Cucina intelligente di Massimo

Intelligent Agent Systems

Ausarbeitung

für das **Center for Advanced Studies der Dualen Hochschule Baden-Württemberg**

von

Jonathan Diebel, Jannis Kaniaros und Dario Nieddu

31. März 2025

Matrikelnummern	9883850, 5934448,
Kurs	T3M40501
Dozent	Prof. Dr. Nathan Sudermann-Merx

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Restaurants sind Orte, an denen Menschen nicht nur gutes Essen genießen, sondern auch eine Auszeit vom Alltag nehmen können. Ob für ein romantisches Dinner, ein schnelles Mittagessen oder ein entspanntes Treffen mit Freunden – die Gastronomie bietet für jeden Anlass das passende Ambiente. Doch in einer Welt, die immer schneller wird und in der Zeit oft zur kostbarsten Ressource wird, stehen auch Restaurants vor der Herausforderung, sich an die Bedürfnisse ihrer Gäste anzupassen.

Ein neues Konzept revolutioniert nun das Restauranterlebnis: Kunden können bereits beim Betreten des Restaurants oder bei der Reservierung angeben, wie viel Zeit sie für ihren Besuch einplanen. Ob 30 Minuten für eine schnelle Mahlzeit oder zwei Stunden für ein ausgedehntes Dinner – dieses innovative Modell ermöglicht es, den Service individuell auf die Wünsche der Gäste abzustimmen. So wird nicht nur die Effizienz gesteigert, sondern auch das Restauranterlebnis persönlicher und flexibler gestaltet.

In dieser Arbeit wird das Konzept der zeitbasierten Reservierung genauer untersucht und ein optimiertes intelligentes Mutliagentensystem vorgestellt, das die Umsetzung in einem Restaurant, der *Cucina intelligente di Massimo*, unterstützt.

Abbildung 1.1: Cover

2 Design

2.1 Agenten

Im Modell werden drei verschiedene Agenten eingesetzt: *Customer Agents*, *Service Agents* sowie ein *Manager Agent*. Die Agenten sind in einem Multi-Agenten-System organisiert, in dem sie miteinander kommunizieren und sich gegenseitig beeinflussen.

2.1.1 Customer Agents

Customer Agents beschreiben eine Gruppe von Kunden, die ein Restaurant besuchen und eine Bestellung aufgeben wollen. Die Attribute der Agenten werden größtenteils randomisiert generiert. Dazu zählen die genaue Anzahl der Personen, die maximale Wartezeit sowie das ausgewählte Gericht.

Die Agenten werden über einen internen Status gesteuert, der den aktuellen Zustand des Agenten beschreibt. Die möglichen Zustände sind *Waiting for Service Agent*, *Waiting for food*, *Eating*, *Finished Eating*, *Rejected* und *Done*. Der Status wird durch die verschiedenen Aktionen des Agenten verändert.

Wichtig im Zuge dieses Konzepts ist, dass die Agenten das Restaurant nicht verlassen, sobald ihre Zeit abgelaufen ist. Stattdessen wird das Essen beendet, dafür sinkt jedoch die Bewertung.

Wenn die Customer Agents ihr Essen beendet haben, geben sie eine Bewertung ab. Diese Bewertung ist eine Funktion mit mehreren Zufallsfaktoren:

$$r = \text{round}\left(\max\left(r_{\min}, \min\left(r_{\max}, r_{\max} - \alpha \cdot \text{exceedance} - \beta \cdot \text{error} + \gamma\right)\right), 2\right)$$

wobei:

- r die finale Bewertung ist,
- $r_{\min}$ die minimal mögliche Bewertung ist,
- $r_{\max}$ die maximal mögliche Bewertung ist,
- α die Gewichtung der Wartezeitstrafe ist,

- *exceedance* der Quotient aus der tatsächlichen Wartezeit und der angegebenen maximalen Zeit ist (nur bei Überschreitung, sonst 0),
- β die Gewichtung für fehlerhafte Bestellungen ist,
- *error* eine zufällige Fehlerquote ist,
- γ eine zufällige Bewertungsvariabilität abhängig von der Gruppengröße ist.

Wird ein Kunde abgewiesen, gibt er immer die niedrigste Bewertung ab.

Im *PEAS-Framework* lassen sich die Customer Agents wie folgt beschreiben:

Performance Measure Einhaltung der maximalen Zeit

Environment Das Restaurant, Service Agents

Actuators Bestellung aufgeben, Essen bewerten

Sensors Wartezeit, interner Status

Bei den Customer Agents handelt es sich gemäß des *AIMA-Framework*s um *Simple Reflex Agents*, da sie ihre Aktionen basierend auf dem aktuellen Zustand und den wahrgenommenen Informationen ausführen, ohne eine interne Zustandsrepräsentation der Welt zu verwenden. Sie reagieren direkt auf die wahrgenommenen Reize, wie z.B. die erhaltene Bestellung, um ihre nächsten Aktionen zu bestimmen.

2.1.2 Service Agents

Service Agents sind für die Bedienung der Customer Agents zuständig. Sie haben die Aufgabe, die Bestellungen der Kunden entgegenzunehmen, diese zuzubereiten und das Essen zu servieren. Jeder Service Agent hat eine Kapazität an Customer Agents, die er bedienen kann. Einmal zugewiesen, werden die Customer Agents nicht mehr von anderen Service Agents bedient, außer der Service Agent beendet seine Arbeit, dann werden seine Customer Agents an die übrigen Service Agents übergeben.

Jeder Service Agent kann pro Zeitschritt nur zwei Customer Agents bedienen, dabei wird ein Customer Agent entweder platziert oder abgewiesen und in die Status *Waiting for food* bzw *Rejected* versetzt, während für einen anderen Customer Agent die Zubereitung der Speisen durchgeführt wird. Da die Zubereitung der Speisen im Menü festgelegt ist und unterschiedlich lange dauert, wird deshalb entweder die Zubereitungszeit um einen

Zeitschritt reduziert oder, sofern die Zubereitung abgeschlossen ist, das Essen serviert und der Customer Agent in den Status *Eating* versetzt.
Kunden werden dann abgewiesen, wenn sie im Status *Waiting for Service Agent* sind, aber die Summe aus Zubereitungs- und Essenszeit der Speisen die maximale Wartezeit überschreitet, also ein Service in der angegeben Zeit überhaupt nicht möglich ist.

Die zu bedienenden Customer Agents werden von den Service Agents in jedem Schritt dynamisch sortiert und entsprechend bedient. Dabei werden verschiedene Faktoren berücksichtigt und gewichtet, um die Interessen der Kunden (also die Minimierung der Wartezeit) sowie die Interessen des Managers (also die Maximierung des Gewinns) zu berücksichtigen.
Dabei wird für jeden Kunden ein Rang bestimmt - je höher der Rang, desto früher wird der Kunde bedient. Die Sortierung für neue Kunden erfolgt wie folgt:

$$r = \alpha \cdot p \cdot n + \beta \cdot \text{total\_time} + \gamma \cdot \text{time\_left}$$

wobei:

- r der Rang des Kunden ist,
- p der Preis des Gerichts ist,
- n die Anzahl der Personen ist,
- α die Gewichtung des Profits ist,
- $total\_time$ die Gesamtzeit ist, die der Kunde bereits im Restaurant verbracht hat,
- β die Gewichtung der Gesamtzeit ist,
- $time\_left$ die verbleibende Zeit ist,
- γ die Gewichtung der verbleibenden Zeit ist

Für die Sortierung der bereits im Restaurant befindlichen Kunden wird die gleiche Formel verwendet, zusätzlich jedoch noch die Zubereitungszeit der Speisen berücksichtigt wird. Damit ergibt sich folgende Formel:

$$r = \alpha \cdot p \cdot n + \beta \cdot \text{total\_time} + \gamma \cdot \text{time\_left} + \delta \cdot \text{cooking\_time}$$

wobei die Werte entsprechend der Formel für neue Kunden definiert sind, *cooking_time* die Zubereitungszeit des Gerichts und δ die Gewichtung der Zubereitungszeit ist.

Im *PEAS-Framework* lassen sich die Service Agents wie folgt beschreiben:

Performance Measure Wartezeit der Kunden minimieren, möglichst hohe Bewertung, Maximierung des Gewinns

Environment Das Restaurant, Customer Agents

Actuators Bestellung aufnehmen, Essen zubereiten, Essen servieren, Kunden abweisen

Sensors Neue Kunden vorhanden, Zustand der Customer Agents

Bei den Service Agents handelt es sich gemäß des *AIMA-Framework*s ebenfalls um *Simple Reflex Agents*, da sie ihre Aktionen basierend auf dem aktuellen Zustand und den wahrgenommenen Informationen ausführen, ohne eine interne Zustandsrepräsentation der Welt zu verwenden. Sie reagieren direkt auf die wahrgenommenen Reize, wie z.B. den Status der Kunden.

2.1.3 Manager Agent

Der Manager Agent ist für die Koordination der Service Agents zuständig. Er existiert im Restaurant nur ein Mal und hat die Aufgabe, die Service Agents zu verwalten.

3 Mathematische Formulierung des Optimierungsproblems

Wir haben ein engagiertes Team von Mitarbeitern und eine Reihe von Einschränkungen, die sich vor allem aus dem Arbeitsschutzgesetz ergeben. Einige Mitarbeiter arbeiten weniger Stunden und Schichten, während andere mehr arbeiten. Das muss nicht ausgewogen oder fair sein, sondern muss bestmöglich zur Zielfunktion beitragen. Die Herausforderung besteht darin, die Mitarbeiter für den nächsten Tag so einzusetzen, dass der Gesamtgewinn maximiert wird. Die Mitarbeiter werden im Laufe des Tages weder versetzt noch ausgewechselt. Darüber hinaus hat jeder Mitarbeiter einen spezifischen `customer_capacity`-Faktor, der seine Effizienz oder sein Qualifikationsniveau darstellt – je höher dieser Faktor, desto höher das Gehalt des Mitarbeiters.

Nachfolgend wird das Optimierungsproblems der Schichtplanung mathematisch mit allen erforderlichen Entscheidungsvariablen, Parametern, Zielfunktionen und Nebenbedingungen definiert.

Entscheidungsvariablen

Sei:

- $x_{a,t} \in \{0, 1\}$
  Eine binäre Variable, die angibt, ob der Service-Agent a im Zeitfenster $t = \{0, \ldots, 23\}$ arbeitet (1 = arbeitet, 0 = arbeitet nicht).
- $y_{a,s} \in \{0, 1\}$
  Eine binäre Variable, die angibt, ob der Service-Agent a der Schicht s zugewiesen wurde (1 = zugewiesen, 0 = nicht zugewiesen).

Parameter

Sei:

- c_a
  Gehalt pro Zeitschritt für den Service-Agenten a.

- p_a

  Kapazität des Service-Agenten a. Gibt an, wie viele Kunden er gleichzeitig in einem Zeitschritt bedienen kann.

- v_t

  Prognostizierte Anzahl an Besuchern im Zeitfenster t.

- $S_a = 3$

  Maximale Anzahl an Schichten, die ein Service-Agent a pro Tag arbeiten darf.

- $T = \{1, 2, ..., 144\}$

  Menge aller Zeitfenster eines Tages, wobei jedes Zeitfenster eine Dauer von 10 Minuten hat (144 Zeitfenster für einen Zeitraum von 24 Stunden).

- $S = \{[0, 1, 2, 3, 4, 5], [6, 7, 8, 9, 10, 11], [12, 13, 14, 15, 16, 17], [18, 19, 20, 21, 22, 23]\}$

  Stunden der Schichten, wobei jede Schicht 6 Stunden dauert

Zielfunktion

Das Ziel ist es, den Gewinn des Restaurants zu maximieren, indem die Kundennachfrage erfüllt und gleichzeitig die Personalkosten minimiert werden. Die Zielfunktion lautet:

$$\text{Minimiere } f(x) = \sum_{t \in T} \sum_{a \in A} c_a x_{a,t}$$

Dabei:

- Der erste Term repräsentiert den Gesamterlös basierend auf der Anzahl der bedienten Kunden. Dieser wird durch die prognostizierte Besuchernachfrage (v_t) und die Gesamtkapazität der arbeitenden Agenten ($p_a x_{a,t}$) begrenzt.
- Der zweite Term repräsentiert die gesamten Gehaltskosten für alle arbeitenden Agenten.

Nebenbedingungen

1. **Erfüllung der Nachfrage**

   Die Gesamtkapazität der zugewiesenen Agenten muss mindestens so groß sein wie die Besuchernachfrage in jedem Zeitfenster:

$$\sum_{a \in A} p_a x_{a,t} \geq v_t, \quad \forall t \in T$$

2. **Maximale Arbeitszeit pro Agent**

   Jeder Agent darf maximal $M_a = 48$ Zeitschritte pro Tag arbeiten:

$$\sum_{t \in T} x_{a,t} \leq M_a, \quad \forall a \in A$$

3. **Mapping der Service Agents auf Schichten**

   Wenn ein Agent in einer Schicht arbeitet, muss die binäre Variable x für den Zeitschritt den selben Wert haben wie die Variable y für die Schicht. Daraus ergibt sich folgende Implikation: $y_{a,s} = 1 \Rightarrow x_{a,t} = 1, \quad \forall t \in T, s \in S$.

   Gleichzeitig muss die umgekehrte Implikation gelten, denn für jeden Zeitschritt, an dem der Service Agent arbeitet, ist er der zugehörigen Schicht zugewiesen: $x_{a,t} = 1 \Rightarrow y_{a,s} = 1, \quad \forall t \in T, s \in S$.

   Durch eine Und-Verknüpfung können die beiden Gleichungen anschließend kombiniert werden.

   Da Implikationen in der PyOptInterface-Syntax nicht möglich sind, muss die Formel umformuliert werden:

$$\begin{aligned}
c_1 = y_{a,s} \leq x_{a,t}, & \qquad \forall t \in T, s \in S \\
c_2 = x_{a,t} \leq y_{a,s}, & \qquad \forall t \in T, s \in S \\
c_{ges} = c_1 \wedge c_2 & \\
y_{a,s} \leq x_{a,t} \wedge x_{a,t} \leq y_{a,s}, & \qquad \forall t \in T, s \in S \\
y_{a,s} - x_{a,t} \leq 0 \wedge x_{a,t} - y_{a,s} \leq 0, & \qquad \forall t \in T, s \in S
\end{aligned}$$

   Dass die logische Und-Verknüpfung korrekt ist, zeigt folgende Tabelle (ausgehend für einen willkürlichen Agenten a):

s	t	Soll-Wert	c_1	c_2	$c_1 \wedge c_2$
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	1	1	1	1

4. **Maximale Anzahl an Schichten**

   Jeder Agent darf maximal $S_a = 3$ Schichten pro Tag übernehmen:

$$\sum_{s \in S} y_{a,s} \leq S_a, \quad \forall a \in A$$

5. **Binäre Entscheidungsvariablen**

   Die Variablen müssen binär sein:

$$x_{a,t}, y_{a,s} \in \{0, 1\}, \quad \forall a, t, s$$

Diese mathematische Formulierung deckt die Anforderungen des Problems ab und dient als präzise Grundlage für die Implementierung eines Optimierungsmodells zur Schichtplanung in Python.

$$r = round(max(r_{min}, min(r_{max}, r_{max} - \alpha \cdot \text{exceedance} - \beta \cdot \text{error} + c)), 2)$$

Variable	**Bedeutung**
r	finale Bewertung
r_{min}	minimal mögliche Bewertung
r_{max}	maximal mögliche Bewertung
α	Gewichtung der Wartezeitstrafe
exceedance	Quotient aus der tatsächlichen Wartezeit und der angegebenen maximalen Zeit (nur bei Überschreitung, sonst 0)
β	Gewichtung für fehlerhafte Bestellungen
error	zufällige Fehlerquote
c	zufällige Bewertungsvariabilität abhängig von der Gruppengröße

4 Offene Fragen

5 Resultate