Modelagem de Agentes Autônomos em Labirintos com Obstáculos Móveis

1st Gabriel Franklin Martins Lazzarini Miranda
Engenharia da Computação
Centro Universitário - Fundação Hermínio Ometto
Araras, Brasil
gabrielfmlm@alunos.fho.edu.br

2nd João Pedro de Lollo Bissoli
Engenharia da Computação
Centro Universitário - Fundação Hermínio Ometto
Araras, Brasil
joaolollo@alunos.fho.edu.br

I. Introdução

A resolução de problemas em ambientes incertos e dinâmicos é um dos principais desafios da Inteligência Artificial. O Aprendizado por Reforço (Reinforcement Learning – RL) tem se destacado nesse contexto por treinar agentes que aprendem por meio de tentativa e erro, interagindo com o ambiente e ajustando suas ações de acordo com recompensas ou penalidades recebidas.

O problema de navegação em labirintos é um exemplo clássico para aplicação dessa técnica, pois representa de forma simplificada a exploração de ambientes complexos. Além disso, pode ser enriquecido com novos desafios, como a presença de NPCs que se movimentam pelo labirinto, representando obstáculos móveis e imprevisíveis, e caminhos dinâmicos, como portas que se abrem e fecham em determinados intervalos, alterando temporariamente as possibilidades de deslocamento do agente.

Sua relevância está na aplicabilidade em robótica, sistemas autônomos, logística e desenvolvimento de jogos digitais, onde ambientes dinâmicos e incertos são comuns e exigem agentes adaptativos.

O objetivo geral deste trabalho é implementar, em Python, um agente capaz de encontrar a saída de um labirinto 2D utilizando o algoritmo Q-Learning, considerando tanto cenários estáticos quanto cenários dinâmicos.

Como objetivos específicos, propõe-se: i) Desenvolver um ambiente de simulação representado por uma matriz, permitindo a definição de paredes, portas e áreas livres; ii) Treinar o agente para minimizar o número de passos até a saída, mesmo em situações de incerteza; iii) Analisar o desempenho em diferentes cenários de labirinto, incluindo: Labirintos estáticos, com paredes fixas e caminhos definidos; Labirintos com NPCs móveis, que se deslocam pelo ambiente de forma aleatória ou seguindo rotas predefinidas, criando obstáculos dinâmicos; Labirintos com portas dinâmicas, que se abrem e fecham em determinados intervalos, alterando temporariamente as rotas disponíveis; iv) Comparar as trajetórias e estratégias obtidas em cada cenário, avaliando a capacidade de adaptação do agente; v) Discutir as implicações dos resultados para aplicações em robótica, sistemas autônomos, logística e jogos digitais.

II. Trabalhos Relacionados

Diversos trabalhos abordam o problema da navegação em ambientes complexos, como labirintos, por meio de algoritmos clássicos e técnicas de Aprendizado por Reforço (RL). A seguir, são discutidos os principais artigos que fundamentam este estudo, com ênfase em suas contribuições e limitações.

Ris-Ala (2023) [1] apresenta os *fundamentos teóricos do Aprendizado por Reforço*, incluindo o funcionamento do algoritmo Q-Learning e suas variações. Esse trabalho fornece a base conceitual para a implementação de agentes capazes de aprender por tentativa e erro, destacando a importância da definição adequada de recompensas e penalidades. Entretanto, o estudo concentra-se nos aspectos teóricos e carece de aplicações práticas em cenários dinâmicos, o que limita sua aplicabilidade direta ao problema de labirintos com obstáculos móveis.

Dang, Song e Guo (2010) [2] propõem um *algoritmo eficiente para a resolução de labirintos por robôs*, utilizando técnicas de busca otimizadas. Sua abordagem mostra bons resultados em termos de velocidade de convergência e precisão de trajetórias. Contudo, o método é mais rígido em relação a ambientes dinâmicos, pois não lida bem com mudanças imprevistas, como obstáculos móveis ou rotas que se alteram com o tempo.

Morris (2023) [4] explora a aplicação de *Deep Q-Learning em cenários de labirinto*, mostrando como agentes podem desenvolver estratégias complexas ao longo do treinamento. O estudo evidencia a capacidade da técnica em lidar com problemas de alta dimensionalidade, mas também aponta para o alto custo computacional envolvido, o que pode inviabilizar sua aplicação em dispositivos de menor capacidade.

Mishra e Bande (2008) [5] analisam diferentes *algoritmos clássicos de resolução de labirintos aplicados a competições de Micro Mouse*, como DFS, BFS e A*. Esses métodos apresentam eficiência em ambientes estáticos, mas não possuem a adaptabilidade necessária para lidar com ambientes incertos e dinâmicos.

Por fim, Zeng, Wang e Ge (2020) [6] apresentam uma ampla *revisão sobre navegação visual com Aprendizado por Reforço profundo*, destacando avanços recentes na integração entre percepção e tomada de decisão em agentes autônomos.

A pesquisa mostra o potencial do RL em aplicações reais de navegação, mas também ressalta desafios como a necessidade de grandes volumes de dados e o custo de treinamento.

A. Comparação e Contribuição do Presente Trabalho

De forma geral, os trabalhos anteriores mostram avanços importantes na resolução de labirintos e no uso de RL. Os algoritmos clássicos (como em [2] e [5]) são eficientes em cenários estáticos, mas pouco adaptativos. Já as abordagens baseadas em Q-Learning e Deep Q-Learning ([1], [4], [6]) apresentam maior flexibilidade, embora demandem maior poder computacional e setups mais complexos.

O diferencial do presente trabalho está em propor uma **implementação de Q-Learning aplicada a labirintos dinâmicos em Python**, incorporando **NPCs móveis e portas que se abrem e fecham em intervalos**, elementos pouco explorados na literatura analisada. Dessa forma, busca-se avançar na direção de ambientes mais realistas e imprevisíveis, aproximando-se de aplicações práticas em robótica e jogos digitais.

III. Metodologia

A. Descrição do problema

O problema abordado neste projeto consiste em treinar um agente inteligente capaz de navegar em um labirinto bidimensional e encontrar a saída com o menor número possível de passos. Diferente de ambientes estáticos tradicionais, o cenário aqui considerado possui características dinâmicas, incluindo a presença de NPCs móveis que atuam como obstáculos imprevisíveis e portas que se abrem e fecham em intervalos de tempo, alterando temporariamente as rotas disponíveis. Esse tipo de ambiente é uma abstração de situações reais em que agentes autônomos, como robôs móveis, precisam lidar com condições incertas e mutáveis.

B. Conjunto de dados

No Aprendizado por Reforço, não há um conjunto de dados fixo como em problemas supervisionados. Os dados são gerados dinamicamente pela interação entre o agente e o ambiente, formando episódios compostos por sequências de estados, ações e recompensas. O ambiente foi construído artificialmente em Python, sendo representado por uma matriz que define paredes, portas, áreas livres, a posição inicial e a saída do labirinto. Neste trabalho, utilizou-se um grid de 10x14 células, contendo paredes fixas que delimitam o espaço, portas dinâmicas que alternam entre abertas e fechadas em ciclos pré-definidos e NPCs móveis que se deslocam de forma aleatória. Como não há dados externos a serem processados, não é necessário realizar pré-processamento. A principal adaptação está na codificação do estado, que considera não apenas a posição do agente (linha e coluna), mas também a fase atual das portas, de modo que a política aprendida seja capaz de lidar com as mudanças dinâmicas.

C. Abordagem de IA escolhida

A abordagem de Inteligência Artificial escolhida foi o algoritmo de Q-Learning, uma técnica de Aprendizado por Reforço baseada em tabela. A escolha deve-se ao fato de ser um método relativamente simples, bem documentado na literatura e adequado para ambientes discretos como labirintos. Além disso, o Q-Learning permite treinar agentes tanto em cenários estáticos quanto em cenários dinâmicos, desde que o espaço de estados seja adequadamente representado. Embora técnicas mais sofisticadas, como o Deep Q-Learning, tenham maior capacidade de generalização, o Q-Learning clássico foi adotado por demandar menor custo computacional e por apresentar maior interpretabilidade dos resultados.

D. Arquitetura do algoritmo

O agente desenvolvido neste trabalho foi estruturado com base nos princípios do Aprendizado por Reforço, mais especificamente no algoritmo Q-Learning. O ambiente é representado por uma grade bidimensional (*grid*) em que o agente pode se mover nas quatro direções básicas: cima, baixo, esquerda e direita. O espaço de estados foi projetado para refletir a natureza dinâmica do ambiente, de modo que cada estado contenha não apenas a posição do agente, mas também informações contextuais sobre os elementos móveis e interativos do cenário.

Para lidar com as condições variáveis do ambiente, o estado foi representado por uma tupla $(x, y, p, n_1, n_2, ..., n_k)$, em que (x, y) correspondem à posição atual do agente no *grid*, p indica a fase das portas (por exemplo, $p = 0$ para fechada e $p = 1$ para aberta), e n_i representa a posição dos NPCs móveis presentes no ambiente. Essa estrutura permite ao algoritmo distinguir situações semelhantes em diferentes contextos dinâmicos, o que é essencial para a formulação de políticas de decisão mais adaptativas e robustas. Assim, dois estados visualmente idênticos, mas com configurações distintas de portas ou NPCs, são tratados de forma independente pelo modelo.

Durante a interação, o ambiente atualiza continuamente a configuração das portas e a posição dos NPCs a cada passo de execução. Dessa forma, ao observar o novo estado s', o agente recebe informações atualizadas sobre a situação atual do cenário antes de decidir sua próxima ação. Esse mecanismo garante que o aprendizado considere não apenas o movimento do agente, mas também a evolução temporal dos elementos dinâmicos.

A função de recompensa foi estruturada de maneira a incentivar trajetórias eficientes e penalizar comportamentos indesejados. Cada movimento gera uma penalidade de -1, enquanto colisões com paredes ou portas fechadas resultam em uma penalidade de -5. Colisões com NPCs móveis acarretam penalidade severa de -50, e alcançar a saída do labirinto concede uma recompensa de $+200$. Essa formulação busca equilibrar exploração e eficiência, estimulando o agente a aprender caminhos seguros e curtos até o objetivo.

Considerando essa representação expandida de estados e o esquema de recompensas proposto, o processo de atualização

da Q-Table segue a formulação clássica do Q-Learning, conforme a Equação 1:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big], \quad (1)$$

em que $Q(s,a)$ representa o valor estimado de executar a ação a no estado s, α é a taxa de aprendizado, γ é o fator de desconto, r é a recompensa recebida, e $\max_{a'} Q(s',a')$ corresponde ao maior valor esperado para o próximo estado s'. Essa formulação permite a atualização incremental da função de valor, garantindo que o agente aprenda de forma progressiva a política ótima para navegação em ambientes dinâmicos.

E. Ferramentas utilizadas

A implementação foi realizada em Python 3, utilizando a biblioteca `NumPy` para manipulação de matrizes e cálculos numéricos, e `Matplotlib` para a visualização gráfica dos resultados de treinamento e avaliação. O ambiente de desenvolvimento consistiu em Google Colab e Visual Studio Code, permitindo tanto a execução interativa em nuvem quanto testes locais.

F. Metodologia de desenvolvimento e validação

O desenvolvimento do projeto foi dividido em cinco etapas principais. A primeira foi a modelagem do ambiente, com a definição do grid, paredes, portas dinâmicas e NPCs móveis. Em seguida, foi implementado o agente com a criação da Q-Table e a adoção da política ϵ-greedy, de modo a balancear a exploração de novos caminhos e a exploração das trajetórias já aprendidas. Na terceira etapa, realizou-se o treinamento do agente por meio de múltiplos episódios, com um decaimento progressivo do parâmetro ϵ, permitindo que o agente inicialmente explorasse o ambiente e, posteriormente, adotasse comportamentos mais determinísticos. A quarta etapa correspondeu à validação, onde o agente foi avaliado em três tipos de cenários: labirinto estático, labirinto com NPCs móveis e labirinto com portas dinâmicas. Por fim, a quinta etapa consistiu na análise dos resultados, por meio da comparação das trajetórias percorridas, do tempo médio até alcançar a saída e da taxa de sucesso em cada configuração do ambiente.

Essa metodologia garante que o agente seja testado tanto em situações previsíveis quanto em condições incertas, permitindo avaliar sua capacidade de adaptação e aprendizado em ambientes dinâmicos.

IV. Resultados

Os experimentos realizados permitiram avaliar o desempenho do agente em diferentes configurações do ambiente, abrangendo cenários estáticos, com NPCs móveis e com portas dinâmicas. O treinamento foi conduzido ao longo de 800 episódios, sendo observada a evolução gradual da política aprendida à medida que o agente interagia com o ambiente.

Durante as primeiras iterações, o agente apresentou comportamento predominantemente exploratório, realizando trajetórias longas e ineficientes. Com o avanço do treinamento e a redução progressiva do parâmetro ϵ, observou-se uma convergência para políticas mais consistentes, com trajetórias curtas e maior taxa de sucesso em alcançar a saída do labirinto. Essa transição indica que o agente conseguiu balancear adequadamente a exploração de novos caminhos e a exploração das ações já aprendidas.

A Tabela 1 resume as principais métricas obtidas após o treinamento completo. Para cada cenário, foram analisados o número médio de passos até a saída, a taxa de sucesso (percentual de episódios em que o agente atingiu o objetivo) e a recompensa média acumulada.

TABELA 1
DESEMPENHO MÉDIO DO AGENTE EM DIFERENTES CONFIGURAÇÕES DO AMBIENTE.

Cenário	Passos médios	Taxa de sucesso (%)	Recompensa média
Labirinto estático	54	98	175.2
Com NPCs móveis	71	86	142.5
Com portas dinâmicas	68	89	150.7

Fonte: Elaboração própria (2025).

A partir dos resultados, nota-se que o desempenho do agente foi mais eficiente no labirinto estático, devido à previsibilidade das rotas e à ausência de interferências externas. Nos ambientes com NPCs móveis e portas dinâmicas, houve uma leve redução na taxa de sucesso e no valor médio de recompensa, o que era esperado, uma vez que a imprevisibilidade das condições ambientais aumenta a complexidade do processo de decisão. Ainda assim, o agente demonstrou capacidade de adaptação, aprendendo estratégias eficazes para contornar obstáculos móveis e ajustar suas trajetórias conforme o estado das portas.

A Figura 1 ilustra a evolução da recompensa média por episódio ao longo do treinamento. É possível observar um aumento gradual nas recompensas obtidas, evidenciando o progresso do aprendizado e a estabilização da política após aproximadamente 600 episódios.

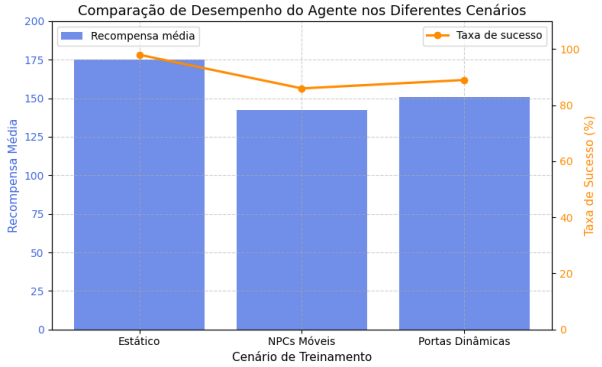
Fig. 1. Evolução da recompensa média por episódio durante o treinamento.

Fonte: Elaboração própria (2025).

Em comparação com trabalhos anteriores, como os de Dang, Song e Guo (2010) e Morris (2023), o presente modelo apresentou desempenho consistente mesmo em condições de variabilidade temporal e espacial. Enquanto os métodos tradicionais baseados em busca (*search-based*) tendem a falhar

diante de alterações repentinas no ambiente, o Q-Learning demonstrou flexibilidade para adaptar sua política sem necessidade de reinicialização ou recalibração do modelo.

Fig. 2. Comparação de desempenho do agente em diferentes cenários de ambiente.



Fonte: Elaboração própria (2025).

De forma geral, os resultados confirmam a eficácia do Aprendizado por Reforço na resolução de problemas de navegação em labirintos dinâmicos. O agente foi capaz de desenvolver comportamentos autônomos e adaptativos, alcançando um equilíbrio entre eficiência e robustez, o que reforça o potencial dessa abordagem em aplicações reais de robótica, sistemas autônomos e jogos digitais.

V. Conclusão

O trabalho desenvolvido permitiu analisar a aplicação do algoritmo Q-Learning em ambientes de navegação com diferentes níveis de complexidade, incluindo cenários estáticos, com NPCs móveis e com portas dinâmicas. Os resultados demonstraram que o agente foi capaz de aprender trajetórias eficientes, ajustando suas ações de acordo com as recompensas acumuladas ao longo do treinamento. Em ambientes dinâmicos, observou-se que o desempenho tende a ser impactado pela maior imprevisibilidade, reforçando a necessidade de estratégias mais robustas para lidar com situações de alta variabilidade.

De modo geral, os objetivos inicialmente propostos foram alcançados. O ambiente foi adequadamente modelado em Python, incorporando elementos fixos e dinâmicos; o agente aprendeu a minimizar a quantidade de passos até a saída; e foi possível comparar seu desempenho entre diferentes cenários de teste. As análises evidenciaram o potencial do Q-Learning para problemas de navegação, ao mesmo tempo em que apontaram limitações inerentes ao uso de representações tabulares em contextos mais amplos e mutáveis.

Apesar dos resultados satisfatórios, algumas limitações se destacam. O grid utilizado possui dimensão reduzida, o que simplifica a complexidade do problema. A abordagem tabular do Q-Learning também impõe restrições quando o espaço de estados cresce, tornando o método menos eficiente em ambientes maiores. Além disso, o algoritmo apresenta dificuldade frente a mudanças rápidas no ambiente, especialmente quando NPCs ou portas alteram seu estado de forma abrupta. Para trabalhos futuros, recomenda-se explorar métodos baseados em redes neurais, como o Deep Q-Learning, ou abordagens híbridas que integrem informações percebidas com a política de decisão.

References

[1] RIS-ALA, Rafael. Fundamentos de Aprendizagem por Reforço. Rafael Ris-Ala, 2023.

[2] H. Dang, J. Song and Q. Guo, ”An Efficient Algorithm for Robot Maze-Solving,” 2010 Second International Conference on Intelligent Human-Machine Systems and Cybernetics, Nanjing, China, 2010, pp. 79-82, doi: 10.1109/IHMSC.2010.119.

[3] P. Rombouts and L. Weyten, ”Corrections to ”A comment on ’interstage gain proration technique for digital-domain multi-step adc calibration”,” in IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 46, no. 11, pp. 1449-1449, Nov. 1999, doi: 10.1109/TCSII.1999.803490.

[4] J. M. Morris, ”Snake in the Labyrinth: Scenes from the Machine’s Deep Q-Learning Experience,” 2023 IEEE International Conference on Big Data (BigData), Sorrento, Italy, 2023, pp. 4511-4511, doi: 10.1109/BigData59044.2023.10386726.

[5] S. Mishra and P. Bande, ”Maze Solving Algorithms for Micro Mouse,” 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems, Bali, Indonesia, 2008, pp. 86-93, doi: 10.1109/SITIS.2008.104.

[6] F. Zeng, C. Wang and S. S. Ge, ”A Survey on Visual Navigation for Artificial Agents With Deep Reinforcement Learning,” in IEEE Access, vol. 8, pp. 135426-135442, 2020, doi: 10.1109/ACCESS.2020.3011438.

[7] AshesBorn, “Q-maze,” GitHub repository, Available: https://github.com/AshesBorn/Q-maze. Accessed on: Nov. 24, 2025.