata da una **Gaussiana troncata**, avente come tempo di servizio medio, per il singolo servente, $E[S_i] = 5400$ secondi (pari a due ora). Questo tempo di servizio può oscillare tra un minimo di 3600 secondi (pari a un'ora) ed un massimo di 7200 secondi (pari a due ore).
- Nessun job può lasciare la coda di riparazione.



7 Modello computazionale

Architettura

Per la realizzazione del progetto, si è optato per l'utilizzo di Java, il quale presenta, tra i suoi punti di forza, aspetti quali *portabilità* ed un ampio *supporto*. Oltretutto, l'uso di Java ha permesso l'esecuzione in remoto (mediante Maven) di simulazioni esose in termini di risorse. Il paradigma di simulazione è di tipo *next-event*. La simulazione si avvia al tempo *START*, e termina al tempo *STOP*, corrispondente ad un giorno lavorativo.

Struttura del progetto

Il progetto è composto da tre blocchi fondamentali:

- **Controllers**: Contiene la logica dei vari nodi formanti il sistema in esame, oltre che un EventHandler per la gestione centralizzata degli eventi, di cui disquisiremo più in avanti.
- **Models**: Contiene codice per l'implementazione di una Multi-Server-Queue, e costanti di uso comune nel sistema. Presenta inoltre la definizione degli eventi del sistema, tempi di servizio, **tasso λ degli arrivi** e probabilità di essere indirizzati ad un centro rispetto che un altro.
- **Utils**: Contiene codice prodotto da **Steve Park** e **Dave Geyer**, il quale implementa facility per la generazione di numeri multi-stream, e produzione di statistiche di output

Descrizione dell'Event Handler

La logica adottata segue quella proposta dal libro *Discrete event simulation*, nella quale si mantiene una entry per ciascun server, più una per gli ingressi. Inoltre, viene fatta una distinzione tra gli eventi provenienti dall'esterno del sistema (i quali possono confluire unicamente nello *scarico* o in *accettazione*) da quelli interni, che coinvolgono maggiormente le officine. La classe fornisce metodi di tipo *get* e *set* per gli eventi distribuiti nel sistema. Implementa, nel modello migliorativo, la nuova logica di gestione degli eventi.

Descrizione dei controller di Sistema

Questa classe, valida sia per i singoli centri sia per il sistema completo, è caratterizzata da:

```
long number =0;                 /*number in the node*/
long numberV1 =0;               /*number in the node v1*/
long numberV2 =0;               /*number in the node v2*/
int e;                          /*next event index*/
int s;                          /*server index*/
private long jobServed=0;       /*contatore jobs processati*/
private double area=0.0;        /*time integrated number in the node (mezzi totali)*/
private double area1=0.0;       /*time integrated number in the node (mezzi piccoli)*/
private double area2=0.0;       /*time integrated number in the node (mezzi grandi)*/

private final EventHandler eventHandler;  /*istanza dell'EventHandler per ottenere
                                            le info sugli eventi*/
```

Nel metodo **baseSimulation**, ovvero la simulazione nel transiente, si preleva la lista degli eventi per il centro in esame; si verifica se la simulazione possa continuare, possibile se non sono verificate insieme le condizioni di "chiusura delle porte per gli arrivi" ed "eventi presenti nel sistema".

```
List<EventListEntry> eventList = this.eventHandler.getEventsAccettazione();
```

```
if(eventList.get(0).getX()==0 && this.number==0){
    eventHandler.getEventsSistema().get(1).setX(0);
    return;
}
```

Se tali condizioni non sono rispettate, si può procedere all'ottenimento del next-event, impostando il tempo di questo evento prima come prossimo tempo per il calcolo dell'area e poi come tempo corrente

```
e=EventListEntry.getNextEvent(eventList, SERVERS_ACCETTAZIONE);
this.time.setNext(eventList.get(e).getT());
this.area=this.area+(this.time.getNext()-this.time.getCurrent())*this.number;
this.time.setCurrent(this.time.getNext());
```

Si procede con l'analisi dell'evento:

- Se l'evento è un *arrivo*, di genera il tempo del *prossimo arrivo* come:

  ```
  eventList.get(0).setT(this.time.getCurrent()+this.rnd.getJobArrival(1))
  ```

  Si genera un nuovo evento di tipo arrivo, avente come tempo la somma tra tempo corrente e tempo di inter-arrivo di parametro $\frac{1}{\lambda}$

- Si verifica se tale tempo, ottenuto per il nuovo next-event, eccede il tempo di orizzonte finito pari a **STOP**, il quale porterebbe alla chiusura delle porte per gli arrivi. Se ciò è vero, questo job non entrerà nel sistema. Successivamente, si incrementa il numero di job presenti nel centro.

  ```
  if(eventList.get(0).getT()> STOP_FINITE){
              eventList.get(0).setX(0); //chiusura delle porte
              this.eventHandler.setEventsAccettazione(eventList);
          }
  ```

- `if(this.number<=SERVERS_ACCETTAZIONE)`

  Si verifica la presenza di server liberi attualmente, in caso positivo si ottiene un tempo di servizio, e si produce il tempo di completamento del servizio, con il server in questione occupato fino alla terminazione di questo tempo, considerando anche il tempo corrente di entrata in servizio.

  ```
  double service=this.rnd.getService(0); //ottengo tempo di servizio
  this.s=findOneServerIdle(eventList); //ottengo l'indice di un server libero
  sum.get(s).incrementService(service);
  sum.get(s).incrementServed();
  //imposta nella lista degli eventi che il server s è busy
  eventList.get(s).setT(this.time.getCurrent() +service);
  eventList.get(s).setX(1);
  eventList.get(s).setVehicleType(vType);
  ```

Se l'evento è di tipo partenza, ovvero l'indice è $\mathbf{e} \neq \mathbf{0}$, allora devo processare una partenza:

- Si decrementa il numero di job presenti nel centro.

  ```
  this.number--;
  this.jobServed++;
  this.s=e; //il server con index e è quello che si libera
  ```

- `if(this.number>=SERVERS_ACCETTAZIONE)`

Si verifica la presenza di altri job nel sistema, in caso affermativo si ottiene un nuovo tempo di servizio, e si imposta il tempo di occupazione del server al tempo attuale, sommato al tempo di servizio. Si aumenta il numero di job serviti.

```
double service=this.rnd.getService(0);
sum.get(s).incrementService(service);
sum.get(s).incrementServed();
eventList.get(s).setT(this.time.getCurrent()+service);
eventList.get(s).setVehicleType(queueAccettazione.get(0).getVehicleType());
queueAccettazione.remove(0);
```

Come accennato in precedenza, ogni centro del sistema presenta alcune caratteristiche che lo differenzia dagli altri:

Aspetti caratteristici del sistema

Gestione degli arrivi esterni

Il centro accettazione riceve arrivi dall'esterno, e ciò che produce in uscita fungerà da ingresso per i nodi dell'officina. Ciò avviene reindirizzando un mezzo guasto presso una certa officina, mediante probabilità p_i. Viene realizzato mediante:

```
eventHandler.getInternalEventsOfficina(off).add(new EventListEntry(event.getT(),
                                                event.getX(),
                                                event.getVehicleType()));
```

Ottenuto l'indice dell'officina 'off' nel quale il mezzo andrà in riparazione, si aggiunge una nuova entry, contenente il tempo di arrivo nell'officina, lo stato dell'evento, il tipo di mezzo.

Un vincolo implementato in questo centro è correlato al numero massimo di veicoli consentiti nel sistema. Gli arrivi esterni possono essere associati a due tipologie di mezzi, identificate da un numero intero. Se il sistema raggiunge la sua capacità massima, nuovi arrivi non possono essere accettati, restituendo quindi una tipologia di veicolo *invalida*. Questa restrizione viene attuata mediante:

```
int vType=rnd.getExternalVehicleType(); // tipo di veicolo in entrata
if(vType==Integer.MAX_VALUE) { // veicolo non valido
                eventList.get(0).setX(0);
                eventHandler.setEventsAccettazione(eventList);
                return;
            }
```

Questo controllo è presente anche per gli arrivi diretti nel centro di *Scarico*.

Questo controllo può essere ripristinato solo quando avviene l'uscita di mezzi dal sistema, consentendo così l'ingresso di nuovi veicoli. Tale operazione è gestita nei centri *Scarico* (nel caso in cui il mezzo esca direttamente dal sistema) e *Checkout*, attraverso:

```
eventHandler.decrementVType(event.getVehicleType());
```

Politica di selezione del server libero

La politica prevede di scegliere, come server libero, l'indice del server libero da più tempo. Viene realizzata dal metodo, proposto dal libro di testo, di seguito enunciato:

```
findOneServerIdle(List <EventListEntry> eventListNode)
```

Pseudo Random Number Generator

Anche in questo caso, la simulazione prende come riferimento il libro di testo, in particolare sfruttando la classe *Rngs.java* Definito il parametro *SEED*, questo viene impiegato per la generazione di numeri pseudo-casuali attraverso l'algoritmo di Lehmer. I valori generati vengono utilizzati sia per determinare i tempi di arrivo che per quelli di servizio. In ciascuna simulazione, viene utilizzato un singolo generatore, seguendo un approccio *multi-stream*. Questo si traduce nell'uso di stream distinti per ciascun tipo di evento generato. Tale approccio assicura l'indipendenza tra i numeri generati e richiede una sola invocazione del metodo *rngs.plantSeed()*. Questa implementazione risulta fondamentale per la gestione delle *Replicazioni*, come sarà approfondito nelle sezioni successive.

Realizzazione di una gaussiana troncata

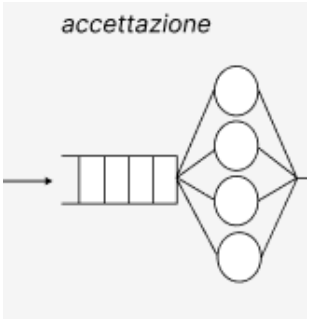
Per la gestione dei tempi di servizio, la scelta migliore è ricaduta su una *distribuzione Gaussiana Troncata*, questo perché dato un certo servente questo avrà un tempo medio di risposta con eventuali variazioni ma non è possibile che questo sia pari a 0. Come espresso dal libro di testo, per la produzione di un troncamento non basta definire dei limiti superiori ed inferiori, e *tagliare* la distribuzione agli estremi, in quanto questo produrrebbe degli *accumuli* agli estremi. Il seguente codice realizza una *Normale troncata*, seguendo le linee guida poste dal libro.

```
public double idfTruncatedNormal(double mean,           //media
                                 double std,            //dev. standard
                                 double lowerBound,
                                 double upperBound,
                                 double r)              //valore random tra 0 e 1
 {
      double a= cdfNormal(mean, std, lowerBound-1);
      double b= 1.0-cdfNormal(mean, std, upperBound);
      double u=idfUniform(a,1.0-b, r);
      return idfNormal(m, s, u);
}
```

8 Fase di verifica

Giunti a questa fase, ci chiediamo se il modello computazione implementato risulti corretto. Per questa fase, come era già stato anticipato, si farà uso di tempi di *servizio esponenziali*, in quanto, altrimenti, non si avrebbe un confronto valutabile dal punto di vista analitico sfruttando distribuzioni *normali troncate*. Per la realizzazione delle verifiche, il confronto avverrà considerando il caso stazionario. Ogni risultato è coadiuvato da un intervallo di confidenza del 95 %, di cui tratteremo meglio nelle sezioni successive. Il seed è pari a 123456789.

Accettazione



$\lambda_1 = 0.0055 \cdot 0.3$ job/sec

$E(S_i) = 600$ sec

$N = 4$

$E(S) = \frac{E(S_i)}{N} = 150$ sec

$\mu = \frac{1}{E(S)} = 0.00056$ job/sec

$\rho = \frac{\lambda}{\mu} = 0.2475$

$p_0 = \left[\left(\sum_{i=0}^{N-1} \frac{(N\cdot\rho)^i}{i!} + \frac{(N\cdot\rho)^N}{N!(1-\rho)}\right)\right]^{-1} = 0,371063621$

$P_q = \frac{(N\rho)^N \cdot p_0^{-1}}{N!(1-\rho)} = 0,019736558$

$E(T_q) = \frac{P_q \cdot E(S)}{1-\rho} = 3,934197563$ sec

$E(N_q) = E(T_q) \cdot \lambda_1 = 0,006491426$

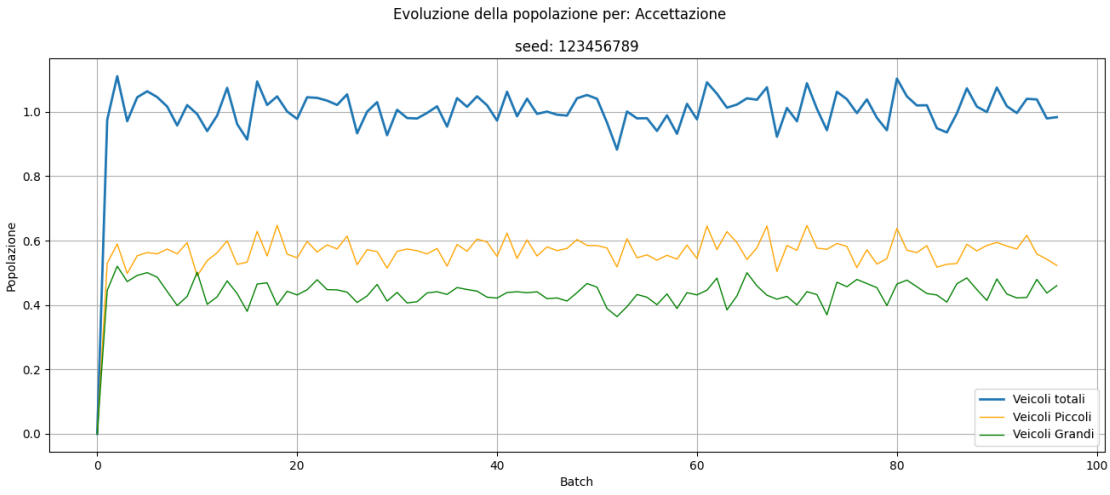
$E(T_s) = E(T_q) + E(S_i) = 603,9341976$ sec

$E(N_s) = E(T_s) \cdot \lambda_1 = 0,996491426$

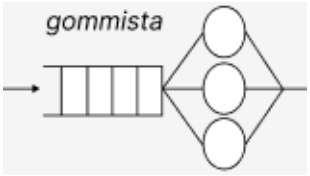
I risultati prodotti dalla simulazione sono:

```
Accettazione
Statistiche per E[Tq] Critical endpoints 3.9283527942177083 +/- 0.48724631608792296
statistiche per E[Nq] Critical endpoints 0.0065520388544007295 +/- 8.106930723449465E-4
statistiche per rho Critical endpoints 0.25047792978926114 +/- 0.005668989243929341
statistiche per E[Ts] Critical endpoints 605.6384378052535 +/- 13.177046438653104
statistiche per E[Ns] Critical endpoints 1.0084637580114462 +/- 0.022913798605430705
```

Visibili anche dal grafico prodotto, il quale si concentra sul mostrare la popolazione completa $E(N_s)$ nel centro in esame:



Gommista



$\lambda_2 = 0.00165 \cdot 0.3$ job/sec

$E(S_i) = 3600$ sec

$N = 3$

$E(S) = \frac{E(S_i)}{N} = 1200$ sec

$\mu = \frac{1}{E(S)} = 0.000883$ job/sec

$\rho = \frac{\lambda}{\mu} = 0.594$

$p_0 = \left[\left(\sum_{i=0}^{N-1} \frac{(N\cdot\rho)^i}{i!} + \frac{(N\cdot\rho)^N}{N!(1-\rho)}\right)\right]^{-1} = 0,1494155$

$P_q = \frac{(N\rho)^N \cdot p_0^{-1}}{N!(1-\rho)} = 0,3470896$

$E(T_q) = \frac{P_q \cdot E(S)}{1-\rho} = 1025,880617$ sec

$E(N_q) = E(T_q) \cdot \lambda_1 = 0,507810906$

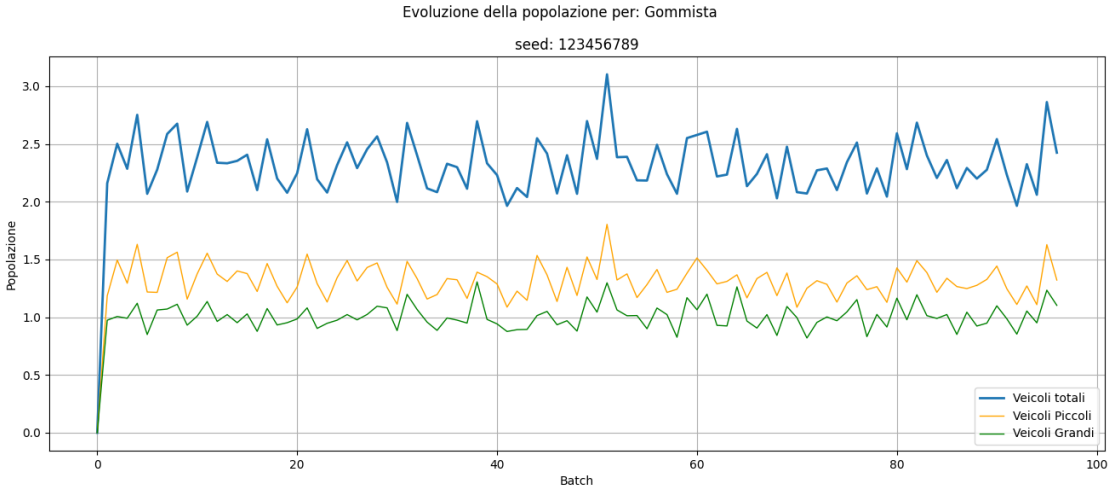
$E(T_s) = E(T_q) + E(S_i) = 4625,880617$ sec

$E(N_s) = E(T_s) \cdot \lambda_1 = 2,289810906$

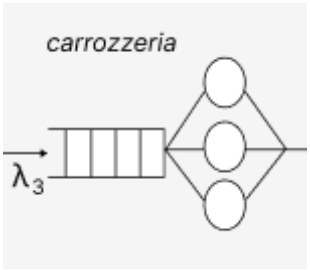
I risultati prodotti dalla simulazione sono:

```
Gommista
Statistiche per E[Tq] Critical endpoints 1048.4011383832624 +/- 65.23729644804527
statistiche per E[Nq] Critical endpoints 0.5250312795470018 +/- 0.03424069270030117
statistiche per rho Critical endpoints 0.6009428927401637 +/- 0.01352376715915484
statistiche per E[Ts] Critical endpoints 4661.325075514952 +/- 123.74677494405113
statistiche per E[Ns] Critical endpoints 2.3278599577674925 +/- 0.06613873007286733
```

Visibili anche dal grafico prodotto, il quale si concentra sul mostrare la popolazione completa $E(N_s)$ nel centro in esame:



Carrozzeria



$\lambda_3 = 0.00165 \cdot 0.1 \ \text{job/sec}$

$E(S_i) = 5400 \ \text{sec}$

$N = 3$

$E(S) = \frac{E(S_i)}{N} = 1800 \ \text{sec}$

$\mu = \frac{1}{E(S)} = 0.000556 \ \text{job/sec}$

$\rho = \frac{\lambda}{\mu} = 0.297$

$p_0 = \left[\left(\sum_{i=0}^{N-1} \frac{(N\cdot\rho)^i}{i!} + \frac{(N\cdot\rho)^N}{N!(1-\rho)}\right)\right]^{-1} = 0,40722615$

$P_q = \frac{(N\rho)^N \cdot p_0^{-1}}{N!(1-\rho)} = 0,068290799$

$E(T_q) = \frac{P_q \cdot E(S)}{1-\rho} = 174,8555313 \ \text{sec}$

$E(N_q) = E(T_q) \cdot \lambda_1 = 0,028851163$

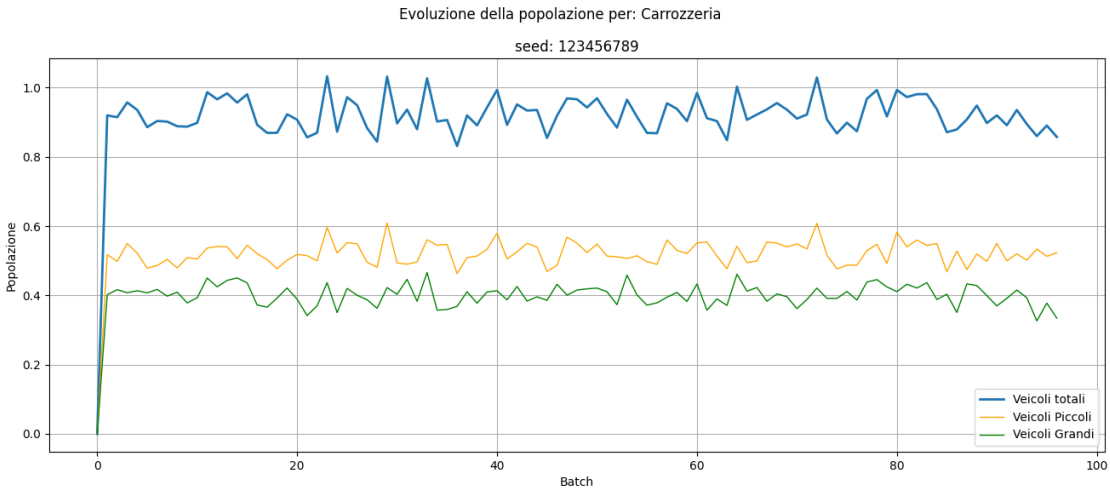
$E(T_s) = E(T_q) + E(S_i) = 5574,855531 \ \text{sec}$

$E(N_s) = E(T_s) \cdot \lambda_1 = 0,919851163$

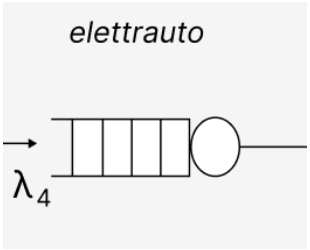
I risultati prodotti dalla simulazione sono:

```
Carrozzeria
Statistiche per E[Tq] Critical endpoints 183.60000343478274 +/- 13.00326527767447
statistiche per E[Nq] Critical endpoints 0.030340140559112813 +/- 0.002185138187298723
statistiche per rho Critical endpoints 0.29764684002567454 +/- 0.00675981972064844
statistiche per E[Ts] Critical endpoints 5598.748805381074 +/- 123.26838912880692
statistiche per E[Ns] Critical endpoints 0.9232806606361366 +/- 0.02132452463702056
```

Visibili anche dal grafico prodotto, il quale si concentra sul mostrare la popolazione completa $E(N_s)$ nel centro in esame:



Elettrauto



$\lambda_3 = 0.00165 \cdot 0.1 \ \text{job/sec}$

$E(S_i) = 5400 \ \text{sec}$

$N = 1$

$E(S) = \frac{E(S_i)}{N} = 5400 \ \text{sec}$

$\mu = \frac{1}{E(S)} = 0,000185185 \ \text{job/sec}$

$\rho = \frac{\lambda}{\mu} = 0,891$

$p_0 = \left[\left(\sum_{i=0}^{N-1} \frac{(N\cdot\rho)^i}{i!} + \frac{(N\cdot\rho)^N}{N!(1-\rho)}\right)\right]^{-1} = 0,099351119$

$P_q = \frac{(N\rho)^N \cdot p_0^{-1}}{N!(1-\rho)} = 0,812127035$

$E(T_q) = \frac{P_q \cdot E(S)}{1-\rho} = 40233,81639$ sec

$E(N_q) = E(T_q) \cdot \lambda_1 = 6,638579704$

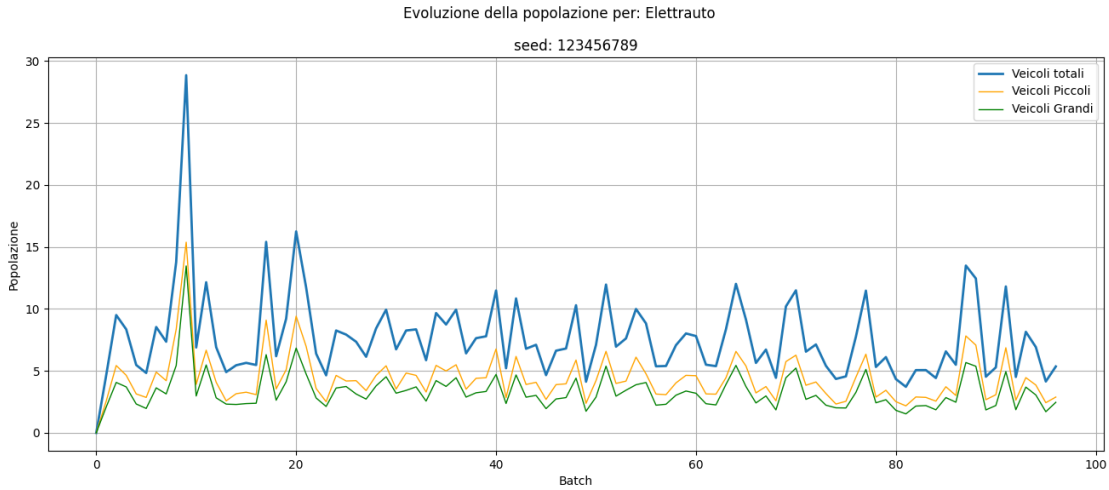
$E(T_s) = E(T_q) + E(S_i) = 45633,81639$ sec

$E(N_s) = E(T_s) \cdot \lambda_1 = 7,529579704$

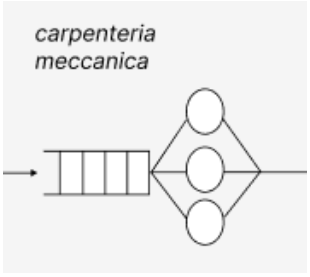
I risultati prodotti dalla simulazione sono:

```
Elettrauto
Statistiche per E[Tq] Critical endpoints 41520.12673856671 +/- 4097.787074404759
statistiche per E[Nq] Critical endpoints 6.881936043926111 +/- 0.7104929042694466
statistiche per rho Critical endpoints 0.8842020033725269 +/- 0.01987827926057042
statistiche per E[Ts] Critical endpoints 46890.28538921787 +/- 4138.806865617686
statistiche per E[Ns] Critical endpoints 7.76613804729864 +/- 0.719582852907409
```

Visibili anche dal grafico prodotto, il quale si concentra sul mostrare la popolazione completa $E(N_s)$ nel centro in esame:



Carpenteria



$\lambda_3 = 0.00165 \cdot 0.2$ job/sec

$E(S_i) = 5400$ sec

$N = 3$

$E(S) = \frac{E(S_i)}{N} = 1800 \text{ sec}$

$\mu = \frac{1}{E(S)} = 0,000555556 \text{ job/sec}$

$\rho = \frac{\lambda}{\mu} = 0,594$

$p_0 = \left[\left(\sum_{i=0}^{N-1} \frac{(N\cdot\rho)^i}{i!} + \frac{(N\cdot\rho)^N}{N!(1-\rho)}\right)\right]^{-1} = 0,14941555$

$P_q = \frac{(N\rho)^N \cdot p_0^{-1}}{N!(1-\rho)} = 0,347089609$

$E(T_q) = \frac{P_q \cdot E(S)}{1-\rho} = 1538,820926 \text{ sec}$

$E(N_q) = E(T_q) \cdot \lambda_1 = 0,507810906$

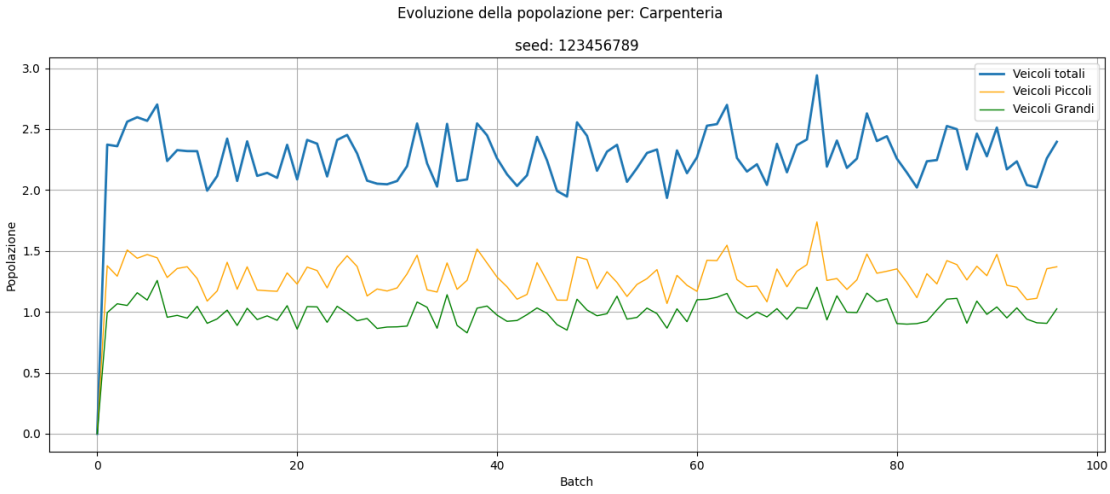
$E(T_s) = E(T_q) + E(S_i) = 6938,820926 \text{ sec}$

$E(N_s) = E(T_s) \cdot \lambda_1 = 2,289810906$

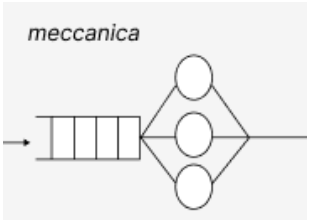
I risultati prodotti dalla simulazione sono:

```
Carpenteria
Statistiche per E[Tq] Critical endpoints 1547.1157559560882 +/- 91.04488094736809
statistiche per E[Nq] Critical endpoints 0.5109015956412454 +/- 0.031065894409843108
statistiche per rho Critical endpoints 0.5914936479872669 +/- 0.013095126221282156
statistiche per E[Ts] Critical endpoints 6936.1419185792065 +/- 178.04375990253428
statistiche per E[Ns] Critical endpoints 2.2853825396030456 +/- 0.0618491347644086
```

Visibili anche dal grafico prodotto, il quale si concentra sul mostrare la popolazione completa $E(N_s)$ nel centro in esame:



Meccanica



$\lambda_3 = 0.00165 \cdot 0.3 \text{ job/sec}$

$E(S_i) = 5400 \text{ sec}$

$N = 3$

$E(S) = \frac{E(S_i)}{N} = 1800 \text{ sec}$

$\mu = \frac{1}{E(S)} = 0,000555556 \text{ job/sec}$

$\rho = \frac{\lambda}{\mu} = 0,891$

$p_0 = \left[\left(\sum_{i=0}^{N-1} \frac{(N\cdot\rho)^i}{i!} + \frac{(N\cdot\rho)^N}{N!(1-\rho)}\right)\right]^{-1} = 0,02743642$

$P_q = \frac{(N\rho)^N \cdot p_0^{-1}}{N!(1-\rho)} = 0,801210392$

$E(T_q) = \frac{P_q \cdot E(S)}{1-\rho} = 13230,99729 \text{ sec}$

$E(N_q) = E(T_q) \cdot \lambda_1 = 6,549343659$

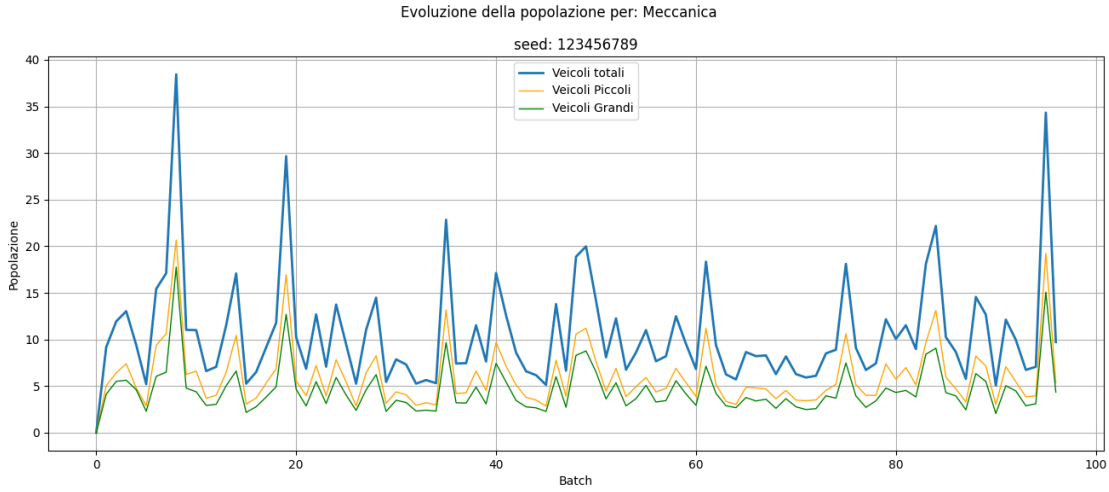
$E(T_s) = E(T_q) + E(S_i) = 18630,99729 \text{ sec}$

$E(N_s) = E(T_s) \cdot \lambda_1 = 9,222343659$

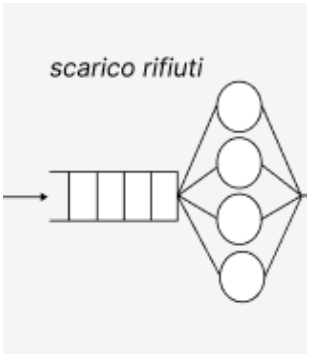
I risultati prodotti dalla simulazione sono:

```
Meccanica
Statistiche per E[Tq] Critical endpoints 15676.3882447926 +/- 2457.296648198602
statistiche per E[Nq] Critical endpoints 8.023194831677312 +/- 1.485474535675924
statistiche per rho Critical endpoints 0.8985906428860488 +/- 0.0204304780384617
statistiche per E[Ts] Critical endpoints 21076.984326560934 +/- 2494.288472448443
statistiche per E[Ns] Critical endpoints 10.438966760335455 +/- 1.5129865984680073
```

Visibili anche dal grafico prodotto, il quale si concentra sul mostrare la popolazione completa $E(N_s)$ nel centro in esame:



Scarico



$\lambda_3 = 0.0055$ job/sec

$E(S_i) = 600$ sec

$N = 4$

$E(S) = \frac{E(S_i)}{N} = 150$ sec

$\mu = \frac{1}{E(S)} = 0,006666667$ job/sec

$\rho = \frac{\lambda}{\mu} = 0,825$

$p_0 = \left[\left(\sum_{i=0}^{N-1} \frac{(N\cdot\rho)^i}{i!} + \frac{(N\cdot\rho)^N}{N!(1-\rho)}\right)\right]^{-1} = 0,02274241$

$P_q = \frac{(N\rho)^N \cdot p_0^{-1}}{N!(1-\rho)} = 0,642159554$

$E(T_q) = \frac{P_q \cdot E(S)}{1-\rho} = 550,4224751$ sec

$E(N_q) = E(T_q) \cdot \lambda_1 = 3,027323613$

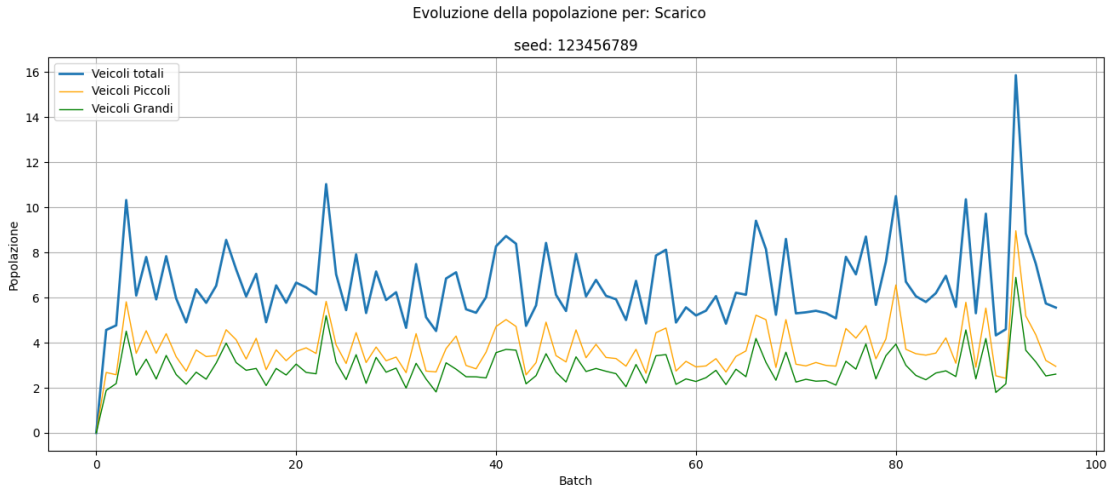
$E(T_s) = E(T_q) + E(S_i) = 1150,422475$

$E(N_s) = E(T_s) \cdot \lambda_1 = 6,327323613$ sec

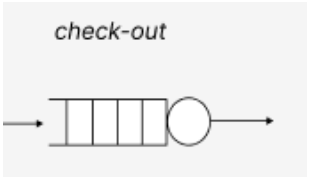
I risultati prodotti dalla simulazione sono:

```
Scarico
Statistiche per E[Tq] Critical endpoints 589.5876595998931 +/- 58.773053368036294
statistiche per E[Nq] Critical endpoints 3.297107770606008 +/- 0.34338301335711396
statistiche per rho Critical endpoints 0.8341608917070502 +/- 0.01898588540218348
statistiche per E[Ts] Critical endpoints 1190.8514640683547 +/- 64.49768771257095
statistiche per E[Ns] Critical endpoints 6.633751337434209 +/- 0.3852337815415976
```

Visibili anche dal grafico prodotto, il quale si concentra sul mostrare la popolazione completa $E(N_s)$ nel centro in esame:



Checkout



$\lambda_3 = 0.0055 \cdot 0.1$ job/sec

$E(S_i) = 1200$ sec

$N = 1$

$E(S) = \frac{E(S_i)}{N} = 1200$ sec

$\mu = \frac{1}{E(S)} = 0,000833333$ job/sec

$\rho = \frac{\lambda}{\mu} = 0,66$

$E(T_q) = \frac{\rho \cdot E(S)}{1-\rho} = 2329.41$ sec

$E(N_q) = E(T_q) \cdot \lambda_1 = 1,281176$

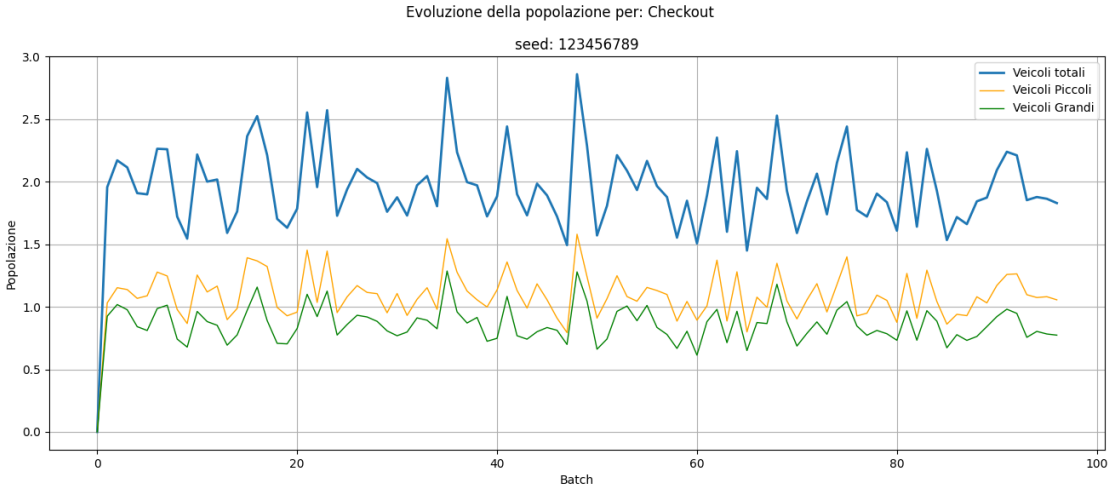
$E(T_s) = E(T_q) + E(S_i) = 3529,41$ sec

$E(N_s) = E(T_s) \cdot \lambda_1 = 1,94095$

I risultati prodotti dalla simulazione sono:

```
ControllerCheckout
Statistiche per E[Tq] Critical endpoints 2344.617550461787 +/- 102.33200377210726
statistiche per E[Nq] Critical endpoints 1.2963365414503911 +/- 0.06060036910882929
statistiche per rho Critical endpoints 0.6651665731794177 +/- 0.014981780537475765
statistiche per E[Ts] Critical endpoints 3551.078737075841 +/- 119.40279900712218
statistiche per E[Ns] Critical endpoints 1.961503114629809 +/- 0.07151781449038921
```

Visibili anche dal grafico prodotto, il quale si concentra sul mostrare la popolazione completa $E(N_s)$ nel centro in esame:



Controlli di consistenza

Nei risultati proposti, oltre alla corrispondenza tra valori prodotti dalla simulazione, e valori analitici, si può osservare anche il rispetto dei seguenti criteri di consistenza:

- $E[T_s] = E[T_q] + E[Si]$
- $E[N_s] = E[N_q] + m \cdot \rho$
- $0 < \rho \leq 1$

9 Validazione

Arrivati a questa sezione, ci chiediamo se il modello computazionale è consistente con il sistema analizzato. Per iniziare, dai report forniti dall'azienda, mediamente, nell'arco di 3 ore, entrano nel sistema circa 60 mezzi. Tipicamente, non c'è un'eccessiva scissione del lavoro in fasce orarie, motivo per il quale i test sono condotti nell'arco di un giorno intero. Ciò ci porta a pensare che, mediamente, nell'arco delle 24 ore, i mezzi entranti nel sistema si attestino intorno ai 480, considerando il fatto che un mezzo possa passare più volte nel sistema.

Dalle nostre simulazioni, con servizi *normali troncati*, e tecnica della replicazione, gli eventi di arrivo nel sistema oscillano tra un minimo di 466 arrivi ad un massimo di 521, non distaccandosi troppo dal valore appena discusso. Altri aspetti legati alla validazione sono legati al comportamento di alcuni centri:

- **Scarico**: Nonostante i tempi di servizio ridotti, ci aspettiamo comunque un afflusso di veicoli al centro di scarico, dato che tutti i mezzi del sistema vi convergono. Inoltre, alla chiusura delle porte, vi accederanno tutti i mezzi provenienti dalle officine, portando quindi, nelle fasi finali della simulazione, del lavoro da espletare in questo centro.
- **Accettazione**: Essendo uno dei prin-

cipali centri di ingresso, e presentando tempi di servizio di ridotti, oltre che numerosi serventi, ci aspettiamo una popolazione totale ridotta e costante, aspetto visibile nei grafici precedenti.

- ❒ **Officina**: I centri con maggior popolazione sono *meccanica* ed *elettrauto*. Ciò è coerente sia con le probabilità assegnate ad ogni officina, sia ai report ottenuti dal polo impiantistico. In particolare, si conferma che le tipologie di guasti più frequenti sono quelli associati al primo centro citato:

Descrizione RdI
PERDITA OLIO GRUPPO DI PRESA

Descrizione RdI
PERDITA ARIA IMPIANTO

seguiti da guasti associati all'officina *elettrauto*, che coinvolge spesso il controller dei mezzi.

10 Design degli esperimenti

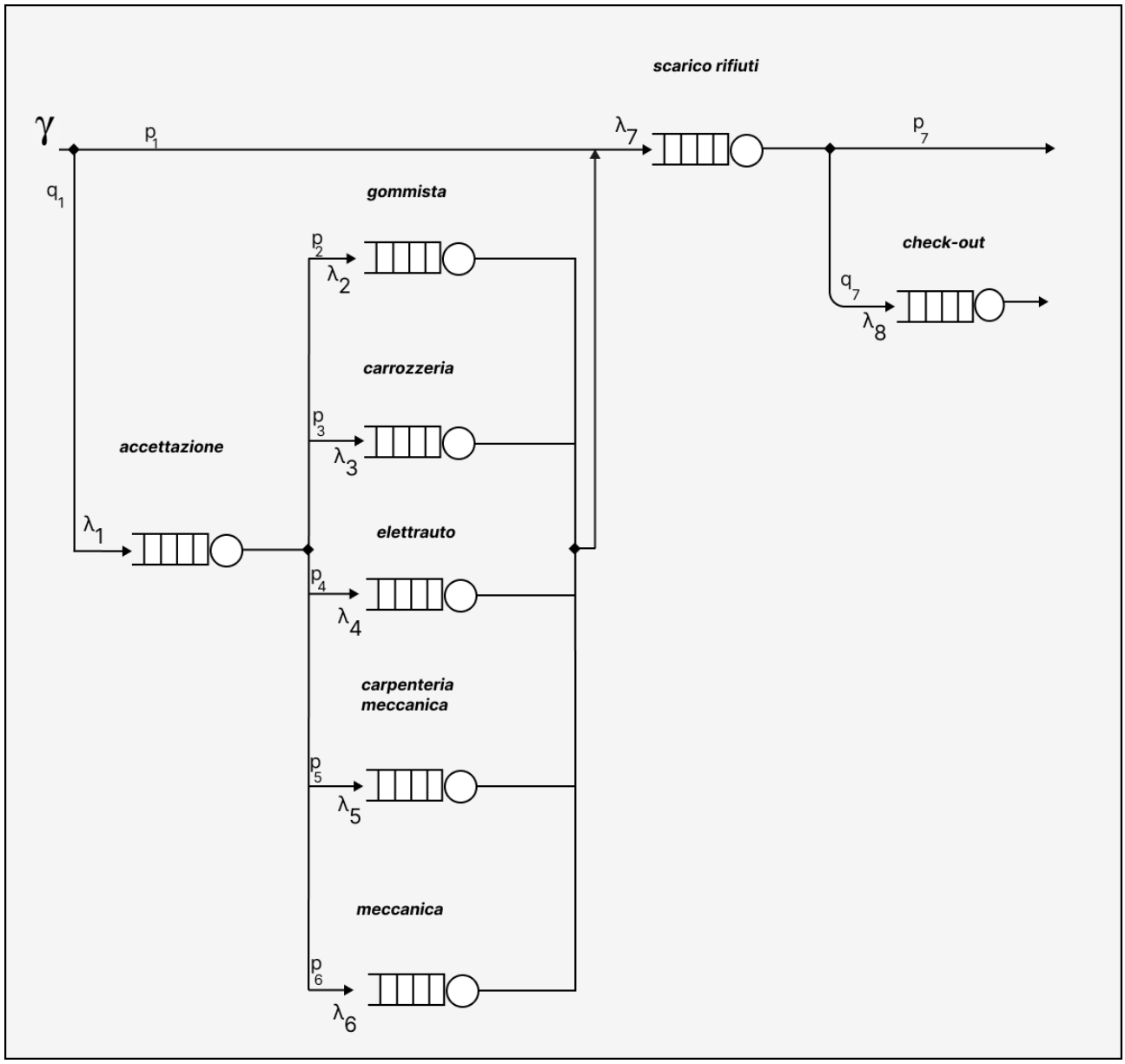
La progettazione degli esperimenti si articola in tre fasi principali:

1) **Analisi del Collo di Bottiglia:** Nella prima fase, si esegue un'analisi operazionale per calcolare la popolazione media per ciascun centro del sistema. L'obiettivo è identificare i centri con la domanda più elevata, che rappresentano il collo di bottiglia del sistema.

2) **Simulazione a Orizzonte Finito:** La simulazione a orizzonte finito coinvolge il sistema simulato per 24 ore. Si fa uso del metodo delle *Replicazioni*, nella quale vengono prodotte 96 esecuzioni per ottenere statistiche mediate.

3) **Simulazione a Orizzonte Infinito:** La simulazione a orizzonte infinito prevede la simulazione del sistema per un periodo molto superiore a quello reale. Il metodo delle Batch Means viene impiegato per suddividere la run in 96 batch di dimensione 1080 job. La scelta della dimensione del batch, è stata guidata dal criterio di "*Banks, Carson, Nelson, and Nicol* del 2001, secondo cui il valore di b deve aumentare fintanto che il valore di lag-autocorrelation, tra le medie dei batch, non scenda sotto 0.2. Ciò è stato verificato tramite l'analisi dei dati di output implementata nella classe Acs.java, fornita sempre dal libro *Discrete Event Simulation - A first course* di Lemmis Park.

10.1 Analisi del collo di bottiglia



Servizi medi dei centri nel sistema

Disclaimer: identifichiamo con λ_i il tasso di ingresso nel centro i, e con γ il tasso di ingresso nel sistema. Per questa analisi, impostiamo tutti i centri con un unico servente.

$S_1 = \frac{1}{\mu_1} = 600 \text{ sec}$

$S_2 = \frac{1}{\mu_2} = 3600 \text{ sec}$

$S_3 = \frac{1}{\mu_4} = 5400 \text{ sec}$

$S_4 = \frac{1}{\mu_4} = 5400 \text{ sec}$

$S_5 = \frac{1}{\mu_5} = 5400 \text{ sec}$

$S_6 = \frac{1}{\mu_6} = 5400 \text{ sec}$

$S_7 = \frac{1}{\mu_8} = 600 \text{ sec}$

$S_8 = \frac{1}{\mu_7} = 1200 \text{ sec}$

Calcolo del numero medio di visite ai centri del sistema

$\lambda_1 = (1 - p_1) \cdot \gamma = 0.3 \cdot \gamma$
$\lambda_2 = p_2 \cdot \lambda_1 = 0.3 \cdot (0.3 \cdot \gamma) = 0.09 \cdot \gamma$
$\lambda_3 = p_3 \cdot \lambda_1 = 0.1 \cdot (0.3 \cdot \gamma) = 0.03 \cdot \gamma$
$\lambda_4 = p_4 \cdot \lambda_1 = 0.1 \cdot (0.3 \cdot \gamma) = 0.03 \cdot \gamma$
$\lambda_5 = p_5 \cdot \lambda_1 = 0.2 \cdot (0.3 \cdot \gamma) = 0.06 \cdot \gamma$
$\lambda_6 = p_6 \cdot \lambda_1 = 0.3 \cdot (0.3 \cdot \gamma) = 0.09 \cdot \gamma$
$\lambda_7 = p_1\gamma + \lambda_1 p_2 + \lambda_1 p_3 + \lambda_1 p_4 + \lambda_1 p_5 + \lambda_1 p_6 = p_1\gamma + \lambda_1(p_2 + p_3 + p_4 + p_5 + p_6) = p_1\gamma + \lambda_1 = p_1\gamma + (1 - p_1)\gamma = \gamma$
$\lambda_8 = \lambda_7 \cdot (1 - p_7) = 0.1 \cdot \gamma$

Da cui ricaviamo:

$v_1 = \frac{\lambda_1}{\gamma} = 0.3$

$v_2 = \frac{\lambda_2}{\gamma} = 0.09$

$v_3 = \frac{\lambda_3}{\gamma} = 0.03$

$v_4 = \frac{\lambda_4}{\gamma} = 0.03$

$v_5 = \frac{\lambda_5}{\gamma} = 0.06$

$v_6 = \frac{\lambda_6}{\gamma} = 0.09$

$v_7 = \frac{\lambda_7}{\gamma} = 1$

$v_8 = \frac{\lambda_8}{\gamma} = 0.1$

Passiamo al calcolo della domanda:

$D_1 = v_1 \cdot S_1 = 0.3 \cdot 600 = 180 \text{ sec}$

$D_2 = v_2 \cdot S_2 = 0.09 \cdot 3600 = 324 \text{ sec}$

$D_3 = v_3 \cdot S_3 = 0.03 \cdot 5400 = 162 \text{ sec}$

$D_4 = v_4 \cdot S_4 = 0.03 \cdot 5400 = 162 \text{ sec}$

$D_5 = v_5 \cdot S_5 = 0.06 \cdot 5400 = 324 \text{ sec}$

$D_6 = v_6 \cdot S_6 = 0.09 \cdot 5400 = 486 \text{ sec}$

$D_7 = v_7 \cdot S_7 = 1 \cdot 600 = 600 \text{ sec}$

$D_8 = v_8 \cdot S_8 = 0.1 \cdot 1200 = 120 \text{ sec}$

Questi valori sono consistenti con i ottenuti dalla simulazione:

	Accettazione	Gommista	Carrozzeria	Elettrauti	Carpenteria	Meccanica	Scarico	Checkout
V	0.300158	0.090122	0.029928	0.029888	0.059984	0.090235	1	0.099851
D (sec)	180.094748	324.439330	161.610493	161.395675	323.915256	487.270761	600	119.821711

I risultati sono coerenti con il sistema reale in quanto il centro di *Scarico*, essendo attraversato da tutti i job nel sistema, presenta una domanda complessiva maggiore. Tra le officine, il centro *Meccanica* è il più frequentato, evidenziando una domanda media superiore. Questo risultato è in linea con le aspettative, considerando che il centro *Meccanica* risulta essere uno dei più frequentati tra le officine, con un tempo di servizio medio comparabile o addirittura superiore alle altre officine.

10.2 Simulazione ad orizzonte finito

La simulazione è stata effettuata sul sistema con la sua configurazione standard di serventi. In questa fase, si è utilizzata la tecnica della **Replicazione**. Le replicazioni sono utilizzate per generare stime indipendenti della stessa statistica transitoria. Pertanto, il seed iniziale per ogni replicazione viene scelto in modo che non ci sia sovrapposizione tra le replicazioni nella sequenza di numeri casuali utilizzata. Il modo standard per evitare questa sovrapposizione è utilizzare lo stato finale di ciascuna sequenza di numeri casuali (ottenuta dalla classe *rngs*) da una replicazione come stato iniziale per la replicazione successiva. Ciò viene automaticamente realizzato chiamando PlantSeeds una volta al di fuori del ciclo principale di replicazione. I risultati prodotti, per 96 replicazioni, sono:

Categoria	E[Tq]	E[Nq]	ρ	E[Ts]	E[Ns]
Accettazione	2.9386 ± 0.5618	0.0051 ± 0.0010	0.2517 ± 0.0067	601.99 ± 12.67	1.012 ± 0.0273
Carpenteria	442.171 ± 54.983	0.118 ± 0.015	0.358 ± 0.008	5822.98 ± 144.57	1.554 ± 0.043
Carrozzeria	900.80 ± 119.23	0.211 ± 0.028	0.316 ± 0.008	6294.6 ± 190.65	1.476 ± 0.048
Elettrauto	5393.59 ± 571.06	0.780 ± 0.088	0.193 ± 0.005	10801.09 ± 621.9	1.554 ± 0.099
Gommista	757.51 ± 89.47	0.39 ± 0.048	0.45 ± 0.012	4365.09 ± 131.76	2.189 ± 0.077
Meccanica	2681.77 ± 327.47	1.16 ± 0.156	0.57 ± 0.015	8077.03 ± 371.14	3.423 ± 0.199
Scarico	902.257 ± 165.43	4.265 ± 0.836	0.790 ± 0.019	1586.27 ± 167.91	7.425 ± 0.866
Checkout	815.467 ± 39.429	0.417 ± 0.021	0.481 ± 0.011	1756.34 ± 52.715	0.898 ± 0.0299

I risultati prodotti sono coerenti con le nostre attese:

- L' *accettazione* riesce a smaltire bene il carico di lavoro, avendo numerosi serventi e un flusso in entrata ampiamente gestibile.
- Lo *scarico*, essendo il centro in cui passano tutti i veicoli, si mostra come il nodo con popolazione maggiore.
- Le officine di tipo *Gommista* e *Meccanica*, le quali presentano più probabilità di essere accedute, sono quelle che presentano una popolazione maggiore.

L'intervallo di confidenza è stato calcolato con la seguente formula:

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

dove:

- s è la deviazione standard,
- $n-1$ è il numero di ripetizioni considerate,
- t^* è il valore critico ottenuto dall'inverso della distribuzione di Student con $n-1$ gradi di libertà. In particolare, t^* è calcolato come $t^* = \text{idfStudent}(n-1, 1-\alpha/2)$. Il parametro di confidenza α, solitamente scelto come 0.05 (5%), determina quanto ci si può scostare dalla media in positivo o in negativo.

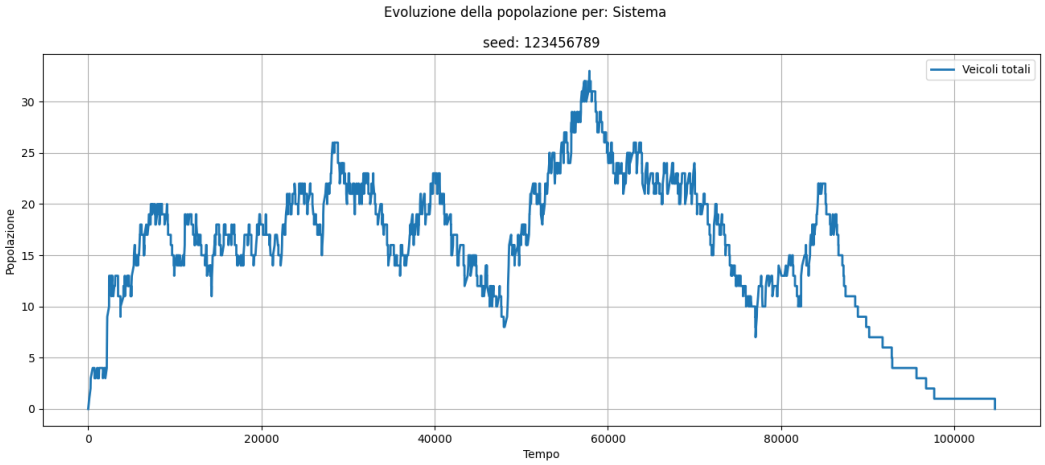
A livello di codice, il calcolo di t^* è effettuato utilizzando la classe `Rvms.java`, la quale fornisce gli strumenti necessari. In particolare, viene utilizzata la funzione `idfStudent(long n, double u)`, che calcola la distribuzione inversa di una variabile di Student, passando come parametri $n-1$ e α.

Una volta ottenuto t^*, che è uguale per ogni statistica poiché il numero di ripetizioni è costante, è possibile calcolare la deviazione standard. Questa viene ottenuta prendendo le singole esecuzioni e calcolando la differenza rispetto alla media delle medie delle esecuzioni.

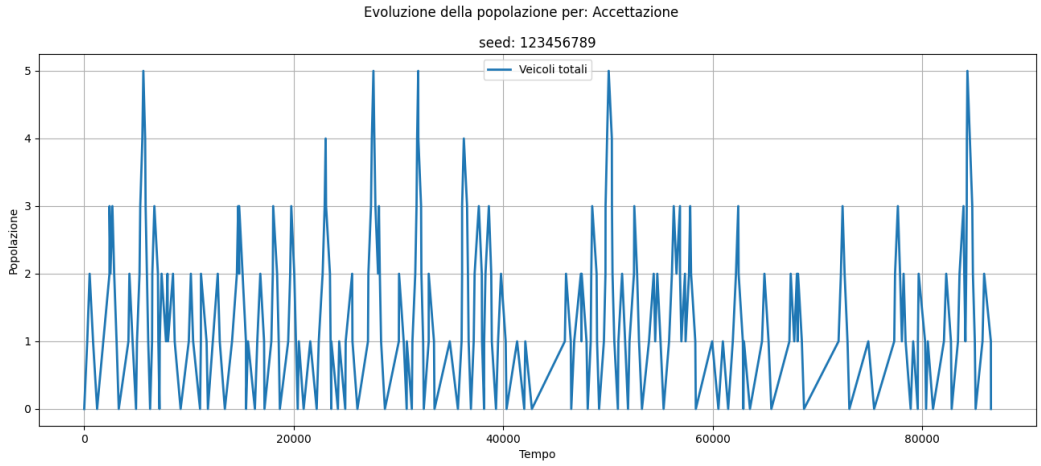
Come già accennato più volte, il nostro obiettivo non è volto a migliorare le tempistiche del sistema, bensì a fornire, esternamente al sistema, un numero di mezzi sufficiente ad espletare i servizi di

pulizia. Per valutare questo criterio, è stata aggiunta una nuova metrica, la quale conta il numero di volte in cui nel sistema non si osservano questi Quality Of Service.

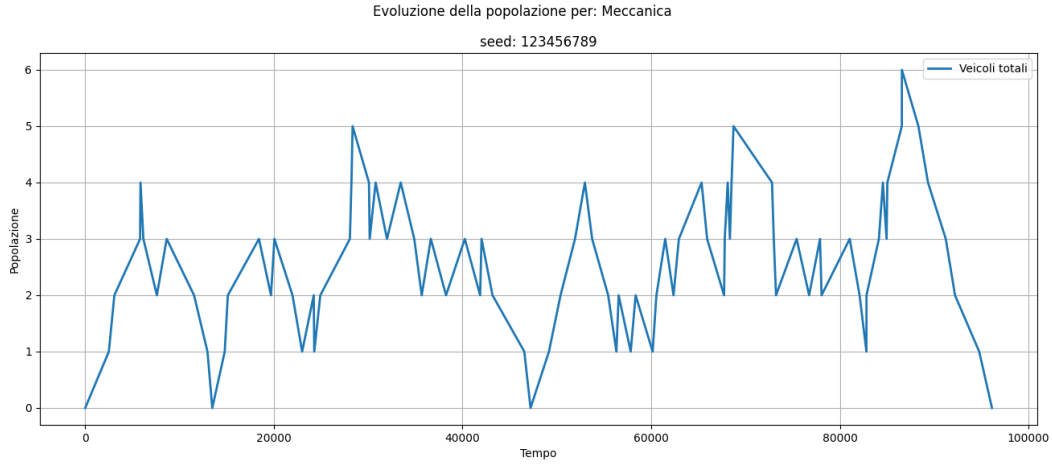
Dall'analisi transiente si nota che se il sistema parte da vuoto (tutti i mezzi a disposizione) nell'arco delle 24 ore questo limite non viene mai superato. Ciò è rafforzato anche dall'evidenza grafica esposta di seguito:



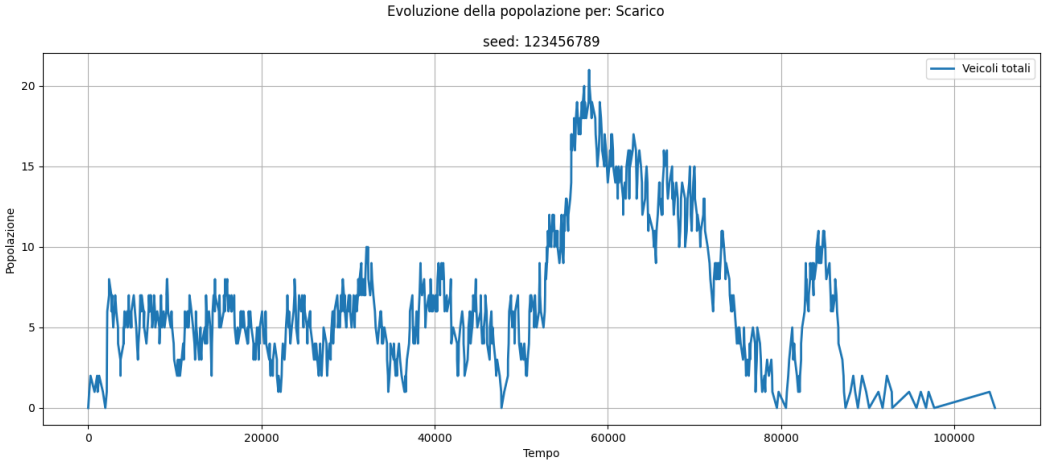
Tra i vari centri, esaminiamo l'**Accettazione**, che ha servizi ridotti e svariati serventi, il che la porta ad espletare in modo rapido la diagnosi dei mezzi (questo porta il grafico a toccare spesso il punto sulle ordinate associato alla popolazione 0).



Mentre per l'**Officina Meccanica** tale condizione è più complessa da raggiungere, portando ad una popolazione media maggiore.

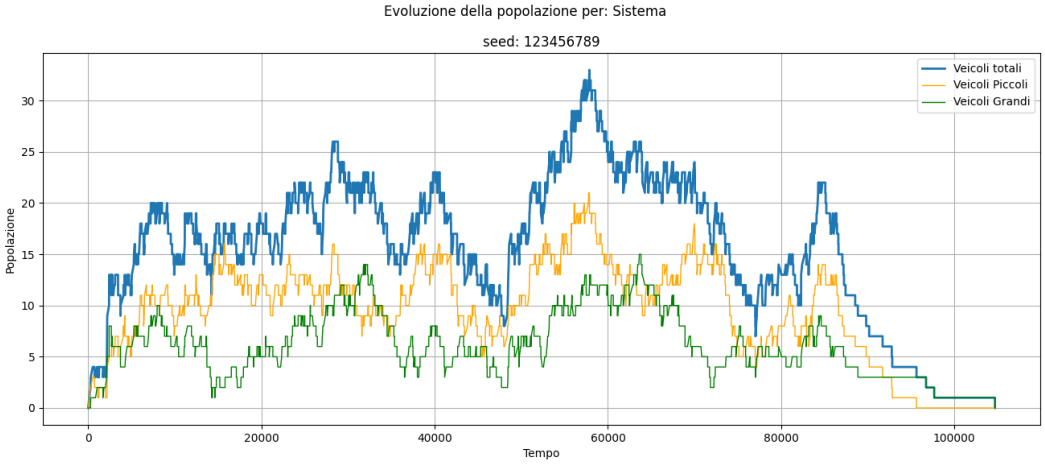


Un centro di rilievo, come già accennato più volte, è quello dello **Scarico**, il quale, come già detto, presenta domande e flusso in entrata maggiori.

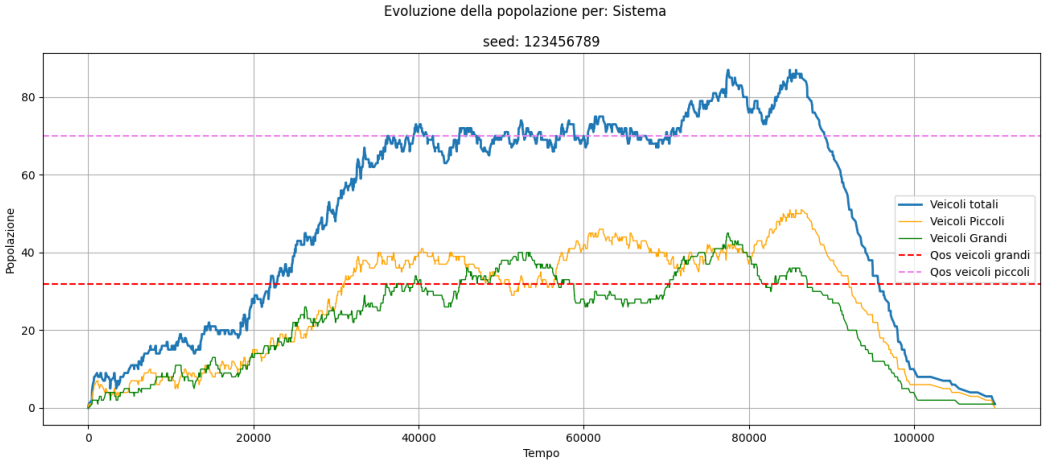


Dati e grafici sono stati prodotti con servizi basati su distribuzioni *Gaussiane troncate*. Anche in questo caso, la simulazione nel transiente produce caratteristiche vicine ai risultati attesi in questa fase. Tuttavia, la simulazione ad orizzonte finito non ci è molto di aiuto nell'analisi del nostro obiettivo: permettere al sistema di partire da vuoto, e fornirgli il tempo necessario per espletare tutte le richieste sono delle condizioni troppo ottimistiche.

Volendo suddividere la popolazione in due tipologie di veicoli, ciò che si ottiene nella prima run è:



e, a maggior dimostrazione dell'indipendenza delle run, ne esaminiamo una in cui ci sono molteplici violazioni dei QoS (run numero 92), in cui il QoS per i mezzi grandi viene violato più volte:



Dalle nostre simulazioni, su 96 run totali, il requisito di QoS (inteso come inadempimento della disponibilità di mezzi piccoli o di mezzi grandi) è stato violato 536 volte, sempre da veicoli di tipo 2. Tale violazione verrà analizzata in modo migliore nella simulazione ad orizzonte infinito, nella quale il sistema viene simulato in condizioni più vicine alla realtà, in quanto nel polo impiantistico vi è un lavoro continuo.

```
Numero di volte in cui è stato superato il limite di veicoli nel sistema: 536
Per i veicoli di tipo 1: 0 e per i veicoli di tipo 2:  536
```

10.3 Simulazione ad orizzonte infinito

La simulazione ad orizzonte infinito è stata effettuata sul sistema con la sua configurazione standard di serventi. In questa fase si è utilizzata la tecnica delle **Batch Means**. Questa viene utilizzata per vedere come si comporta il sistema nello stato stazionario cercando però di ridurre l'influenza dello stato iniziale (essendo la run molto lunga, lo stato iniziale viene perso poiché presente solo a inizio run). Per fare ciò viene presa una simulazione molto lunga (simulata con tempo infinito, `Double.MAX_VALUE`) che viene poi divisa in batch. Lo stato in-

iziale di ogni batch è dato da quello finale del precedente. Dato il campione di eventi questa tecnica prevede di dividerlo in **K** batch di lunghezza **B** per poi calcolare la media di ogni singolo batch.

La teoria prevede che B debba essere abbastanza grande da far tendere la media delle RVs considerate ad una normale e da ridurre l'auto-correlazione presente nelle time-sequences.

Per la scelta di questi valori ci siamo aiutati con la classe `Acs.java`, utile, data una serie, per calcolare l'auto-correlazione dei valori presenti in essa. In particolare, abbiamo visto che i valori di B e K per avere auto-correlazione inferiore a 0.2 erano rispettivamente 1080 e 96.

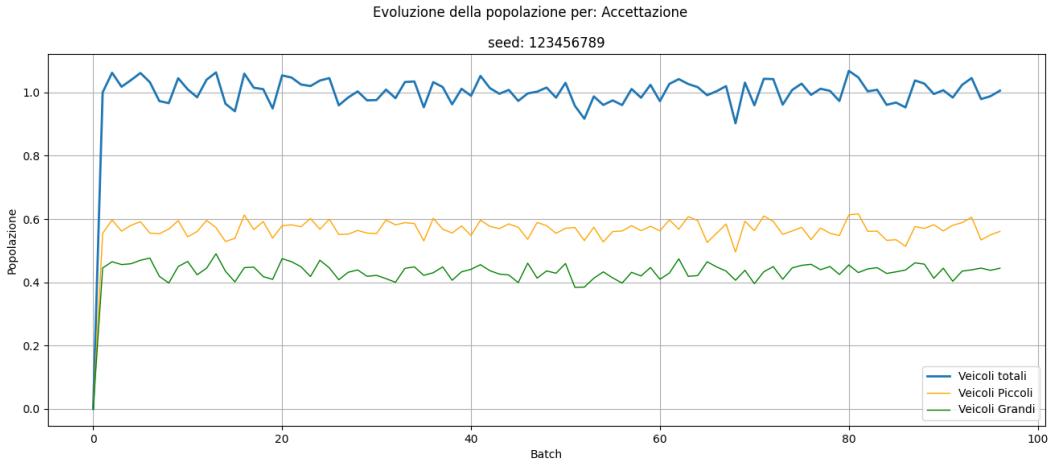
Una volta fissati questi valori abbiamo ottenuto dalla simulazione i seguenti risultati:

Categoria	E[Tq]	E[Nq]	ρ	E[Ts]	E[Ns]
Accettazione	2.662 ± 0.254	0.004 ± 0.000	0.250 ± 0.005	602.871 ± 12.550	1.004 ± 0.022
Carpenteria	843.613 ± 34.624	0.279 ± 0.012	0.593 ± 0.013	6242.42 ± 133.53	2.057 ± 0.047
Carrozzeria	110.815 ± 6.625	0.018 ± 0.001	0.297 ± 0.006	5510.71 ± 114.97	0.909 ± 0.020
Elettrauto	22841.7 ± 1780.2	3.789 ± 0.313	0.890 ± 0.019	28246.8 ± 1815.0	4.679 ± 0.322
Gommista	604.079 ± 26.540	0.302 ± 0.014	0.599 ± 0.013	4207.65 ± 91.02	2.101 ± 0.049
Meccanica	7972.74 ± 987.78	4.032 ± 0.530	0.899 ± 0.020	13370.9 ± 1013.3	6.728 ± 0.553
Scarico	1568.72 ± 337.61	8.867 ± 1.982	0.947 ± 0.021	2251.73 ± 339.38	12.657 ± 2.006
Checkout	451.848 ± 14.067	0.250 ± 0.009	0.519 ± 0.011	1393.64 ± 30.89	0.769 ± 0.019

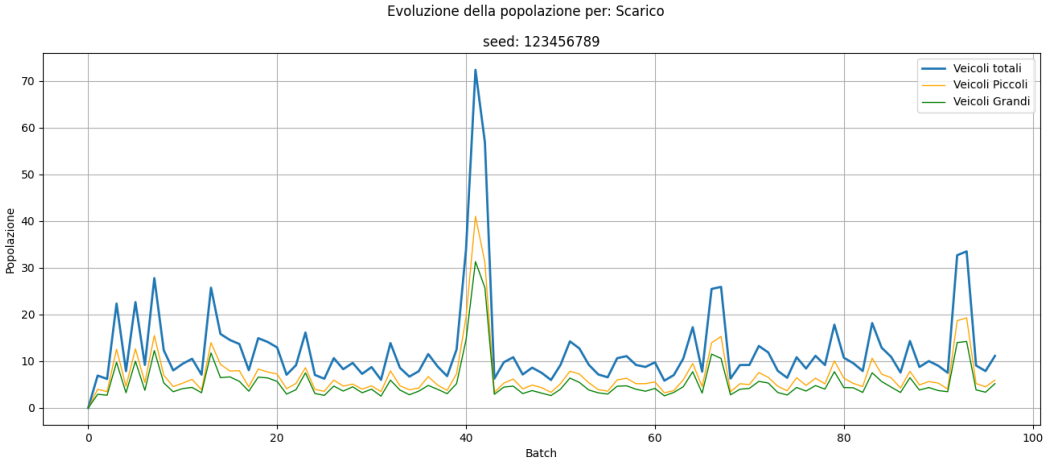
Anche in questo caso i risultati sono coerenti con le nostre attese per gli stessi motivi illustrati in precedenza.

La simulazione ha prodotto i seguenti grafici:

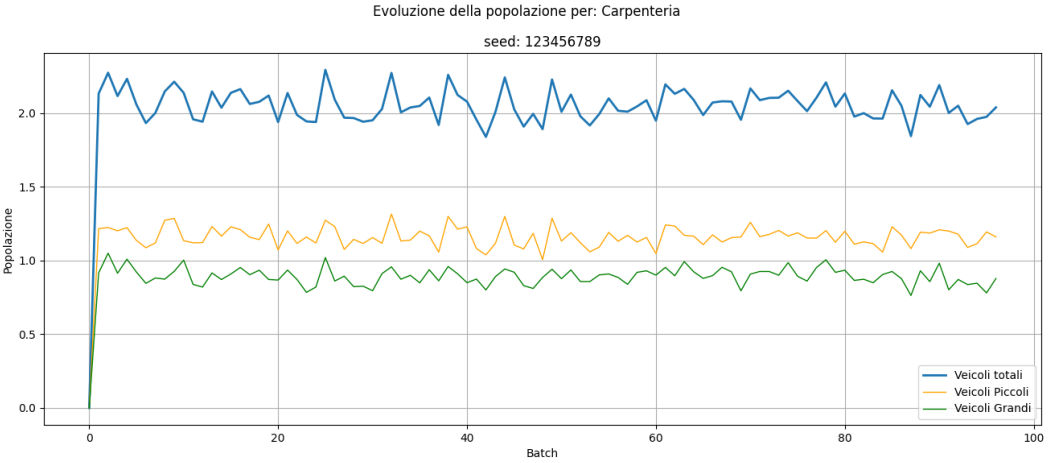
L'**accettazione** presenta un andamento costante, e come è possibile vedere, troviamo sempre una maggior quantità di mezzi di piccole dimensioni.

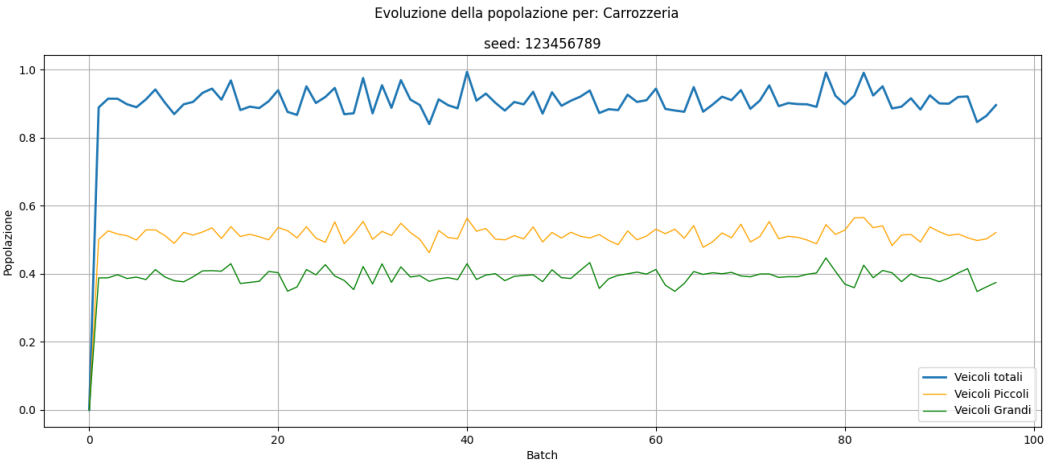


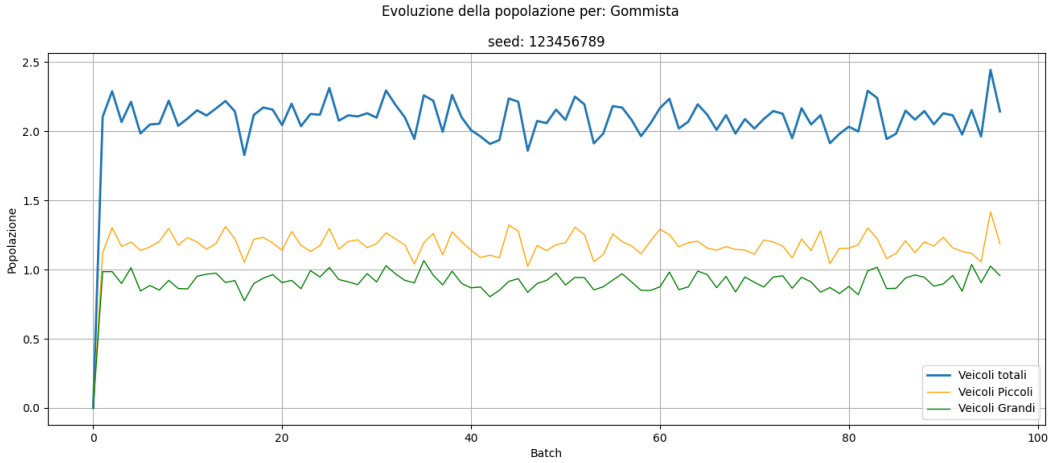
Lo **Scarico** presenta dei picchi di domanda in alcune fasi della simulazione. Questi picchi sono prevedibili, in quanto i mezzi passanti per le officine, le quali hanno in media stessi tempi di servizio, verranno rilasciati dopo un certo tempo, andandosi a sommare ai mezzi entrati direttamente nello scarico.



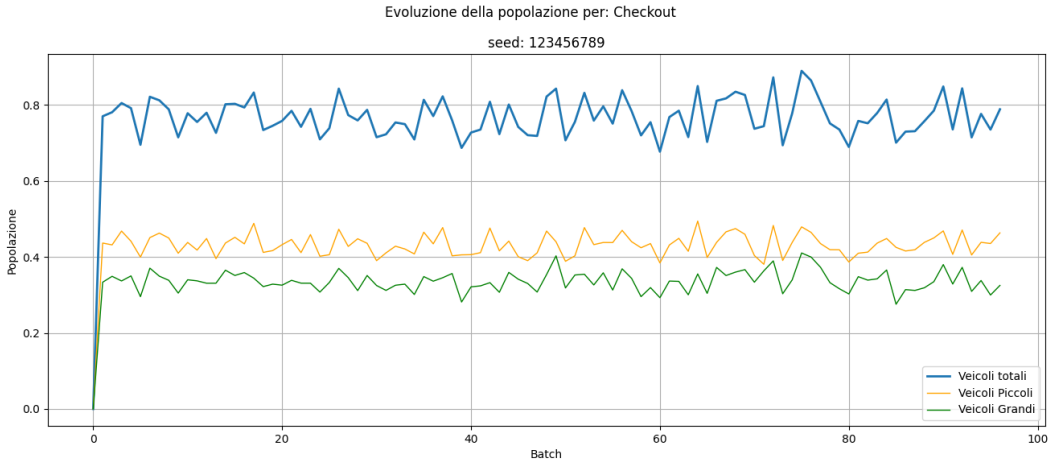
Per i centri delle officine **Carpenteria**,**Carrozzeria**,**Gommista**, l'andamento è concorde con le stime, mostrando una richiesta in linea con le aspettative nata dai dati ottenuti.



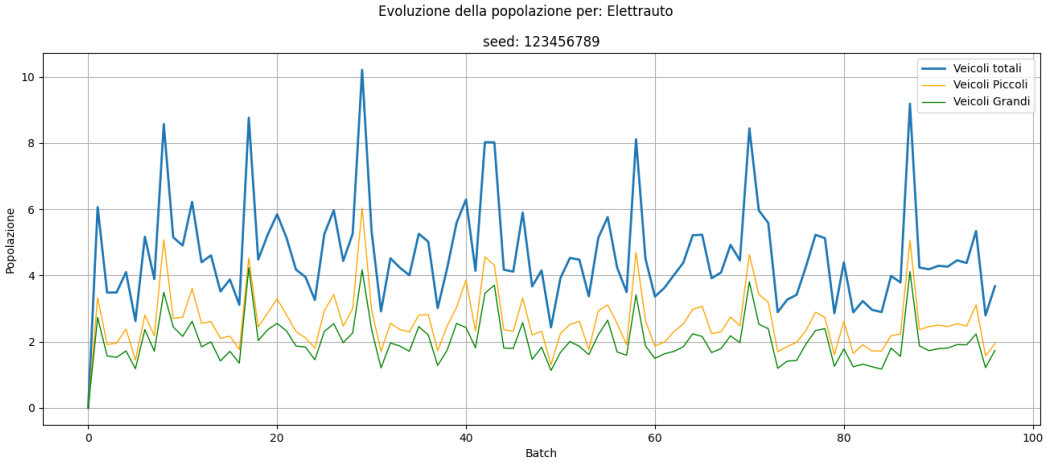




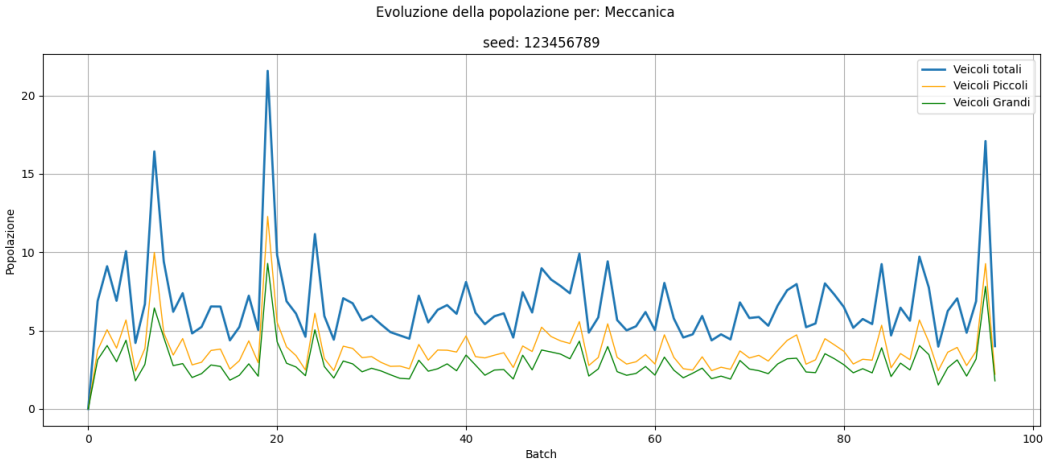
Il **Checkout** è un centro che tipicamente presenta una popolazione inferiore, in quanto il rifornimento di un mezzo è possibile anche esternamente al sistema, e la sanitizzazione, a meno di rifiuti di particolare entità, non è obbligatoriamente imposta.



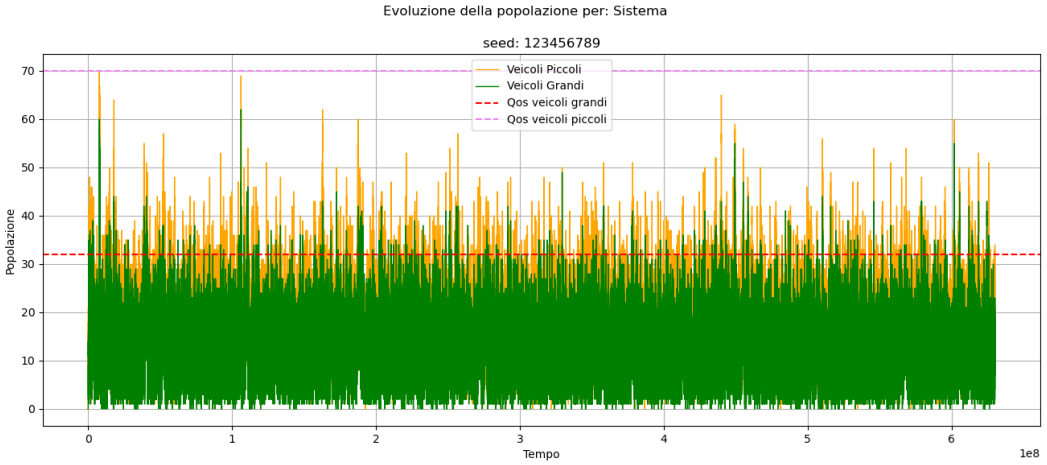
Concludiamo la trattazione con le due officine che presentano una popolazione media abbastanza elevata: **Elettrauto** e **Meccanica**. Per il primo, abbiamo una probabilità di visita minore, ma il tempo in cui si rimane nel centro è non indifferente.



Per l'officina **Meccanica**, abbiamo una maggior probabilità di accesso, coadiuvata da un tempo di servizio medio dello stesso tipo dell' Elettrauto, e ciò si traduce, com'è possibile vedere in figura, in una popolazione media maggiore.



Esaminando il comportamento generale del sistema, visto come insieme di centri, si ha che:



11 Conclusioni

In questa simulazione, viene evidenziata la criticità del limite dei veicoli imposto dai QoS, il quale non viene rispettato per 59285 volte (dove il numero di volte viene aggiornato per ogni nuovo evento che si ha nel sistema).

```
Numero di volte in cui è stato superato il limite di veicoli nel sistema: 59285
Per i veicoli di tipo 1: 0 e per i veicoli di tipo 2:  59285
```

In particolare, la simulazione evidenzia che il limite viene superato tutte le volte dai veicoli di tipo 2 (*mezzi grandi*). Anche questo è coerente con quello che ci aspettavamo, in quanto il vincolo imposto su questo tipo di veicoli, risulta più stringente. Una prima conclusione che possiamo ottenere da queste simulazioni è che la gestione attuale del Polo Impiantistico, nella quale la catena di servizio è dettata unicamente dall'ordine con cui i mezzi entrano nel sistema, favorisce eccessivamente i veicoli di piccole dimensioni. Nella realtà, ciò si traduce in un servizio eccessivamente sottodimensionato in aree che richiedono veicoli più capienti. Tale osservazione è alla base del nostro modello migliorativo, nella quale ci chiediamo se, una gestione diversa del lavoro interna al sistema, possa apportare benefici tangibili all'esterno del sistema. Tale miglioramento non si affida all'introduzione di nuovi mezzi, in quanto il costo di un singolo mezzo, o dell'aumento dei serventi (il quale include anche l'acquisto di strumenti aggiuntivi per le officine, o incremento delle dimensioni della zona di scarico) sono possibilità non realizzabili per il polo impiantistico.

12 Modello migliorativo

12.1 Descrizione del modello migliorativo

Come visto finora, nel modello reale proposto nel caso stazionario (caso di interesse poiché è come lavora l'azienda nella realtà) i QoS ripetutamente violati. L'idea dietro il modello migliorativo è vedere come si comporta il sistema, rispetto ai QoS imposti, nel caso in cui i veicoli nel sistema non vengano gestiti più per ordine di arrivo.

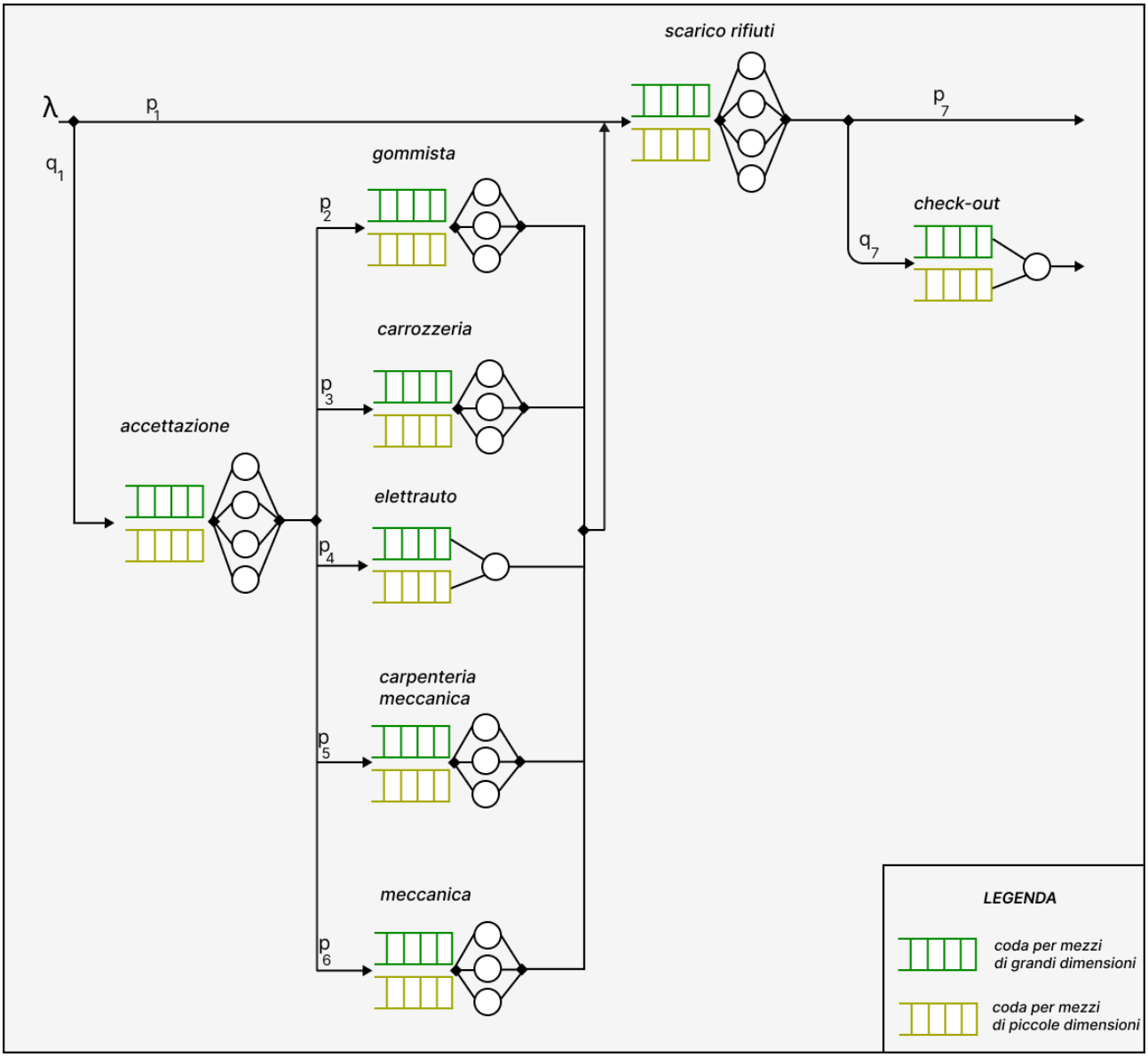
Obiettivi del modello migliorativo

Nel modello migliorativo abbiamo fissato lo stesso obiettivo del modello di base, ossia minimizzare il numero di mezzi nel sistema per ogni categoria, in modo da rispettare per complementarietà i QoS richiesti (meno mezzi si hanno nel sistema, più mezzi sono effettivamente in uso per raccogliere rifiuti).

- ❒ Il numero di mezzi CSL2 (veicoli *"piccoli"*) al di fuori del sistema deve superare il 27%.
- ❒ Il numero di mezzi CSL3 (veicoli *"grandi"*) al di fuori del sistema deve superare il 64%.

13 Modello concettuale

Il system diagram del sistema evoluto è illustrato nella seguente figura:



Politiche di scheduling dei job all'interno delle code

Questo modello prevede gli stessi utenti del modello presentato precedentemente, condividendo gli stessi stati ed eventi.

Il miglioramento risiede nel fatto che in questa variante cambia la *gestione delle code*, introducendo di fatto l'uso di code di priorità. Tale priorità è **dinamica** ed **adattiva**, a seconda dello stato del sistema. In particolare, si hanno due classi (una per veicolo) e la priorità viene assegnata alla classe che è più vicina a raggiungere una certa soglia fissata (che è stata impostata al 60% del numero massimo dei veicoli dello stesso tipo nel sistema dato dai QoS). Questa percentuale è stata ricavata dalle simulazioni prodotte nell'ambiente migliorativo.

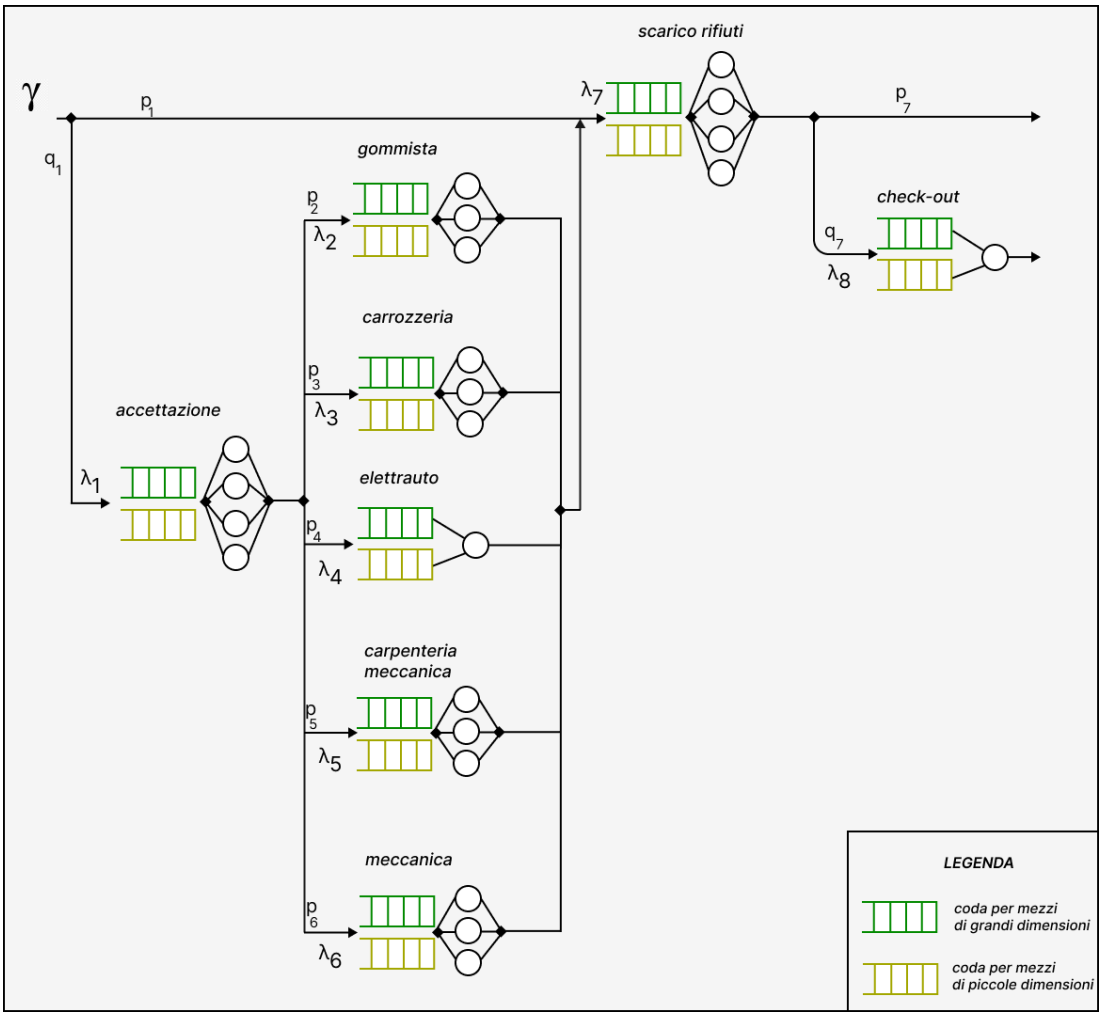
In questo modo si cerca di rispettare entrambi gli obiettivi senza favorire una delle due classi a discapito dell'altra.

14 Modello delle specifiche

Per quanto riguarda il modello delle specifiche, sono stati utilizzati gli stessi dati ricevuti dall'AMA in precedenza, in quanto nel modello migliorativo il numero di mezzi presenti nel sistema e i tempi necessari per i servizi nei vari centri non vengono modificati.

Per facilitare la trattazione, presentiamo nuovamente la rappresentazione del sistema, supportata questa volta dalle probabilità di entrare in un centro, fornendo quindi le **equazioni di traffico** la **matrice di routing**.

Equazioni di traffico



$$\lambda_1 = (1 - p_1) \cdot \lambda$$
$$\lambda_2 = p_2 \cdot \lambda_1$$
$$\lambda_3 = p_3 \cdot \lambda_1$$
$$\lambda_4 = p_4 \cdot \lambda_1$$
$$\lambda_5 = p_5 \cdot \lambda_1$$
$$\lambda_6 = p_6 \cdot \lambda_1$$
$$\lambda_7 = \lambda$$
$$\lambda_8 = \lambda_7 \cdot (1 - p_7)$$

Di seguito, viene invece illustrata la matrice di routing.

Matrice di routing

	Esterno	Accettazione	Gommista	Carrozzeria	Elettrauti	Carpenteria	Meccanica	Scarico	Checkout
Esterno	0	$1 - p_1$	0	0	0	0	0	p_1	0
Accettazione	0	0	p_2	p_3	p_4	p_5	p_6	0	0
Gommista	0	0	0	0	0	0	0	1	0
Carrozzeria	0	0	0	0	0	0	0	1	0
Elettrauti	0	0	0	0	0	0	0	1	0
Carpenteria	0	0	0	0	0	0	0	1	0
Meccanica	0	0	0	0	0	0	0	1	0
Scarico	p_7	0	0	0	0	0	0	0	$1 - p_7$
Checkout	1	0	0	0	0	0	0	0	0

15 Modello computazionale

Il modello computazionale è rimasto invariato rispetto al caso precedente. Le uniche variazioni si hanno nelle linee di codice riguardanti la gestione della coda. Queste in particolare si hanno nei controller.

Controller di Sistema

Per quanto riguarda il modello migliorativo, viene introdotto un controllo dinamico che si adatta in base al numero e al tipo di veicoli presenti nel sistema. Inizialmente, è stato implementato un primo controllo per incrementare un contatore, il quale verifica che siano rispettati i Quality of Service (QoS), garantendo così che i limiti massimi imposti sui veicoli non vengano superati.

Successivamente, sono stati introdotti controlli aggiuntivi per gestire la variazione della priorità dinamica. Se viene superata la soglia predefinita, la priorità viene adeguatamente modificata. In caso contrario, la classe più prossima alla soglia diventa prioritaria. Questo approccio consente di adattare la priorità in tempo reale in base alle condizioni del sistema, garantendo una gestione efficiente delle risorse.

```
if(eventHandler.getNumberV1()>MAX_VEICOLI1 || eventHandler.getNumberV2()>MAX_VEICOLI2){
            eventHandler.incrementSuperatoMax();
            if (eventHandler.getNumberV1()>MAX_VEICOLI1)
            eventHandler.incrementSuperatoMaxVeicoli1();
            else eventHandler.incrementSuperatoMaxVeicoli2();
        }
        if(eventHandler.getNumberV1()>=(MAX_VEICOLI1*BOUNDV1)){
            eventHandler.setPriorityClassV1();
            System.out.println("priorità veicoli 1");
        }
        if(eventHandler.getNumberV2()>=(MAX_VEICOLI2*BOUNDV2)){
            eventHandler.setPriorityClassV2();
            System.out.println("priorità veicoli 2");
        }

        if(eventHandler.getNumberV1()>MAX_VEICOLI1*BOUNDV1 &&
        eventHandler.getNumberV2()>MAX_VEICOLI2*BOUNDV2){
            if ( (eventHandler.getNumberV1() - MAX_VEICOLI1*BOUNDV1) >=
            (eventHandler.getNumberV2() - MAX_VEICOLI2*BOUNDV2)){
                eventHandler.setPriorityClassV1();
            } else eventHandler.setPriorityClassV2();
        }
```

Per quanto riguarda i controller dei vari nodi nel sistema, è stata modificata la gestione nella scelta del prossimo job che deve essere servito. In particolare, nella classe `EventHandler.java`, è stato aggiunto il seguente metodo per la selezione del prossimo evento da gestire in base al tipo di veicolo ad esso associato.

```
public int getNextEventFromQueue(List<EventListEntry> queue){
    for(int i=0;i<queue.size();i++){
        if(queue.get(i).getVehicleType()==this.priorityClass){
            return i;
        }
    }
    return 0;
}
```

Dopo aver ottenuto l'indice nella coda del prossimo evento da gestire dal metodo precedente, questo viene gestito dall'opportuno controller.

Per separare l'esecuzione della simulazione del modello base rispetto a quella del modello avanzato, si è impostato uno switch nella classe `ControllerSistema.java` che invoca un metodo diverso a seconda del tipo di simulazione desiderata.

16 Fase di verifica

Poiché il modello migliorativo fa uso di tempi di servizio basati **Gaussiane Troncate**, non è possibile metterci in condizioni di effettuare una verifica con tale distribuzione. Tale verifica è realizzabile sfruttando **Distribuzioni esponenziali** per i tempi di servizio ma, poiché questo cambiamento ci riporterebbe nelle condizioni Paragrafo 7 (**Fase di verifica nel modello base**), concludiamo che anche in questo caso la fase di verifica confermi la coerenza tra risultati analitici e dati dalla simulazione. Ciò è ulteriormente rafforzato dai seguenti aspetti:

- Dalla teoria, emerge che, in condizioni di servizi esponenziali (quindi quelli che abbiamo deciso di utilizzare in questa fase), l'introduzione di classi di priorità, di cui non sappiamo la taglia del job (quindi contesto *Abstract Priority*) e nella quale non è possibile sorpassare un job già in servizio (*Non preemptive*), apporta miglioramenti ai tempi solo nel caso **locale**, ovvero solo per i tempi della classe prioritaria, ma **generalmente**, non miglioriamo i tempi del centro, il quale mostra comportamenti analoghi ad uno scheduling astratto *KP*.
- Eseguendo la simulazione nel contesto migliorativo, otteniamo, per ciascun centro, le stesse statistiche su tempi in coda, utilizzazione e tempi di servizio.

Di conseguenza, oltre alla corrispondenza tra valori prodotti dalla simulazione, e valori analitici, si può osservare anche il rispetto dei seguenti criteri di consistenza, per i motivi appena citati:

- $E[T_s] = E[T_q] + E[Si]$
- $E[N_s] = E[N_q] + m \cdot \rho$
- $0 < \rho \leq 1$

17 Validazione

Anche in questo modello, come nel precedente gli arrivi sono stati ottenuti dai report ricevuti dall'AMA.

Poiché è stato modificato solo l'ordine con cui i mezzi vengono scelti per essere serviti non cambia il numero di eventi che arrivano al sistema nello stesso arco di tempo durante la simulazione, motivo per cui si mantiene la consistenza tra i dati reali e la simulazione.

18 Design degli esperimenti

18.1 Analisi collo bottiglia

Per i motivi sopra citati, le conclusioni ottenute nel modello base si ripropongono anche nel modello migliorativo.

18.2 Simulazione ad orizzonte finito

La simulazione è stata effettuata sul sistema con la sua configurazione standard di serventi, riproponendo la tecnica della **Replicazione**. I risultati prodotti, per 96 replicazioni, sono:

Categoria	$E[Tq]$	$E[Nq]$	ρ	$E[Ts]$	$E[Ns]$
Accettazione	2.938 ± 0.561	0.005 ± 0.001	0.252 ± 0.007	601.984 ± 12.675	1.012 ± 0.027
Carpenteria	442.171 ± 54.984	0.119 ± 0.015	0.359 ± 0.009	5822.98 ± 144.57	1.554 ± 0.043
Carrozzeria	900.81 ± 119.23	0.211 ± 0.028	0.316 ± 0.008	6294.57 ± 190.65	1.476 ± 0.048
Checkout	815.466 ± 39.428	0.418 ± 0.021	0.481 ± 0.012	1756.34 ± 52.715	0.9 ± 0.030
Scarico	902.26 ± 165.43	4.265 ± 0.836	0.790 ± 0.019	1586.28 ± 167.91	7.425 ± 0.866
Elettrauto	5393.59 ± 571.06	0.781 ± 0.088	0.193 ± 0.005	10801.1 ± 621.89	1.554 ± 0.098
Gommista	757.510 ± 89.462	0.384 ± 0.048	0.451 ± 0.011	4365.1 ± 131.76	2.189 ± 0.076
Meccanica	2681.77 ± 327.47	1.161 ± 0.156	0.565 ± 0.015	8077.0 ± 371.144	3.423 ± 0.199

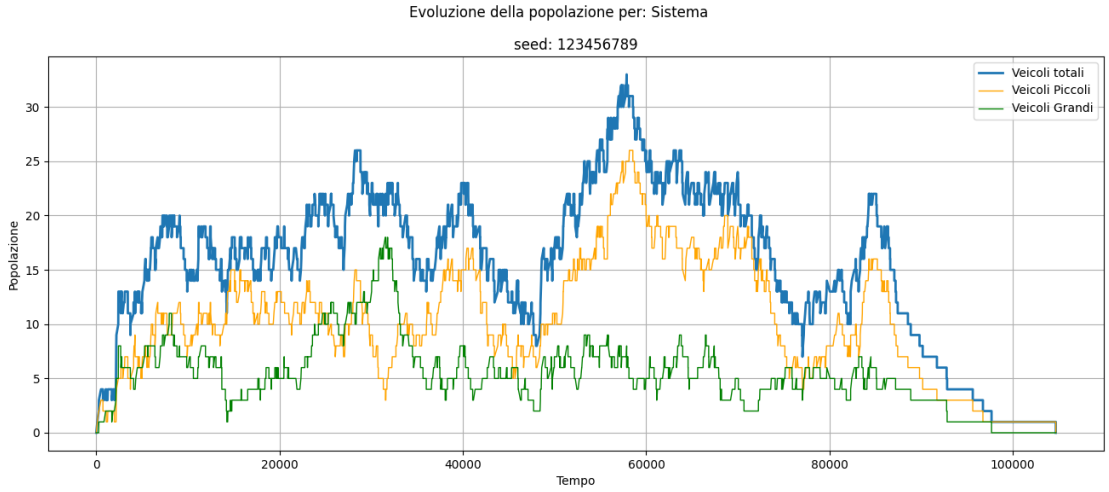
Il dato più interessante è però quello legato al nostro obiettivo, ovvero la *riduzione dei contesti in cui i QoS non vengono rispettati.* Nelle 96 replicazioni, l'inadempimento dei requisiti di qualità non viene mai riscontrato, impostando i seguenti coefficienti:

- La tipologia di mezzi di tipo CSL2 (veicoli *"piccoli"*) diventa prioritaria se il numero di tali veicoli nel sistema è pari a "$\text{MaxVeicoli}_1 \cdot \text{Bound}_1$", dove:
  - MaxVeicoli_1 indica il numero di CSL2 che violerebbero il QoS.
  - Bound_1 è un coefficiente di *prevenzione del Qos* pari a 0.6.
- La tipologia di mezzi di tipo CSL3 (veicoli *"grandi"*) diventa prioritaria se il numero di tali veicoli nel sistema è pari a "$\text{MaxVeicoli}_2 \cdot \text{Bound}_2$":
  - MaxVeicoli_2 indica il numero di CSL3 che violerebbero il QoS.
  - Bound_2 è un coefficiente di *prevenzione del QoS* pari a 0.5.

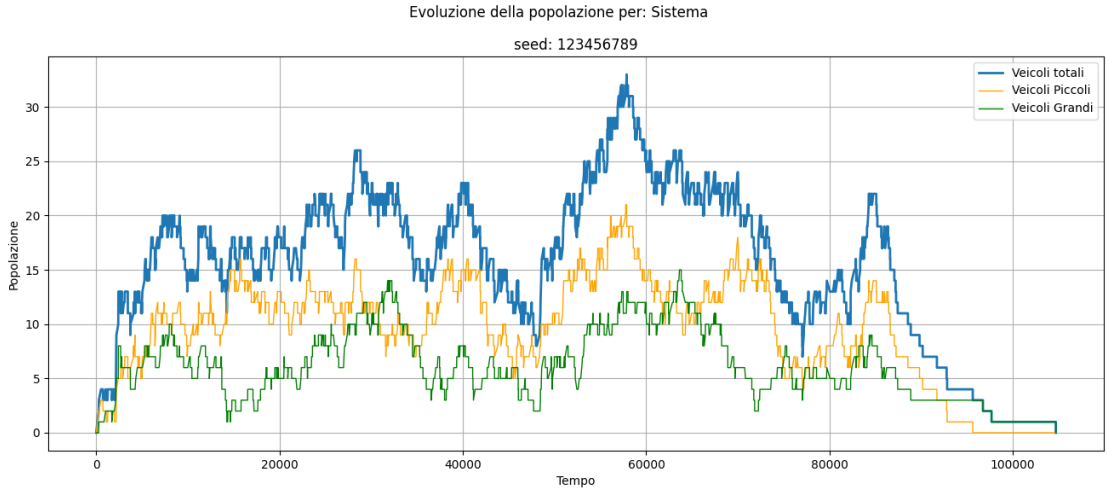
Tuttavia, come precedentemente discusso nel modello di base, la simulazione nel transiente non fornisce una valutazione estremamente accurata della validità dei risultati ottenuti. Ciò è attribuibile al fatto che l'impianto AMA opera in condizioni diverse, che non contemplano la partenza da un sistema vuoto né un intervallo temporale durante il quale non sono più accettati mezzi nel sistema, al fine di completare i servizi per i mezzi interni.

Ciò viene mostrato dal risultato della nostra simulazione:

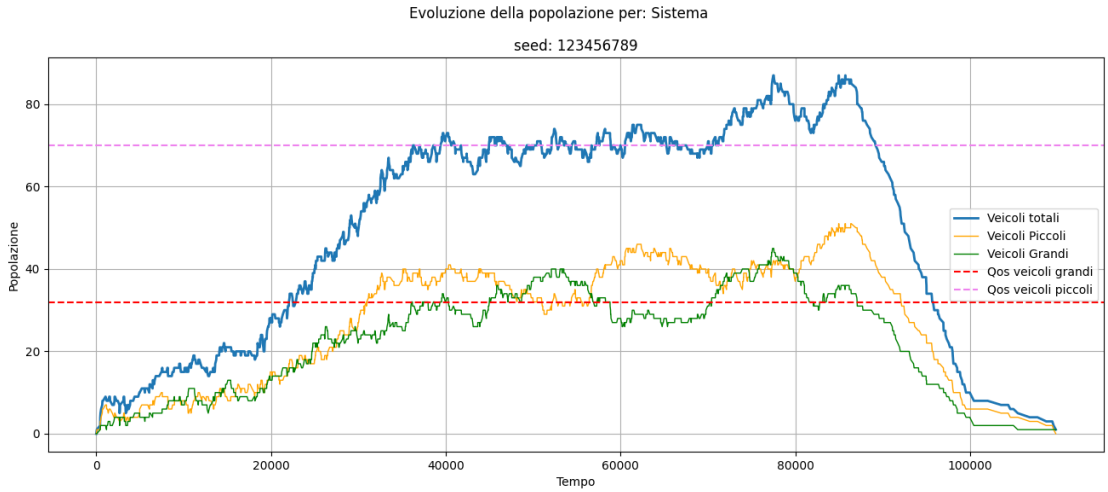
```
Numero di volte in cui è stato superato il limite di veicoli nel sistema: 0
Per i veicoli di tipo 1: 0 e per i veicoli di tipo 2:  0
```



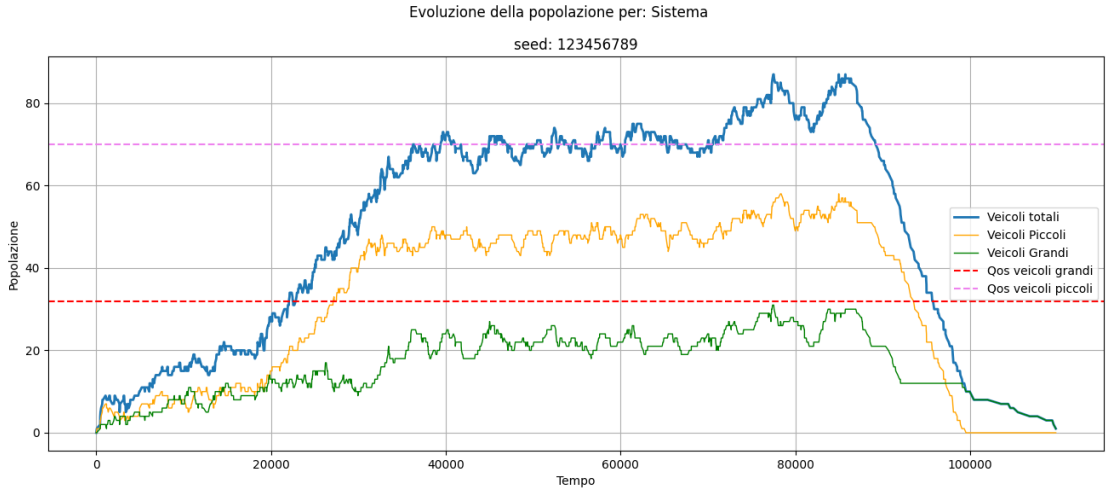
E' interessante notare come tale risultato sia unicamente dovuto ad una diversa scelta dei prossimi eventi in coda nei vari centri. Nel modello base, a parità di condizioni, si otteneva:



Nel caso FIFO, riprendendo, la run con indice di replicazione 92 RUN che ha violato più volte il requisito sui mezzi di tipo 2, ciò che si otteneva era:



mentre, con una priorità dinamica, a parità di popolazione, si ottiene:



la quale riduce queste violazioni.

18.3 Simulazione ad orizzonte infinito

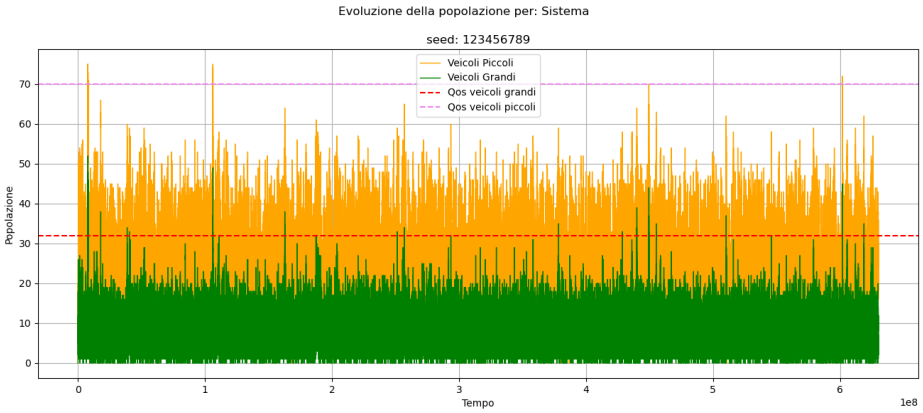
La simulazione ad orizzonte infinito è stata condotta sul sistema nella sua configurazione standard di serventi. Durante questa fase, è stata nuovamente impiegata la tecnica dei **Batch Means**, in quanto si cerca di minimizzare l'influenza dello stato iniziale. Per raggiungere questo obiettivo, è stata eseguita una simulazione estremamente prolungata (con tempo simulato infinito, `Double.MAX_VALUE`), successivamente suddivisa in batch. Lo stato iniziale di ciascun batch deriva dallo stato finale del batch precedente.

Per garantire una comparabilità con il modello di base, vengono nuovamente utilizzati i valori ottimali di B e K pari a 1080 e 96, rispettivamente. Questa selezione è stata effettuata utilizzando la classe `Acs.java`, che permette di calcolare l'autocorrelazione di una serie di dati.

Una volta fissati questi valori abbiamo ottenuto dalla simulazione i seguenti risultati:

Categoria	E[Tq]	E[Nq]	ρ	E[Ts]	E[Ns]
Accettazione	2.662 ± 0.254	0.004 ± 0.000	0.250 ± 0.005	602.871 ± 12.550	1.004 ± 0.022
Carpenteria	843.613 ± 34.624	0.279 ± 0.012	0.593 ± 0.013	6242.42 ± 133.53	2.057 ± 0.047
Carrozzeria	110.815 ± 6.625	0.018 ± 0.001	0.297 ± 0.006	5510.71 ± 114.97	0.909 ± 0.020
Elettrauto	22841.7 ± 1780.2	3.789 ± 0.313	0.890 ± 0.019	28246.8 ± 1815.0	4.679 ± 0.322
Gommista	604.079 ± 26.540	0.302 ± 0.014	0.599 ± 0.013	4207.65 ± 91.02	2.101 ± 0.049
Meccanica	7972.74 ± 987.78	4.032 ± 0.530	0.899 ± 0.020	13370.9 ± 1013.3	6.728 ± 0.553
Scarico	1568.72 ± 337.61	8.867 ± 1.982	0.947 ± 0.021	2251.73 ± 339.38	12.657 ± 2.006
Checkout	451.848 ± 14.067	0.250 ± 0.009	0.519 ± 0.011	1393.64 ± 30.89	0.769 ± 0.019

Anche in questo caso i risultati sono coerenti con le nostre attese per gli stessi motivi illustrati in precedenza. Concentrandoci sulla violazione dei QoS, impostando come *Soglia di prevenzione* i valori $\text{BOUND}_1 = 0.6$ e $\text{BOUND}_2 = 0.5$ il sistema produce il seguente risultato:




```
Numero di volte in cui è stato superato il limite di veicoli nel sistema: 10184
Per i veicoli di tipo 1: 798 e per i veicoli di tipo 2:  9386
```

Ciò si traduce in una riduzione di circa l'83% delle situazioni in cui il QoS non viene generalmente rispettato. Dalle simulazioni condotte, la coppia di *BOUND* proposta è quella che fornisce le prestazioni migliori, in quanto aumentare eccessivamente la differenza nella coppia comporta un dislivello eccessivo tra le violazioni dei QoS.

19 Conclusioni e confronto

Attraverso un'ottimizzazione del modello, concentrata sulla gestione delle code nei nodi del sistema, siamo riusciti a ridurre significativamente le violazioni del limite, passando da 59.285 a soli 10.184 casi, ottenendo un miglioramento del 83%. È importante notare che, sebbene vi sia un lieve peggioramento per i veicoli di tipo 1, questo è giustificato dal miglioramento netto dei veicoli di tipo 2, oltre che del Quality of Service generale. Questa ottimizzazione ha contribuito in modo significativo a migliorare l'efficienza *complessiva del sistema*, enfatizzando il bilanciamento tra i diversi tipi di veicoli e promuovendo una gestione più equa delle risorse. Nella realtà, la conclusione trae ampia giustificazione. In situazioni in cui mancano mezzi di tipo 2 (*veicoli grandi*), il polo impiantistico potrebbe sopperire a questa carenza fornendo, per lo stesso tipo di ricognizione, mezzi di tipo 1. Anche se tali veicoli potrebbero eseguire parzialmente il lavoro, la loro capacità minore per i rifiuti si tradurrebbe in più servizi di prelievo dei rifiuti per pulire una strada o una zona.

D'altro canto, la soluzione non può essere semplicemente orientata verso il favorire esclusivamente veicoli di tipo 2 (*veicoli grandi*). A causa delle loro dimensioni, essi non possono completare tutti i servizi dei veicoli di tipo 1 (*veicoli piccoli*) in strade poco viabili. Pertanto, la strategia ottimale sembra essere una combinazione bilanciata di entrambi i tipi di veicoli, garantendo così una copertura completa e efficiente delle esigenze di pulizia delle strade.