Progetto di Sistemi Complesi: Modelli e Simulazione

Supermarket Simulation

Tonelli Lidia Lucrezia - 813114

Marco Grassi - 829664

Gianluca Giudice - 830694

Contents

1 Introduzione **2**

2 Stato dell'arte **3**

3 Descrizione del modello **4**
3.1 Framework adottato e descrizione degli agenti 4
3.2 Sistema 4
3.3 Agente di tipo Cliente 4
3.3.1 Workflow cliente 4
3.3.2 Architettura cliente 6
3.4 Agente di tipo Cassa 6
3.5 Ambiente 6
3.5.1 Interazione tra agenti 6
3.6 Parametri del modello 6
3.7 Workflow degli agenti 6

4 Implementazione dei comportamenti e delle strategie **7**
4.1 Fase di scelta della coda 7
4.2 Fase di attesa in coda e jockeying 9

5 Simulazione **10**

6 Analisi dei risultati **11**

7 Conclusioni **12**

Chapter 1

Introduzione

In questo progetto è stato costruito un modello basato su agenti con lo scopo di simulare il comportamento dei clienti all'interno di un supermercato durante la fase di scelta della coda relativa a diversi tipi di casse.

Un grande supermercato è composto molte casse, ognuna di queste ha un comportamento diverso. Come vederemo in dettaglio nel capitolo successivo, esistono diversi tipi di casse: la cassa standard, in cui si trova una cassiera che passa i prodotti provenienti dal nastro, la cassa self-service, in cui è il cliente stesso a scannerizzare i prodotti e la cassa self-scan, comparsa negli ultimi anni, in cui il cliente deve soltanto pagare, in quanto ha già effettuato la scannerizzazione dei prodotti man mano che li ha raccolti nel carrello. La configurazione delle casse nel supermercato può prevedere che ogni cassa abbia la sua coda dedicata, oppure che tante casse condividano la stessa coda.

Inserire il riferimento al capitolo

Dal momento che in un grande supermercato sono presenti più clienti che casse, è fondamentale l'utilizzo di code per gestire la grande quantità di clienti. Un supermercato può essere considerato a tutti gli effetti un sistema complesso in quanto si verificano diversi aspetti che considerati contemporaneamente fanno emergere un comportamento complessivo difficilmente prevedibile, tra questi:

- Flusso di clienti in ingresso variabile (che influisce sulle persone in coda)
- Numero di prodotti che un cliente acquista durante la spesa (che influisce sul tempo passato in cassa, scatenando eventualmente il cambio della coda da parte di altri clienti)
- Numero di casse aperte contemporaneamente nel supermercato
- Strategia di scelta della coda dei clienti (un cliente può avere diversi criteri per la scelta della coda)
- Strategia di cambio della coda dei clienti

Lo scopo di questo lavoro è costruire un modello basato su agenti per simulare il comportamento del supermercato. Dopo una prima fase di validazione, sfruttando i pochi dati a disposizione, vengono considerati diversi casi "what-if" con lo scopo di sperimentare diverse configurazioni di casse e strategie di scelta della coda per una gestione ottimale del flusso dei clienti.

Chapter 2

Stato dell'arte

In questo capitolo viene introdotto il problema, il dominio di riferimento e l'approccio adottato per la risoluzione.

Chapter 3

Descrizione del modello

3.1 Framework adottato e descrizione degli agenti

reference

Mesa è un framework in Python usato per la modellazione basata su agenti (ABM). Permette di creare il modello con componenti *built-in* come griglie spaziali e scheduler di agenti, e di visualizzare i componenti del modello con un'interfaccia browser. Sfruttando Mesa è possibile definire sia l'ambiente che gli agenti estendendo le relative classi messe a dispozione dal framework. Mesa infine comprende strumenti per l'analisi del modello creato.

Nel nostro modello di supermercato, il modello vero e proprio di Mesa è la classe **Supermarket** e gli agenti sono i clienti, classe **Customer**, e le casse, classe **CashDesk**.

3.2 Sistema

Supermercato non predicibile, presente una componente stocastica.

Il sistema è eterogeneo, due tipi di agenti. Sistema multiagente.

Come comunicano

Come avviene l'interazione

Descrizione dell'ambiente

3.3 Agente di tipo Cliente

I clienti sono gli agenti principali che vengono compongono il modello. Questi interagiscono con l'ambiente per raggiungere il goal "Fare la spesa", per portare a termine questo compito l'agente esegue acluni macro-step in cui è necessaria anche una fase di pianificazione e valutazione della bontà (utility function) della scelta adottata in alcune di queste.

3.3.1 Workflow cliente

I macro-step che l'agente deve seguire per il raggiungimento del goal sono:

1. **Attesa all'entrata del supermercato**: In questo stato il cliente si mette in attesa finchè non è possibile entrare nel supermercato.

Sperimenti sul covid

2. **Fase di shopping**: A questo punto il cliente è entrato e può iniziare a "fare la spesa", con questo si intende raggiungere il numero di prodotti desiderato. Infatti nella

simulazione viene considerato il "basket size", un'astrazione del carello modellato con un numero interno. Il basket size è il target da raggiungere nella fase di shopping, a ogni step il numero di prodotti aumenta di una certa quantità (la velocità di shopping è un parametro del modello section 3.6).

3. **Scelta della coda**: In questo stato il cliente ha finito di fare la spesa e vuole mettersi in coda ad una cassa. Dal momento che sono disponbili più code e ognuga di queste avrà tendinzialmente altri clienti già in coda, il cliente deve scegliere in quale coda andare in base alla coda che lui considera migliore.

   Formalmente viene definita una metrica calcolata per mezzo di una funzione: l'utility function. Il cliente sceglie quindi la coda che minimizza questa metrica.

$$q^* = \underset{q \in Q}{\operatorname{argmin}} f(q) \tag{3.1}$$

   dove Q è l'insieme delle code dedicate e f varia a seconda del tipo di strategia che il cliente può utilizzare, ognuga delle quali modella un diverso comportamento per la scelta della coda.

4. **Attesa in coda e jockeying**: Una volta che il cliente ha scelto la coda deve aspettare il proprio turno per essere servito. Nel lavoro di Tomasz Antczak e altri [**?**] viene suggerito come sviluppo futuro lo studio del fenomeno di jockeying. Con questa espressione si intende cambiare la coda che è stata scelta inizialmente in quanto un'altra risulta essere più conveniente, ad esempio perchè più scorrevole.

   Ad ogni timestamp l'agente considera le 2 (parametro spiegato in section 3.6 e scelto a priori per il modello) code adiacenti alla propria e per ognuga di queste ricalcola la coda migliore Equation 3.1. A questo punto l'agente valuta se per lui è conveniente cambiare la coda o rimanere in quella dove è già presente. Si noti come per effettuare questo confronto è necessario utilizzare l'approccio di Equation 3.1 sulla coda in cui l'agente è in quel momento andando ad escludere i clienti che si sono inseriti in coda dopo di lui. Una volta fatta questa valutazione, nel caso in cui sia per lui conveniente cambiare coda, il cliente la cambia in modo stocastico secondo una certa distribuzione di probabilità spiegata in chapter 4 che dipende da quanto è conveniente il cambio di coda.

   La scelta modellistica di effettuare jockeying stocasticamente e in funzione dell'effettivo guadagno di tempo è stata fatta per cercare di avere una rappresentazine il più fedele possibile al mondo reale. Infatti nella realtà un cliente al supermercato potrebbe scegliere di non cambiare la coda nel momento in cui ha un guadagno minimo in quanto questo richiede uno sforzo fisico. Inoltre nel lavoro di Tomasz Antczak e altri [**?**] viene fatto notare come questo comportamento non sia molto frequente, tuttavia seppur essendo poco frequente noi lo consideriamo un fattore importante da modellare, adottando le dovuto precauzioni sfruttando la stocasticità dell'azione.

5. **Attesa alla cassa**: In questa fase il cliente, essendo allo step precedente il primo della coda, è arrivato alla cassa. Da questo punto in poi verrà servito dall'agente di tipo cassa che avrà la responsabilità di processare il basket size del cliente. Dal momento che ci sono diversi tipi di casse, questa fase dipende dall'agente di tipo cassa e non dal cliente. Il cliente deve attendere la fine del processamento della spesa e quindi uscire dal negozio.

3.3.2 Architettura cliente

3.4 Agente di tipo Cassa

3.5 Ambiente

La classe **Supermarket** si occupa di inizializzare la griglia che verrà poi mostrata in fase di simulazione sull'interfaccia, inizializzare le casse, l'ambiente e i clienti.

Per inizializzare la griglia, l'ambiente viene diviso in zone: zona d'entrata, zona di shopping, zona casse normali, zona casse self-service, zona casse self-scan. Questa divisione permette una gestione più semplice dello spazio e dei movimenti degli agenti. Ogni zona ha come parametri la dimensione o il numero di casse che deve contenere, questi parametri vengono inizializzati nella classe **main**, come si vedrà nella prossima sezione.

Ogni zona è responsabile della propria costruzione, ovvero del proprio collocamento nella griglia dell'interfaccia in base alle proprie dimensioni ed eventualmente del posizionamento delle casse che contiene. Inoltre ogni zona è responsabile dei movimenti dei clienti: nel momento in cui un cliente vuole muoversi da una zona all'altra infatti, è la zona di destinazione che fornisce il metodo per posizionarsi correttamente in essa.

Ad ogni step della simulazione, il modello crea dei clienti e li posiziona nella **Entering Zone** del supermercato in coordinate random, dunque si occupa della attivazione e disattivazione delle casse standard in base al numero di clienti presenti nel negozio in quello step, secondo dei parametri che si vedranno nella sezione Workflow degli agenti. Quindi il modello chiama gli scheduler degli agenti, clienti e casse, e fa eseguire i loro step.

Parlare di quanto dura uno step e quanto dura una simulazione

citare la sezione

Caratteristiche dell'ambiente

- Accessibile vs. Inaccessibile - Deterministico vs. Non-deterministico - Episodicità vs. Non-episodicità - Statico vs. Dinamico - Discreto vs. Continuo

3.5.1 Interazione tra agenti

3.6 Parametri del modello

Casse adiacenti considerate

Inizializzazione della griglia e parametri per creare casse (numero casse, coda condivisa o no), customer, basket size (analisi dei dati di Gianluca).

3.7 Workflow degli agenti

Steps: ad ogni step entrano clienti in base alla distribuzione data come parametro; ad ogni step le casse decidono se aprire o chiudere in base al numero di clienti presenti; mettere grafici dei comportamenti e degli stati dei clienti e delle casse (quindi descrizione in dettaglio di ogni cassa e tipo di coda, di ogni stato della cassa e del cliente).

Chapter 4

Implementazione dei comportamenti e delle strategie

In questo capitolo verranno analizzate le strategie di scelta della coda da parte del cliente. In particolare queste strategie determinano la coda che il cliente deciderà di seguire durante la *fase di scelta della coda* e durante la *fase di attesa in coda e jockeying*.

Forse qui serve parlare dei paper (quello dei polacchi e quello delle fork)

Come introdotto all'inizio della relazione, obiettivo di questo progetto è analizzare le varie configurazioni di casse all'interno del supermercato, al variare del tipo di casse, della quantità di clienti presenti nel negozio e alla strategia di scelta della coda dei clienti, pertanto è necessario disporre di strategie che sfruttano calcoli basati su variabili diverse: le variabili in gioco saranno il numero di elementi nel carrello, il numero di persone e il tempo medio di attesa.

Se un cliente entra nel supermercato con l'intenzione di usare le casse self-scan, chiaramente non sarà dotato di strategie di scelta della coda e di jockeying, dal momento che le casse self-scan hanno sempre un'unica coda condivisa. Anche nel caso in cui le casse standard abbiano un'unica coda condivisa, come capita in molti negozi, il cliente non avrà una strategia di scelta della coda o di jockeying. Al contrario invece, dopo la fase di shopping il cliente dovrà scegliere la coda tra le code disponibili delle casse standard e delle casse self-service; una volta che esso si accoda in una cassa standard, può decidere di cambiare coda e quindi effettuare il jockeying, se ritiene, con la propria strategia, che nelle casse a lui più vicine ci sia un tempo minore di attesa.

Lo scopo del progetto è indagare sulla configurazione di casse migliori, quindi ad ogni simulazione tutti i clienti avranno la stessa strategia di scelta della coda e di jockeying per creare una netta distinzione tra simulazioni diverse.

4.1 Fase di scelta della coda

Alla fine della *fase di shopping*, il cliente che non vuole andare alla cassa self-scan, deve scegliere quale coda seguire tra quelle disponibili, ovvero tra le code associate a casse aperte.

Una strategia è una scelta della coda q che minimizza una certa funzione $f(q)$, come riportato nel capitolo precedente alla 3.1.

Al fine di definire le strategie, è importante notare che la funzione basket-size(c_i) non deve essere deterministica, come già introdotto nel capitolo precedente: il cliente fa una stima del numero di elementi nel carrello degli altri elementi, non ne è però certo, per questo questa funzione ha una probabilità di errore che rende la stima più veritiera, e si può ridefinire:

$$\text{estimate-basket-size}(c_i) = \text{basket-size}(c_i) + e_i \tag{4.1}$$

dove e_i è l'errore commesso nella stima e può essere sia positivo che negativo. L'errore dipende dalla grandezza del carrello, in particolare più elementi ci sono nel carrello più l'errore aumenta; intuitivamente in questo modello si è deciso di far variare l'errore in modo logaritmico rispetto alla grandezza del carrello, in base alla formula:

$$e_i = log_b(\text{basket-size}(c_i)) \tag{4.2}$$

dove $b \in \mathbb{R}$, $b > 1$. La quantità estimate-basket-size diventa dunque una variabile aleatoria normale con media la basket-size reale e deviazione standard l'errore commesso sulla stima.

Le 4 strategie di scelta della coda prese in considerazione per il modello sono:

1. **Minimo numero di elementi**: viene scelta la coda con il minimo numero di elementi nei carrelli di tutte le persone in attesa. La funzione da minimizzare è quindi per una coda $q \in Q$ dove Q è l'insieme di tutte le code disponibili:

$$f(q) = \sum_{i=1}^{N} \text{estimate-basket-size-size}(c_i) \tag{4.3}$$

   dove $c_i \in q$ è un cliente in coda, estimate-basket-size-size(c_i) è il numero di elementi che ha nel suo carrello e $N = |q|$ è il numero di clienti in coda in q.

2. **Minimo numero di persone**: viene scelta la coda con il minimo numero di persone accodate. La funzione da minimizzare per $q \in Q$ è:

$$f(q) = |q| \tag{4.4}$$

3. **Minimo tempo d'attesa in base al tempo di servizio medio**: viene scelta la coda con il tempo d'attesa minimo, calcolato in base al tempo di servizio medio. Il *tempo di servizio totale di una coda* è calcolato come somma del tempo di servizio per ogni cliente della coda. Il tempo di servizio per un cliente comprende il **tempo di transazione** e il **tempo di pausa** tra un cliente e un altro. I dati sui tempi di transazione e di pausa sono stati estrapolati dall'articolo, in particolare i coefficienti $a, b, c, d \in \mathbb{R}$, e sono, per un cliente $c_i \in q$:

reference al paper

$$\text{transaction-time}_i = e^{a log(\text{basket-estimate-basket-size}(c_i))+b} \tag{4.5}$$

$$\text{break-time}_i = e^{c log(\text{basket-estimate-basket-size}(c_i))+d} \tag{4.6}$$

   I tempi di servizio per ogni cliente di una coda sono quindi sommati per calcolare il tempo servizio totale per quella coda. Il tempo di servizio totale di una coda q_j, $j = 1, \ldots, M$, dove M è il numero totale di code del supermercato, è pertanto:

$$\text{total-service-time}(q_j) = \sum_{i=1}^{N} (\text{transaction-time}_i + \text{break-time}_i) \tag{4.7}$$

   Il tempo di servizio medio per le code è la somma dei tempi totali di servizio divisa per il numero di code. Viene scelta la coda con il tempo totale minimo, mettendo insieme la 4.5, la 4.6 e la 4.7 si ottiene la funzione da minimizzare:

$$f(q) = |q| * \frac{1}{M} \sum_{j=1}^{M} (\text{total-service-time}(q_j)) \tag{4.8}$$

4. **Minimo tempo d'attesa in base alla *power regression***: viene scelta la coda con il tempo d'attesa minimo, calcolato in base al tempo di transazione e il tempo di pausa medi per quella coda. Il tempo di servizio totale per una coda è calcolato anche qui in base alle 4.5, 4.6 e 4.7. La funzione da minimizzare è:

$$f(q) = |q| * \text{total-service-time}(q) \tag{4.9}$$

4.2 Fase di attesa in coda e jockeying

Quando il cliente è nella *fase di attesa in coda* ha la possibilità di cambiare la propria scelta se nota che nelle code adiacenti il tempo di attesa è minore che nella propria; le code adiacenti implicate nel calcolo sono quelle che hanno distanza minore o uguale del **parametro di adiacenza**, descritto nel capitolo precedente, rispetto a dove il cliente è già accodato. Una coda è distante 1 da un'altra coda se fisicamente nella griglia della simulazione esse compaiono una di fianco all'altra.

Le strategie seguite nel jockeying sono due, e coincidono con le prime due strategie di scelta della coda, descritte nella sezione precedente. Anche in questo caso il cliente stima le grandezze dei carrelli degli altri clienti in base alle 4.1 e 4.2.

1. **Minimo numero di elementi**: viene scelta la coda con il minimo numero di elementi nei carrelli di tutte le persone in attesa.
2. **Minimo numero di persone**: viene scelta la coda con il minimo numero di persone accodate.

Se il cliente, grazie alla sua strategia, calcola che nelle code adiacenti ci sia un tempo di attesa minore rispetto alla propria, cambia coda e si mette in fondo alla nuova coda scelta.

Chapter 5

Simulazione

Chapter 6

Analisi dei risultati

Mettere grafico fondamentale, ma prima definire densità e flusso per come li abbiamo calcolati nella classe Supermarket.

Chapter 7

Conclusioni

Ciao