



UNIVERSITÀ DEGLI STUDI DI SALERNO

PROGETTO DI FONDAMENTI DI INTELLIGENZA ARTIFICIALE

MAÎTRE 2.0: AGENTE DECISIONALE IN UN RISTORANTE

29 DICEMBRE 2025

*Autore*

*Matricola*

---

Dario Juri Pepe

0512120228

<https://github.com/dariojuri/Maitre-2.0>

# INDICE

<b>1</b>	<b>Contesto e sistema proposto .....</b>	<b>3</b>
1.1	Introduzione generale.....	3
1.2	Sistema attuale.....	3
1.3	Maître 2.0 .....	3
<b>2</b>	<b>Definizione del problema.....</b>	<b>5</b>
2.1	Obiettivi.....	5
2.2	Specifica PEAS .....	5
2.3	Caratteristiche dell'ambiente simulato.....	5
2.4	Analisi del problema .....	6
<b>3</b>	<b>Soluzione proposta .....</b>	<b>7</b>
3.1	Modellazione della simulazione event-driven .....	7
3.2	Strategia baseline: Round Robin.....	7
3.3	Strategia con euristica: Best-First .....	8
3.4	Metriche e metodologia di valutazione.....	8
3.4.1	Configurazione degli scenari.....	8
3.4.2	Metriche di valutazione .....	9
<b>4</b>	<b>Risultati sperimentali.....</b>	<b>10</b>
4.1	Scenario A – Carico normale.....	10
4.2	Scenario B – Picco di richieste.....	10
4.3	Scenario C – Camerieri con efficienze diverse.....	11
4.4	Discussione dei trade-off.....	11
<b>5</b>	<b>Conclusioni.....</b>	<b>13</b>

# **1 CONTESTO E SISTEMA PROPOSTO**

## **1.1 Introduzione generale**

In un servizio di ristorazione il maître di sala deve prendere diverse decisioni in pochi secondi: quale tavolo servire per primo, quando portare i piatti e come organizzare il lavoro tra i camerieri. Picchi di richieste, ritardi e imprevisti rendono difficile mantenere tempi di attesa regolari per tutti i clienti.

Maître 2.0 è una simulazione che riproduce questo contesto e introduce un agente che decide come assegnare i task ai camerieri utilizzando strategie diverse. Attraverso l'analisi dei risultati, il progetto consente di osservare come varino le attese dei clienti e il carico di lavoro del personale.

## **1.2 Sistema attuale**

Nella gestione attuale del servizio, le decisioni vengono prese in modo prevalentemente reattivo e non sistematico. Il personale interviene sulle situazioni più evidenti man mano che si presentano. Le priorità dipendono principalmente dell'esperienza dei singoli e dalla percezione del momento, piuttosto che da una valutazione complessiva dello stato della sala.

Questo approccio può funzionare in condizioni normali, ma diventa insostenibile nei momenti di picco: alcuni tavoli accumulano tempi di attesa elevati mentre altri vengono serviti più rapidamente, e allo stesso tempo alcuni camerieri risultano sovraccarichi mentre altri rimangono sottoutilizzati. La mancanza di uno schema decisionale esplicito rende difficile prevedere e analizzare il comportamento del sistema, e quindi migliorarne le prestazioni.

## **1.3 Maître 2.0**

Per superare i limiti del sistema attuale, Maître 2.0 introduce un agente decisionale che opera all'interno di una simulazione del ristorante. L'agente osserva lo stato dell'ambiente (tavoli, richieste e disponibilità camerieri) e decide in tempo reale come assegnare i task.

Il sistema permette di confrontare strategie decisionali diverse, dalla baseline più semplice a soluzioni più informate, valutandone gli effetti sui tempi di attesa e sulla distribuzione del lavoro tra i camerieri. In questo modo è possibile studiare il

comportamento del servizio in condizioni controllate, senza intervenire direttamente su un contesto reale.

## 2 DEFINIZIONE DEL PROBLEMA

### 2.1 Obiettivi

Lo scopo del progetto è studiare come diverse strategie decisionali influenzino il comportamento del servizio in un ristorante simulato. In particolare, il progetto si propone di:

- Modellare il ristorante come ambiente dinamico basato su eventi;
- Implementare un agente che assegna i task ai camerieri;
- Confrontare una strategia baseline con una strategia euristica;
- Valutare gli effetti delle strategie sui tempi di attesa dei clienti e sul bilanciamento del carico di lavoro;
- Fornire uno strumento di configurazione che permetta di modificare i parametri principali della simulazione (numero di tavoli, numero ed efficienza dei camerieri, frequenza degli eventi), così da analizzare scenari differenti.

### 2.2 Specifica PEAS

Di seguito è riportata la descrizione PEAS dell'ambiente operativo.

Performance	Riduzione dei tempi di attesa e bilanciamento del carico di lavoro tra i camerieri
Environment	Sala con più tavoli, eventi stocastici e risorse limitate
Actuators	Assegnazione dei task ai camerieri
Sensors	Informazioni sullo stato dei tavoli, sulle richieste pendenti, sui piatti pronti e sulla disponibilità dei camerieri

I sensori sono attivati dagli eventi della simulazione che notificano all'agente i cambiamenti nello stato dell'ambiente.

### 2.3 Caratteristiche dell'ambiente simulato

L'ambiente simulato rappresenta il servizio di sala di un ristorante durante un turno di lavoro. Nel corso della simulazione si verificano diversi tipi di eventi, tra cui l'arrivo di nuovi tavoli, la generazione degli ordini, la preparazione dei piatti e le richieste di

pagamento. Tali eventi compaiono in modo stocastico, introducendo variabilità e incertezza, come nel mondo reale.

Ogni tavolo può trovarsi in stati diversi (attesa, ordinazione, servizio, conto), mentre i camerieri sono modellati come risorse con disponibilità e livello di efficienza. L'agente può osservare lo stato corrente dei tavoli, dei task e del personale, ma non conosce in anticipo la sequenza esatta degli eventi futuri.

Di conseguenza, l'ambiente è parzialmente osservabile, stocastico, sequenziale, dinamico e discreto, ed è modellato come ambiente a singolo agente.

## **2.4 Analisi del problema**

La gestione del servizio di un ristorante comporta la necessità di prendere decisioni in condizioni di incertezza, con risorse limitate e nel minor tempo possibile. L'agente deve stabilire quale compito eseguire per primo e a quale cameriere assegnarlo, cercando di ridurre i tempi di attesa e di evitare che il carico di lavoro si concentri su pochi elementi del personale.

La difficoltà principale deriva dal fatto che gli eventi non sono completamente prevedibili, nuovi tavoli possono arrivare in qualsiasi momento, la cucina può accumulare ritardi e alcune richieste possono assumere priorità maggiore rispetto ad altre. Ogni decisione presa in un certo momento influenza quelle successive, modificando lo stato dei tavoli, della sala e la disponibilità dei camerieri.

Ne consegue che il problema non può essere affrontato soltanto con regole statiche, è quindi necessario adottare strategie decisionali capaci di reagire ai cambiamenti dell'ambiente e di gestire in modo equilibrato il compromesso tra rapidità del servizio e distribuzione del lavoro.

## 3 SOLUZIONE PROPOSTA

### 3.1 Modellazione della simulazione event-driven

La simulazione è basata su un modello event-driven, in cui il sistema è guidato da una sequenza di eventi ordinati nel tempo. Ogni evento rappresenta un cambiamento significativo nello stato dell'ambiente.

Gli eventi sono gestiti tramite una coda temporale, che consente di estrarre di volta in volta quello più prossimo e di aggiornare lo stato della sala. L'elaborazione di un evento può generare nuovi eventi futuri e modificare lo stato dei tavoli, dei task e dei camerieri.

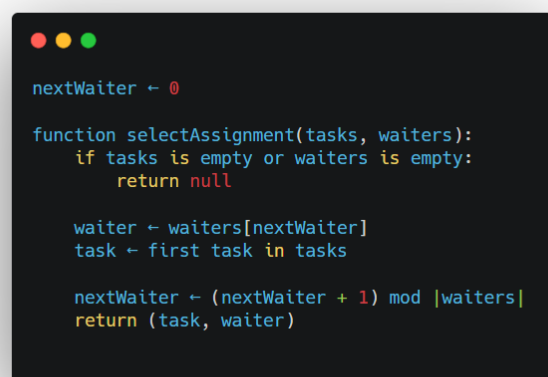
Quando, a seguito di un evento, risultano sia task da eseguire sia camerieri liberi, viene invocato l'agente decisionale. L'agente osserva lo stato corrente, valuta in base alla strategia in uso e produce un'assegnazione dei task, che viene applicata all'ambiente e genera a sua volta nuovi eventi.

Il progetto costruisce una simulazione ad hoc che genera dati sintetici. In particolare, il flusso di arrivo dei tavoli è modellato tramite una funzione sigmoidea che fa sì che, all'inizio del servizio, gli arrivi siano pochi, aumentino gradualmente fino a un picco centrale e diminuiscano verso la chiusura. Questo consente di riprodurre andamenti realistici con fasi di calma e di congestione.

### 3.2 Strategia baseline: Round Robin

La strategia di riferimento utilizzata è Round Robin, scelta come baseline in quanto non utilizza alcuna informazione sull'urgenza dei task né sull'efficienza dei camerieri. Questa strategia assegna i compiti in modo ciclico, garantendo una distribuzione semplice e prevedibile del lavoro, ma non informata dallo stato del sistema.

In particolare, l'algoritmo mantiene un indice che identifica il prossimo cameriere a cui assegnare un task. Ogni volta che



```
nextWaiter ← 0

function selectAssignment(tasks, waiters):
  if tasks is empty or waiters is empty:
    return null

  waiter ← waiters[nextWaiter]
  task ← first task in tasks

  nextWaiter ← (nextWaiter + 1) mod |waiters|
  return (task, waiter)
```

un compito deve essere assegnato, questo viene affidato al cameriere indicato e l'indice viene aggiornato al successivo, realizzando una rotazione continua tra il personale.

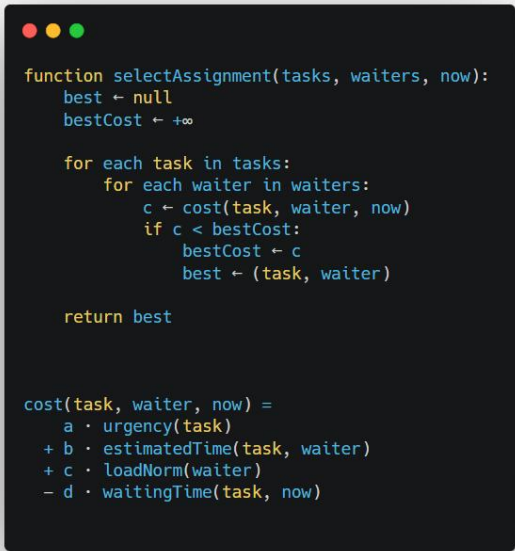
In questo modo, Round Robin fornisce un comportamento semplice e riproducibile, utile come termine di paragone per valutare il miglioramento ottenuto dalle strategie euristiche.

### 3.3 Strategia con euristica: Best-First

La strategia euristica adottata utilizza un approccio di tipo best-first, in cui le possibili assegnazioni vengono valutate in base a una funzione di costo. A differenza della baseline, la decisione non riguarda solo quale task eseguire, ma anche a quale cameriere assegnarlo.

L'algoritmo considera coppie (task, cameriere) e assegna una priorità a ciascuna combinazione in base a più fattori: l'urgenza del task, l'efficienza del cameriere, il tempo di attesa già accumulato e il carico di lavoro sostenuto. L'efficienza viene utilizzata per stimare il tempo di completamento, mentre l'urgenza rappresenta l'importanza relativa dei diversi tipi di richieste.

La coppia con il costo complessivo minore viene selezionata ed eseguita, consentendo all'agente di prendere decisioni più informate rispetto al semplice Round Robin e di adattarsi meglio alle condizioni dell'ambiente.



```
function selectAssignment(tasks, waiters, now):
    best ← null
    bestCost ← +∞

    for each task in tasks:
        for each waiter in waiters:
            c ← cost(task, waiter, now)
            if c < bestCost:
                bestCost ← c
                best ← (task, waiter)

    return best

cost(task, waiter, now) =
    a · urgency(task)
    + b · estimatedTime(task, waiter)
    + c · loadNorm(waiter)
    - d · waitingTime(task, now)
```

### 3.4 Metriche e metodologia di valutazione

#### 3.4.1 Configurazione degli scenari

Per valutare il comportamento delle strategie in condizioni differenti, sono stati definiti tre scenari di test che cariano il carico del sistema e le caratteristiche del personale. Ogni scenario è stato eseguito più volte utilizzando seed casuali differenti, al fine di ridurre l'influenza della componente stocastica.

La tabella seguente riassume i principali parametri utilizzati per ciascuno scenario.

Scenario	Camerieri (eff.)	MinIA	MaxIA	MaxTables	Cucina
A	1.0, 1.0, 1.0, 1.0, 1.0	2.0	7.0	60	5, 1, 3
B	1.0, 1.0, 1.0, 1.0	0.8	3.0	90	5, 1, 3
C	1.4, 1.2, 1.0, 0.9, 0.8	1.5	6.0	70	5, 1, 3

Gli scenari sono stati scelti per rappresentare situazioni operative realistiche: uno scenario di funzionamento normale, uno scenario di forte carico e uno scenario con personale eterogeneo in termini di efficienza.

### 3.4.2 Metriche di valutazione

Per confrontare le strategie decisionali sono state utilizzate due metriche principali.

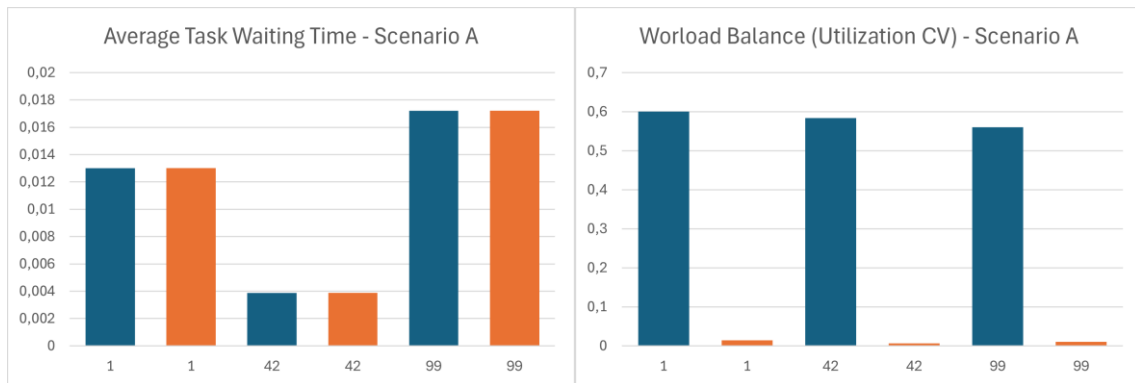
La prima è il tempo medio di attesa dei task (*Average Task Waiting Time*), che misura il tempo medio trascorso tra la generazione di una richiesta e il momento in cui essa viene assegnata a un cameriere. Questa metrica rappresenta un indicatore diretto della qualità del servizio percepita dai clienti: valori più bassi corrispondono a un servizio più rapido.

La seconda metrica utilizzata è *UtilizationCV*, che misura il grado di bilanciamento del carico di lavoro tra i camerieri. Per ogni cameriere viene calcolato il livello di utilizzo come rapporto tra il tempo totale di lavoro svolto e il tempo massimo disponibile, normalizzato rispetto la sua efficienza. Il coefficiente di variazione di questi valori indica quanto l'utilizzo dei camerieri differisce tra loro: valori elevati indicano forte sbilanciamento, mentre valori prossimi a zero indicano una distribuzione equa del lavoro.

Per ogni esecuzione, le metriche vengono esportate automaticamente in formato CVS, permettendo l'analisi dei risultati e la costruzione dei grafici utilizzati nel capitolo dei risultati sperimentali.

## 4 RISULTATI SPERIMENTALI

### 4.1 Scenario A – Carico normale

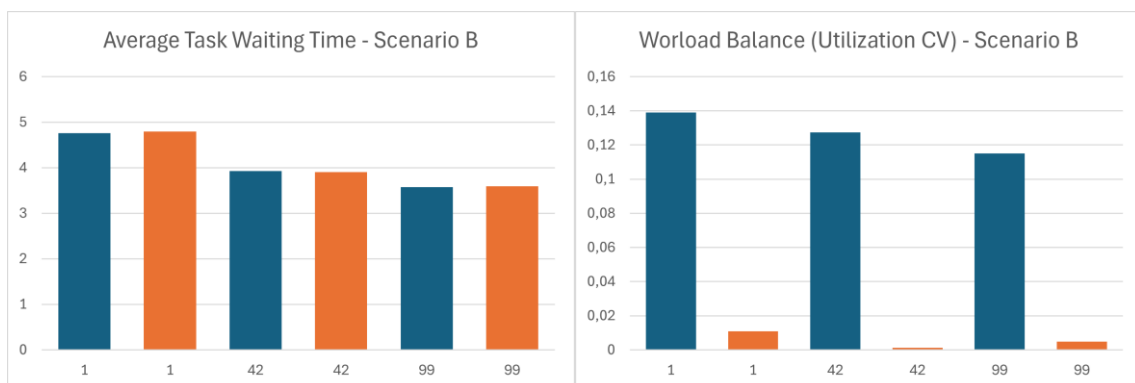


Nel primo scenario, che rappresenta una situazione di carico normale con camerieri di efficienza identica, si osserva che le strategie Round Robin e Best-First producono tempi di attesa medi praticamente equivalenti.

Tuttavia, il valore di UtilizationCV risulta significativamente più elevato per il Round Robin, indicando una distribuzione del lavoro fortemente sbilanciata. Al contrario, la strategia Best-First mantiene un UtilizationCV prossimo allo zero, segno che i camerieri vengono utilizzati in modo proporzionale alle loro capacità.

Questo mostra che, in condizioni stabili, l'euristica riesce a migliorare il bilanciamento delle risorse senza penalizzare la qualità del servizio.

### 4.2 Scenario B – Picco di richieste



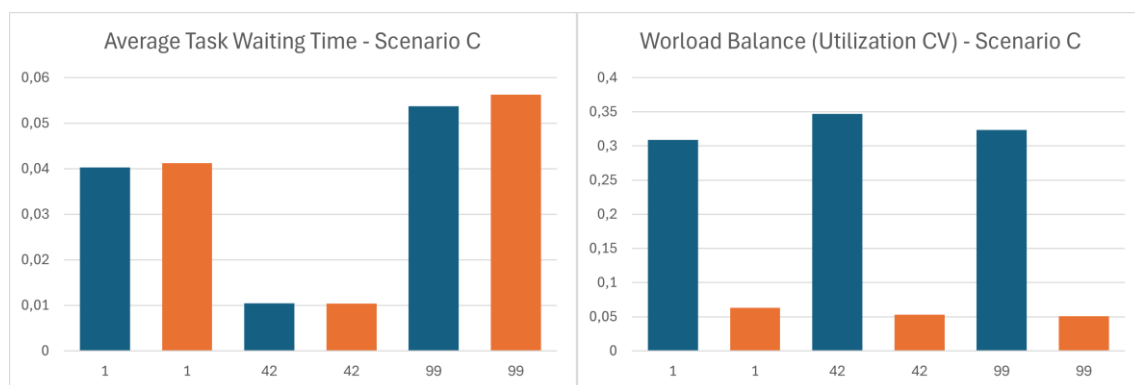
Nel secondo scenario, caratterizzato da un forte carico operativo, la differenza tra le due strategie rimane simile a quella osservata nello Scenario A.

Entrambe producono tempi di attesa elevati e comparabili, a indicare che il sistema è vicino alla saturazione e che il collo di bottiglia non è più l'assegnazione dei camerieri, ma il flusso stesso degli eventi.

In questo contesto, sebbene la strategia Best-First presenti ancora un Utilization CV inferiore, il Round Robin mostra un miglior bilanciamento rispetto allo scenario A, poiché l'elevato numero di richieste costringe tutti i camerieri a lavorare quasi continuamente.

Questo evidenzia che, in condizioni di saturazione, il margine di miglioramento dell'euristica si riduce.

### 4.3 Scenario C – Camerieri con efficienze diverse



Nel terzo scenario, in presenza di camerieri con capacità operative differenti, la strategia euristica il suo principale vantaggio.

Il Best-First assegna più lavoro ai camerieri più efficienti e limita quelli più lenti, ottenendo un utilizzo delle risorse molto più equilibrato, pur a fronte di una lieve perdita in termini di tempo medio di attesa.

In questo caso l'euristica sfrutta in modo più efficace l'eterogeneità del personale, evitando che risorse lente colli di bottiglia e migliorando la qualità complessiva del servizio.

### 4.4 Discussione dei trade-off

Dai risultati emerge un chiaro quadro delle due strategie.

Il Round Robin tende a essere leggermente più rapido in termini di tempo medio di attesa, grazie alla sua semplicità e al basso overhead decisionale, ma a costo di un utilizzo fortemente sbilanciato del personale.

La strategia Best-First, invece, introduce un costo computazionale maggiore per valutare le possibili assegnazioni, ma in cambio ottiene una distribuzione del lavoro molto più equa e una maggiore robustezza in presenza di personale eterogeneo.

## 5 CONCLUSIONI

Maître 2.0 ha dimostrato come un problema reale e complesso, quale la gestione del servizio di sala in un ristorante, possa essere modellato come un ambiente di decisione per un agente intelligente. Attraverso una simulazione event-driven e l'utilizzo di strategie di assegnazione differenti, è stato possibile analizzare in modo quantitativo l'impatto delle scelte decisionali sui tempi di attesa dei clienti e sull'utilizzo delle risorse.

Il confronto tra una strategia semplice come Round Robin e una strategia euristica di tipo Best-First ha messo in evidenza un compromesso fondamentale: la semplicità garantisce rapidità, mentre una maggiore informazione consente un uso più efficiente ed equo delle risorse.