Performance Modeling Of Computer Systems And Networks

Marco Lorenzini
0353515

Eugenio Di Gaetano
0349278

Università degli studi di Roma Tor Vergata

Abstract

Il presente studio analizza la gestione di un Pronto Soccorso, focalizzandosi in particolare sul centro Policlinco Tor Vergata. Il Pronto Soccorso è un'unità cruciale all'interno di un ospedale, dedicata all'accoglienza e alla prima assistenza di pazienti in condizioni urgenti e non programmabili. Nello studio viene analizzato l'intero processo che ogni paziente che si rivolge ad un Pronto Soccorso deve attraversare, partendo dal triage iniziale fino alla dimissione o al ricovero ospedaliero.

1. Introduzione

Il Pronto Soccorso deve gestire un flusso di pazienti derivanti da arrivo diretto in struttura o da arrivo tramite ambulanza, il primo contatto del paziente con la struttura è il triage, questa fase determina la priorità del trattamento in base alla gravità delle condizioni del paziente e all'urgenza del caso e assegna ai pazienti uno dei seguenti codici:

- **Livello 1 (Rosso)**: Interruzione o compromissione di una o più funzioni vitali.
- **Livello 2 (Arancione)**: Rischio di compromissione delle funzioni vitali. Condizione con rischio evolutivo o dolore severo.
- **Livello 3 (Azzurro)**: Condizione stabile senza rischio evolutivo con sofferenza e ricaduta sullo stato generare che solitamente richiede prestazioni complesse.
- **Livello 4 (Verde)**: Condizione stabile senza rischio evolutivo che solitamente richiede prestazioni diagnostico terapeutiche semplici monospecialistiche.
- **Livello 5 (Bianco)**: Problema non urgente o di minima rilevanza clinica.

Dopo il triage, il paziente deve svolgere la prima visita medica. Durante questa visita, il medico stabilisce le condizioni del paziente e pianifica gli esami diagnostici e i trattamenti necessari. Questa fase è fondamentale per garantire un'assistenza mirata e tempestiva, adeguata alle necessità specifiche del paziente. Terminati gli esami richiesti, il medico valuta gli esiti e decide se il paziente può lasciare il Pronto Soccorso con le adeguate indicazioni per il follow-up. In caso contrario, il paziente può essere ricoverato in ospedale per ricevere cure specialistiche continue e monitoraggio delle condizioni di salute.

2. Problematiche del sistema

Dall'analisi dei dati riportati nei vari annuari statistici della Regione Lazio, emerge che i tempi di attesa per la prima visita presso il Pronto Soccorso del Policlinico Tor Vergata risultano essere significativamente elevati. In particolare, i pazienti assegnati ai codici di priorità più bassi devono affrontare attese prolungate, che possono compromettere la qualità e tempestività delle cure. Di seguito i tempi medi di attesa per i diversi livelli di gravità relativi all'anno 2022:

- **Livello 2 (Arancione)**: 57 minuti

- **Livello 3 (Azzurro)**: 164 minuti
- **Livello 4 (Verde)**: 154 minuti
- **Livello 5 (Bianco)**: 148 minuti

Questi dati evidenziano un significativo disallineamento tra la domanda di servizi e la capacità del sistema di rispondere in tempi adeguati. Tali ritardi possono non solo generare disagi per i pazienti, ma anche impattare negativamente sugli esiti clinici, in particolare per i casi con urgenza intermedia.

3. Obiettivi

Nel 2019 sono state rilasciate delle nuove linee guida a livello nazionale con l'obiettivo di migliorare l'efficacia e l'efficienza del sistema di emergenza-urgenza. Queste linee guida stabiliscono tempi massimi di attesa che i pronto soccorso devono rispettare per garantire una rapida presa in carico e un avvio tempestivo del trattamento. I tempi massimi previsti per l'accesso all'area di trattamento sono:

- **Livello 1 (Rosso)**: accesso immediato
- **Livello 2 (Arancione)**: accesso entro 15 minuti
- **Livello 3 (Azzurro)**: accesso entro 60 minuti
- **Livello 4 (Verde)**: accesso entro 120 minuti
- **Livello 5 (Bianco)**: accesso entro 240 minuti

Lo studio condotto sul sistema si propone di raggiungere come obiettivo quello di rispettare le linee guida presentate. Tale obiettivo richiede un'approfondita analisi dei processi interni e l'individuazione di strategie di miglioramento che possano essere implementate senza alterare la struttura organizzativa esistente. In particolare, l'intervento dovrà essere realizzato mantenendo invariato il numero di medici attualmente disponibili, ottimizzando dunque l'efficienza operativa attraverso una migliore gestione delle risorse e dei flussi di pazienti.

4. Modello Concettuale

In questa fase della trattazione, ci concentriamo sulla definizione astratta delle caratteristiche del sistema, ponendo particolare attenzione agli stati e agli eventi che si verificano al suo interno. Come prima cosa, definiamo gli utenti del nostro sistema come tutte le persone che si rivolgono al Pronto Soccorso per ricevere assistenza medica. Ogni persona sarà rappresentata da un certo livello di urgenza. Successivamente possiamo suddividere il sistema in diverse macroaree funzionali, ognuna delle quali svolge un ruolo cruciale nel processo di gestione del paziente. Di seguito, descriviamo le principali aree:

- **Triage**: Il triage è la prima area di interazione tra il paziente e il sistema di pronto soccorso. Qui, il personale infermeristico valuta rapidamente la gravità delle condizioni del paziente e assegna un codice di priorità (rosso, arancione, azzurro, verde, bianco) che determina l'ordine con cui il paziente verrà visitato. Questa fase è fondamentale per garantire che i casi più urgenti vengano trattati tempestivamente.
- **Area di trattamento**: Dopo il triage, i pazienti vengono indirizzati all'area di trattamento. In questa fase, il paziente viene visitato da un medico che valuta la condizione generale e stabilisce un piano d'azione. Questo piano può includere l'esecuzione di esami diagnostici e ulteriori visite specialistiche. L'area di trattamento rappresenta il punto nevralgico del pronto soccorso, dove le decisioni cliniche vengono prese in base alla condizione del paziente.
- **Aree di analisi**: Se il medico ritiene necessario approfondire la diagnosi, il paziente viene indirizzato alle aree di analisi, dove vengono effettuati esami specifici. Queste aree sono suddivise in diverse sottoaree specializzate, ognuna dedicata a un tipo specifico di esame. Le principali aree di analisi includono:
  - **Emocromo**: Esecuzione di esami del sangue per valutare parametri fondamentali come i livelli di globuli rossi, globuli bianchi e piastrine.
  - **Ecografia**: Utilizzo di ultrasuoni per visualizzare organi e strutture interne, utile per diagnosticare varie patologie.
  - **Radiografia**: Esami radiologici per visualizzare ossa, organi e altre strutture corporee, spesso usati per diagnosticare fratture o infezioni.

- **TAC**: Esame avanzato che permette di ottenere immagini dettagliate delle strutture interne del corpo, spesso utilizzato in casi complessi.
- **Elettrocardiogramma(ECG)**: Esame non invasivo che registra l'attività elettrica del cuore, essenziale per diagnosticare patologie cardiache.
- **Altro**: Questa categoria include tutte le altre visite specialistiche che non rientrano nelle categorie precedenti, come ad esempio l'endoscopia o la spirometria.

Una volta completato il triage e la prima visita medica nell'area di trattamento, il paziente viene inviato nelle varie aree di analisi in base alle necessità diagnostiche stabilite dal medico. In questa fase, il paziente può passare da un'area all'altra per completare tutti gli esami richiesti. Dopo aver effettuato gli esami necessari, il paziente torna nell'area di trattamento per una seconda valutazione medica. A questo punto, il medico può decidere se sono necessari ulteriori esami o se il paziente può essere dimesso. In caso di dimissione, al paziente vengono fornite le indicazioni per il follow-up e l'eventuale terapia da seguire. Se, invece, è necessario un monitoraggio o un trattamento più approfondito, il paziente può essere ricoverato o trasferito in altre strutture specialistiche. Il flusso del paziente all'interno del sistema può essere schematizzato in un diagramma che illustra le varie fasi, dall'ingresso al pronto soccorso fino alla dimissione o al ricovero. Questo schema rappresenta visivamente il percorso del paziente attraverso le diverse aree del sistema, evidenziando le decisioni chiave che determinano il percorso di cura.

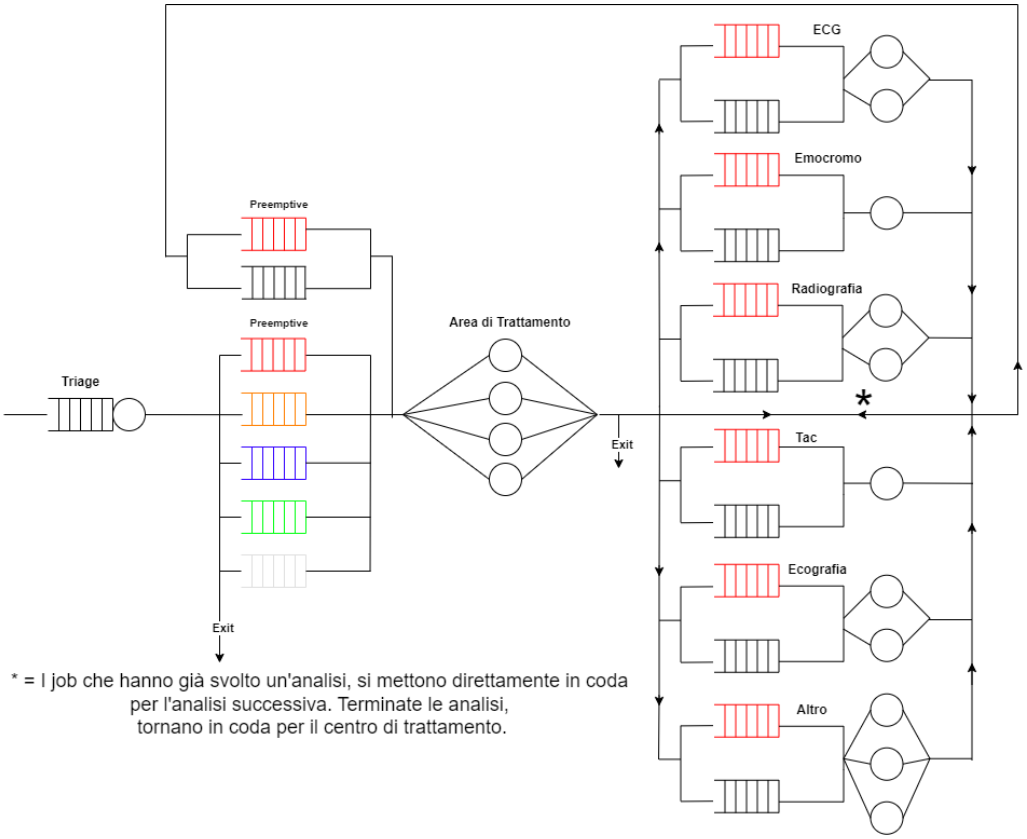

Figure 1: Schema del sistema in oggetto

Eventi

In questa trattazione identifichermo con il termine *paziente* il singolo job che entra nel sistema e che richiederà una prestazione medica.
Con il termine *sistema* intendenderemo l'insieme delle aree (triage, area di trattamento, area di analisi).
Gli eventi da noi considerati sono:

- **Per il sistema**:
  - *Arrivo di un job*: è l'ingresso di un nuovo job nel sistema
  - *Uscita di un job*: può avvenire prima o dopo la visita nel centro di trattamento.
- **Per il triage**:
  - *Arrivo di un job*: coincide con l'arrivo di un nuovo job nel sistema.
  - *Uscita di un job*: è dato dal processamento del job, che coincide con l'assegnazione del codice.
- **Per l'area di trattamento**:
  - *Arrivo di un job*: coincide con l'evento di uscita dal triage o da un'area di analisi.
  - *Uscita di un job*: è dato dall'abbandono della coda oppure dal processamento del job, che coincide con la terminazione della visita medica.
- **Per una singola area di analisi**:
  - *Arrivo di un job*: coincide con l'evento di uscita dall'area di trattamento o con l'evento di uscita da un'altra area di analisi.
  - *Uscita di un job*: è dato dal processamento del job, che coincide con lo svolgimento dell'esame.

Descrizione degli eventi

1. Un paziente che entra nel sistema inizia con il triage, durante il quale gli viene assegnato un codice di priorità e viene indirizzato all'area di trattamento appropriata.
2. In base al codice ricevuto, il paziente dovrà attendere in una coda FIFO. Il sistema prevede sette code: le prime due sono preemptive e riservate ai codici rossi (la prima per i pazienti in condizioni critiche in attesa della prima visita e la seconda per i pazienti che escono dalle aree di analisi), la terza è riservata ai pazienti, con codice diverso dal rosso di rientro dalle aree di analisi. Le ultime quattro sono suddivise in base ai codici, rispettivamente: arancioni, azzurri, verdi e bianchi, che attendono la loro prima visita medica dopo il triage.
2.1. Se l'attesa per la prima visita diventa troppo lunga, i pazienti possono decidere di abbandonare il sistema e rinunciare all'assistenza.
3. Dopo la visita medica, ai pazienti può essere richiesto di effettuare ulteriori accertamenti o possono essere dimessi dal sistema per continuare le cure a casa o essere ricoverati in ospedale.
4. I pazienti che devono eseguire degli esami entreranno nelle aree di analisi fino al completamento delle procedure prescritte.
5. Successivamente, rientreranno nell'area di trattamento per una nuova visita medica, ripartendo dal punto 2.

5. Modello delle specifiche

I dati utilizzati per la modellazione sono stati ricavati da report ufficiali redatti dalla Regione Lazio. A questo livello, il sistema è rappresentato come una collezione di variabili di stato (di tipo matematico) e da equazioni che descrivono le loro interrelazioni. Di seguito, presentiamo nuovamente la rappresentazione del sistema, evidenziando i vari flussi entranti in ogni centro e fornendo le relative equazioni di traffico.

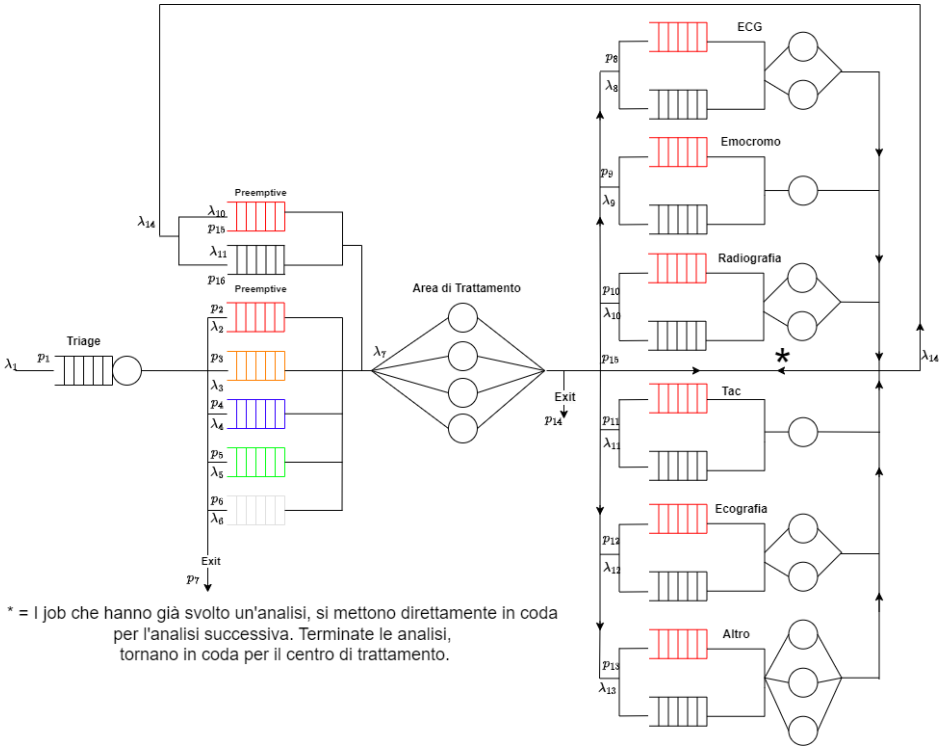

Figure 2: Schema del sistema in oggetto, con flussi e probabilità

Equazioni di traffico

$\lambda_2 = \lambda_1 \cdot p_2$
$\lambda_3 = \lambda_1 \cdot p_3$
$\lambda_4 = \lambda_1 \cdot p_4$
$\lambda_5 = \lambda_1 \cdot p_5$
$\lambda_6 = \lambda_1 \cdot p_6$
$\lambda_7 = \lambda_1 \cdot (1 - p_7) + \lambda_{14}$
$\lambda_8 = \lambda_7 \cdot p_8$
$\lambda_9 = \lambda_7 \cdot p_9$
$\lambda_{10} = \lambda_7 \cdot p_{10}$
$\lambda_{11} = \lambda_7 \cdot p_{11}$
$\lambda_{12} = \lambda_7 \cdot p_{12}$
$\lambda_{13} = \lambda_7 \cdot p_{13}$
$\lambda_{14} = \lambda_7 \cdot p_{15}$
$p_8 + p_9 + p_{10} + p_{11} + p_{12} + p_{13} = p_{15}$
$p_{14} + p_{15} = 1$

Modellazione dei centri

In questa sezione analizzeremo le scelte di progettazione per ogni centro effettuate durante la modellazione del sistema.

Triage

Il centro di Triage viene modellato con un server con coda M/G/1.

- La distribuzione dei tempi di interarrivo è Esponenziale di parametro $\frac{1}{\lambda_1}$
- La distribuzione dei servizi è rappresentata da una Normale Troncata, avente come tempo di servizio medio E[S] = 5.5 minuti. Questo tempo di servizio può oscillare tra un minimo di 2 e un massimo di 7 minuti.
- In questa fase non è presente nessuna possibilità di uscita.

Centro di trattamento

Il centro di trattamento viene modellato con un multiserver con coda M/G/4.

- La distribuzione dei tempi di interarrivo è Esponenziale di parametro $\frac{1}{\lambda_7}$.

- La distribuzione dei servizi è rappresentata da una Normale Troncata, avente come tempo di servizio medio E[S] = 16 minuti. Questo tempo di servizio può oscillare tra un minimo di 10 e un massimo di 27 minuti.

- In questa fase un job può abbandonare il sistema in due casi:

  - Con probabilità p_7 prima di esser elaborato (valido solo per chi non ha ancora svolto le analisi).
  - Con probabilità p_{14} dopo essere stato elaborato.

Elettrocardiogramma (ECG)

L'area di ECG viene modellata con un multiserver con coda M/G/2.

- La distribuzione dei tempi di interarrivo è Esponenziale di parametro $\frac{1}{\lambda_8}$.

- La distribuzione dei servizi è rappresentata da una Normale Troncata, avente come tempo di servizio medio E[S] = 5 minuti. Questo tempo di servizio può oscillare tra un minimo di 3.5 e un massimo di 9 minuti.

- In questa fase non è presente nessuna possibilità di uscita.

Emocromo

L'area di Emocromo viene modellata con un multiserver con coda M/G/2.

- La distribuzione dei tempi di interarrivo è Esponenziale di parametro $1\frac{1}{\lambda_9}$.

- La distribuzione dei servizi è rappresentata da una Normale Troncata, avente come tempo di servizio medio E[S] = 5 minuti. Questo tempo di servizio può oscillare tra un minimo di 3.5 e un massimo di 8 minuti.

- In questa fase non è presente nessuna possibilità di uscita.

Radiografia

L'area di Radiografia viene modellata con un multiserver con coda M/G/2.

- La distribuzione dei tempi di interarrivo è Esponenziale di parametro $\frac{1}{\lambda_{10}}$.

- La distribuzione dei servizi è rappresentata da una Normale Troncata, avente come tempo di servizio medio E[S] = 25 minuti. Questo tempo di servizio può oscillare tra un minimo di 20 e un massimo di 33 minuti.

- In questa fase non è presente nessuna possibilità di uscita.

Tac

L'area di Tac viene modellata con un multiserver con coda M/G/2.

- La distribuzione dei tempi di interarrivo è Esponenziale di parametro $\frac{1}{\lambda_{11}}$.

- La distribuzione dei servizi è rappresentata da una Normale Troncata, avente come tempo di servizio medio E[S] = 25 minuti. Questo tempo di servizio può oscillare tra un minimo di 17 e un massimo di 38 minuti.

- In questa fase non è presente nessuna possibilità di uscita.

Ecografia

L'area di Ecografia viene modellata con un multiserver con coda M/G/2.

- La distribuzione dei tempi di interarrivo è Esponenziale di parametro $\frac{1}{\lambda_{12}}$.

- La distribuzione dei servizi è rappresentata da una Normale Troncata, avente come tempo di servizio medio E[S] = 15 minuti. Questo tempo di servizio può oscillare tra un minimo di 10 e un massimo di 20 minuti.

- In questa fase non è presente nessuna possibilità di uscita.

Altro

L'area adibita alle altre analisi viene modellata con un multiserver con coda M/G/2.

- La distribuzione dei tempi di interarrivo è Esponenziale di parametro $\frac{1}{\lambda_{13}}$
- La distribuzione dei servizi è rappresentata da una Normale Troncata, avente come tempo di servizio medio E[S] = 35 minuti. Questo tempo di servizio può oscillare tra un minimo di 20 e un massimo di 40 minuti.

- In questa fase non è presente nessuna possibilità di uscita.

6. Modello computazionale

Il linguaggio di programmazione utilizzato è stato Python; la simulazione è di tipo Next-Event.

Struttura

Il progetto è strutturato come segue:

- la directory controller, che contiene la logica della simulazione, relativa alla gestione dei job nel Triage, nell'area di Trattamento e nelle varie Aree di Analisi.
- la directory model contiene l'implementazione del singolo job come classe.
- la directory utility contiene tutte le facility e le librerie esterne utilizzate.

Pseudo Random Number Generator

Per la generazione di numeri pseudo-casuali è stato utilizzato l'algoritmo di Lehmer contenuto nelle librerie prodotte da Steve Park e Dave Geyer. Viene utilizzato un approccio multi-stream che garantisce l'indipendenza tra i numeri generati.

Realizzazione di una normale troncata

Di seguito riportiamo il codice per la realizzazione di una Normale Troncata seguendo le linee guida riportate nel libro di testo [3], il troncamento è necessario affinché non vi siano accumuli agli estremi.

```
def idfTruncatedNormal (m, s, lower_bound, upper_bound):
    a = cdfNormal(m, s, lower_bound)
    b = 1.0-cdfNormal(m, s, upper_bound)
    u = idfUniform(a, 1.0-b, random())
    return idfNormal(m, s, u)
```

Descrizione del Simulation Controller

Per garantire modularità nel codice si è optato per una gestione decentralizzata dei vari centri. Per effettuare una simulazione unica si allineano i singoli clock in maniera tale da ottenere in maniera univoca il prossimo evento da simulare. Di seguito la funzione *next_event*, che seleziona in base ai valori dei vari clock qual è il prossimo evento da eseguire:

Selezione prossimo evento

```
def next_event(current_triage: int, current_queue: int, t_analisi: list):
    prox_evento = INFINITY
    for i in range(len(t_analisi)):
        if t_analisi[i].current > 0:
            prox_evento = min(prox_evento, t_analisi[i].current)

    if current_queue > 0:
        prox_evento = min(prox_evento, current_queue)

    if current_triage > 0:
        prox_evento = min(prox_evento, current_triage)

    if prox_evento >= INFINITY:
        prox_evento = -1

    return prox_evento
```

Mostriamo ora la funzione switch che sincronizza i clock, ed esegue l'evento selezionato da *next_event*.

```
def switch(prox_operazione, t_triage, t_queue, t_analisi):
    t_triage.current = prox_operazione
    t_queue.current = prox_operazione
    for i in range(len(t_analisi)):
        t_analisi[i].current = prox_operazione

    if prox_operazione == t_triage.arrival:
        processa_arrivo_triage()
    elif prox_operazione == t_triage.min_completion:
        processa_completamento_triage()
    elif prox_operazione == t_queue.min_completion:
        processa_completamento_queue()
    else:
        for i in range(len(t_analisi)):
            if prox_operazione == t_analisi[i].min_completion:
                processa_completamento_analisi(i)
                break
```

Gestione Code

Prendiamo come esempio la gestione dell'area di trattamento, potendo così mostrare la gestione delle code preemptive presenti. Mostriamo di seguito il metodo *arrival_queue*, ovvero il metodo utilizzato per gestire l'arrivo di un nuovo job nell'area di trattamento.

```
def arrival_queue(t, servers_busy, queue_q):
    global job_att_inter_queue
    full = True
    index = -1
    codice = -1
```

```
if t_queue.arrival > Parameters.STOP:
    t_queue.arrival = INFINITY

for i in range(NUMERO_DI_SERVER_QUEUE):
    if not servers_busy[i]:  # check if server is free
        full = False
        job_to_serve = get_next_job_to_serve(queue_q, t_queue)
        if job_to_serve:
            servers_busy[i] = True
            server_queue[i] = job_to_serve
            if job_to_serve.get_tempo_rimanente() == 0:
                temp = GetServiceQueue()
                area_queue.service_color[
                    get_queue(job_to_serve.get_codice(), job_to_serve.get_uscita())] += temp
                area_queue.service[i] += temp
                # print(area_queue.service)
                t_queue.completion[i] = t.current + temp
                job_to_serve.set_queue_time(t.current)

            else:
                job_att_inter_queue = job_att_inter_queue - 1
                t_queue.completion[i] = t.current + job_to_serve.get_tempo_rimanente()
                area_queue.service[i] += job_to_serve.get_tempo_rimanente()
                area_queue.service_preemption[get_queue(job_to_serve.get_codice(),
                job_to_serve.get_uscita())] += t.current - job_to_serve.get_interrotto()
            break
```

All'arrivo di un nuovo job si seleziona il prossimo job da servire, si verifica che ci siano server liberi e si inizia il processamento.

```
# preemption
if full and (len(queue_q[0]) + len(queue_q[1])) > 0:
    job_to_serve = get_next_job_to_serve(queue_q, t_queue)

    for i in range(NUMERO_DI_SERVER_QUEUE):
        if isinstance(server_queue, list):
            temp = server_queue[i].get_codice()
            if temp > codice:
                codice = temp
                index = i

    job_interrotto = server_queue[index]

    if job_interrotto.get_codice() == 1 or t.completion[index] == t.current:
        coda_preemptive(queue_q, job_to_serve)

    else:

        job_to_serve.set_queue_time(t.current)
```

```
            job_interrotto.set_interrotto(t.current)
            job_att_inter_queue = job_att_inter_queue + 1
            job_interrotto.set_tempo_rimanente(t.completion[index] - t.current)
            area_queue.service[index] -= (t.completion[index] - t.current)
            server_queue[index] = job_to_serve
            coda_preemptive(queue_q, job_interrotto)
            temp = GetServiceQueue()
            t_queue.completion[index] = t_queue.current + temp
            area_queue.service_color[get_queue(job_to_serve.get_codice(),
            job_to_serve.get_uscita())] += temp
            area_queue.service[index] += temp
```

Se tutti i server sono occupati ma c'è un codice rosso si seleziona il job con codice meno grave e si procede con l'interruzione del suddetto job e l'inizio del processamento del job con codice rosso. Notiamo che i codici rossi occupano code di tipo preemptive, per tanto non è possibile interrompere la loro lavorazione.

Introduciamo ora la funzione get_job_old necessaria per la selezione del prossimo job, questa funzione va a controllare se un determinato job supera il tempo limite impostato a 7 ore, e se riceve esito positivo, lo seleziona come prossimo job da processare.

```
def get_job_old(list_of_queues, t):
    for queue in list_of_queues:
        if queue and queue[0] and (t.current - queue[0].get_id()) > TEMPO_LIMITE:
            return queue.pop(0)
    return None
```

Illustriamo ora il processo per la selezione del prossimo job da elaborare, esso avviene tramite la funzione next_job_to_serve. Andiamo inanzitutto a controllare se le due code preemptive, quindi di proprità assoluta contengono job, e in caso negativo, andiamo a controllare se sono presenti job che superano il tempo limite di permanenza, se anche questo controllo risulta negativo, selezioniamo il prossimo job partendo dalla coda con priorità maggiore. All'interno della singola coda, lo scheduling utilizzato è di tipo FIFO.

```
def get_next_job_to_serve(list_of_queues, t: Time = 0):
    if len(list_of_queues) == 7:
        if (len(list_of_queues[0]) + len(list_of_queues[1])) <= 0:
            job = get_job_old(list_of_queues, t)
            if job:
                return job

    """Selects the next job to serve based on a priority policy"""
    for queue in list_of_queues:
        if queue:
            return queue.pop(0)
            # Simple FIFO (First In, First Out)
            # policy for each priority queue
    return None
```

7. Verifica

Per verificare la correttezza dell'implementazione del modello computazionale, effettuiamo la fase di verifica. Per poter effettuare un confronto analitico vengono utilizzati tempi di servizio di tipo esponenziale. Inoltre,

per semplificare il calcolo, è stata utilizzata in ogni area una coda singola. Il confronto avverrà confrontando i risultati ottenuti con simulazione ad orizzonte infinito.

Triage

$\lambda_1 = 0.092163242\ job/min$

$E(S) = 5.5\ min$

$m = 1$

$\rho_1 = \lambda_1 \cdot E(S) = 0.506898\ min$

$E(T_S) = \frac{1}{\mu_1 - \lambda_1} = 11.153875\ min$

```
Stats for triage:
Rho: 0.499177
E[Ts]: 10.852717 min
```

Figure 3: Statistiche del Triage dalla simulazione

Area di trattamento

$\lambda_{14} = \lambda_7 * p_{15}$

$\lambda_7 = \frac{\lambda_1 * (1 - p_7)}{1 - p_{15}} = 0.233620 job/min$

$E(S_i) = 16\ min$

$m = 4$

$E(S) = 4\ min$

$\rho = 0.93448$

$p_7 = 0.02$

$p_{15} = 0.377141$

$p(0) = \left[\sum_{i=0}^{m-1} \frac{(m \cdot \rho)^i}{i!} + \frac{(m \cdot \rho)^m}{m! \cdot (1 - \rho)}\right]^{-1} = 6.916827 * 10^{-3}$

$P_Q = \frac{(m\rho)^m}{m!(1-\rho)} p(0) = 0.8587$

$E(T_Q) = 52.42373\ min$

$E(T_S) = 68.423729\ min$

```
Stats for queue:
Rho: 0.934498
E[Tq]: 52.626304 min
E[Ts]: 68.620094 min
```

Figure 4: Statistiche dell'area di trattamento dalla simulazione

Aree di Analisi

Dato il complesso funzionamento delle aree di analisi, dove ogni paziente può spostarsi tra le diverse analisi senza dover ripassare per il centro di trattamento, risulta impossibile testare quest'area in modo analitico. Pertanto, abbiamo semplificato il sistema, assumendo che ogni paziente, una volta superata l'area di trattamento, si sottoponga a una singola analisi.

Elettrocardiogramma (ECG)

$p_8 = 0.124126$

$E(S_i) = 5\ min$

$E(S) = 2.5\ min$

$m = 2$

$\lambda_8 = 0.028998\ job/min$

$\rho_8 = 0.0725$

$p(0) = \left[\sum_{i=0}^{m-1} \frac{(m\cdot\rho)^i}{i!} + \frac{(m\cdot\rho)^m}{m!\cdot(1-\rho)}\right]^{-1} = 0.864802$

$P_Q = \frac{(m\rho)^m}{m!(1-\rho)} p(0) = 0.009802$

$E(T_S) = 5.02642\ min$

```
Stats for ECG:
Rho: 0.077112
E[Ts]: 5.077760 min
```

Figure 5: Statistiche dell'ECG dalla simulazione

Emocromo

$p_9 = 0.145311$

$E(S) = 5\ min$

$m = 1$

$\lambda_9 = 0.033947\ job/min$

$\rho_9 = 0.169738$

$E(T_S) = 6.022174\ min$

```
Stats for Emocromo:
Rho: 0.169152
E[Ts]: 6.071372 min
```

Figure 6: Statistiche dell'emocromo dalla simulazione

Radiografia

$p_{10} = 0.098751$

$E(S_i) = 25\ min$

$E(S) = 12.5\ min$

$m = 2$

$\lambda_{10} = 0.02307\ job/min$

$\rho_{10} = 0.288377$

$p(0) = \left[\sum_{i=0}^{m-1} \frac{(m\cdot\rho)^i}{i!} + \frac{(m\cdot\rho)^m}{m!\cdot(1-\rho)}\right]^{-1} = 0.55234$

$P_Q = \frac{(m\rho)^m}{m!(1-\rho)} p(0) = 0.129095$

$E(T_S) = 27.26761\ min$

```
Stats for Radiografia:
Rho: 0.328218
E[Ts]: 26.692143 min
```

Figure 7: Statistiche della radiografia dalla simulazione

Tac

$p_{11} = 0.030975$

$E(S) = 25\ min$

$m = 1$

$\lambda_{11} = 7.236379 * 10^{-3}\ job/min$

$\rho_{11} = 0.180909$

$E(T_S) = 30.521656\ min$

```
Stats for Tac:
Rho: 0.182388
E[Ts]: 31.181200 min
```

Figure 8: Statistiche della tac dalla simulazione

Ecografia

$p_{12} = 0.059289$

$E(S_i) = 15\ min$

$E(S) = 7.5\ min$

$m = 2$

$\lambda_{12} = 0.013851\ job/min$

$\rho_{12} = 0.103883$

$p(0) = \left[\sum_{i=0}^{m-1} \frac{(m\cdot\rho)^i}{i!} + \frac{(m\cdot\rho)^m}{m!\cdot(1-\rho)}\right]^{-1} = 0.811786$

$P_Q = \frac{(m\rho)^m}{m!(1-\rho)} p(0) = 0.022279$

$E(T_S) = 15.186461\ min$

```
Stats for Ecografia:
Rho: 0.115537
E[Ts]: 15.587382 min
```

Figure 9: Statistiche dell'ecografia dalla simulazione

Altro

$p_{13} = 0.158199$

$E(S_i) = 35\ min$

$E(S) = 11.67\ min$

$m = 3$

$\lambda_{13} = 0.036958\ job/min$

$\rho_{13} = 0.431177$

$p(0) = \left[\sum_{i=0}^{m-1} \frac{(m\cdot\rho)^i}{i!} + \frac{(m\cdot\rho)^m}{m!\cdot(1-\rho)}\right]^{-1} = 0.265654$

$P_Q = \frac{(m\rho)^m}{m!(1-\rho)} p(0) = 0.03865$

$E(T_S) = 35.792945\ min$

```
Stats for Altro:
Rho: 0.539056
E[Ts]: 39.018222 min
```

Figure 10: Statistiche delle altre analisi dalla simulazione

8. Validazione

Arrivati a questo punto ci chiediamo se il modello computazionale è consistente con il sistema analizzato. Possiamo effettuare questo controllo andando a controllare i dati riportati nel report effettuato dalla Regione Lazio che riportano:

- **Livello 2 (Arancione)**: 57 minuti

- **Livello 3 (Azzurro)**: 164 minuti
- **Livello 4 (Verde)**: 154 minuti
- **Livello 5 (Bianco)**: 148 minuti

Possiamo confrontarli con i nostri dati che invece riportano:

```
Delay Time Metrics Queue:
E[Tq] per coda Rossa = 0.000000 +/- 0.000000
E[Tq] per coda Arancione = 44.806592 +/- 24.208421
E[Tq] per coda Azzurro: 4 = 83.651043 +/- 26.095426
E[Tq] per coda Verde: 5 = 207.080806 +/- 33.528570
E[Tq] per coda Bianco: 6 = 476.535784 +/- 259.405288
```

Considerando le ipotesi adottate, la complessità del sistema composto da molteplici centri e con presenza di feedback, e l'elevata variabilità degli arrivi, la differenza tra i dati del report e i risultati della simulazione è imputabile a fattori non prevedibili e non simulabili

.

9. Design degli esperimenti

Segue la sezione dedicata agli esperimenti sul sistema in esame contenente simulazioni ad orizzonte finito ed infinito. Nonostante il QoS sia basato sui tempi di attesa abbiamo deciso di mostrare i grafici rappresentanti la popolazione media, queste due metriche sono infatti direttamente correlate tra di loro e un aumento nella popolazione media in un'area del sistema implica un incremento dei tempi di attesa. Pertanto, analizzare la popolazione media permette di trarre le stesse conclusioni, ma con un'interpretazione grafica più comprensibile.

Simulazione ad orizzonte finito

La simulazione del sistema è stata effettuata con la tecnica della Replicazione, in modo da ottenere stime indipendenti per le statistiche transitorie. Il seed iniziale è quello di default della libreria pari a '123456789', inizializzato esternamente al ciclo che ripete le simulazioni, utilizzando lo stato finale di ciascuna sequenza di numeri casuali come stato iniziale della replicazione successiva. Il tempo di simulazione è stato posto pari a 10080 che, utilizzando come scala il minuto, è pari ad una settimana. L'intervallo di confidenza è stato individuato seguendo i passi descritti dall'algoritmo 8.1.1. del libro di testo [3]. Scegliendo come livello di confidenza $95\% = (1 - \alpha\%)$, considerando il tipico valore $\alpha = 0.05$, e computando le statistiche campionarie quali media e deviazione standard, possiamo calcolare il valore $t^* = \text{idfStudent}(n - 1, 1 - \frac{\alpha}{2})$, con il quale verranno identificati gli estremi degli intervalli. Dato un numero di ripetizioni sufficientemente grande possiamo affermare di essere al $(1 - \alpha) \cdot 100\%$ confidenti che la media ricada nell'intervallo definito, ovvero, se andassimo a considerare molteplici intervalli di confidenza, approssimativamente il $(1 - \alpha) \cdot 100\%$ di questi coprirebbe il vero valore della media. Di seguito riportiamo i risultati ottenuti dalla simulazione effettuata con 64 ripetizioni:

```
Delay Time Metrics Queue:
    E[Tq] per coda: 0 = 0.0 +/- 0.0
    E[Tq] per coda: 1 = 0.0 +/- 0.0
    E[Tq] per coda: 2 = 5.396782445287604 +/- 0.3027137050006729
    E[Tq] per coda: 3 = 25.83206773475233 +/- 9.103358680251546
    E[Tq] per coda: 4 = 60.54372596466341 +/- 16.497651875962568
    E[Tq] per coda: 5 = 164.98239720090498 +/- 22.9239013176139
    E[Tq] per coda: 6 = 256.7190775740093 +/- 39.558459045923414

Response Time Metrics Queue:
    E[Ts] per coda: 0 = 16.44959037439959 +/- 0.12439659370358215
    E[Ts] per coda: 1 = 16.51468201625013 +/- 0.0961656193685675
    E[Ts] per coda: 2 = 22.163293679915853 +/- 0.3142099278466246
    E[Ts] per coda: 3 = 42.322571367058956 +/- 9.108178367113688
    E[Ts] per coda: 4 = 77.62021698558107 +/- 16.535715440752526
    E[Ts] per coda: 5 = 184.46383301363693 +/- 22.891412875326647
    E[Ts] per coda: 6 = 278.69548217270005 +/- 39.63629107574517

Utilization Metrics Queue:
    Rho per server: 0 = 0.9616266791439448 +/- 0.004299611401061055
    Rho per server: 1 = 0.9484960749881073 +/- 0.005439736683876707
    Rho per server: 2 = 0.9328513988870035 +/- 0.007370504382997699
    Rho per server: 3 = 0.9108562996982836 +/- 0.010240457401664363
```

Figure 11: Risultati simulazione finita con 64 ripetizioni

Di seguito invece riportiamo il numero di violazioni registrate in media durante la simulazione suddivise per codici:

```
Il colore:  ROSSO  non ha sforamenti.

Ci sono state mediamente  60 violazioni in ogni ripetizione per il colore ARANCIONE . La media per singola violazione è di:  53.10805846560702 .
Con una percentuale di job che violano il QoS di:  0.33330461008186124

Ci sono state mediamente  137 violazioni in ogni ripetizione per il colore AZZURRO . La media per singola violazione è di:  113.5476182401635 .
Con una percentuale di job che violano il QoS di:  0.385114119244754

Ci sono state mediamente  162 violazioni in ogni ripetizione per il colore VERDE . La media per singola violazione è di:  213.01483384305232 .
Con una percentuale di job che violano il QoS di:  0.576833720544308

Ci sono state mediamente  28 violazioni in ogni ripetizione per il colore BIANCO . La media per singola violazione è di:  287.7465859114174 .
Con una percentuale di job che violano il QoS di:  0.5930764206401045
```

Figure 12: Violazioni riportate durante la simulazione finita con 64 ripetizioni

Riportiamo ora i grafici sviluppati durante la simulazione ad orizzonte finito:

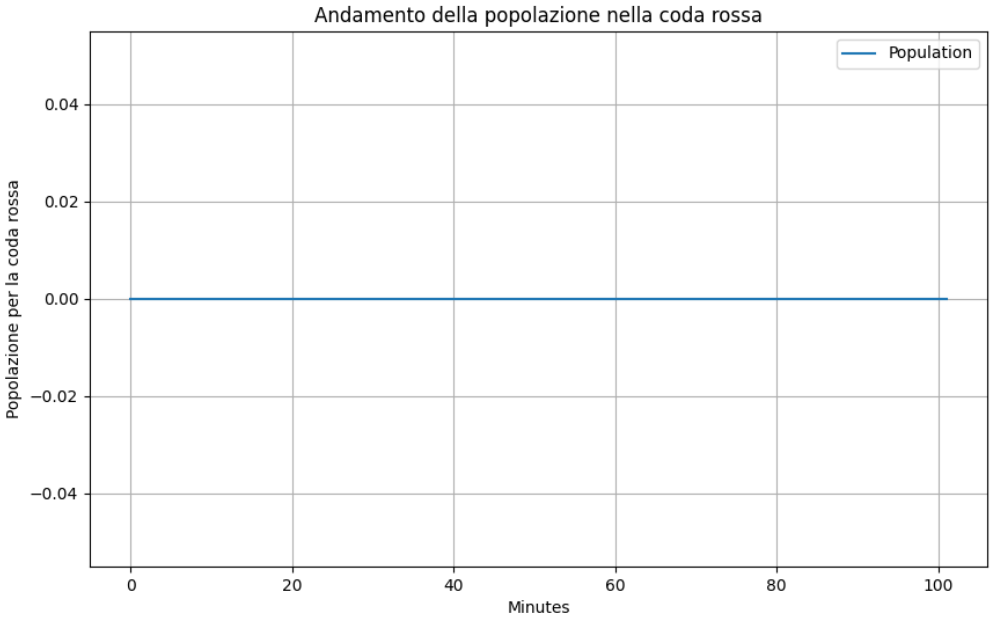

Figure 13: Popolazione media nella coda rossa

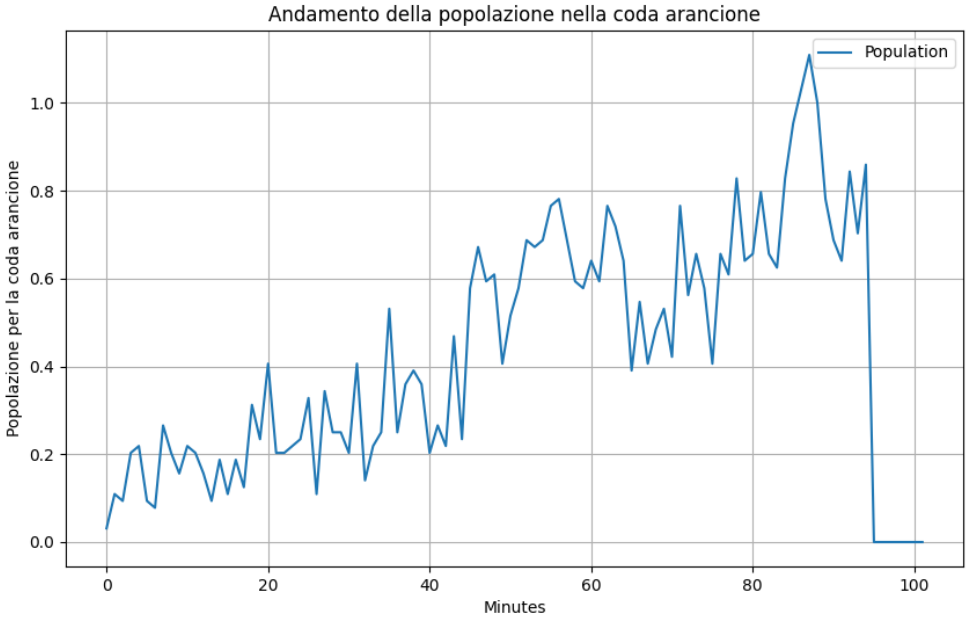

Figure 14: Popolazione media nella coda arancione

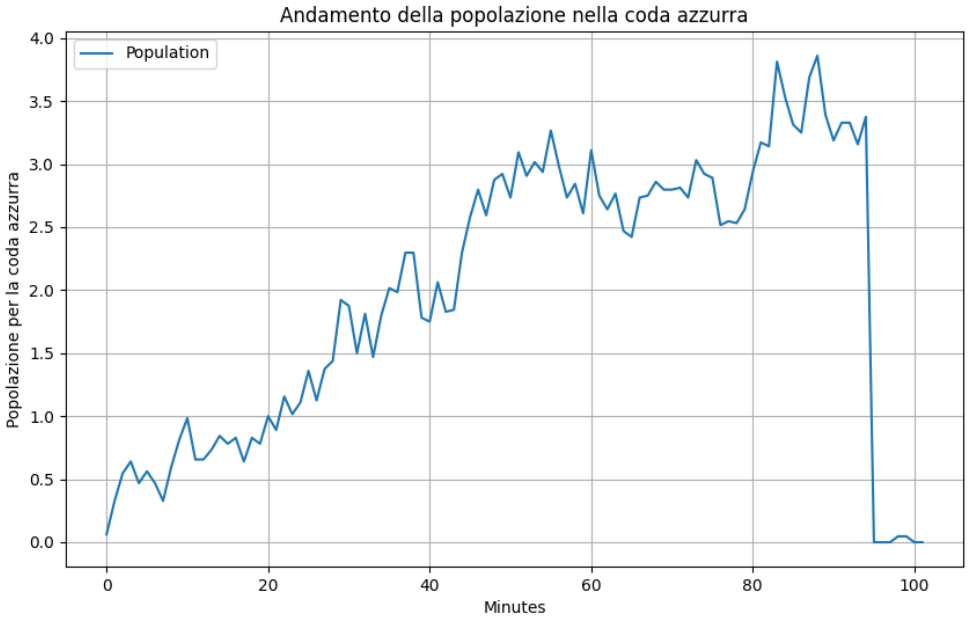

Figure 15: Popolazione media nella coda azzurra

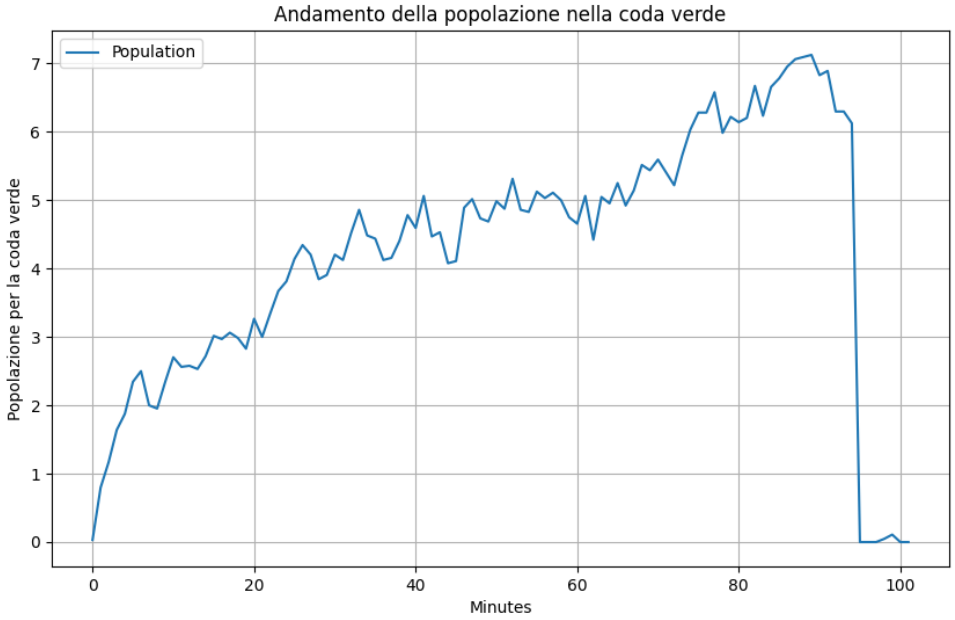

Figure 16: Popolazione media nella coda verde

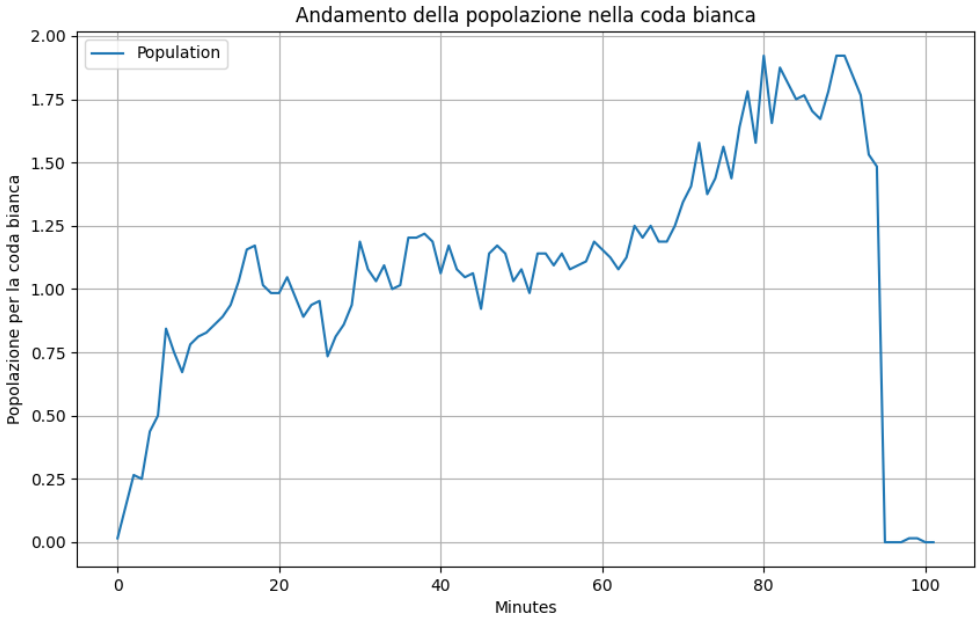

Figure 17: Popolazione media nella coda bianca

Simulazione ad orizzonte infinito

La simulazione a orizzonte infinito vede il comportamento del sistema per un tempo tendente all'infinito, per approssimare tale scenario, è stato scelto un tempo di simulazione molto lungo. Per studiare il comportamento del sistema nello stato stazionario, è stato adottato il metodo Batch Means, che aiuta a eliminare il bias dovuto allo stato iniziale del sistema. Questo metodo prevede che le statistiche di ogni batch, eccetto il primo, vengano inizializzate allo stato del sistema al momento del reset dei contatori statistici del batch.

Per scegliere i parametri b e k, rispettivamente il numero di batch e il numero di elementi per batch, è stato utilizzato il programma *acs.py*. Questo programma ha fornito indicazioni sui valori ottimali per questi parametri, al fine di garantire una stima affidabile e precisa delle statistiche del sistema. I valori determinati sono stati $b = 64$ e $k = 512$, indicando che il tempo di simulazione è stato suddiviso in 64 batch, ciascuno contenente 512 osservazioni. Questi parametri sono stati scelti per bilanciare l'accuratezza delle stime con la variabilità all'interno dei batch e tra i batch, assicurando così risultati robusti e rappresentativi dello stato stazionario del sistema.

```
Delay Time Metrics Queue:
    E[Tq] per coda: 0 = 0.000000 +/- 0.000000
    E[Tq] per coda: 1 = 0.000000 +/- 0.000000
    E[Tq] per coda: 2 = 5.693010 +/- 0.333541
    E[Tq] per coda: 3 = 44.806592 +/- 24.208421
    E[Tq] per coda: 4 = 83.651043 +/- 26.095426
    E[Tq] per coda: 5 = 207.080806 +/- 33.528570
    E[Tq] per coda: 6 = 476.535784 +/- 259.405288

Response Time Metrics Queue:
    E[Ts] per coda: 0 = 16.428008 +/- 0.163011
    E[Ts] per coda: 1 = 16.531337 +/- 0.119973
    E[Ts] per coda: 2 = 22.467729 +/- 0.344672
    E[Ts] per coda: 3 = 61.382088 +/- 24.221325
    E[Ts] per coda: 4 = 100.790541 +/- 26.120698
    E[Ts] per coda: 5 = 226.634399 +/- 33.550272
    E[Ts] per coda: 6 = 498.670899 +/- 260.378295

Utilization Metrics Queue:
    E[Rho] per server: 0 = 0.978552 +/- 0.005068
    E[Rho] per server: 1 = 0.972042 +/- 0.006709
    E[Rho] per server: 2 = 0.961912 +/- 0.009401
    E[Rho] per server: 3 = 0.949600 +/- 0.012005
```

Figure 18: Risultati ottenuti dalla simulazione infinita

Di seguito invece riportiamo il numero di violazioni registrate durante la simulazione suddivise per codici:

```
Il colore:  ROSSO  non ha sforamenti.

Ci sono state  2784 violazioni per il colore ARANCIONE . La media per singola violazione è di:  85.18889201780038 .
Con una percentuale di job che violano il QoS di:  0.3845835060091173

Ci sono state  6348 violazioni per il colore AZZURRO . La media per singola violazione è di:  143.4526718730426 .
Con una percentuale di job che violano il QoS di:  0.4462565905096661

Ci sono state  7378 violazioni per il colore VERDE . La media per singola violazione è di:  241.3679941517809 .
Con una percentuale di job che violano il QoS di:  0.6539620634639248

Ci sono state  1275 violazioni per il colore BIANCO . La media per singola violazione è di:  540.0016199913874 .
Con una percentuale di job che violano il QoS di:  0.6616502335236119
```

Figure 19: Violazioni registrate nella simulazione infinita

Riportiamo ora i grafici sviluppati durante la simulazione ad orizzonte infinito:

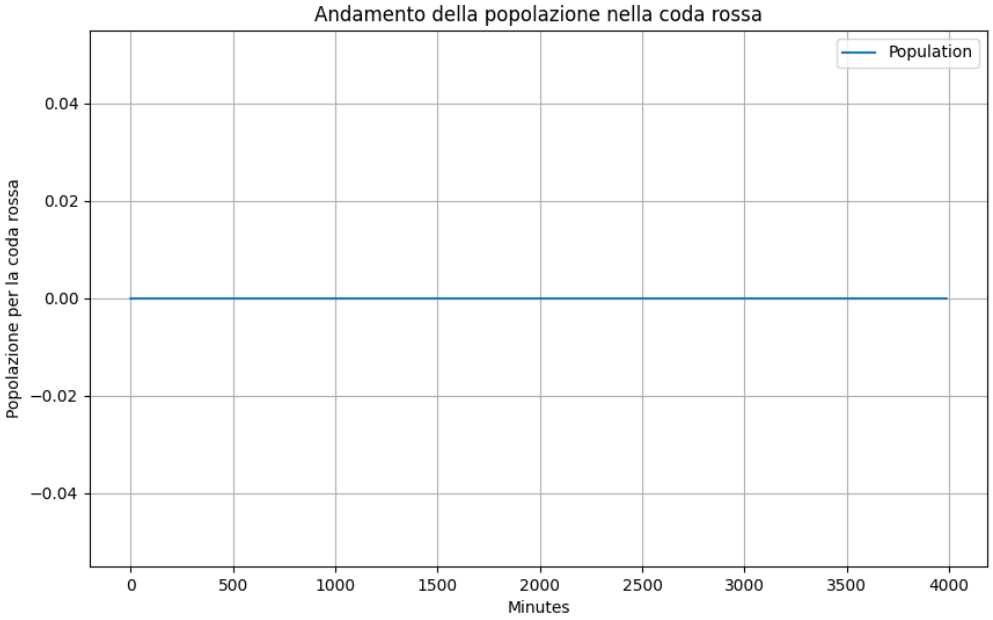

Figure 20: Popolazione media nella coda rossa

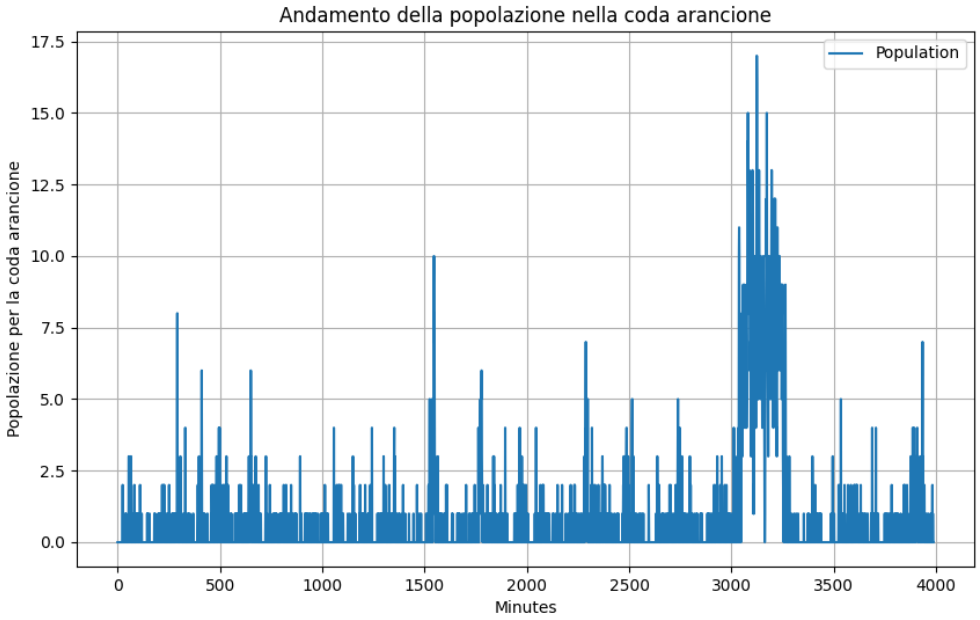

Figure 21: Popolazione media nella coda arancione

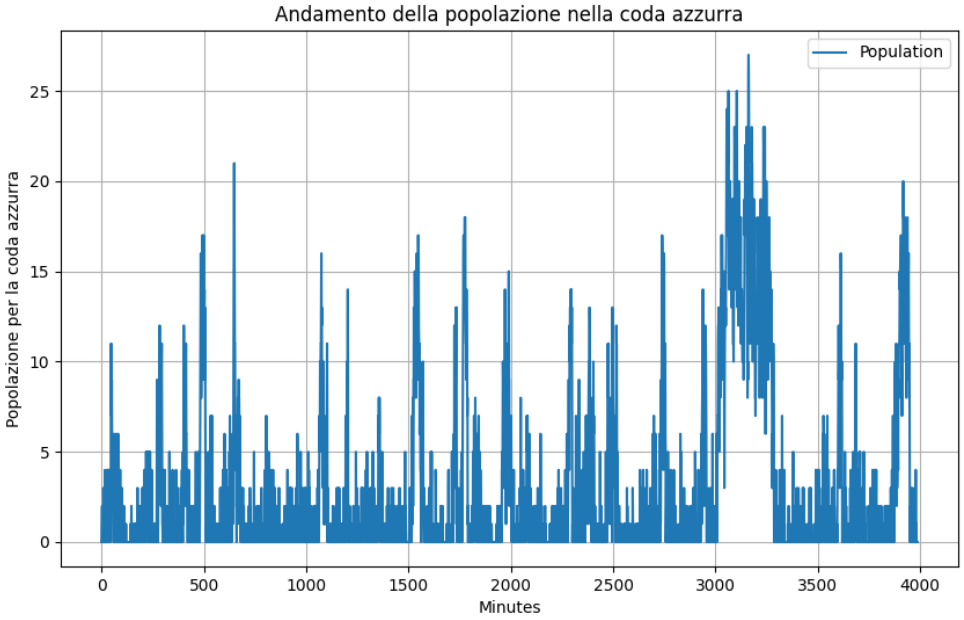

Figure 22: Popolazione media nella coda azzurra

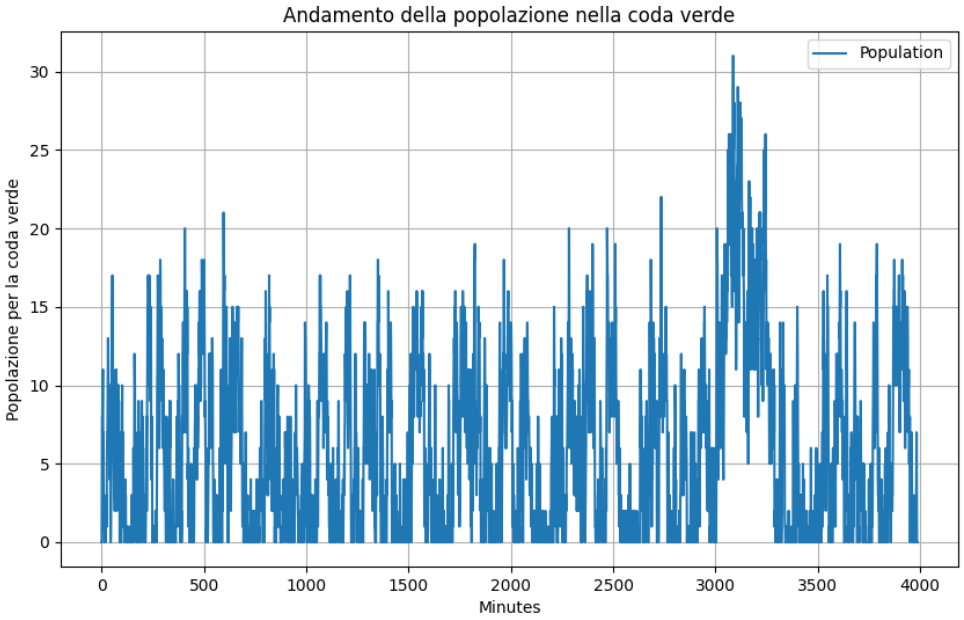

Figure 23: Popolazione media nella coda verde

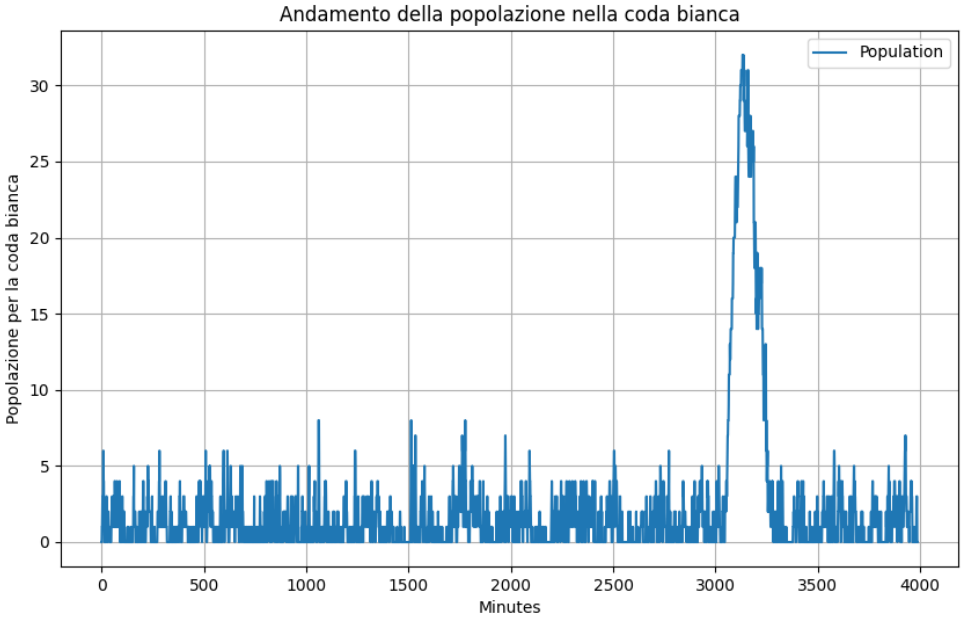

Figure 24: Popolazione media nella coda bianca

10. Considerazioni

Dall'analisi del sistema di pronto soccorso emergono considerazioni significative riguardo l'efficienza e l'efficacia del servizio. I dati relativi alla simulazione ad orizzonte finito indicano che una percentuale elevata di pazienti (job) non riesce ad accedere all'area di trattamento entro i tempi limite stabiliti. In particolare, i dati mostrano che il 33% dei pazienti con codici di priorità arancione fino al 59% di quelli con codici bianchi non ricevono il trattamento entro il tempo previsto. Questa evidenza sottolinea una problematica critica nella gestione del flusso dei pazienti, suggerendo che il sistema attuale non è sufficientemente adeguato a garantire il rispetto delle tempistiche di trattamento.

11. Modello migliorativo

Le precedenti considerazioni hanno portato allo sviluppo di un modello migliorativo mirato a ottimizzare il processo di gestione dei pazienti. Il modello proposto non si concentra sull'aumento del personale medico, ma piuttosto su una riorganizzazione strategica del sistema di scheduling. L'idea centrale è quella di migliorare l'efficienza del sistema attraverso la ristrutturazione delle procedure di assegnazione dei pazienti ai trattamenti, ottimizzando così l'uso delle risorse disponibili. Modificando le modalità di scheduling, si intende ridurre i tempi di attesa e garantire un accesso più tempestivo alle aree di trattamento, specialmente per i casi urgenti e critici. I codici rossi, che rispettano già le tempistiche e necessitano di una priorità assoluta, sono esclusi da questo miglioramento.

Schedulazione dei Job

La nuova politica prevede che i job che abbiano superato il nuovo tempo limite, impostato specificamente per la coda di appartenenza, abbiano la precedenza. Questo criterio viene applicato a partire dalla coda arancione, mantenendo quindi la priorità tra i nuovi job che devono essere serviti. Il nuovo tempo limite è pari all'obiettivo da raggiungere a cui sono stati sottratti dieci minuti. Il numero di minuti è stato scelto simulando vari scenari, e scegliendo quello che produce i migliori risultati in termini di violazioni totali.

```
def get_job_old(list_of_queues, t):
     """Selects the next job to serve based on a priority policy"""
    if Parameters.migliorativo:
        for queue in list_of_queues:
            if queue and queue[0] and (t.current - queue[0].get_id()) > OBIETTIVO_MIGLIORATIVO[
                queue[0].get_codice() - 1]:
                return queue.pop(0)
```

In questo codice, il parametro `OBIETTIVO_MIGLIORATIVO` è popolato nel seguente modo:

```
OBIETTIVO_MIGLIORATIVO = [0, 5, 20, 50, 110, 230]
```

I tempi indicati rappresentano il limite imposto dal governo, a cui vengono sottratti 10 minuti per ogni codice di priorità. Questo approccio mira a ridurre le tempistiche, garantendo che i pazienti, sebbene non in condizioni critiche, ricevano comunque assistenza nei tempi previsti.

12. Verifica

Per poter eseguire una verifica sul modello migliorativo, dovremmo applicare alcune semplificazioni, che ci porterebbero, con parametri diversi, nelle stesse condizioni del modello standard. Per queste motivazioni concludiamo che anche in questo caso la verifica conferma la coerenza tra risultati analitici e dati dalla simulazione,

come indicato nel paragrafo 6.

13. Validazione

Poiché il modello proposto rappresenta un miglioramento rispetto alle attuali pratiche e non è ancora stato implementato in nessun pronto soccorso, non è possibile condurre una validazione completa e definitiva.

14. Design degli esperimenti

Segue la sezione dedicata agli esperimenti sul sistema in esame coinvolgendo simulazioni ad orizzonte finito ed infinito.

Simulazione ad orizzonte finito

Per mantenere la consistenza con la simulazione del modello standard, abbiamo mantenuto lo stesso seed, lo stesso numero di replicazioni e lo stesso tempo di simulazione. Di seguito riportiamo i risultati ottenuti dalla simulazione:

```
Delay Time Metrics Queue:
    E[Tq] per coda: 0 = 0.0 +/- 0.0
    E[Tq] per coda: 1 = 0.0005861281066733189 +/- 0.0011805428613302189
    E[Tq] per coda: 2 = 4.714901036897741 +/- 0.1272913918671659
    E[Tq] per coda: 3 = 11.469204309580842 +/- 0.5540786771300429
    E[Tq] per coda: 4 = 29.506781367601903 +/- 2.0389598452303024
    E[Tq] per coda: 5 = 137.26301583848402 +/- 22.33346155860314
    E[Tq] per coda: 6 = 705.8214881088984 +/- 201.54491488161614

Response Time Metrics Queue:
    E[Ts] per coda: 0 = 16.56944523375428 +/- 0.11059437531361181
    E[Ts] per coda: 1 = 16.47624222578281 +/- 0.08782054442769607
    E[Ts] per coda: 2 = 21.470645961339287 +/- 0.14314625347235552
    E[Ts] per coda: 3 = 28.084499849316348 +/- 0.5628597150424071
    E[Ts] per coda: 4 = 46.56167406336773 +/- 2.0592821310670515
    E[Ts] per coda: 5 = 157.182733982915 +/- 22.460918251927897
    E[Ts] per coda: 6 = 735.8824807308974 +/- 201.60717546065143

Utilization Metrics Queue:
    Rho per server: 0 = 0.9623078584589548 +/- 0.003977684554244986
    Rho per server: 1 = 0.9493724073562497 +/- 0.005395594828971421
    Rho per server: 2 = 0.9330629858310744 +/- 0.007203536494245334
    Rho per server: 3 = 0.9115841184345319 +/- 0.009618883366822316
```

Figure 25: Risultati simulazione finita modello migliorativo con 64 ripetizioni

Di seguito invece riportiamo il numero di violazioni registrate in media durante la simulazione suddivise per codici:

```
Il colore:  Rosso  non ha sforamenti.

Ci sono state mediamente  46 violazioni in ogni ripetizione per il colore Arancione . La media per singola violazione è di:  14.591765200862701 .
Con una percentuale di job che violano il QoS di:  0.25497630331753557

Ci sono state mediamente  129 violazioni in ogni ripetizione per il colore Azzurro . La media per singola violazione è di:  28.766757563254608 .
Con una percentuale di job che violano il QoS di:  0.36430542778288866

Ci sono state mediamente  157 violazioni in ogni ripetizione per il colore Verde . La media per singola violazione è di:  170.87552269949936 .
Con una percentuale di job che violano il QoS di:  0.5559243279123797

Ci sono state mediamente  27 violazioni in ogni ripetizione per il colore Bianco . La media per singola violazione è di:  1068.4466284611747 .
Con una percentuale di job che violano il QoS di:  0.5685826257348139
```

Figure 26: Violazioni medie registrate durante la simulazione ad orizzonte finito

Riportiamo ora i grafici sviluppati durante la simulazione ad orizzonte finito:

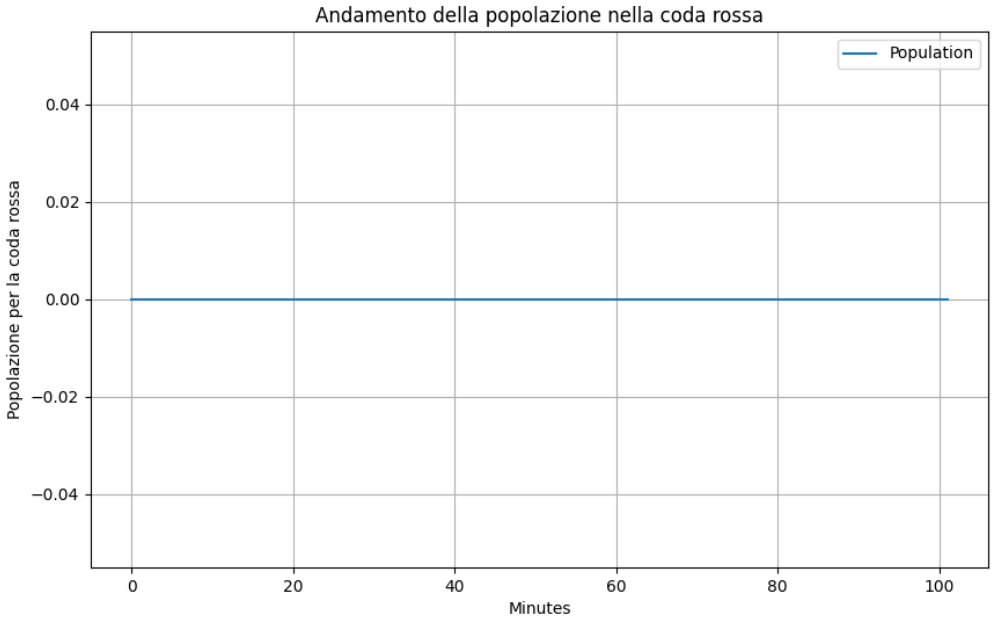

Figure 27: Popolazione media nella coda rossa

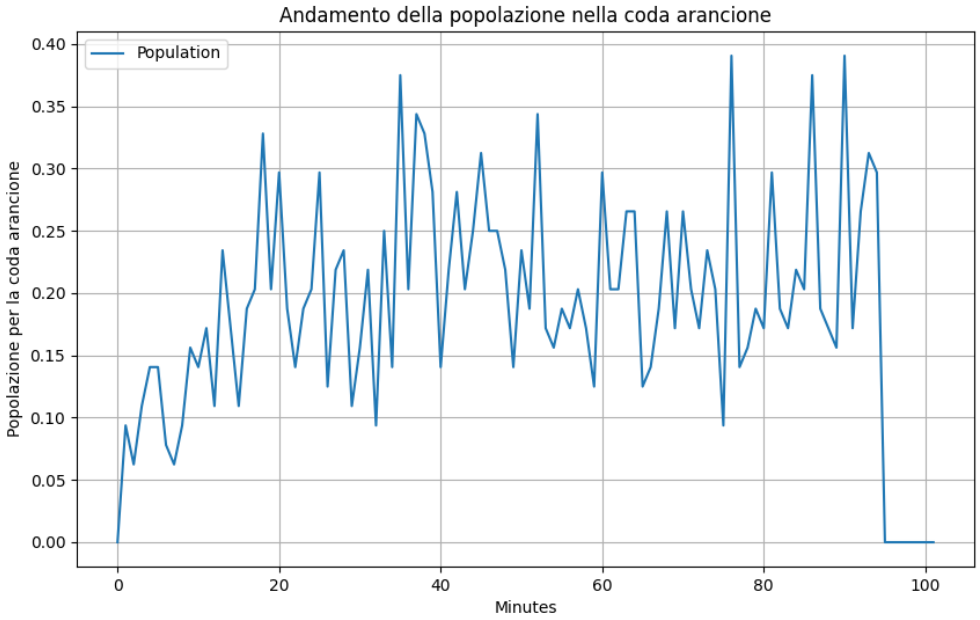

Figure 28: Popolazione media nella coda arancione

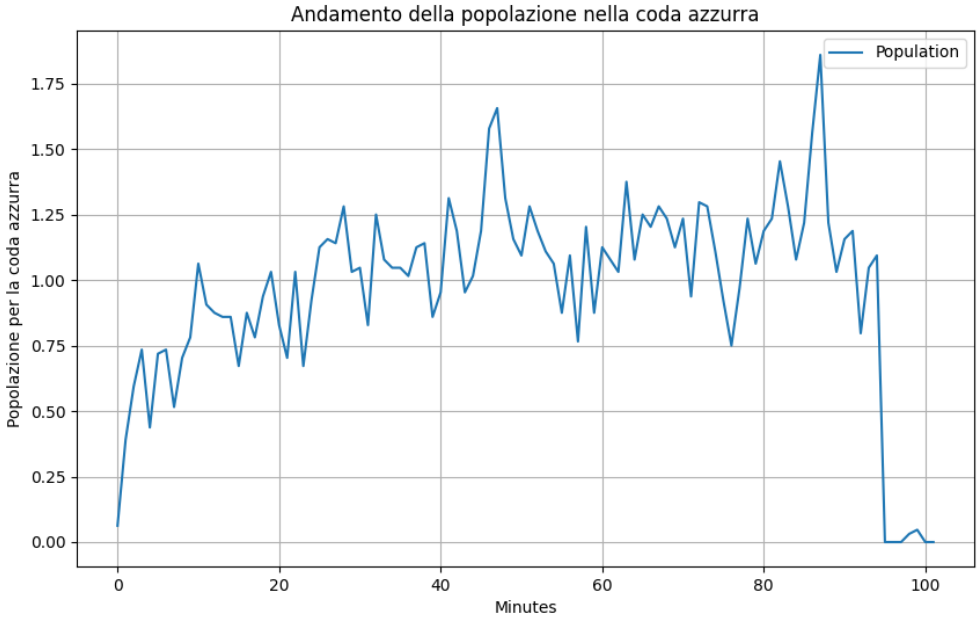

Figure 29: Popolazione media nella coda azzurra

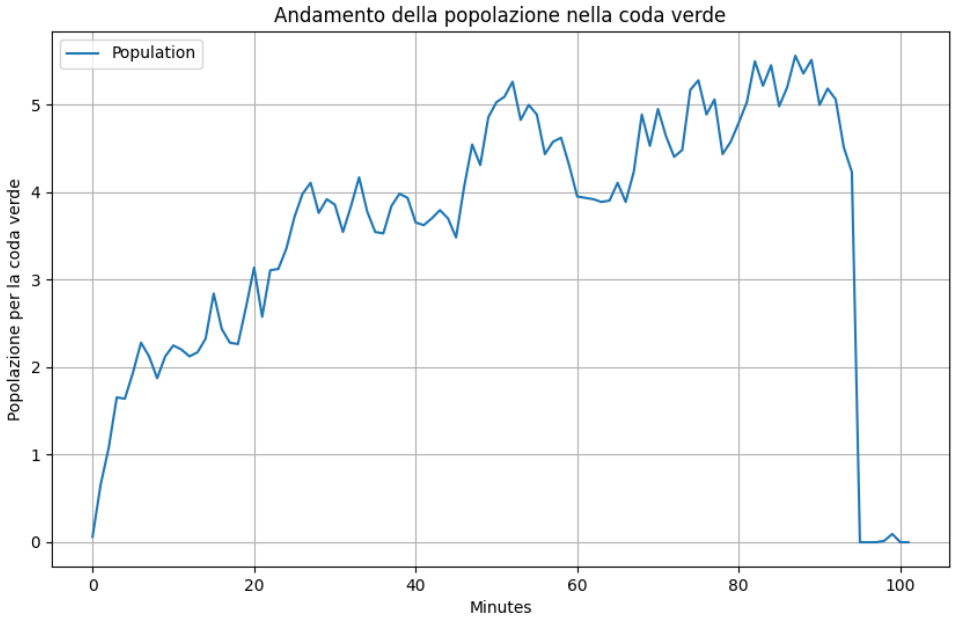

Figure 30: Popolazione media nella coda verde

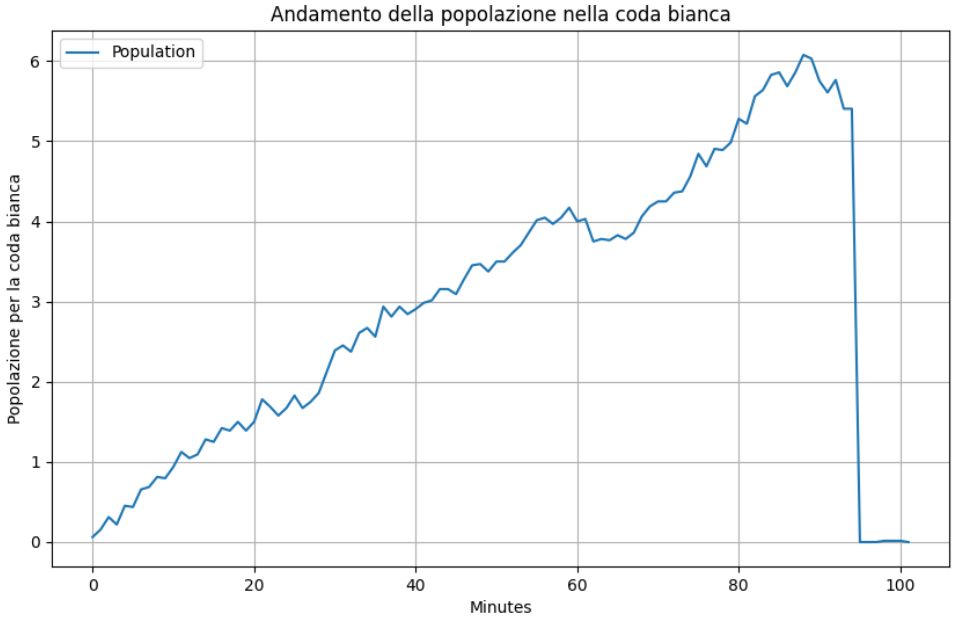

Figure 31: Popolazione media nella coda bianca

Simulazione ad orizzonte infinito

Analogamente alla simulazione ad orizzonte finito, abbiamo mantenuto la coerenza con il modello standard utilizzando gli stessi parametri b e k. Di seguito riportiamo i risultati ottenuti dalla simulazione:

```
Delay Time Metrics Queue:
    E[Tq] per coda: 0 = 0.001322 +/- 0.002663
    E[Tq] per coda: 1 = 0.000000 +/- 0.000000
    E[Tq] per coda: 2 = 4.979316 +/- 0.160357
    E[Tq] per coda: 3 = 12.289185 +/- 0.716587
    E[Tq] per coda: 4 = 34.208843 +/- 2.714771
    E[Tq] per coda: 5 = 153.482421 +/- 24.202679
    E[Tq] per coda: 6 = 1110.470555 +/- 618.225915

Response Time Metrics Queue:
    E[Ts] per coda: 0 = 16.605658 +/- 0.162528
    E[Ts] per coda: 1 = 16.603831 +/- 0.120830
    E[Ts] per coda: 2 = 21.725100 +/- 0.180167
    E[Ts] per coda: 3 = 28.877940 +/- 0.745776
    E[Ts] per coda: 4 = 51.222436 +/- 2.757246
    E[Ts] per coda: 5 = 173.502269 +/- 24.362014
    E[Ts] per coda: 6 = 1216.048868 +/- 623.254556

Utilization Metrics Queue:
    E[Rho] per server: 0 = 0.977889 +/- 0.005248
    E[Rho] per server: 1 = 0.971430 +/- 0.006829
    E[Rho] per server: 2 = 0.962358 +/- 0.008552
    E[Rho] per server: 3 = 0.946996 +/- 0.012416
```

Figure 32: Risultati simulazione infinita modello migliorativo

Di seguito invece riportiamo il numero di violazioni registrate in media durante la simulazione suddivise per codici:

```
Ci sono state  1 violazioni per il colore Rosso . La media per singola violazione è di:  3.215969157696236 .
Con una percentuale di job che violano il QoS di:  0.0004424778761061947

Ci sono state  1989 violazioni per il colore Arancione . La media per singola violazione è di:  15.05475728733163 .
Con una percentuale di job che violano il QoS di:  0.2747617074181517

Ci sono state  5955 violazioni per il colore Azzurro . La media per singola violazione è di:  31.68536830637251 .
Con una percentuale di job che violano il QoS di:  0.4186291739894552

Ci sono state  7078 violazioni per il colore Verde . La media per singola violazione è di:  170.1990235934102 .
Con una percentuale di job che violano il QoS di:  0.6273710335046977

Ci sono state  1224 violazioni per il colore Bianco . La media per singola violazione è di:  1778.0891712037887 .
Con una percentuale di job che violano il QoS di:  0.6351842241826674
```

Figure 33: Numero di violazioni nella simulazione ad orizzonte infinito

Riportiamo ora i grafici sviluppati durante la simulazione ad orizzonte infinito:

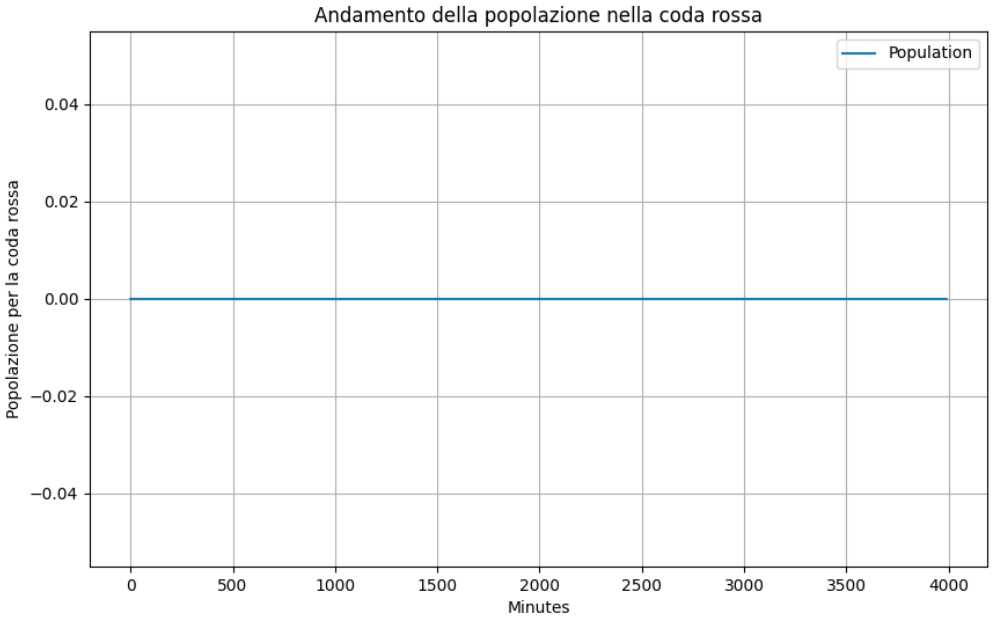

Figure 34: Popolazione media nella coda rossa

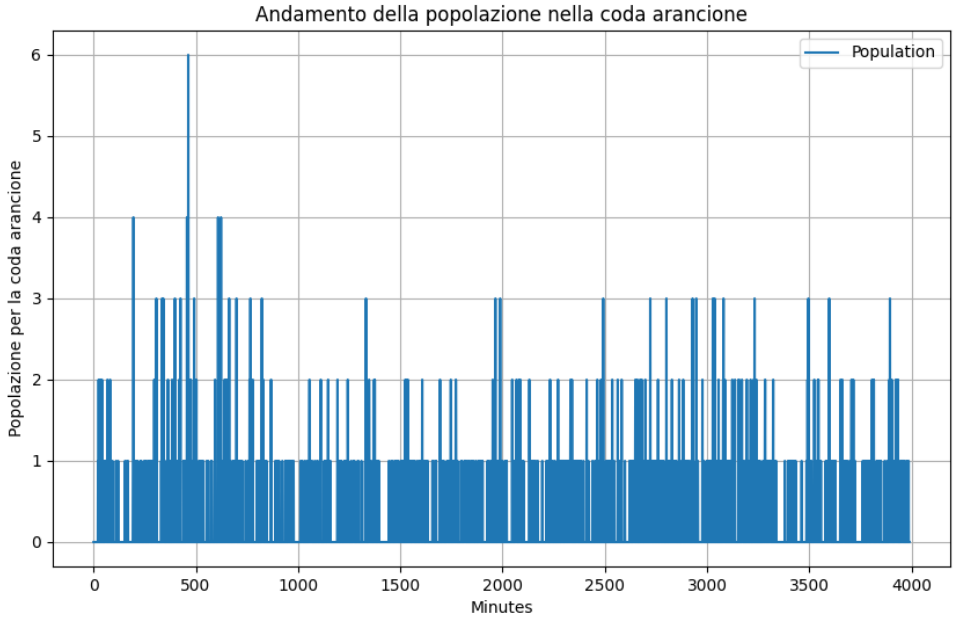

Figure 35: Popolazione media nella coda arancione

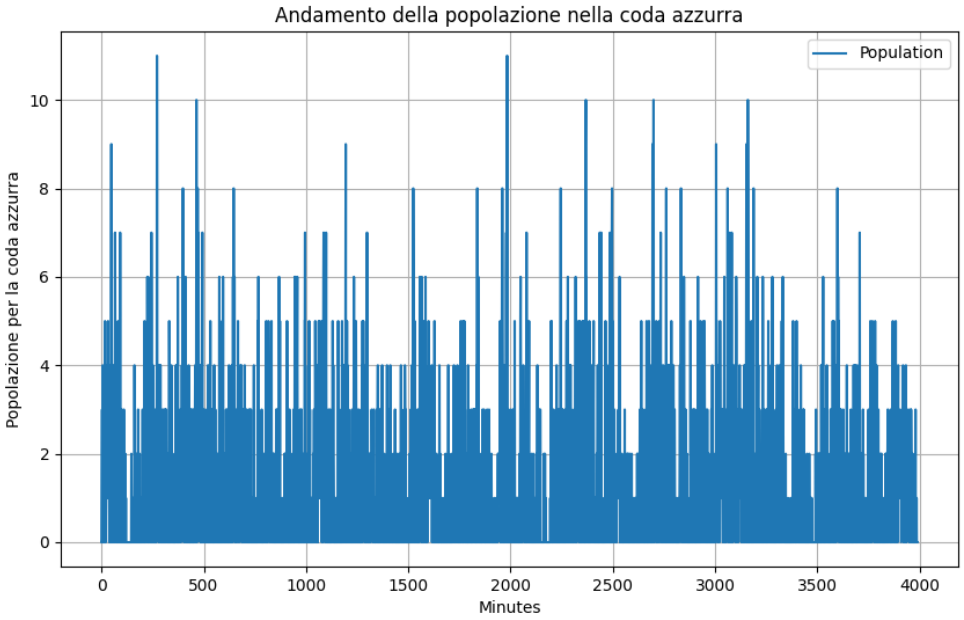

Figure 36: Popolazione media nella coda azzurra

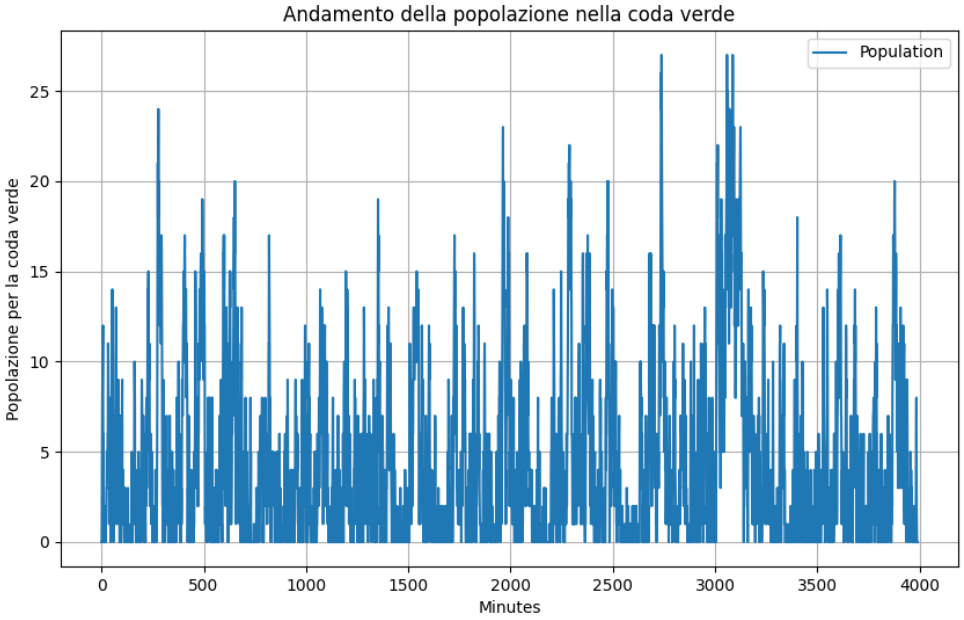

Figure 37: Popolazione media nella coda verde

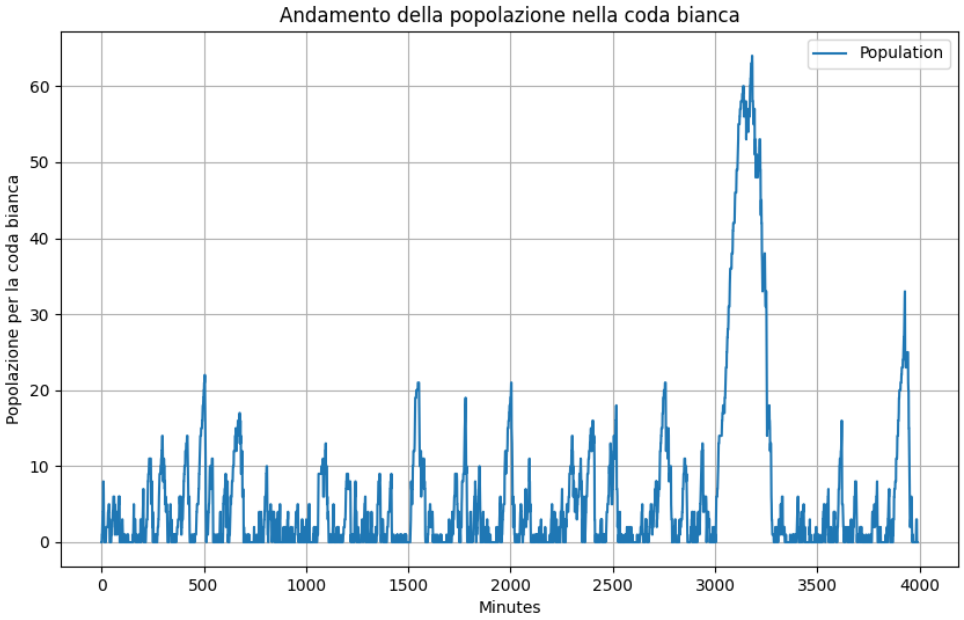

Figure 38: Popolazione media nella coda bianca

Risultati del modello migliorativo

Vediamo che sono stati registrati, utilizzando il modello migliorativo i seguenti miglioramenti in punti percentuali rispetto al modello standard:

- **Livello 2 (Arancione)**: 23.55%
- **Livello 3 (Azzurro)**: 5.4%
- **Livello 4 (Verde)**: 3.62
- **Livello 5 (Bianco)**: 4.13%

Questi miglioramenti dimostrano l'efficacia del modello migliorativo nell'ottimizzare la gestione delle risorse e nel ridurre le violazioni dei tempi di attesa. Sebbene le percentuali di miglioramento varino tra i diversi livelli di priorità, l'adozione del nuovo modello ha portato a una riduzione complessiva delle attese per tutti i codici di gravità, contribuendo a un sistema di pronto soccorso più efficiente e reattivo.

15. Considerazioni sul personale

Per completezza, abbiamo effettuato una simulazione ad orizzonte finito aumentando il numero di medici, e garantendo quindi un medico in più per ogni turno di lavoro.

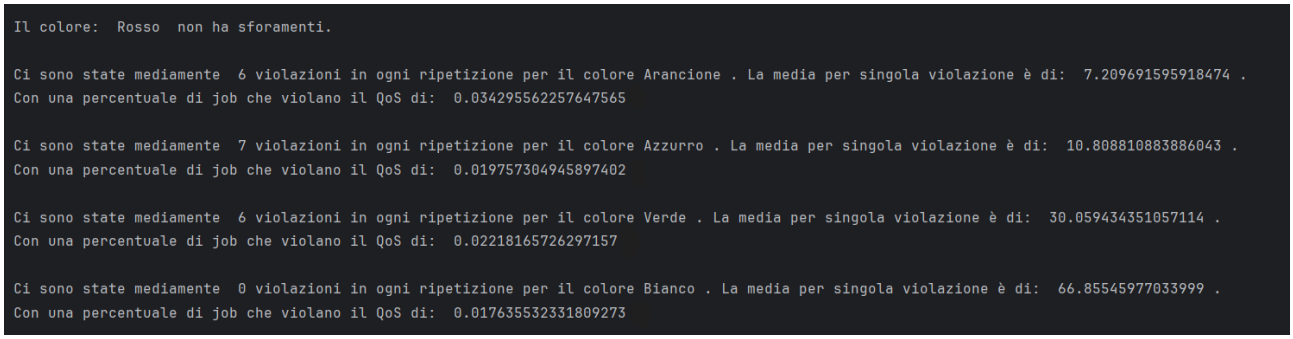

Figure 39: Numero di violazioni nella simulazione ad orizzonte infinito

Dai risultati della nostra simulazione, è emerso che il potenziamento del personale nel pronto soccorso ha avuto un impatto significativo su vari aspetti operativi e clinici. L'incremento del personale ha consentito di ridurre significativamente i tempi di attesa, migliorando l'accesso alle cure. Questi risultati confermano l'importanza di investire nel personale sanitario per garantire un servizio efficiente e di qualità.

16. Considerazioni sugli accessi al pronto soccorso

Recenti dichiarazioni, riportate dall'Ansa [5], riportano che la percentuale di accessi evitabili, identificati tramite i codici bianco e verde, supera il 40% del totale. In risposta a questi dati, abbiamo condotto una simulazione mirata a ridurre la frequenza di tali codici bianchi di circa il 50%. Questo approccio ci consente di valutare l'impatto di una significativa diminuzione degli accessi evitabili sui risultati complessivi e sull'efficienza del sistema sanitario. Riportiamo di seguito il numero di violazioni registrate durante questa simulazione ad orizzonte finito, effettuata utilizzando il modello migliorativo:

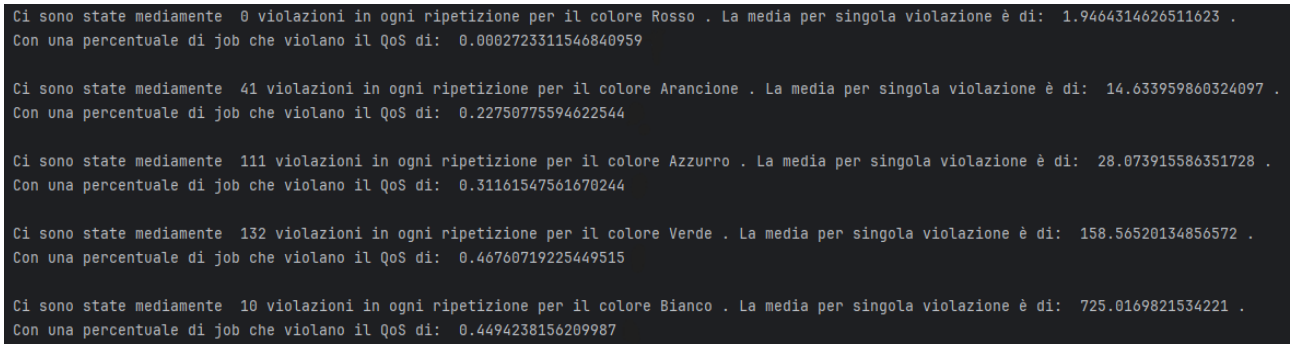

Figure 40: Violazioni con riduzione dei codici bianchi

Come vediamo dai risultati, la riduzione dei codici bianchi, porta ad una significativa diminuzione delle violazioni su tutti i codici.

17. Risultati e Confronto

Attraverso un'ottimizzazione del modello, concentrata sulla gestione dello scheduling nell'area di trattamento, siamo riusciti a ridurre significativamente le violazioni del limite imposto. Vediamo il confronto tra i risultati del modello standard, del modello migliorativo, del modello migliorativo con l'aggiunta di un medico e del modello migliorativo dopo aver dimezzato il numero di codici bianchi ottenuti nelle simulazioni ad orizzonte finito e riassunti nella tabella seguente:

Table 1: Numero di violazioni per colore delle code per i vari modelli

Colore	**Standard**	**Migliorativo**	**Aumento Personale**	**Bianchi dimezzati**
Rosso	0%	0%	0%	0%
Arancione	33.33%	25.49%	3.43%	22.75%
Azzurro	38.51%	36.43%	1.98%	31.16%
Verde	57.68%	55.59%	2.22%	46.76%
Bianco	59.30%	56.85%	1.76%	44.94%

I risultati evidenziano come il modello migliorativo, specialmente quando combinato con l'aumento di personale o la riduzione degli accessi non urgenti, porti a significative riduzioni delle violazioni dei tempi limite. Queste strategie di ottimizzazione dimostrano l'importanza di una gestione più efficiente delle risorse per migliorare le performance del pronto soccorso in modo sostenibile.

Bibliografia

[1] https://www.salute.gov.it/portale/prontoSoccorso/dettaglioContenutiProntoSoccorso.jsp?lingua=italiano&id=1190&area=118%20Pronto%20Soccorso&menu=vuoto&tab=1
[2] https://github.com/pdsteele/DES-Python/tree/master
[3] Leemis, Lawrence M., and Stephen Keith Park. Discrete-event simulation: A first course. Upper Saddle River: Pearson Prentice Hall, 2006.
[4] https://www.comune.roma.it/web-resources/cms/documents/04_SaluteSanita_Annuario_2023_AGGPDF_NON_A.pdf
[5] https://www.ansa.it/canale_saluteebenessere/notizie/sanita/2024/02/20/-evitabile-oltre-il-40-degli-accessi-al-pronto-soccorso_9247fe1e-fdbf-483c-9697-d7a5b07d5600.html