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.

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.

1 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.

2 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.

3 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.

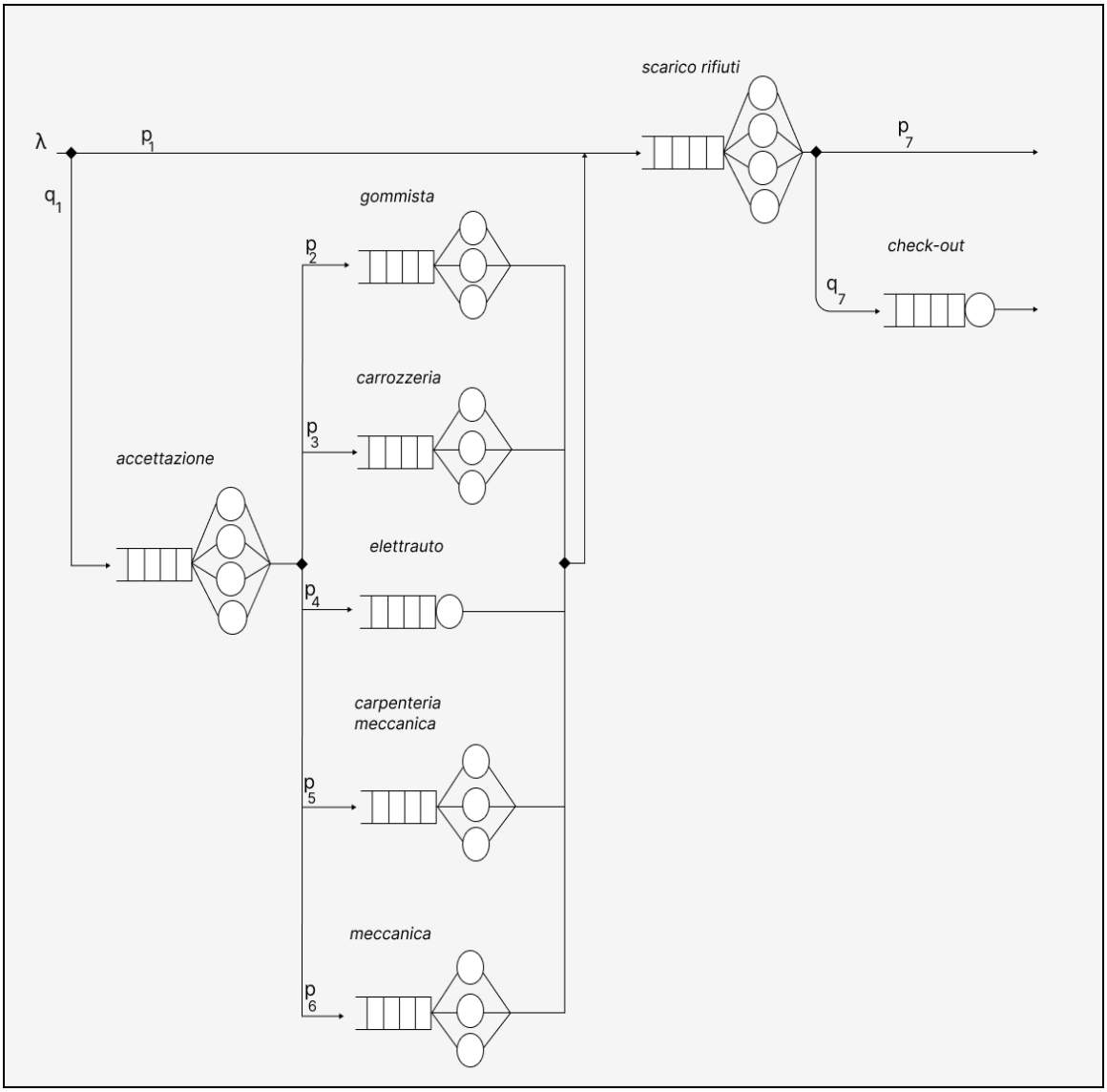
Il sistema è suddivisibile nelle seguenti macroaree:

- **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. Tale area si compone di 5 officine specializzate: *Gommista, Carrozzeria, Elettrauto, Carpenteria, Meccanica.*
- **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.
- 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.

4 Obiettivo e profitto del sistema

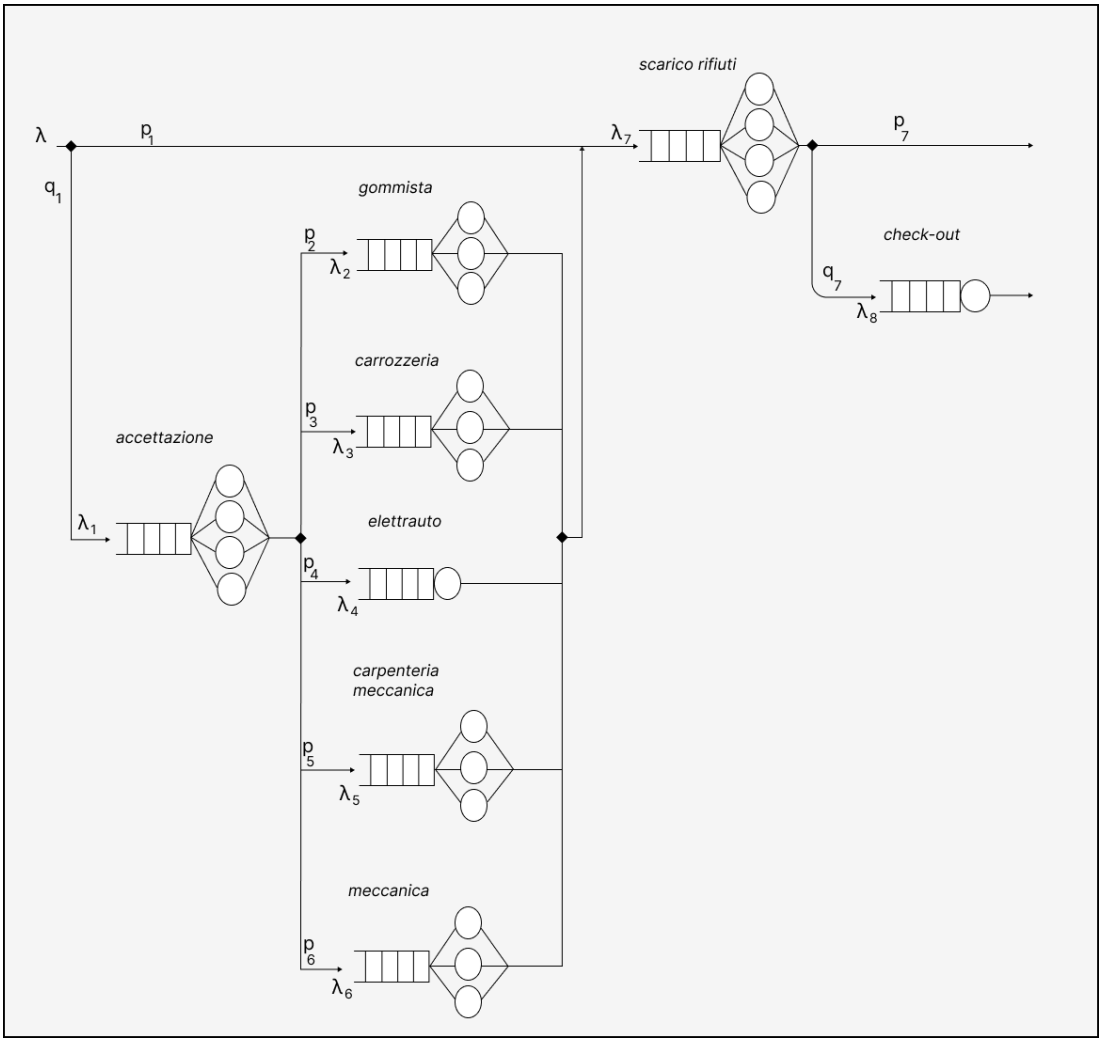
In quanto azienda pubblica, è inappropriato utilizzare parametri finanziari come indicatori di performance. Tuttavia, è essenziale considerare l'impatto culturale del decoro urbano sul turismo e sulla percezione del territorio. L'analisi si concentra sulla qualità del servizio offerto, correlata al numero 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.

5 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



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

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

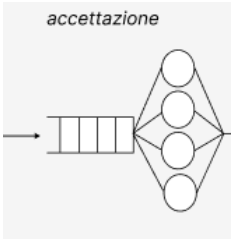
Modellazione dei centri

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**.

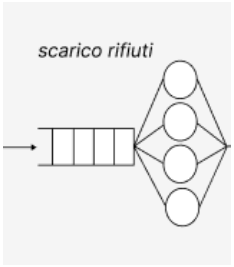
- ❒ La distribuzione dei tempi di interrarivo è **Esponenziale** di parametro $\frac{1}{\lambda_1}$.
- ❒ La distribuzione dei servizi è rappresentata da una **Gaussiana troncata**, avente come tempo di servizio medio, per il singolo servente, $E[S_i] = 600$ secondi (pari a 10 minuti). Questo tempo di 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.

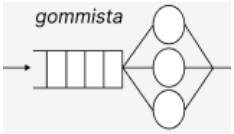


Autofficina

Gommista

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

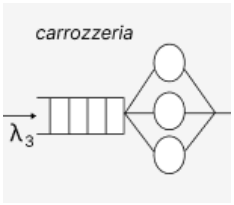
- ❒ La distribuzione dei tempi di interrarivo è **Esponenziale** di parametro $\frac{1}{\lambda_2}$.
- ❒ La distribuzione dei servizi è rappresentata da una **Gaussiana troncata**, avente come tempo di servizio medio, per il singolo servente, $E[S_i] = 3600$ secondi (pari a un'ora). Questo tempo di servizio 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**.

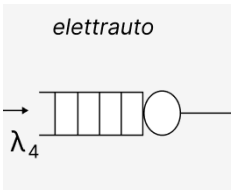
- ❒ La distribuzione dei tempi di interrarivo è **Esponenziale** di parametro $\frac{1}{\lambda_3}$.
- ❒ La distribuzione dei servizi è rappresentata da una **Gaussiana troncata**, avente come tempo di servizio medio, per il singolo servente, $E[S_i] = 5400$ secondi (pari a due ora). Questo tempo di 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**.

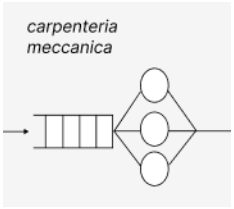
- ❒ La distribuzione dei tempi di interrarivo è **Esponenziale** di parametro $\frac{1}{\lambda_4}$.
- ❒ La distribuzione dei servizi è rappresentata da una **Gaussiana troncata**, avente come tempo di servizio medio, per il singolo servente, $E[S_i] = 5400$ secondi (pari a due ora). Questo tempo di 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**.

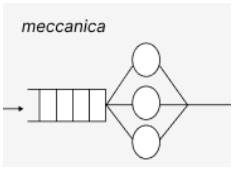
- ❒ La distribuzione dei tempi di interrarivo è **Esponenziale** di parametro $\frac{1}{\lambda_5}$.
- ❒ La distribuzione dei servizi è rappresentata da una **Gaussiana troncata**, avente come tempo di servizio medio, per il singolo servente, $E[S_i] = 5400$ secondi (pari a due ora). Questo tempo di 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.



6 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 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

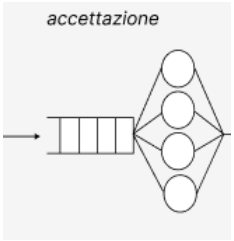
Per la gestione dei tempi di servizio, la scelta migliore è ricaduta su una *distribuzione Gaussiana Troncata*. Come espresso dal libro di testo, per la produzione di un troncamento non basta definire dei limiti superiori ed inferiori, e *tagliare* la distribuzione agli estremi, in quanto questo produrrebbe 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);
}
```

7 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 faranno uno di tempi di *servizio esponenziali*, in quanto, per un confronto di tipo analitico, non si avrebbe un confronto valutabile 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



$\lambda_1 = 0.0055 \cdot 0.3 \text{ job/sec}$

$E(S_i) = 600 \text{ sec}$

$N = 4$

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

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

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

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

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

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

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

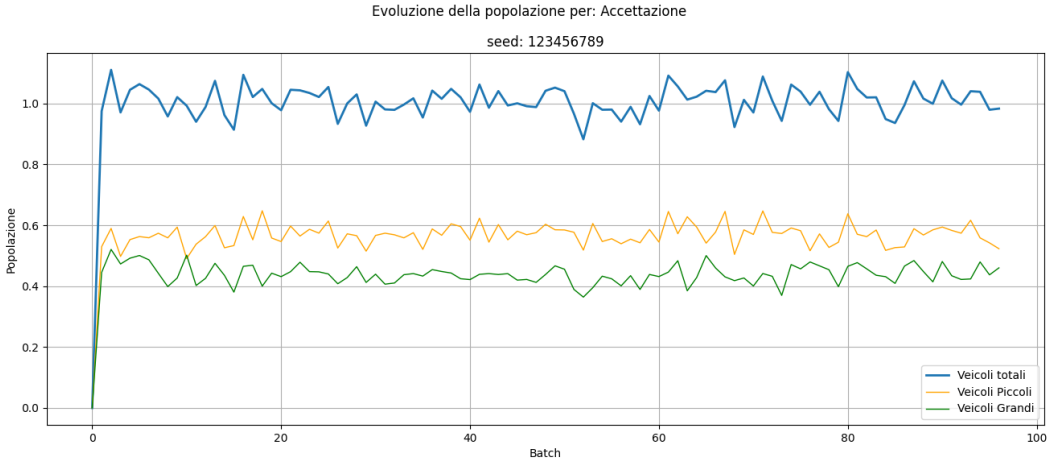
$E(T_s) = E(T_q) + E(S_i) = 603,9341976 \text{ sec}$

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

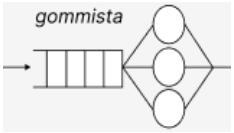
I risultati prodotti dalla simulazione sono:

```
Accettazione
Statistiche per E[Tq] Critical endpoints 3.8867430424900333 +/- 0.5243468929305334
statistiche per E[Nq] Critical endpoints 0.006497032812644441 +/- 8.845282301381028E-4
statistiche per rho Critical endpoints 0.2476836837926347 +/- 0.005678168383475196
statistiche per E[Ts] Critical endpoints 598.7581945237913 +/- 12.935732176940641
statistiche per E[Ns] Critical endpoints 0.9972317679831831 +/- 0.022953849081016605
```

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



Gommista



$\lambda_2 = 0.00165 \cdot 0.3 \ \text{job/sec}$

$E(S_i) = 3600 \ \text{sec}$

$N = 3$

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

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

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

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

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

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

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

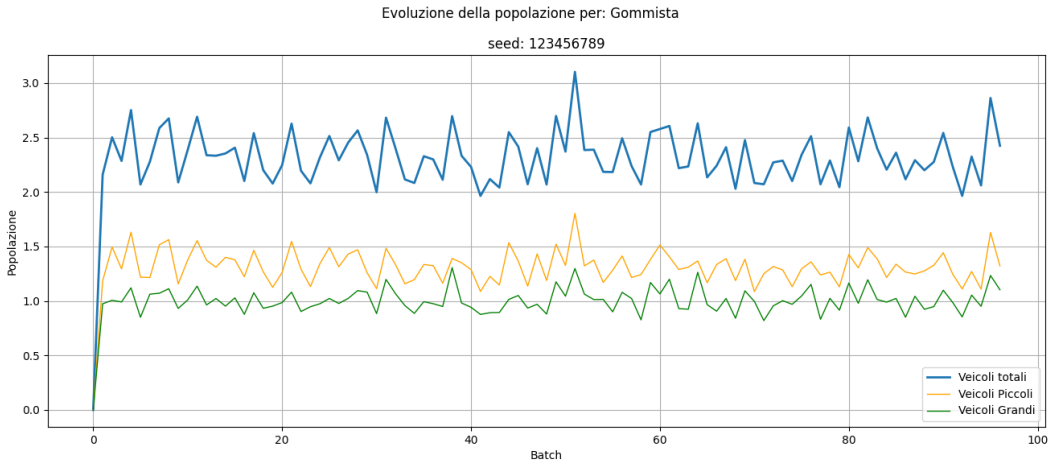
$E(T_s) = E(T_q) + E(S_i) = 4625,880617 \ \text{sec}$

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

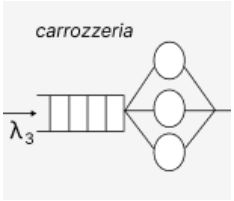
I risultati prodotti dalla simulazione sono:

```
Gommista
Statistiche per E[Tq] Critical endpoints 1067.5296746776733 +/- 65.47302311757407
statistiche per E[Nq] Critical endpoints 0.5343002677110158 +/- 0.03408238653895891
statistiche per rho Critical endpoints 0.6012604138724726 +/- 0.013603976438465434
statistiche per E[Ts] Critical endpoints 4682.030449267476 +/- 123.25868713405984
statistiche per E[Ns] Critical endpoints 2.3380815093284335 +/- 0.06577325100699413
```

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



Carrozzeria



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

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

$N = 3$

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

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

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

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

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

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

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

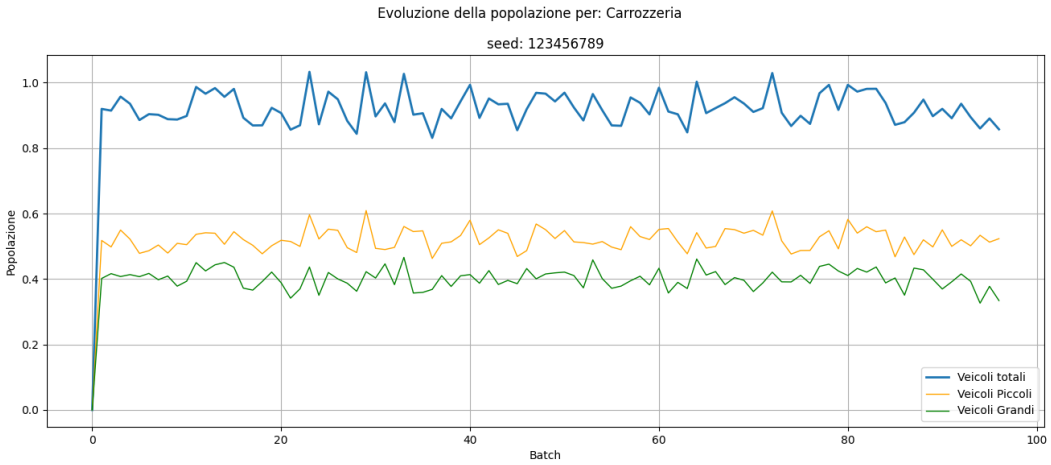
$E(T_s) = E(T_q) + E(S_i) = 5574,855531 \ \text{sec}$

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

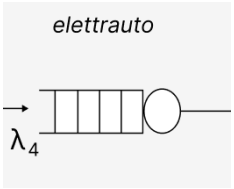
I risultati prodotti dalla simulazione sono:

```
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



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

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

$N = 1$

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

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

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

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

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

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

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

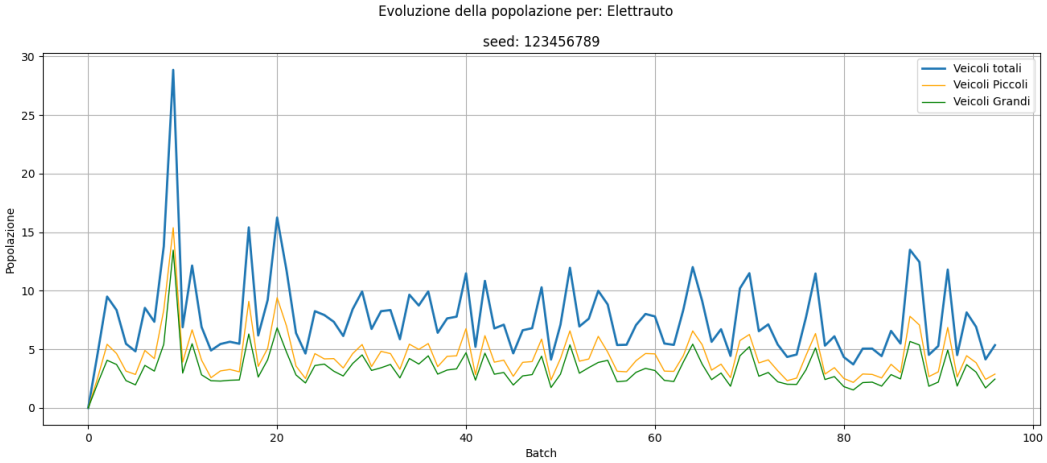
$E(T_s) = E(T_q) + E(S_i) = 45633,81639$ sec

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

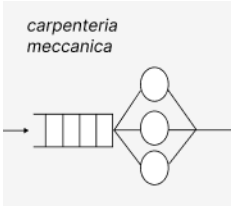
I risultati prodotti dalla simulazione sono:

```
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

$N = 3$

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

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

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

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

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

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

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

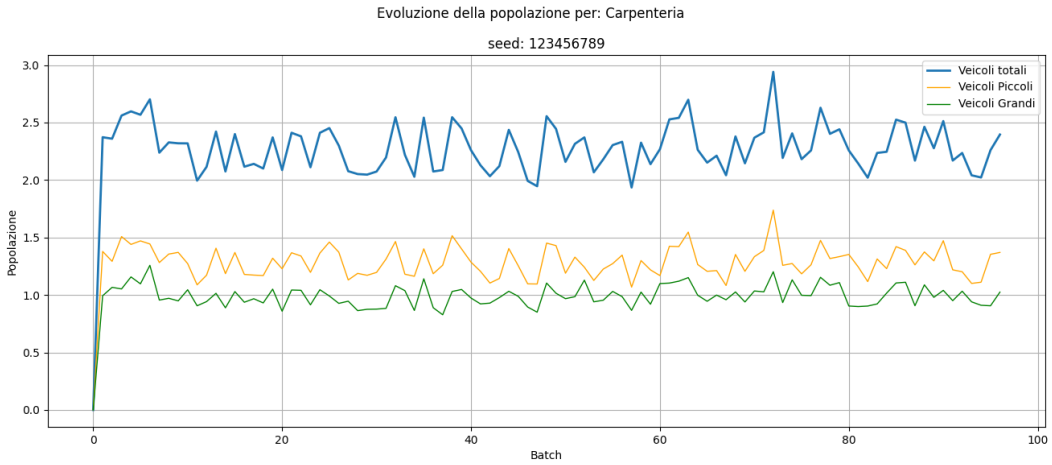
$E(T_s) = E(T_q) + E(S_i) = 6938,820926 \text{ sec}$

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

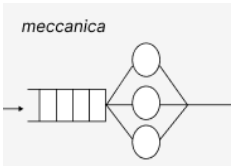
I risultati prodotti dalla simulazione sono:

```
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



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

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

$N = 3$

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

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

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

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

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

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

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

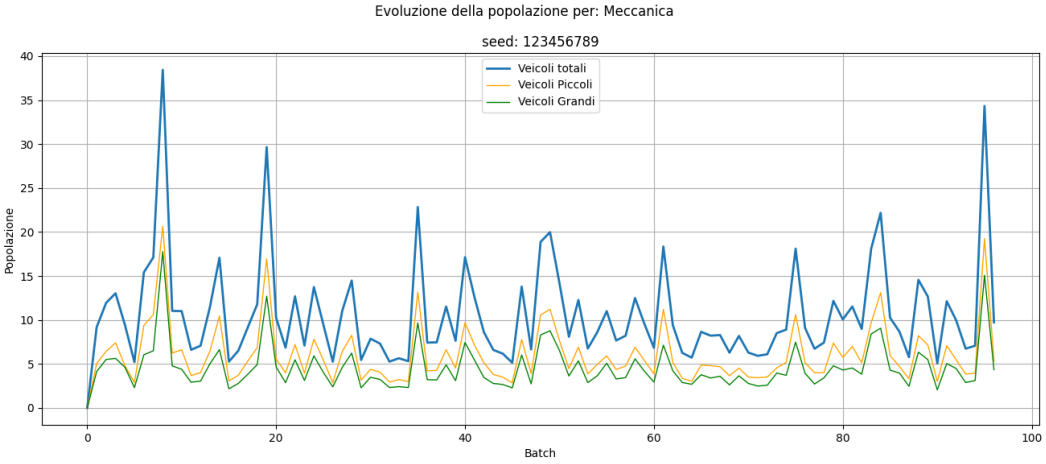
$E(T_s) = E(T_q) + E(S_i) = 18630,99729 \text{ sec}$

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

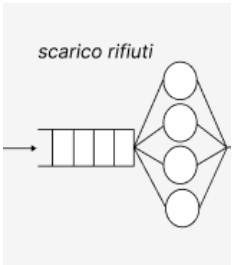
I risultati prodotti dalla simulazione sono:

```
Meccanica
Statistiche per E[Tq] Critical endpoints 15876.3882447926 +/- 2257.296648198602
statistiche per E[Nq] Critical endpoints 8.023194831677312 +/- 1.185474535675924
statistiche per rho Critical endpoints 0.8985906428860488 +/- 0.0204304780384617
statistiche per E[Ts] Critical endpoints 21272.984326560934 +/- 2294.288472448443
statistiche per E[Ns] Critical endpoints 10.718966760335455 +/- 1.2129865984680073
```

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

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

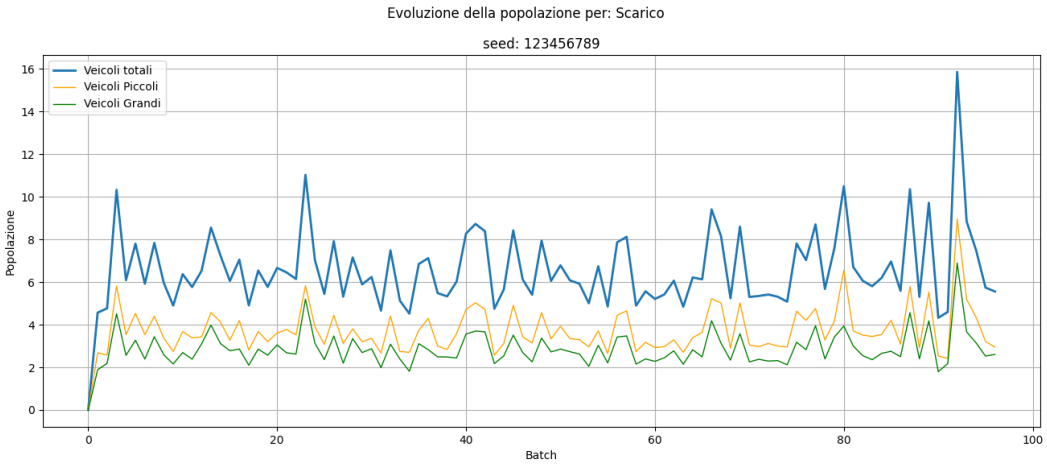
$E(T_s) = E(T_q) + E(S_i) = 1150,422475$

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

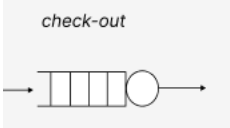
I risultati prodotti dalla simulazione sono:

```
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



$\lambda_3 = 0.0055 \cdot 0.1 \ \text{job/sec}$

$E(S_i) = 1200 \ \text{sec}$

$N = 1$

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

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

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

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

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

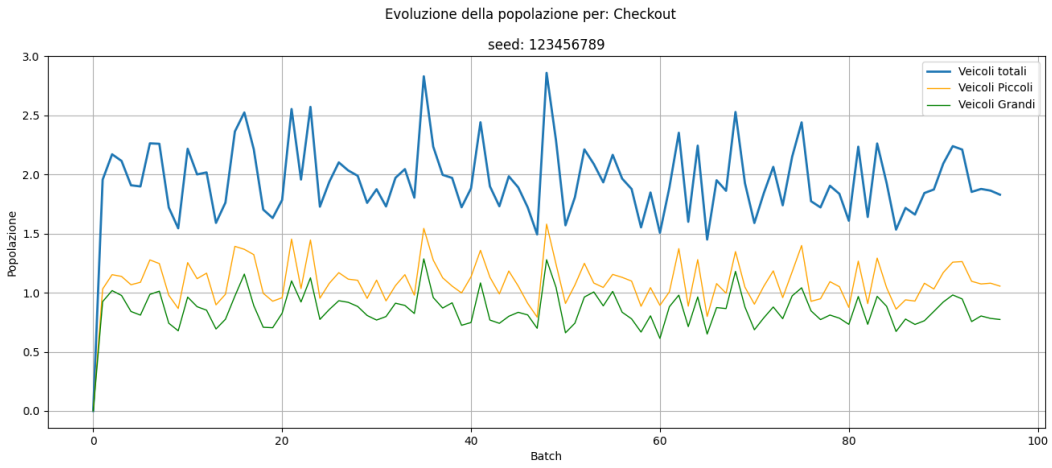
$E(T_s) = E(T_q) + E(S_i) = 3529,41 \ \text{sec}$

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

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:

- $E[T_s] = E[T_q] + E[Si]$
- $E[N_s] = E[N_q] + N \cdot \lambda$
- $0 \leq \rho \geq 1$

8 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**: essendo l'ultimo centro *obbligatorio* per tutti i mezzi, nel momento di chiusura delle porte, non avremo più ingressi esterni, ma confluiranno solo gli ingressi interni provenienti dalle officine.
- **Accettazione**: Essendo uno dei principali 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.
- **Elettrauto**: Essendo l'unica officina a servente singolo, ci aspettiamo, rispetto ad altri centri, una popolazione maggiore, oltre che tempi di servizio ampiamente dilazionati nel tempo.