Progetto PMCSN

a.a.2021/22

|  |
| --- |
| Gestione di un cinema |

|  |  |  |
| --- | --- | --- |
| Brinati Anastasia  Appetito Giulio  Brinati Anastasia  BB |  |  |
| Università degli Studi di Roma Tor Vergata |  |  |

SOMMARIO

#### **MODELLO INIZIALE**

[0. **INTRODUZIONE** 3](#_Toc110092570)

[0.1 Descrizione del sistema 3](#_Toc110092571)

[0.2 Aspetti critici 3](#_Toc110092572)

[0.3 Conseguenze 3](#_Toc110092573)

[1. **OBIETTIVI** 5](#_Toc110092574)

[2.  **MODELLO CONCETTUALE** 6](#_Toc110092575)

[2.1 Politiche di scheduling nelle code 6](#_Toc110092577)

[2.2 Stato del sistema 7](#_Toc110092577)

[2.3 Eventi 7](#_Toc110092577)

[2.3.1 Descrizione degli eventi 7](#_Toc110092577)

[3.  **MODELLO DELLE SPECIFICHE** 9](#_Toc110092576)

[3.1 Dataset Considerati 9](#_Toc110092577)

[3.2 Valori Numerici 9](#_Toc110092577)

[4.  **MODELLO COMPUTAZIONALE** 8](#_Toc110092578)

[5. **VERIFICA** 9](#_Toc110092579)

[6. **VALIDAZIONE** 10](#_Toc110092580)

[7. **CONCLUSIONE** 11](#_Toc110092581)

[8. **MODELLO MIGLIORATIVO**  11](#_Toc110092581)

#### **MODELLO MIGLIORATIVO**

# introDUZIONE

## Descrizione del sistema

Il sistema considerato per il nostro studio è un cinema multisala. La scelta è stata dettata da esperienze personali, essendo il cinema un luogo da noi frequentato solitamente: questo ci ha dato diversi spunti per lo sviluppo del progetto, avendo toccato con mano le problematiche che una attività di questo tipo può presentare. Il riferimento scelto per il nostro lavoro è il cinema Ariston di Colleferro (RM), grazie al quale abbiamo avuto modo di ottenere dei dati tramite un’intervista al titolare dell’attività.

Gli utenti che accedono al sistema possono essere di due tipologie: *senza biglietto*, o *con biglietto online*.

* Gli utenti del primo tipo dovranno acquistare il biglietto presso uno dei due sportelli della biglietteria, presente all’esterno del cinema, a partire dai 60 minuti che precedono l’inizio della proiezione;
* Gli utenti con biglietto online non avranno necessità di accodarsi presso gli sportelli della biglietteria, avendo accesso diretto alla struttura.

Una volta ottenuto l’accesso alla struttura, gli utenti devono sottoporsi al *controllo dei biglietti*, che viene effettuato - da uno o più addetti - immediatamente dopo l’ingresso del cinema.

Successivamente, gli utenti che hanno superato il controllo dei biglietti hanno a disposizione dei servizi facoltativi prima di entrare all’interno della sala per la proiezione del film. In particolare, la struttura offre:

* un’*area food*, in cui i clienti hanno modo di acquistare cibi e bevande da consumare durante la proiezione;
* un’*area gadgets*, in cui i clienti hanno la possibilità di acquistare merchandising a tema.

Il cinema in questione offre tre diversi orari per la visione di film:

* *16:00 15-16-17*
* *20:00 19-20-21*
* *23:00*. 22-23-24

Inoltre, i film proiettati sono principalmente di 3 tipologie:

* *film per bambini / cartoni animati*
* *supereroi / azione*
* *horror.*

## 0.2 Aspetti critici

Gli aspetti critici individuati relativi ai clienti sono i seguenti:

* attesa dei clienti in coda per acquistare il biglietto fisicamente presso gli sportelli della biglietteria;
* attesa dei clienti in coda per la convalida del proprio biglietto presso l’entrata del cinema;
* attesa dei clienti in coda per essere serviti presso l’area food, e conseguente attesa per il pagamento presso le rispettive casse;
* attesa dei clienti in coda per il pagamento presso le casse dell’area gadgets.

## 0.3 Conseguenze

Relativamente ai profitti da parte del cinema, gli aspetti critici individuati nel punto precedente hanno le seguenti conseguenze:

* i clienti che attendono troppo per acquistare i biglietti presso la biglietteria potrebbero decidere di rinunciare allo spettacolo, rimandandone la visione o – nel peggiore dei casi – scegliere di vedere il film presso un cinema competitor, causando una riduzione del numero di biglietti venduti;
* un arco di tempo eccessivamente elevato per accedere alla sala impatterebbe i guadagni derivanti dagli sponsor pubblicitari i quali, proiettando i propri spot prima dell’inizio del film, hanno concordato con il cinema un pagamento proporzionale al numero di spettatori presenti in sala durante la proiezione dei suddetti spot.

# obiettivi

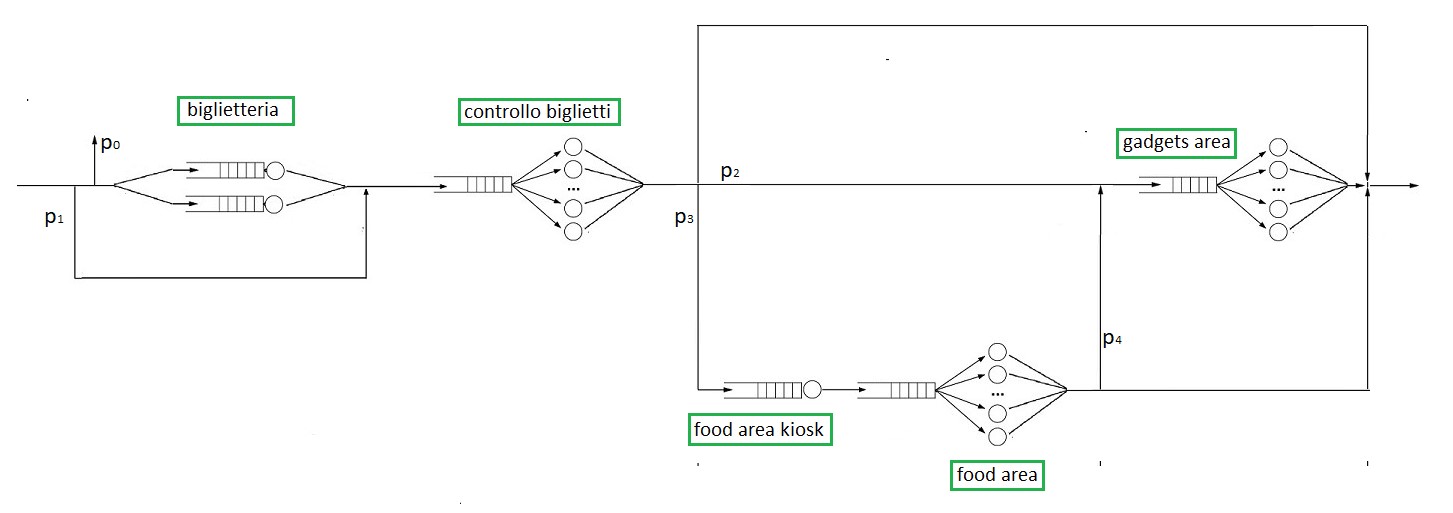
L’obiettivo del presente studio è quello di massimizzare i profitti da parte del cinema.

Pertanto, si vuole determinare la configurazione ottima del sistema, individuando il numero ottimale di serventi per ognuno dei sottosistemi nelle varie fasce orarie considerate.

Inoltre si vuole garantire i seguenti QoS (Quality of Service):

1. Il tempo di risposta complessivo del sistema deve essere mantenuto sotto i x minuti per qualsiasi fascia oraria, non considerando i tempi di percorrenza necessari per spostarsi da un centro all’altro;
2. Il 60% degli utenti deve assistere alla proiezione degli spot pubblicitari;
3. Il y% degli utenti acquistano il biglietto/non abbandonano la fila (biglietteria). #si ha che almeno una certa soglia/perc di clienti deve accedere necessariamente al sistema (-> minimizzare la prob perdita).

# modello concettuale





Il cinema è stato modellato con la rete riportata in figura. I sottosistemi di cui il sistema si compone sono i seguenti:

* *Biglietteria*: Il sottosistema relativo alla biglietteria è stato modellato (rispecchiando la realtà) mediante due **M/M/1** con code infinite, poichè l’esaurimento dei biglietti disponibili non è stato considerato un evento rilevante dal punto di vista probabilistico (relativamente al cinema considerato per il nostro studio);
* *Controllo biglietti:* il sottosistema relativo al controllo dei biglietti è stato modellato con una **M/M/K** con coda infinita, dal momento che è possibile avere più addetti al lavoro contemporaneamente;
* *Cassa Food area:*il sottosistema relativo alla cassa dell’area food è stata modellato, invece, con una **M/M/1** con coda infinita*,* rispettando la configurazione prevista dal cinema in esame;
* *Food area:*il sottosistema relativo all’area food è stato modellato mediante una **M/M/k** con coda infinita;
* *Gadgets area:*il sottosistema è stato modellato con una **M/M/K** con coda infinita.

## 2.1 Politiche di scheduling nelle code

Per le code in ciascuno dei sottosistemi considerati abbiamo ragionato seguendo il naturale andamento dei clienti, scegliendo di adottare per ciascuno una politica di scheduling **FIFO,** in quanto è ragionevole pensare che un utente presso uno qualsiasi dei centri venga servito in ordine di arrivo.

Inoltre, le politiche adottate sono **non-preemptive,** dal momento che, seguendo un ragionamento analogo al precedente, in nessuno dei sottosistemi ha senso interrompere il servizio di un utente per riprenderlo in un secondo momento, favorendo il servizio di un altro cliente.

Infine, si è prestata particolare attenzione all’instradamento che avviene in biglietteria: sono presenti due casse distinte, presso le quali gli utenti si indirizzano a seconda della meno affollata; abbiamo deciso di optare per una politica di routing ‘least work left’, di modo da assicurare che ogni cliente in arrivo si accodi presso il servente meno carico, seguendo la realtà dei fatti.

## Stato del sistema

Ad ogni istante di tempo il sistema viene descritto dalle seguenti variabili di stato:

* *Numero di utenti (per centro)*
* *Numero di utenti in servizio (per centro)*
* *Numero di utenti in coda (per centro)*
* *Stato di ogni servente xi ∈ [busy, idle]*
* *Stato di ogni coda yi ∈ [empty, not\_empty]*

## Eventi

Un cambiamento nello stato da parte del sistema può verificarsi a seguito di una serie di eventi. Gli eventi considerati sono di diverse tipologie:

1. Completamento del servizio di un utente/job;
2. Arrivo di un nuovo utente ad un centro;
3. Inizio proiezione spot pubblicitari;
4. Inizio proiezione film;
5. Apertura biglietteria;
6. Perdita di un utente in coda presso la biglietteria;

## Descrizione degli eventi

Ognuno dei precedenti eventi produce un cambiamento nello stato del sistema. Per ciascuno dei centri del sistema, si hanno situazioni in parte differenti.

* *Biglietteria*:
  + *Arrivo :* all’arrivo di un nuovo utente presso la biglietteria, questo deve accodarsi presso uno delle due code presenti, ciascuna per uno dei due serventi. Nel caso in cui dei due serventi uno solo sia attivo, la scelta risulta ovviamente obbligata; in caso contrario, la scelta verte sulla coda con il minimo traffico (secondo una regola ***LWL***). Nel caso in cui la coda scelta sia vuota (*yi == empty)*, l’utente viene servito presso lo sportello selezionato; in caso contrario (*yi == not\_empty)* l’utente dovrà attendere il servizio degli utenti in coda, senza la possibilità di cambiare coda dopo averne selezionata una.
  + *Completamento :* dopo aver ottenuto il servizio presso uno dei due sportelli, l’utente può lasciare il centro e il servente in questione passa dallo stato *busy* allo stato *idle*, procedendo a servire un nuovo utente nella propria coda (se la coda è *not\_empty)*.
* *Controllo biglietti:* 
  + *Arrivo :* all’arrivo di un utente presso il centro *Controllo biglietti* questo deve accodarsi presso l’unica coda a disposizione (nel caso in cui, ovviamente, la coda sia *not\_empty*). Nel caso in cui vi siano più operatori in servizio, l’utente di turno sceglierà il primo operatore libero.
  + *Completamento :* come nel caso precedente, dopo aver completato il servizio di un utente, il singolo operatore passa dallo stato *busy* allo stato *idle*, così da offrire di nuovo servizio agli eventuali altri utenti nella coda, mentre l’utente può scegliere se proseguire direttamente verso la sala cinematografica (concludendo di fatto la propria permanenza nel sistema), passare per l’area food o per l’area gadgets.
* *Cassa Area food:*
  + *Arrivo* : gli utenti che dopo aver completato il servizio scelgono di recarsi presso l’area food devono accodarsi dapprima presso l’unica coda a disposizione (nel caso in cui la coda sia *not\_empty*) in attesa di ricevere servizio.
  + *Completamento* : una volta arrivato il proprio turno, l’utente riceve servizio presso la cassa (che passa dallo stato *busy* allo stato *idle*), così da avere accesso all’area food.
* *Area food:*
  + *Arrivo :* una volta completato il servizio presso la cassa, gli utenti che accedono all’area food devono accodarsi ancora presso l’unica coda disponibile, in attesa che uno dei serventi a disposizione (se più di uno) sia nello stato *idle*. La selezione del servente avviene selezionando il primo operatore *idle*.
  + *Completamento :* una volta completato il servizio presso uno degli operatori quest’ultimo passa dallo stato *busy* allo stato *idle*, mentre l’utente servito ha la possibilità di scegliere tra una sosta presso l’*area gadgets* e la prosecuzione diretta verso la sala cinematografica, così concludendo il proprio percorso all’interno del sistema.
* *Area gadgets:*
  + *Arrivo :* gli utenti che scelgono di sostare presso l’area gadgets, una volta selezionati i propri prodotti, deve recarsi presso l’unica coda a disposizione (nel caso la coda sia *not\_empty*).
  + *Completamento :* Una volta giunto il proprio turno, l’utente riceve servizio presso uno degli operatori attivi (se più di uno). La selezione dell’operatore avviene, come prima, scegliendo il primo operatore *idle*. Completato il servizio, l’utente si dirige verso la sala cinematografica, concludendo così il proprio percorso.

## Profitto

Si intende ora descrivere ricavi e costi che determinano il profitto del cinema.

Il cinema Ariston di Colleferro prevede, per ciascuna fascia oraria, un numero fisso di 10 dipendenti in totale, che possono ricoprire le varie posizioni presso le varie postazioni (es. controllo biglietti, ..).

Ciascun dipendente lavora per turni della durata di 3h:

* 15:00-17:00
* 19:00-21:00
* 22:00-24:00

e percepisce in media 10 euro/h.

Da queste informazioni ricaviamo che la spesa del cinema per fascia oraria rispettivamente agli stipendi dei dipendenti è: 300euro.

Inoltre, in cinema offre il servizio di ristorazione, che dispone di bibite (porzione media 33cl) e popcorn (porzione media 70g), e che vende rispettivamente al prezzo medio di 2,5 euro e 3,5 euro (a porzione).

Infine, il biglietto per ciascuna proiezione ha il prezzo medio di 7euro.

Queste info sono da calcolare in base al numero di spettatori, possiamo fare però delle stime teoriche in base ai valori numerici ricavati nel modello delle specifiche.

e per i quali spende rispettivamente 160 euro (->120litri) e 90euro (->25kg), ad ogni fascia oraria.

[DUNQUE ad ogni fascia oraria ho una richiesta in media di 24,08kg di popcorn]

Da cui ricaviamo che la spesa media del cinema per l’area food è circa: 250 euro.

Dunque deduciamo che l’income del cinema proveniente dalle vendite è all’incirca: 2.064 euro

-> guadagno di : income-spesa = 2064-250-300

# 3. modello delle specifiche

## 3.0 Dataset considerati

I dati tenuti in considerazione per lo sviluppo del modello sono stati ottenuti tramite un’intervista al titolare del *cinema multisale Ariston di Colleferro (RM)*.

[…]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | N° medio di spettatori 1°fascia oraria | N° medio di spettatori 2°fascia oraria | N° medio di spettatori 3°fascia oraria | % utenti con biglietto online | % utenti che sosta presso l’area food | % utenti che sosta presso l’area gadget |
| **Cartoni animati** | 170 | 80 | 30 | 10% | 80% | 40% |
| **Horror** | 10 | 30 | 80 | 15% | 20% | 5% |
| **Supereroi** | 250 | 300 | 200 | 40% | 90% | 25% |

## 3.1 Valori numerici

I dati di input al modello sono: il tasso medio di arrivo da parte degli utenti al cinema, i vari tassi medi di servizio - per ogni singolo servente - nei diversi nodi del sistema e le probabilità di routing tra i sottosistemi.

## 3.1.1 Tasso medio di arrivo

Il *tasso medio di arrivo* è stato ottenuto a partire dai dati relativi al numero medio di spettatori per spettacolo, tenendo in considerazione le tre diverse tipologie di film offerte dal cinema in questione.

Dal momento che l’arco temporale in cui la biglietteria eroga i biglietti è pari ai 60 minuti che precedono l’inizio del film, abbiamo ottenuto le seguenti statistiche:

* Fascia 1 (15:00 – 16:00): ***λ*** = (170 + 10 + 250) / 60 = 7.166 arrivi/min
* Fascia 2 (19:00 – 20:00): ***λ*** = (80 + 30 + 300) / 60 = 6.833 arrivi/min
* Fascia 3 (22:00 – 23:00): ***λ*** = (30 + 80 + 200) / 60 = 5.166 arrivi/min

## 3.1.2 Tasso medio di servizio

I *tassi medi di servizio* considerati nello sviluppo del sistema sono gli stessi riportati dall’intervistato.

* *Biglietteria*: il tempo medio di servizio (a utente) è pari a 20 secondi. Dunque, si ottiene un tasso ***μbiglietteria*** = 1 utente /(1/3 min) = 3 utenti/min.
* *Controllo biglietti:* il tempo medio di servizio (per utente) per il controllo dei biglietti è pari a 15 secondi. Dunque, si ottiene un tasso ***μcontrollo*** = 1 utente/ (1/4 minuti) = 4 utenti/min.
* *Food area kiosk:* A partire da statistiche raccolte durante l’intervista, il tempo medio di servizio presso la cassa dell’area food è pari a 20 secondi per utente. Ne deriva un tasso medio di servizio pari a ***μkiosk*** = 3 utenti/min.
* *Food area:* sempre seguendo le informazioni ottenute dal direttore del cinema, il tempo medio di servizio per utente è pari a 30 secondi, ottenendo un tasso ***μfoodArea*** = 2 utenti/min.
* *Gadgets area:* infine, sempre dalle statistiche ottenute dall’intervista, si ha un tempo medio di servizio pari a 1 minuto, con tasso pari a ***μgadgetsArea*** = 1 utente/min.

## 3.1.3 Probabilità di routing

Per quanto riguarda le probabilità di routing, i valori seguono quanto riportato dall’intervistato. In particolare, dai dati raccolti si evince come le probabilità di routing variano a seconda della fascia oraria considerata. Questo è dovuto al fatto che – come spiegatoci dall’intervistato – ogni genere di film ha un proprio pubblico ben specifico, con caratteristiche differenti.

I valori percentuali delle persone che sostano presso l’area gadgets forniti dall’intervista erano relativi all’afflusso complessivo, motivo per il quale abbiamo scelto in seguito di suddividere il flusso in due porzioni, una relativa alle persone che sostano sia presso l’area food che l’area gadgets, ed una seconda relativa agli utenti che scelgono di sostare unicamente presso l’area gadgets. Dunque, a sostare presso l’area gadgets dopo aver concluso il servizio presso l’area food sono il 15% degli utenti per film di *supereroi*, il 20 degli utenti per i film *cartoni animati* e il 3% degli utenti per i film *horror;* conseguentemente, le restanti percentuali – rispettivamente, il 10%, 20% e 2% - degli utenti sosta presso l’area gadgets direttamente, senza passare per l’area food.7

Riassumendo per le tre diverse fasce orarie, i valori individuati sono i seguenti:

Fascia 1 (15:00 – 16:00)

* P1 = (0.1\*170 + 0.15\*10 + 0.4\*250) / (170 + 10 + 250) = 0.2755 = 27.55%
* P2 = (0.2\*170 + 0.02\*10 + 0.10\*250) / (170 + 10 + 250) = 0.1376 = 13.76%
* P3 = (0.8\*170 + 0.2\*10 + 0.9\*250) / (170 + 10 + 250) = 0.8441 = 84.41%
* P4 = (0.2\*170 + 0.03\*10 + 0.15\*250) / (170 + 10 + 250) = 0.1669 = 16.69%

Fascia 2 (19:00 – 20:00)

* P1 = (0.1\*80 + 0.15\*30 + 0.4 \*300) / (80 + 30 + 300) = 0.3231 = 32.31%
* P2 = (0. 2\*80 + 0. 02\*30 + 0. 10\*300) / (80 + 30 + 300) = 0.1136 = 11.36%
* P3 = (0.8\*80 + 0.2\*30 + 0.9\*300) / (80 + 30 + 300) = 0.8292 = 82.92%
* P4 = (0. 2\*80 + 0. 03\*30 + 0. 15\*300) / (80 + 30 + 300) = 0.1509 = 15.09%

Fascia 3 (22:00 – 23:00)

* P1 = (0.1\*30 + 0.15\*80 + 0.4\*200) / (30 + 80 + 200) = 0.3064 = 30.64%
* P2 = (0. 2\*30 + 0.02\*80 + 0. 10\*200) / (30 + 80 + 200) = 0.0890 = 8.90%
* P3 = (0.8\*30 + 0.2\*80 + 0.9\*200) / (30 + 80 + 200) = 0.7096 = 70.96%
* P4 = (0. 2\*30 + 0.03\*80 + 0. 15\*200) / (30 + 80 + 200) = 0.1238 = 12.38%

## 3.1.4 Costi proiezione spot pubblicitari

Inoltre, al fine di avere un’idea dei costi per la proiezione di spot pubblicitari in una sala cinematografica, abbiamo preso spunto dal sito web di un cinema di Pisa, in cui sono riportate le tariffe disponibili.

 “<http://pubblicitacinemapisa.com/?page_id=3025>”

Abbiamo deciso di scalare, in base ai dati del cinema in esame, i vari prezzi, andando quindi a pesare i guadagni provenienti dalle pubblicità in base al numero di spettatori in sala per una proiezione.

Poiché il proprietario del cinema ci ha fornito come unica informazione al riguardo il fatto che offrono un'unica opzione con prezzo fisso, per la proiezione di uno spot per due settimane, abbiamo deciso di utilizzare come riferimento il costo dell’abbonamento mensile riportato in figura.

* 330euro/2w -> 165euro/w -> 23.60 euro/gg -> 7.86 euro/proiezione=8$
* 15 000 spettatori/m -> 500 spettatori/gg

Dunque nel nostro caso, abbiamo deciso di riprodurre ogni spot tre volte al giorno, per 15 giorni, ad un pubblico di circa 500 spettatori per ogni proiezione, con un abbonamento dal costo di:

8$\*15gg = 360 euro.

# 4. modello computazionale

## 4.1 Intro

Per lo sviluppo del modello computazionale è stato scelto di utilizzare un linguaggio general purpose quale *C,* con il quale è stato scritto il codice per il simulatore.

Abbiamo iniziato sviluppando un simulatore in grado di completare una singola esecuzione e restituire, stampando su schermo, le statistiche ottenute.

Successivamente, il codice relativo al simulatore è stato inglobato in un programma in grado di effettuare più run consecutivamente. […]

## 4.2 Strutture dati utilizzate

*struct{*

*double current; /\* current time \*/*

*double next; /\* next (most imminent) event time \*/*

*double last; /\* last arrival time \*/*

*} t;*

**t.** Per tenere traccia del tempo simulato abbiamo usato questa struct *t* , che memorizza nei propri tre campi, rispettivamente, il tempo dell’evento corrente, del prossimo evento con il tempo più imminente e l’istante dell’ultimo evento. Ad ogni nuovo evento, le variabili di questa struct vengono man mano aggiornate. La simulazione, partendo dal valore *t.current* inizializzato con un valore specifico (0.0) arresta gli arrivi ad un certo istante di tempo (*close the door*), dopo il quale il programma prosegue fintanto che vi sono jobs da servire all’interno del sistema.

*typedef struct{*

*double node; /\* time integrated number in the node \*/*

*double queue; /\* time integrated number in the queue \*/*

*double service; /\* time integrated number in service \*/*

*double index; /\* used to count departed jobs \*/*

*double number; /\* number in the node \*/*

*double servers;*

*char \*name;*

*}center;*

**center.** Per modellare a livello computazionale ognuno dei diversi sottosistemi (centri) che compongono il sistema abbiamo introdotto la struct *center*, la quale permette di descrivere, istante per istante, lo stato di un centro.

I campi *node, queue* mantengono rispettivamente il numero di jobs presenti nel centro e in coda integrato nel tempo; il campo service, invece [???] ; il campo *index* viene utilizzato come contatore dei jobs serviti dal centro considerato; il campo number, invece, tiene traccia, istante per istante, dei jobs presenti all’interno del centro; infine, il campo *servers* memorizza il numero di serventi di cui il centro dispone (adeguatamente inizializzato in base ai dati di input al modello), mentre il campo *name* non ha un’utilità reale, ma viene utilizzata semplicemente per riferire un centro con il proprio nome.

Ad ogni occorrenza di un evento i campi di questa struct vengono ad essere modificati al fine di ottenere, una volta terminata la simulazione, le statistiche desiderate per ognuno dei vari centri. In particolare, ad ogni avanzamento del clock si va a sommare al valore corrente dei campi *node, queue* il valore, rispettivamente, delle persone nel centro e in coda moltiplicati per la differenza di tempo tra l’istante precedente e l’istante corrente.

*typedef struct serviceData{*

*double mean;*

*int stream;*

*}serviceData;*

**serviceData.** Questa struct è stata introdotta al fine di mantenere i parametri relativi ad ogni singolo servente di un dato centro. In particolare, il campo *mean* memorizza il tasso medio di servizio di un singolo server, mentre il campo *stream* memorizza l’indice dello stream utilizzato per la generazione di numeri pseudocasuali, necessario a generare i tempi di servizio.

*typedef struct event{ /\* the next-event \*/*

*double t; /\* next event time \*/*

*int x; /\* event status, 0 or 1 \*/*

*}event;*

**event.**La struct *event*  è stata utilizzata per tenere traccia dei vari tipi di eventi che possono modificare lo stato del sistema : durante l’esecuzione della simulazione viene allocato un array di *event*  di lunghezza pari al numero di tipologie di eventi considerato (nel nostro caso, pari a 12). Ognuno di questi *event* mantiene nel campo *t*  l’istante di tempo del prossimo evento, mentre il campo *x* funge da “semaforo” per un certo tipo di evento : la variabile viene settata ad *1* nel caso in cui un evento di questa tipologia sia disponibile per essere gestito, mentre viene settata a *0*  nel caso in cui non sia attivo nessun evento di questo tipo. Nello specifico, l’array utilizzato nella simulazione ordina nel seguente modo i tipi di evento possibili (ordinati per indice) :

1. Arrivo alla *biglietteria*
2. Completamento della *biglietteria* [0]
3. Completamento della *biglietteria* [1]
4. Arrivo al *controlloBiglietti* (proveniente dalla *biglietteria*[0])
5. Arrivo al *controlloBiglietti* (proveniente dalla *biglietteria*[1])
6. Completamento del *controlloBiglietti*
7. Arrivo alla *cassaFoodArea*
8. Completamento della *cassaFoodArea*
9. Arrivo alla *foodArea*
10. Completamento della *foodArea*
11. Arrivo alla *gadgetsArea*
12. Completamento della *gadgetsArea*

*typedef struct multiserver{*

*double service;*

*int served;*

*int occupied;*

*} multiserver;*

**multiserver.**Questa struct è risultata utile a modellare l’insieme di server di un centro multiservente. […]

## 4.3 Generazione di numeri pseudocasuali

Per la generazione di numeri pseudocasuali abbiamo fatto uso della librera *rngs.h* , la quale ci è stata utile in particolare delle funzioni *PlantSeeds(SEED),* necessaria ad inizializzare tutti gli streams, *SelectStream(stream)* per selezionare un particolare stream per un dato processo stocastico,[…].

Dal momento che sia i tempi di interarrivo che di servizio sono stati modellati con la distribuzione Esponenziale (ognuna con la rispettiva media) per la generazione dei suddetti tempi è stata utilizzata la funzione – sempre offerta dalla libreria *rngs.h* – *GetArrival()*, generando di volta in volta valori grazie alla funzione *Exponential()*.

Per quanto riguarda le probabilità di routing, invece, abbiamo utilizzato delle variabili con distribuzione uniforme, sfruttando dunque a tal proposito la funzione *Uniform(a,b)* (nel nostro caso scegliendo l’intervallo [0,1]).

Una volta individuati i vari processi stocastici in gioco nel sistema abbiamo proceduto a selezionare uno stream per ognuno di essi.

* STREAM\_ARRIVALS 0 : utilizzato per la generazione dei tempi di arrivo al sistema;
* STREAM\_BIGLIETTERIA0 1 : utilizzato per la generazione dei tempi di servizio della *biglietteria[0];*
* STREAM\_BIGLIETTERIA1 2 : utilizzato per la generazione dei tempi di servizio della *biglietteria[1];*
* STREAM\_CONTROLLOBIGLIETTI 3 : utilizzato per la generazione dei tempi di servizio del centro *controlloBiglietti;*
* STREAM\_CASSA\_FOOD\_AREA 4 : utilizzato per la generazione dei tempi di servizio del centro *cassaFoodArea;*
* STREAM\_FOOD\_AREA 5 : utilizzato per la generazione dei tempi di servizio del centro *foodArea;*
* STREAM\_GADGETS\_AREA 6 : utilizzato per la generazione dei tempi di servizio del centro *gadgetsArea;*
* STREAM\_ROUTING\_TICKETMODE 7 : utilizzato per la generazione della probabilità di routing relativa all’arrivo di utenti con biglietto *online* o *fisico;*
* STREAM\_ROUTING\_CONTROLLO 8 : utilizzato per la generazione della probabilità di routing relativa alla distinzione tra utenti che decidono di sostare presso la *foodArea,* presso la *gadgetsArea* o che decidono di proseguire direttamente, uscendo dal sistema*;*
* STREAM\_ROUTING\_GADGETS 9 : utilizzato per la generazione della probabilità di routing relativa alla distinzione tra utenti che decidono di sostare presso la *gadgetsArea* dopo la *foodArea* e quelli che decidono di proseguire, uscendo dal sistema.

# 5. verifica

Giunti a questo punto, è risultato necessario andare ad effettuare una verifica del simulatore sviluppato. Questa pratica consiste nell’attestare che il modello computazionale realizzato produca risultati corretti per il modello in esame: ci siamo preoccupati di confrontare i valori delle statistiche ottenuti dalle varie run di simulazione con i valori teorici provenienti dal modello analitico.

Per valutare il simulatore abbiamo scelto di considerare le statistiche dello stato stazionario, effettuando una simulazione ad orizzonte infinito con k=64 batch di dimensione b=1024 job, avendo come configurazione di serventi

{biglietteria, controllo biglietti, cassa area food, area food, gadgets area}={2, 2, 1, 4, 2}.

Tale scelta è legata al fatto per cui i valori teorici delle varie statistiche sono relative ad un sistema stazionario.

Per il nostro studio abbiamo considerato i seguenti punti per svolgere la verifica:

* Il tempo medio di risposta deve essere sempre uguale alla somma tra tempo medio d’attesa e tempo medio di servizio;
* Il numero medio di utenti in ogni centro deve essere sempre uguale alla somma tra numero medio di utenti in coda e numero medio di utenti in servizio;
* Confronto dei valori ottenuti dalla simulazione e valori teorici per ogni statistica ed ogni centro.

## 4.4.1 Biglietteria

Immagine che contiene testo

Descrizione generata automaticamente

job/min

2

3.537

=3.703 min

**Consistency check:**

* *min*

Check mark, Wingdings font, character code 252 decimal.

Check mark, Wingdings font, character code 252 decimal.

## 4.4.2 ControlloBiglietti

Immagine che contiene testo

Descrizione generata automaticamente

**Consistency check:**

Check mark, Wingdings font, character code 252 decimal.

Check mark, Wingdings font, character code 252 decimal.

## 4.4.3 CassaFoodArea

Immagine che contiene testo

Descrizione generata automaticamente

**Consistency check:**

Check mark, Wingdings font, character code 252 decimal.

Check mark, Wingdings font, character code 252 decimal.

## 4.4.4 FoodArea Immagine che contiene testo Descrizione generata automaticamente

**Consistency check:**

Check mark, Wingdings font, character code 252 decimal.

Check mark, Wingdings font, character code 252 decimal.

## 4.4.5 GadgetsArea

Immagine che contiene testo

Descrizione generata automaticamente

**Consistency check:**

Check mark, Wingdings font, character code 252 decimal.

Check mark, Wingdings font, character code 252 decimal.

# 6. validazione

Dal momento che non ci è stato possibile ottenere dati reali riguardo i vari tempi che si sperimentano all’interno del sistema reale con cui fare la validazione, abbiamo deciso di svolgere il processo di validazione creando dei casi di test, utili al fine di valutare se l’andamento del sistema sia affine alle aspettative. I casi di validazione considerati sono i seguenti:

1. Fissata una certa configurazione,ci aspettiamo che i tempi in coda aumentino in ogni centro all’aumentare del tasso medio di arrivo. In altre parole, la prima fascia avrà tempi in coda mediamente maggiori rispetto a quelli sperimentati nella seconda fascia oraria, mentre quest’ultima avrà mediamente tempi in coda maggiori rispetto alla terza fascia.

2. Fissato un tasso medio di arrivo (dunque, fissata una fascia oraria), ci aspettiamo che i tempi medi in coda diminuiscano all’aumentare del numero di serventi per ogni centro multiservente.

# 7. PROGettazione degli esperimenti

La progettazione degli esperimenti si articola di 3 passi principali:

1. Determinare il numero minimo di serventi per ogni centro;
2. Analisi del collo di bottiglia;
3. Simulazione ad orizzonte infinito;
4. Simulazione ad orizzonte finito.

## 7.1 Calcolo configurazioni minime

Il calcolo del numero minimo di serventi per ogni centro è necessario per garantire la condizione di stazionarietà del sistema. Avendo a disposizione i dati relativi al tasso medio di arrivo e di servizio di ogni centro, oltre che le varie probabilità di routing, abbiamo calcolato il numero necessario minimo di serventi per ogni centro seguendo la formula

.

Questo calcolo è stato reiterato per ognuna delle tre fasce orarie: la tabella seguente riporta il numero minimo di serventi per ogni coppia (centro, fascia oraria).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | biglietteria\_0 | controlloBiglietti | cassaFoodArea | foodArea | gadgetsArea |
| **Fascia oraria 1** | 1 | 2 | 1 | 4 | 2 |
| **Fascia oraria 2** | 1 | 2 | 1 | 3 | 2 |
| **Fascia oraria 3** | 1 | 1 | 1 | 2 | 1 |

## 7.1 Analisi del collo di bottiglia

Facendo uso dell’analisi operazionale, abbiamo svolto l’analisi del collo di bottiglia, considerando la configurazione e le probabilità di routing della prima fascia.

**Servizi medi dei centri**

**Calcolo numero medio di visite ai centri**

Da qui si ottiene:

**Calcolo domanda media per ogni centro**

= 0.083

=0.119

=0.105

=0.0975

## Risulta pertanto che il collo di bottiglia sia il centro cassaFoodArea, tuttavia nella realtà del sistema questo sottosistema non è modificabile a piacimento o a seconda della fascia oraria; infatti, il cinema ha la possibilità di gestire il numero di serventi delle tre aree indicate come multiserver, mentre la cassaFoodArea ha una conformazione ‘costringente’ ad un solo operatore per volta.

## Per questo motivo abbiamo deciso di intraprendere gli esperimenti seguenti a partire dalla foodArea.

## 7.1 Simulazione ad orizzonte infinito

A partire dalle configurazioni minime del sistema per ogni fascia oraria, abbiamo utilizzato la simulazione ad orizzonte infinito al fine di determinare la configurazione ottima del sistema: in questo ambito, per configurazione ottima intendiamo quella che garantisce il soddisfacimento del QOS relativo al tempo di riposta.

La scelta della simulazione ad orizzonte infinito per questo fine deriva dal fatto che, sebbene in una situazione reale la finestra di tempo di operatività del sistema risulti relativamente breve, è possibile eliminare il bias dello stato iniziale del sistema. Realisticamente questo è legato al fatto per cui, nel momento in cui gli sportelli della biglietteria aprono, non è sempre vero che il sistema sia vuoto.

Dunque, gli esperimenti in questo ambito hanno l’obiettivo di testare varie configurazioni del sistema, così da determinarne la migliore.

## 7.1 Simulazione ad orizzonte finito

Poiché le configurazioni ottenute dalla simulazione precedente prendono in considerazione solamente il QoS relativo al tempo di risposta, non è detto che siano quelle che forniscono il maggior profitto da parte del cinema.

Dunque, considerando il reale tempo di operatività della biglietteria pari a 60 minuti, consideriamo la variabile legata al profitto del cinema: in particolare, dal momento che i guadagni relativi alla proiezione di spot cinematografici sono direttamente proporzionali al numero di persone che assistono alla proiezione, potrebbe essere vantaggioso modificare la configurazione del sistema in questo senso

# 8. ESECUZIONE DELLE SIMULAZIONI

## 8.1 Simulazione ad orizzonte infinito

Nella simulazione ad orizzonte infinito il sistema viene simulato per un tempo ‘infinito’, ovvero considerando un arco temporale di gran lunga superiore alle tempistiche di una qualsiasi situazione reale in cui il sistema si può trovare.

Le statistiche ottenute da questo tipo di simulazione sono quelle dello stato stazionario del sistema.

Effettuando una simulazione per un tempo molto più lungo di quello ‘realistico’ di una fascia oraria (circa un’ora), si riduce il bias dello stato iniziale, la cui scelta diventa appunto sempre meno rilevante al crescere del tempo di simulazione; ‘il sistema perde memoria del proprio stato iniziale’.

Per ottenere la media campionaria del tempo di risposta globale abbiamo utilizzato il metodo *Batch Means*, suddividendo la run di simulazione in *k* batches di size *b* (numero di job). È previsto che ogni centro del sistema raggiunga k batches, e la simulazione terminerà solamente quando questo sarà vero per tutti i centri.

Per ogni metrica di interesse viene calcolata una media per ciascuna batch, per ciascun centro; una volta raggiunto il numero indicato di batches, si termina la raccolta delle stime di un determinato centro e viene calcolato l’intervallo (?) proprio a partire dalle batch means computate.

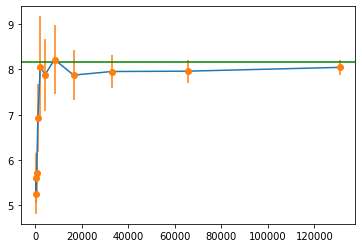
In questo caso il bias dello stato iniziale viene eliminato, poichè le statistiche per ogni batch sono calcolate di volta in volta a partire da uno stato iniziale differente, ovvero come è stato ‘lasciato’ il sistema dal precedente batch; al termine di ciascuna batch le statistiche di ogni centro vengono salvate in strutture dati apposite, quali nel nostro caso una matrice di struct output che tiene conto del centro osservato e della batch in questione, per essere poi resettate di modo da prepararle al calcolo delle statistiche della batch successiva.

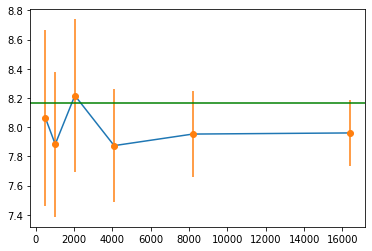
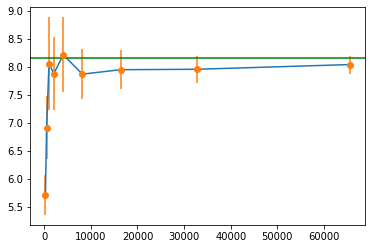
Questo metodo riduce inoltre l’autocorrelazione fra i dati, dato che l’unico punto di contatto fra le varie osservazioni è che lo stato finale in cui si trova il sistema alla fine di una singola batch, è lo stato iniziale del sistema per la successiva.

La scelta dei parametri (b, k) è avvenuta seguendo quanto suggerito dal libro di testo, facendo riferimento a Batch Means come un algoritmo “black box”: fissando il numero di batch *k* = 64, e testando con valori incrementali di *b* fintanto che viene raggiunta la “convergenza”, duplicando di volta in volta i valori di *b*.

Di seguito sono riportati i grafici che mostrano l’andamento del tempo di risposta dell’intero sistema al variare dei parametri (b,k). Abbiamo deciso di valutare la bontà della scelta in base a questa metrica poiché è tra gli obiettivi principali dello studio quello di ridurre il tempo di risposta complessivo, pertanto una scelta dei parametri che ci garantisse un valore vicino a quello teorico ci è sembrata la miglior scelta.

In ordine, compaiono i grafici relativi ad una scelta di k (numero di batch) pari a **64**, **128** e **256**. Sull’asse delle ascisse compare la taglia delle batch (b), mentre la linea in verde riporta il valore teorico calcolato per il tempo di risposta complessivo. Abbiamo deciso di iniziare con un valore di b pari a 16.





## 8.2 Simulazione ad orizzonte finito

Per la simulazione ad orizzonte finito è stato considerato il tempo di simulazione limitato a 60 minuti (*close the door*), riflettendo come nella realtà l’intervallo di tempo in cui c’è un flusso di arrivi al sistema.

Poiché le tre fasce orarie considerate non sono consecutive nel tempo, abbiamo deciso di effettuare ogni simulazione in riferimento ad una fascia oraria solamente.

* Computazionale: il sistema simulato permette di scegliere in quale fascia oraria intraprendere la simulazione, in quanto cambieranno diversi parametri (es: *lambda*, *p\_routing*);

Per la simulazione ad orizzonte finito la scelta dello stato iniziale del sistema ha un impatto importante sulle statistiche, nel nostro caso abbiamo fatto in modo che il sistema fosse *idle* sia all’inizio che alla fine della simulazione.

Allo scopo di effettuare un’analisi statistica dei valori ottenuti abbiamo operato replicando la simulazione per un numero pari a 128 volte, cambiando soltanto lo stato iniziale del generatore di numeri pseudocasuali. Ogni *replica* è volta a misurare indipendentemente le stesse metriche, fornendo un punto di stima (*datapoint*) utile al fine di calcolare gli intervalli di confidenza per ognuna di esse. Dunque, l’initial seed deve essere scelto in modo tale da non avere nessuna sovrapposizione fra repliche nella sequenza di numeri random utilizzata: a tale scopo, abbiamo utilizzato la tecnica standard che prevede di utilizzare lo stato finale di ogni stream del generatore per una replica come stato iniziale per la prossima. Questo è ottenuto effettuando la chiamata alla funzione *PlantSeeds(SEED)* una sola volta al di fuori del ciclo di replicazione.

Per memorizzare i dati relativi alle varie repliche abbiamo utilizzato la struct *outputStats:* nello specifico, abbiamo creato una struct di questo tipo per ogni centro del sistema e per ogni replica, andando successivamente a memorizzarla all’interno di una matrice di *outputStats* (*matrix*), in cui la riga i-ima corrisponde alla replica i-esima e la colonna j-ima corrisponde al centro j-imo.

Dopo aver eseguito la totalità delle repliche, la matrice è stata utilizzata – con l’ausilio del programma *estimate.c* – per calcolare gli intervalli di confidenza per ogni centro e per ogni statistica di interesse.

# 9. CONCLUSIONI

# ALGORITMO MIGLIORATIVO