s

UNIVERSIDADE FEDERAL DO MARANHÃO

BACHARELADO INTERDISCIPLINAR EM CIÊNCIA E TECNOLOGIA

INTELIGÊNCIA ARTIFICIAL

ARTHUR FELLIPE LIMA REIS

JUAN PABLO

PEDRO HENRIQUE

JOÃO PEDRO

São Luís

2024

SUMÁRIO

1. INTRODUÇÃO
2. DESENVOLVIMENTO
3. RESULTADOS
4. DISCUSSÕES
5. CONCLUSÃO

INTRODUÇÃO

A inteligência artificial (I.A.) se divide em várias abordagens e classificações, e uma das mais interessantes envolve a categorização de agentes, especialmente os agentes reativos. Esses agentes respondem a estímulos ou condições no ambiente de forma imediata, sem necessariamente possuir uma memória de estados anteriores ou um raciocínio complexo. A programação de agentes reativos oferece uma oportunidade para estudar comportamentos simples e eficazes em ambientes dinâmicos e interativos.

Neste projeto, utilizaremos o NetLogo, uma plataforma de modelagem e simulação de sistemas complexos, para realizar um estudo prático sobre agentes reativos. A escolha do NetLogo se justifica pela sua interface amigável e pela facilidade de modelagem de comportamentos de agentes em um ambiente visual. Nosso objetivo é programar agentes reativos, permitindo que eles façam escolhas baseadas nas condições do ambiente, como o movimento em direção a um alvo ou a interação com outros agentes.

No contexto teórico, destacamos que:

1. **Características principais dos agentes reativos**:
    - Ausência de memória ou aprendizado.
    - Decisões baseadas no estado atual do ambiente.
    - Dependência de um conjunto fixo de regras ou condições.
2. **Relevância prática**:
    - Agentes reativos são ideais para sistemas que requerem alto desempenho com baixo custo computacional.
    - Exemplos incluem robôs que seguem linhas, sistemas de resposta rápida e agentes em jogos simples.

O principal objetivo deste trabalho é explorar a implementação e o comportamento de **agentes reativos** no ambiente de modelagem e simulação do **NetLogo**, com o intuito de compreender suas características e limitações. Para isso, serão realizados estudos práticos que envolvem a programação de agentes capazes de tomar decisões baseadas em regras simples e interagir com o ambiente de forma dinâmica.

Mais especificamente, buscamos:

1. **Compreender o conceito teórico de agentes reativos** e como eles se diferenciam de outros tipos de agentes em I.A.
2. **Implementar agentes reativos no NetLogo**, programando regras e comportamentos que permitam sua interação com o ambiente simulado.
3. **Analisar as escolhas e comportamentos desses agentes**, observando como suas decisões impactam o sistema e as dinâmicas do ambiente.
4. **Fomentar a aprendizagem prática de conceitos de I.A.**, utilizando o NetLogo como ferramenta para simplificar a modelagem de agentes e tornar o estudo mais acessível.

Ao final, espera-se não apenas consolidar o entendimento sobre agentes reativos, mas também fornecer uma base para aplicações práticas e estudos mais avançados, tanto no NetLogo quanto em outras plataformas.

Netlogo

NetLogo é uma linguagem de programação e um ambiente de simulação especializado em modelagem baseada em agentes. Ele é amplamente utilizado em áreas como ciências sociais,

biologia, economia, física e ensino, para explorar sistemas complexos e dinâmicas emergentes.

Características do NetLogo

1. Modelagem Baseada em Agentes:
    - Cada agente é um indivíduo com comportamento programável.
    - Existem três tipos principais de agentes:
        - Turtles (tartarugas): agentes móveis que representam entidades individuais, como pessoas, animais ou partículas.
        - Patches: células fixas em uma grade que podem conter informações sobre o ambiente, como altitude ou temperatura.
        - Observer: um agente "global" que controla a execução da simulação.
2. Interface Visual:
    - Possui uma interface gráfica onde os agentes e o ambiente podem ser visualizados em tempo real, (que permite adaptarmos botões com comandos do código).
    - Oferece ferramentas para criar gráficos, botões, controles deslizantes e campos de entrada para manipular parâmetros da simulação.
3. Foco Educacional:
    - Projetado para facilitar o aprendizado de conceitos complexos de sistemas através de experimentação.
    - Muito utilizado em aulas e laboratórios para ensinar dinâmica de sistemas e programação.
4. Biblioteca de Modelos Prontos:
    - O NetLogo inclui uma biblioteca extensa de modelos pré-desenvolvidos cobrindo temas como ecossistemas, mercados econômicos e redes sociais.
5. Simulação Iterativa e Explorativa:
    - Permite ajustar parâmetros, executar simulações repetidas e observar como pequenas mudanças podem levar a grandes efeitos no sistema.

GITHUB

O GitHub é uma plataforma de hospedagem de código que usa o sistema de controle de versão Git. Ele permite que desenvolvedores colaborem em projetos, gerenciem versões do código e integrem-se a fluxos de trabalho modernos de desenvolvimento.

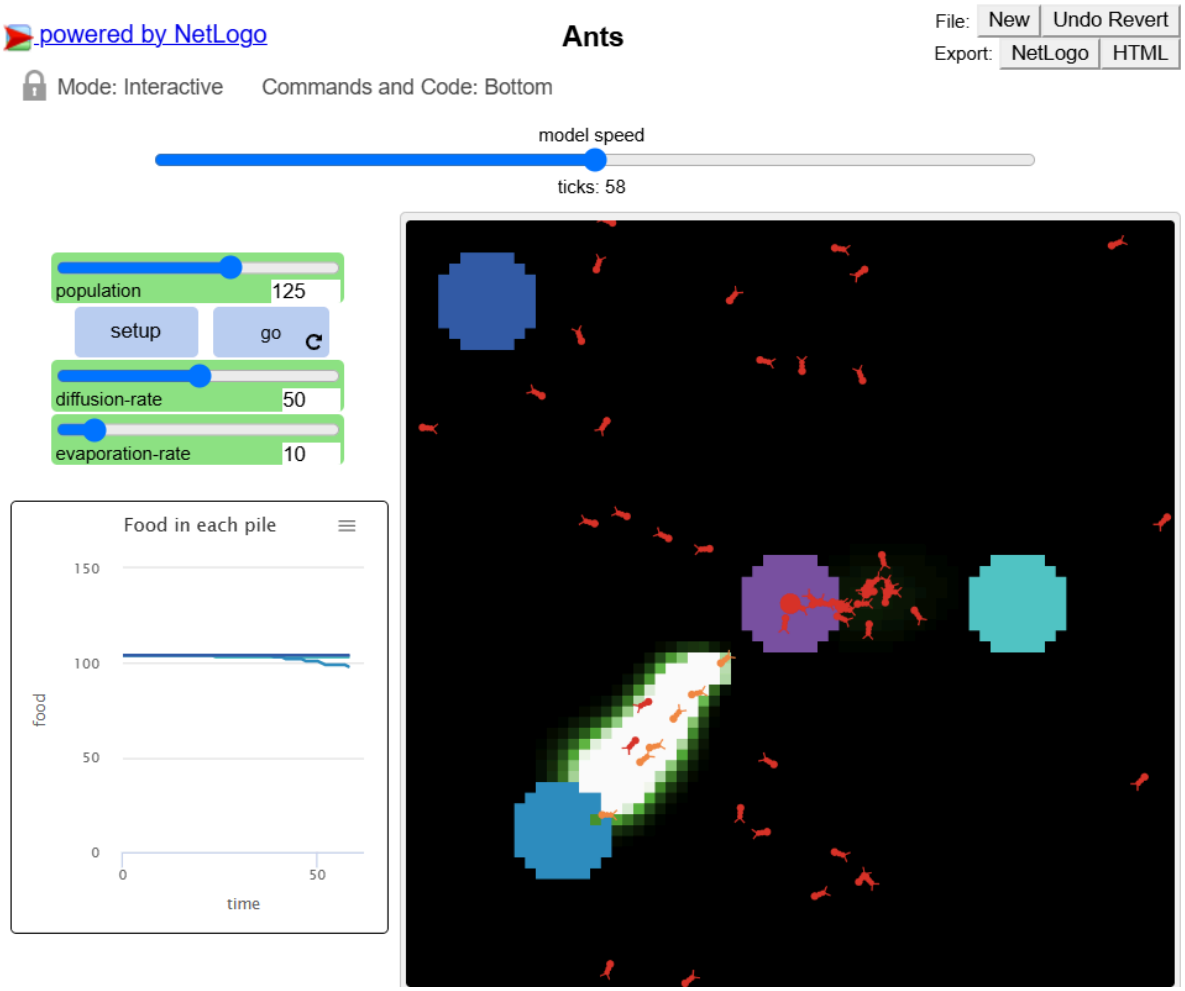
Nesse projeto essa ferramenta vai ser amplamente utilizada, pois é um trabalho em equipe e isso faz com que o código seja sempre atualizado a cada modificação.

Faremos um repositório no git que será copiado para uma máquina local, que irá fazer alterações nessa cópia, que será preparada para fazer um commit (alteração) local, depois essa alteração vai ser upada no repositório remoto do github pra que todos os outros colegas de equipe possam utilizar a nova versão do código, não só por isso, mas esses commits antigos no repositório remoto podem ser acessados, basicamente podemos “voltar no tempo” para corrigir algum erro.

Principais comandos utilizados:

```
# 1. Clonar o repositório
git clone <URL_DO_REPOSITORIO>

# 2. Entrar na pasta do repositório
cd <nome-do-repositorio>

# 3. Atualizar a cópia local com o repositório remoto
git pull origin <branch>

# 4. Adicionar alterações ao staging area
git add .

# 5. Fazer um commit local com uma mensagem descritiva
git commit -m "Descrição do commit"

# 6. Enviar alterações para o repositório remoto
git push origin <branch>

# Comandos auxiliares
git status    # Verifica o status do repositório local
git log       # Exibe o histórico de commits
```

"No NetLogo, vamos realizar um estudo sobre agentes reativos utilizando uma biblioteca pronta do NetLogo, com base no modelo localizado em Biology > Ants, cujo código original é o seguinte:"
https://www.netlogoweb.org/launch#https://www.netlogoweb.org/assets/modelslib/Sample%20Models/Biology/Ants.nlogo



Resumo da análise:

- **Formigas reativas**: Elas buscam comida, pegam, retornam ao formigueiro e repetem o processo.
- **Patches de comida**: São áreas onde as formigas encontram comida, que pode ser retirada (mudando a cor do patch).

DESENVOLVIMENTO

UM DETALHAMENTO SOBRE AS ALTERAÇÕES FEITAS NO CÒDIGO

Portanto, fizemos adaptações ao código original, este modelo é uma evolução do código original da simulação de formigas no NetLogo, incorporando mais complexidade com a adição de predadores, obstáculos, e feromônios (químicos). As mudanças incluem:

1. **Novas Raças de Agentes:**
    - Formigas: Representadas pela raça `formigas`, que podem ou não estar carregando comida.
    - Predadores: Agora há uma raça de `predadores` que caçam as formigas.
2. **Propriedades Adicionadas aos Patches:**
    - chemical: Representa a quantidade de feromônio (substância química) nos patches.
    - food: Representa a quantidade de comida disponível nos patches.
    - nest?: Indica se um patch é parte do ninho (formigueiro).
    - nest-scent: Refere-se à quantidade de feromônio emitido perto do ninho.
    - food-source-number: Número que identifica diferentes fontes de comida (1, 2 ou 3).
3. **Criação de Obstáculos:**
    - Agora há uma função `setup-obstacles` que coloca obstáculos em 2% dos patches, impedindo o movimento das formigas e predadores.
4. **Funções de Movimentação e Ações das Formigas:**
    - As formigas agora podem procurar comida (`look-for-food`) e retornar ao ninho (`return-to-nest`), mudando sua cor ao carregar comida.
    - Evitar obstáculos: As formigas agora ajustam sua direção aleatoriamente quando encontram um obstáculo (`avoid-obstacles`).
5. **Ações dos Predadores:**
    - Caçar formigas: Os predadores agora caçam formigas, movendo-se em direção a elas. Se a distância entre o predador e a formiga for menor que 1 unidade, o predador tem uma chance de 5% de matar a formiga.
    - Caso o predador não encontre uma formiga, ele gira aleatoriamente.
6. **Interação com Feromônios:**
    - As formigas agora podem seguir trilhas de feromônios (chemicals) quando não encontram comida. Elas se movem em direção aos patches com feromônios mais fortes usando a função `uphill-chemical`.
    - Quando as formigas retornam ao ninho, elas seguem o feromônio do ninho para encontrar o caminho de volta mais facilmente, usando a função `uphill-nest-scent`.
7. **Difusão de Feromônios:**
    - A quantidade de feromônios nos patches se difunde ao longo do tempo, com um valor específico de evaporação e taxa de difusão `(diffuse chemical)`.
8. **Visualização e Cores:**
    - Os patches têm cores específicas para indicar seu estado: violetas para o ninho, ciano, céu e azul para as fontes de comida, e diferentes tonalidades de verde para indicar a intensidade dos feromônios nos patches.

Trechos de Código relevantes:

Funções de Movimentação do Agente Formiga:¨Setup Procedures e Go Procedures

O Setup define o ponto de partida da simulação.

O Go controla a dinâmica contínua, ou seja, como o modelo evolui no tempo.

Ambos trabalham juntos para criar uma simulação funcional e coerente, com o Setup garantindo um estado inicial adequado e o Go gerenciando as mudanças ao longo do tempo.

As formigas podem procurar comida com a função(`look-for-food`) e retornar ao ninho (`return-to-nest`), mudando sua cor ao carregar comida.

As formigas agora ajustam sua direção aleatoriamente quando encontram um obstáculo (`avoid-obstacles`).

COMPORTAMENTO DO AGENTE FORMIGA

```
to go
 ;; Formigas realizam suas tarefas
 ask formigas [
   ifelse color = red [
     look-for-food
   ] [
     return-to-nest
   ]
   avoid-obstacles
   fd 1
 ]
```

As formigas (uma variável): vão ser vermelhas enquanto a função look-for-food for falsa, caso contrário vão ser laranja(pois dentro da função look-for-food,ela infere que se for verdadeira a formiga será laranja).além de mudarem de cor, elas rotacionam 180 graus, e “consomem” um pedaço da comida.

Dessa forma, se look-for-food é verdadeira, portanto, a função return-to-nest também é verdadeira,esta função é responsável.

Função look-for-food(No Agente Formiga)

```
to look-for-food
  if food > 0 [
    set color orange
    set food food - 1
    rt 180
    stop
  ]
  if (chemical >= 0.05) and (chemical < 2) [
    uphill-chemical
  ]
end
```

Podemos resumir a função look-for-food da seguinte forma:

`if food > 0`**:** Verifica se o patch onde a formiga está atualmente contém comida (variável `food` > 0).

Ações caso haja comida:

`set color orange`**:** Altera a cor da formiga para laranja, indicando que ela está carregando comida.

`set food food - 1`**:** Reduz a quantidade de comida disponível no patch em 1 unidade, simulando a coleta pela formiga.

`rt 180`**:** Faz a formiga girar 180 graus, apontando na direção oposta, para começar a retornar ao ninho.

`stop`**:** Interrompe a execução do restante do procedimento, pois a tarefa atual (encontrar comida) foi concluída.

Ações caso não haja comida:

`if (chemical >= 0.05) and (chemical < 2):` Verifica se a quantidade de feromônio no patch está em uma faixa que indica proximidade de comida (valores entre 0.05 e 2). Esse intervalo evita que a formiga siga rastros muito fracos ou muito fortes (que podem indicar o ninho).

Ação caso esteja no intervalo:

`uphill-chemical`**:** E também uma função que move a formiga na direção do gradiente de feromônio, procurando áreas onde a concentração de feromônio é maior. Isso simula a comunicação química das formigas na busca por comida.

Função avoid-obstacles do Agente Formiga:Um Aprimoramento de Mecanismo do Agente para contornar Obstáculos

```
to avoid-obstacles
  let next-patch patch-ahead 1
  if next-patch = nobody or [pcolor] of next-patch = grey [
    rt random 180 ;; Ajusta direção aleatoriamente para evitar
bloqueios
  ]
end
```

Dentro de to avoid-obstacles, as seguintes funções ou primitivas do NetLogo são utilizadas:

Let next-patch e if next-patch

`patch-ahead:` Retorna o patch à frente da tartaruga.

`[pcolor]` of patch: Acessa a propriedade de cor de um patch.

`rt`: Faz a tartaruga girar.

`random:` Gera um número aleatório dentro de um intervalo especificado.

`patch-ahead`**:** Esta é uma **primitiva do NetLogo**. Ela retorna o patch que está a uma distância específica (`1` neste caso) na direção atual da tartaruga (formiga ou predador).

Função implícita: `patch-ahead` não é definida diretamente pelo programador, mas é uma função interna do NetLogo.

`let next-patch`: Cria uma variável local que armazena o patch à frente.

`if next-patch`: Verifica as condições relacionadas ao patch à frente (se existe ou se é um obstáculo).

Setup Procedures

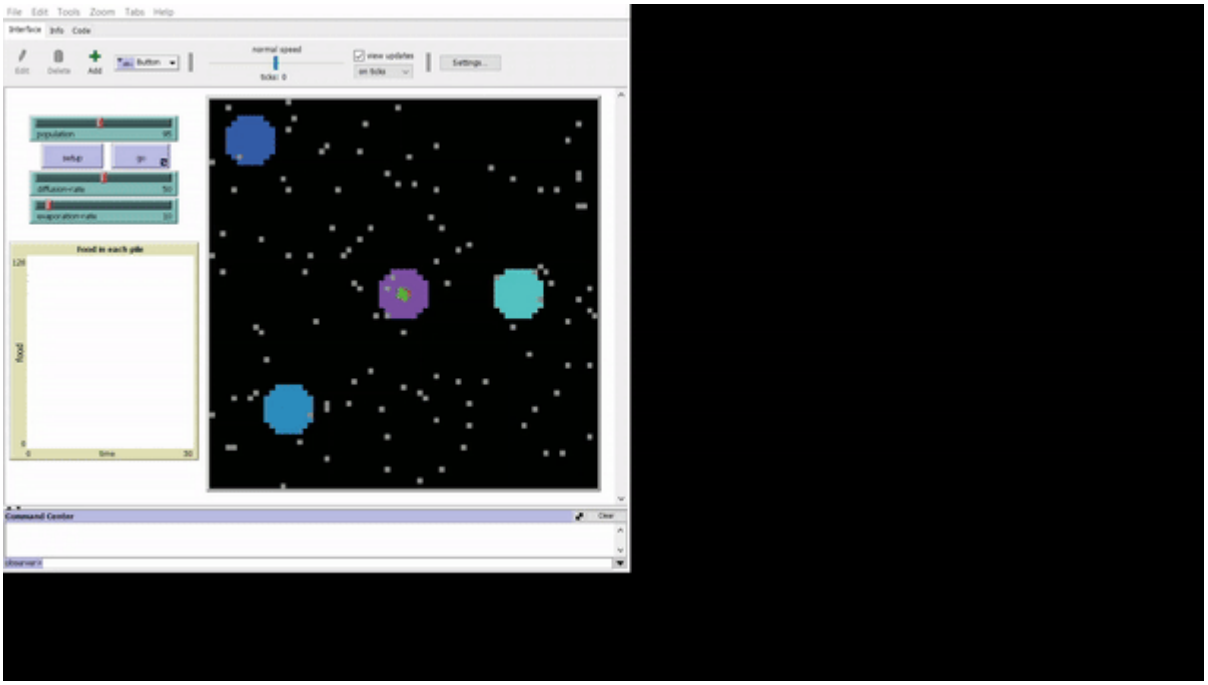
```
to setup-food
 if (distancexy (0.6 * max-pxcor) 0) < 5 [
   set food-source-number 1
 ]
 if (distancexy (-0.6 * max-pxcor) (-0.6 * max-pycor)) < 5 [
   set food-source-number 2
 ]
 if (distancexy (-0.8 * max-pxcor) (0.8 * max-pycor)) < 5 [
   set food-source-number 3
 ]
 if food-source-number > 0 [
   set food one-of [1 2]
 ]
end
```

```
to setup-obstacles
 ask patches [
   if random 100 < 2 [ ;; 2% dos patches serão obstáculos
     set pcolor grey
   ]
 ]
end
```

```
to setup-patches
 ask patches [
   setup-nest
   setup-food
   recolor-patch
 ]
```

Como setup-nest não foi abordado ele não foi incluido

RESULTADOS



Esse código implementa um modelo em NetLogo para simular a interação entre formigas e predadores em um ambiente com fontes de comida e obstáculos. Aqui está um resumo dos resultados esperados:

Formigas e Predadores:

As formigas têm comportamento de busca e transporte de comida para o ninho, usando feromônios para navegar de forma eficiente.

Os predadores caçam formigas ativamente, diminuindo sua população ao capturá-las.

Patches (Ambiente):

O ambiente é dividido em células (patches) que podem conter comida, obstáculos ou o ninho.

A comida está localizada em três fontes específicas, e o ninho é centralizado.

Os patches têm níveis de feromônios que influenciam os movimentos das formigas.

Movimento e Interação:

As formigas seguem gradientes químicos (feromônios) ou o cheiro do ninho para localizar comida e retornar.

Os predadores se movem aleatoriamente até encontrar formigas, que eles perseguem e eliminam ao se aproximarem.

Obstáculos:

Elementos no ambiente (representados como patches cinzas) bloqueiam o movimento direto e forçam os agentes a recalcular sua rota.

Dinâmica do Sistema:

Com o tempo, os feromônios evaporam e se difundem, formando trilhas que representam as interações das formigas com o ambiente.

A população de formigas pode diminuir se os predadores forem muito eficazes, criando um equilíbrio dinâmico entre caça e busca por recursos.

Resultados Potenciais:

- Eficiência de Busca: As formigas devem ser eficientes na coleta de comida devido ao uso de trilhas de feromônio.
- Impacto dos Predadores: A presença de predadores pode limitar a coleta de recursos e influenciar o comportamento das formigas.
- Influência de Obstáculos: A dispersão de agentes e a eficiência na coleta de comida serão afetadas pela distribuição de obstáculos.
- Evolução Dinâmica: O modelo pode revelar padrões emergentes, como rotas preferenciais ou áreas de alto risco para formigas.

Esse modelo é útil para estudar ecossistemas, comportamento de enxames, e até mesmo para aplicações em inteligência artificial e otimização.

DISCUSSÃO

Esse modelo gera diversas discussões relevantes:

1. Ecologia e comportamento animal:

    - O comportamento coletivo das formigas no uso de feromônios reflete como elas coordenam ações em grupo sem liderança central. Isso pode ser explorado para entender a eficiência do comportamento emergente em populações.
    - A interação entre predadores e presas representa uma dinâmica ecológica clássica. O equilíbrio entre o número de formigas e o sucesso dos predadores pode ser estudado.
    - Os obstáculos no ambiente afetam o comportamento de busca e navegação. Essa interação levanta questões sobre como barreiras físicas

influenciam ecossistemas.

2. Modelagem computacional:
    - O modelo mostra como o NetLogo é útil para simular interações locais que levam a comportamentos globais. A alteração de parâmetros, como taxa de evaporação de feromônios ou velocidade dos agentes, pode revelar padrões interessantes.
    - Ajustes e otimizações nos parâmetros podem ajudar a entender melhor como os sistemas adaptativos funcionam.
3. Aplicações práticas:
    - Em robótica, esse modelo inspira algoritmos de enxame para resolver problemas de busca e navegação, como drones explorando áreas.
    - Em logística, pode ser usado para planejar rotas otimizadas com base no comportamento coletivo das formigas.
4. Aspectos teóricos:
    - O modelo também levanta questões sobre como o comportamento emergente pode ser simulado e controlado. Por exemplo, como garantir que as formigas sejam eficientes mesmo com predadores presentes?
    - A interação entre agentes e ambiente também pode ser explorada para compreender melhor a relação entre estímulos locais e decisões globais.

Esse modelo oferece um ponto de partida para estudar tanto sistemas naturais quanto artificiais, destacando a relevância de simulações na ciência e na engenharia.

CONCLUSÃO

A implementação e simulação de agentes reativos no NetLogo permitiram uma análise prática das características desses sistemas, confirmando os conceitos teóricos discutidos. Os agentes demonstraram que, mesmo sem memória ou aprendizado, podem executar tarefas específicas de maneira eficaz quando guiados por regras simples e estímulos do ambiente.

Além disso, o uso do NetLogo como ferramenta de modelagem foi fundamental para facilitar o entendimento e a visualização do comportamento dos agentes, promovendo um aprendizado mais acessível e prático. Os ajustes realizados no modelo básico de formigas adicionam complexidade ao sistema, aproximando-o de situações reais, como a presença de predadores e obstáculos, o que ampliou a relevância do estudo.

No entanto, as limitações dos agentes reativos foram evidenciadas, especialmente em ambientes com maior imprevisibilidade ou requisitos de adaptação. Esses resultados reforçam

a importância de explorar outros tipos de agentes em inteligência artificial para aplicações que demandam maior flexibilidade ou aprendizado.

Por fim, o trabalho contribuiu para a compreensão do papel e do potencial dos agentes reativos, ao mesmo tempo em que destacou a necessidade de abordagens mais avançadas para resolver desafios mais complexos.