

Nome: Bárbara Braga Gualberto Correa e Julia Mello Lopes Gonçalves

**1. Tanto a medida de desempenho quanto a função de utilidade medem o desempenho de um agente. Explique a diferença entre as duas medidas.**

Medida de Desempenho: É o critério que define o grau de sucesso de um agente na realização de uma dada tarefa, em suma, sendo mais focada e objetiva. A escolha errada da medida de desempenho pode acarretar em um comportamento indesejado. Não considera as preferências individuais do agente, apenas se a tarefa foi corretamente concluída.

Função de Utilidade: Atribui um número para expressar a desejabilidade de um estado para o agente, ou seja, mapeia um estado (ou sequência de estados) em um número real que descreve o grau de satisfação associado ao estado. As utilidades são combinadas com probabilidades dos estados para tomada de decisão, além de considerar se a tarefa foi concluída considera também o quão desejável foi para o agente.

**2. Vamos examinar a racionalidade de várias funções do agente aspirador de pó. Mostre que a função do agente aspirador de pó simples descrito na Figura 1 é realmente racional, conforme as suposições listadas abaixo.**

- Suposição 1: medida de desempenho - um ponto para cada quadrado limpo em um período de tempo no total de 1000 períodos de tempo.
- Suposição 2: Geografia do ambiente é conhecida a priori, mas não se conhece a distribuição da sujeira e posição inicial do agente e desconhecida.
- Suposição 3: ações - esquerda, direita, aspirar e NoOp (não fazer nada).
- Suposição 4: sensores - agente percebe corretamente local e sujeira.
- Estratégia: função do agente - limpar se o quadrado tem sujeira e ir para o outro quadrado, caso contrário.

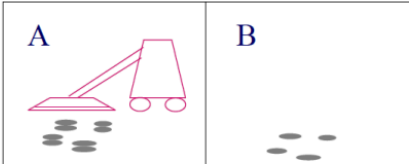
	
Sequência de percepts	Ação
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Clean], [A, Clean]	Righ
[A, Clean], [A, Dirty]	Suck
:	:
[A, Clean], [A, Clean], [A, Clean]	Righ
[A, Clean], [A, Clean], [A, Dirty]	Such
:	:

Figure 1. Agente aspirador de pó.

O agente aspirador de pó demonstra racionalidade, maximizando a medida de desempenho específica definida pela quantidade de quadrados limpos em 1000 períodos de tempo (Suposição 1). Além disso, mantém uma história perceptiva, registrando a sequência de percepções e ações tomadas (Figura 1), o que permite ao agente conhecer o ambiente e planejar sua rota, mesmo sem conhecer a distribuição inicial de sujeira e sua posição inicial (Suposição 2). Com uma variedade de ações disponíveis, incluindo a capacidade de aspirar e perceber corretamente o ambiente (Suposição 3 e 4), o agente assegura que tome as ações necessárias para limpar os quadrados sujos. Assim, o agente aspirador de pó atende plenamente aos critérios de racionalidade.

**3. Desenvolva uma descrição PEAS do ambiente de tarefa de um robô jogador de futebol.**

Tipo de Agente	Performance	Environment	Actuators	Sensors
Robô Jogador de Futebol	Número de gols a favor, número de defesas (caso seja goleiro), passes corretos	Campo de Futebol, técnico, colegas de equipe, outros jogadores, árbitro	Bola, tênis, uniforme, luvas (caso seja goleiro)	câmera, sensor de distância, sensor de toque, GPS, microfone, acelerômetro, giroscópio

**4. Considere uma versão modificada do ambiente do aspirador de pó, onde o agente é agora penalizado com um ponto para cada movimento que faz. Um agente reativo simples pode ser perfeitamente racional para esse ambiente? Explique.**

Se alterarmos a medida de desempenho do aspirador de pó para penalizar cada movimento, isso redefinirá o método de desempenho. Nessa nova versão, o agente considerará que a melhor estratégia é minimizar os movimentos, priorizando a imobilidade. Portanto, essa modificação na medida de desempenho pode comprometer a eficácia do agente em cumprir sua função primária de limpeza.

**5. Implemente um agente reativo simples para o ambiente do aspirador. Execute o ambiente com este agente para todas as configurações iniciais de sujeira possíveis e locais dos agentes. Registre a pontuação de desempenho para cada configuração e a pontuação média geral.**

**function** REFLEX-VACUUM-AGENT(*[location,status]*) **returns** an action  
**if** *status* = Dirty **then return** Suck  
**else if** *location* = A **then return** Right  
**else if** *location* = B **then return** Left

Figure 2. Pseudocódigo do agente reativo.

Figure 2. Pseudocódigo do agente reativo.

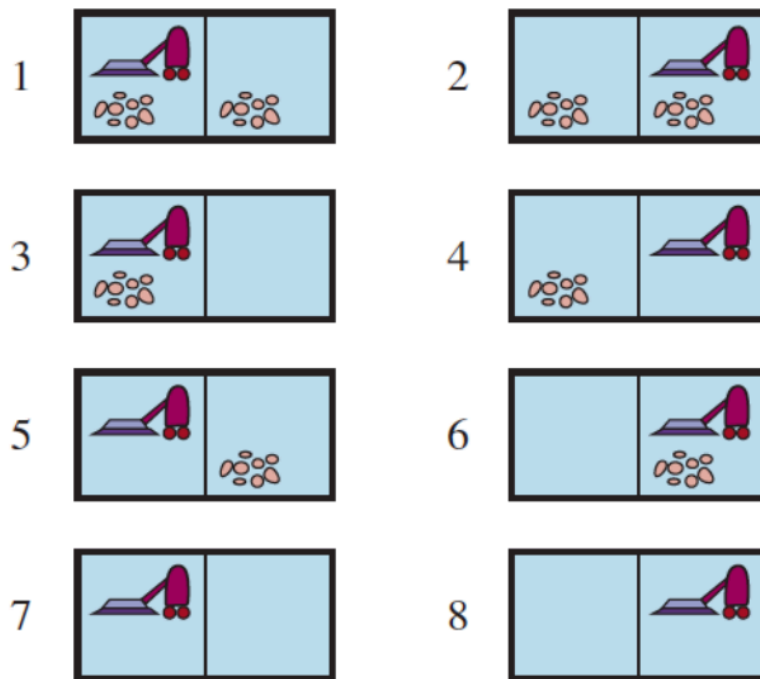


Figure 3. Possíveis estados do agente aspirador de pó.

```
main.py
import json

# função do agente aspirador
def reactive_vacuum_agent(location, status, performance):
    #se a sala estiver suja, muda o status para limpo
    if status == "Dirty":
        performance += 1
        status = "Clean"
        # print("Suck")
    #se o aspirador estiver na localização A, muda para a localização B
    if location == "A":
        location = "B"
        # print("Right")
        return location, status, performance
    #de o aspirador estiver na localização B, muda para a localização A
    elif location == "B":
```

```

        location = "A"
        # print("Left")
        return location, status, performance

#abre o arquivo .json com os estados iniciais (1 a 8)
with open("initial_states.json", "r") as f:
    initial_states = json.load(f)

total_performance = 0 #variável que armazena todas os métodos de
desempenho
num_configurations = len(initial_states) #obtem o tamanho do
dicionário initial_states

# itera sobre as localizações e os valores dos estados iniciais
for state_key, state_values in initial_states.items():
    print("\nState:", state_key)
    performance = 0 #inicializa o desempenho para o estado atual para
    calcular a performance de cada estado
    #itera sobre a lista de localizações e status do estado atual
    for location_status in state_values:
        for location, status in location_status.items():
            #atualiza o local, status e desempenho usando a função do
            agente do aspirador
            location, status, performance =
            reactive_vacuum_agent(location, status, performance)
            total_performance += performance #atualiza o método de
            desempenho total
            print("Performance:", performance) #printa o desempenho de
            cada estado

#calcula e printa a média geral do desempenho
print("Media geral:", total_performance / num_configurations)

```

#### initial\_states.json

```

{
    "state1": [
        {"A": "Dirty"},
        {"B": "Dirty"}
    ],
    "state2": [
        {"A": "Dirty"},
        {"B": "Dirty"}
    ]
}

```

```
],
  "state3": [
    {"A": "Dirty"},
    {"B": "Clean"}
  ],
  "state4": [
    {"A": "Dirty"},
    {"B": "Clean"}
  ],
  "state5": [
    {"A": "Clean"},
    {"B": "Dirty"}
  ],
  "state6": [
    {"A": "Clean"},
    {"B": "Dirty"}
  ],
  "state7": [
    {"A": "Clean"},
    {"B": "Clean"}
  ],
  "state8": [
    {"A": "Clean"},
    {"B": "Clean"}
  ]
}
```