Inteligência artificial

Vinicius Ribeiro de Santana (1272221567)
Arley do Nascimento Vinagre (12722132338)
Bruno Sales Fiaes Carneiro (12723119186)
Tauan Santos Santana (12722216126)
Vitor Pio Vieira (1272220376)

Salvador – Bahia
2025

Projeto Hellmaze: Análise de diferentes tipos de algoritmos através do jogo do labirinto

RESUMO:

Este relatório tem como finalidade abordar os diferentes tipos de algoritmos de busca em um cenário com labirinto. O código foi escrito em Python e utilizamos a biblioteca matplotlib para criar o ambiente de análise e comparar os resultados entre os algoritmos A* e Q-learning, dentro do Agente adaptativo (Ambiente)

Palavras chaves:

Python, algoritmos de largura, profundidade, A*, Q-Learning e Agente Adaptativo.

INTRODUÇÃO:

Este projeto apresenta o desenvolvimento de algoritmos de busca capazes de navegar em um ambiente de labirinto, onde será usado o agente adaptativo para fazer o labirinto que altera a dificuldade de acordo com a velocidade que o algoritmo de busca resolve, como parte do trabalho A3 da Universidade Salvador (UNIFACS). A proposta central é investigar e implementar diversos algoritmos de busca, comparando seu desempenho em diferentes configurações de labirintos, variando tamanho, densidade de paredes e nível de dificuldade.

Para isso, foi utilizado o Python como linguagem base, com a biblioteca matplotlib para visualização e análise de resultados. Os algoritmos foram projetados para operar em ambientes de complexidade crescente, servindo como plataforma exploratória para estudos em sistemas multi-agentes, armazenamento de memória e escalabilidade de algoritmos de busca.

Com esse protótipo, buscamos não apenas demonstrar a navegação eficiente em labirintos, mas também oferecer uma estrutura agnóstica para futuros estudos em inteligência artificial, por exemplo, explorando frameworks para agentes de IA e expandindo para tarefas cooperativas entre múltiplos agentes. A expectativa é que, a partir deste trabalho, seja possível avaliar as diferenças de desempenho entre algoritmos (como B A*, Q-learning.), e como essas diferenças se manifestam em ambientes mais desafiadores.

METODOLOGIA:

O projeto Hellmaze foi conduzido seguindo um ciclo de desenvolvimento científico de quatro fases para garantir a validade e a estruturação da investigação sobre Algoritmos de busca em um ambiente de labirinto dinâmico. As fases foram: Planejamento, Implementação, Testes e Análise de Resultados.

Planejamento e definição de algoritmos:

Esta fase estabeleceu os pilares teóricos e práticos do projeto.

Algoritmos de busca: Foram escolhidos o algoritmo Busca A* e o Q-Learning.

Função do A*: O A* é idealizado para este projeto como referência de performance. Devido ao seu nível ideal de raciocínio e eficiência fixa, ele estabelece o padrão ótimo de solução a ser alcançado.

Função do Q-Learning: O Q-Learning foi selecionado devido ao seu grande potencial de crescimento e adaptabilidade . A expectativa é que, com o tempo, ele possui chances reais de superar o A*.

Agente Adaptativo (Ambiente): Foi concebido um Agente Adaptativo para atuar como labirinto. Sua função é ser o "campo de treinamento" dos algoritmos e ajustar a dificuldade de forma dinâmica de acordo com o desempenho observado.

Implementação e Otimização

A implementação concentrou-se na codificação e na integração dos componentes.

Linguagem e Ferramentas: Os algoritmos foram desenvolvidos em Python. A biblioteca Matplotlib foi integrada com o objetivo de gerar visualizações gráficas das rodadas para facilitar a análise.

Monitoramento: Um conjunto de tabelas é gerado para permitir a análise em tempo real do desempenho durante as rodadas de execução.

Otimização de aprendizado: Devido ao baixo rendimento inicial do Q-Learning em relação ao A*, onde o Q-Learning não foi capaz de alcançar o benchmark , foi necessária a aplicação do mecanismo de decaimento epsilon. O decaimento epsilon foi fundamental para reforçar o aprendizado do algoritmo Q-Learning.

Testes e Análises de Resultados

A fase de testes validou a eficácia do mecanismo de aprendizado, sendo concluída após 3600 rodadas.

Comprovação de evolução: Após a aplicação do decaimento epsilon, uma nova tabela de testes demonstrou que o algoritmo Q-Learning estava evoluindo como o esperado.

Paridade de Eficiência: Foi constatado um marco significativo após 2623 rodadas, quando o Q-Learning atingiu a mesma eficiência (1,06) que a Busca A*. Nesta mesma rodada de paridade, o Q-Learning resolveu o labirinto em 38 passos, ficando extremamente próximo ao mínimo do A*, que foi de 36 passos. Análise da convergência e potencial de superação o resultado reforça o potencial de adaptabilidade do Q-Learning.

A vs. Q-Learning*: O A* possui uma eficiência fixa, baseada em seu planejamento heurístico. Por outro lado, o Q-Learning é adaptativo e projetado para aprender continuamente e ajustar sua política de ação.

Tendência Futura: O ponto mais significativo é a tendência esperada de superação. Após a tabela Q ser totalmente preenchida, o Q-Learning deve ultrapassar a eficácia do A* em testes prolongados e em ambientes dinâmicos. Isso se deve à sua capacidade de encontrar soluções mais robustas e adaptar-se a mudanças ambientais para as quais o A* não estava preparado.

Evidência Gráfica: O gráfico de "Número de Passos por Episódio" valida visualmente esta progressão. A oscilação inicial da linha vermelha (Q-Learning) reflete a exploração, enquanto a compressão de sua base em direção à linha ideal do A* (Ótimo) demonstra a efetiva convergência do Q-learning.

Linguagens utilizadas:

Utilizamos a linguagem python e a biblioteca matplotib.

CARACTERÍSTICAS:

As características dos algoritmos de controle definem a essência da inteligência do Agente de Software:

Algoritmo de Busca A* (Raciocínio e Planejamento)

Busca Informada Heurística: O A* é classificado como um algoritmo de Busca Informada. Ele exibe inteligência ao usar uma função heurística para estimar o custo e a distância restantes até o objetivo.

Eficiência e Otimização: Sua inteligência reside na eficiência de encontrar a melhor solução conhecida em um ambiente estático. Nos resultados, sua baixa contagem de Passos (36 a 56) e Razão 1,00 comprovam sua eficácia em encontrar a rota ideal rapidamente.

Algoritmo Q-Learning (Aprendizado e Adaptação)

Decaimento de Epsilon (epsilon-Decay): Este é o mecanismo que permite ao algoritmo Q-Learning equilibrar o desejo de descobrir novas estratégias (exploração) com a necessidade de usar o que já aprendeu para obter recompensas máximas (explotação).

Epsilon Atual (eps): Este atributo é o "termômetro da curiosidade" do algoritmo. Ele representa a probabilidade de o algoritmo de busca escolher uma ação aleatória (exploração) em vez da ação que ele considera a melhor (explotação).

Taxa de Decaimento (epsilon-Decay): Atuando como um fator multiplicativo (ex: 0.995), ele reduz o valor de epsilon gradualmente após cada episódio.

Epsilon Mínimo (eps min): Este limite (ex: 0.05) garante que sempre haja uma chance residual de exploração, vital para evitar que o algoritmo fique preso em ótimos, locais sub-ótimos.

Representação de Conhecimento (Tabela Q): A Tabela Q é a memória do Q-Learning, onde ele armazena o valor esperado de cada par do Estado-Ação. A inteligência reside na forma como o algoritmo racionaliza as recompensas futuras ao tomar uma decisão no presente.

FUNCIONALIDADES DO LABIRINTO HELLMAZE:

O projeto Hellmaze opera como um sistema sofisticado de Inteligência Artificial, unindo três funcionalidades centrais: o Planejamento Racional, o Aprendizado Adaptativo e o Desafio Dinâmico.

Funcionalidade de Planejamento Racional (Algoritmo A*)

O algoritmo de Busca A* é o motor de raciocínio eficiente do projeto. Sua funcionalidade principal é encontrar o caminho com maior eficiência em questão de passos e servir de parâmetro comparativo com o Q-learning.

Ele utiliza uma heurística para estimar o custo restante até o objetivo.Esta estimativa permite que o A* evite explorar caminhos claramente ineficientes.

Nos testes, o A* funciona como um medidor de dificuldade, estabelecendo o referencial da menor quantidade de Passos possível em um ambiente estático (evidenciado pela Razão 1,00 nos resultados).

Funcionalidade de Aprendizado Adaptativo (Algoritmo Q-Learning)

O Q-Learning é o algoritmo de busca, que funciona com aprendizado por reforço. Aprende por experiência a melhor política de ação, em vez de seguir um caminho pré-definido.

Construção de Conhecimento: O Q-Learning armazena o valor de cada ação em uma Tabela Q (sua "memória").

Balanceamento da Estratégia: O mecanismo de decaimento de epsilon (epsilon-decay) regula a transição do comportamento do Q-Learning.

Inicialmente, o parâmetro epsilon-decay (eps), alto, garante a exploração do ambiente. Gradualmente, o método (epsilon-decay) reduz esse valor, forçando o algoritmo a explorar o conhecimento adquirido para maximizar as recompensas (evitando estagnação graças ao epsilon-decay.

Funcionalidade de Desafio Dinâmico (Agente Adaptativo de Ambiente)

O agente adaptativo é a entidade responsável por gerar e gerenciar o ambiente Hellmaze. Seu papel transcende o de um mero gerador de mapas; ele executa uma Inteligência Artificial de ambiente.

Controle de Complexidade: Sua função principal é aumentar a dificuldade do labirinto dinamicamente, forçando os algoritmos de busca a se adaptarem a novos obstáculos e tamanhos de grid.

Otimização do Treinamento: Este agente garante que o ambiente seja sempre desafiador, percebendo o desempenho dos algoritmos de buscas e ajustando o nível do labirinto para evitar que o algoritmo atinja a complacência, mantendo-o constantemente engajado no processo de adaptação e aprendizado.

Em resumo, as funcionalidades do projeto Hellmaze trabalham em conjunto: o A* mostra o que é ideal (raciocínio), o Q-Learning descobre o que é melhor (aprendizado adaptativo), e o Agente Adaptativo garante que a prova de fogo (o desafio) seja contínua e relevante.

RESULTADOS:

A Busca A* e o Q-learning foram comparados após um total de 3600 rodadas. Durante os testes, após 2623 rodadas, Q-learning atingiu a mesma eficiência (1,06) que a Busca A* nessa mesma rodada ele resolveu o labirinto em 38 passos, próximo ao mínimo de A* (que foram 36 passos).

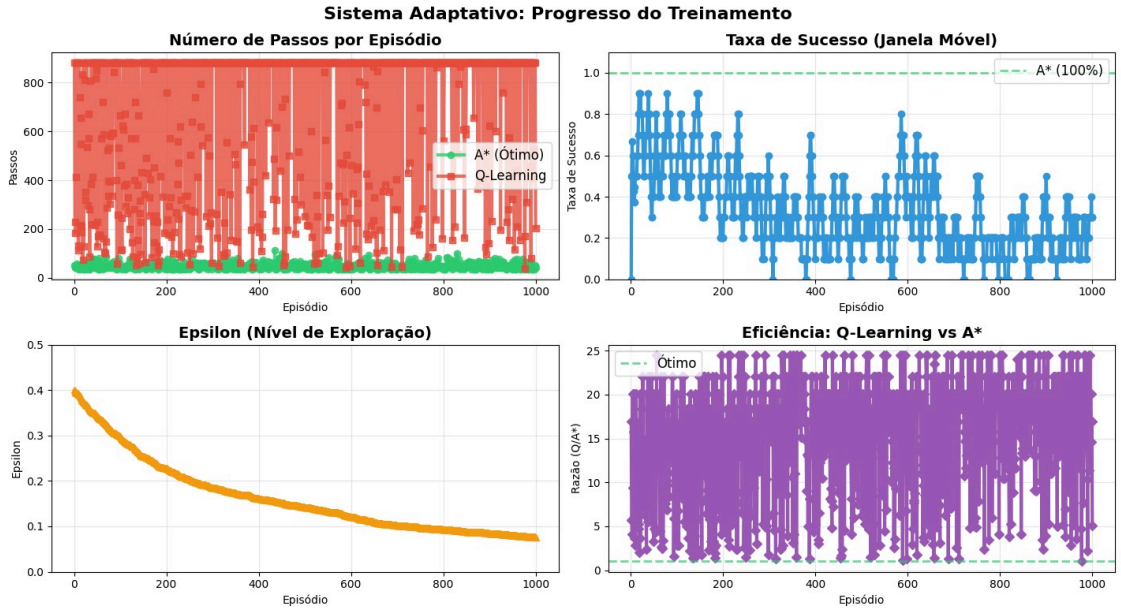
Melhores Rodadas

ALGORITIMO DE BUSCA	PASSOS	EFICIENCIA
A*	36	1,06
	40	5,75
Q-Learning	38	1,06
	42	1,17

O ponto mais significativo reside no comportamento esperado do Q-Learning. Enquanto o algoritmo A* tem uma eficiência fixa, o Q-Learning é adaptativo. Ele é projetado para aprender continuamente e ajustar sua política de ação. Uma vez que a Tabela Q esteja totalmente preenchida, a tendência é que o Q-Learning, em testes prolongados e em ambientes dinâmicos, ultrapasse a eficácia do A* ao encontrar soluções mais robustas e generalizáveis, ou ao se adaptar a mudanças ambientais para as quais o A* não foi preparado.

O gráfico de "Número de Passos por Episódio" valida a convergência do Q-Learning. A oscilação inicial da linha vermelha (Q-Learning) reflete a exploração, mas sua base se comprime em direção à linha ideal do A* (Ótimo).

Gráficos Demonstrativos de Desempenho



Número de passos por episódio:

Este gráfico (superior esquerdo) ilustra o custo de cada tentativa do agente em termos de ações tomadas.

Busca A* (Ótimo): A linha verde representa o benchmark ideal e estável (próximo de 50 passos).

Q-Learning: A linha vermelha mostra a volatilidade inicial. Os picos altos (próximos de 900 passos) indicam que, no início, o Agente de Software falha frequentemente e explora o labirinto de forma ineficiente. A compressão gradual dos pontos vermelhos em direção à linha verde (A*) comprova o aprendizado e a convergência do Q-Learning.

Taxa de Sucesso (Janela Móvel):

Este gráfico (superior direito) mede a porcentagem de vezes em que o Agente de Software encontra a saída do labirinto com sucesso ao longo de um conjunto de episódios (janela móvel).

Padrão de Aprendizado: A linha azul mostra que o sucesso varia bastante no início, mas a tendência é que a taxa de acerto aumente com o tempo, embora com oscilações devido à exploração residual. O objetivo é que essa linha se aproxime da linha tracejada (A* 100%).

Epsilon (Nível de exploração):

Este gráfico (inferior esquerdo) é a chave que explica o comportamento dos outros três. Ele mapeia o mecanismo de decaimento de epsilon.
Início (Exploração): O epsilon começa alto (cerca de 0,4), fazendo com que o

Q-Learning priorize ações aleatórias, o que explica a alta volatilidade e as muitas falhas no gráfico de passos. Decaimento (Explotação): O epsilon diminui gradualmente em cada episódio, o que faz o Agente de Software confiar cada vez mais na sua Tabela Q (o conhecimento aprendido), resultando em menos passos e maior taxa de sucesso ao longo do tempo.

Eficiência: Q-Learning vs A*:

Este gráfico (inferior direito) é uma métrica de desempenho relativo. Ele compara a eficiência do Q-Learning com a do A*.
No início, os picos altos (acima de 20) mostram que o Q-Learning é muito ineficiente.A linha tracejada inferior representa a Eficiência Ótima.
O objetivo do treinamento é que o Q-Learning, em média, atinja esse nível, confirmando a paridade de eficiência observada nos resultados tabulares.
A compressão dos pontos em direção à base do gráfico, próxima à razão 1, indica o sucesso do aprendizado.

CONCLUSÃO:

A análise comparativa entre o A* e o Q-Learning, junto com o Agente Adaptativo, evidencia três papéis complementares dentro do sistema.

O Agente Adaptativo é responsável pela criação e equilíbrio do ambiente que os algoritmos vão atuar. O A* é determinístico, capaz de fornecer um caminho curto e medir com precisão a dificuldade real dos labirintos. Sua estabilidade e confiabilidade permitem que ele funcione como referência absoluta, servindo de base para avaliar desempenho, taxa de sucesso. Em contraste, o Q-Learning apresenta um comportamento adaptativo: embora não alcance o nível ótimo do A*, demonstra evolução gradual ao longo do treinamento, reduzindo o número de passos e aumentando sua capacidade de navegar no ambiente. Suas limitações, como dependência da exploração e lentidão em ambientes complexos, refletem a natureza de um agente que aprende por tentativa e erro.

Assim, a combinação dos dois métodos permite tanto medir objetivamente a dificuldade do ambiente quanto observar o progresso do agente, oferecendo uma estrutura robusta para avaliar aprendizado, adaptação e eficiência na navegação autônoma.

Links e Referências:

Github:

https://github.com/VitorPio7/Prot-tipo-Agente?tab=readme-ov-file

Artificial Intelligence: A Modern Approach, 4th US ed. Stuart J. Russell e Peter Norvig

Datacamp:

https://www.datacamp.com/pt/tutorial/introduction-q-learning-beginner-tutorial

Stackoverflow:

https://stackoverflow.com/questions/53198503/epsilon-and-learning-rate-decay-in-epsilon-greedy-q-learning

Artigo científico:

Silva, R. C. G., & Parpinelli, R. S. (s.d.). Análise do desempenho de algoritmos clássicos de busca de caminho em ambiente de navegação. Universidade do Estado de Santa Catarina, Departamento de Ciência da Computação.