

# Progetto di Sistemi Complessi: Modelli e Simulazione

## Supermarket Simulation

Tonelli Lidia Lucrezia (m. 813114)  
Grassi Marco (m. 830694)  
Giudice Gianluca (m. 829664)

University of Milano Bicocca

Settembre 2021

- 1 Introduzione
- 2 Stato dell'arte
- 3 Descrizione del modello
  - Overview
  - Agenti del modello
- 4 Implementazione dei comportamenti e delle strategie
- 5 Simulazione
- 6 Conclusioni

# Introduzione

Il nostro progetto è un modello basato su agenti che **simula** il comportamento dei clienti in un supermercato durante le fasi di scelta della coda relativa a diversi tipi di casse.

Prenderemo in considerazione 3 tipi di casse diverse (standard, self-service e self-scan) e 2 tipi di scelte fatte dai clienti (scelta della coda, jockeying). L'utilizzo di code è fondamentale per gestire le grandi quantità di clienti.

## Obiettivo

Sperimentare diverse **configurazioni di casse** e **strategie di scelta della coda** per gestire in modo ottimale il flusso di clienti e ridurre al minimo il tempo d'attesa passato in coda.

Un supermercato è un **sistema complesso** in cui agiscono diverse entità, come clienti e casse.

Si verificano aspetti emergenti difficilmente prevedibili dovuti a diversi aspetti:

- Flusso di clienti in ingresso variabile
- Numero di prodotti che un cliente acquista
- Numero di casse aperte contemporaneamente
- Strategia di scelta della coda dei clienti
- Strategia di cambio della coda dei clienti

# Stato dell'arte

Il principale spunto per il modello è stato l'articolo *Data-driven simulation modeling of the checkout process in supermarkets: Insights for decision support in retail operations*<sup>1</sup>, che utilizza 5 strategie di scelta della coda confrontando i **tempi d'attesa medi** dei clienti.

---

<sup>1</sup>Antczak, Tomasz and Weron, Rafał and Zabawa, Jacek, 2020

# Estensioni

Abbiamo voluto estendere il modello introducendo nuovi concetti:

- **Jockeying**<sup>2</sup>: quando un cliente sta attendendo in coda, confronta i tempi d'attesa della propria coda con quelli delle code vicine e può decidere di spostarsi di conseguenza.
- **Code parallele e N-Fork**<sup>3</sup>: vogliamo indagare sull'effetto della disposizione delle code sui tempi di attesa medi, questa può essere **parallela**, se ogni cassa ha una coda dedicata, oppure **N-Fork**, se c'è un'unica coda condivisa.

---

<sup>2</sup>*On jockeying in queues*, E. Koenigsberg, 1966

<sup>3</sup>*Methods for improving efficiency of queuing systems*, Yanagisawa, D and Suma, Y and Tanaka, Y and Tomoeda, A and Ohtsuka, K and Nishinari, K, 2011

# Estensioni

- **Casse self-scan:** l'articolo sopra citato prende in considerazione solo 2 tipi di casse, le casse **standard** e le casse **self-service**. Nel modello sono presenti le casse **self-scan**, usate attualmente in molti supermercati, che permettono di scannerizzare i prodotti in fase di spesa e rendere la fase di pagamento molto più veloce.
- **Simulazione non deterministica:** per far emergere comportamenti non banali nel supermercato e rendere più realistiche le simulazioni sono state aggiunte alcune variabili probabilistiche.

# Approccio ad agenti

- **Sistema multiagente** sviluppato in Python con il framework Mesa
  - **Ambiente:** supermercato
    - Il supermercato è una struttura divisa in zone, composta da code e casse
    - I clienti entrano nel supermercato per fare la spesa minimizzando il tempo impiegato
  - **Agenti:** clienti e casse
- I clienti sono agenti intelligenti (pianificano e decidono) con una componente imprevedibile che fa emergere un comportamento complesso interagendo con gli altri agenti



# Ambiente

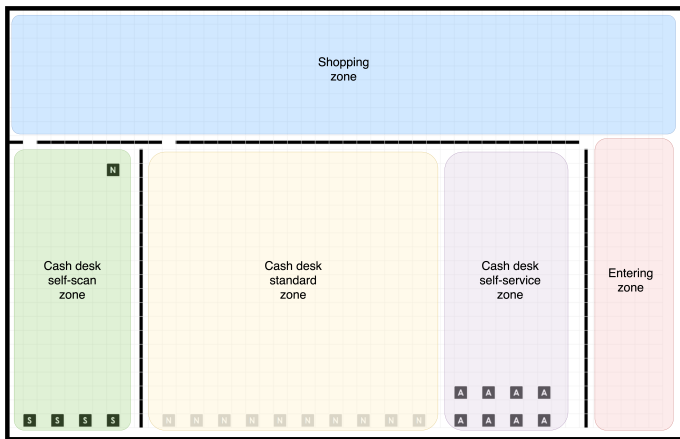


Figure: Stuttura del supermercato diviso in zone.

# Ambiente

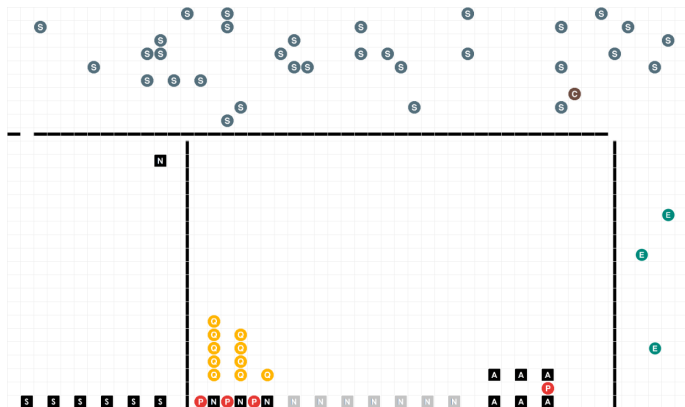


Figure: Interazione tra agenti e ambiente.

# Agente Cliente

- I clienti sono gli agenti principali, si muovono nel supermercato con l'obiettivo di fare la spesa e attendere il minimo tempo possibile in coda
- Per minimizzare il tempo in coda il cliente usa strategie di **scelta della coda** e di **jockeying**, quindi ha bisogno di una pianificazione
- Il cliente è un agente di tipo *utility-based* perchè per la pianificazione e la valutazione dei tempi d'attesa utilizza una utility function

# Agente Cliente - workflow

- ➊ **Attesa all'entrata del supermercato:** l'agente è in attesa fino a che non può mettersi nella zona d'entrata.
- ➋ **Fase di shopping:** si muove nella zona di shopping e inizia a raccogliere elementi fino a raggiungere il *basket size* desiderato; questa fase dipende dalla velocità di shopping, un parametro del modello.
- ➌ **Scelta della coda:** finita la spesa, deve scegliere la cassa in base alla utility function; viene scelta la coda  $q^*$  tale che

$$q^* = \operatorname{argmin}_{q \in Q} f(q)$$

dove  $Q$  è l'insieme delle code dedicate e  $f$  varia con la strategia.

## Agente Cliente - workflow

- 1 **Attesa in coda e jockeying:** mentre il cliente è in coda può decidere di lasciarla per una coda migliore. Considera le 2 code adiacenti (parametro del modello) alla propria e per ognuna calcola la coda migliore secondo la sua strategia di jockeying. Se il "guadagno" risulta maggiore di un certo **threshold**, allora il cliente può decidere di cambiare coda. Si estrae quindi un numero casuale che determina se cambiare coda o no (perchè non tutte le persone fanno jockeying).
- 2 **Attesa alla cassa:** il cliente viene servito dalla cassa e deve attendere la fine del pagamento per uscire dal supermercato.

# Agente Cassa

- I clienti, una volta conclusa la fase di spesa, scelgono una coda in attesa di essere serviti in cassa per il pagamento.
- Ogni cassa ha al più una coda associata.
- La cassa è un'agente di tipo *model-based reflex*, in cui lo stato è il cliente che si sta processando in un determinato momento.
- Il comportamento di una cassa è piuttosto semplice:
  - 1 Prendi un cliente dalla coda (se disponibile)
  - 2 Processa il cliente
  - 3 Ripeti
- Le code (FIFO) ammissibili per ogni cassa sono di 2 tipi:
  - 1 Coda dedicata: ogni cassa ha una coda dedicata
  - 2 Coda condivisa: una coda è associata a più casse, tutte le casse serviranno i clienti che si sono accodati alla coda condivisa
- Nel modello sono stati modellati 4 tipi di casse diverse.

# Agente Cassa - Tipo 1: Standard

- Rappresenta la classica cassa di un supermercato.
- Questa cassa può avere una coda dedicata o condivisa, in entrambi i casi:
  - 1 Il cliente si accoda
  - 2 La cassa prende il primo cliente dalla coda
  - 3 Il cassiere processa gradualmente tutti gli articoli del cliente
  - 4 Ripeti

## Agente Cassa - Tipo 2: Self-service

- Cassa in cui non è presente un cassiere ma è il cliente stesso a dover passare uno alla volta gli articoli acquistati.
- Tutte le casse self-service hanno una coda condivisa
  - ① Il cliente si accoda alla coda condivisa
  - ② La cassa prende il primo cliente dalla coda
  - ③ Il cliente processa ogni articolo
  - ④ Il cliente lascia il supermercato



## Agente Cassa - Tipo 3: Self-scan

- Il cliente scannerizza gli articoli durante la spesa mediante un dispositivo fornito dal supermercato.
- Avendo già scannerizzato gli articoli a priori non è necessario farlo in cassa.
- Vengono effettuati controlli a campione per verificare il corretto comportamento dei clienti (tutti gli articoli devono essere stati effettivamente scannerizzati durante la spesa).
- Tutte le casse hanno una coda condivisa
  - ① Il cliente si accoda alla coda condivisa
  - ② La cassa prende il primo cliente dalla coda
  - ③ Nel caso in cui il cliente è stato estratto per una rilettura della spesa si reca ad una cassa riservata, altrimenti paga ed esce dal supermercato

## Agente Cassa - Tipo 4: Riservata

- Comportamento analogo alla cassa standard.
- Ha una coda dedicata.
- Nel caso di rilettura alla cassa "self-scan", il cliente viene normalmente processato in questa nuova cassa.
- La rilettura può essere **parziale** (vengono controllati solo 10 elementi), o **totale** (viene controllata tutta la spesa).
- Nessun cliente può venire processato nella cassa riservata a meno di una rilettura.

# Considerazioni

- Il modello da noi descritto é stato implementato con l'obiettivo di essere facile da modificare.
- Il comportamento degli agenti viene gestito con una macchina a stati finiti implementata tramite il pattern **State**. Questo isola logicamente aspetti quali azioni da compiere e rappresentazione grafica da assumere in determinati stati.
- Dove esistono diversi approcci ad una scelta, per esempio nella scelta della coda, questi vengono gestiti con il pattern **Strategy**, dando possibilità di scelta tra più opzioni e una facilitazione nel crearne di nuove.
- I parametri che regolano il modello descritti a seguire sono configurabili a piacere.

# Parametri

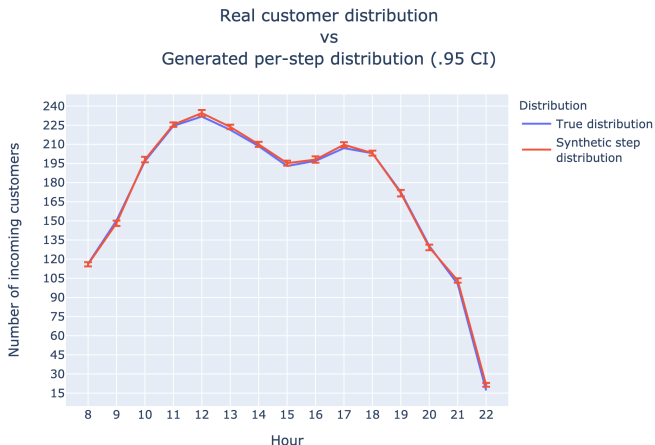
- Configurazione del supermercato:
  - Dimensione delle zone (entering zone, shopping zone)
  - Numero di casse (standard, self-service o self-scan + 1 riservata)
  - Code N-fork o parallele per le casse standard
- Parametro per l'errore di stima del basket size: usato nelle strategie di scelta della coda e jockeying per simulare l'errore commesso dagli umani nello stimare la quantità di oggetti nei carrelli
- Jockeying:
  - Numero di code adiacenti considerate
  - Threshold
  - Probabilità di fare jockeying
- Distribuzione dei clienti in entrata (presa dai dati di Antczak e altri<sup>4</sup>)
- Distribuzione dei basket size (presa dai dati di Antczak e altri<sup>4</sup>)

---

<sup>4</sup>Antczak, Tomasz and Weron, Rafał and Zabawa, Jacek, 2020

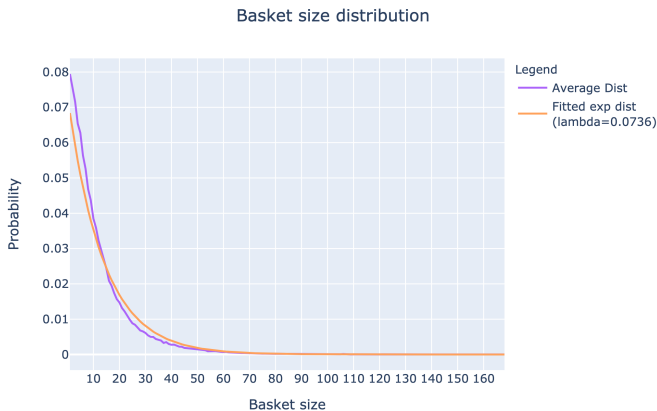
# Parametri - distribuzione dei clienti

Nella figura si riporta la distribuzione di entrata dei clienti reale e la distribuzione generata effettuando normalizzazione, media e divisione per step.



## Parametri - distribuzione dei basket size

Nella figura viene mostrata la distribuzione reale dei dati e la distribuzione esponenziale derivata aggregandoli, la quale viene usata nel modello per generare il basket size di ogni cliente.



# Parametri

- Parametri di tempo:
  - Durata di uno step (attualmente 30 secondi)
  - Velocità di shopping del cliente: numero di articoli messi nel carrello ad ogni step
  - Tempo di elaborazione della spesa da parte delle casse: per le self-scan è 1 step, per le standard e le self-service è governato dai parametri  $a, b, \alpha, \beta \in \mathbb{R}$ , presi dall'articolo di Antczak e altri sopra citato

# Implementazione dei comportamenti e delle strategie

- Il cliente una volta presi tutti gli articoli si reca alla coda.
- La cassa self-scan deve essere scelta prima di fare la spesa e non è possibile cambiare.
- Il cliente vuole minimizzare il tempo speso all'interno del supermercato tramite:
  - **Scelta iniziale della coda:** il cliente sceglie la coda ottima rispetto ad una determinata strategia
  - **Fase di jockeying:** una volta in coda il cliente può scegliere di cambiarla se ne esiste una migliore



## Scelta della coda

- Terminata la fase di spesa un cliente decide tra le casse aperte dove accodarsi individuando la coda ottimale per una determinata metrica.
- A questo comportamento fanno eccezione i clienti che hanno inizialmente optato per la modalità di spesa self scan per i quali é obbligatorio recarsi alle casse di tipo self scan.
- Un cliente può accodarsi ad una cassa self-service solo se ha un numero di prodotti inferiore al limite imposto.

# Strategie di scelta della coda 1-2

La scelta della coda può avvenire in base a 4 strategie:

- 1 Minor numero di elementi

$$\arg \min_q \sum_{i=1}^N \text{estimate-basket-size}(c_i) \quad (1)$$

Per rendere più realistico il calcolo è possibile "sbagliare" il conto di articoli per cliente tramite il parametro *standard deviation coefficient*.

- 2 Minor numero di persone

$$\arg \min_q |q| \quad (2)$$

Dove  $q$  è una coda e  $c_i$  è l' $i$ -esimo cliente.

## Strategie di scelta della coda 3-4

- 3 Minor tempo di attesa rispetto al tempo di servizio medio

$$\arg \min_q |q| * \frac{1}{M} \sum_{j=1}^M (\text{total-service-time}(q_j)) \quad (3)$$

Dove  $M$  é il numero di code nel supermercato

- 4 Minor tempo di attesa rispetto alla *power regression*

$$\arg \min_q |q| * \text{total-service-time}(q) \quad (4)$$

La quantità total-service-time è calcolata secondo le seguenti formule.

## Formule cassa standard

- Transaction time

$$\text{transaction-time}_i = e^{a \log(\text{estimate-basket-size}(c_i)) + b} \quad (5)$$

- Break Time

$$\text{break-time}_i = \frac{\beta^\alpha \text{estimate-basket-size}(c_i)^{\alpha-1} e^{-\beta \text{estimate-basket-size}(c_i)}}{\Gamma(\alpha)} \quad (6)$$

- Dove  $\Gamma$  è la funzione **gamma**

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx \quad \forall z \in \mathbb{C} \quad (7)$$

- Total service time

$$\text{total-service-time}(q_j) = \sum_{i=1}^N (\text{transaction-time}_i + \text{break-time}_i) \quad (8)$$

$$a = 0.6984, \quad b = 2.1219, \quad \alpha = 3.074209, \quad \beta = \frac{1}{4.830613} \quad (9)$$

# Formule cassa self-service

- Transaction time

$$\text{transaction-time}_i = e^{a \log(\text{estimate-basket-size}(c_i)) + b} \quad (10)$$

- Break Time

$$\text{break-time}_i = e^{c \log(\text{estimate-basket-size}(c_i)) + d} \quad (11)$$

- Total service time

$$\text{total-service-time}(q_j) = \sum_{i=1}^N (\text{transaction-time}_i + \text{break-time}_i) \quad (12)$$

$$a = 0.6725, \quad b = 3.1223, \quad c = 0.2251, \quad d = 3.5167 \quad (13)$$

# Jockeying

- Un cliente fa **jockeying** se calcola che nelle code adiacenti a quella in cui è in attesa c'è un tempo di attesa minore, e quindi si sposta.
- Il *parametro di adiacenza* determina il numero di code adiacenti che il cliente prende in considerazione per il suo calcolo.
- Il cliente calcola un *guadagno* di tempo nel cambiare coda, se questo supera un certo threshold, allora fa jockey, altrimenti no perchè per lui "non ne vale la pena".
- Anche se esistono code migliori di altre, può non avvenire il jockey: viene estratto un parametro che rende il jockey aleatorio, in quanto non tutte le persone lo fanno.

# Jockeying - strategie

- Sono 2 le strategie per fare jockeying:

- 1 **Minimo numero di elementi:** è scelta la coda con il minor numero di elementi nei carrelli di tutti i clienti. Il guadagno è:

$$g = \# \text{elementi nei carrelli nella coda pivot} - \min_{q \in Q_{adj}} \# \text{elementi nei carrelli}$$

- 2 **Minimo numero di persone:** è scelta la coda con il minor numero di persone accodate. Il guadagno è:

$$g = i - \min_{q \in Q_{adj}} |q|$$

dove  $i$  è la posizione del cliente nella coda pivot

# Simulazione

Introduciamo la definizione di densità di clienti ad ogni step al fine di avere una misura della gestione dell'affluenza di clienti nel negozio per le simulazioni.

La **densità** di clienti per step corrisponde al numero di clienti medio per ogni cassa; la densità allo step  $i$  è:

$$\text{density}_i = \frac{\# \text{ customers in the supermarket}}{\# \text{ open cashdesks}}$$



# Validazione

Simulazione con gli stessi parametri del lavoro di Antczak e altri<sup>5</sup>:

- 20 casse standard con code parallele
- 6 casse self-service
- No jockeying

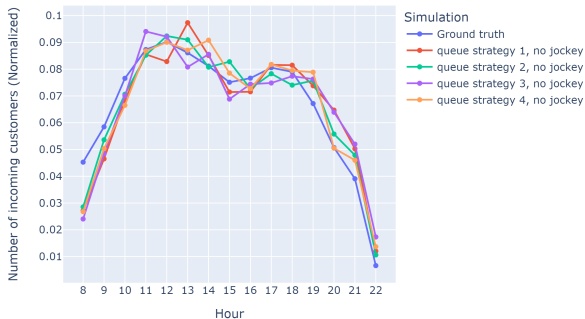
4 simulazioni, una per ogni strategia di scelta della coda

---

<sup>5</sup> *Data-driven simulation modeling of the checkout process in supermarkets: Insights for decision support in retail operations*, Antczak, Tomasz and Weron, Rafał and Zabawa, Jacek, 2020

# Validazione

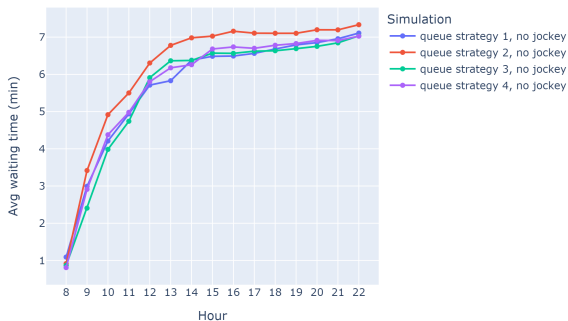
Total customers - Ground truth vs simulations for validation (Normalized)



A parità di parametri il nostro modello e quello dell'articolo sopportano allo stesso modo il numero di clienti nel negozio.

# Validazione

Average waiting time - simulations for validation



In assenza di jockey le strategie 3 e 4 sono quelle che tengono il tempo d'attesa medio più basso.

## Simulazione con jockey

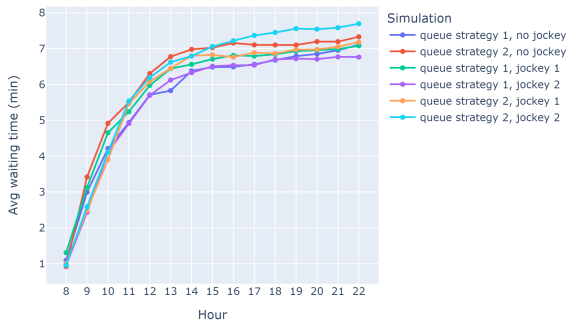
Aggiungiamo le 2 strategie di jockey. I parametri saranno quindi:

- 20 casse normali con code parallele
- 6 casse self-service
- 4 strategie di scelta della coda
- 2 strategie di jockey

Per un totale di 8 simulazioni.

# Simulazione con jockey

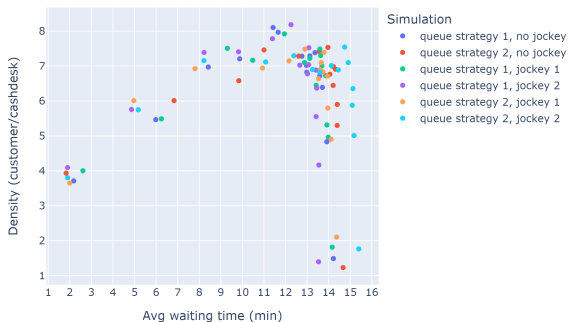
Average waiting time - simulations with jockey A



I risultati peggiori si raggiungono con la strategia che sceglie la coda con il minimo numero di persone.

# Simulazione con jockey

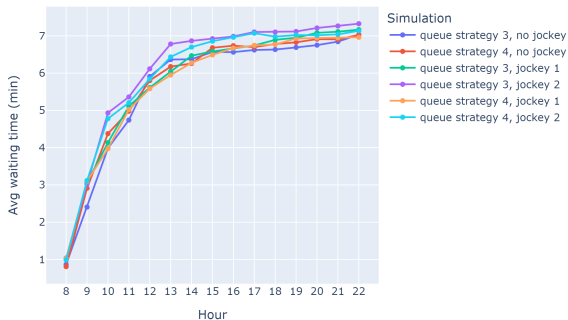
KDE (Kernel Density Estimates) - simulations with jockey A



Si raggiunge un punto critico di densità dopo il quale il sistema reagisce e permette alle code di scorrere abbassando il tempo d'attesa.

# Simulazione con jockey

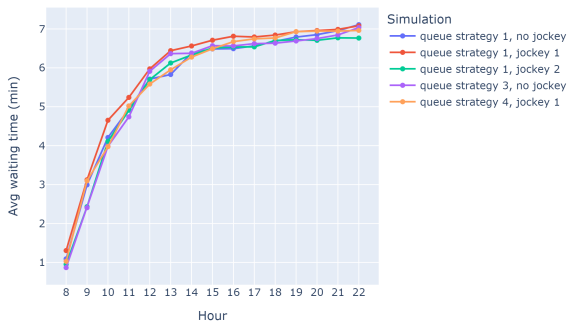
Average waiting time - simulations with jockey B



La strategia di scelta della coda migliore sembra essere quella in base al tempo medio di servizio delle casse; ancora una volta la strategia 2 peggiora i risultati.

# Simulazione con jockey

Average waiting time - best simulations with jockey



Le migliori 5 strategie di scelta della coda fino ad ora. I tempi medi d'attesa sono paragonabili.



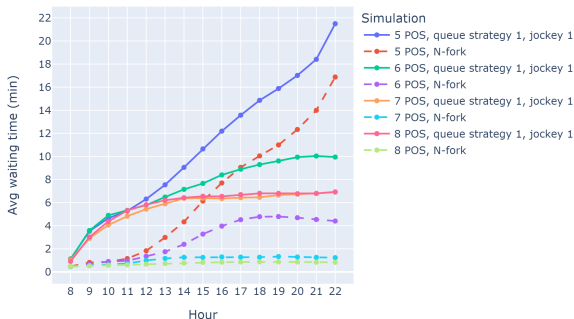
## Simulazione con coda condivisa

Mettiamo a confronto le code parallele con l'N-fork.

Osservando le simulazioni già fatte ci accorgiamo che il numero massimo di casse aperto in un istante è 8, per questo indaghiamo i casi con 5, 6, 7 e 8 casse.

# Simulazione con coda condivisa

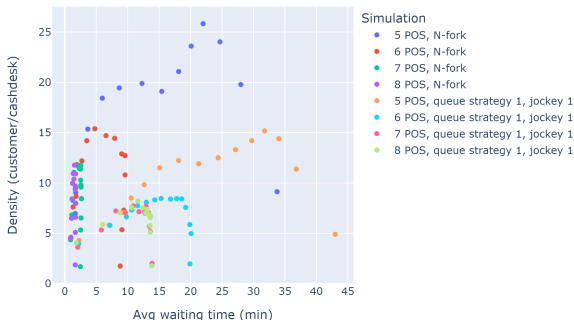
Average waiting time - jockey and parallel queues vs N-fork queue



5 casse sono troppo poche sia per le code parallele che per l'N-fork. Si nota la divisione netta tra i tempi d'attesa delle due configurazioni nel caso di 6, 7 e 8 casse.

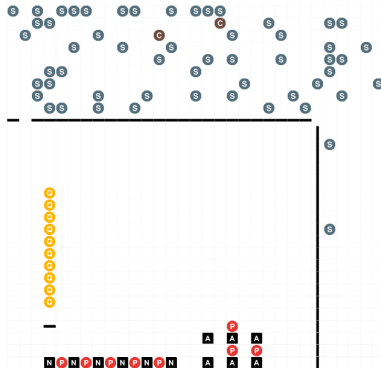
# Simulazione con coda condivisa

KDE (Kernel Density Estimates) - jockey and parallel queues vs N-fork queue



Di nuovo 5 casse sono troppo poche. La coda condivisa arriva a densità molto più alte, questo è un limite del modello, infatti quando la fila si riempie i clienti non possono mettersi in coda. In tutti i casi la coda condivisa sopporta più densità e abbassa i tempi medi.

# Simulazione con coda condivisa



Questo è un esempio di coda piena con clienti in attesa.

# Simulazione con casse self-scan

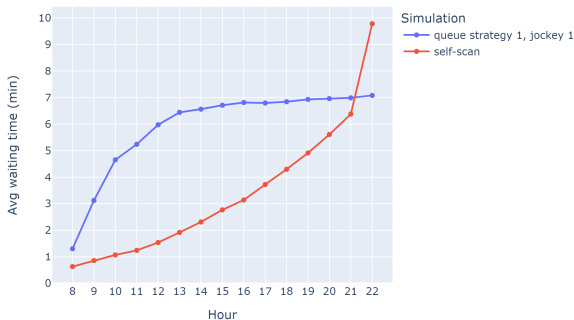
Simulazione con sole casse self-scan, in particolare 7.

Le probabilità di rilettura sono:

- 97% nessuna rilettura
- 2% rilettura parziale
- 1% rilettura totale

# Simulazione con casse self-scan

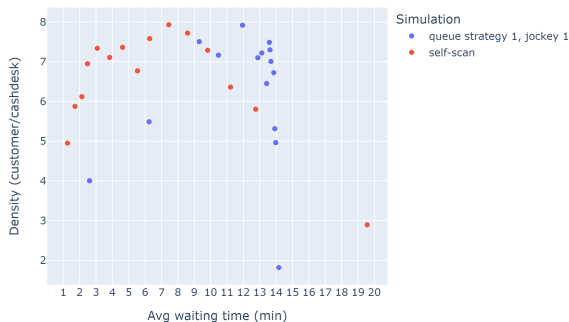
Average waiting time - standard cashdesks vs self-scan cashdesks



Il tempo d'attesa medio delle self-scan aumenta in modo esponenziale: la cassa riservata per le riletture è un collo di bottiglia. Bisognerebbe avere dei dati più completi sulla distribuzione di clienti self-scan.

# Simulazione con casse self-scan

KDE (Kernel Density Estimates) - standard cashdesks vs self-scan cashdesks



Le casse standard gestiscono il momento di massima densità abbassando i tempi medi d'attesa, le self-scan no.

# Simulazione non deterministica

Introduciamo i parametri randomici per rendere la simulazione più verosimile, e non per abbassare i tempi d'attesa. I parametri sono:

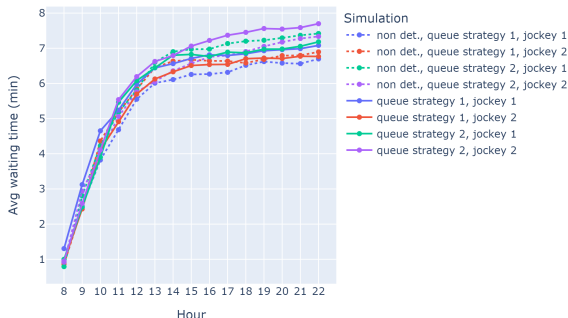
- 20 casse standard
- 6 casse self-service
- 5 casse self-scan
- Probabilità cliente self-scan: 50%
- Errore di stima della quantità di elementi nei carrelli con 0.1 di deviazione standard
- 4 strategie di scelta della coda
- 2 strategie di jockey

Per un totale di 8 simulazioni.



# Simulazione non deterministica

Average waiting time - probabilistic vs deterministic



I tempi medi sono paragonabili per tutte le strategie, osserviamo quindi che i parametri randomici introdotti sono pensati con il buon senso e non scombinano il modello. Servono a simulare più scenari possibili.

# Conclusioni

- Il modello di supermercato che abbiamo illustrato è basato su agenti ed ha lo scopo di simulare il comportamento dei clienti in fase di scelta della coda.
- Un supermercato è un sistema complesso composto da diversi attori che hanno obiettivi diversi. Simularlo con un modello ad agenti permette di indagare sulle configurazioni di casse e le strategie migliori che portano a una diminuzione del tempo d'attesa medio.
- A causa dell'emergenza di COVID-19 è necessario analizzare i flussi di persone, le distanze interpersonali e i tempi medi passati nei luoghi chiusi la cui frequentazione è necessaria nella vita di tutti i giorni.

## Conclusioni - sviluppi futuri

- **Introduzione dell'elemento spaziale:** nel nostro modello i clienti non hanno la concezione di distanza, si muovono in maniera istantanea da una cella all'altra.  
Si può considerare una configurazione di casse a D-fork, in cui si calcola la distanza per trovare anche la coda più vicina.  
Questo aiuterebbe a condurre uno studio sul distanziamento sociale nei supermercati.
- **Introduzione di diverse strategie:** avendo adottato il pattern Strategy è molto semplice definire e utilizzare nuove strategie di scelta della coda e jockey.

## Conclusioni - sviluppi futuri

- **Dati sui supermercati italiani:** è possibile utilizzare il modello con dati diversi; ad esempio per le casse self-scan si potrebbe fare uno studio più approfondito con dei dati a disposizione.
- **Criticità:** in caso di densità alta le code si riempiono al massimo, causando un'attesa prolungata dei clienti. È giusto prevedere una capienza massima ma si potrebbe introdurre una nuova zona di attesa soprattutto quando si utilizza una coda condivisa.