NEXT STEP AI

João Paulo Brandão Ressoni[1], Lucca Soares Astrini[2], Emerson de Oliveira Batista[3]

[1]Centro Universitário das Faculdades Associadas de Ensino - Unifae, São João da Boa Vista – SP, Brasil (joao.ressoni@sou.fae.br)
[2,3] Centro Universitário das Faculdades Associadas de Ensino - Unifae, São João da Boa Vista, - SP, Brasil

Resumo: A expansão de centros comerciais amplia a complexidade de emergências e torna treinamentos reais custosos. O presente trabalho propõe agentes autônomos treinados por aprendizado por reforço no Unity ML-Agents, usando o método *Proximal Policy Optimization*, para adaptar comportamentos em cenários com múltiplas ameaças. Resultados preliminares mostram sucesso em ambientes simples, mas indicam que, para cenários complexos, os hiperparâmetros e as recompensas devem ser otimizados.

Palavras-chave: ML-Agents; Unity; Treinamento por Reforço; Agentes Inteligentes; Simulação.

INTRODUÇÃO

A cada dia construções são erguidas, incluindo centros comerciais e edificações de grande porte, o que, consequentemente, traz novas situações que ampliam as diversidades de situações de risco. O treinamento de técnicas de resgate por parte do corpo de bombeiros deve ser constante para que possam ser capazes de se adaptar a ambientes distintos, além disso a validação da estrutura para identificar se ela está de acordo com as normas e oferece segurança para os ocupantes deve sempre ser um ponto de preocupação.

Nesse sentido, táticas simuladas previamente sobre combate à ocorrência de incêndio podem aumentar as taxas de sucesso e diminuir a quantidade de vítimas. De acordo com Kolb (1984), ideias não são fixas e imutáveis, pelo contrário, elas são formuladas e reformuladas a partir da experiência.

Diante deste cenário, o uso de técnicas de aprendizado por reforço (RL - *Reinforcement Learning*), implementadas através do *framework* ML-Agents da Unity, permite desenvolver agentes autônomos capazes de aprender comportamentos adaptativos em ambientes simulados de emergência (Juliani et al., 2018). Diferentemente do treinamento de pessoas, trata-se de modelos de Inteligência Artificial (IA) capazes de aprender e lidar com cenários complexos, os quais podem ser posteriormente utilizados como ferramenta de apoio ao treinamento humano. Além disso, tais soluções permitem a criação de cenários diversos, inclusive aqueles que seriam inviáveis de reproduzir em condições reais, podendo testar estratégias de diferentes abordagens operacionais (rotas de evacuação, táticas de resgate, contenção de riscos etc.).

A abordagem escolhida para o desenvolvimento dos modelos foi o método de treinamento por reforço, pertencente ao campo de IA que se destaca por sua semelhança com o tipo de aprendizado que humanos e animais fazem durante suas vidas (Sutton e Barto, 2018).

Essa abordagem caracteriza-se por permitir que o agente aprenda ao interagir com o ambiente, desenvolvendo uma política de comportamento adaptativa que maximize a probabilidade de evacuação bem-sucedida. Conforme destacado por Sutton e Barto (2018, p.16), “Os problemas de aprendizagem por reforço envolvem aprender o que fazer — como mapear situações para ações — de forma a maximizar um sinal de recompensa numérica.”

Tal investigação é relevante não apenas pela aplicabilidade prática em treinamento de equipes de emergência, mas também pela contribuição teórica ao campo de IA aplicada a sistemas autônomos de segurança.

No entanto, ainda é necessária uma tecnologia capaz de simular tanto o ambiente, quanto o comportamento humano para que haja uma interação com o ambiente sem a intervenção do humano real, permitindo uma evolução das interações focadas na otimização no ato de realizar a tarefa designada. A

tecnologia disponibilizada pela Unity, ML-Agents, permite que tal façanha seja realizada, utilizando o treinamento por reforço.

O kit de ferramentas Unity ML-Agents, é uma ferramenta *open source* disponibilizada pela Unity, ela permite que os desenvolvedores criem modelos de IA capazes de interagir com o ambiente em que está inserido, coletar observações e aprender com tais interações, sendo um excelente candidato para lidar com o ambiente 3D e sua complexidade (Juliani et al., 2018). A Figura 1 ilustra o funcionamento do ML-Agent.

Figura 1 - Funcionamento do ML-Agents

Já a *engine* Unity, se consolidou no mercado por ser uma plataforma de desenvolvimento de jogos robusta e amplamente usada pelo mercado em jogos 3D, devido aos recursos gráficos, de física e possível expansão, compartilhamento de projetos e funcionalidades através de scripts. Com o kit de ferramentas Unity ML-Agents, a *engine* Unity se tornou um grande aliado em quesitos de pesquisas acadêmicas e projetos industriais.

Utilizando a *engine* Unity 6 e o kit de ferramentas Unity ML-Agents (versão 2.0.2), é possível simular ambientes dinâmicos e que, como resultado, encontre abordagens para lidar com ambientes de risco dinâmicos.

O ML-Agents utiliza como base o Processo de Decisão de Markov (MDP - *Markov Decision Process*), um método que se baseia em estados e que em determinados momentos pode ser interferido (períodos de decisão) para que ações sejam tomadas, cada ação possui uma recompensa positiva ou negativa. O efeito de uma ação (recompensa) leva em consideração a ação e o estado do sistema, desconsiderando o processo de chegar aonde o agente está (Pellegrini e Wainer, 2007).

Portanto, este trabalho objetiva simular ambientes de risco, como incêndios, e desenvolver agentes inteligentes que sejam capazes de encontrar e otimizar rotas de evacuação e de resgate em tais ambientes com a finalidade de reduzir as vítimas de incêndios e ajudar no treinamento constante de agentes reais de resgate, como bombeiros.

MATERIAL E MÉTODOS

A metodologia se fundamenta na utilização do framework Unity ML-Agents v. 2.0.2 integrado ao motor gráfico Unity 6, empregando o algoritmo *Proximal Policy Optimization (PPO)* para treinamento de agentes. A modelagem do problema se baseia em um Processo de Decisão de Markov (MDP), onde o agente interage com o ambiente através de um ciclo de observação, ação e recompensa.

Este treinamento atualiza a política iterativamente, baseando-se nas decisões e trajetórias adotadas pelo agente ao longo do episódio. O método utiliza a nova política (gerada no episódio atual) e compara com a política anterior, utilizando uma razão de probabilidade para controlar o aprendizado sem que ele saia de uma faixa segura. Isso é feito através de um gradiente otimizado chamado de *clipping* (Schulman et al., 2017) demonstrado na Equação 1.

$$L^{CLIP}(\theta) = \hat{E}_t\left[\min\left(r_t(\theta)\widehat{A_t}, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A_t}\right)\right] \quad (1)$$

A Figura 2 apresenta a ligação entre o Python e o framework Unity.

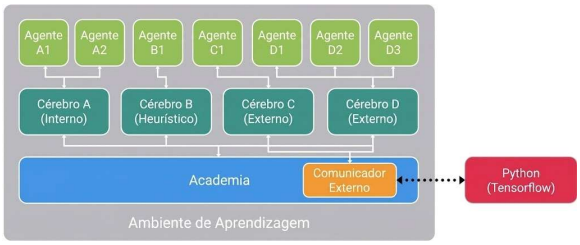

Figura 2 - Arquitetura do ML-Agents

O treinamento foi estruturado de forma incremental em quatro ambientes de complexidade crescente:

1. Cenário I: plano livre com objetivo posicionado aleatoriamente, como apresentado na Figura 3.
2. Cenário II: inclusão de obstáculos estáticos centrais, como exposto na Figura 4.
3. Cenário III: labirinto com paredes complexas para testar a navegação em espaços confinados, o qual pode ser visto na Figura 5.
4. Cenário IV: ambiente dinâmico com obstáculos móveis representando focos de incêndio.

A função de recompensa foi projetada para incentivar a eficiência da rota e penalizar colisões. Foram aplicadas penalidades de –0.2 para colisões iniciais e –0.003 para colisões contínuas, além de um coeficiente de penalidade temporal variando entre –0.005 a –0.02 por *time step*, visando minimizar o tempo e evacuação. Os hiperparâmetros críticos, como *learning rate* e *entropy regularization* (beta),

foram ajustados conforme detalhado na Tabela 1, visando estabilizar a convergência política.

Tabela 1. Hiperparâmetros

Parâmetro	**Valor antigo**	**Valor atual**
batch_size	10	10
buffer_size	100	100
learning_rate	3.0e-4	0.003
beta	5.0e-4	0.005
epsilon	0.2	0.2
lambda	0.99	0.99
num_epoch	3	3

RESULTADOS E DISCUSSÃO

Nos cenários estáticos (I, II e III), os agentes demonstraram uma convergência satisfatória, aprendendo políticas de navegação que priorizam a distância euclidiana até o objetivo e a preservação da integridade (evitar obstáculos). Observou-se que o agente desenvolveu a capacidade de contornar estruturas complexas sem a necessidade de um mapeamento prévio por meio de NavMesh.

Assim, especificamente no cenário I, mostrado na Figura 3, o agente foi introduzido a um plano com paredes e um objetivo que no início de todo episódio tem sua posição randomizada. Neste caso, o agente faz observações sobre sua própria posição e a do objetivo. Foi possível observar que ele colidiu algumas vezes nas paredes, porém com essa penalidade, ele logo aprendeu a evitá-las.

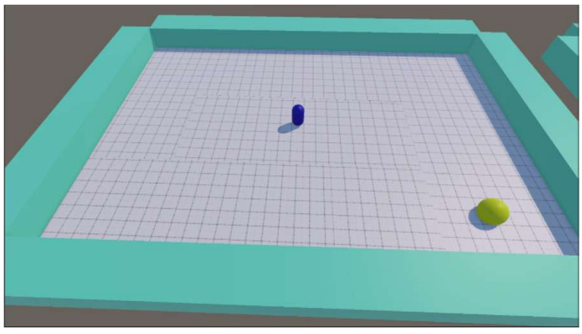

Figura 3 - Primeiro Ambiente

Já no cenário II, apresentado na Figura 4, o objetivo foi ensinar o agente a esquivar de obstáculos estáticos em seu caminho.

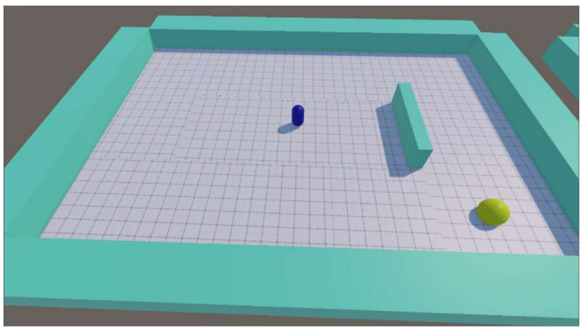

Figura 4 – Segundo Ambiente

Neste teste, o agente inicialmente teve dificuldades para entender que precisava subir ou descer um pouco e depois seguir para o objetivo, porém no processo de evolução do treinamento, ele foi desenvolvendo este entendimento e corrigindo sua rota na busca do objetivo.

No entanto, para este cenário um comportamento foi identificado, quando o objetivo ficava colado ou atrás do obstáculo, o agente não conseguia dar a volta e chegar até o local designado. Isso pode ser explicado pela recompensa negativa, ou seja punição, que o agente recebia ao se afastar do objetivo, o que levou a necessidade de ajustes nos hiperparâmetros apresentados na Tabela 1.

No cenário III foram adicionadas mais paredes para que o ambiente se tornasse mais complexo como na Figura 5.

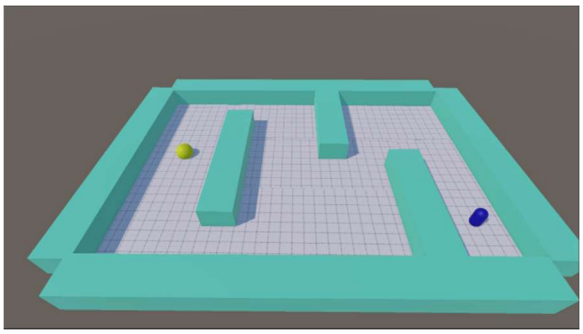

Figura 5 - Terceiro Ambiente

Apesar do agente ter aprendido a desviar de uma parede no teste anterior, neste o agente conseguiu desviar, porém foi encontrado um comportamento inesperado, isto porque, em determinadas posições que o objetivo estava, o agente entendia que desviar das paredes e fazer o caminho até o objetivo era custoso demais, pois em determinados momentos ele tinha que descer e subir de novo, o que fazia ele perder pontos.

Diante disso, foi identificada a ocorrência de *reward hacking*. Em ambientes com múltiplos riscos móveis, os agentes passaram a explorar lacunas na função de recompensa, priorizando a obtenção de bônus intermediários de exploração em detrimento da

conclusão do objetivo principal (evacuação). Esse comportamento subótimo indica que a função de recompensa atual necessidade um *reward shaping* mais refinado para equilibrar a exploração (*exploration*) e a explotação (*exploitation*) em ambientes de alta incerteza.

Com isso o tamanho dos episódios aumentou e a recompensa acumulada diminuiu em comparação ao treinamento anterior como visto na Figura 6. A linha laranja demonstra os treinamentos anteriores e a linha cinza o treinamento no terceiro ambiente.

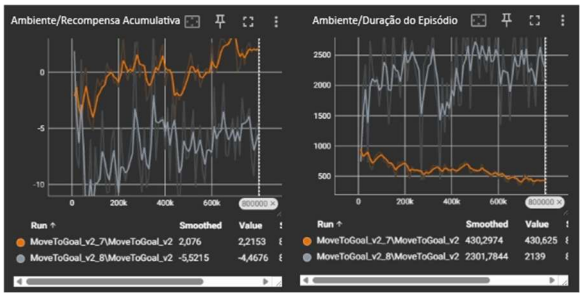

Figura 6 - Comparação de treinamentos

O cenário IV, tem como objetivo ensinar o agente a continuar desviando de obstáculos estáticos, porém agora introduzindo um obstáculo dinâmico, o fogo.

Representado por caixas vermelhas para facilitar a visualização, o fogo possui uma área que ele pode se espalhar em pontos aleatoriamente, dentro desta área o agente precisa esquivar dos focos de incêndio e depois desviar de uma parede para alcançar o objetivo.

No entanto, o agente aprendeu que desviar de tantos obstáculos fazia com que ele perdesse muitos pontos, o que não compensava tentar atingir o objetivo, com isso, ele decidiu fazer apenas o caminho de aproximação do objetivo como demonstrado na Figura 7. Logo em seguida ele ficava andando para frente e para trás, ou seja, abusando das recompensas de aproximação e afastamento do objetivo.

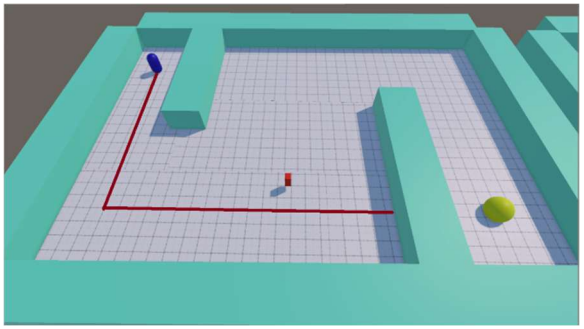

Figura 7 - Caminho do agente no quarto ambiente

Como próximos passos após os resultados até o momento, serão removidas as recompensas, positiva e negativa, de aproximação e afastamento do agente em relação ao objetivo, esta abordagem permite o agente ter uma exploração maior do ambiente sem uma penalização tão grande. Além disso, mais ajustes nos hiperparâmetros serão feitos para que o treinamento se torne mais efetivo e comportamentos indesejados sejam mitigados e até, possivelmente, eliminados.

CONCLUSÃO

Este estudo demonstrou a viabilidade do uso de Aprendizagem por Reforço para o desenvolvimento de sistemas de evacuação adaptativos. A transição de algoritmos determinísticos para modelos baseados em PPO permite que os agentes respondam a ameaças dinâmicas de forma autônoma.

Com os resultados apresentados, pôde ser percebido que ajustes nos hiperparâmetros são necessários para que o agente consiga se adaptar melhor a novos ambientes, ajustes nas recompensas *(reward shaping)* podem ser necessários caso os hiperparâmetros não sejam suficientes para corrigir o comportamento de *Reward Hacking* do agente.

Além disso, após o agente conseguir se adaptar ao quarto ambiente com a mecânica de fogo, é necessário que ambientes mais reais, como um modelo de uma casa, sejam utilizados para o treinamento do agente visando ter um comportamento mais real para que seja atingido o objetivo deste projeto que é simular o comportamento de vítimas e agentes de resgate em ambientes que o ser humano poderia estar inserido.

Embora os resultados em ambientes estáticos sejam promissores, o fenômeno de *reward hacking* observado nos testes dinâmicos destaca a complexidade de modelar funções de recompensa com cenários de risco. Os planos futuros para o projeto têm foco na implementação de modelos de habitações realistas e na comparação quantitativa de desempenho entre a política de treinada e os algoritmos clássicos de *pathfinding*.

REFERÊNCIAS

JULIANI, Arthur. et al. Unity: A General Platform for Intelligent Agents. arXiv e-prints, 2018. Disponível em https://arxiv.org/pdf/1809.02627. Acesso em: 17 abr. 2026.

KOLB, David A. Experiential Learning: Experience as the Source of Learning and Development. Englewood Cliffs, NJ: Prentice Hall. 1984. p. 22. Disponível em https://www.researchgate.net/publication/235701029_Experiential_Learning_Experience_As_The_Source_Of_Learning_And_Development. Acesso em 27, Mar. 2026.

PELLEGRINI, J; WAINER, J. Processos de Decisão de Markov: um tutorial. Revista De Informática Teórica E Aplicada, v. 14, n. 2, p. 134, 2007. Disponível em: https://doi.org/10.22456/2175-2745.5694. Acesso em: 12 mai. 2026.

SCHULMAN, J. et al. Proximal Policy Optimization Algorithms. 2017. Disponível em https://arxiv.org/abs/1707.06347. Acesso em 18 abr. de 2026.

SUTTON, R. S; BARTON, A. G. Reinforcement learning: An introduction. 2nd ed. The MIT Press. 2018. 16. Disponível em https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf. Acesso em 23, Mar. 2026.