ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA HỆ THỐNG THÔNG TIN

BÁO CÁO ĐỒ ÁN

HỆ THỐNG HỖ TRỢ QUYẾT ĐỊNH

Chủ đề:

SIMPLE DECISION

GVHD: **Th.S Nguyễn Hồ Duy Trí**

Lớp: **IS254. P11**

Nhóm thực hiện:

Không	Họ và tên	Mã học sinh
1	Nguyễn Dương Chí Tâm	21520439
2	Nguyễn Trương Đình Giang	21520215
3	Đỗ Hiền Thảo	21520460
4	Huỳnh Mạnh Huy	21520259

LỜI CẢM ƠN

Lời đầu tiên, nhóm chúng em xin gửi lời cảm ơn chân thành đến Thầy Nguyễn Hồ Duy Trí vì đã chia sẻ những kiến thức chuyên môn quý báu, đồng thời tận tình hướng dẫn và hỗ trợ chúng em trong suốt quá trình thực hiện đồ án. Sự đồng hành và tận tâm của Thầy thực sự là nguồn động lực to lớn, giúp chúng em tự tin hoàn thành công việc.

Đồ án lần này không chỉ là cơ hội để mỗi thành viên trong nhóm hợp tác và phát triển kỹ năng làm việc nhóm, mà còn là dịp để chúng em học hỏi lẫn nhau, áp dụng kiến thức đã được học, và sáng tạo trong việc triển khai sản phẩm. Dù đã cố gắng hết sức, chúng em biết rằng không thể tránh khỏi những sai sót do giới hạn về thời gian, kinh nghiệm và khả năng. Vì vậy, rất mong nhận được những ý kiến đóng góp quý giá từ Thầy để nhóm có thể hoàn thiện hơn, không chỉ cho đồ án lần này mà còn cho những đồ án của môn khác trong tương lai.

Cuối cùng, nhóm chúng em xin kính chúc Thầy thật nhiều sức khỏe và thành công để tiếp tục truyền đạt kiến thức cho các bạn sinh viên khóa tiếp theo. Một lần nữa, xin cảm ơn Thầy!

Nhóm thực hiện,

Chí Tâm – Đình Giang – Hiền Thảo – Mạnh Huy.

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI SIMPLE DECISION

1. Bối cảnh và ý nghĩa

Trong cuộc sống hàng ngày, việc đưa ra quyết định là một kỹ năng quan trọng giúp chúng ta giải quyết các vấn đề và đạt được mục tiêu. Từ những quyết định đơn giản như lựa chọn bữa ăn cho đến các quyết định phức tạp hơn trong công việc hay học tập, mỗi lựa chọn đều ảnh hưởng đến kết quả cuối cùng. Đề tài **"Simple Decision"** ra đời với mục tiêu giúp người dùng hiểu và ứng dụng các phương pháp đưa ra quyết định một cách dễ dàng, nhanh chóng và hiệu quả.

2. Mục tiêu nghiên cứu

- Xây dựng hệ thống hỗ trợ ra quyết định dựa trên các tiêu chí đơn giản.
- Phân tích và lựa chọn các phương pháp phù hợp cho việc đưa ra quyết định, đảm bảo tính khả thi và chính xác.
- Tích hợp các công cụ hỗ trợ trực quan để người dùng dễ dàng áp dụng.

3. Phạm vi nghiên cứu

Đề tài tập trung vào các quyết định đơn giản, thường gặp trong cuộc sống hàng ngày hoặc trong môi trường học tập, làm việc. Các tình huống cụ thể có thể bao gồm:

- Lựa chọn sản phẩm hoặc dịch vụ dựa trên các tiêu chí nhất định.
- Đánh giá và so sánh các phương án khi có dữ liệu hạn chế.
- Hỗ trợ cá nhân hoặc nhóm trong việc ra quyết định nhanh.

4. Ý nghĩa thực tiễn

Việc nghiên cứu và ứng dụng các phương pháp ra quyết định không chỉ giúp nâng cao khả năng giải quyết vấn đề mà còn tiết kiệm thời gian, tối ưu hóa nguồn lực, và giảm thiểu các rủi ro từ việc đưa ra lựa chọn sai lầm. Đề tài này sẽ mang lại một công cụ hữu ích, dễ sử dụng và phù hợp với nhiều đối tượng người dùng.

CHƯƠNG 2: NỘI DUNG VỀ SIMPLE DICISION

2. Khái niệm, đặc điểm và vai trò của Simple Dicision

Khái niệm:

Simple Decision (Quyết định đơn giản) là quá trình lựa chọn một giải pháp hoặc hành động từ một tập hợp các phương án dựa trên những tiêu chí hoặc điều kiện cụ thể, thường trong bối cảnh ít phức tạp và dễ dàng đưa ra quyết định. Đây là loại quyết định không đòi hỏi nhiều dữ liệu, tính toán phức tạp hoặc phân tích sâu rộng mà chủ yếu dựa trên các yếu tố cơ bản và rõ ràng.

Các đặc điểm chính:

- **Ít yếu tố tác động:** Các quyết định đơn giản thường chỉ liên quan đến một hoặc một vài yếu tố quan trọng.
  Ví dụ: Chọn món ăn trong thực đơn, chọn quần áo mặc hàng ngày.
- **Không yêu cầu phân tích phức tạp:** Quyết định thường dựa vào thông tin sẵn có, kinh nghiệm cá nhân, hoặc trực giác.
  Ví dụ: Quyết định đi bộ hay đi xe khi quãng đường ngắn.
- **Thời gian ra quyết định ngắn:** Do mức độ đơn giản, các quyết định này không mất nhiều thời gian suy nghĩ.
- **Ít rủi ro:** Hậu quả của việc đưa ra quyết định thường không quá nghiêm trọng nếu chọn sai.
  Ví dụ: Chọn mua một loại trái cây thay vì loại khác.

Vai trò:

- **Tiết kiệm thời gian:** Giúp xử lý các tình huống thường ngày nhanh chóng.
- **Giảm căng thẳng:** Tránh việc phải suy nghĩ quá nhiều về những vấn đề nhỏ nhặt.
- **Tạo nền tảng cho các quyết định lớn:** Các quyết định đơn giản thường hỗ trợ hoặc là bước đầu để giải quyết các bài toán phức tạp hơn.

Ví dụ thực tế:

- Lựa chọn đường đi ngắn nhất khi không gặp vấn đề về thời tiết hoặc giao thông.

- Quyết định mua sản phẩm có giá hợp lý và phù hợp với nhu cầu cơ bản mà không cần cân nhắc nhiều lựa chọn.

Simple Decision tuy nhỏ gọn và đơn giản nhưng là một phần quan trọng trong cuộc sống hàng ngày, giúp tối ưu hóa các hoạt động thường nhật và hỗ trợ sự tập trung cho những quyết định lớn hơn.

2.1 Nội dung và thuật toán về Simple Dicision

2.1.1 Constraints on Rational Preferences (Ràng buộc trên các sở thích hợp lý)

Phần này tập trung vào việc định nghĩa các ràng buộc cần thiết để đảm bảo các sở thích của một cá nhân (hoặc tác nhân) là hợp lý khi họ đưa ra quyết định trong điều kiện không chắc chắn. Những ràng buộc này là cơ sở để phát triển các mô hình toán học, cho phép chúng ta sử dụng lý thuyết tiện ích để mô tả và tối ưu hóa quá trình ra quyết định.

Chúng ta thể hiện sở thích của mình bằng các toán tử sau:

- $A \succ B$: Nếu chúng ta ưa thích A hơn B.
- $A \sim B$: Nếu chúng ta không phân biệt được sự khác biệt giữa A và B.
- $A \succeq B$: Nếu chúng ta ưa thích A hơn hoặc không phân biệt giữa A và B.

Giống như niềm tin có thể mang tính chủ quan, sở thích cũng có thể mang tính chủ quan.

Ngoài việc so sánh các sự kiện, các toán tử sở thích này có thể được sử dụng để so sánh các sở thích đối với các kết quả không chắc chắn. Một xổ số (lottery) là một tập hợp các xác suất liên kết với một tập hợp các kết quả.

Ví dụ: Nếu S1, S2, ..., Sn là một tập hợp các kết quả và p1, p2, ..., pn là các xác suất tương ứng của chúng, thì xổ số liên quan đến các kết quả và xác suất này được viết như sau:

$$[S1 : p1; S2 : p2; ...; Sn : pn]$$

Sở thích hợp lý là: Khi một người hoặc một hệ thống phải chọn giữa các kết quả khác nhau, họ thường dựa vào sở thích (preferences). Ví dụ:

Bạn có thể thích kết quả A (một chiếc xe đắt tiền) hơn kết quả B (một chiếc xe giá rẻ), và B hơn C (không có xe).

Để đảm bảo rằng những sở thích này có thể được mô hình hóa, chúng phải tuân theo một số nguyên tắc hợp lý.

- **Hoàn chỉnh (Completeness):** Chỉ có một trong ba mối quan hệ sau đây là đúng: $A \succ B$, $B \succ A$, hoặc $A \sim B$.

Nếu không có tính hoàn chỉnh, sẽ không thể xây dựng được một mô hình nhất quán.

- **Tính bắc cầu (Transitivity):** Nếu $A \succeq B$ và $B \succeq C$, thì $A \succeq C$.

Tính bắc cầu đảm bảo rằng các sở thích không bị mâu thuẫn, một điều kiện quan trọng để xây dựng các mô hình tiện ích nhất quán.

- **Liên tục (Continuity):** Nếu $A \succ C \succ B$, thì tồn tại một xác suất sao cho:

$$[A : p; B : 1 - p] \sim C$$

Điều này có nghĩa là một người có thể "làm trung hòa" giữa các sở thích thông qua các xác suất khác nhau, dẫn đến một mô hình toán học liên tục.

- **Độc lập (Independence):** Nếu $A \succ B$, thì với mọi C và xác suất p, ta có:

$$[A : p; C : 1 - p] \succ [B : p; C : 1 - p]$$

Điều này đảm bảo rằng các sở thích không bị ảnh hưởng bởi các kết quả bên ngoài, giúp giữ tính nhất quán.

Những ràng buộc này được áp đặt đối với các sở thích hợp lý. Chúng không nói gì về sở thích thực tế. Thực tế, có rất nhiều bằng chứng cho thấy không phải lúc nào cũng hợp lý. Mục tiêu chính là hiểu việc ra quyết định hợp lý từ góc độ tính toán, để chúng ta có thể xây dựng các hệ thống hữu ích. Việc mở rộng lý thuyết này nhằm hiểu rằng hành vi ra quyết định chỉ là mối quan tâm thứ yếu.

❖ **Ví dụ minh hoạ:**

```
import random

# Định nghĩa hàm tiện ích cho các kết quả
utility = {
    "A": 10,  # Giá trị tiện ích của kết quả A
    "B": 7,   # Giá trị tiện ích của kết quả B
    "C": 5    # Giá trị tiện ích của kết quả C
}
```

1. **Tính hoàn chỉnh:**

```python
# 1. Hoàn chỉnh
def check_completeness(result1, result2):
    """Kiểm tra tính hoàn chỉnh: Có thể so sánh hai kết quả."""
    if utility[result1] > utility[result2]:
        return f"{result1} ≻ {result2}"
    elif utility[result1] < utility[result2]:
        return f"{result2} ≻ {result1}"
    else:
        return f"{result1} ~ {result2}"
```

```python
print("1. Hoàn chỉnh:")
print(check_completeness("A", "B"))
print(check_completeness("B", "C"))
print()
```

```
1. Hoàn chỉnh:
A ≻ B
B ≻ C
```

➢ Kết quả:

Giá trị của A là 10, B là 7, C là 5.

- *A ≻ B vì 10 ≻ 7*
- *B ≻ C vì 7 ≻ 5*

2. Tính bắc cầu:

```python
# 2. Tính bắc cầu
def check_transitivity(result1, result2, result3):
    """Kiểm tra tính bắc cầu."""
    if utility[result1] >= utility[result2] and utility[result2] >= utility[result3]:
        if utility[result1] >= utility[result3]:
            return f"Tính bắc cầu thỏa mãn: {result1} ≽ {result2}, {result2} ≽ {result3} → {result1} ≽ {result3}"
    return f"Tính bắc cầu không thỏa mãn."
```

```python
print("2. Tính bắc cầu:")
print(check_transitivity("A", "B", "C"))
print()
```

```
2. Tính bắc cầu:
Tính bắc cầu thỏa mãn: A ≽ B, B ≽ C → A ≽ C
```

➢ Kết quả:

A ≻ B và B ≻ C (Vì 10 ≻ 7, 7 ≻ 5) nên ta có thể kết luận: A ≻ C (10 ≻ 5)

3. Tính liên tục:

```python
# 3. Liên tục
def check_continuity(result1, result2, result3, p):
    """Kiểm tra tính liên tục với xác suất p."""
    lottery_utility = p * utility[result1] + (1 - p) * utility[result3]
    if abs(lottery_utility - utility[result2]) < 1e-6:  # Kiểm tra xấp xỉ
        return f"Tính liên tục thỏa mãn với p = {p}: [{result1}: {p}; {result3}: {1-p}] ~ {result2}"
    return f"Tính liên tục không thỏa mãn với p = {p}."
```

```python
print("3. Liên tục:")
p = 0.4
print(check_continuity("A", "B", "C", p))
print()
```

```
3. Liên tục:
Tính liên tục thỏa mãn với p = 0.4: [A: 0.4; C: 0.6] ~ B
```

➢ Kết quả:

*Ta có: A * p + C * (1 - p) = 10 * 0.4 + 5 * 0.6 = 7 ~ B*

4. Tính độc lập:

```python
# 4. Độc lập
def check_independence(result1, result2, result3, p):
    """Kiểm tra tính độc lập."""
    lottery_1 = p * utility[result1] + (1 - p) * utility[result3]
    lottery_2 = p * utility[result2] + (1 - p) * utility[result3]
    if lottery_1 > lottery_2:
        return f"Tính độc lập thỏa mãn: [{result1}: {p}; {result3}: {1-p}] ≻ [{result2}: {p}; {result3}: {1-p}]"
    elif lottery_1 < lottery_2:
        return f"Tính độc lập không thỏa mãn: [{result2}: {p}; {result3}: {1-p}] ≻ [{result1}: {p}; {result3}: {1-p}]"
    else:
        return f"Tính độc lập thỏa mãn: [{result1}: {p}; {result3}: {1-p}] ~ [{result2}: {p}; {result3}: {1-p}]"
```

```python
print("4. Độc lập:")
p = 0.7
print(check_independence("A", "B", "C", p))
```

```
4. Độc lập:
Tính độc lập thỏa mãn: [A: 0.7; C: 0.3] ≻ [B: 0.7; C: 0.3]
```

➢ Kết quả:

*[A : 0.7; C : 0.3] = 10 * 0.7 + 5 * 0.3 = 8.5*

*[B : 0.7; C : 0.3] = 7 * 0.7 + 5 * 0.3 = 6.4*

Vậy [A : 0.7; C : 0.3] ≻ [B : 0.7; C : 0.3]

2.1.2 Utility Functions (Hàm tiện ích)

Cũng như các ràng buộc trong việc so sánh tính khả dĩ của các phát biểu khác nhau dẫn đến sự tồn tại của một thước đo xác suất dạng số thực, các ràng buộc trên sở thích hợp lý dẫn đến sự tồn tại của một thước đo tiện ích dạng số thực. Từ các ràng buộc về sở thích hợp lý, ta suy ra rằng tồn tại một hàm tiện ích U dạng số thực sao cho:

- $U(A) > U(B)$ *nếu và chỉ nếu* $A \succ B$, *và*
- $U(A) = U(B)$ *nếu và chỉ nếu* $A \sim B$.

Hàm tiện ích giúp lượng hóa các sở thích hợp lý của cá nhân hoặc hệ thống đối với các trạng thái hoặc kết quả. Chúng được xây dựng dựa trên các ràng buộc hợp lý trong việc so sánh các lựa chọn.

Ví dụ:

- **Nếu bạn thích A hơn B**, hàm tiện ích sẽ đảm bảo $U(A) > U(B)$.
- **Nếu bạn thấy A và B ngang nhau**, thì $U(A) = U(B)$.

Hàm tiện ích giúp chuyển sở thích định tính (ví dụ: "thích", "ngang nhau") thành số học để dễ dàng phân tích và so sánh.

Hàm tiện ích là duy nhất theo một **phép biến đổi affine dương**. Nói cách khác, với mọi hằng số $m > 0$ *và* $b, U'(S) = mU(S) + b$ nếu và chỉ nếu sở thích được xác định bởi U'giống với U và giữ nguyên thứ tự ưu tiên giữa các trạng thái.

$$m > 0 \text{ và } b, U'(S) = mU(S) + b$$

Trong đó:

- $m > 0$: Đảm bảo rằng hướng của sở thích không bị thay đổi.
- b: Dịch chuyển toàn bộ hàm tiện ích, không làm thay đổi quan hệ giữa các trạng thái.

Ví dụ:

- Nhiệt độ có thể được đo bằng Kelvin, Celsius, hoặc Fahrenheit. Các thang đo này là các phép biến đổi affine của nhau nhưng không làm thay đổi ý nghĩa so sánh giữa các nhiệt độ.

Từ các ràng buộc về sở thích hợp lý, **tiện ích của một xổ số**. Công thức này phản ánh cách con người ra quyết định khi đối mặt với rủi ro:

$$U([S_1 : p_1; S_2 : p_2; \cdots S_n : p_n]) = \sum_{i=1}^{n} p_i U(S_i)$$

Trong đó:

- S_i: Các kết quả có thể xảy ra.
- p_i: Xác suất của từng kết quả S_i
- $U(S_i)$: Tiện ích của từng kết quả S_i.

Ví dụ:

Giả sử bạn tham gia một trò chơi với hai kết quả:

- S_1: Nhận được 1 triệu đồng với xác suất 70% ($p_1 = 0.7$), tiện ích $U(S_i) = 1$
- S_2: Nhận được 100 nghìn đồng với xác suất 30% ($p_2 = 0.3$), tiện ích $U(S_i) = 0.1$

Tiện ích của trò chơi:

$$U = (0.7 \cdot 1) + (0.3 \cdot 0.1) = 0.73$$

Kết quả này cho thấy trò chơi có giá trị kỳ vọng khá cao về tiện ích.

Nếu hàm tiện ích bị chặn, ta có thể định nghĩa một **hàm tiện ích chuẩn hóa**, trong đó kết quả tốt nhất được gán tiện ích là 1 và kết quả tệ nhất được gán tiện ích là 0. Tiện ích của các kết quả khác sẽ được tỷ lệ hóa và dịch chuyển phù hợp.

Hàm tiện ích có thể được chuẩn hóa để tiện lợi hơn trong việc so sánh:

- **Trạng thái tốt nhất**: Gán tiện ích $U = 1$.
- **Trạng thái tệ nhất**: Gán tiện ích $U = 0$.
- **Các trạng thái còn lại**: Tiện ích được tính theo tỉ lệ giữa trạng thái tốt nhất và tệ nhất.

Ví dụ:

- Một hệ thống có ba trạng thái:
  - $S_{tốt\ nhất}$: Không có va chạm (gán $U = 1$).
  - $S_{trung\ bình}$: Báo động nhưng không va chạm (gán $U = 0.5$).
  - $S_{tệ\ nhất}$: Báo động sai và có va chạm (gán $U = 0$).

Hàm chuẩn hóa đảm bảo tiện ích nằm trong khoảng $[0, 1]$ giúp trực quan hóa và so sánh hiệu quả hơn.

❖ **Ví dụ minh họa:**

1. **Phép biến đổi affine dương:**

$$U(S_1) = 0.7, U(S_2) = 0.4, U(S_3) = 0.2 \; với \; m = 2, b = 0.5$$

```python
#1. Phép biến đổi affine dương:
import numpy as np

# Phép biến đổi affine dương
def affine_transform(U, m, b):
    return m * np.array(U) + b

# Tiện ích ban đầu
utilities = [0.7, 0.4, 0.2]

# Tham số affine
m, b = 2, 0.5

# Tính tiện ích sau phép biến đổi
U_affine = affine_transform(utilities, m, b)

# Kết quả
print("Tiện ích ban đầu:", utilities)
print("Tiện ích sau phép biến đổi affine:", U_affine)
```

➢ Kết quả:

```
Tiện ích ban đầu: [0.7, 0.4, 0.2]
Tiện ích sau phép biến đổi affine: [1.9 1.3 0.9]
```

➢ Áp dụng công thức:

- $U'(S_1) = 2 \cdot 0.7 + 0.5 = 1.9$
- $U'(S_1) = 2 \cdot 0.4 + 0.5 = 1.3$
- $U'(S_1) = 2 \cdot 0.2 + 0.5 = 0.9$

2. **Tiện ích của một xổ số:**

- $S_1 : U(S_1) = 1, p_1 = 0.6$

- $S_2 : U(S_2) = 0.5, p_2 = 0.3$
- $S_3 : U(S_3) = 0, p_3 = 0.1$

```python
#2. Tiện ích của một xổ số
def lottery_utility(probabilities, utilities):
    return np.dot(probabilities, utilities)

# Dữ liệu xổ số
probabilities = [0.6, 0.3, 0.1]
utilities = [1, 0.5, 0]

# Tính tiện ích kỳ vọng
U_lottery = lottery_utility(probabilities, utilities)

# Kết quả
print("Tiện ích của xổ số:", U_lottery)
```

➢ Kết quả:

```
Tiện ích của xổ số: 0.75
```

➢ Áp dụng công thức:

$$U(lottery) = (0.6 \cdot 1) + (0.3 \cdot 0.5) + (0.1 \cdot 0) = 0.75$$

3. Hàm tiện ích chuẩn hóa:

$$U(S_1) = 50, U(S_2) = 30, U(S_3) = 10$$

```python
#3. Hàm tiện ích chuẩn hóa
def normalize_utility(U, U_min, U_max):
    return (np.array(U) - U_min) / (U_max - U_min)

# Tiện ích ban đầu
utilities = [50, 30, 10]

# Tính tiện ích chuẩn hóa
U_min, U_max = min(utilities), max(utilities)
U_normalized = normalize_utility(utilities, U_min, U_max)

# Kết quả
print("Tiện ích ban đầu:", utilities)
print("Tiện ích chuẩn hóa:", U_normalized)
```

- Kết quả:

```
Tiện ích ban đầu: [50, 30, 10]
Tiện ích chuẩn hóa: [1.  0.5 0. ]
```

- Áp dụng công thức:
  - $U_{normalized}(S_1) = \frac{50-10}{50-10} = 1$
  - $U_{normalized}(S_2) = \frac{30-10}{50-10} = 0.5$
  - $U_{normalized}(S_3) = \frac{10-10}{50-10} = 0$

2.1.3 Utility Elicitation (Lấy tiện ích)

Khi xây dựng một hệ thống ra quyết định hoặc hỗ trợ ra quyết định, việc suy ra hàm tiện ích từ một cá nhân hoặc một nhóm người thường rất hữu ích. Phương pháp này được gọi là **lấy tiện ích** hoặc **lấy sở thích**. Một cách tiếp cận là cố định tiện ích của kết quả tệ nhất $S_{tệ\ nhất}$ là 0 và của kết quả tốt nhất $S_{tốt\ nhất}$ là 1. Miễn là các giá trị tiện ích bị chặn, chúng ta có thể dịch chuyển và tỷ lệ hóa các tiện ích mà không làm thay đổi sở thích.

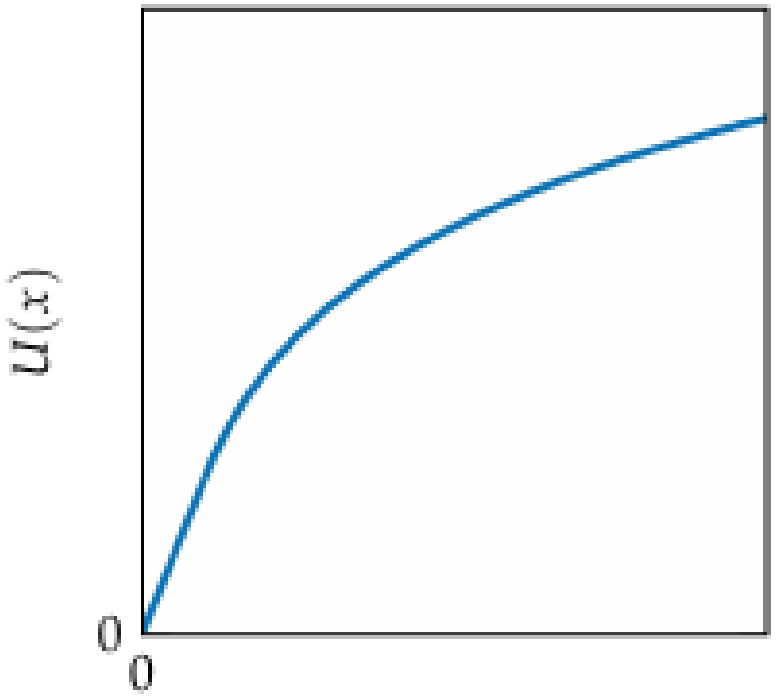
Nếu chúng ta muốn xác định tiện ích của một kết quả S, ta tìm một xác suất p sao cho:

$$S \sim [S_{tốt\ nhất} : p; S_{tệ\ nhất} : 1 - p]$$

- Khi đó, tiện ích $U(S) = p$.

Tiện ích dựa trên giá trị tiền tệ

Thay vì cố gắng tối đa hóa giá trị tài sản kỳ vọng, chúng ta thường muốn tối đa hóa tiện ích kỳ vọng của tài sản. Tất nhiên, mỗi người có hàm tiện ích riêng. Hình 6.1 minh họa một ví dụ về hàm tiện ích. Với các khoản tài sản nhỏ, đường cong này gần như tuyến tính, trong đó 100 đô la tương ứng với mức tốt gấp đôi so với 50 đô la. Tuy nhiên, với các khoản tiền lớn hơn, đường cong có xu hướng phẳng ra; ví dụ, 1.000 đô la có giá trị ít hơn đối với một tỷ phú so với một người bình thường. Hiện tượng này được gọi là **giảm dần tiện ích biên (diminishing marginal utility)**.



Ba loại tiện ích dựa trên rủi ro:

Khi thảo luận về các hàm tiện ích tiền tệ, thường sử dụng ba thuật ngữ sau:

- **Trung lập với rủi ro (Risk neutral):** Hàm tiện ích tuyến tính. Không có sự ưu tiên giữa việc nhận 50 đô la hoặc 50% cơ hội nhận 100 đô la ($A \sim B$).
- **Thích rủi ro (Risk seeking):** Hàm tiện ích lồi. Có sự ưu tiên đối với cơ hội 50% nhận 100 đô la ($A \prec B$).
- **Không thích rủi ro (Risk averse):** Hàm tiện ích lõm. Có sự ưu tiên đối với việc nhận chắc chắn 50 đô la ($A \succ B$).

Các dạng hàm tiện ích phổ biến:

1. **Hàm tiện ích bậc hai (quadratic utility):**

$$U(x) = \lambda x - x^2$$

Trong đó, tham số $\lambda > 0$ kiểm soát mức độ ngại rủi ro. Để mô hình hóa tiện ích của các đại lượng như tài sản, chúng ta thường giới hạn hàm này tại $x = \lambda/2$, bởi vì sau điểm đó, tiện ích bắt đầu giảm.

2. **Hàm tiện ích mũ (exponential utility):**

$$U(x) = 1 - e^{-\lambda x}$$

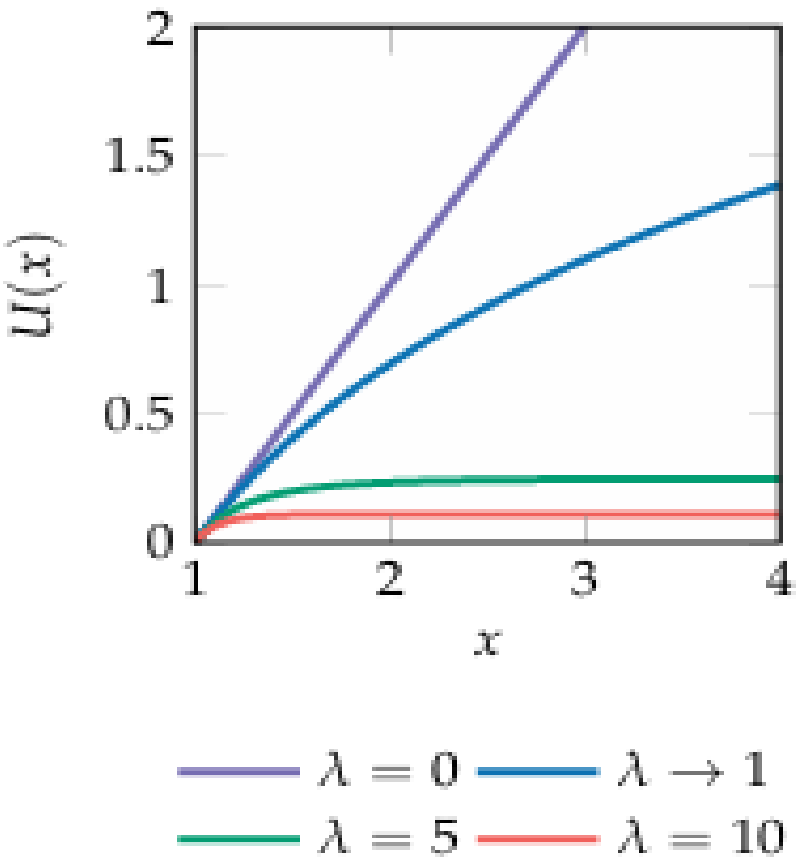
Với $\lambda > 0$. Hàm này có dạng toán học thuận tiện, nhưng thường không được coi là mô hình phù hợp cho tiện ích tài sản.

3. **Hàm tiện ích lũy thừa (power utility):**

$$U(x) = \frac{x^{1-\lambda}}{1-\lambda}, \lambda \geq 0, \lambda \neq 1$$

Trong trường hợp đặc biệt $\lambda \to 1$, hàm này trở thành **hàm tiện ích logarithmic**:

$$U(x) = \log(x), x > 0$$



Loại tiện ích	Đặc điểm	Ý nghĩa
Bậc hai (Quadratic)	Hình parabol, tăng đến một giá trị tối đa rồi giảm.	Phù hợp khi cần giới hạn tiện ích, thường mô hình hóa ngại rủi ro.
Mũ (Exponential)	Hàm lõm, tiệm cận 1 khi $x \to \infty$.	Thích hợp trong các bài toán bảo hiểm hoặc đầu tư, mô hình hóa ngại rủi ro mạnh.
Lũy thừa (Power)	Tùy theo λ : tuyến tính (trung lập), lõm (ngại rủi ro), hoặc lồi (ưa rủi ro).	Linh hoạt để mô hình hóa nhiều loại sở thích khác nhau, đặc biệt trong tài chính.
Logarithmic (Đặc biệt)	Trường hợp đặc biệt của hàm lũy thừa, tăng chậm và phản ánh mức độ ngại rủi ro trung bình.	Được dùng rộng rãi trong kinh tế học, biểu diễn mức độ thỏa mãn giảm dần với tài sản tăng.

❖ **Ví dụ minh họa:**

1. **Hàm tiện ích bậc hai (quadratic utility):**
   Giả sử $\lambda = 3$
   Tính tiện ích tại $x = 1, x = 2, x = 3$

```python
#1. Hàm tiện ích bậc hai:
def quadratic_utility(x, lambd=3):
    return lambd * x - x**2

# Các giá trị x cần tính
x_values = [1, 2, 3]

# Tính tiện ích cho từng giá trị x
utilities = [quadratic_utility(x) for x in x_values]

# Xuất kết quả
for x, u in zip(x_values, utilities):
    print(f"U({x}) = {u}")
```

- Kết quả:

```
U(1) = 2
U(2) = 2
U(3) = 0
```

- Áp dụng công thức:
  - $U(1) = 3 \cdot 1 - 1^2 = 2$
  - $U(2) = 3 \cdot 2 - 2^2 = 2$
  - $U(3) = 3 \cdot 3 - 3^2 = 0$

2. **Hàm tiện ích mũ (exponential utility):**
   Giả sử $\lambda = 0.2$
   Tính tiện ích tại $x = 1, x = 2, x = 3$

```python
#2. Hàm tiện ích mũ
def exponential_utility(x, lambd=0.2):
    return 1 - np.exp(-lambd * x)

# Các giá trị x cần tính
x_values = [1, 2, 3]

# Tính tiện ích cho từng giá trị x
utilities = [exponential_utility(x) for x in x_values]

# Xuất kết quả
for x, u in zip(x_values, utilities):
    print(f"U({x}) = {u:.4f}")
```

- Kết quả:

```
U(1) = 0.1813
U(2) = 0.3297
U(3) = 0.4512
```

- Áp dụng công thức:
  - $U(1) = 1 - e^{-0.2 \cdot 1} = 0.1813$
  - $U(2) = 1 - e^{-0.2 \cdot 2} = 0.3297$
  - $U(3) = 1 - e^{-0.2 \cdot 3} = 0.4512$

3. **Hàm tiện ích lũy thừa (power utility):**
   Giả sử $\lambda = 0.5$
   Tính tiện ích tại $x = 1, x = 2, x = 3$

```python
#3. Hạm tiện ích lũy thừa:
def power_utility(x, lambd=0.5):
    if lambd == 1:
        return np.log(x)  # Trường hợp đặc biệt
    return (x**(1 - lambd)) / (1 - lambd)

# Các giá trị x cần tính
x_values = [1, 2, 3]

# Tính tiện ích cho từng giá trị x
utilities = [power_utility(x) for x in x_values]

# Xuất kết quả
for x, u in zip(x_values, utilities):
    print(f"U({x}) = {u:.4f}")
```

- Kết quả:

```
U(1) = 2.0000
U(2) = 2.8284
U(3) = 3.4641
```

- Áp dụng công thức:
  - $U(1) = \frac{1^{1-0.5}}{1-0.5} = 2$
  - $U(2) = \frac{2^{1-0.5}}{1-0.5} \approx 2.8284$
  - $U(3) = \frac{3^{1-0.5}}{1-0.5} \approx 3.4641$

2.1.4 Maximum Expected Utility Principle (Nguyên lý Tiện ích Kỳ vọng Tối đa)

Nguyên lý Tiện ích Kỳ vọng Tối đa (Maximum Expected Utility Principle) là một nguyên tắc cơ bản trong lý thuyết quyết định, giúp tác nhân lựa chọn hành động tối ưu khi đối mặt với sự không chắc chắn.

Nguyên tắc cốt lõi:

- Tác nhân ra quyết định sẽ chọn hành động tối đa hóa **giá trị kỳ vọng của tiện ích.**
- **Tiện ích** là một hàm số ánh xạ từ các trạng thái của thế giới tới các giá trị thực, biểu thị mức độ "hài lòng" hoặc lợi ích mà tác nhân đạt được khi trạng thái đó xảy ra.
- Kỳ vọng được tính dựa trên xác suất xảy ra của các trạng thái đó, kết hợp với giá trị tiện ích tương ứng.

Tóm lại, nguyên lý này dựa trên khái niệm tiện ích kỳ vọng là giá trị trung bình có trọng số của các kết quả dự kiến, với trọng số là xác suất xảy ra của từng kết quả.

Công thức:

$$EU(a|o) = \sum_{s'} P(s'|a,o)U(s')$$

Trong đó:

- $EU(a|o)$: Tiện ích kỳ vọng khi thực hiện hành động a dựa trên quan sát o.
- $P(s'|a,o)$: Xác suất có điều kiện, thể hiện khả năng trạng thái s' xảy ra nếu tác nhân thực hiện hành động a, sau khi quan sát o.
- $U(s')$: Giá trị tiện ích của trạng thái s', thể hiện mức độ hài lòng hoặc lợi ích mà trạng thái đó mang lại.

 Tiện ích kỳ vọng của hành động a, khi quan sát o, là giá trị trung bình của tiện ích U(s') trên trạng thái s', với trọng số xác suất $P(s'|a,o)$.

Nguyên tắc tối đa hóa kỳ vọng tiện ích nói rằng một tác nhân hợp lý nên chọn hành động **tối đa** hóa kỳ vọng tiện ích.

Hành động tối ưu a* được chọn bằng cách:

$$a^* = argmax_a EU(a|o)$$

Trong đó:

- a*: Hành động tối ưu.
- $EU(a|o)$: Tiện ích kỳ vọng của hành động a, đã được tính bằng công thức trên.

- $argmax_a$: Là toán tử, dùng để tìm giá trị của hành động a mà hàm $EU(a|o)$ đạt GTLN.

❖ **Ví dụ minh họa:**

Tình huống: **Quyết định có nên mua bảo hiểm cho xe hay không?**

		States (s')	
		Xe bị tai nạn (s_1) (P = 0.1)	**Xe không bị tai nạn** (s_2) (P = 0.9)
Acts (a)	**Mua bảo hiểm** (a_1)	Chi phí bảo hiểm được chi trả (U = -200)	Chỉ trả tiền bảo hiểm, không tai nạn (U = -100)
	Không mua bảo hiểm (a_2)	Chịu toàn bộ chi phí tai nạn (U = -5000)	Không tốn gì (U = 0)

1. Xác định Hành động, trạng thái, xác suất, tiện ích

```
# 1. Define Actions, States, Xác suất, Tiện ích
actions = ["Action 1 (Buy Insurance)", "Action 2 (No Insurance)"] #Mua bảo hiểm; Không mua bảo hiểm
states = ["State 1 (Accident)", "State 2 (No Accident)"] #Tai nạn; Không tai nạn

# Xác suất P(s|a): Xác suất trạng thái s xảy ra khi chọn hành động a
probabilities = {
    "Action 1 (Buy Insurance)": [0.1, 0.9],  # Xác suất tai nạn và không tai nạn khi mua bảo hiểm
    "Action 2 (No Insurance)": [0.1, 0.9]  # Xác suất tai nạn và không tai nạn khi không mua bảo hiểm
}

# Tiện ích U(s): Giá trị tiện ích của mỗi trạng thái
utilities = {
    "State 1 (Accident)": {"Action 1 (Buy Insurance)": -200, "Action 2 (No Insurance)": -5000},
    "State 2 (No Accident)": {"Action 1 (Buy Insurance)": -100, "Action 2 (No Insurance)": 0}
}
```

2. Tính toán tiện ích kỳ vọng cho từng hành động

```
# 2. Tính toán tiện ích kỳ vọng cho từng hành động
def compute_expected_utility(action, probabilities, utilities):
    return sum(probabilities[action][i] * utilities[states[i]][action] for i in range(len(states)))

expected_utilities = {action: compute_expected_utility(action, probabilities, utilities) for action in actions}
```

- Nếu **Mua bảo hiểm** (a_1):

$$EU(a_1) = P(s_1).U(s_1) + P(s_2).U(s_2)$$
$$= 0.1 * (-200) + 0.9 * (-100) = -110$$

- Nếu **Không mua bảo hiểm** (a_2):

$$EU(a_2) = P(s_1).U(s_1) + P(s_2).U(s_2) = -500$$

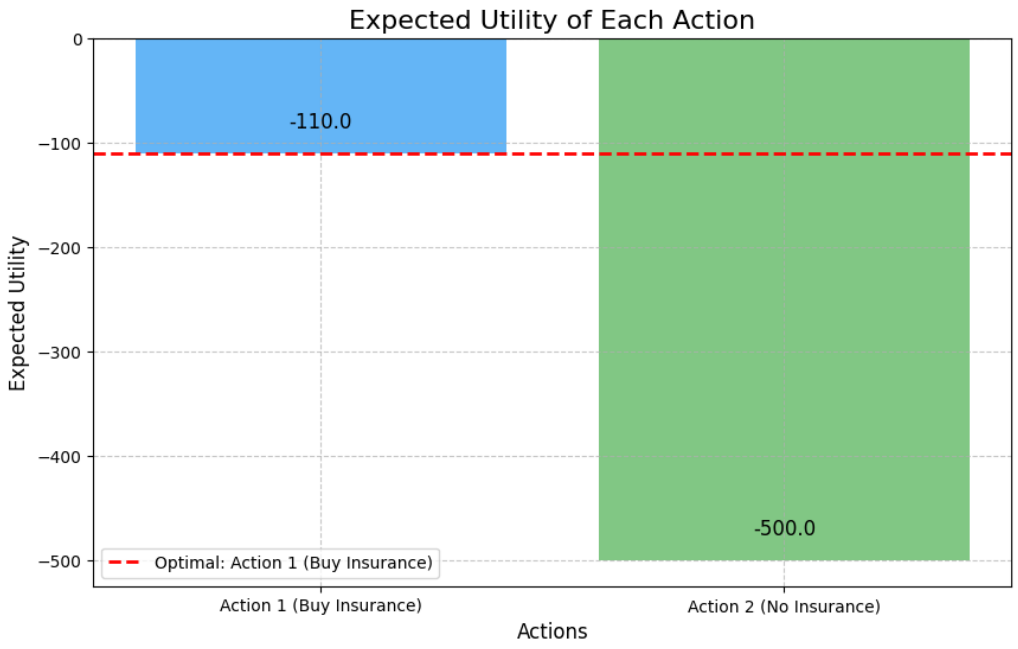
3. Tính hành động tối ưu

```
# 3. Identify Hành động tối ưu
optimal_action = max(expected_utilities, key=expected_utilities.get)
```

$$a^* = argmax_a EU(a|o)$$

Vì $EU(a_1) = -110 > EU(a_2) = -500$, hành động tối ưu là a_1.

4. Kết quả




```
Expected Utilities: {'Action 1 (Buy Insurance)': -110.0, 'Action 2 (No Insurance)': -500.0}
Optimal Action: Action 1 (Buy Insurance)
```

→ Quyết định **Mua bảo hiểm**.

2.1.5 Decision Networks (Mạng Quyết Định)

Mạng quyết định (Decision Networks), còn gọi là **đồ thị ảnh hưởng** (Influence Diagrams) được sử dụng để biểu diễn các bài toán ra quyết định phức tạp trong bối cảnh không chắc chắn. Đây là một **sự mở rộng của mạng Bayes**, kết hợp thêm các nút hành động và tiện ích nhằm tối ưu hóa quyết định.

- **Cấu trúc của Mạng Quyết định**:

Mạng Quyết định bao gồm 3 loại nút:

1. **Nút cơ hội (Chance Node)**: Mô hình hóa sự không chắc chắn trong bài toán. Các nút này thường được gắn với bảng xác suất có điều kiện (CPT - Conditional Probability Table).
    - Tương ứng với 1 biến ngẫu nhiên.
    - Được biểu diễn bằng hình tròn (circle).
    - Xác suất của biến dựa trên các biến cha của nó.
2. **Nút quyết định (Action Node)**: Đại diện cho các hành động hoặc quyết định mà người ra quyết định có thể chọn. Không có CPT vì nút quyết định không phải là biến ngẫu nhiên mà là các hành động có thể được chọn.
    - Tương ứng với 1 biến quyết định.
    - Được biểu diễn bằng hình vuông (square).
    - Không liên quan đến xác suất, vì hành động là thứ mà người dùng lựa chọn.
3. **Nút tiện ích (Utility Node)**: Biểu thị tiện ích hoặc giá trị của kết quả, giúp đánh giá và so sánh các đường dẫn quyết định khác nhau để tìm hành động tối ưu.
    - Tương ứng với 1 biến tiện ích.
    - Được biểu diễn bằng hình thoi (diamond).
    - Không thể có các nút con vì tiện ích không phụ thuộc trực tiếp vào các nút khác ngoài các biến cha của nó.

Cấu trúc của mạng quyết định thường được biểu diễn dưới dạng **đồ thị phi chu trình có hướng (DAG)**; kết hợp giữa các nút ngẫu nhiên, nút hành động và nút tiện ích**.** Trong mạng này, có 3 loại cạnh có hướng chính:

1. **Cạnh điều kiện (Conditional Edge)**: Cạnh này kết thúc ở nút cơ hội (Chance Node), thể hiện rằng sự không chắc chắn của biến tại nút này phụ thuộc vào các giá trị của các nút cha.
2. **Cạnh thông tin (Informational Edge)**: Cạnh này kết thúc ở nút quyết định (Action Node), thể hiện rằng quyết định tại nút này được đưa ra dựa trên thông tin từ các giá trị của nút cha. Loại cạnh này thường được biểu diễn bằng đường gạch đứt và có thể được lược bỏ trong sơ đồ để làm đơn giản hóa.
3. **Cạnh chức năng (Functional Edge)**: Cạnh này kết thúc ở nút tiện ích (Utility Node) và chỉ ra rằng giá trị tiện ích tại nút này được xác định bởi kết quả của

các nút cha. Loại cạnh này giúp đánh giá giá trị lợi ích hoặc chi phí của một kết quả.

- Tiện ích liên quan đến một hành động được tính bằng tổng giá trị của tất cả các nút tiện ích trong mạng.

❖ **Nguyên lý hoạt động của Mạng Quyết định:**

Để giải một bài toán trong mạng quyết định, chúng ta thực hiện các bước sau:

1. **Xây dựng đồ thị mạng quyết định**:
   - Xác định các nút ngẫu nhiên, quyết định và tiện ích.
   - Xác định các liên kết giữa chúng để biểu thị các mối quan hệ phụ thuộc hoặc nhân quả.
2. **Tính toán xác suất**: Tính toán xác suất có điều kiện của các biến ngẫu nhiên.
3. **Tính tiện ích kỳ vọng (Expected Utility – EU).**
4. **Chọn hành động tối ưu.**

Ví dụ minh họa:

Tình huống: **Một bác sĩ phải quyết định có nên thực hiện xét nghiệm y tế để chẩn đoán bệnh nhân hay không?**

1. **Nút cơ hội:**
   - Trạng thái sức khỏe của bệnh nhân:
     - s_1: Bệnh nhân khỏe mạnh.
     - s_2: Bệnh nhân mắc bệnh.
2. **Nút quyết định**:
     - a_1: Xét nghiệm.
     - a_2: Không xét nghiệm.
3. **Nút tiện ích:** Hàm tiện ích dựa trên chi phí xét nghiệm và kết quả sức khỏe của bệnh nhân.
     - Nếu s_1 và a_1 : U = 100 (Khỏe mạnh, không tốn chi phí)
     - Nếu s_1 và a_2 : U = 100 (Khỏe mạnh, không tốn chi phí)
     - Nếu s_2 và a_2 : U = -500 (Bệnh nặng, không được phát hiện)
     - Nếu s_2 và a_1 : U = 80 (Bệnh được phát hiện, giảm tác hại)

```python
# Xác suất trạng thái sức khỏe
probabilities = {
    "Health": {"Healthy": 0.7, "Sick": 0.3},  # P(s1) = 0.7, P(s2) = 0.3
}

# Nút Tiện ích
utilities = {
    "Healthy": {"No Test": 100, "Test": 100},  # Khỏe mạnh
    "Sick": {"No Test": -500, "Test": 80},     # Bệnh nặng
}

# Nút quyết định
actions = ["No Test", "Test"] #Không xét nghiệm; Xét nghiệm
```

Tính toán:

1. **Xác suất trạng thái:**
   - $P(s_1) = 0.7$
   - $P(s_2) = 0.3$
2. **Tiện ích kỳ vọng:**
   - Nếu xét nghiệm (a_1):

$$EU(a_1) = P(s_1).U(s_1, a_1) + P(s_2).U(s_2, a_1)$$
$$= 0.7 * 100 + 0.3 * 80 = 94$$

   - Nếu không xét nghiệm (a_2):

$$EU(a_2) = P(s_1).U(s_1, a_2) + P(s_2).U(s_2, a_2)$$
$$= 0.7 * 100 + 0.3 * (-500) = -80$$

3. **Chọn hành động tối ưu:**

Vì $EU(a_1) = 94 > EU(a_2) = -80$ → Hành động tối ưu là a_1.

```python
# 2. Calculate Expected Utility for Each Action
def compute_expected_utility(probabilities, utilities, action):
    expected_utility = 0
    for health, health_prob in probabilities["Health"].items():
        utility = utilities[health][action]
        expected_utility += health_prob * utility
    return expected_utility

# Tính lợi ích kỳ vọng (EU)
expected_utilities = {action: compute_expected_utility(probabilities, utilities, action) for action in actions}

# Hành động tối ưu
optimal_action = max(expected_utilities, key=expected_utilities.get)
```

➔ Quyết định nên thực hiện xét nghiệm.

2.1.6 Value of Information (Giá trị của thông tin)

Giá trị của thông tin (Value of Information - VOI) đo lường mức độ cải thiện trong quyết định khi có thêm thông tin mới. VOI cho thấy thông tin hữu ích đến mức nào khi giúp đưa ra quyết định tối ưu hơn, đặc biệt trong điều kiện không chắc chắn. Ví dụ, trong trường hợp điều trị bệnh, việc thực hiện thêm các xét nghiệm chẩn đoán có thể giúp giảm nguy cơ không điều trị một căn bệnh thực sự tồn tại.

Để tính giá trị của thông tin, chúng ta sử dụng **EU*(o)** để biểu thị tiện ích kỳ vọng của một hành động tối ưu, dựa trên quan sát o . Giá trị của thông tin về biến ***O'*** , dựa trên bằng chứng ban đầu ***o*** , được xác định như sau:

$$VOI(O'|o) = (\sum_{o'} P(o'|o)EU^*(o,o')) - EU^*(o)$$

Trong đó:

- O' : Biến thông tin mới
- o: Biến bằng chứng hiện tại.
- P(o'|o) : là xác xuất của biến O' dựa trên bằng chứng o .
- EU*(o,o'): Tiện ích kỳ vọng tối ưu khi biết thêm thông tin o′.
- VOI(O'|o) : Giá trị của Thông tin O′.

Ta có thể hiển rằng VOI được tính bằng hiệu số giữa $\boldsymbol{EU_{info}}$ (tiện ích kỳ vọng có thông tin) và $\boldsymbol{EU_{no\ info}}$ (tiện ích kỳ vọng không có thông tin).

$$VOI = EU_{info} - EU_{no\ info}$$

Giá trị của VOI không bao giờ âm . Tiện ích kỳ vọng chỉ có thể tăng nếu các quan sát bổ sung dẫn đến các quyết định tối ưu khác nhau. Nếu quan sát 1 biến mới O' không ảnh hưởng đến quyết định thì EU*(o,o') = EU*(o) với mọi O', khi đó giá trị thông tin bằng 0. Còn nếu VOI > 0 thì thông tin giúp cải thiện quyết định.

❖ **Ví dụ minh họa :**

Tình huống: Quyết định mang ô

Bạn quyết định có nên mang ô hay không, dựa trên dự báo thời tiết O′.

1. **Xác suất ban đầu**:

- P(Sunny)=0.7
- P(Rainy)=0.3

2. **Tiện ích của hành động**:

- **Mang ô (A)**:

  Sunny:U=8

  Rainy: U=7

- **Không mang ô (B)**:

  Sunny: U=10

  Rainy: U=−5

3. **Quan sát thêm thông tin (Dự báo thời tiết)**:

- **Nếu O′=Forecast:**

  P(Sunny|O′)=0.9

  P(Rainy|O′)=0.1

- **Nếu O′=Forecast:**
  P(Sunny|O′)=0.2
  P(Rainy|O′)=0.8

```
#1. Xác định xác xuất và tiện ích
states = ["Sunny", "Rainy"] # Trạng thái hiện tại : nắng, mưa
probabilities = {"Sunny": 0.7, "Rainy": 0.3}  #Xác suất thời tiết nắng, mưa trước khi có bất kỳ thông tin nào.

utilities = {
    "Sunny": {"Umbrella": 8, "No Umbrella": 10},
    "Rainy": {"Umbrella": 7, "No Umbrella": -5},
}  #Tiện ích của mỗi hành động (Umbrella, No Umbrella) trong từng trạng thái (Sunny, Rainy)

forecast_probabilities = {
    "Sunny": {"Forecast: Sunny": 0.9, "Forecast: Rainy": 0.1},
    "Rainy": {"Forecast: Sunny": 0.2, "Forecast: Rainy": 0.8},
} # Xác suất điều kiện cho dự báo
```

Tiếp theo ta thực hiện tính EU_{noinfo} với :

$$EU_{no\ info} = max\{P(Sunny) * U(Sunny \mid A) + P(Rainy) * U(Rainy \mid A), P(Sunny) * U(Sunny \mid B) + P(Rainy) * U(Rainy \mid B)\}$$

```python
#2. Tính toán với EU_no info (Tiện ích kỳ vọng khi chưa có thông tin)
def eu_without_info(probabilities, utilities):
    actions = ["Umbrella", "No Umbrella"] # danh sách các hành động có thể thực hiện
    expected_utilities = {} # lưu trữ tiện ích kỳ vọng của từng hành động
    for action in actions:  # Duyệt qua từng hành động
        eu = sum(probabilities[state] * utilities[state][action] for state in states) # tính tiện ích của mỗi kì vọng
        expected_utilities[action] = eu # lưu tiện ích kỳ vọng của từng hành động vào expected_utilities
    optimal_action = max(expected_utilities, key=expected_utilities.get) # tìm hành động tối ưu nhất
    return max(expected_utilities.values()), optimal_action  # trả về kỳ vọng cao nhất và hành động tối ưu nhất

eu_no_info, optimal_action_no_info = eu_without_info(probabilities, utilities) # chạy hàm
```

Với mang ô (A):

EU(Umbrella)= 0.7*8+0.3*7=5.6+2.1=7.7

Với không mang ô (B):

EU(No Umbrella)=0.7*10+0.3*(−5)=7−1.5=5.5.

Vì thế ta chọn A (Mang ô) : EU_{noinfo}=7.7

Tiếp theo ta thực hiện tính EU_{info} :

Ta tính tổng xác Dự Báo (P(o')) :

- **Tổng xác suất cho "Forecast: Sunny" :**

$$P(Forecast: Sunny) = P(Sunny) * P(Forecast: Sunny \mid Sunny) + P(Rainy) * P(Forecast: Sunny \mid Rainy)$$

Ta tính được :

P(Forecast: Sunny)=0.7*0.9+0.3*0.2=0.63+0.06=0.69

- **Tổng xác suất cho "Forecast: Rainy":**

$$P(Forecast: Rainy) = P(Sunny) * P(Forecast: Rainy \mid Sunny) + P(Rainy) * P(Forecast: Rainy \mid Rainy)$$

Ta tính được :

P(Forecast: Rainy)=0.7*0.1+0.3*0.8=0.07+0.24=0.31

Ta thực hiện tính xác suất trạng thái bằng cách sử dụng định lý Bayles :

- **Với "Forecast: Sunny"**

$$P(Sunny \mid Forecast{:}\ Sunny) = \frac{P(Sunny) * P(Forecast{:}\ Sunny \mid Sunny)}{P(Forecast{:}\ Sunny)}$$

P(Sunny| Forecast: Sunny)=(0.7*0.9)/0.69 ≈ 0.913

$$P(Rainy \mid Forecast{:}\ Sunny) = \frac{P(Rainy) * P(Forecast{:}\ Sunny \mid Rainy)}{P(Forecast{:}\ Sunny)}$$

P(Rainy| Forecast: Sunny)=(0.3*0.2)/0.69 ≈ 0.087

- **Với "Forecast: Rainy"**

$$P(Sunny \mid Forecast{:}\ Rainy) = \frac{P(Sunny) * P(Forecast{:}\ Rainy \mid Sunny)}{P(Forecast{:}\ Rainy)}$$

P(Sunny| Forecast: Rainy)= (0.7*0.1)/0.31 ≈0.226

$$P(Rainy \mid Forecast{:}\ Rainy) = \frac{P(Rainy) * P(Forecast{:}\ Rainy \mid Rainy)}{P(Forecast{:}\ Rainy)}$$

P(Rainy| Forecast: Rainy)= (0.3*0.8)/0.31 ≈ 0.774

Tiếp theo ta tính tiện ích cho mỗi hành động :

- **Với "Forecast: Sunny"**
  - **Tiện ích cho "Umbrella":**

EU(Umbrella | Forecast: Sunny) = P(Sunny | Forecast: Sunny) * U(Sunny, Umbrella) + P(Rainy | Forecast: Sunny) * U(Rainy, Umbrella)

EU(Umbrella|Forecast: Sunny)=0.913*8+0.087*7=7.913

  - **Tiện ích cho "No Umbrella":**

EU(No Umbrella|Forecast: Sunny)=0.913*10+0.087*(−5) =8.695

⇨ Ta chọn hành động tối ưu là **"No Umbrella"**

- **Với "Forecast: Rainy"**
  - **Tiện ích cho "Umbrella":**

    EU(Umbrella|Forecast: Rainy)=0.226*8+0.774*7=7.226

  - **Tiện ích cho "No Umbrella":**

    EU(No Umbrella|Forecast: Rainy)=0.226*10+0.774*(−5)= −1.61

⇨ Ta chọn hành động tối ưu là **"Umbrella"**

Ta tính tổng tiện ích kỳ vọng (EU_{info}):

$$EU_{info} = P(Forecast: Sunny) * max\big(EU(Umbrella|Forecast: Sunny), EU(No\ Umbrella \mid Forecast: Sunny)\big) + P(Forecast: Rainy) * max(EU(Umbrella \mid Forecast: Rainy), EU(No\ Umbrella \mid Forecast: Rainy))$$

  - Với "Forecast: Sunny":
    P(Forecast: Sunny)*max(7.913,8.695)=0.6*·8.695=5.99955
  - Với "Forecast: Rainy":
    P(Forecast: Rainy)*max(7.226,−1.61)=0.31*7.226=2.24006

⇨ Tổng tiện ích kỳ vọng:
$EU_{info} = 5.99955+2.24006= 8.23961 \approx 8.24$

```python
#3. Tính toán với EU_info (Tiện ích kỳ vọng khi có thông tin)
def eu_with_info(probabilities, forecast_probabilities, utilities):
    eu_with_info = 0  # biến lưu tiện ích kỳ vọng
    for forecast in ["Forecast: Sunny", "Forecast: Rainy"]:
        posterior_prob = {}
        total_prob = sum(
 probabilities[state] * forecast_probabilities[state][forecast] for state in states
        ) # Tính tổng xác suất dự báo dựa trên xác suất ban đầu và xác suất điều kiện
        for state in states:
            posterior_prob[state] = (
                probabilities[state] * forecast_probabilities[state][forecast]
            ) / total_prob #Sử dụng Định lý Bayes để cập nhật xác suất trạng thái

        # Tính tiện ích kỳ vọng cho mỗi hành động
        actions = ["Umbrella", "No Umbrella"]
        expected_utilities = {}
        for action in actions:
            eu = sum(posterior_prob[state] * utilities[state][action] for state in states)
            expected_utilities[action] = eu
        eu_with_info += total_prob * max(expected_utilities.values()) #chọn hành động tối ưu và cộng tiện ích kỳ vọng
    return eu_with_info    #Tiện ích kỳ vọng tối ưu khi có thêm thông tin

eu_info = eu_with_info(probabilities, forecast_probabilities, utilities) # chạy hàm
```

Cuối cùng ta tính được VOI bằng :

$$VOI = EU_{info} - EU_{no\ info} = 8.24 - 7.7 \approx 0.54$$

```
#4. Tính VOI (Giá trị thông tin)
voi = eu_info - eu_no_info

#5. Kết quả
print(f"EU_no_info(Tiện ích kỳ vọng khi chưa có thông tin): {eu_no_info} (Lựa chọn tối ưu nhất: {optimal_action_no_info})")
print(f"EU_info(Tiện ích kỳ vọng khi có thông tin): {eu_info}")
print(f"Giá trị thông tin (VOI): {voi}")
```

```
EU_no_info(Tiện ích kỳ vọng khi chưa có thông tin): 7.699999999999999 (Lựa chọn tối ưu nhất: Umbrella)
EU_info(Tiện ích kỳ vọng khi có thông tin): 8.239999999999998
Giá trị thông tin (VOI): 0.5399999999999991
```

2.1.7 Irrationality (Bất hợp lý trong quyết định của con người)

Bất hợp lý trong quyết định của con người là ra quyết định thảo luận về những tình huống mà con người đưa ra quyết định không theo các nguyên tắc tối ưu hóa theo lý thuyết, chẳng hạn như nguyên tắc tối đa hóa tiện ích kỳ vọng. Đây là một khía cạnh quan trọng trong việc hiểu hành vi ra quyết định của con người, đặc biệt trong các tình huống có rủi ro hoặc không chắc chắn.

Bất hợp lý trong quyết định

1. **Không tối ưu hóa tiện ích kỳ vọng**:
    - Con người không luôn chọn hành động tối ưu dù có đầy đủ thông tin.
    - Ví dụ: Chọn một hành động dựa trên cảm xúc hoặc thiên kiến thay vì tính toán tiện ích.
2. **Ảnh hưởng từ yếu tố tâm lý và nhận thức**:
    - **Hiệu ứng khung hình (Framing Effect):**
        - Quyết định bị ảnh hưởng bởi cách thông tin được trình bày.
        - Ví dụ: Một phương pháp điều trị được mô tả là "90% thành công" có thể được chấp nhận hơn "10% thất bại".
    - **Thiên kiến xác nhận (Confirmation Bias):**
        - Con người có xu hướng tìm kiếm thông tin củng cố ý kiến sẵn có thay vì xem xét tất cả bằng chứng.
3. **Không nhất quán trong sở thích**:

- Vi phạm các tiên đề như **tính bắc cầu** hoặc **tính liên tục**.
- Ví dụ: Thích A>B, B>C, nhưng lại chọn C>A.

Nguyên nhân dẫn đến bất hợp lý

- **Cảm xúc**: Sợ hãi, lo lắng hoặc phấn khích có thể làm sai lệch quyết định.
- **Kiến thức không đầy đủ**: Không biết tất cả các xác suất hoặc tiện ích.
- **Quá tải thông tin**: Khi phải xử lý quá nhiều thông tin, con người có thể chọn hành động đơn giản thay vì tối ưu.

Ví dụ minh họa:

Thí nghiệm về sự lựa chọn

- **Trường hợp 1 :**
  - A: Chắc chắn mất 75 mạng người.
  - B: 80% mất 100 mạng người và 20% không mất ai.

  =>Kết quả: Đa số chọn B, dù lý trí sẽ chọn A vì tổng mất mát kỳ vọng thấp hơn.
- **Trường hợp 2 :**
  - C: 10% mất 75 mạng người.
  - D: 8% mất 100 mạng người.

  =>Kết quả: Đa số có xu hướng chọn C, mặc dù theo logic thì D tốt hơn.

```
# Các tùy chọn trong 2 trường hợp
options = {
    'A': {'probability': 1.0, 'loss': 75},  # Chắc chắn mất 75 người
    'B': {'probability': 0.8, 'loss': 100}, # 80% mất 100 người, 20% không mất
    'C': {'probability': 0.1, 'loss': 75},  # 10% mất 75 người
    'D': {'probability': 0.08, 'loss': 100} # 8% mất 100 người
}
```

***Giá trị kỳ vọng (Expected Value)**:*

- **Trường hợp 1:**

- Phương án A: EV=75 (chắc chắn)
- Phương án B: EV=0.8×100+0.2×0=80

- **Trường hợp 2:**
  - Phương án C: EV=0.1×75=7.5
  - Phương án D: EV=0.08×100=8

```
# Ta định nghĩa tiện ích của mất mát (mất càng lớn, tiện ích càng nhỏ)
def utility(loss):
    return -loss  # Hàm tuyến tính cho đơn giản
```

```
# Tính giá trị kỳ vọng
def calculate_expected_value(option):
    return option['probability'] *(utility(option['loss']))
```

➢ **Về các lựa chọn :**

Trường hợp 1:

- **Lý trí:** Khi so sánh giá trị kỳ vọng, A là lựa chọn tốt hơn vì tổn thất là 75 (ít hơn 80 của B).
- **Thực tế:** Đa số mọi người chọn **B** vì họ muốn tránh **sự chắc chắn về tổn thất** (hiệu ứng chắc chắn). Lựa chọn này **phi lý trí** vì nó không tối ưu về mặt giá trị kỳ vọng.

Trường hợp 2:

- **Lý trí:** C là lựa chọn tốt hơn vì tổn thất kỳ vọng (7.5) thấp hơn D (8).
- **Thực tế:** Đa số mọi người chọn **C**, điều này hợp lý với lý thuyết tối ưu.

```python
# Trường hợp 1 và 2
print("Giá trị kỳ vọng cho trường hợp 1:")
for option in ['A', 'B']:
    ev = calculate_expected_value(options[option])
    print(f"Lựa chọn {option}: EV = {ev:.2f}")

print("\nGiá trị kỳ vọng cho trường hợp 2:")
for option in ['C', 'D']:
    ev = calculate_expected_value(options[option])
    print(f"Lựa chọn {option}: EV = {ev:.2f}")

# Mô phỏng lựa chọn của người (phi lý trí)
choices_human = {
    'Trường hợp 1': 'B',  # mọi người người chọn B vì họ muốn tránh sự chắc chắn về tổn thất
    'Trường hợp 2': 'C'   # mọi người người chọn C vì điều này hợp lý với lý thuyết tối ưu.
}

print("\nLựa chọn:")
for case, choice in choices_human.items():
    print(f"{case}: Người chọn {choice}")
```

```
Giá trị kỳ vọng cho trường hợp 1:
Lựa chọn A: EV = -75.00
Lựa chọn B: EV = -80.00

Giá trị kỳ vọng cho trường hợp 2:
Lựa chọn C: EV = -7.50
Lựa chọn D: EV = -8.00

Lựa chọn:
Trường hợp 1: Người chọn B
Trường hợp 2: Người chọn C
```

- Ví dụ này minh họa rằng con người không luôn hành động theo cách tối ưu hoặc lý trí khi đưa ra quyết định. Hành vi này có thể dự đoán được nhờ các nguyên tắc tâm lý học như hiệu ứng chắc chắn và hiệu ứng khung.

CHƯƠNG 3: TỔNG KẾT

3.1 Kết luận:

❖ Ưu điểm

Dễ hiểu và áp dụng trong thực tế

- Phương pháp Simple Decision được xây dựng dựa trên những nguyên tắc đơn giản, dễ hiểu và phù hợp với cả những người không có kiến thức chuyên môn sâu. Điều này giúp nó trở nên hữu ích trong nhiều tình huống thường ngày và dễ dàng được áp dụng trong thực tế.

Ứng dụng rộng rãi

- Với tính linh hoạt, Simple Decision có thể được sử dụng trong nhiều lĩnh vực khác nhau, bao gồm kinh doanh, tài chính, y tế và thậm chí là trí tuệ nhân tạo, nơi việc đưa ra quyết định nhanh chóng và hiệu quả là rất quan trọng.

Tính linh hoạt

- Mặc dù bắt đầu từ những nguyên tắc đơn giản, phương pháp này có thể được mở rộng và điều chỉnh để xử lý các vấn đề phức tạp hơn. Điều này giúp nó thích nghi tốt với các tình huống đa dạng.

Nguyên tắc tối đa hóa tiện ích kỳ vọng

- Một trong những điểm mạnh của Simple Decision là khả năng áp dụng nguyên tắc tối đa hóa tiện ích kỳ vọng, cho phép người dùng lựa chọn hành động mang lại giá trị cao nhất trong các điều kiện không chắc chắn.

Tối ưu hóa hiệu quả

- Các công cụ như "Giá trị của Thông tin" (Value of Information - VOI) giúp định lượng lợi ích của việc bổ sung thông tin mới. Điều này hỗ trợ người dùng ra quyết định một cách hiệu quả hơn, đồng thời tối ưu hóa các nguồn lực.

Hỗ trợ ra quyết định có cơ sở dữ liệu

- Simple Decision cho phép tích hợp dữ liệu và các phân tích định lượng, giúp đưa ra các quyết định có cơ sở, minh bạch và đáng tin cậy hơn.

❖ **Nhược điểm**

Đơn giản hóa thực tế

- Simple Decision thường giả định rằng các yếu tố như xác suất, tiện ích và các trạng thái có thể được xác định rõ ràng và chính xác. Tuy nhiên, trong thực tế, các yếu tố này thường không rõ ràng hoặc khó đo lường, dẫn đến sai lệch trong quá trình ra quyết định.

Khó khăn trong hệ thống phức tạp

- Khi áp dụng vào các hệ thống phức tạp, việc xác định tiện ích, xác suất hoặc mối quan hệ giữa các yếu tố trở nên rất khó khăn. Điều này làm giảm hiệu quả và độ chính xác của phương pháp.

Không phù hợp với các tình huống liên tục hoặc dài hạn

- Simple Decision không được thiết kế để xử lý các tình huống liên quan đến các hành động liên tục hoặc có sự phụ thuộc vào thời gian, chẳng hạn như việc lập kế hoạch dài hạn hoặc quản lý các dự án phức tạp.

Không tính đến thời gian hoặc động lực học

- Một hạn chế khác của Simple Decision là không xem xét đến các yếu tố thời gian hoặc sự thay đổi động lực học của hệ thống, dẫn đến các quyết định thiếu sự phù hợp trong các bối cảnh thay đổi liên tục.

Chi phí tính toán cao khi mở rộng

- Khi số lượng trạng thái, hành động hoặc thông tin cần xử lý tăng lên, chi phí tính toán tiện ích kỳ vọng có thể tăng đáng kể. Điều này gây khó khăn trong việc áp dụng Simple Decision vào các hệ thống lớn hoặc phức tạp.

3.2 Hướng phát triển

Ứng dụng trong các hệ thống phức tạp, không chắc chắn và đa mục tiêu

- **Mở rộng phạm vi áp dụng:** Phương pháp Simple Decision có thể được nâng cấp để xử lý các hệ thống phức tạp với nhiều yếu tố không chắc chắn và mâu thuẫn giữa các mục tiêu. Điều này đòi hỏi phải tích hợp thêm các mô hình đa mục tiêu (Multi-Objective Decision Making) và kỹ thuật phân tích rủi ro.
- **Tăng cường tính chính xác:** Sử dụng các công cụ phân tích tiên tiến như Monte Carlo Simulation, Decision Trees, hoặc Bayesian Networks để đánh giá các kịch bản và xác suất một cách chi tiết hơn.

Tích hợp công nghệ tiên tiến

- **Trí tuệ nhân tạo (AI) và học máy (Machine Learning):**
  - Sử dụng AI và học máy để phân tích dữ liệu lớn (Big Data) và đưa ra các dự đoán chính xác hơn trong điều kiện không chắc chắn.

    - Áp dụng các thuật toán học tăng cường (Reinforcement Learning) để hỗ trợ ra quyết định liên tục trong thời gian thực.
- **Blockchain:**
    - Blockchain giúp nâng cao tính minh bạch và bảo mật cho các quyết định, đặc biệt trong các lĩnh vực như tài chính, chuỗi cung ứng, và y tế.
    - Ghi lại và xác minh các thông tin liên quan đến quá trình ra quyết định, giúp đảm bảo tính toàn vẹn dữ liệu.
- **Tối ưu hóa thời gian thực (Real-Time Optimization):**
    - Phát triển các hệ thống ra quyết định dựa trên tối ưu hóa thời gian thực để đưa ra giải pháp nhanh chóng, phù hợp với các tình huống thay đổi liên tục như quản lý giao thông, logistics, hoặc hệ thống sản xuất.

Ứng dụng trong các lĩnh vực cụ thể

- **Tài chính:**
    - Hỗ trợ nhà đầu tư trong việc quản lý rủi ro, phân bổ danh mục đầu tư, hoặc lựa chọn các chiến lược giao dịch dựa trên dự đoán thị trường và tiện ích kỳ vọng.
- **Quản lý nguồn lực:**
    - Ra quyết định tối ưu trong việc phân bổ nguồn lực như lao động, vật liệu, và ngân sách nhằm tối đa hóa hiệu quả và giảm thiểu lãng phí.
- **Logistics và chuỗi cung ứng:**
    - Tối ưu hóa các tuyến đường vận chuyển, dự báo nhu cầu, và quản lý hàng tồn kho thông qua hệ thống hỗ trợ ra quyết định thông minh.
- **Y tế:**
    - Hỗ trợ bác sĩ trong việc chẩn đoán và lựa chọn phương pháp điều trị dựa trên dữ liệu bệnh nhân, kết hợp với các yếu tố không chắc chắn như phản ứng thuốc.
    - Cải thiện hiệu quả trong việc quản lý nguồn lực y tế, như phân bổ giường bệnh, nhân sự, và trang thiết bị.
- **Chính sách công:**

- Đánh giá các tác động dài hạn của các chính sách và lựa chọn phương án tối ưu để giải quyết các vấn đề xã hội như môi trường, giáo dục, và y tế cộng đồng.

Tăng cường khả năng thích ứng và mở rộng

- Hệ thống hỗ trợ đa cấp độ: Phát triển các hệ thống có khả năng ra quyết định ở nhiều cấp độ khác nhau, từ cá nhân đến tổ chức, giúp mở rộng phạm vi áp dụng trong các tình huống thực tế.
- Tích hợp dữ liệu đa nguồn: Kết hợp dữ liệu từ nhiều nguồn khác nhau, như dữ liệu thời gian thực, dữ liệu lịch sử, và dữ liệu cảm biến, để tăng cường chất lượng thông tin đầu vào cho các quyết định.
- Ứng dụng trong giáo dục và đào tạo: Phát triển các công cụ và bài giảng giúp sinh viên, nhân viên hoặc nhà quản lý hiểu và áp dụng Simple Decision vào các lĩnh vực chuyên môn.

Nghiên cứu và phát triển thuật toán nâng cao

- Cải tiến thuật toán ra quyết định: Tăng cường khả năng xử lý với các trạng thái, hành động hoặc thông tin phức tạp hơn.
- Tích hợp động lực học và yếu tố thời gian: Phát triển các mô hình có khả năng xem xét các thay đổi động lực học và tác động của thời gian để đưa ra quyết định chính xác hơn.

CHƯƠNG 4: TÀI LIỆU THAM KHẢO

[1] https://plato.stanford.edu/entries/rationality-normative-utility/

[2] "Decision-Making: A Theoretical Review"

[3] "Understanding the Dynamics of Decision-Making and Choice"

[4] "Decision-Making Styles and Their Associations with Competencies"

[5] "Simply Rational: Decision Making in the Real World"

[6] "Cognitive Biases and Decision-Making Strategies in Times of Change"