

PMCSN Report

Marco Marcucci

marco.marcucci96@gmail.com
0286352

Giuseppe Lasco

giuseppe.lasco17@gmail.com
0286045

Valentina Valentina Francesca falaschi

valentinayffalas@gmail.com
0295947

University of Roma Tor Vergata — December 30, 2021

Introduzione

Lo studio ha lo scopo di analizzare due sistemi a code, stocastici e dinamici, che simulano lo scenario di una sala giochi. Il primo caso di studio si riferisce ad un modello base descrivendone il comportamento ed i limiti. Il secondo caso di studio propone un modello avanzato, che ha lo scopo di enfatizzare le differenze e i miglioramenti rispetto al primo caso.

L'intera analisi viene condotta procedendo attraverso i passi di modellazione, simulazione ed analisi delle statistiche di output.

1 Modello Base

Il modello base si riferisce al comportamento di una sala giochi, al cui interno sono presenti dei videogiochi (arcade). L'accesso alla struttura può avvenire in due modi:

- Attraverso l'acquisto di un biglietto online, esibendo al momento della prenotazione il proprio green-pass, saltando così i controlli;
- Attraverso l'acquisto di un biglietto in loco, sottponendosi al controllo del proprio green-pass oppure, in mancanza di quest'ultimo, al test antigenico rapido.

L'uscita dalla struttura può avvenire in tre modi:

- In seguito al riscontro della positività al test antigenico rapido;
- In seguito al riscontro di un green-pass non idoneo;
- In seguito al completamento della partita.

Nei primi due casi, il giocatore esce dal sistema senza pagare il biglietto. Inoltre, la politica aziendale della sala giochi prevede che il giocatore abbia diritto ad un rimborso del biglietto proporzionale al tempo di attesa che ha sperimentato.

Inserire figura 1 (alto livello)

1.1 Obiettivi

Controllare tutti i parametri definiti

Il primo obiettivo dello studio è quello di massimizzare i profitti della sala giochi. A tale scopo è stata definita una funzione guadagno:

$$G = costo biglietto * \#giocatori - costo elettricità * \#arcade attivi - \\ - \#giocatori * rimborso biglietto(attesa media nella coda) * costo biglietto \quad (1)$$

Il costo del biglietto è di 10 €. Per quanto riguarda l'elettricità è stata considerata una spesa mensile per Arcade pari a 200 €. La funzione *rimborso biglietto*, che calcola la percentuale del costo biglietto da restituire al giocatore, è proporzionale al tempo di attesa speso da quest'ultimo in coda. La massima percentuale rimborsabile è del 75%. L'attesa minima in coda per cui è previsto il rimborso è di 10 minuti.

Il secondo obiettivo dello studio è quello di assicurare un tempo medio di risposta del sistema inferiore a 40 minuti. Infatti, da un calcolo teorico basato sulle caratteristiche del sistema, specificate in seguito, si è dedotto che il tempo medio dei servizi del sistema è di circa 19 minuti. Per cui, è stata impostata una tolleranza del tempo medio di attesa nelle code sperimentato dall'utente, pari a circa il tempo di servizio.

1.2 Modello Concettuale

Inserire figura 2 (basso livello)

La sala giochi è modellata attraverso una rete aperta composta da N nodi rappresentanti i videogiochi e da un nodo rappresentante il controllo Covid-19. Ogni nodo si compone di un servente e di una coda infinita, poiché tale configurazione è la più verosimile.

La struttura è aperta H24 e la giornata è suddivisa in 4 fasce orarie caratterizzate da tassi di arrivo differenti:

- 08:00-12:00
- 12:00-17:00
- 17:00-22:00
- 22:00-08:00

Le variabili di stato del sistema considerate sono:

- Numero di clienti in ogni nodo;
- Numero di nodi attivi;
- Tipologia di cliente.

Le code inizialmente vengono considerate vuote, conseguentemente il numero di clienti ad ogni nodo è 0. La tipologia di ogni cliente (Green-pass, tampone o biglietto online) è decisa aleatoriamente al suo arrivo. I nodi per ogni fase vengono definiti aperti o chiusi in base alla configurazione scelta durante lo studio, che può essere ottima o non ottima.

Gli eventi considerati sono:

- Arrivo di un cliente;
- Completamento di un servizio;
- Cambio di fascia oraria.

Tali eventi possono causare cambiamenti dello stato. L'arrivo di un cliente causa l'aumento di clienti in un dato nodo e l'aumento del numero di clienti del sistema. Il completamento di un servizio causa il passaggio di un cliente da un nodo ad un altro o la sua uscita dal sistema. Il cambio di fascia oraria può causare l'accensione o lo spegnimento degli arcade. Se un nodo arcade viene spento quando ha ancora clienti in servizio o in coda, questo continuerà a processarli, pur non ricevendo altri arrivi. Il tempo in cui un nodo arcade è idle, ma acceso, viene considerato nel calcolo dei costi.

Implementare transient_simulation_income.py

1.3 Modello delle Specifiche

Controllare tutti i parametri definiti

Gli arrivi sono modellati con un processo di Poisson, con media variabile a seconda della fascia oraria. In particolare, la media per la prima fascia oraria è di $0.0\bar{6}$ minuti (15 clienti/minuto), la media per la seconda fascia oraria è di 0.2 minuti (5 clienti/minuto), la media per la terza fascia oraria è di $0.0\bar{6}$ minuti (15 clienti/minuto), la media per la quarta fascia oraria è di $0.0\bar{2}$ minuti (45 clienti/minuto).

La percentuale di clienti con biglietto online è del 20%. La percentuale di clienti che si sottopone al controllo del Green-pass è del 0.48%. La percentuale di clienti che si sottopone al tampone è del 32%.

La distribuzione del tempo di servizio per il controllo del green-pass è stata modellata come una distribuzione Normale con media 2 minuti (0.5 passeggeri/minuto) e varianza 1.5 minuti, troncata tra 1 e 3 minuti.

La distribuzione del tempo di servizio per il tampone è stata modellata come una distribuzione Normale con media 10 minuti (0.1 passeggeri/minuto) e varianza 1.5 minuti, troncata tra 8 e 12 minuti.

È stata scelta una Normale troncata con questi valori poiché permette di modellare un'operazione con bassa variabilità e di scartare valori eccessivamente piccoli o grandi che non rispecchierebbero tempi di servizio reali.

La distribuzione del tempo di servizio per un generico arcade è stata modellata come una distribuzione Normale con media $0.0\bar{6}$ minuti (0.1 passeggeri/minuto) e varianza 3 minuti, troncata tra 3 e 25 minuti.

La percentuale di clienti che risulta positiva al Covid-19 oppure presenta un green-pass non idoneo è in totale del 5%.

Le probabilità per la scelta del node arcade è stata modellata tramite una distribuzione Uniforme in modo che ciascun servente abbia la stessa probabilità di essere selezionato. Questa scelta permette di mantenere circa la stessa utilizzazione per tutti i serventi dato che hanno lo stesso tasso di servizio.

Si considera un numero di nodi per il controllo Covid-19 pari a 1 ed un numero massimo di nodi arcade pari a 20.

1.4 Modello Computazionale

L'approccio di simulazione utilizzato è stato quello Next-Event Simulation. Il linguaggio di programmazione utilizzato è *Python v.3.8*. Il codice di base per la simulazione segue quanto descritto nell'algoritmo 1. Quest'ultimo si avvale di ulteriori strutture dati e funzioni necessarie al corretto funzionamento.

Algorithm 1 M/G/1 - Rete Aperta - Scheduling FIFO

```

1: procedure MODELLO BASE
2:   Inizializzazione clock
3:   Inizializzazione lista dei nodi
4:   Inizializzazione struttura Integrali
5:   Selezione nodo su cui generare il primo arrivo
6:   Generazione primo arrivo
7:   while ultimo arrivo < STOP or ci sono job nel sistema do
8:     Scelta del prossimo evento
9:     for  $i \leftarrow 0$ , numero nodi do
10:    if nodi non vuoti then
11:      Aggiornamento statistiche
12:      Avanzamento del clock
13:      if Evento = Arrivo then                                 $\triangleright$  Arrivo
14:        if Evento = Cambio fascia oraria then
15:          Selezione tasso arrivi corretto
16:          Aggiornamento stato nodo
17:          Aggiornamento stato sistema
18:          Schedulazione prossimo arrivo
19:          if Jobs nel centro = 1 then
20:            Schedulazione prossimo servizio
21:          else                                                  $\triangleright$  Completamento
22:            Aggiornamento stato nodo
23:            if Nodo Arcade then
24:              Aggiornamento stato sistema
25:            if Jobs nel centro > 0 then
26:              Schedulazione prossimo servizio
27:            if Nodo Covid-Check then
28:              if Covid test negativo and Green-pass valido then
29:                Aggiornamento stato sistema

```

1.5 Verifica

Le strutture e le funzioni precedentemente accennate sono state sottoposte ad una fase di testing in modo da garantire il loro corretto funzionamento. Durante l'implementazione del progetto, l'applicazione iterativa di tale fase ha permesso di individuare e correggere diversi difetti, emersi a seguito del manifestarsi di malfunzionamenti del sistema.

1.6 Validazione

Al fine di validare il modello implementato, sono state eseguite diversi run variando determinati parametri. I risultati possono essere osservati in tabella 1 e 2.

Table 1: System statistics

# nodi arcades	avg interarrival time	avg wait	avg # node
2	15.042965	26.475121	1.759960
5	15.042963	21.542646	1.432069
9	15.042963	20.694456	1.375685

* seed = 1234567891, $\lambda_{system} = \frac{1}{15}$, #jobs = 482453,

Per quanto riguarda la tabella 1, è possibile evincere come il tempo di interarrivo medio del sistema converga al parametro impostato λ_{system}^{-1} .

Table 2: Arcade statistics

# nodi arcades	avg interarrival time	avg wait	avg delay	avg # node	avg # queue	utilization
2	31.347341	21.898857	6.903420	0.698586	0.2202234	0.478363
5	78.255828	16.770644	1.788041	0.214305	0.022849	0.191456
9	141.269751	15.911570	0.900827	0.112629	0.006376	0.106252

* seed = 1234567891, $\lambda_{system} = \frac{1}{15}$, #jobs = 482453,

Per quanto riguarda la tabella 2, si può notare come la relazione di Little sia soddisfatta e come l'utilizzazione e il tempo di interarrivo medio, rispettivamente, decresca ed aumenti al crescere del numero del numero degli arcade. Tale comportamento è giustificato dal fatto che i jobs vengono distribuiti in modo equiprobabile tra i diversi nodi arcade.

Come è possibile osservare dalle tabelle, inoltre, all'aumentare dei nodi nel sistema i tempi di attesa e il numero di jobs nei rispettivi centri diminuisce coerentemente a ciò che ci si aspetta.

1.7 Esperimenti di Simulazione

Al fine di raggiungere gli obiettivi prefissati, è utile dedurre il minimo numero di nodi che permette al sistema di raggiungere la stazionarietà e, in seguito, ricavare la configurazione ottimale del sistema.

Il raggiungimento della stazionarietà o meno viene studiato attraverso l'impiego dell'analisi a orizzonte finito, mentre quella a orizzonte infinito permette di confrontare le configurazioni del sistema a partire dai risultati ottenuti dal comportamento transiente.

1.7.1 Analisi del comportamento transiente

Attraverso l'analisi del comportamento transiente viene studiato il tempo medio di risposta del sistema, per ogni fascia oraria, utilizzando il metodo delle repliche. In particolare, vengono effettuate run di simulazione consecutive, utilizzando ogni volta come punto di partenza lo stato del sistema raggiunto dalla replica precedente. Il seed, infatti, viene settato una volta, all'inizio della prima replica. La tecnica delle repliche permette di calcolare stime puntuali e intervallari ad ogni acquisizione, ognuna di queste corrispondente ad un certo numero di richieste processate.

Una prima analisi ha portato ad osservare che il sistema non può raggiungere la stazionarietà per via dei limiti introdotti dalla coda del controllo Covid-19, essendo unica per modellazione. Infatti, per un $\lambda_{arrival} \gtrsim \frac{1}{4.1}$, l'utilizzazione di tale centro è approssimativamente pari ad 1, mentre quella dei centri Arcade risulta essere inferiore. Questo studio permette di concludere che il collo di bottiglia è rappresentato dalla coda del controllo Covid-19, per questo motivo, segue solamente lo studio dei nodi Arcade, considerando un $\lambda_{arrival} < \frac{1}{4.1}$.

Per quanto riguarda la prima fascia e la terza fascia oraria, rispettivamente (08:00-12:00) e (17:00-22:00), è possibile osservare i risultati dell'analisi in figura 1 e figura 2. Tali fasce, infatti, hanno lo stesso $\lambda_{arrival}$ da modellazione.

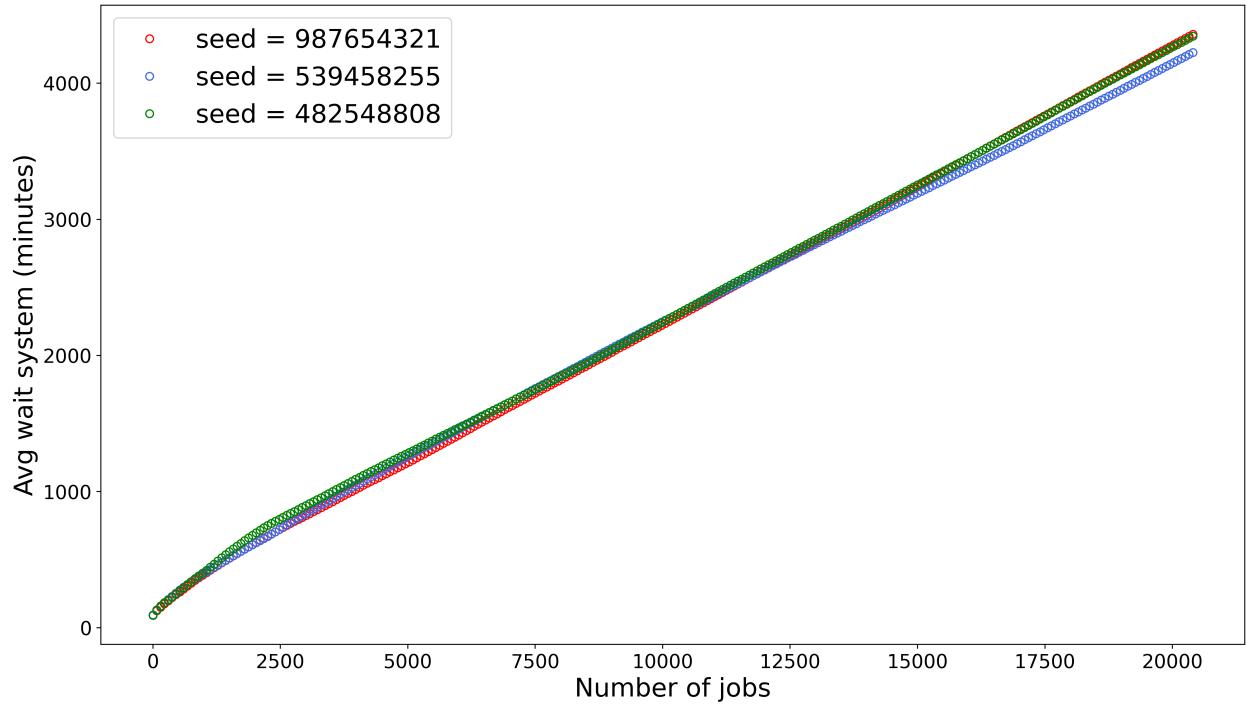


Figure 1: Average Wait System, 08:00-12:00 & 17:00-22:00, $\lambda_{arrival} = \frac{1}{14} \frac{job}{min}$, Arcades: 1, repliche=64, Sampling_freq=75

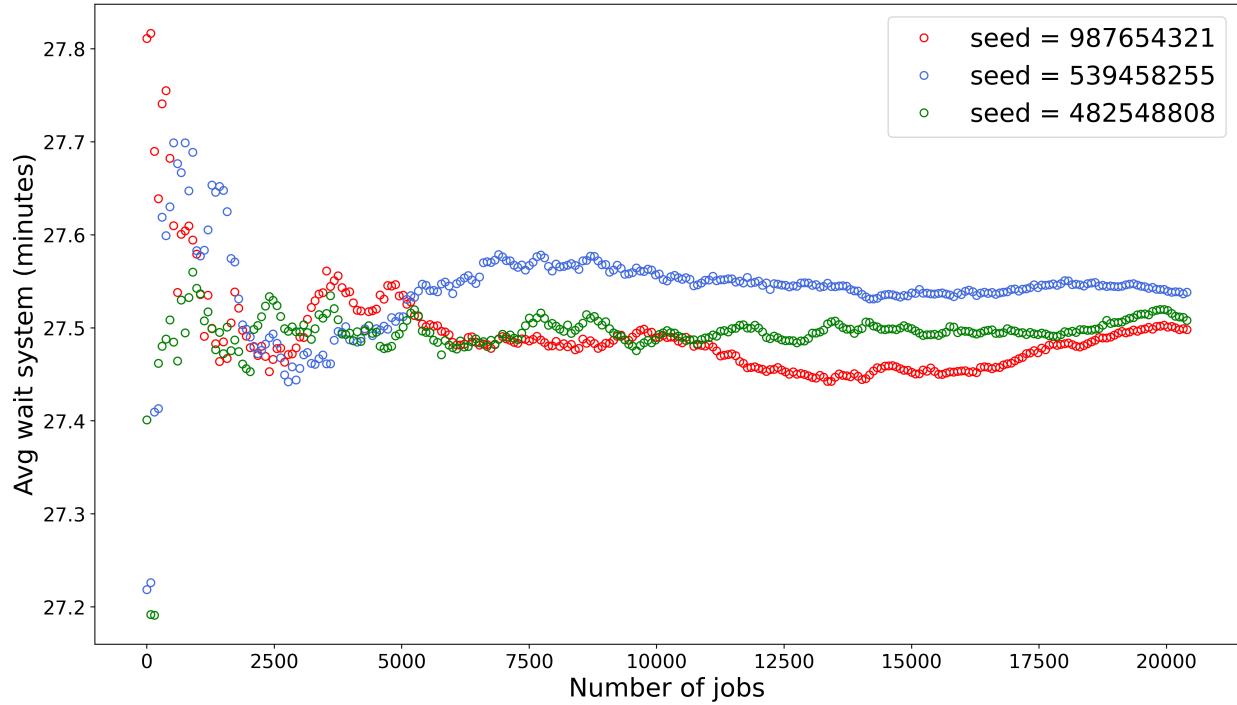


Figure 2: Average Wait System, 08:00-12:00 & 17:00-22:00, $\lambda_{arrival} = \frac{1}{14} \frac{job}{min}$, Arcades: 2, repliche=64, Sampling_freq=75

Come si evince dai grafici, il numero minimo di nodi Arcade necessari per il raggiungimento della stazionarietà, nella fascia oraria 08:00-12:00 e 17:00-22:00, è pari a 3. Infatti, nella figura 2 il tempo medio di risposta del sistema assume un comportamento stabile dopo il processamento di circa 5000 jobs. Inoltre, sono stati utilizzati 3 seed iniziali diversi, in modo tale da simulare il comportamento in condizioni

aleatorie differenti. Tali considerazioni, con le dovute differenze, valgono per le altre fasce orarie.

Per quanto riguarda la seconda fascia oraria (12:00-17:00), è possibile osservare i risultati dell'analisi in figura 3 e figura 4.

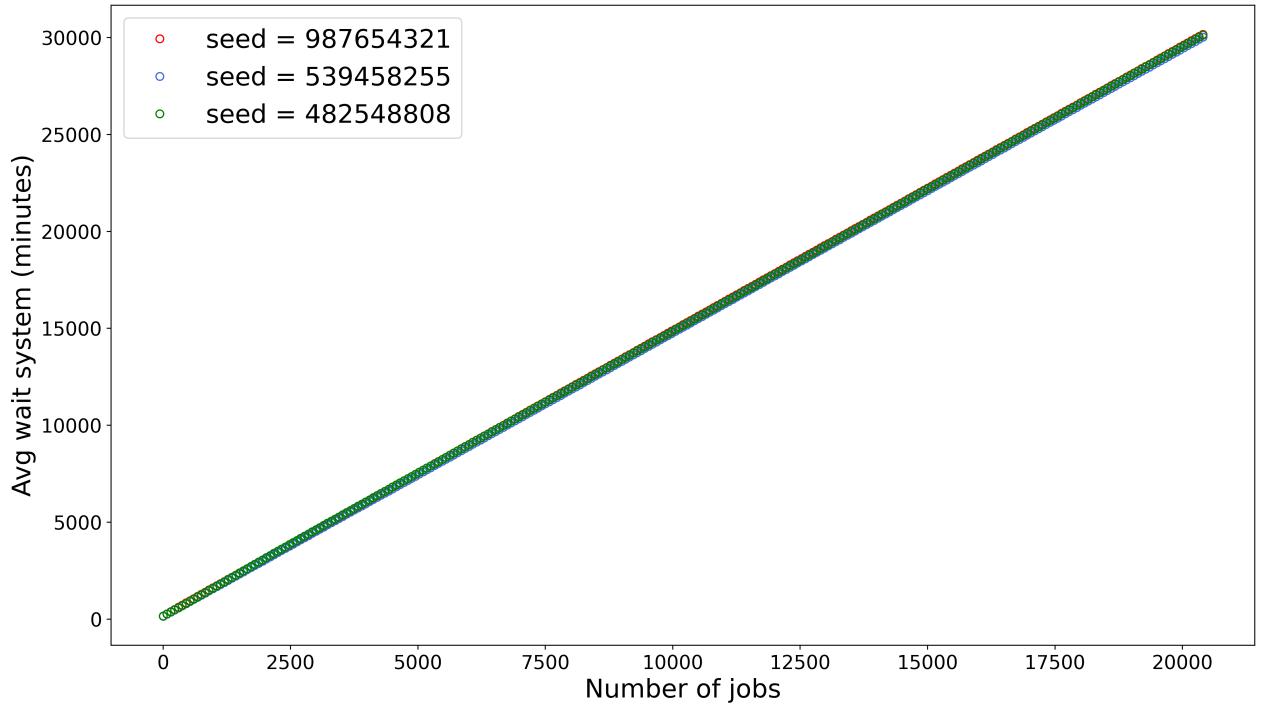


Figure 3: Average Wait System, 12:00-17:00, $\lambda_{arrival} = \frac{1}{5} \frac{job}{min}$, Arcades: 2,
repliche=64, Sampling_freq=75

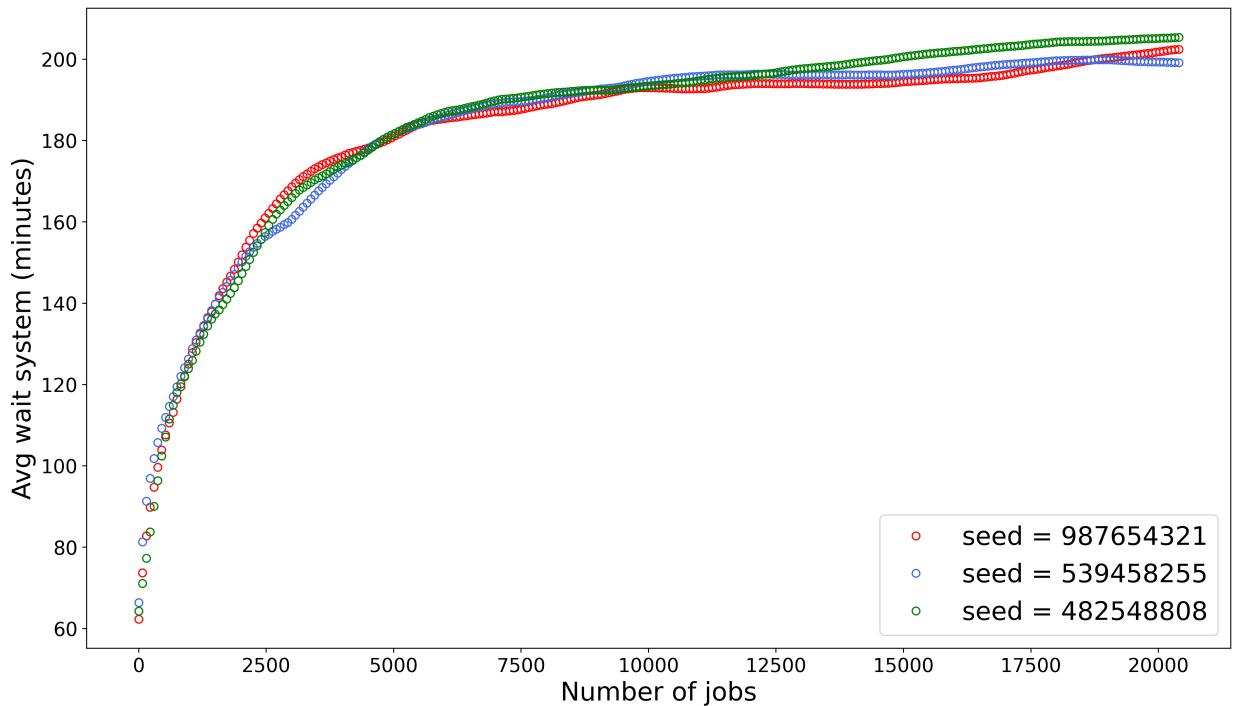


Figure 4: Average Wait System, 12:00-17:00, $\lambda_{arrival} = \frac{1}{5} \frac{job}{min}$, Arcades: 3,
repliche=64, Sampling_freq=75

Per quanto riguarda la quarta fascia oraria (22:00-08:00), è possibile osservare i risultati dell'analisi in figura 5.

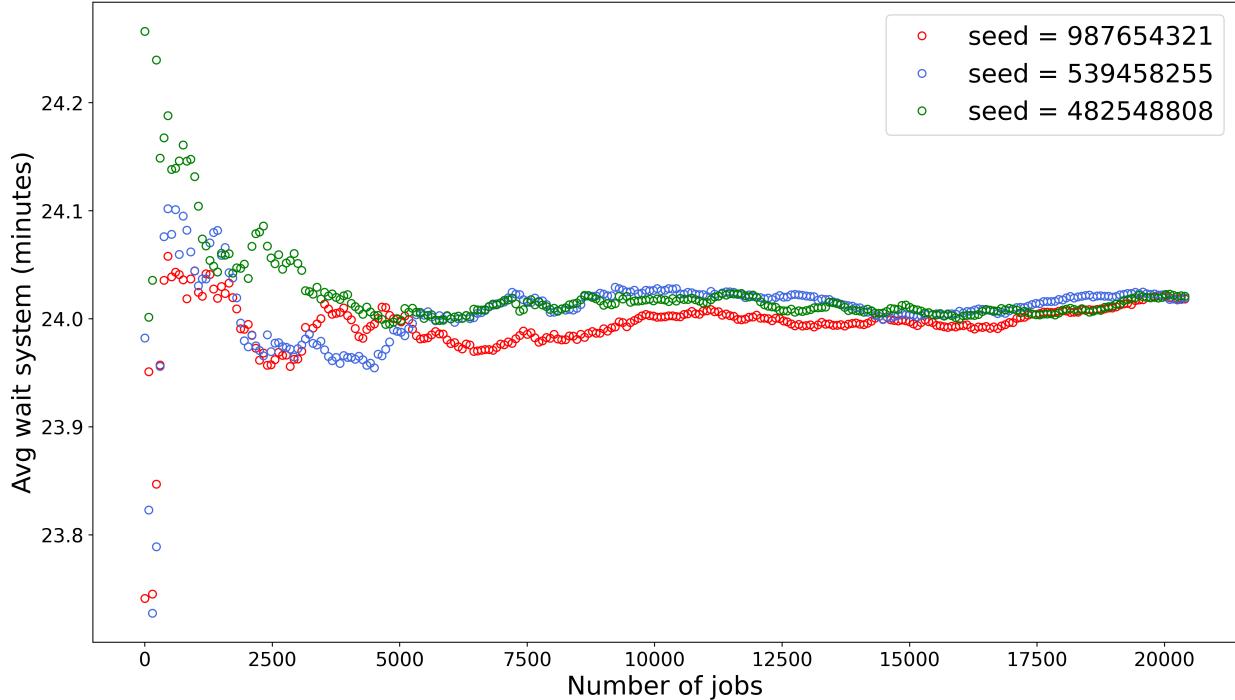


Figure 5: Average Wait System, 22:00-08:00, $\lambda_{arrival} = \frac{1}{35} \frac{job}{min}$, Arcades: 1, repliche=64, Sampling_freq=75

In questo caso, il tempo di risposta medio del sistema si stabilizza anche con l'utilizzo di un solo nodo Arcade.

1.7.2 Analisi del comportamento stazionario

Attraverso l'analisi del comportamento stazionario si vuole trovare la configurazione ottima di nodi arcade, per ogni fascia oraria, che permetta di raggiungere gli obiettivi prefissati in 1.1, partendo dai risultati ottenuti dallo studio precedente. In questa fase sono state effettuate simulazioni con tempi d'osservazione molto lunghi, utilizzando la tecnica delle batch-means per ottenere delle stime degli indici di prestazione in regime stazionario, evitando la problematica del bias dovuto alle condizioni iniziali del sistema. Cruciale in questa fase la scelta di due parametri: b e k , rispettivamente la grandezza del batch e il numero di batch in cui si divide la simulazione. Per la scelta di b è stata utilizzata la linea guida di Banks, Carson, Nelson, e Nicol (2001), la quale afferma che il batch size venga aumentato fintantoché l'autocorrelazione a lag 1 tra batch means sia minore di 0.2. Nel caso in esame è stato constatato che tale direttiva viene rispettata con un b pari a 256. Nonostante ciò, è stato scelto un valore di b pari a 512 al fine di ridurre maggiormente la dipendenza tra i batch.

Controllare ultima frase

Per la scelta del parametro k è stato usato il valore pari a 64 come consigliato dalle linee guida.

In seguito per ogni fascia oraria viene riportato il grafico relativo alla funzione guadagno e del tempo medio di risposta del sistema al variare del numero di nodi Arcade, mediata su i seed. L'asse y "income" riportato nelle figure non rappresenta l'effettivo guadagno della sala giochi in una specifica fascia oraria, che verrà analizzato in uno studio successivo, ma esprime l'effettivo andamento del guadagno, utile ad individuare la miglior configurazione da utilizzare.

Per quanto riguarda la prima fascia e la terza fascia oraria, rispettivamente (08:00-12:00) e (17:00-22:00), è possibile osservare i risultati dell'analisi in figura figura 6. Tali fasce, infatti, hanno lo stesso $\lambda_{arrival}$ da modellazione.

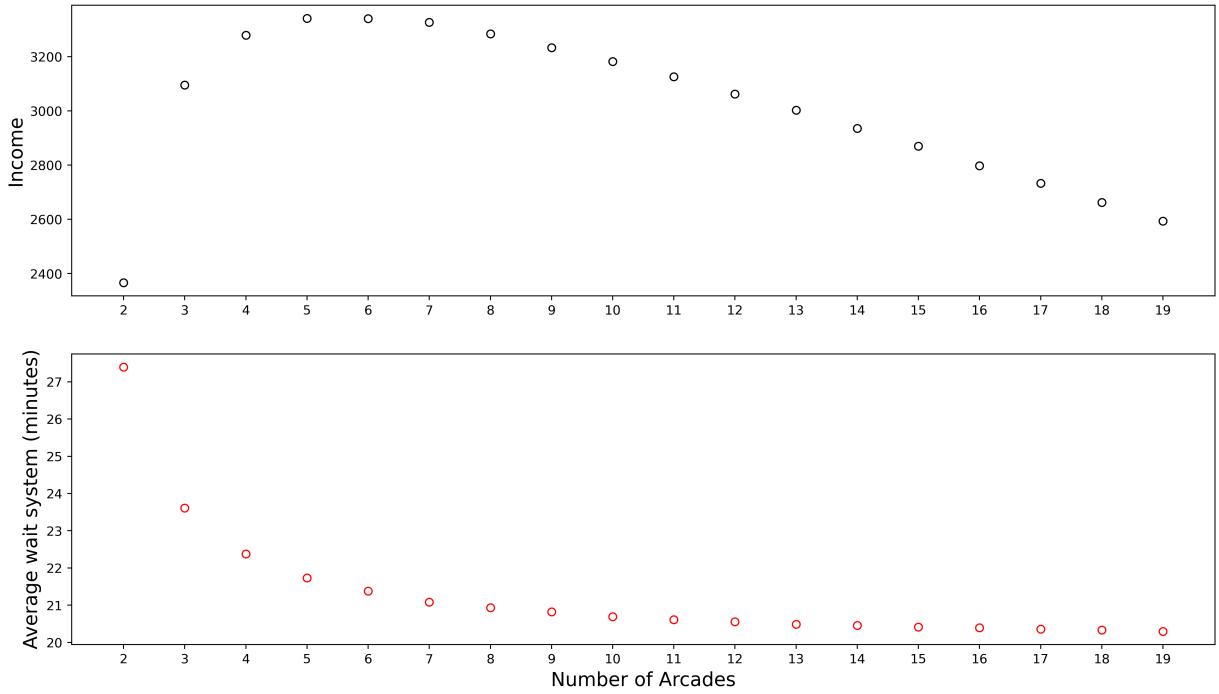


Figure 6: Income, 08:00-12:00 & 17:00-22:00, $\lambda_{arrival} = \frac{1}{14} \frac{job}{min}$, b=256, k=64

Usare 1 seed o fare la media??

Come si evince dal grafico, il numero ottimale di nodi Arcade per la prima e la terza fascia oraria, che permette di raggiungere entrambi gli obiettivi prefissati, è pari a 5.

Per quanto riguarda la seconda fascia oraria (12:00-17:00), è possibile osservare i risultati dell'analisi in figura 7.

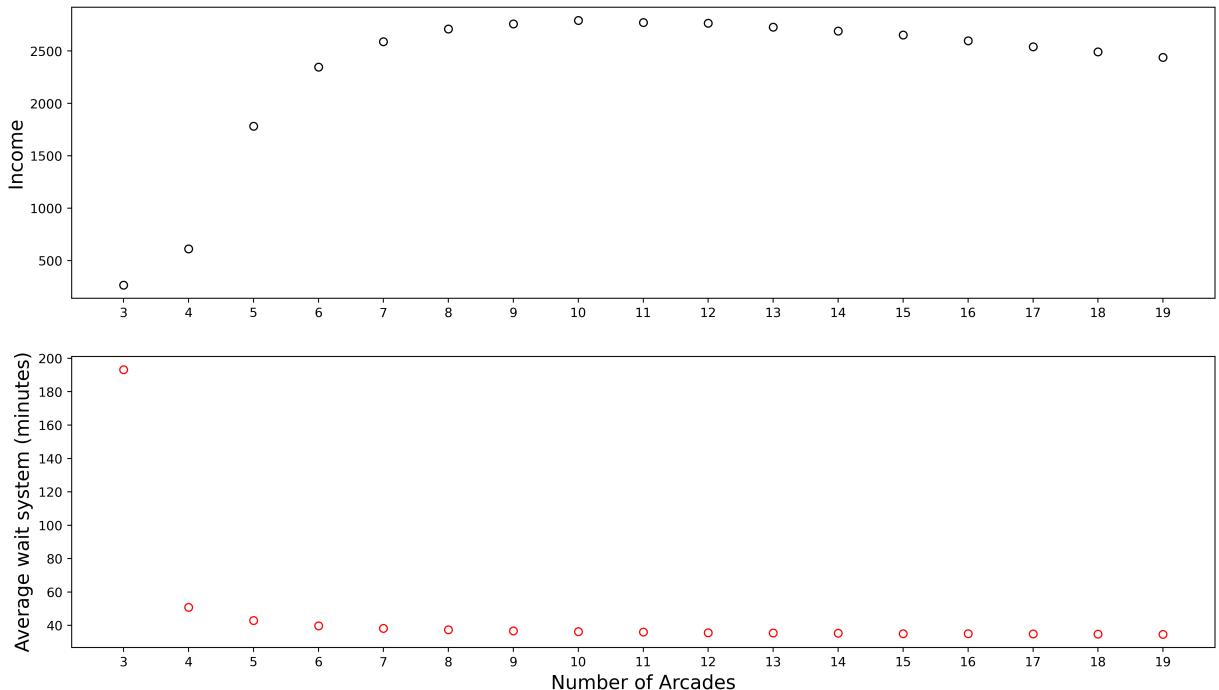


Figure 7: Income, 12:00-17:00, $\lambda_{arrival} = \frac{1}{5} \frac{job}{min}$, b=256, k=64

Il numero ottimale di server Arcade, per la seconda fascia oraria, è pari a 10.

Per quanto riguarda la quarta fascia oraria (22:00-08:00), è possibile osservare i risultati dell'analisi in figura 8.

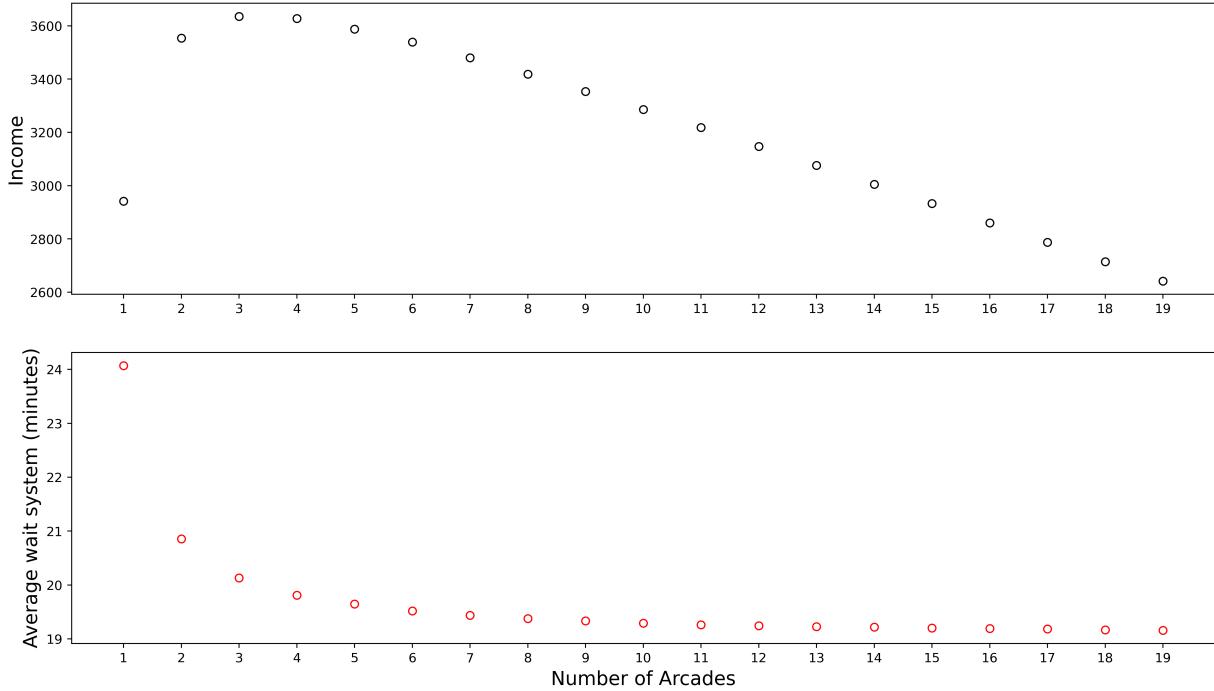


Figure 8: Income, 22:00-08:00, $\lambda_{arrival} = \frac{1}{35} \frac{job}{min}$, b=256, k=64

In questo caso, il numero ottimale di server Arcade da utilizzare è pari a 3.

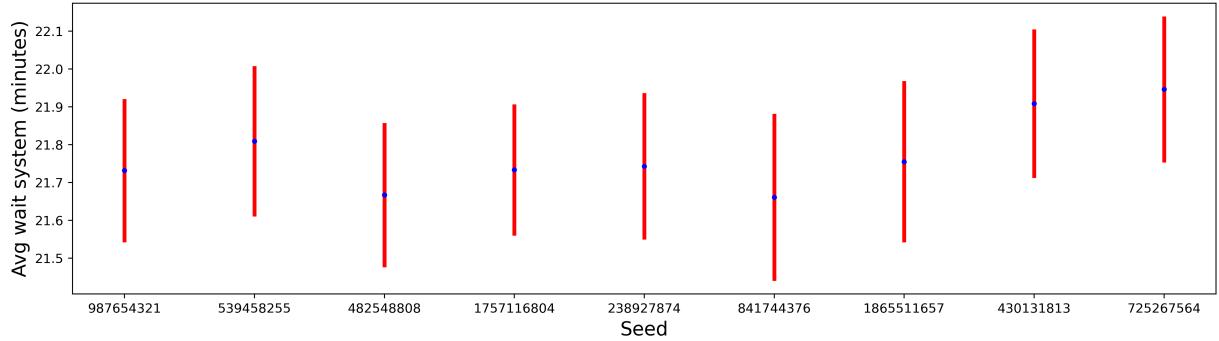
In tabella 3 è possibile individuare la configurazione ottimale dei nodi Arcade, per ogni fascia oraria.

Table 3: Configurazione ottimale

Fascia oraria	Configurazione ottimale *
08:00-12:00	5
12:00-17:00	10
17:00-22:00	5
22:00-08:00	3

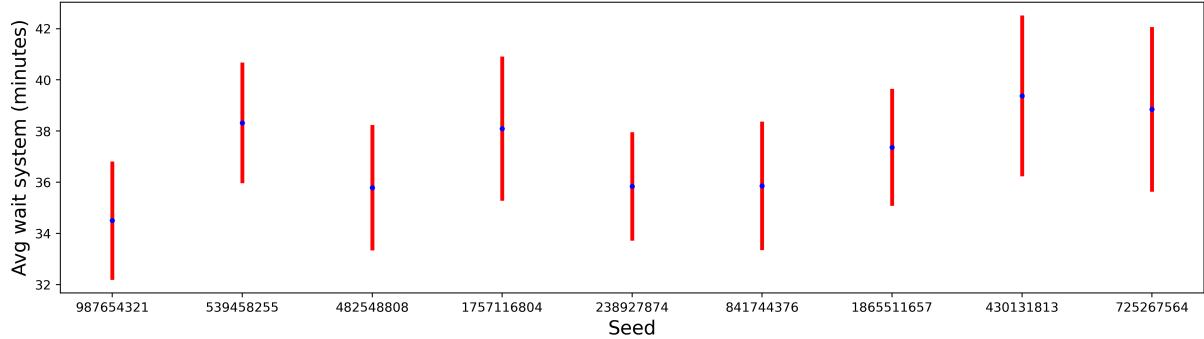
* Numero di nodi Arcade nel sistema.

Sono stati effettuati run con seed diversi con la configurazione ottimale, precedentemente ricavata, per studiare il tempo di risposta medio, per ogni fascia oraria, con un intervallo di confidenza pari al 95%.



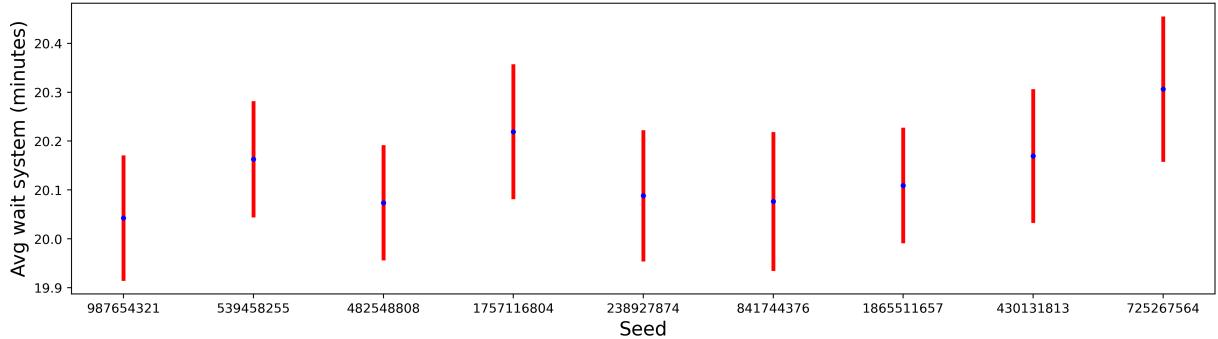
SEED	MEAN VALUE	STD	CONFIDENCE INTERVAL	CONFIDENCE LEVEL
987654321	21.73139159874175	0.7278992755036976	± 0.18962315438362418	95%
539458255	21.809212808024988	0.7628715753780063	± 0.19873369761589693	95%
482548808	21.666685360018118	0.7320529654790335	± 0.19070522139751125	95%
1757116804	21.733187590110223	0.6658261867805824	± 0.1734526548624279	95%
238927874	21.742766801070932	0.7434122717449919	± 0.19366440484259734	95%
841744376	21.660830598797475	0.8470157213826731	± 0.2206538710867113	95%
1865511657	21.755024295060352	0.8186747356682376	± 0.21327083432548677	95%
482548808	21.666685360018118	0.7320529654790335	± 0.19070522139751125	95%
430131813	21.90874969573461	0.754130418552268	± 0.19645656149825008	95%
725267564	21.946094727089662	0.7402658474250582	± 0.19284473799492488	95%

Figure 9: Average Wait System,, 08:00-12:00 & 17:00-22:00,
 $\lambda_{arrival} = \frac{1}{14} \frac{job}{min}$, b=256, k=64



SEED	MEAN VALUE	STD	CONFIDENCE INTERVAL	CONFIDENCE LEVEL
987654321	34.503513570706325	8.870492435880855	± 2.3108290023559195	95%
539458255	38.32180640228681	9.039012185696797	± 2.35472964576947	95%
482548808	35.78978563889555	9.40094307753797	± 2.4490153246959654	95%
1757116804	38.097774394834445	10.814676179580884	± 2.817303272338745	95%
238927874	35.8393774525391	8.131588759718287	± 2.118339120067404	95%
841744376	35.861527566504954	9.63067884947371	± 2.5088603390916693	95%
1865511657	37.366859832892494	8.772646653340434	± 2.285339451050061	95%
482548808	35.78978563889555	9.40094307753797	± 2.4490153246959654	95%
430131813	39.376704254043986	12.050194731783382	± 3.1391650093297665	95%
725267564	38.850427538905706	12.35301797962913	± 3.218052709056448	95%

Figure 10: Average Wait System,, 12:00-17:00, $\lambda_{arrival} = \frac{1}{5} \frac{job}{min}$, b=256, k=64



SEED	MEAN VALUE	STD	CONFIDENCE INTERVAL	CONFIDENCE LEVEL
987654321	20.042817450512057	0.4928057548917107	± 0.12837955042104535	95%
539458255	20.16313702162444	0.45637347402921813	± 0.11888867132413258	95%
482548808	20.07398647587404	0.45137382221992145	± 0.11758622498464064	95%
1757116804	20.219277263269855	0.5301617655476288	± 0.13811106797319805	95%
238927874	20.08825426254847	0.5156308266627769	± 0.1343256507317884	95%
841744376	20.06775034976272	0.545370928642871	± 0.14207316764647823	95%
1865511657	20.109315177477768	0.45398169156273066	± 0.11826559427053757	95%
482548808	20.07398647587404	0.45137382221992145	± 0.11758622498464064	95%
430131813	20.169560910235067	0.5253247790569793	± 0.1368509715442542	95%
725267564	20.306340602623465	0.5702115939307782	± 0.14854434500219413	95%

Figure 11: Average Wait System,, 22:00-08:00, $\lambda_{arrival} = \frac{1}{35} \frac{job}{min}$, b=256, k=64

1.7.3 Analisi guadagno giornaliero

2 Implementation

Proin lobortis efficitur dictum. Pellentesque vitae pharetra eros, quis dignissim magna. Sed tellus leo, semper non vestibulum vel, tincidunt eu mi. Aenean pretium ut velit sed facilisis. Ut placerat urna facilisis dolor suscipit vehicula. Ut ut auctor nunc. Nulla non massa eros. Proin rhoncus arcu odio, eu lobortis metus sollicitudin eu. Duis maximus ex dui, id bibendum diam dignissim id. Aliquam quis lorem lorem. Phasellus sagittis aliquet dolor, vulputate cursus dolor convallis vel. Suspendisse eu tellus feugiat, bibendum lectus quis, fermentum nunc. Nunc euismod condimentum magna nec bibendum. Curabitur elementum nibh eu sem cursus, eu aliquam leo rutrum. Sed bibendum augue sit amet pharetra ullamcorper. Aenean congue sit amet tortor vitae feugiat.

In congue risus leo, in gravida enim viverra id. Donec eros mauris, bibendum vel dui at, tempor commodo augue. In vel lobortis lacus. Nam ornare ullamcorper mauris vel molestie. Maecenas vehicula ornare turpis, vitae fringilla orci consectetur vel. Nam pulvinar justo nec neque egestas tristique. Donec ac dolor at libero congue varius sed vitae lectus. Donec et tristique nulla, sit amet scelerisque orci. Maecenas a vestibulum lectus, vitae gravida nulla. Proin eget volutpat orci. Morbi eu aliquet turpis. Vivamus molestie urna quis tempor tristique. Proin hendrerit sem nec tempor sollicitudin.

```
hello.py
#!/usr/bin/python

import sys
sys.stdout.write("HelloWorld!\n")
```

Fusce eleifend porttitor arcu, id accumsan elit pharetra eget. Mauris luctus velit sit amet est sodales rhoncus. Donec cursus suscipit justo, sed tristique ipsum fermentum nec. Ut tortor ex, ullamcorper varius congue in, efficitur a tellus. Vivamus ut rutrum nisi. Phasellus sit amet enim efficitur, aliquam nulla id, lacinia mauris. Quisque viverra libero ac magna maximus efficitur. Interdum et malesuada fames ac ante

ipsum primis in faucibus. Vestibulum mollis eros in tellus fermentum, vitae tristique justo finibus. Sed quis vehicula nibh. Etiam nulla justo, pellentesque id sapien at, semper aliquam arcu. Integer at commodo arcu. Quisque dapibus ut lacus eget vulputate.

Command Line

```
$ chmod +x hello.py  
$ ./hello.py
```

```
Hello World!
```

Vestibulum sodales orci a nisi interdum tristique. In dictum vehicula dui, eget bibendum purus elementum eu. Pellentesque lobortis mattis mauris, non feugiat dolor vulputate a. Cras porttitor dapibus lacus at pulvinar. Praesent eu nunc et libero porttitor malesuada tempus quis massa. Aenean cursus ipsum a velit ultricies sagittis. Sed non leo ullamcorper, suscipit massa ut, pulvinar erat. Aliquam erat volutpat. Nulla non lacus vitae mi placerat tincidunt et ac diam. Aliquam tincidunt augue sem, ut vestibulum est volutpat eget. Suspendisse potenti. Integer condimentum, risus nec maximus elementum, lacus purus porta arcu, at ultrices diam nisl eget urna. Curabitur sollicitudin diam quis sollicitudin varius. Ut porta erat ornare laoreet euismod. In tincidunt purus dui, nec egestas dui convallis non. In vestibulum ipsum in dictum scelerisque.



Notice: In congue risus leo, in gravida enim viverra id. Donec eros mauris, bibendum vel dui at, tempor commodo augue. In vel lobortis lacus. Nam ornare ullamcorper mauris vel molestie. Maeceenas vehicula ornare turpis, vitae fringilla orci consectetur vel. Nam pulvinar justo nec neque egestas tristique. Donec ac dolor at libero congue varius sed vitae lectus. Donec et tristique nulla, sit amet scelerisque orci. Maecenas a vestibulum lectus, vitae gravida nulla. Proin eget volutpat orci. Morbi eu aliquet turpis. Vivamus molestie urna quis tempor tristique. Proin hendrerit sem nec tempor sollicitudin.