Università degli Studi di Milano-Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione

Sistemi Complessi: Modelli e Simulazione

Simulazione e Confronto tra due Tipologie di Coda al Supermercato

Francesco Stranieri 816551
Davide Marchetti 815990
Mattia Vincenzi 860579

Anno Accademico 2019 - 2020

Indice

1 Introduzione

Un *supermercato* è una grande organizzazione di vendita limitata a generi alimentari e a prodotti per uso di casa, o personale. Esso consiste in un grande ambiente che può occupare un'area di vendita dai 400 ai 1500 m^2. E' suddiviso tra più espositori nei quali la merce è esposta, in modo che il cliente abbia larga possibilità di scelta e possa servirsi da solo, pagando la merce acquistata prima dell'uscita.

L'emergenza sanitaria legata al *nuovo coronavirus*, ha determinato un'impennata delle vendite nei supermercati. Secondo Nielsen, le vendite tra lunedì 24 febbraio e domenica 1 marzo hanno registrato una crescita del $12,2\%$ [1]. Questo ha portato ad un vero e proprio assalto ai supermarket, provocando una notevole crescita nel numero totale di persone all'interno di essi. Come conseguenza, le già consistenti *code* sono sensibilmente aumentate, causando non pochi disagi. Sono quindi nate diverse soluzioni, atte a contrastare questo fenomeno [2] [3].

Questo progetto si pone l'obiettivo di studiare, e confrontare, le differenze tra le tipologie di code quotidianamente utilizzate all'interno dei supermercati [4]. In Italia, sono per lo più diffuse due diverse filosofie, come mostrato in Figura 1:

- **Classic** (1a): rappresenta la modalità più classica; è quindi presente un numero di code pari al numero di casse aperte in quel preciso momento.
- **Snake** (1b): rappresenta invece la modalità 'a serpentone'; si compone di un unico ingresso per l'accesso ad una coda comune, la quale porta all'uscita che permette il raggiungimento delle casse.

Gli elementi chiave utilizzati per confrontare le diverse tipologie di code sono: *il numero totale, e medio, di persone in coda per ogni istante temporale*, *il tempo medio di attesa in coda*, *il tempo medio di permanenza* e il *comportamento delle persone all'interno del supermercato*.

Per modellare lo scenario oggetto di studio è stato utilizzato il framework open-source MESA [5], sviluppato in Python. MESA consente di realizzare in maniera rapida ed intuitiva modelli basati su agenti, utilizzando le componenti integrate come la griglia e lo scheduler degli agenti; permette inoltre una visualizzazione, basata su un'interfaccia web, per l'analisi dei risultati.

(a) Modalità **Classic**.

(b) Modalità **Snake**.

Figura 1: Le due modalità implementate.

2 Modello

Il *modello* sviluppato mira a simulare le dinamiche interne ad un supermercato. Dal punto di vista grafico, quest'ultimo è composto da un insieme di celle, ognuna avente un'area pari a $1\,m^2$. Questa misura è giustificata dal fatto che una singola persona occupa, mediamente, uno spazio inferiore a 0.9 m [6]. L'insieme delle celle compone la *griglia* (`SingleGrid`). Lo spazio tra gli scaffali è di 3 m. Ogni corsia, alla quale ci si riferisce per convenzione con un numero da 1 a 5, riesce a garantire così un agevole passaggio per i clienti. Anche la distanza tra una cassa e l'altra risulta essere pari a 3 m. Nella sola modalità **Snake**, gli spazi destinati all'ingresso e all'uscita hanno, a loro volta, un'ampiezza pari a 3 m. Tali misure sono riportate in Figura 2.

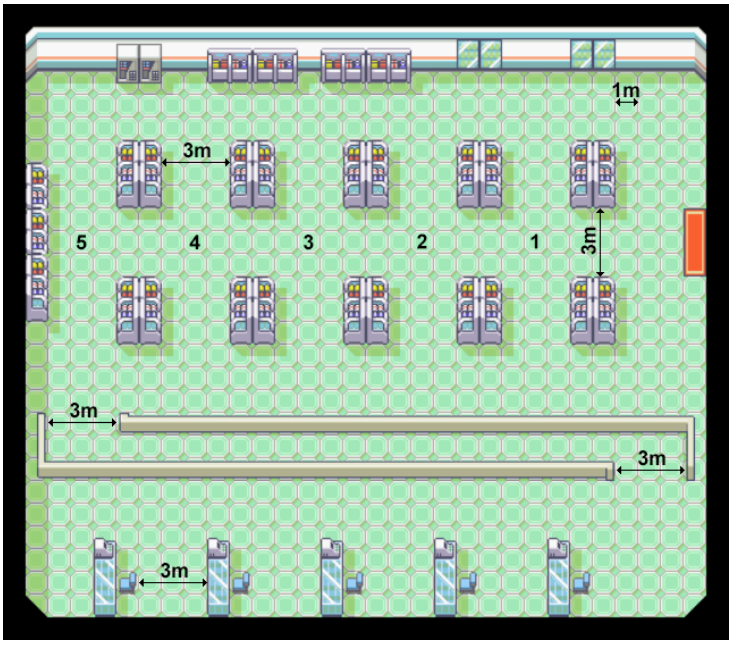

Figura 2: Misure della mappa **Snake**.

Per riprodurre un movimento credibile degli agenti, ogni passo temporale del modello, denominato *step*, sarà pari a 1 secondo. Questa fondamentale assunzione risulta essere alla base delle scelte implementative dell'intero progetto. Ogni cliente avrà perciò una velocità pari a 1 m/s [7], dal momento che ad ogni step (1 s) si muoverà al più di una cella (1 m). Per simulare

l'affluenza e la defluenza dei clienti, il periodo simulato equivale a 2 ore, cioè 7200 step.

2.1 Struttura

La struttura del supermarket viene caricata tramite un apposito file `txt`. Tale file è strutturato in modo tale da fornire al modello alcune informazioni essenziali:

- La prima riga specifica la *dimensione* della griglia.
- Sulla seconda riga è presente un parametro dipendente dall'area, ossia la *capienza massima*.
- Infine, nelle restanti righe, è presente la *mappa* vera e propria. Questa è composta da un insieme di simboli, ognuno con la propria semantica, riportati in Tabella 1. I punti per la generazione degli agenti nell'area casse sono necessari in quanto la fase di acquisto viene modellata dal modello come una *black-box*.

In questo progetto è stato modellato un supermercato di medie dimensioni, con un'area totale pari a $32x28$ celle, una capienza massima di 35 persone e un numero di corsie pari a 5. Dal momento che esiste una corrispondenza $1:1$ tra il numero di corsie e il numero di casse, queste ultime saranno 5.

Simbolo	Descrizione
`X`	Ostacolo
`0`	Zona libera
`1..N`	Cassa
`A..J`	Punto di generazione
`S`	Ingresso per la coda Snake
`Z`	Uscita dalla coda Snake

Tabella 1: Simboli utilizzati per la struttura della mappa.

2.2 Floor Field

Nella modalità **Classic**, la scelta della destinazione ottimale, ossia della cassa più vicina e con il minor numero di persone in coda, avviene grazie all'utilizzo

dei *floor field*, a livello tattico. Nel dettaglio, l'implementazione prevede l'associazione di un floor field a ciascuna cassa q, dove il valore di ogni cella p, viene calcolato come una distanza Euclidea:

$$d\left(p,q\right)=\sqrt{\sum_{i=1}^{n}\left(q_i-p_i\right)^2}. \tag{1}$$

In corrispondenza di un ostacolo la distanza risulterà essere $+\infty$. Un esempio di tale implementazione è mostrato in Figura 3.

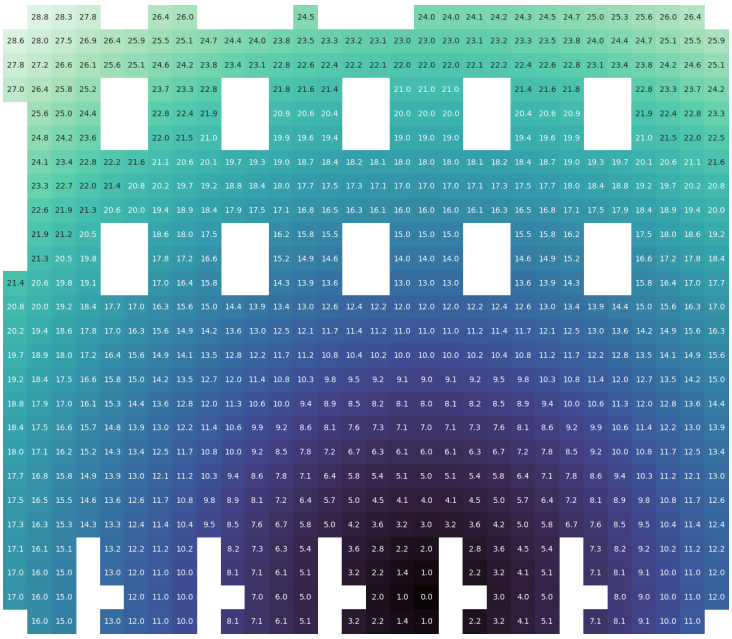

Figura 3: Floor field associato alla cassa 2.

2.3 Step del Modello

Ad ogni step del modello vengono eseguite, nell'ordine, le due operazioni dettagliate in seguito. Tra di esse viene inoltre eseguito lo step di ciascun agente (Sezione 3.6) e, successivamente, vengono invocate le funzioni necessarie alla computazione delle metriche di interesse (Sezione 4.1). Questo è reso possibile grazie all'utilizzo di uno *scheduler* (`BaseScheduler`).

Ingresso degli Agenti

L'ingresso degli agenti è stato pensato in modo tale da ottenere 3 diverse stadi:

- Un *riempimento* graduale del supermercato.
- Il raggiungimento di un *picco*.
- Un progressivo *svuotamento* dello stesso.

Ad ogni step del modello, viene generato un numero $p_1 \in [0; 1]$, secondo la seguente equazione:

$$p_1 = -\frac{\cos\left(\frac{t\pi}{3600} + 1\right)}{2} + \frac{1}{2}. \tag{2}$$

Successivamente viene estratto un numero casuale $p_2 \in [0; 1]$. Se $p_2 \leq p_1$, allora l'agente verrà creato con una probabilità di $0,85$ ed effettuerà l'ingresso nel supermercato.

La Figura 4 mostra il grafico relativo all'Equazione (2). E' possibile notare come la probabilità di ingresso tende a crescere, fino al raggiungimento del picco, quando il numero di step è circa 2450; questo garantisce un'afluenza sempre maggiore. Superato questo valore, tale probabilità tenderà a scendere, determinando così una situazione diametralmente opposta.

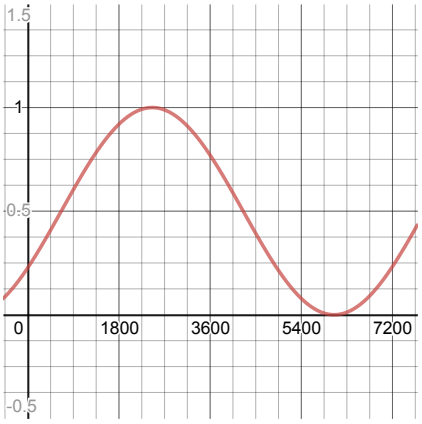

Figura 4: Grafico della funzione utilizzata per l'ingresso degli agenti.

Gestione delle Casse

In concomitanza dell'ingresso dei clienti all'interno del supermercato, le casse dovranno essere gestite in maniera efficace. Questo avviene in accordo alla percentuale di avanzamento della simulazione:

$$sim\_percentage = \frac{step \,\%\, steps\_in\_day}{steps\_in\_day} \tag{3}$$

e all'indice *ICN* (Ideal Customer Number):

$$ICN = (|open\_cashiers| + 1) \cdot queue\_length\_limit \tag{4}$$

Questo indice calcola il numero ideale di persone gestibili con l'apertura di una nuova cassa, rispetto alla lunghezza di un'ipotetica fila che dipende dalla capacità. Nel progetto, questa lunghezza è stata settata con un valore pari a 5.

L'apertura e la chiusura sono state progettate in modo tale da garantire un numero di casse aperte adeguato al *numero di clienti presenti in quel determinato momento*.

La funzione *IOC* (Ideal Opened Cashiers), definita in Tabella 2, associa un numero ideale di cassieri aperti data la percentuale di avanzamento della simulazione.

In conclusione, l'apertura di una nuova cassa avverrà solo se:

$$|open\_cashiers| < (IOC(sim\_percentage)) \wedge (current\_agents > ICN). \tag{5}$$

sim_percentage	**IOC(sim_percentage)**
$[0; 0.265] \cup [0.88; +\inf)$	2
$(0.265; 0.36] \cup [0.765; 0.88)$	3
$(0.36; 0.44] \cup [0.66; 0.765)$	4
$(0.44; 0.66)$	5

Tabella 2: Output della funzione IOC.

Per la chiusura è applicabile il ragionamento opposto. Qualora una cassa con ancora delle persone in coda debba chiudere, questa dovrà prima provvedere a smaltirle e, al contempo, non potrà essere scelta come destinazione dal resto dei clienti. E' importante sottolineare come il numero minimo di cassieri operativi rimane fissato a 2.

3 Agenti

Le classi di *agenti* considerate nella modellazione sono le seguenti:

- **Ostacolo** (`ObstacleAgent`): rappresenta una cella non calpestabile da un agente in movimento, come un muro o uno scaffale.
- **Cassiere** (`CashierAgent`): rimane fermo nella propria posizione in modo da poter servire i clienti. Un cassiere viene aggiunto allo scenario quando una cassa apre e rimosso quando questa chiude. Adiacente a questo agente viene posizionata la relativa cassa, ovvero il punto di pagamento verso il quale i clienti si dirigono prima di uscire. Questa rappresenta, come gli ostacoli, una zona non calpestabile del terreno ma contiene anche proprietà aggiuntive come il *tempo* per cui deve rimanere aperta e lo *stato* attuale, aperta o chiusa.
- **Cliente** (`CustomerAgent`): esso può muoversi all'interno dell'area casse. Nella modalità **Snake** ha un obiettivo molto semplice, ovvero quello di accodarsi e aspettare che gli venga assegnata una cassa, una volta raggiunta la fine della coda. Nella modalità **Classic**, invece, il cliente ricerca quella che è la soluzione che ritiene migliore per poter raggiungere una cassa aperta nel minor tempo possibile. Nell'effettuare questa questa scelta tiene in considerazione diversi fattori tra cui la distanza da ogni cassa e il numero di persone nelle corrispondenti code. Possiede diversi attributi, utili all'esecuzione del proprio compito, tra cui la *fase* attuale (Sezione 3.1), il *numero di prodotti* da acquistare (Sezione 3.2) e la prossima *destinazione*.

Nelle sezioni successive gli agenti considerati saranno esclusivamente i clienti, in quanto sono gli unici che hanno la possibilità di muoversi e di accodarsi.

3.1 Fasi dell'Agente

Un cliente, durante la permanenza all'interno del supermarket, può trovarsi in una delle diverse *fasi*, ognuna delle quali corrisponde ad un obiettivo che dovrà perseguire.

Classic

In Figura 5 vengono mostrate le fasi dell'agente nella modalità **Classic**:

- **Shopping**: il cliente si trova tra gli scaffali e sta facendo la spesa. Essendo il supermercato gestito come una *black-box*, il cliente non viene mostrato sulla mappa durante questa fase. Il tempo trascorso in questa fase è dipendente dal numero di prodotti che esso dovrà acquistare, come verrà motivato nella Sezione 3.3. Una volta che il cliente ha terminato la spesa passa nello stato **Reaching Queue**.
- **Reaching Queue**: il cliente è appena entrato nella zona casse attraverso una delle 5 corsie e valuta, in base alla *distanza* e al *numero di persone in ogni coda*, quale sia la destinazione ottimale per minimizzare il tempo di attesa. Una volta scelta la destinazione l'agente cerca di raggiungerla, avendo comunque la possibilità di cambiare la propria decisione a fronte di variazioni della situazione circostante (Sezione 3.6). Una volta raggiunta una coda entra nello stato **In Queue**.
- **In Queue**: il cliente si trova in coda ed attende il proprio turno per essere servito. Ad ogni step si muove in avanti, qualora sia libera la cella per il movimento. Una volta raggiunta la cassa, il cliente passa nello stato **Paying**.
- **Paying**: l'acquirente, una volta terminato il pagamento, esce dal supermercato. La stima del tempo trascorso in questa fase verrà commentato nella Sezione 3.4.

Snake

In Figura 6 vengono riportate le fasi di vita dall'agente nella modalità **Snake**. Di seguito verranno descritte solamente quelle fasi che differiscono dal caso precedente:

- **Shopping**.
- **Reaching Queue**: il cliente si muove verso l'ingresso della coda a serpentone; una volta raggiunto entra nello stato **In Queue**.

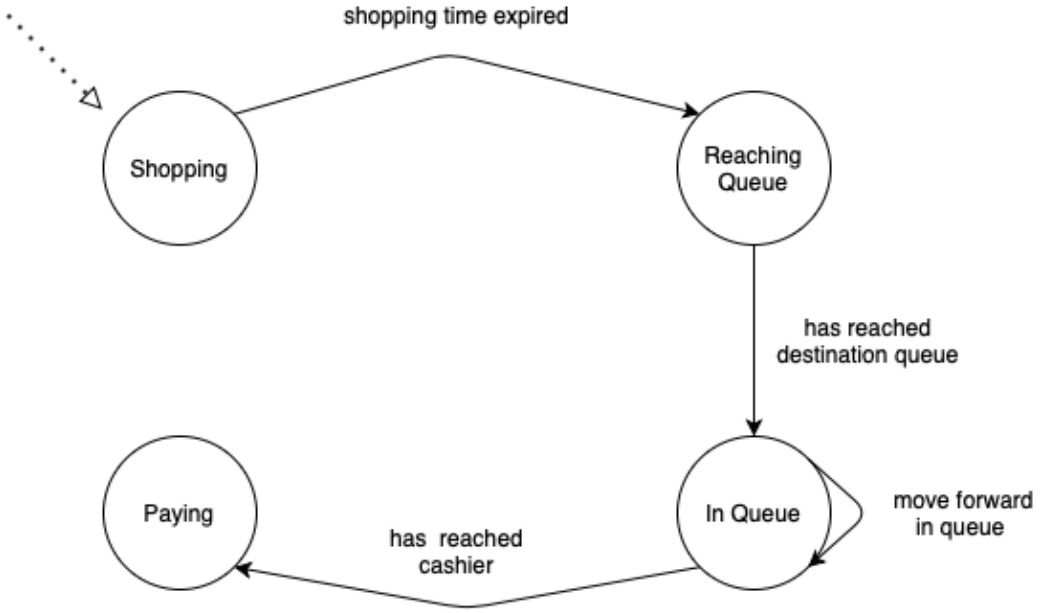

Figura 5: *Fasi dell'agente in modalità* ***Classic****.*

- **In Queue**: il cliente si muove fino alla posizione di inizio della coda, se presente. Una volta raggiunta rimane fermo e si muove seguendo il movimento dei clienti che lo precedono. L'agente rimane in questa fase fino all'uscita del serpentone, passando successivamente nello stato **Reaching Cashier**.
- **Reaching Cashier**: l'acquirente rimane in questa fase fino a quando raggiunge la prima cassa libera a lui assegnata, passando così nello stato **Paying**).
- **Paying**.

3.2 Numero di Prodotti

Un parametro importante che viene definito al momento della creazione di un `CustomerAgent` è il numero di prodotti che esso dovrà acquistare. Questo viene utilizzato per il calcolo del tempo necessario a fare la spesa e per il tempo necessario al pagamento.

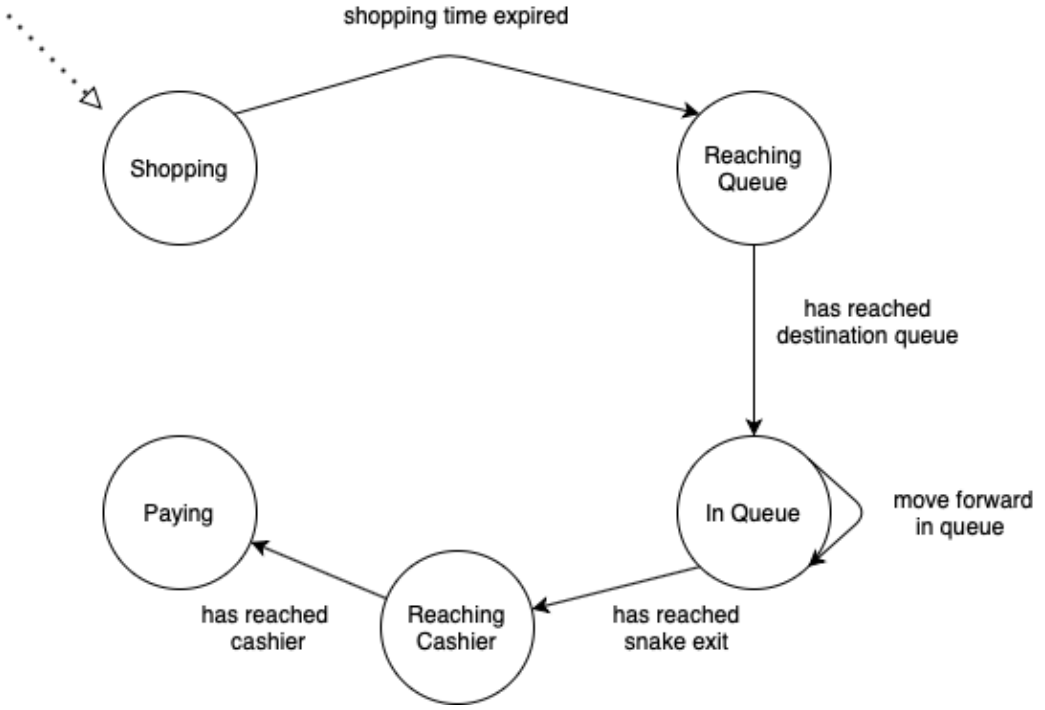

Figura 6: Fasi dell'agente in modalità **Snake**.

Questo parametro, denominato `product_count`, viene estratto da una *distribuzione normale* con media e deviazione standard espresse in funzione della capienza massima. Il valore viene successivamente arrotondando per eccesso, in quanto il numero di prodotti deve essere necessariamente intero.

La normale viene espressa nel seguente modo:

$$N \sim (\mu = capienza/2, \sigma = capienza/4), \tag{6}$$

dove la media μ e la deviazione standard σ sono state scelte dopo un'opportuna fase di calibrazione.

In Figura 7 è possibile notare che gli estremi di tale distribuzione sono stati *troncati*, in modo da scartare i valori minori di 1 (numero minimo di prodotti acquistabili) e maggiori di un limite superiore, stabilito a 35.

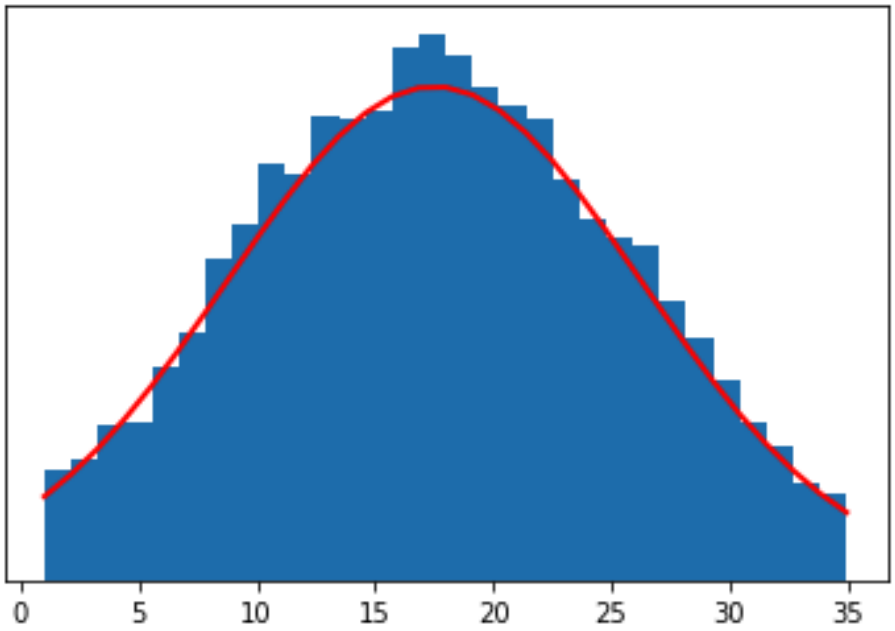

Figura 7: Distribuzione normale troncata, con $\mu = 17.5$ e $\sigma = 8.75$.

3.3 Tempo di Shopping

Alla creazione di un cliente, e sulla base del numero di prodotti ad esso associati, viene calcolato quello che è lo *shopping time*, ovvero il numero di step che l'agente impiegherà per la fase di **Shopping**. Durante questa fase viene stimato il tempo che il cliente trascorrerà tra gli scaffali, cioè all'interno dell'area considerata dal modello come una black-box.

La formula utilizzata per stimare questo tempo, dipendente dal parametro `product_count`, risulta essere la seguente:

$$shopping\_time = 3 + product\_count * 0.75$$

In Tabella 3 vengono mostrati alcuni dei possibili tempi, calcolati al variare del numero di prodotti. Ad esempio, un cliente impiegherà 630 secondi (circa 10 minuti) per acquistare un totale di 10 prodotti.

3.4 Tempo di Pagamento

Un altro parametro che viene assegnato ad un cliente nel momento della sua creazione è il *paying time*, ovvero il tempo che l'agente trascorrerà nella fase **Paying**. Questo rappresenta un'approssimazione del tempo impiegato per svuotare il carrello, insacchettare i prodotti ed effettuare il pagamento; durante questo periodo il cliente sarà posizionato nella casella adiacente alla cassa, in attesa di poter abbandonare il supermercato.

Questo tempo, dipendente a sua volta da `product_count`, viene computato nel seguente modo:

$$paying\_time = 1 + product\_count * 0.25$$

In Tabella 3 vengono mostrate alcuni stime di tale tempo, calcolate al variare del numero di prodotti. Ad esempio, supponendo che un cliente abbia appena acquistato 10 prodotti, allora impiegherà 210 secondi (circa 3 minuti) per ultimare la fase di pagamento.

Numero di Prodotti	**Shopping Time (in secondi)**	**Paying Time (in secondi)**
1	225	75
2	270	90
3	315	105
4	360	120
5	405	135
6	450	150
7	495	165
8	540	180
9	585	195
10	630	210
11	675	225
12	720	240
13	765	255
14	810	270

Tabella 3: Tempi stimati per lo shopping e il paying time, al variare del numero di prodotti.

3.5 Area Casse

I clienti, una volta terminata la spesa, devono accedere all'*area casse*, in modo da poter pagare per poi uscire dal supermarket. La generazione degli agenti, in corrispondenza di una delle 5 corsie, non sarà però uniforme, in quanto sarà molto probabile che un cliente le percorra tutte, o quasi.

Difatti, nei supermercati, prodotti di categorie diverse sono in corsie differenti e spesso, quando ci si reca a fare la spesa, si ha la necessità di acquistare prodotti di vario tipo. Inoltre, la disposizione degli espositori e degli articoli è simile nei vari supermercati, in modo da predeterminare il percorso dei clienti, mostrando loro più prodotti di quelli che sarebbero inizialmente intenzionati ad acquistare [8].

Su questa ipotesi, un agente può accedere all'area casse tramite una determinata corsia, definita dal valore di probabilità riportato in Tabella 4.

Corsia	**Probabilità di Accesso**
1	10%
2	10%
3	20%
4	25%
5	35%

Tabella 4: Probabilità di accesso alla zona casse, in corrispondenza di una specifica corsia.

3.6 Step dell'Agente

L'obiettivo generale dell'agente consiste nell'eseguire una scelta ottimale nella decisione della destinazione, cercando di minimizzare il tempo necessario per il raggiungimento di una cassa. La scelta della destinazione implica, di conseguenza, il successivo movimento che l'agente andrà a compiere. Questa risulta essere una fase critica e viene gestita in maniera differente, in base alla tipologia di coda considerata.

Classic

Nella modalità **Classic**, una volta che il cliente entra nell'area casse (fase **Reaching Queue**) deve scegliere una destinazione, ovvero una cassa verso la quale dirigersi, cercando di eseguire una scelta locale ottima.

La Figura 8 mostra i passaggi che articolano tale scelta:

- **Find Best Move**: in questo primo passo vengono considerate le 8 celle più vicine, le quali compongono il *vicinato di Moore*. Per ognuna delle casse aperte si considerano i valori di queste 8 celle, prendendo come riferimento i floor field, descritti nella Sezione 2.2. Per ogni cassa viene estratta la cella che minimizza la distanza a partire dall'attuale posizione dell'agente. A questa distanza viene successivamente sommato un valore, equivalente al numero di persone in coda in quella determinata cassa. In questo modo ogni possibile destinazione, cioè ogni cassa aperta, ha associato un **costo**, espresso in funzione della distanza e della *congestione*. Viene quindi selezionata come destinazione la cassa che minimizza il costo complessivo.

- **Update Objective**: ora l'agente può rivalutare, per un numero finito di volte, la decisione presa. Viene confrontato il nuovo obiettivo con il precedente e, qualora questi due non coincidano, l'agente aggiorna la destinazione con una probabilità del 25%. Questa probabilità è stata aggiunta in modo da rendere più realistica la simulazione e impedendo i continui cambiamenti di corsia che possono condurre ad effetti confusionari.

- **Find Queue Start Position**: sulla base della destinazione scelta, viene individuata la posizione di inizio della coda, ovvero quella verso cui l'agente dovrà dirigere il proprio movimento.

- **Next Move**: con quest'ultimo passo l'agente effettuerà il movimento verso l'inizio della coda, in corrispondenza della cassa scelta come destinazione.

Queste operazioni vengono eseguite fino a quando il cliente si accoda definitivamente. Una volta raggiunta la coda esso si muoverà in avanti con l'avanzamento della stessa.

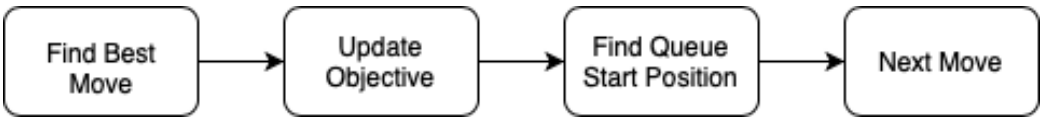

Figura 8: Scelta della destinazione nella modalità **Classic**.

Snake

Nella modalità **Snake** non è più il cliente a dover scegliere la propria destinazione sulla base delle proprie percezioni locali come nella modalità **Classic**, ma sarà l'ambiente ad assegnargli un obiettivo. L'agente dovrà solamente effettuare il movimento verso la destinazione.

L'agente avrà assegnato una diversa destinazione, sulla base della fase in cui si trova. In **Reaching Queue** questa equivarrà all'ingresso della coda a serpentone, mentre in **In Queue** all'uscita. Infine, in **Reaching Cashier** la destinazione sarà uguale alla prima cassa libera che l'ambiente assegnerà all'agente.

In Figura 9 sono mostrati i passaggi che l'agente esegue, in corrispondenza delle tre fasi appena citate:

- **Find Best Path to Destination**: il cliente ricerca il percorso ottimale, dalla propria posizione verso la destinazione assegnata, attraverso l'algoritmo $A*$ [9].
- **Next Move**: una volta calcolato il percorso, l'agente si muoverà per raggiungere la destinazione, spostandosi nella successiva cella, relativa al percorso ottimale calcolato allo step precedente.

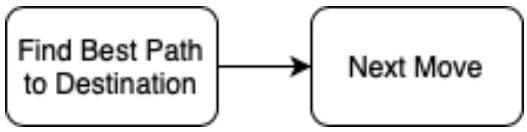

Figura 9: Scelta della destinazione nella modalità **Snake**.

4 Implementazione

Uno dei principali obiettivi riguardo la modellazione agent-based risulta essere il calcolo di *metriche rilevanti*. Queste permettono di condurre specifiche valutazioni, in modo da poter trarre delle conclusioni.

MESA offre un'apposita classe che consente di gestire la *raccolta dati*. Nello specifico, il suo **Data Collector** permette di memorizzare:

- Variabili a livello di *modello*.
- Variabili a livello di *agente*.
- Variabili non appartenenti alle due categorie precedenti, raggruppate in *tabelle*.

Le prime due vengono aggiunte al **Data Collector** tramite una specifica funzione di aggregazione.

4.1 Raccolta Dati

Le funzioni implementate nel progetto per la raccolta dati agiscono esclusivamente a livello di modello.

I dati, calcolati e raccolti ad ogni step del modello, risultano essere i seguenti:

- *Numero totale di agenti* presenti all'interno del supermercato.
- Numero di clienti in fase **Shopping**.
- Numero di clienti in fase **Paying**.
- Numero di clienti in fase **In Queue**.
- *Numero medio di agenti in coda* sul numero totale di casse aperte in quel preciso istante temporale.
- *Tempo medio trascorso in coda* dalle persone attualmente presenti nello scenario.
- *Tempo medio trascorso nel supermercato* dalle persone attualmente presenti.

Queste informazioni vengono salvate in un dizionario, in cui a ciascuno step viene associato il corrispondente valore prodotto dalle funzioni di aggregazione.

I dati raccolti vengono successivamente utilizzati per la comparazione dei risultati. Questo avviene tramite la realizzazione di specifici grafici che mostrano l'andamento delle variabili prese in esame, in funzione del numero di step. Tali grafici, presenti nella Sezione 5, hanno permesso di condurre un'approfondita analisi tra le due metodologie di coda implementate.

4.2 Mappa

La visualizzazione classica offerta da MESA è stata personalizzata in modo tale da ottenere una rappresentazione grafica più familiare all'ambiente di un supermercato, grazie all'utilizzo di un *tileset*. Questa modifica ha comportato sostanziali cambiamenti in alcuni file del framework, i quali sono stati riscritti in virtù delle nuove necessità.

In aggiunta sono state applicate alcune ottimizzazioni atte a ridurre la complessità, per ogni step, del processo di visualizzazione della mappa.

Tileset

Un tileset consiste in un'immagine composta da celle quadrate, chiamate *tile*, aventi tutte la stessa dimensione. Durante la progettazione della mappa queste sono state combinate in diversi modi, al fine di realizzare un ambiente più completo.

Ad esempio, la Figura 10 mostra come è possibile comporre alcuni tile in modo da ottenere un ostacolo.

Figura 10: Scaffale composto da tile distinti.

Ambiente

La rappresentazione grafica dell'ambiente è basata su un tileset di *Pokémon* [10]. Quando necessario, come ad esempio per la modellazione del percorso a serpentone, si è intervenuti manualmente per personalizzarlo. L'intero tileset utilizzato è riportato in Figura 11.

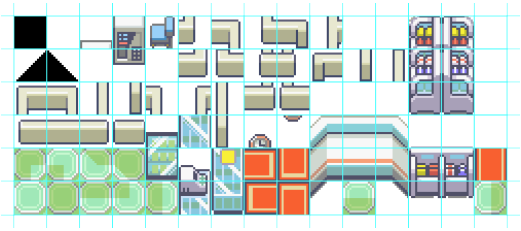

Figura 11: Tileset utilizzato per la rappresentazione dell'ambiente.

Per la composizione della mappa è stato utilizzato un software open-source chiamato *Tiled* [11], il quale ha permesso inoltre di esportarla in un formato utile per la successiva visualizzazione sulla griglia.

Personaggi

La stessa logica è stata utilizzata per i clienti. Essi vengono difatti visualizzati sulla griglia come personaggi dalle sembianze umane. Durante la creazione di ciascun agente gli viene associata un'immagine, scelta casualmente da un insieme predefinito. Questo prevede sei personaggi, illustrati in Figura 12, distinguibili per il genere e per l'età.

Figura 12: Insieme dei personaggi disponibili.

4.3 Diagramma delle Classi

Il *diagramma delle classi*, illustrato in Figura 13, presenta la struttura del codice del progetto. MESA fornisce alcune classi che possono essere estese, agevolando così la scrittura del modello, dal momento che bisogna implementare la sola logica applicativa.

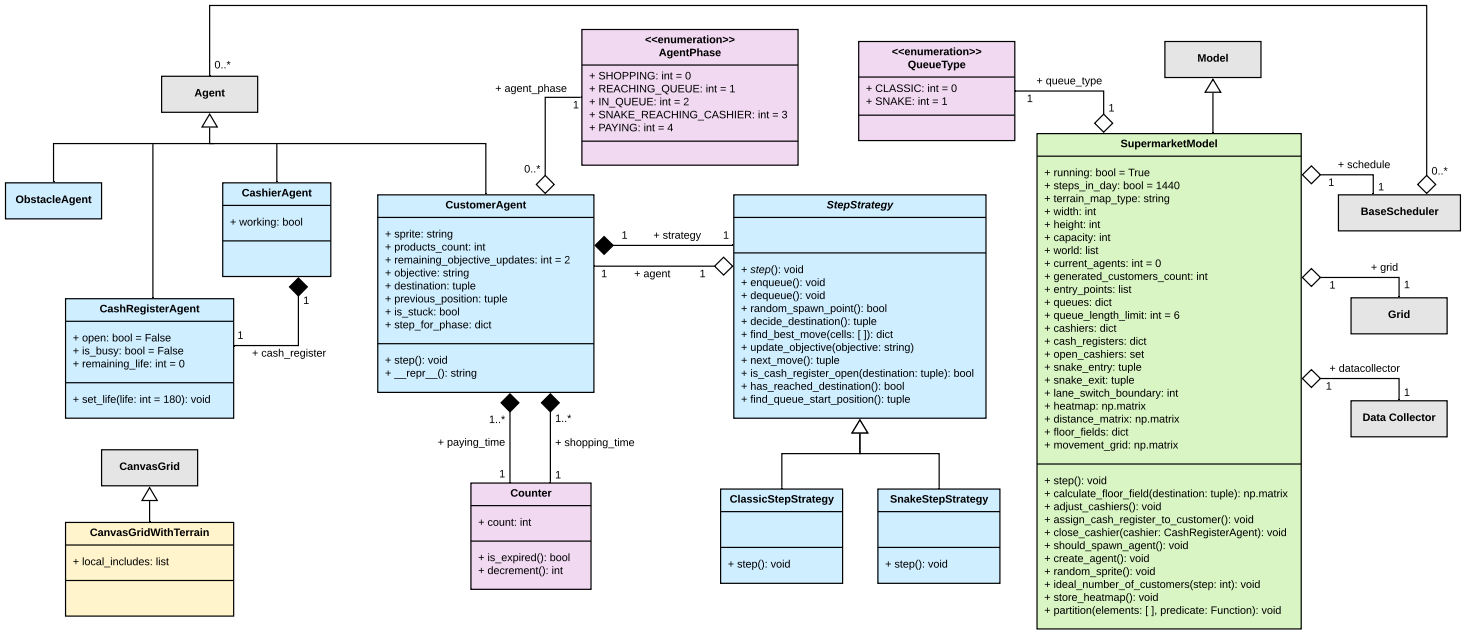

Figura 13: Diagramma delle classi.

Per semplificare la lettura dello schema, sono state dettagliate esclusivamente le classi implementate autonomamente nel progetto, evidenziandole con colori di sfondo diversi in base alla propria finalità:

- **Verde**: classe principale del modello; coordina l'interazione tra MESA e gli agenti.
- **Azzurro**: definisce gli agenti e le strategie legate al movimento.
- **Grigio**: permettono la simulazione del modello attraverso MESA.
- **Giallo**: visualizzazione e controllo dell'interazione tramite un'interfaccia web.
- **Viola**: utilità ed enumerazioni.

Strategy Pattern

La gestione delle due tipologie di coda presenta, a livello implementativo, diverse istruzioni in comune; tuttavia un agente dovrà comportarsi diversamente a seconda della tipologia scelta.

A tal proposito, si è scelto di implementare il design pattern *Strategy* [12]. La dichiarazione di una classe comune, chiamata `StepStrategy`, permette di astrarre i comportamenti dell'agente in sottoclassi esterne. Questo consente di ridurre notevolmente la complessità del codice, andando ad isolare i dettagli implementativi del movimento.

5 Analisi dei Risultati

Le metriche utilizzate per analizzare il comportamento delle due tipologie di coda sono state calcolate sulla base dei dati raccolti dal **Data Collector**.

5.1 Piano Sperimentale

Le informazioni utilizzate per effettuare il confronto comprendono:

- Numero di clienti in fase **In Queue**.
- *Numero medio di agenti in coda* sul numero totale di casse aperte in quel preciso istante temporale.
- *Tempo medio trascorso in coda* dalle persone attualmente presenti nello scenario.
- *Tempo medio trascorso nel supermercato* dalle persone attualmente presenti.

Queste metriche vengono rappresentate tramite dei grafici, integrati nell'interfaccia web di MESA e aggiornati in tempo reale durante la simulazione. Per renderne più immediata la lettura, sono stati raggruppati in un'unica visualizzazione i grafici con il numero di clienti in coda e il numero medio degli stessi. Le due curve assumono lo stesso andamento, seppur su scale differenti, e coincidono quando è presente un singolo agente in coda.

Al fine di ottenere *risultati equiparabili*, sono state adottate le stesse politiche per quanto riguarda, ad esempio, la generazione degli agenti, la probabilità di accesso all'area casse e la gestione dinamica di queste ultime. Infine, per garantire una corretta riproducibilità dei risultati tra le diverse esecuzioni, è stato impostato un *seed* per la generazione dei numeri casuali.

Classic

Nella simulazione in modalità **Classic** i clienti, una volta raggiunta l'area casse, valutano la situazione circostante al fine di decidere la migliore coda verso la quale dirigersi. Una volta accodati attendono il loro turno, in attesa di essere serviti.

In Figura 14 vengono mostrate i due grafici riguardanti il numero totale, e medio, di persone in coda. Le curve risultano essere coerenti con l'Equazione (2), utilizzata per modellare l'ingresso degli agenti. Una volta superata la

fase iniziale, corrispondente a circa 2000 step, la probabilità di ingresso comincia a crescere e questo incide sulla lunghezza delle code. Per far fronte a questa situazione, il supermercato comincia ad aprire, in maniera distribuita nel tempo, un numero sempre maggiore di casse. La curva arancione prosegue con un comportamento oscillante durante il periodo di maggior affluenza, quando il numero di persone entranti ed uscenti tende ad essere equivalente. In questo frangente gli agenti in coda arrivano ad essere 18, con un numero medio di circa 4 persone per ogni cassa. Successivamente, la stessa curva comincia dolcemente a decrescere; questo implica che il supermarket ha smaltito l'afflusso massimo di clienti e può cominciare a chiudere gradualmente le casse. Attorno ai 7200 step i due grafici tornano a coincidere. Le casse aperte tornano ad essere due, come nella situazione iniziale.

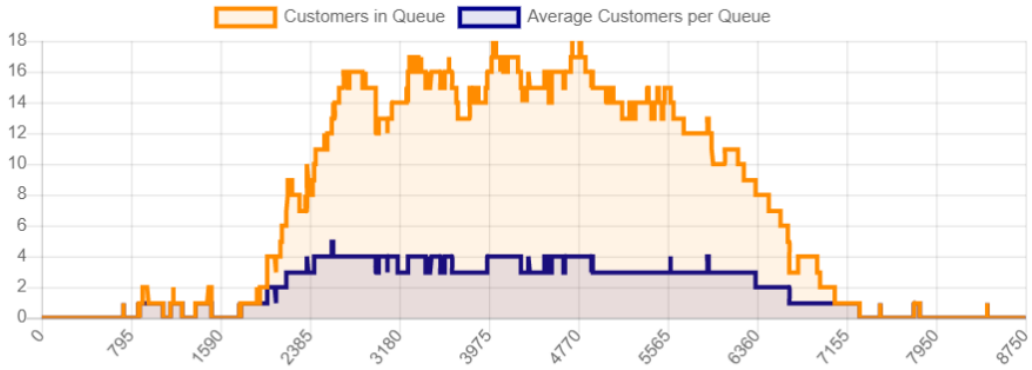

Figura 14: Numero totale, e medio, di agenti in coda nella modalità **Classic**.

Il grafico in Figura 15 mostra il tempo medio trascorso in coda nell'arco temporale simulato. Questo tempo continua a crescere anche dopo che è stato raggiunto il numero massimo di persone in coda, come riportato a circa 4770 step nel grafico visto precedentemente in Figura 14.

Questo comportamento è causato dai clienti entrati nel supermercato durante la fase di maggiore affluenza, i quali si riversano in seguito nell'area casse facendo così aumentare il tempo medio in coda. Quest'ultimo passa da circa 12 minuti fino ad un massimo di 19, raggiunto in concomitanza della progressiva chiusura delle casse.

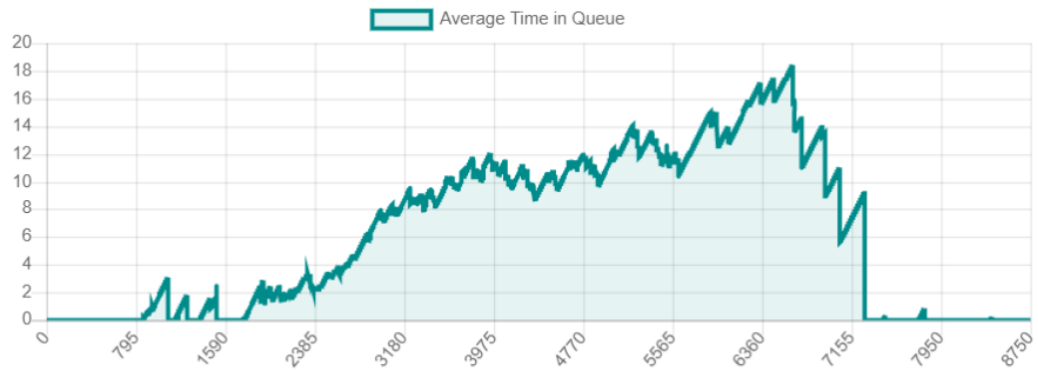

Figura 15: Tempo medio trascorso in coda nella modalità **Classic**.

Il grafico in Figura 16 ritrae il tempo medio trascorso all'interno del supermercato. L'andamento della curva è analogo a quello in Figura 15; una differenza riguarda però la discesa della stessa, la quale non va ad azzerarsi. Questo è causato dal fatto che sono ancora presenti dei clienti, seppur nessuno in coda. Questa decrescita porta il tempo medio da un culmine di 36 minuti ad un minimo di 11. La risalita finale è determinata dal numero crescente di agenti in ingresso, secondo l'Equazione (2).

Figura 16: Tempo medio trascorso nel supermercato nella modalità **Classic**.

In Figura 17 viene riportata la *heatmap* riguardante la simulazione. La distribuzione dei clienti, in corrispondenza delle casse, risulta essere coerente con la relativa probabilità di accesso, come descritto in Tabella 4. Le celle delle prime tre casse risultano difatti essere più intense. Questo è dovuto

al fatto che i clienti preferiscono le corrispondenti code, per una questione legata soprattutto alla vicinanza. Infine, è possibile notare come la cassa 2 risulta rimanere aperta per il minor tempo.

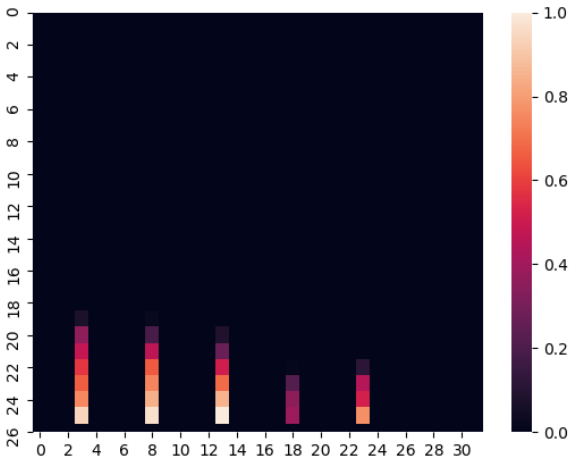

Figura 17: Heatmap relativa alla modalità **Classic**.

Snake

La simulazione in modalità **Snake**, invece, prevede un'unica coda comune per il raggiungimento delle casse. Al cliente, una volta raggiunta l'uscita dopo aver percorso il serpentone, verrà assegnata la prima cassa disponibile dove poter effettuare il pagamento.

Il grafico in Figura 18, raffigurante il numero totale di persone in coda, mostra una situazione di affollamento che si protrae dai 2400 fino ai 5500 step, ossia in misura minore rispetto alla modalità **Classic**. Superato questo periodo, la curva scende in maniera più ripida rispetto al caso precedentemente, segno che la fila viene processata più rapidamente. Il numero massimo di persone in coda risulta essere pari a 15, mentre quello medio si aggira sul 3.

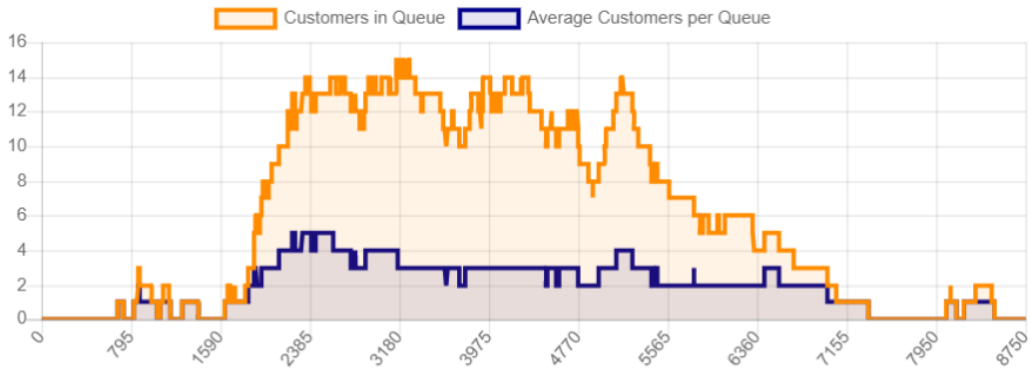

Figura 18: Numero totale, e medio, di agenti in coda nella modalità **Snake**.

Il grafico in Figura 19 descrive l'andamento del tempo medio trascorso in coda. La curva tende a crescere più o meno regolarmente, fino al raggiungimento di un attesa media massima pari a 15 minuti. L'azzeramento del tempo, a circa 7000 step, viene causato dall'ultimo cliente in coda, una volta che raggiunge la cassa per procedere al pagamento. Questo indica che non sono più presenti agenti nello stato **In Queue**.

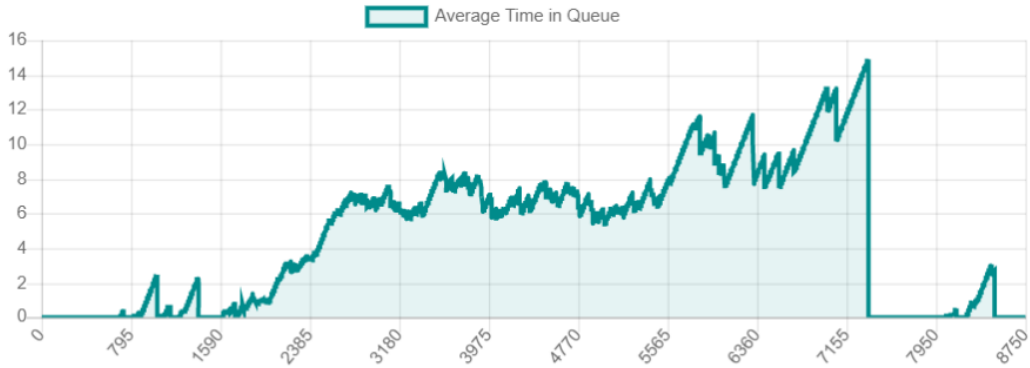

Figura 19: Tempo medio trascorso in coda nella modalità **Snake**.

Il grafico in Figura 20 mostra un profilo simile rispetto alla controparte **Classic**, anche se con una discesa più ripida una volta raggiunto il picco. Ne consegue che il tempo si abbassa notevolmente, passando da 36 minuti a circa 11.

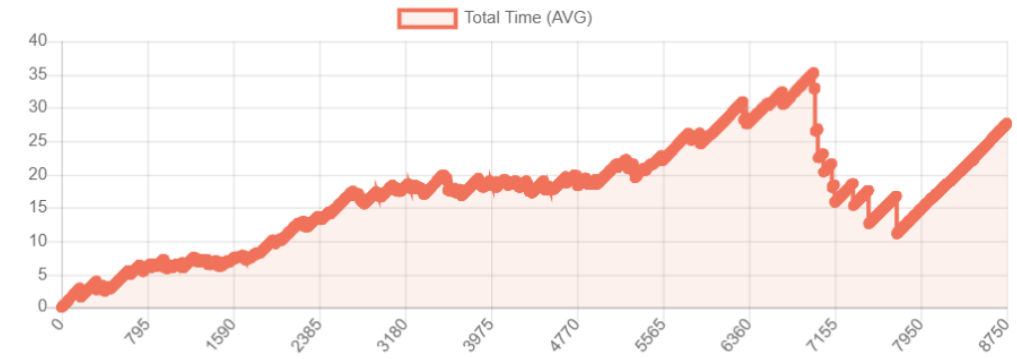

Figura 20: Tempo medio trascorso nel supermercato nella modalità **Snake**.

Infine, in Figura 21 viene riportata la *heatmap*. Questa mostra una maggiore concentrazione degli agenti in corrispondenza dell'uscita del serpentone, dove la coda si accumula maggiormente visto che i clienti devono attendere che gli venga assegnata una cassa libera. La cassa 3 e la cassa 5 risultano essere quelle con la maggior affluenza; questo poiché rimangono aperte per un tempo maggiore rispetto alle altre.

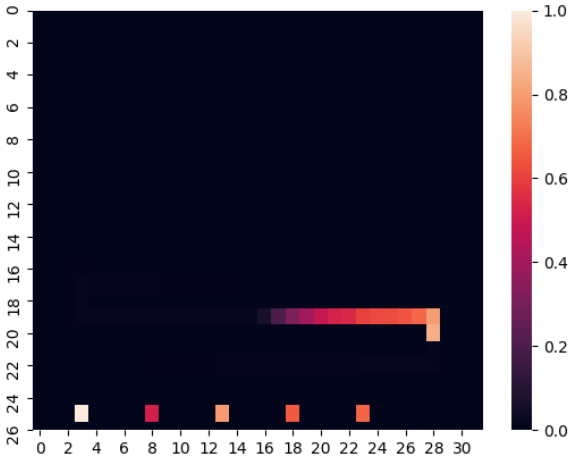

Figura 21: Heatmap relativa alla modalità **Snake**

6 Conclusioni

In questa sezione verranno confrontati i risultati ottenuti. E' importante sottolineare come questi sono stati derivati utilizzando le medesime politiche per entrambe le modalità, in modo da poter garantire un confronto robusto. Successivamente verranno presentati dei possibili miglioramenti, attuabili al modello attualmente impiegato.

6.1 Confronto Risultati

La Figura 22 mostra il numero di clienti in coda all'aumentare del numero di step. E' evidente come la fila nella modalità **Snake** cresce più velocemente rispetto alla controparte **Classic**, dal momento che questa risulta essere unica. Nonostante questo, la modalità Snake riesce a gestire più efficacemente l'afflusso dei clienti, soprattutto nel periodo di maggior carico. Tale comportamento è determinato dal fatto che nella modalità **Classic** i clienti hanno la possibilità di scegliere autonomamente la cassa verso la quale dirigersi, aumentando così la probabilità di effettuare una scelta non ottimale. La differenza tra le due strategie è particolarmente accentuata verso i 4770 step, quando il numero di clienti in coda nella modalità **Classic** risulta essere doppio rispetto alla rivale **Snake**, con valori pari rispettivamente a 18 e a 9. Con il numero minimo di casse aperte la velocità di smaltimento delle code risulta essere molto simile, come riportato dagli estremi del grafico.

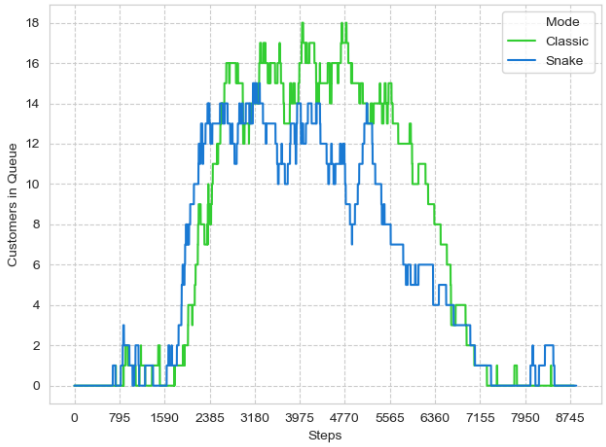

Figura 22: Numero di clienti in coda nelle due modalità.

Il grafico in Figura 23 riporta, invece, il confronto in termini di numero medio di clienti in coda. Il profilo delle due curve segue quanto visto precedentemente in Figura 22. Il valore puntuale viene difatti mediato sul numero di casse aperte in quello specifico istante.

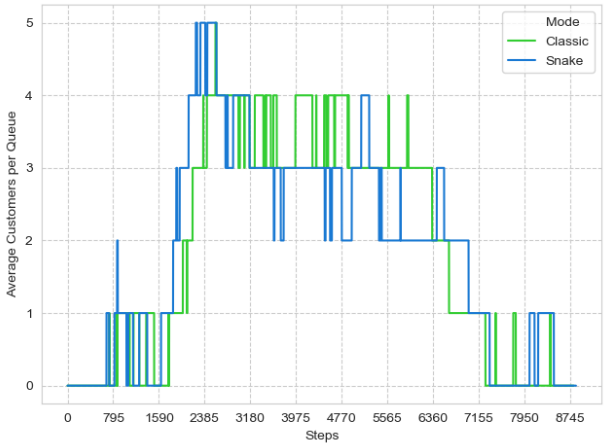

Figura 23: Numero medio di clienti in coda nelle due modalità.

In Figura 24 vengono confrontati i tempi medi di attesa in coda. Quando le due curve sono maggiormente distanti, il tempo nella modalità **Snake** risulta essere sensibilmente inferiore; nel punto di massima distanza, la differenza appare addirittura doppia, con valori pari a 18 e 9 minuti. Verso la fine della simulazione, l'azzeramento del tempo risulta essere più netto rispetto alla modalità **Classic**. Questo è dovuto al fatto che è presente un'unica coda e quando l'ultimo cliente in essa passa allo stato **Paying** azzera tale tempo.

Durante il periodo di maggior afflusso la modalità **Snake** permette di ottenere tempi di permanenza inferiori fino a 5 minuti, come visibile in Figura 25. In entrambe le modalità il picco assume un valore di circa 35 minuti, seppur in instanti temporali differenti. Questa situazione legata al picco si verifica quando il numero dei clienti in coda risulta essere in diminuzione. Nonostante possa sembrare contro intuitivo, tale comportamento risulta giustificabile dal fatto che gli ultimi clienti rimanenti in coda hanno accumulato un tempo di attesa dipendente dalle persone che li precedevano nella fila. Una volta che questi terminano il pagamento non incidono più sul calcolo del tempo, portandolo di conseguenza ad aumentare.

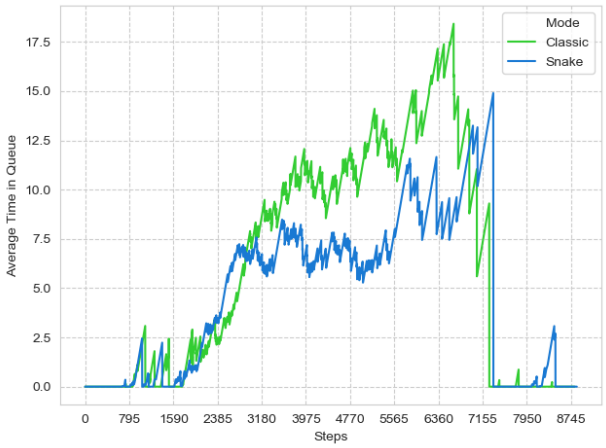

Figura 24: Tempo medio trascorso in coda nelle due modalità.

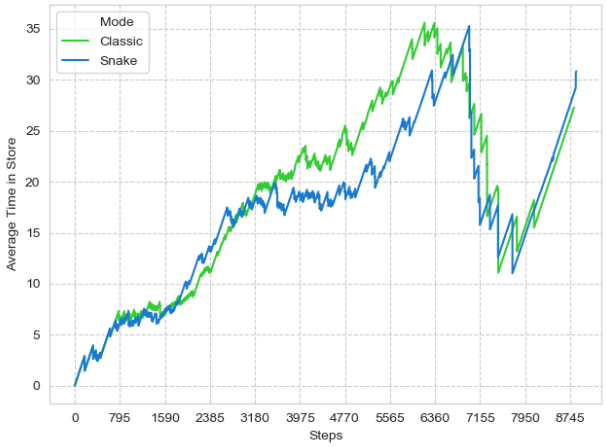

Figura 25: Tempo medio trascorso nel supermercato nelle due modalità.

6.2 Considerazioni Finali

L'obiettivo prefissato del progetto consisteva nello studiare, e confrontare, le differenze tra le tipologie di coda presenti all'interno dei supermarket italiani. Questo è stato reso possibile integrando i classici sistemi di servizio ($M/M/1$ e $M/M/N$), descritti nella *teoria delle code*, all'interno di un modello microscopico a spazio discreto rappresentato da un supermercato, assieme ad alcune semplici dinamiche pedonali.

Alla luce dei risultati ottenuti, e delle analisi svolte, è possibile concludere che la modalità **Snake** risulta essere più efficiente rispetto alla modalità **Classic**. Questo risultato trova conferma in letteratura [13], dove questo argomento viene ampiamente trattato. Difatti, confrontando i risultati ottenuti si evince che i tempi trascorsi in coda, e all'interno del supermercato, nella modalità **Snake** risultano essere mediamente inferiori, sebbene un'unica fila formata da più persone possa indurre ad un pensiero opposto.

I tempi ottenuti tramite la simulazione sono riconducibili ad una situazione reale [14], nonostante le assunzioni e le semplificazioni alla base del modello implementato.

6.3 Sviluppi Futuri

Nell'implementazione attuale, la fase di **Shopping** viene considerata come una black-box. Gli agenti non vengono quindi visualizzati sulla mappa fino al loro arrivo all'area casse. Questo aspetto potrebbe essere modellato, in modo da prevedere anche l'annesso movimento dei clienti tra gli scaffali.

Nel modello presentato i clienti sono considerati come aventi tutti le *medesime proprietà*. Di conseguenza presentano una velocità omogenea, a prescindere dall'età e dal numero di prodotti da acquistare. Una futura estensione potrebbe riguardare questo aspetto, in modo da assegnare ai clienti una velocità coerente alle proprie caratteristiche [7].

Anche la capienza massima del supermercato, assieme alle dimensioni dello stesso, potrebbero essere gestite dinamicamente. Il tutto potrebbe essere personalizzato attraverso l'interfaccia web di MESA, in modo da poter sperimentare diverse circostanze.

Nell'attuale implementazione i clienti possono cambiare la propria destinazione, corrispondente alla cassa verso la quale dirigersi, fino a quando non risultano **In Queue**. Un possibile sviluppo potrebbe consistere nel permettere agli agenti la possibilità di *cambiare coda* anche quando questi risultano già accodati, a fronte di rilevanti variazioni della situazione.

Un ultimo miglioramento potrebbe infine interessare la *gestione dei conflitti*. Allo stato attuale un agente considera solamente le celle non occupate quando deve effettuare un movimento. Potrebbe essere adottato un meccanismo di cooperazione tra gli agenti, in grado di gestire più efficacemente tutte le situazioni di conflitto.

Bibliografia

[1] Coronavirus: continuano a crescere gli acquisti nella gdo. https://www.nielsen.com/it/it/insights/article/2020/coronavirus-continuano-a-crescere-gli-acquisti-nella-gdo/. Accessed: 22-07-2020.

[2] Covid-19 e code al supermercato, da cinque ingegneri una app per evitarle. https://www.openinnovation.regione.lombardia.it/it/b/572/covidecodealsupermercatodacinqueingegneriunaappperevitarle. Accessed: 22-07-2020.

[3] Coronavirus, niente più code con ufirst: l'app che segnala quando sta per arrivare il tuo turno. https://www.corriere.it/tecnologia/servizi-digitali-per-lavoratori-genitori-famiglie/notizie/coronavirus-niente-piu-code-ufirst-l-app-che-segnala-quando-sta-arrivare-tuo-turno-1b4d183c-750a-11ea-b9c4-182209d6cca4.shtml. Accessed: 22-07-2020.

[4] Daichi Yanagisawa, Yushi Suma, Akiyasu Tomoeda, Ayako Miura, Kazumichi Ohtsuka, and Katsuhiro Nishinari. Walking-distance introduced queueing model for pedestrian queueing system: Theoretical analysis and experimental verification. *Transportation Research Part C: Emerging Technologies*, 37:238 – 259, 2013.

[5] Mesa: Agent-based modeling in python 3+. https://mesa.readthedocs.io/en/master/. Accessed: 22-07-2020.

[6] How to design a supermarket, the technical guide. http://biblus.accasoftware.com/en/how-to-design-a-supermarket-the-technical-guide/. Accessed: 22-07-2020.

[7] Addie Middleton, Stacy L. Fritz, and Michelle Lusardi. Walking speed: The functional vital sign. *Journal of Aging and Physical Activity*, 23(2):314 – 322, 2015.

[8] La scienza dei supermercati. https://www.startingfinance.com/approfondimenti/la-scienza-dei-supermercati/. Accessed: 22-07-2020.

[9] Xiang Liu and Daoxiong Gong. A comparative study of a-star algorithms for search and rescue in perfect maze. 04 2011.

[10] The spriters resource. https://www.spriters-resource.com/game_boy_advance/pokemonemerald/. Accessed: 24-07-2020.

[11] Tiled map editor. https://www.mapeditor.org/. Accessed: 24-07-2020.

[12] Strategy. https://refactoring.guru/design-patterns/strategy. Accessed: 24-07-2020.

[13] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S Trivedi. Queueing networks and markov chains: modeling and performance evaluation with computer science applications, 1998.

[14] How long consumers spend in food stores on average in canada as of october 2018. https://www.statista.com/statistics/944368/average-time-spent-in-food-stores-canada/. Accessed: 25-07-2020.