DISSERTAÇÃO DE MESTRADO PROFISSIONAL

EVACUAI: PLANEJAMENTO DE ROTAS DE FUGA ADAPTATIVAS EM AMBIENTES COM OBSTÁCULOS UTILIZANDO APRENDIZADO POR REFORÇO

ANNA CAROLINA FERREIRA ROSA

Programa de Pós-Graduação Profissional em Engenharia Elétrica
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO PROFISSIONAL

EVACUAI: PLANEJAMENTO DE ROTAS DE FUGA ADAPTATIVAS EM AMBIENTES COM OBSTÁCULOS UTILIZANDO APRENDIZADO POR REFORÇO

ANNA CAROLINA FERREIRA ROSA

Dissertação de Mestrado Profissional submetida ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica

Banca Examinadora

Prof Dr. Fábio Lúcio Lopes de Mendonça, ENE/UnB _____
Orientador/Presidente

Prof Dr. Georges Daniel Amvame Nze, PPEE/E-NE/UnB _____
Examinador Interno

Prof Dr. Honório Assis Filho Crispim, CBMDF _____
Examinador Externo

Prof Dr. Daniel Alves da Silva, PPEE/UnB _____
Suplente

FICHA CATALOGRÁFICA

ROSA, ANNA CAROLINA
EVACUAI: PLANEJAMENTO DE ROTAS DE FUGA ADAPTATIVAS EM AMBIENTES COM OBSTÁCULOS UTILIZANDO APRENDIZADO POR REFORÇO [Distrito Federal] 2025.
xvi, 112 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2025).
Dissertação de Mestrado Profissional - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Aprendizado por Reforço	2. Rotas de fuga
3. FUGASegurança	4. Aprendizado de Máquina
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

ROSA, A.C. (2025). *EVACUAI: PLANEJAMENTO DE ROTAS DE FUGA ADAPTATIVAS EM AMBIENTES COM OBSTÁCULOS UTILIZANDO APRENDIZADO POR REFORÇO*. Dissertação de Mestrado Profissional, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 112 p.

CESSÃO DE DIREITOS

AUTOR: ANNA CAROLINA FERREIRA ROSA
TÍTULO: EVACUAI: PLANEJAMENTO DE ROTAS DE FUGA ADAPTATIVAS EM AMBIENTES COM OBSTÁCULOS UTILIZANDO APRENDIZADO POR REFORÇO
GRAU: Mestre em Engenharia Elétrica ANO: 2025

É concedida à Universidade de Brasília permissão para reproduzir cópias desta Dissertação de Mestrado e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. Do mesmo modo, a Universidade de Brasília tem permissão para divulgar este documento em biblioteca virtual, em formato que permita o acesso via redes de comunicação e a reprodução de cópias, desde que protegida a integridade do conteúdo dessas cópias e proibido o acesso a partes isoladas desse conteúdo. O autor reserva outros direitos de publicação e nenhuma parte deste documento pode ser reproduzida sem a autorização por escrito do autor.

Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

DEDICATÓRIA

Dedico este trabalho à memória do meu pai, que partiu ao final desta jornada. Agradeço por todo o carinho que sempre demonstrou por mim, por todo o apoio, incentivo e, especialmente, por todas as vezes em que me levou pessoalmente à universidade, permitindo que eu pudesse estudar e chegar até aqui. Onde quer que o senhor esteja, espero que sinta orgulho do que consegui realizar. Que este trabalho reflita a força que o senhor sempre me deu. No fim, tudo o que fiz foi por você.

AGRADECIMENTOS

Agradeço primeiramente a Deus, que me guiou e sustentou durante toda essa jornada acadêmica e pessoal. Sem Sua presença, nada disso seria possível.

À minha família, que sempre esteve ao meu lado, oferecendo todo o amor, apoio e compreensão. Aos meus pais, aos meus avós, aos meus irmãos e ao Lucas, companheiro de tantas etapas e desafios, agradeço profundamente por fazerem parte da minha vida e por me motivarem a seguir em frente.

Agradeço, ao meu orientador, Professor Dr. Fábio Lúcio Lopes de Mendonça, pela orientação profissional e atenciosa nos momentos mais desafiadores deste trabalho, sempre demonstrando paciência diante das dúvidas e dificuldades surgidas ao longo da elaboração desta dissertação.

Agradeço o apoio técnico e computacional do Laboratório de Tecnologias para Tomada de Decisão - LATITUDE, da Universidade de Brasília, que conta com apoio do CNPq - Conselho Nacional de Pesquisa (Outorgas 312180/2019-5 PQ-2 e 465741/2014-2 INCT em Cibersegurança), da Procuradoria Geral da Fazenda Nacional (Outorga PGFN 23106.148934/2019-67), ao Mestrado Profissional em Engenharia Elétrica, na área de concentração: Segurança Cibernética ao Decanatos de Pesquisa e Inovação projeto (Outorga 7129 FUB/EMENDA/DPI/COPEI/AMORIS) e do Projeto SISTER City –Sistemas Inteligentes Seguros e em Tempo Efetivo Real para Cidades Inteligentes (Outorga 625/2022) e a Fundação de Apoio a Pesquisa do Distrito Federal - FAP/DF.

Sou grata à Universidade de Brasília, instituição que proporcionou as condições e o ambiente necessários para o meu desenvolvimento acadêmico. Em especial, agradeço ao Laboratório UIoT, que tem sido um espaço de aprendizado, colaboração e crescimento.

Agradeço a todos os meus colegas cientistas de dados, com quem compartilho o dia a dia de trabalho e aprendizado. A troca constante de experiências tem sido essencial para o meu crescimento profissional e pessoal.

E por fim, a todos que, direta ou indiretamente, contribuíram para a realização deste trabalho, deixo meu sincero agradecimento.

RESUMO

Incêndios ocorrem diariamente em diversas partes do mundo, tanto em ambientes internos quanto externos. Em ambientes fechados, o fogo pode se espalhar rapidamente, o que torna essencial o conhecimento das rotas de fuga existentes, em especial nos edifícios comerciais com grande circulação de pessoas.

Diversos algoritmos podem ser utilizados para a identificação de rotas de fuga, desde os mais clássicos, como *Dijkstra*, A* e *Bellman-Ford*, até abordagens mais avançadas baseadas em redes neurais. No entanto, quando se busca a otimização dessas rotas considerando múltiplos critérios, o caminho mais curto nem sempre representa a melhor opção. Obstáculos como áreas de risco (como focos de incêndio) e congestionamentos de pessoas podem alterar significativamente a rota mais segura de evacuação. Nesse contexto, o Aprendizado por Reforço surge como uma abordagem promissora, pois permite ao agente aprender a tomar decisões adaptativas com base no ambiente e nas condições apresentadas.

Esse trabalho apresenta uma evolução da abordagem proposta em [1], baseando-se na utilização de Aprendizado por Reforço para auxiliar na identificação de rotas de fuga seguras em casos de incêndio, uma técnica que utiliza um sistema de recompensas positivas para incentivar comportamentos desejados e punições para desencorajar comportamentos indesejados. A solução aqui proposta pretende superar limitações previamente identificadas, como a dificuldade de lidar com grafos de grande escala e a necessidade de retreinamento do agente para cada nova configuração de obstáculos. Os experimentos demonstram que o sistema proposto é capaz de lidar com plantas baixas e grafos significativamente maiores, com desempenho consistente, sendo uma solução robusta e possivelmente aplicável a cenários reais de evacuação.

Palavras-chave: Aprendizado por Reforço, Rotas de fuga, Segurança, Aprendizado de Máquina.

ABSTRACT

Fires occur daily in various parts of the world, both indoors and outdoors. In enclosed spaces, fire can spread rapidly, making it essential to be aware of existing escape routes, especially in commercial buildings with high foot traffic.

Several algorithms can be used to identify escape routes, ranging from classical ones like *Dijkstra*, A*, and *Bellman-Ford*, to more advanced approaches based on neural networks. However, when aiming to optimize these routes while considering multiple criteria, the shortest path is not always the best option. Obstacles such as hazardous areas (e.g., fire hotspots) and crowd congestion can significantly impact the safest evacuation route. In this context, Reinforcement Learning stands out as a promising approach, as it enables an agent to learn how to make adaptive decisions based on the environment and the conditions presented.

This work presents an evolution of the approach proposed in [1], building on the use of Reinforcement

Learning to assist in identifying safe escape routes in the event of a fire. This technique leverages a system of positive rewards to encourage desired behaviors and penalties to discourage undesired ones. The solution proposed here aims to overcome previously identified limitations, such as the difficulty in handling large-scale graphs and the need to retrain the agent for every new obstacle configuration. The experiments show that the proposed system is capable of handling significantly larger floor plans and graphs, with consistent performance, making it a robust solution potentially applicable to real-world evacuation scenarios.

Keywords: Reinforcement Learning, Escape Routes, Safety, Machine Learning.

SUMÁRIO

1 INTRODUÇÃO 1
1.1 Motivação 2
1.2 Objetivos do Trabalho 2
1.2.1 Objetivo Geral 2
1.2.2 Objetivo Específico 2
1.3 Trabalhos Publicados 3
1.4 Metodologia 3
1.5 Organização do Trabalho 4

2 FUNDAMENTAÇÃO TEÓRICA 5
2.1 Inteligência Artificial - IA 5
2.2 Aprendizado de Máquina - *Machine Learning* 5
2.3 Aprendizado por Reforço (AR) 6
2.3.1 Exploração x Explotação 7
2.3.2 Elementos do Aprendizado por Reforço 8
2.3.3 A interface de agente-ambiente 8
2.3.4 Objetivo e Recompensas 9
2.3.5 Métodos de valor de ação - *Action-Values Method* 10
2.3.6 Políticas e Funções de Valor 11
2.3.7 Algoritmo *Q-Learning* 12
2.3.8 *Deep Q Learning* - Aprendizado Q Profundo 13
2.4 Saídas de Emergência e Rotas de Fuga 16
2.5 Planta Baixa 17
2.6 Teoria de Grafos 18
2.6.1 Grafos 19
2.6.2 Definições 20
2.6.3 Grafos Direcionados 20
2.6.4 Grafos Rotulados 21

3 Trabalhos Relacionados 22
3.1 Planejamento de Rotas de Fuga 22
3.2 Representação do Ambiente de Aprendizagem 24

4 ARQUITETURA DA PROPOSTA 27
4.1 Arquitetura Geral 28
4.1.1 Camada de Apresentação 28
4.1.2 Camada de Aplicação 35
4.1.3 Camada de Persistência 38

4.1.4 Camada de Inteligência 41
4.2 Modelagem do Ambiente de Aprendizagem 42
4.2.1 Definição dos estados e ações 44
4.2.2 Função de Recompensa 46
4.2.3 Arquitetura da Rede Neural (*Q-Network*) 49
4.2.4 Atualização da *Q-Network* e *Q-Network* Alvo 50
4.2.5 Obstáculos 52

5 RESULTADOS 55
5.1 Ambiente Computacional 55
5.2 Experimento 1 55
5.2.1 Configurações do Experimento 55
5.2.2 treinamento 58
5.2.3 Visualização das Rotas Aprendidas 59
5.3 Experimento 2 73
5.3.1 Configurações do Experimento 73
5.3.2 treinamento 76
5.3.3 Visualização das Rotas Aprendidas 77
5.4 Avaliação da Topologia de Saídas nos Cenários de Evacuação 90
5.5 Comparação com Trabalhos Relacionados 91

6 CONCLUSÃO 95

REFERÊNCIAS BIBLIOGRÁFICAS 97

Apêndices 102
I.1 Caminhos Calculados até as saídas sem obstáculos 102
I.2 Códigos Fonte da API 106
I.2.1 Rota `/create` 106
I.2.2 Rota `/train` 107
I.2.3 Rota `/inference` 108
I.2.4 Rota `/projects` 109
I.2.5 Rota `/train_ids/{project_id}` 109
I.3 Códigos da Camada de Inteligência 110
I.3.1 Servidor Grpc 110
I.3.2 Rede Neural para Seleção de Ações 111

LISTA DE FIGURAS

2.1 Interface Agente-Ambiente do Aprendizado por Reforço. 9
2.2 Planta baixa do térreo do Laboratório SG11/UnB. 18
2.3 Esquema das sete pontes de Königsberg [2]. 18
2.4 Exemplo do desenho de um grafo. 19

4.1 EvacuAI - Arquitetura Proposta. 28
4.2 Página inicial do front-end. 29
4.3 Página para o usuário desenhar o grafo referente a planta baixa. 30
4.4 Página para iniciar o treinamento de um agente de aprendizagem. 31
4.5 Página para escolha do projeto e identificador de treinamento na inferência. 33
4.6 Página para a configuração do ambiente de inferência. 34
4.7 Página para visualização da inferência conforme a configuração escolhida. 35
4.8 Documentação da API desenvolvida. 36
4.9 Documentos no MongoDB Compass. 40
4.10 Estrutura de arquivos no *object storage.* 41

5.1 Grafo e planta baixa do laboratório SG11 da Universidade de Brasília (UnB). O arquivo original, em alta resolução, está disponível em [3]. 56
5.2 Grafo correspondente à planta baixa do laboratório SG11 da Universidade de Brasília (UnB). 57
5.3 Experimento 1 - Relação entre as taxas de exploração e explotação ao longo do treinamento. 59
5.4 Experimento 1 - Resultados referente a quantidade de passos necessários para concluir um episódio. 60
5.5 Experimento 1 - Resultados referente a média das recompensas. 61
5.6 Experimento 1 - Resultados referente a variância das recompensa. 62
5.8 Experimento 1 - Resultado do agente em um ambiente sem obstáculos. 62
5.7 Experimento 1 - Resultados referente a perda das recompensas. 63
5.9 Experimento 1 - Resultado do agente ao enfrentar congestionamento nos nós 3 e 17. 65
5.10 Experimento 1 - Resultado do agente ao enfrentar um ambiente com fogo no nó 25. 67
5.11 Experimento 1 - Resultado do agente ao enfrentar um ambiente com fogo nó 49. 69
5.12 Experimento 1 - Resultado do agente ao enfrentar um ambiente com fogo no nó 25 e congestionamento no nó 47. 72
5.13 Experimento 1 - Resultado do agente ao enfrentar um ambiente com fogo no nó 49 e congestionamento no nó 41. 72
5.14 Grafo e planta baixa dos dois pisos do laboratório SG11 da Universidade de Brasília (UnB). O arquivo original, em alta resolução, está disponível em: [4]. 74
5.15 Grafo correspondente à planta baixa dos dois pisos do laboratório SG11 da Universidade de Brasília (UnB). 75
5.16 Relação entre as taxas de exploração e explotação ao longo do treinamento no Experimento 1. 76

5.17 Resultados referente a quantidade de passos necessários para concluir um episódio no Experimento 2. 77
5.18 Resultados referente a média das recompensas do Experimento 2. 78
5.19 Resultados referente a variância das recompensas do Experimento 2. 79
5.20 Resultados referente a perda das recompensas do Experimento 2. 80
5.21 Experimento 2 - Resultado do agente em um ambiente sem obstáculos. 81
5.22 Experimento 2 - Resultado do agente ao enfrentar congestionamento nos nós 64 e 91. 84
5.23 Resultado do Experimento 2 - Resultado do agente ao enfrentar um ambiente com fogo no nó 64. 87
5.24 Resultado do Experimento 2 - Resultado do agente ao enfrentar um ambiente com fogo no nó 64 e congestionamento no nó 50. 90
5.25 Resultados referente a média das recompensas do Experimento 1 utilizando o algoritmo do trabalho [1]. 93
5.26 Resultados de perda do Experimento 1 utilizando o algoritmo do trabalho [1]. 94

LISTA DE TABELAS

3.1 Comparação entre trabalhos relacionados e o presente estudo 26

4.1 Arquitetura da Rede Neural proposta. 49

5.1 Hiperparâmetros utilizados no Experimento 1. 58
5.2 Experimento 1 - Caminhos percorridos pelo agente em um ambiente sem obstáculos, com as respectivas recompensas (Subseção 4.2.2) e distâncias aproximadas (Subseção 4.1.2.1). .. 60
5.3 Experimento 1 - Recompensas e distâncias ao forçar o agente a repetir os mesmos caminhos utilizados no ambiente sem congestionamento, agora com congestionamento nos nós 3 e 47. 64
5.4 Experimento 1 - Recompensas e distâncias dos caminhos efetivamente escolhidos pelo agente ao enfrentar o ambiente com congestionamento nos nós 3 e 47. 64
5.5 Experimento 1 - Recompensas e distâncias ao forçar o agente a repetir os mesmos caminhos utilizados no ambiente sem nós em chamas, agora com fogo no nó 25. 66
5.6 Experimento 1 - Recompensas e distâncias dos caminhos efetivamente escolhidos pelo agente ao enfrentar o ambiente com fogo no nó 25. 66
5.7 Experimento 1 - Recompensas e distâncias ao forçar o agente a repetir os mesmos caminhos utilizados no ambiente sem nós em chamas, agora com fogo no nó 49. 68
5.8 Experimento 1 - Recompensas e distâncias dos caminhos efetivamente escolhidos pelo agente ao enfrentar o ambiente com fogo no nó 49. 68
5.9 Experimento 1 - Recompensas e distâncias ao forçar o agente a repetir os mesmos caminhos utilizados no ambiente sem obstáculos, agora com fogo no nó 25 e congestionamento no nó 47. 70
5.10 Experimento 1 - Recompensas e distâncias dos caminhos efetivamente escolhidos pelo agente ao enfrentar o ambiente com fogo no nó 25 e congestionamento no nó 47. 70
5.11 Experimento 1 - Recompensas e distâncias ao forçar o agente a repetir os mesmos caminhos utilizados no ambiente sem obstáculos, agora com fogo no nó 49 e congestionamento no nó 41. 70
5.12 Recompensas e distâncias dos caminhos efetivamente escolhidos pelo agente ao enfrentar o ambiente com fogo no nó 49 e congestionamento no nó 41. 71
5.13 Hiperparâmetros utilizados no Experimento 2. 73
5.14 Experimento 2 - Caminhos percorridos pelo agente em um ambiente sem obstáculos, com as respectivas recompensas (Subseção 4.2.2) e distâncias aproximadas (Subseção 4.1.2.1). .. 78
5.15 Experimento 2 - Recompensas e distâncias ao forçar o agente a repetir os mesmos caminhos utilizados no ambiente sem congestionamento, agora com congestionamento nos nós 64 e 91. 82
5.16 Experimento 2 - Recompensas e distâncias dos caminhos efetivamente escolhidos pelo agente ao enfrentar o ambiente com congestionamento nos nós 64 e 91..... 82
5.17 Experimento 2 - Recompensas e distâncias ao forçar o agente a repetir os mesmos caminhos utilizados no ambiente sem presença de fogo, agora com fogo no nó 64 85

5.18 Experimento 2 - Recompensas e distâncias dos caminhos efetivamente escolhidos pelo agente ao enfrentar o ambiente presença de fogo no nó 64 86
5.19 Recompensas e distâncias dos caminhos efetivamente escolhidos pelo agente do experimento 2 ao enfrentar o ambiente com presença de fogo no nó 64 e congestionamento no nó 50..... 88
5.20 Experimento 2 - Recompensas e distâncias ao forçar o agente a repetir os mesmos caminhos utilizados no ambiente do cenário 3, mas com fogo no nó 64 e congestionamento no nó 50. . 89
5.21 Hiperparâmetros utilizados conforme o trabalho [1]..... 92

1 Caminhos do Experimento 1..... 103
2 Caminhos do Experimento 2..... 106

1 INTRODUÇÃO

Incêndios ocorrem diariamente em todo o mundo, tanto em ambientes externos, como florestas e matas, quanto em ambientes internos, como residências, edifícios comerciais e industriais. Em ambientes fechados, o fogo pode se espalhar rapidamente, influenciado por diversos fatores, como características da construção e os materiais presentes no ambiente. Em edifícios comerciais com alta circulação de pessoas, é essencialmente importante conhecer as rotas de fuga existentes. Assim, em caso de incêndio ou outras emergências, todos poderão deixar o local de forma rápida e segura.

Em 2024, o Instituto Sprinkler Brasil, por meio de um monitoramento diário de incêndios estruturais no país, registrou 2.453 ocorrências desse tipo, atingindo o maior valor histórico desde 2017 [5]. Esse dado representa um aumento de 10,4% em relação a 2023, quando foram contabilizados 2.222 casos.

No Brasil, o Comitê Brasileiro de Segurança Contra Incêndio da Associação Brasileira de Normas Técnicas (CB-024/ABNT) pretende padronizar diversos aspectos relacionados à segurança contra incêndios. Suas atribuições abrangem desde a regulamentação da fabricação de produtos e equipamentos, até o estabelecimento de normas para a formação de profissionais especializados na área [6].

A norma que regulamenta as saídas de emergência no Brasil é a ABNT NBR 9077 [7]. Segundo essa norma, uma saída de emergência é definida como "um caminho contínuo, devidamente protegido, proporcionado por portas, corredores, *halls*, passagens externas, balcões, vestíbulos, escadas, rampas ou outros dispositivos de saída, ou combinações destes, que deve ser percorrido pelo usuário em caso de incêndio, desde qualquer ponto da edificação até atingir a via pública ou um espaço aberto, protegido do incêndio e em comunicação com o logradouro".

Combater incêndios domésticos e comerciais e garantir rotas de fuga eficazes envolve um conjunto de ações planejadas, sistemas de segurança, treinamento e manutenção. Existem várias iniciativas para combater incêndios domésticos e comerciais, como prevenção ativa (ação humana e equipamentos) e a prevenção passiva (elementos construtivos). Entretanto, mesmo com esses mecanismos de prevenção, não se pode prever que esses tipos de acidentes possam ser evitados.

Neste sentido, quando acontecem acidentes desta natureza, é importante que os prédios empresariais, industriais, centros comerciais e shoppings possam ter suas devidas rotas de fuga conhecidas, sendo os caminhos seguros e desobstruídos que permitem a saída rápida de pessoas em caso de emergência, levando-as a um ponto seguro (área externa ou ponto de encontro). Elementos essenciais como sinalização luminosa e visível, indicações de saída de emergência, mesmo no escuro ou com fumaça, luzes automáticas que acendem em caso de queda de energia, mapas de evacuação que devem ser fixados em locais visíveis em ambientes comerciais e condomínios, são essenciais para uma evacuação eficiente. Ainda assim, é importante tomar novas medidas e certas iniciativas para se definir rotas de fuga, principalmente em prédios mais antigos.

Atualmente, é amplamente reconhecido que a Inteligência Artificial (IA) pode auxiliar em diversas tarefas, desde as mais simples até as mais complexas. No contexto de incêndios estruturais, a IA também pode desempenhar um papel crucial ao ajudar na definição de rotas de fuga. Com o conhecimento da planta

do edifício e o mapeamento de todas as possíveis saídas de emergência e caminhos disponíveis, a IA pode otimizar a evacuação, identificando as rotas mais seguras e eficientes em caso de emergência.

Diante disto, diversos algoritmos podem ser empregados para determinar rotas de fuga, variando desde os mais simples e amplamente aceitos, como *Dijkstra*, A* e *Bellman-Ford*, até técnicas mais complexas, como redes neurais. No entanto, quando se considera a otimização de rotas de fuga com múltiplos critérios, nem sempre o caminho mais curto de um indivíduo até a saída de emergência mais próxima vai ser o melhor caminho; é necessária a avaliação de outros critérios e obstáculos presentes no ambiente, e são nesses casos que Aprendizado por Reforço se destaca como uma abordagem particularmente interessante.

Assim, a proposta deste trabalho é utilizar mecanismos de IA para auxiliar na determinação mais eficaz de rotas de fuga. Entre esses mecanismos, temos o Aprendizado por Reforço [8] que é uma técnica de Aprendizado de Máquina que utiliza um sistema de recompensas positivas para incentivar comportamentos desejados e punições para desencorajar comportamentos indesejados. Nesta abordagem, um agente é capaz de interpretar seu ambiente e realizar ações, aprendendo por meio de tentativa e erro qual é seu objetivo, que é maximizar sua recompensa final. Embora frequentemente utilizado em cenários de jogos, o Aprendizado por Reforço também pode ser aplicado na busca de caminhos em grafos e matrizes.

1.1 MOTIVAÇÃO

Esta dissertação de mestrado dá continuidade à pesquisa iniciada em [1], que utilizou plantas baixas de edifícios escolares como grafos e matrizes adjacentes para treinar um agente de Aprendizado por Reforço na otimização de rotas de fuga em cenários simulados de incêndios estruturais. O objetivo desta pesquisa é aprimorar a arquitetura existente, introduzindo múltiplos critérios para a otimização das rotas de fuga, com o intuito de melhorar a eficácia das estratégias de evacuação em cenários simulados de incêndios estruturais.

1.2 OBJETIVOS DO TRABALHO

1.2.1 Objetivo Geral

O objetivo do trabalho é aprimorar a otimização de rotas de fuga em cenários de incêndios estruturais utilizando técnicas avançadas de Aprendizado por Reforço. A pesquisa se baseia na representação de plantas baixas de ambientes fechados como grafos e matrizes, o que possibilita uma análise e definição eficaz das rotas de evacuação por meio de um agente de Aprendizado por Reforço.

1.2.2 Objetivo Específico

O propósito é desenvolver uma abordagem que permita identificar rotas de fuga de maneira rápida e segura, integrando essa solução com as estratégias já existentes para o combate a incêndios. Com isso, busca-se aprimorar a eficácia das rotas de evacuação e contribuir para a segurança global em situações

de emergências, oferecendo uma ferramenta complementar aos sistemas tradicionais de segurança contra incêndios.

Para atingir o objetivo proposto, foram estabelecidos os seguintes objetivos específicos:

- Evoluir a arquitetura de Aprendizado por Reforço: aperfeiçoar o modelo de Aprendizado por Reforço profundo previamente desenvolvido, incorporando melhorias que aumentem sua capacidade de otimizar rotas de fuga.
- Introduzir múltiplos critérios na análise de rotas de fuga: ampliar a análise das rotas de evacuação para considerar outros critérios além do caminho mais curto.
- Avaliar a eficácia do modelo: testar e validar o modelo aprimorado em cenários simulados de incêndios estruturais para medir sua eficácia na melhoria da eficiência e segurança das rotas de evacuação.

1.3 TRABALHOS PUBLICADOS

Durante o estudo para a construção desta dissertação, o artigo [9] foi publicado com o título "EvacuAI: An Analysis of Escape Routes in Indoor Environments with the Aid of Reinforcement Learning".

1.4 METODOLOGIA

Esse trabalho propõe uma solução para otimizar as rotas de fuga em cenários de emergência. Ao se utilizar de Aprendizado por Reforço, o agente de aprendizado deve entender, por meio de tentativa e erro, a identificar as melhores rotas de fuga em ambientes dinâmicos e com obstáculos. Por tanto, essa pesquisa utilizará conceitos de Inteligência Artificial (IA) para otimização de rotas de fuga e segurança em situações de emergências.

Dessa forma, este trabalho pretende desenvolver e testar uma arquitetura baseada em Aprendizado por Reforço para otimizar e adaptar rotas de fuga em tempo real, levando em consideração a presença de obstáculos e variáveis em ambientes fechados, como edifícios comerciais e/ou residenciais. O principal objetivo do trabalho é analisar o impacto de diferentes tipos de obstáculos (focos de incêndio e pontos de congestionamento) no planejamento da rota mais segura. A proposta visa demonstrar a eficácia do modelo de AR em ambientes dinâmicos, resultando em uma solução capaz de se adaptar automaticamente às mudanças do ambiente e de orientar as vítimas de forma eficiente até saídas seguras.

Portanto, a metodologia proposta busca criar um sistema de planejamento de rotas de fuga em casos de incêndio, utilizando plantas baixas de edifícios fechados com Aprendizado por Reforço para otimizar as decisões em ambientes complexos e dinâmicos. A pesquisa trará contribuições tanto para o campo da segurança em emergências quanto para a aplicação de IA em situações práticas e de alto risco.

1.5 ORGANIZAÇÃO DO TRABALHO

Este trabalho está estruturado em seis capítulos, incluindo esta introdução.

O Capítulo 2 apresenta a fundamentação teórica, abordando os principais conceitos relacionados à IA, Aprendizado por Reforço, Teoria dos Grafos e representação de plantas baixas.

O Capítulo 3 descreve os trabalhos relacionados, com foco principal em abordagens voltadas para rotas de fuga e na construção de ambientes de aprendizado a partir de plantas baixas.

O Capítulo 4 descreve a arquitetura proposta, detalhando a modelagem do ambiente com base na planta baixa, bem como os processos de treinamento e inferência do agente de aprendizado.

O Capítulo 5 apresenta os resultados obtidos a partir de dois experimentos. O primeiro utiliza a planta baixa de um único piso, enquanto o segundo considera uma estrutura com dois andares.

Por fim, o Capítulo 6 traz as conclusões do trabalho, consolidando os resultados obtidos, validando a proposta apresentada e sugerindo trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os fundamentos necessários para a construção de um algoritmo de Aprendizado por Reforço (AR) no contexto do planejamento de rotas de fuga em ambientes internos. Os conceitos de AR apresentados nesta seção são amplamente baseados no trabalho [10].

2.1 INTELIGÊNCIA ARTIFICIAL - IA

A Inteligência Artificial (IA) é um campo da computação que envolve o desenvolvimento de algoritmos e tecnologias capazes de resolver tarefas de forma automática, processando dados e buscando simular a inteligência humana. Esse campo vem sendo estudado há várias décadas. Em 1943, Warren McCulloch e Walter Pitts publicaram um trabalho conhecido como o pioneiro da IA, no qual propuseram um modelo matemático de neurônio artificial, baseado em lógica proposicional, sendo essa a base para o desenvolvimento das redes neurais [11].

Em 1955, os cientistas Allen Newell e Herbert Simon, com a colaboração de J. C. Shaw, iniciaram o desenvolvimento do programa de computador conhecido como *Logic Theorist*, que só foi apresentado oficialmente em 1956 durante a Conferência de Dartmouth, evento considerado o marco inicial da IA como um campo de pesquisa independente. O *Logic Theorist* foi um dos primeiros programas de IA, capaz de provar teoremas matemáticos usando regras formais, demonstrando uma forma primitiva de raciocínio automatizado [12].

A partir da década de 1980, a pesquisa em IA passou por avanços significativos, principalmente pelo aumento do poder computacional e pelo desenvolvimento de novas técnicas, como o aprendizado supervisionado em redes neurais multicamadas, viabilizado pelo algoritmo de retropropagação (*backpropagation*) do erro, melhorado por Rumelhart, Hinton e Williams em 1986 [13].

Atualmente, a IA está presente em diversos aspectos da nossa vida, sendo aplicada em áreas como medicina, indústria, redes de computadores e *chatbots*, demonstrando como esse campo continua em constante evolução.

2.2 APRENDIZADO DE MÁQUINA - *MACHINE LEARNING*

O Aprendizado de Máquina (*Machine Learning*) é um ramo da IA voltado para a identificação de padrões em conjuntos de dados por meio de algoritmos computacionais. Segundo [14], o Aprendizado de Máquina é definido como "um processo em que computadores otimizam um critério de desempenho utilizando dados de exemplo ou experiências anteriores". Os modelos de aprendizado podem ser preditivos, quando fazem previsões futuras, ou descritivos, ao extrair conhecimento de dados existentes. Esse aprendizado geralmente se baseia na estatística para a construção de modelos matemáticos e na computação, que

desempenha um papel importante na eficiência dos algoritmos, tanto no treinamento quanto na inferência.

O Aprendizado de Máquina pode ser dividido em três principais tipos: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço [15].

No aprendizado supervisionado, os dados utilizados para treinar o modelo são previamente rotulados, fazendo com que os algoritmos identifiquem padrões e relações entre as entradas e as saídas esperadas. Durante o treinamento, o modelo ajusta seus parâmetros com base nesses exemplos rotulados, melhorando sua capacidade de realizar previsões ou classificações com novos dados de entrada. Esse tipo de aprendizado é principalmente aplicado em áreas como visão computacional, sendo fundamental para tarefas de classificação e detecção de objetos em imagens. Apesar de ser o tipo de Aprendizado de Máquina mais popular para tarefas complexas, a rotulagem de dados ainda depende do trabalho humano, e isso pode ser um fator limitante para a escalabilidade de projetos que utilizam aprendizado supervisionado [15].

No aprendizado não supervisionado, os dados não possuem rótulos prévios, então o modelo não recebe uma saída esperada. Em vez disso, o algoritmo analisa as entradas e busca identificar padrões, estruturas ocultas ou distribuições nos dados enviados sem qualquer supervisão explícita, ou seja, sem a interferência humana. Esse tipo de aprendizado é principalmente aplicado em técnicas de agrupamento (*clustering*), onde os dados são organizados em grupos com características semelhantes, e em redução de dimensionalidade, para simplificar a representação dos dados sem perder informações essenciais. Essas abordagens são frequentemente utilizadas em análises exploratórias, segmentação de clientes e compressão de dados [15].

No aprendizado por reforço, o agente aprende por tentativa e erro, sem rótulos explícitos, de maneira semelhante ao aprendizado não supervisionado. No entanto, nesse tipo de aprendizagem, um agente interage com o ambiente criado e recebe recompensas (positivas ou negativas) com base em suas ações. Com isso, o modelo ajusta sua estratégia de tomada de decisão para maximizar a recompensa total ao longo do tempo. Esse tipo de aprendizado é frequentemente utilizado em robótica, jogos e sistemas autônomos, onde o agente precisa tomar decisões sequenciais em um ambiente dinâmico para atingir um objetivo estabelecido [16].

2.3 APRENDIZADO POR REFORÇO (AR)

Desde o nascimento, os seres humanos estão continuamente aprendendo e interagindo com o ambiente que os cerca, desenvolvendo um entendimento sobre causa e efeito e entendendo as consequências de suas ações e maneiras de alcançar metas propostas. Esse aprendizado por interação ocorre de várias formas ao longo da vida, e é a partir dessa ideia que surge o campo do Aprendizado de Máquina conhecido como Aprendizado por Reforço (AR).

O Aprendizado por Reforço é baseado em tentativa e erro, onde o objetivo é aprender a maximizar recompensas acumuladas em uma tarefa específica. Durante o treinamento, o agente de aprendizagem não recebe instruções explícitas sobre a melhor ação a tomar, mas interage com o ambiente para descobrir quais ações levam à maximização das recompensas acumuladas. É importante observar que, ao escolher uma ação, o agente não está apenas influenciando sua recompensa imediata, mas também afetando suas recompensas futuras [16].

O Aprendizado por Reforço pode ser considerado pertencente à categoria de aprendizado não supervisionado. Embora grande parte das pesquisas ainda se concentre no aprendizado supervisionado, especialmente em tarefas de classificação e regressão, onde os dados de entrada são rotulados previamente, o aprendizado não supervisionado tem atraído cada vez mais interesse. Essa crescente atenção se deve à possibilidade desse algoritmo lidar com situações em que os dados não são ou não podem ser rotulados, permitindo que os modelos descubram padrões e tomem decisões com base em interações diretas com o ambiente, sem orientação humana explícita.

No aprendizado não supervisionado, a rotulação prévia dos dados de treinamento não é necessária, o objetivo está em explorar dados não rotulados por meio da interação com o ambiente de aprendizado e da experiência adquirida ao longo do tempo. No Aprendizado por Reforço, o agente de aprendizagem interage com o ambiente para entender padrões, dinâmicas e ajustar suas ações com base nessas interações. Embora nem todo aprendizado não supervisionado seja considerado Aprendizado por Reforço, a combinação entre o aprendizado de padrões por meio de interações e a maximização de recompensas é o que define o Aprendizado por Reforço.

2.3.1 Exploração x Explotação

Os conceitos de exploração (*exploration*) e explotação (*exploitation*) são centrais quando se trata de Aprendizado por Reforço. O objetivo principal do agente de aprendizagem é maximizar a recompensa acumulada ao longo do tempo, utilizando suas experiências passadas para definir suas decisões futuras. No entanto, existe um equilíbrio necessário entre essas duas políticas, pois a exploração incentiva o agente a escolher ações de forma aleatória para descobrir novas possibilidades no ambiente, enquanto a explotação faz com que o agente baseie suas escolhas nas experiências anteriores, priorizando ações que já demonstraram ser eficazes anteriormente [16].

Se o agente sempre optar por ações com base em experiências passadas, ele pode não explorar todas as possibilidades do ambiente, limitando seu aprendizado ao repetir ações já conhecidas; por isso, a exploração é crucial, pois ela garante que o agente descubra novas ações e aprenda sobre o ambiente da forma mais completa possível, permitindo que ele identifique oportunidades que poderiam ser ignoradas se apenas se baseasse em decisões anteriores. A política de exploração encoraja o agente a selecionar ações aleatórias para explorar novas áreas do ambiente.

No entanto, a explotação é o processo em que o agente usa suas experiências acumuladas para tomar decisões que maximizam sua recompensa. Essa estratégia concentra-se em reforçar as ações que já provaram ser eficazes, otimizando os resultados com base no que o agente já aprendeu.

Para que o agente alcance o equilíbrio ideal entre exploração e explotação, é importante definir cuidadosamente a taxa de exploração. Uma taxa de exploração muito alta pode fazer com que o agente não utilize adequadamente suas experiências passadas, enquanto uma taxa muito baixa pode impedir que ele explore o ambiente completamente. Uma abordagem comum é diminuir gradualmente a taxa de exploração ao longo do processo de aprendizagem, permitindo que o agente explore o ambiente inicialmente e, à medida que ganha mais conhecimento, passe a tomar mais decisões com base em suas experiências passadas.

2.3.2 Elementos do Aprendizado por Reforço

No Aprendizado por Reforço, há seis elementos principais: agente, ambiente, estado, política, recompensa e função de valor [17].

O agente de aprendizagem é o componente responsável por interagir com o ambiente e escolher as ações a serem tomadas. Cada decisão resulta em uma recompensa, que pode ser positiva ou negativa, dependendo da ação escolhida, orientando o aprendizado do agente.

O ambiente é o espaço onde o agente realiza suas ações. Ao realizar uma ação, o agente se encontra em um estado atual (s) dentro desse ambiente e recebe uma recompensa (r) como resultado de sua escolha. Cada nova ação leva o agente a um novo estado (s'), onde o ciclo de interação e aprendizado continua.

A política (π) é a estratégia que orienta o agente na escolha da próxima ação a ser tomada. Em muitos casos, a política utilizada é a probabilidade estocástica. A política desempenha um papel importante no processo de aprendizagem, pois define o comportamento do agente e influencia diretamente a eficácia do aprendizado e a maximização da recompensa.

A recompensa (r) é um dos componentes mais importantes do Aprendizado por Reforço, pois define o objetivo do agente. É através da recompensa que o agente avalia se suas ações imediatas foram boas ou não. Cada ação tomada pelo agente resulta em uma recompensa, que pode ser negativa (como uma punição), positiva ou nula. O objetivo final do agente é maximizar a recompensa acumulada ao longo do tempo e, à medida que ele interage com o ambiente, ele aprenderá quais ações são melhores para alcançar esse objetivo.

A recompensa é recebida após cada ação é imediata, o que significa que sozinha não reflete a recompensa acumulada a longo prazo de todo trajeto do agente, por isso, se utiliza a função de valor. A função de valor pretende fazer com que o agente considere não apenas as recompensas imediatas, mas também as recompensas futuras esperadas. Dessa forma, ela ajuda o agente a avaliar as suas ações, considerando também o impacto a longo prazo, evitando que ele opte por ações que oferecem altas recompensas imediatas, mas que resultam em recompensas menores no futuro. A função de valor é atualizada ao longo das sequências de observações feitas pelo agente durante seu trajeto no ambiente.

2.3.3 A interface de agente-ambiente

Uma das principais características do Aprendizado por Reforço é a arquitetura agente-ambiente. Essa arquitetura, apresentada na Figura 2.1, define como o agente interage com o ambiente para aprender e alcançar seu objetivo final.

O agente é o componente responsável por aprender e tomar decisões com base em suas interações. O ambiente representa tudo o que não é o agente, sendo o espaço onde essas interações ocorrem. De forma contínua, o agente escolhe ações e o ambiente responde a essas ações, apresentando novas situações para o agente, fornecendo recompensas, que são valores numéricos utilizados como *feedback*. O agente visa maximizar essas recompensas ao longo do tempo, ajustando seu comportamento com base nos resultados obtidos [18].

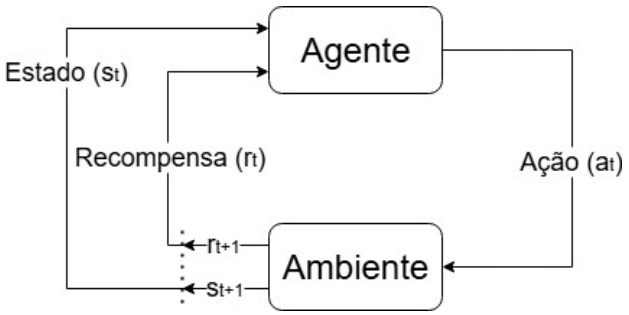

Figura 2.1: Interface Agente-Ambiente do Aprendizado por Reforço.

O agente interage com o ambiente em uma sequência de passos temporais, definidos como $t = 0, 1, 2, 3, \ldots, n$. A cada instante t, o ambiente retorna ao agente o estado s_t, que representa a situação atual do agente no ambiente. Com base nesse estado, o agente escolhe uma ação a_t. Em resposta, o ambiente fornece uma recompensa r_{t+1} e atualiza o estado para s_{t+1}, refletindo o novo estado após a ação tomada.

Assim, a dinâmica da interação entre agente e ambiente pode ser representada pela sequência demonstrada na Equação 2.1. Essa sequência descreve o ciclo contínuo de estado, ação e *feedback* que caracteriza o processo de aprendizado por reforço.

$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, \ldots \tag{2.1}$$

2.3.4 Objetivo e Recompensas

O objetivo final do agente é definido com base nas recompensas retornadas pelo ambiente. Através dessas recompensas, o agente aprende quais ações devem ser tomadas para maximizar não apenas a recompensa imediata recebida em cada estado, mas o total de recompensas acumuladas ao longo do tempo. Esse conceito pode ser descrito como em [19]:

> Tudo aquilo que entendemos como objetivos e finalidades pode ser bem representado como a maximização do valor esperado da soma cumulativa de um sinal escalar recebido (chamado de recompensa).

A soma das recompensas acumuladas pode ser feita simplesmente somando todas as recompensas imediatas recebidas a partir de um determinado tempo t. Esse valor é conhecido como retorno (G_t) e pode ser expresso pela Equação 2.2, onde T representa o tempo final do episódio, ou seja, o último instante em que o agente interage com o ambiente antes do episódio ser encerrado.

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T \tag{2.2}$$

No entanto, é comum utilizar um fator de desconto para ponderar as recompensas futuras. Nesse caso, recompensas mais distantes no tempo têm menor influência na decisão do agente. A fórmula com desconto é definida pela Equação 2.3.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.3}$$

O gama (γ) é o parâmetro de desconto, com $0 \leq \gamma \leq 1$, conhecido como taxa de desconto (*discount rate*). Esse parâmetro serve para equilibrar a importância entre recompensas imediatas e futuras, ou seja, quanto mais próximo de 1 for o valor de γ, maior será a consideração pelas recompensas futuras. Por outro lado, valores menores de γ fazem com que o agente priorize recompensas imediatas.

2.3.5 Métodos de valor de ação - *Action-Values Method*

No Aprendizado por Reforço, é necessário estimar os valores das ações em diferentes estados para que o agente aprenda a maximizar sua recompensa ao longo do tempo. Essa técnica é conhecida como método de valor de ação, e é geralmente representada pela função $Q_t(s, a)$, onde s é o estado, a é a ação, e t é o instante de tempo [20].

Existem diversos métodos para estimar o valor de uma ação, como o método de Monte Carlo, no qual o valor de uma ação em um determinado estado é estimado pela média das recompensas obtidas após executá-la, ao longo de múltiplos episódios completos e o método de diferença temporal (*Temporal Difference*), que combina aspectos dos métodos de Monte Carlo e métodos dinâmicos, permitindo atualizações baseadas na diferença entre estimativas sucessivas.

Também é possível utilizar funções aproximadoras, como redes neurais, para estimar $Q(s, a)$ em espaços de estado-ação muito grandes, sendo o *Deep Q-Network* (DQN) um dos mais utilizados. Em casos mais simples, o valor de uma ação pode ser estimado por uma média empírica das recompensas recebidas ao longo do tempo, conforme descrito na Equação 2.4. Nessa equação R_i é a recompensa recebida no instante i, A_i é a ação tomada no instante i, e $\mathbb{I}[A_i = a]$ é uma função indicadora que vale 1 quando a ação a foi escolhida no tempo i, e 0 caso contrário.

$$Q_t(a) \doteq \frac{\text{soma das recompensas obtidas ao se tomar a ação } a \text{ antes do tempo } t}{\text{número de vezes que a ação } a \text{ foi tomada antes do tempo } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{I}[A_i = a]}{\sum_{i=1}^{t-1} \mathbb{I}[A_i = a]}, \tag{2.4}$$

Uma maneira simples de escolher qual ação deve ser tomada em um determinado instante é selecionar a ação que possui a maior estimativa de valor de ação. Essa abordagem é conhecida como política gananciosa (*greedy policy*), na qual o agente sempre escolhe a melhor ação estimada até o momento, sendo essa a ação com o maior valor $Q_t(a)$.

Essa ação gananciosa é definida pela Equação 2.5, sendo A_t a ação escolhida no instante t, e $Q_t(a)$ a estimativa do valor da ação a nesse momento. Essa estratégia favorece a exploração da melhor ação conhecida, mas pode levar o agente a negligenciar outras ações, potencialmente melhores se não houver uma estratégia de exploração adequada combinada.

$$A_t = \arg\max_a Q_t(a), \tag{2.5}$$

2.3.6 Políticas e Funções de Valor

As funções de valor são fundamentais no processo de aprendizagem do agente, pois permitem estimar o quão boa foi a escolha de uma ação em determinado estado. É por meio dessas funções que o agente avalia as consequências de suas decisões, com base nas recompensas esperadas [21].

A política (π) é o mecanismo que define o comportamento do agente, mapeando cada estado para uma distribuição de probabilidade sobre as ações possíveis [21]. Se o agente segue uma política π no instante t, como na Equação 2.6, isso representa a probabilidade de o agente escolher a ação $A_t = a$, dado que se encontra no estado $S_t = s$.

$$\pi(a|s) = \mathbb{P}(A_t = a \mid S_t = s) \tag{2.6}$$

A função de valor de estado $V^\pi(s)$ representa o valor esperado das recompensas futuras que o agente pode receber ao iniciar no estado s e seguir a política π a partir daí. Essa função é definida na Equação 2.7.

$$V^\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right] \tag{2.7}$$

A função de valor de ação $Q^\pi(s, a)$, por sua vez, representa o valor esperado de iniciar no estado s, tomar a ação a, e depois seguir a política π. Sua definição é conforme a Equação 2.8.

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right] \tag{2.8}$$

Essas funções são importantes para a tomada de decisão, pois permitem ao agente avaliar e comparar diferentes ações com base no retorno esperado.

Existem também as políticas ótimas e as funções de valor ótimas, que representam a melhor forma possível de agir dentro de um determinado ambiente de aprendizado [22]. A política ótima π^* é aquela que maximiza o valor esperado da recompensa acumulada para todos os estados, sendo definida conforme a Equação 2.9. Já a função de valor ótima de ação $Q^*(s, a)$ é definida conforme a Equação 2.10.

$$V^*(s) = \max_\pi V^\pi(s) \tag{2.9}$$

$$Q^*(s, a) = \max_\pi Q^\pi(s, a) \tag{2.10}$$

Essas funções otimizadas são importantes para encontrar a melhor política possível e servem como base para algoritmos como *Value Iteration* e *Q-Learning*.

2.3.7 Algoritmo *Q-Learning*

Um dos algoritmos mais conhecidos do Aprendizado por Reforço é o *Q-Learning* [23]. Nesse algoritmo, o *Q* refere-se à função de valor de ação Q, cujo objetivo é aproximar $Q^*(s, a)$, ou seja, a função de valor ótima para um par estado-ação. A função Q é definida conforme a Equação 2.11.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right] \tag{2.11}$$

A Equação 2.11 traz algumas variáveis importantes para o aprendizado, sendo elas:

- α é a taxa de aprendizado (*learning rate*),
- γ é o fator de desconto (*discount factor*),
- $\max_a Q(S_{t+1}, a)$ é o melhor valor estimado para o próximo estado.

Essa atualização permite que o agente aprenda, ao longo do tempo, quais ações tendem a levar aos maiores retornos. Como demonstrado no trabalho [23], o algoritmo *Q-Learning* é um método de controle baseado em Diferença Temporal *(Temporal Difference)*, que combina ideias de aprendizado por diferença temporal com controle de ações.

Uma característica fundamental do *Q-Learning* é o fato de ser um algoritmo *off-policy*, ou seja, ele aprende a política ótima independentemente da política que está sendo seguida pelo agente durante o treinamento. O agente pode explorar o ambiente utilizando uma política exploratória, como a política ϵ-greedy, mas a atualização dos valores da função $Q(s, a)$ considera a melhor ação possível no próximo estado, conforme a política ótima.

Desse modo, o *Q-Learning* busca aprender diretamente a função de valor ótima $Q^*(s, a)$, mesmo enquanto executa ações baseadas em uma política diferente. Essa característica contribui significativamente para sua popularidade e eficácia em diversos tipos de ambientes.

Outra característica importante do *Q-Learning* é que ele não requer conhecimento prévio das transições do ambiente nem das funções de recompensa; o aprendizado ocorre unicamente por meio de tentativa e erro, com base nas interações do agente com o ambiente.

O Algoritmo 1 apresenta o funcionamento básico do *Q-Learning*.

Algorithm 1 Q-Learning

Parâmetros: taxa de aprendizado $\alpha \in (0, 1]$, fator de desconto $\gamma \in [0, 1]$, parâmetro $\epsilon > 0$
Inicialize $Q(s, a)$ arbitrariamente para todos os $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, exceto que $Q(\text{terminal}, \cdot) = 0$
for cada episódio **do**
    Inicialize o estado S
    while S não for terminal **do**
        Escolha A a partir de S usando uma política derivada de Q (ex.: ϵ-greedy)
        Tome a ação A, observe a recompensa R e o novo estado S'
        Atualize $Q(S, A)$:

$$Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma \max_a Q(S', a) - Q(S, A) \right]$$

        $S \leftarrow S'$
    end while
end for

A *Q-Table*, ou tabela de ação-valor, é uma estrutura importante no algoritmo *Q-Learning*. Ela armazena os valores estimados da função $Q(s, a)$, que representa o valor esperado de se tomar uma determinada ação a em um estado s, como uma estrutura de dados de armazenamento.

A tabela de ação-valor é geralmente representada como uma matriz bidimensional, onde cada linha corresponde a um estado do ambiente, e cada coluna representa uma ação possível. Cada entrada da matriz, $Q(s, a)$, é atualizada durante o treinamento do agente a cada iteração, com base nas recompensas recebidas e na estimativa da melhor ação futura. O objetivo é que, ao final do treinamento, os valores na *Q-Table* representem as estimativas do retorno esperado acumulado para cada par de estado-ação.

Ao utilizar a *Q-Table*, o agente consegue escolher a ação com o maior valor estimado para o estado atual, fazendo com que o agente aprenda quais ações maximizam a recompensa total acumulada, mesmo sem conhecer previamente a dinâmica do ambiente.

Porém, a utilização dessa estrutura de dados é eficaz em ambientes com espaços de estados e ações discretos e relativamente pequenos, pois a *Q-Table* se torna inviável quando o número de estados ou ações é muito grande, ou contínuo. Nesses casos, utilizam-se funções aproximadoras, como redes neurais, para estimar os valores $Q(s, a)$.

2.3.8 *Deep Q Learning* - Aprendizado Q Profundo

Para superar a limitação do *Q-Learning* em ambientes com grandes espaços de estados e ações, onde a construção e atualização de uma *Q-Table* se torna inviável, principalmente pela alta quantidade de memória necessária para armazenamento da matriz, foi desenvolvido o *Deep Q-Learning*. O termo *deep* refere-se à utilização de redes neurais profundas (*deep neural networks*) como aproximadores da função de valor $Q(s, a)$, substituindo assim a tabela utilizada no *Q-Learning* tradicional [24].

A ideia central do *Deep Q-Learning* é utilizar uma rede neural parametrizada por θ para aproximar a função de valor de ação $Q(s, a; \theta)$. Nessa arquitetura, o estado s é fornecido como entrada para a rede, e a saída corresponde aos valores estimados para cada ação possível naquele estado. Essa rede neural substitui

a *Q-Table* tradicional do *Q-Learning* clássico, tornando o algoritmo escalável para ambientes com grandes espaços de estados e ações.

O treinamento da rede é com base na aprendizagem por diferença temporal (*Temporal Difference Learning*), por meio da minimização da função de perda, definida na Equação 2.12, conhecida como *Q-Loss* [25]. Nessa equação, $\mathbb{E}_{(s,a,r,s')}$ é o valor esperado sobre as amostras de transições (s, a, r, s') obtidas da experiência do agente, frequentemente armazenadas em um *buffer* de memória (*replay buffer*). O objetivo dessa função é minimizar o erro quadrático entre o valor estimado pela rede neural $Q(s, a; \theta)$ e o *Q-Target* y, definido na Equação 2.13.

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')} \left[(y - Q(s, a; \theta))^2 \right] \tag{2.12}$$

O termo y nessa equação representa o *Q-Target*, calculado conforme a Equação 2.13, sendo uma estimativa do lado direito da equação de *Bellman* ótima para funções de valor de ação. Essa estimativa considera a recompensa imediata recebida e o valor máximo estimado das ações possíveis no próximo estado, sendo fundamental para guiar o aprendizado da rede neural no *Deep Q-Learning*.

$$y = r + \gamma \max_{a'} Q(s', a'; \theta^-) \tag{2.13}$$

As variáveis envolvidas na Equação 2.13 são definidas da seguinte forma:

- r: recompensa imediata recebida após a execução da ação a no estado s;
- s': novo estado resultante após a transição;
- γ: fator de desconto, responsável por ponderar a importância das recompensas futuras;
- θ^-: parâmetros da *target network*, uma cópia estável da rede neural principal, atualizada periodicamente para melhorar a estabilidade do treinamento.

2.3.8.1 *Experience Replay*

O *Experience Replay* no *Deep Q-Learning* consiste em uma memória que armazena as transições vividas pelo agente durante sua interação com o ambiente. Essa técnica é bastante utilizada para estabilizar e melhorar o treinamento, pois permite que experiências passadas sejam reutilizadas diversas vezes, evitando que o agente esqueça situações anteriormente vividas, promovendo uma melhor generalização do aprendizado [24].

Cada entrada nessa memória é composta por uma tupla (s, a, r, s'), contendo o estado atual s, a ação executada a, a recompensa obtida r e o próximo estado s'. Essas transições podem ser registradas à medida que o agente percorre o ambiente. Porém, em algumas abordagens, a memória pode ser inicialmente preenchida com transições aleatórias para acelerar o início do aprendizado.

A memória de replay possui um tamanho máximo fixo N, quando esse limite é atingido, as transições mais antigas são descartadas para dar lugar a novas experiências, conforme uma política de substituição

definida (geralmente FIFO – *First In, First Out*).

Durante o treinamento da rede neural, são amostrados mini-*batches* aleatórios de transições armazenadas nessa memória. Essa amostragem aleatória é alimentada na rede neural e contribui para uma atualização mais estável e eficiente dos parâmetros da rede.

2.3.8.2 *Target Network* - Rede Alvo

Para melhorar a estabilidade do treinamento no *Deep Q-Learning*, é comum utilizar uma segunda rede neural chamada *Target Network*, ou rede alvo. Essa rede é uma cópia da rede principal, responsável por estimar a função de valor de ação $Q(s, a; \theta)$, mas com parâmetros fixos θ^- que são atualizados com menor frequência [24].

O principal motivo para o uso da *Target Network* é evitar oscilações e instabilidades no processo de aprendizado. Essas oscilações acontecem principalmente quando os alvos utilizados na função de perda mudam muito rapidamente. Portanto, manter os alvos estáveis por alguns passos de tempo torna o aprendizado mais consistente.

A atualização da rede alvo pode ser feita de duas formas principais, sendo a atualização periódica a mais utilizada:

- Atualização periódica: A cada C passos, os parâmetros da rede alvo são substituídos pelos parâmetros atuais da rede principal, ou seja, $\theta^- \leftarrow \theta$.
- Atualização suave: Os parâmetros da rede alvo são atualizados gradualmente a cada passo usando uma taxa τ, como em $\theta^- \leftarrow \tau\theta + (1 - \tau)\theta^-$.

Por fim, o algoritmo final do aprendizado com *Deep Q-Learning* está apresentado no Algoritmo 2.

Algorithm 2 Algoritmo de Deep Q-Learning com *Experience Replay* e *Target Network*

Inicialize a memória de repetição de experiências D com capacidade N
Inicialize a função de valor de ação Q com pesos aleatórios θ
Inicialize a rede alvo $\hat{Q}$ com pesos $\theta^- = \theta$
for episódio $= 1$ até M **do**
    Inicialize o estado s_1 e preprocessamento $\phi_1 = \phi(s_1)$
    for $t = 1$ até T **do**
        Com probabilidade ϵ selecione uma ação aleatória a_t
        Caso contrário, selecione $a_t = \arg\max_a Q(\phi(s_t), a; \theta)$
        Execute a ação a_t, observe a recompensa r_t e próximo estado s_{t+1}
        Preprocessamento: $\phi_{t+1} = \phi(s_{t+1})$
        Armazene a transição $(\phi_t, a_t, r_t, \phi_{t+1})$ em D
        Amostre aleatoriamente um mini-*batch* de transições $(\phi_j, a_j, r_j, \phi_{j+1})$ de D
        Defina o alvo:

$$y_j = \begin{cases} r_j & \text{se o episódio termina em } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{caso contrário} \end{cases}$$

        Realize um passo de descida do gradiente na função de perda:

$$\mathcal{L}(\theta) = \left(y_j - Q(\phi_j, a_j; \theta)\right)^2$$

        A cada C passos, atualize a rede alvo: $\theta^- \leftarrow \theta$
    end for
end for

2.4 SAÍDAS DE EMERGÊNCIA E ROTAS DE FUGA

Em situações de emergências, como incêndios, a evacuação rápida e segura é importante para a preservação da vida das vítimas. As saídas de emergência e rotas de fuga desempenham papel importante nesse processo, sendo caminhos projetados para permitir a saída segura do edifício, além de funcionarem como acessos estratégicos para que equipes de resgate, como o corpo de bombeiros, possam combater o fogo e realizar a retirada de vítimas [26].

Conforme a Nota Técnica n.º 11/2022 do Corpo de Bombeiros Militar do Estado de Goiás [26], todas as saídas de emergência devem permanecer desobstruídas e livres de qualquer tipo de barreira, assegurando o livre fluxo de pessoas durante uma evacuação. Complementarmente, a norma NBR 9077:2001 [7] da Associação Brasileira de Normas Técnicas (ABNT) estabelece requisitos mínimos para a concepção e manutenção dessas rotas, como o dimensionamento adequado, a sinalização eficiente, a iluminação de emergência e a proteção contra incêndios, visando garantir a evacuação segura em situações críticas.

Embora as diretrizes normativas tenham o objetivo de criar condições ideais para a evacuação, a dinâmica real de um incêndio pode gerar cenários imprevisíveis, como a obstrução de passagens, a propagação rápida do fogo e a formação de áreas de risco inesperadas. Todos esses fatores podem comprometer a eficiência do planejamento estático das rotas de fuga previamente estabelecidas, evidenciando a necessidade de abordagens mais adaptativas e inteligentes para apoiar a evacuação em tempo real.

Nesse contexto, o uso de técnicas de aprendizado por reforço apresenta-se como uma solução inovadora. Por meio de agentes treinados para explorar ambientes complexos, o Aprendizado por Reforço permite a identificação de rotas que minimizem o tempo de evacuação e evitem zonas de perigo, mesmo diante de mudanças dinâmicas no ambiente. Esses agentes podem ser ensinados a respeitar as restrições físicas impostas pelas normas de segurança, considerar variáveis dinâmicas, como bloqueios e fumaça, e utilizar informações auxiliares para a tomada de decisão eficiente.

A integração de técnicas de Aprendizado por Reforço ao planejamento de rotas de evacuação representa uma oportunidade de tornar as evacuações mais seguras, realistas e adaptáveis, além de fornecer suporte potencial às operações de resgate realizadas por equipes especializadas, não apenas para o aprimoramento da segurança das vítimas, mas também para a otimização das estratégias de resposta a emergências.

2.5 PLANTA BAIXA

Antes da construção de qualquer edifício, seja ele comercial ou residencial, é necessário um planejamento especial, realizado por profissionais como engenheiros e arquitetos. Uma das etapas fundamentais desse processo é a elaboração da planta baixa.

A planta baixa é uma representação visual bidimensional (2D) do *layout* de uma construção, vista de cima. Ela oferece uma visão clara da disposição dos cômodos, portas, janelas e demais elementos arquitetônicos existentes, permitindo uma compreensão espacial detalhada da estrutura planejada.

Atualmente, existem diversas formas de se elaborar uma planta baixa, desde métodos tradicionais com desenho manual até o uso de ferramentas computacionais avançadas, como o AutoCAD [27]. Apesar de poderem variar em estilo e nível de detalhamento, muitas seguem padrões técnicos que garantem a legibilidade e a consistência das informações representadas.

No contexto deste trabalho, a planta baixa serve como base para a modelagem do ambiente em forma de grafo, permitindo a simulação de trajetórias e tomada de decisão mediante Aprendizado por Reforço. Ao transformar essa representação arquitetônica em uma estrutura de dados adequada, é possível realizar simulações de navegação e evacuação com base em ambientes reais.

Um exemplo de planta baixa é apresentado na Figura 2.2, correspondente ao térreo do laboratório SG11 da Universidade de Brasília.

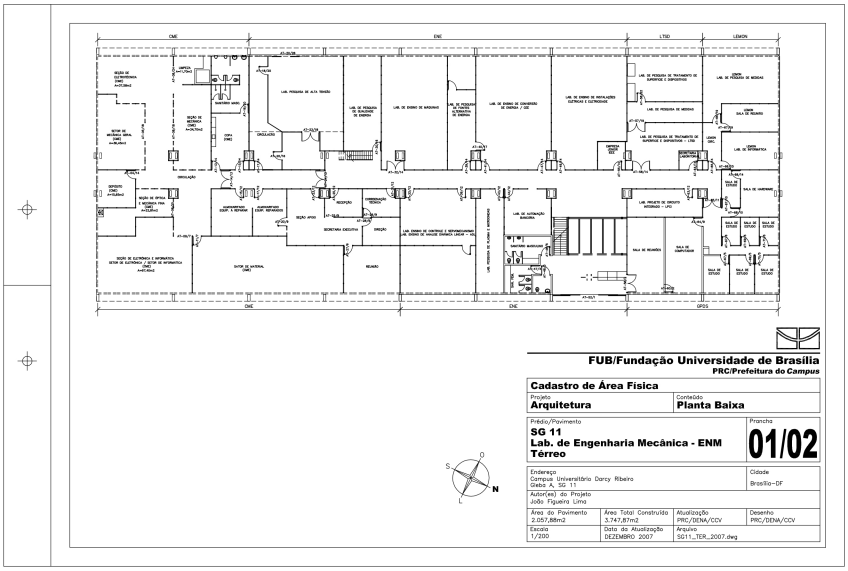

Figura 2.2: Planta baixa do térreo do Laboratório SG11/UnB.

2.6 TEORIA DE GRAFOS

A Teoria dos Grafos é um importante ramo da matemática que iniciou em meados do século XVIII com o matemático suíço Leonhard Euler, ao publicar um artigo discutindo o famoso problema das sete pontes de *Königsberg* [28]. *Königsberg* era uma cidade localizada no território da Prússia até 1945, composta por duas grandes ilhas separadas pelo rio Prególia e conectadas por sete pontes, conforme ilustrado na Figura 2.3. O desafio consistia em determinar se seria possível atravessar todas as pontes uma única vez, sem repetir nenhuma. Em 1736, Euler demonstrou que tal travessia era impossível. Para isso, ele representou o problema por meio de nós e arestas, criando assim o primeiro grafo conhecido e estabelecendo os fundamentos da Teoria dos Grafos.

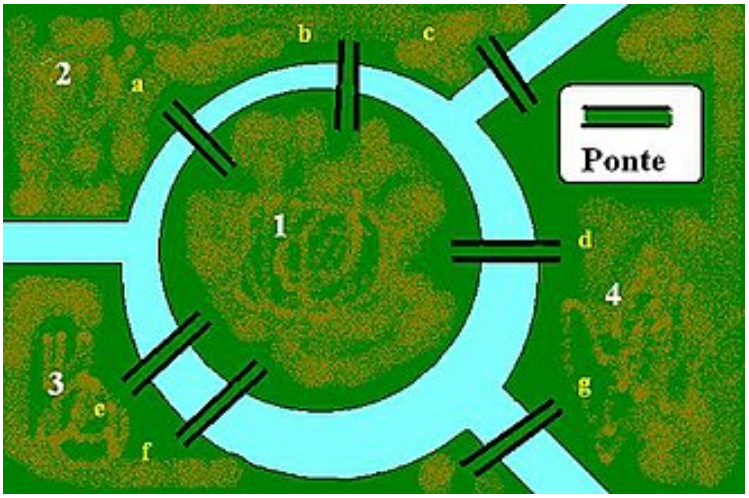

Figura 2.3: Esquema das sete pontes de Königsberg [2].

2.6.1 Grafos

Um grafo é uma estrutura matemática utilizada para representar e modelar relações entre elementos de um conjunto. Essa estrutura é composta por vértices (ou nós), que representam os elementos, e por arestas, que representam as conexões ou relações entre esses elementos.

Um grafo é geralmente representado pela letra G e possui dois componentes principais: um conjunto de vértices V e um conjunto de arestas E. A sua definição formal pode ser expressa da seguinte maneira:

> *Um grafo $G = (V, E)$ é definido por um conjunto não vazio V de vértices e um conjunto E de arestas, tal que cada aresta $e \in E$ é definida por um par não ordenado (u, v), onde $u, v \in V$.* [29]

O grafo é frequentemente representado de forma visual, onde os vértices (ou nós) são desenhados como pontos e as arestas como linhas ou curvas conectando pares de vértices. Em termos conceituais, uma aresta representa uma relação entre dois elementos do conjunto de vértices. Um exemplo do desenho de um grafo pode ser encontrado na Figura 2.4.

A estrutura de um grafo está diretamente relacionada à teoria dos conjuntos, pois um grafo é definido como um par ordenado $G = (V, E)$, onde V é um conjunto não vazio de vértices e E é um subconjunto do produto cartesiano $V \times V$, representando as conexões (arestas) entre os elementos de V. Essa base formal permite utilizar grafos de maneira matemática, facilitando a análise de suas propriedades e aplicações [30].

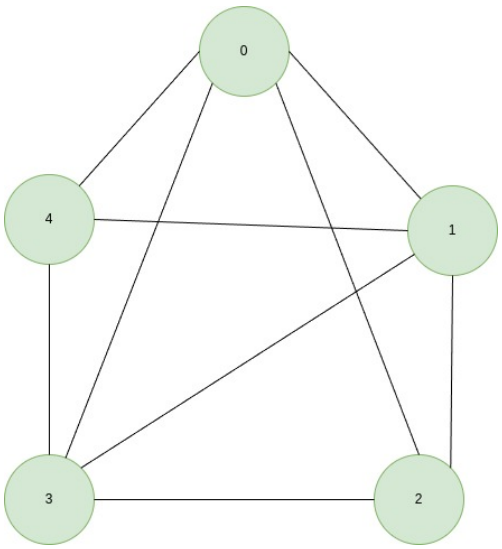

Figura 2.4: Exemplo do desenho de um grafo.

Um grafo pode ser representado por um conjunto de vértices $V = \{v_1, v_2, v_3, v_4, \ldots\}$ e um conjunto de arestas $E = \{(v_1, v_2), (v_1, v_4), (v_1, v_3), (v_2, v_3), (v_4, v_3), \ldots\}$, onde cada aresta é um par não ordenado de vértices que indica uma conexão entre eles [29].

2.6.2 Definições

As definições fundamentais na teoria básica dos grafos são apresentadas a seguir, conforme demonstrado em [29].

Definição 2.6.1 (Incidência)**.** *Diz-se que uma aresta é incidente a um vértice quando está conectada a ele. Ou seja, uma aresta $e = (u, v)$ é incidente aos vértices u e v.*

Definição 2.6.2 (Adjacência)**.** *Dois vértices são considerados adjacentes quando estão conectados por uma aresta. Assim, se existe uma aresta $e = (u, v) \in E$, então os vértices u e v são adjacentes. Vértices adjacentes também são comumente chamados de vizinhos.*

Definição 2.6.3 (Ordem)**.** *A ordem de um grafo corresponde à cardinalidade do conjunto V. Em outras palavras, a ordem indica a quantidade de vértices do grafo, sendo denotada por $|V|$ ou pela letra n.*

Definição 2.6.4 (Tamanho)**.** *O tamanho de um grafo corresponde à cardinalidade do conjunto E. Em outras palavras, o tamanho indica a quantidade de arestas do grafo, sendo denotada por $|E|$ ou pela letra m.*

Definição 2.6.5 (Densidade)**.** *A densidade de um grafo é uma medida que relaciona seu tamanho à sua ordem, representando a proporção entre a quantidade de arestas existentes e o número máximo possível de arestas entre os vértices.*

Definição 2.6.6 (Percurso)**.** *Em um grafo não dirigido ou direcionado, um percurso P é uma sequência alternada de vértices e arestas, dada por*

$$P = \langle v_0, e_1, v_1, e_2, \ldots, e_n, v_n \rangle,$$

onde cada aresta e_i conecta os vértices v_{i-1} e v_i, isto é, o vértice final de uma aresta coincide com o inicial da próxima.

Definição 2.6.7 (Comprimento)**.** *O comprimento de um percurso é dado pela quantidade de arestas percorridas. Denota-se por $\delta(s, v)$ o comprimento do percurso entre os vértices s e v. Se não existir percurso possível entre v_0 e v_n, define-se $\delta(v_0, v_n) = \infty$.*

Definição 2.6.8 (Trilha)**.** *Em um grafo, uma trilha é um percurso que não repete arestas.*

Definição 2.6.9 (Caminho)**.** *Um caminho é uma trilha que não repete vértices, exceto, eventualmente, os vértices inicial e final.*

Definição 2.6.10 (Caminho Mínimo)**.** *Dados dois vértices s e v, o caminho mínimo entre s e v é definido como o caminho de menor comprimento dentre todos os caminhos existentes entre s e v. Note que, dependendo da estrutura do grafo, pode haver mais de um caminho mínimo entre s e v.*

2.6.3 Grafos Direcionados

Existem também os grafos direcionados, ou dirigidos, nos quais as arestas possuem um sentido definido entre os vértices. Nesse caso, cada aresta é representada por um par ordenado de vértices, em que o primeiro

elemento indica o vértice de origem e o segundo o vértice de destino. Esses grafos são conhecidos como dígrafos, e são geralmente utilizados para modelar relações assimétricas, como rotas unidirecionais, fluxos de informação ou dependências entre tarefas.

Considere o dígrafo $G = (V, E)$, onde:

$$V = \{v_1, v_2, v_3\} \quad \text{e} \quad E = \{(v_1, v_2), (v_2, v_3), (v_3, v_1)\}.$$

A aresta (v_1, v_2) indica uma conexão do vértice v_1 para v_2, mas não necessariamente o contrário. Assim, as conexões têm um sentido definido, diferente dos grafos não direcionados.

2.6.4 Grafos Rotulados

Grafos rotulados são aqueles cujos vértices e arestas possuem algum tipo de identificação individual, como números, letras ou pesos, que permitem distinguir ou atribuir significado a cada elemento do grafo.

Por exemplo, considere o grafo $G = (V, E)$, onde

$$V = \{v_1, v_2, v_3\}, \quad \text{com rótulos} \quad \{A, B, C\}$$

e

$$E = \{(v_1, v_2), (v_2, v_3), (v_3, v_1)\}, \quad \text{com pesos} \quad \{5, 3, 7\}.$$

Aqui, cada vértice tem um rótulo (A, B, C) e cada aresta possui um peso numérico que pode representar, por exemplo, distâncias ou custos entre os vértices.

3 TRABALHOS RELACIONADOS

3.1 PLANEJAMENTO DE ROTAS DE FUGA

A análise de melhor caminho é um campo da matemática e da teoria dos grafos que começou a se desenvolver no século XX [31]. Este campo se concentra em encontrar o caminho mais eficiente entre pontos em uma rede ou grafo, levando em conta critérios como distância, custo ou tempo. A análise de melhor caminho é fundamental para diversas aplicações práticas, incluindo roteamento em redes de computadores, planejamento de trajetos em sistemas de navegação, otimização de processos logísticos e definição de rotas de fuga.

Entre os algoritmos mais importantes para resolver problemas de melhor caminho estão o Algoritmo de Dijkstra, o Algoritmo de Bellman-Ford e o Algoritmo A*. O Algoritmo de Dijkstra, desenvolvido por Edsger W. Dijkstra em 1956, encontra o caminho mais curto a partir de um vértice de origem para todos os outros em um grafo com pesos não negativos, utilizando uma abordagem gulosa que seleciona iterativamente o vértice com a menor distância conhecida [32].

O Algoritmo de Bellman-Ford, criado por Richard Bellman e Lester Ford na década de 1950, é adequado para grafos com pesos negativos nas arestas e pode detectar ciclos de peso negativo [33]. Já o Algoritmo A*, desenvolvido por Peter Hart, Nils Nilsson e Bertram Raphael em 1968, combina uma heurística com a busca de caminho para otimizar a solução, utilizando uma função de custo total estimado para direcionar a busca de forma mais eficiente [34].

Vários estudos utilizam esses algoritmos para encontrar os melhores caminhos em ambientes complexos, principalmente no contexto de jogos. Existem trabalhos que comparam os diferentes algoritmos de busca para avaliar sua eficácia [35], [36], [37]. Esses estudos evidenciam como o algoritmo A* é o mais indicado no contexto de busca de caminhos, demonstrando sua superioridade em cenários dinâmicos e desafiadores.

Apesar desses algoritmos clássicos serem eficientes para muitos problemas de caminho, eles enfrentam limitações em contextos complexos e dinâmicos. Por isso, a IA se mostra uma boa opção nesses contextos, pois pode lidar com complexidades e variabilidades que algoritmos tradicionais encontram dificuldades. Modelos de IA, como redes neurais e algoritmos de Aprendizado de Reforço, podem aprender a adaptar suas estratégias a diferentes condições e ajustar suas decisões em tempo real.

Diversos estudos têm explorado a identificação de rotas seguras em emergências e desastres utilizando IA. Essas pesquisas focam no desenvolvimento de soluções para aprimorar o planejamento de rotas de evacuação, visando aumentar a segurança e reduzir o tempo de resposta em momentos críticos.

Rotas de fuga em casos de incêndios florestais e ambientes abertos são amplamente estudadas na literatura. O trabalho de [38] propõe um método de planejamento de rotas de fuga em cenários de incêndios florestais, utilizando um algoritmo A* aprimorado, com dados provenientes de imagens de sensoriamento remoto por satélites. Essa abordagem considera múltiplos fatores ambientais, como vegetação, topografia e comportamento do fogo, para minimizar a exposição ao risco ao longo da rota. Diferentemente do tra-

balho [38], este estudo não tem como foco a análise de ambientes abertos e também não utiliza algoritmos clássicos de busca.

Além das rotas de fuga em incêndios ambientais, desastres naturais como terremotos apresentam desafios adicionais. O trabalho de [39] aborda esse problema ao propor uma solução baseada em aprendizado supervisionado híbrido clássico-quântico, utilizando uma arquitetura com redes FiLM (*Feature-wise Linear Modulation*) combinadas com uma rede neural quântica. O objetivo do modelo é aprender a imitar o comportamento do algoritmo de Dijkstra na identificação dos menores caminhos em um grafo urbano dinâmico, representando o cenário de evacuação veicular em uma cidade após um terremoto, considerando ainda incertezas e bloqueios. Uma diferença em relação ao presente trabalho é que [39] foca na evacuação veicular em larga escala, enquanto este estudo se concentra em ambientes fechados discretizados, modelados como grafos.

Problemas de rotas de fuga também são relevantes na área de robótica móvel, especialmente em cenários com baixa visibilidade. O trabalho de [40] propõe uma solução baseada em Aprendizado por Reforço para identificação de rotas de evacuação em incêndios estruturais, considerando ambientes com elevada concentração de fumaça, onde métodos tradicionais de localização, como SLAM, tornam-se inviáveis. A abordagem utiliza um sistema embarcado em um drone, equipado com LiDAR, sensores de sonar e módulos de detecção de temperatura e umidade, para criar mapas de visibilidade e atribuir escores de perigo aos nós do ambiente. Embora o trabalho também utilize Aprendizado por Reforço, o presente estudo não foca na navegação autônoma de robôs e sim de pessoas. Além disso, o agente desenvolvido em [40] está voltado principalmente para a navegação em condições de baixa visibilidade, com foco na percepção espacial por sensores físicos embarcados.

O trabalho [41] explora a evacuação em ambientes fechados, como supermercados, utilizando um algoritmo de Otimização por Colônia de Formigas (IACO), que leva em conta parâmetros dinâmicos como temperatura, concentração de fumaça e monóxido de carbono, obtidos por sensores físicos e simulações computacionais. Apesar deste trabalho também tratar da evacuação em ambientes fechados (*indoor*), a abordagem utilizada é baseada em uma meta-heurística clássica (IACO), com um processo de otimização que depende de informações detalhadas e em tempo real sobre as condições ambientais, como temperatura e concentração de fumaça, obtidas por sensores físicos e simulações.

O trabalho [42] realiza uma análise de rotas de fuga em ambientes fechados combinando o algoritmo A* com o algoritmo *K-Nearest Neighbors* (KNN) para prever as melhores rotas de evacuação. O estudo propõe um processo de pré-seleção de saídas, utilizando o KNN para estimar quais saídas são mais adequadas para diferentes posições iniciais no edifício, antes de aplicar o A* para o cálculo do caminho. Um dos principais focos da pesquisa é verificar se essa estratégia de pré-seleção mantém uma taxa de sucesso elevada em diferentes layouts de planta baixa, demonstrando que a escolha antecipada da saída pode reduzir o tempo total de evacuação.

Vários estudos focam em rotas de fuga para cenários de incêndios em ambientes fechados utilizando Aprendizado por Reforço. O trabalho [43] propõe um ambiente de evacuação de incêndio modelado como um grafo, permitindo o treinamento de agentes baseados em Deep Q-Learning (DQN). Um dos principais diferenciais do estudo é a utilização de *Transfer Learning*, em que o agente é inicialmente treinado para aprender os caminhos mais curtos até a saída e então esse conhecimento prévio é transferido quando o

agente precisa entender cenários mais complexos e dinâmicos, como o espalhamento do fogo e gargalos de evacuação (*bottlenecks*), exigindo que o agente adapte suas decisões em tempo real. Para cada nova configuração de foco de incêndio, o agente é re-treinado para refletir as mudanças no ambiente.

O trabalho [43] foi uma das principais inspirações para a etapa inicial da presente pesquisa, especialmente no que diz respeito à representação do ambiente como grafo e à utilização de técnicas de Aprendizado por Reforço para definição de rotas de evacuação. Entretanto, o presente estudo avança ao remover a necessidade de re-treinamento a cada mudança de cenário, permitindo que o agente realize inferência em tempo real para diferentes configurações de obstáculos (fogo e/ou congestionamento) sem a necessidade de atualização da política aprendida.

Muitas variáveis podem influenciar a determinação da melhor rota de fuga, incluindo o congestionamento gerado pelo alto fluxo de pessoas em ambientes. O trabalho [44] tem como foco a otimização das rotas de evacuação visando minimizar o tempo total necessário para que todas as vítimas alcancem uma saída segura. Para isso, as vítimas são agrupadas de acordo com sua localização, e o algoritmo calcula o momento ideal para que cada grupo inicie seu deslocamento, para evitar a formação de gargalos e congestionamentos. Como método de busca de caminho, o estudo utiliza o algoritmo de Dijkstra, um dos algoritmos clássicos mais utilizados em problemas de rotas de evacuação.

A maioria dos trabalhos citados anteriormente foca na análise das rotas de fuga, sem explorar detalhadamente a transformação de dados reais, como plantas baixas de edifícios, em ambientes de aprendizado para algoritmos de IA. Este trabalho dá continuidade ao estudo apresentado em [1], que, além de propor um algoritmo de Aprendizado por Reforço profundo para rotas de fuga em incêndios em ambientes fechados, também desenvolveu uma interface para a criação do grafo do ambiente. Essa interface possibilita a transformação de plantas baixas em grafos por meio de uma ferramenta manual de desenho, facilitando a criação e a experimentação de ambientes para o treinamento e a análise de agentes inteligentes em cenários de evacuação.

3.2 REPRESENTAÇÃO DO AMBIENTE DE APRENDIZAGEM

Existem duas abordagens reconhecidas para transformar um cenário de rotas de saída em um ambiente de aprendizado: uma baseada em matrizes e outra em grafos.

A abordagem matricial é bastante utilizada em jogos e no problema conhecido como *Multi-Agente Pathfinding* (MAPF), onde o ambiente é representado por uma matriz. Essa representação facilita a navegação e a busca de caminhos para múltiplos agentes simultaneamente [45]. No estudo [42], a matriz é utilizada como ambiente de aprendizado para a identificação da melhor rota de fuga.

Para problemas com ambientes de grande escala, a representação por grafos é geralmente preferida, pois a iteração sobre um grafo e sua matriz de adjacência geralmente apresenta menor custo computacional do que a iteração sobre uma matriz completa. Em um grafo, são consideradas apenas as conexões relevantes entre os nós, reduzindo a complexidade ao eliminar a necessidade de examinar cada célula individualmente, como ocorre na matriz, onde todos os elementos da grade devem ser avaliados. Essa eficiência torna a representação por grafos especialmente adequada para ambientes grandes, nos quais o uso de matrizes pode

se tornar inviável devido ao crescimento exponencial do número de operações computacionais exigidas.

No trabalho [46], uma rede rodoviária é modelada como um grafo, e técnicas de Aprendizado por Reforço Profundo são empregadas para simular cenários de busca e perseguição em operações policiais. De forma semelhante, os estudos [1], [43] e [44] utilizam a representação por grafos para modelar plantas baixas de edifícios, aplicando essa estrutura para análise e planejamento de rotas de fuga em ambientes fechados.

No entanto, a representação automática de imagens de plantas de edifícios ou ambientes abertos em representações baseadas em grafos representa um grande desafio. Isso se deve ao fato de que essa transformação requer a identificação precisa de nós e arestas que correspondam às características físicas e estruturais do ambiente. É necessário interpretar a planta baixa para reconhecer pontos de entrada e saída, como portas e janelas, além de corredores, salas e obstáculos, e então traduzir essas informações em uma estrutura de grafo que represente adequadamente o *layout* e as conexões do ambiente.

Já existem trabalhos que exploram a conversão de imagens de plantas baixas de edifícios em grafos, muitos deles utilizando IA. O trabalho [47] propõe o uso de segmentação semântica por meio de aprendizado profundo para reconhecer características das plantas, como paredes, portas e escadas, e transformá-las em grafos com base em regras pré-definidas. Eles utilizam o conjunto de dados LIFULL HOME's [48], voltado para imagens de alta resolução de imóveis para locação. Embora o método alcance uma precisão de 92%, ele é testado e aplicado apenas em plantas baixas com baixo nível de complexidade e em ambientes pequenos, o que não reflete a realidade de análises em plantas baixas maiores e mais complexas.

Dois projetos interessantes que surgiram nessa área foram o CubiCasa5k [49] e o CubiGraph5k [50]. O CubiCasa5k foi desenvolvido para criar um dataset para o treinamento de redes neurais profundas, visando a conversão de imagens matriciais de plantas baixas em polígonos codificados que representam os diferentes ambientes presentes na planta. Esse dataset contém 5000 imagens anotadas e foi gerado a partir de uma arquitetura convolucional de encoder-decoder.

O projeto CubiGraph5k [50] utilizou o dataset CubiCasa5k para converter plantas baixas rasterizadas em uma representação de grafo, aplicando o conceito de adjacência entre os ambientes e conexões, como portas. Embora esses projetos sejam promissores, eles apresentam limitações significativas, principalmente pela utilização apenas de plantas baixas pequenas de casas ou apartamentos, o que reduz a complexidade e a capacidade de generalização das soluções propostas. Além disso, ambos os projetos não foram atualizados nos últimos anos, o que limita sua capacidade de lidar com desafios mais complexos e com plantas baixas maiores, como as de edifícios comerciais ou grandes instalações, necessárias para avanços mais significativos na área.

As soluções baseadas em *softwares* de criação de plantas baixas específicas apresentam resultados promissores [51]. No entanto, esses métodos geralmente não utilizam imagens matriciais como entrada, o que limita sua aplicabilidade em cenários onde apenas plantas digitalizadas ou fotografadas estão disponíveis.

Uma revisão de literatura abrangente foi conduzida em [52], onde foram identificadas diversas abordagens e conjuntos de dados disponíveis para esse contexto. A revisão destacou que algumas soluções empregam métodos baseados em regras, especialmente aquelas que utilizam arquivos CAD como base, enquanto outras exploram o Aprendizado de Máquina, com ênfase em aprendizado profundo e modelos

generativos. Um dos principais pontos de conclusão deste estudo é coerente com os desafios já mencionados; a maioria dos conjuntos de dados utilizados para avaliar esses métodos são compostos por plantas de casas e apartamentos, sem considerar plantas mais complexas, como escritórios, edifícios públicos, universidades e hospitais.

A conversão automática de uma imagem de planta baixa em um grafo é um desafio complexo e não trivial. Considerando que o foco deste trabalho é a aplicação de Aprendizado por Reforço em ambientes fechados representados por grafos, optou-se por uma abordagem manual para essa representação. Para isso, foi desenvolvida uma interface que permite ao usuário carregar a imagem da planta baixa e sobrepor o grafo diretamente a ela. Embora esse processo seja mais trabalhoso, ele tende a proporcionar resultados mais precisos em comparação com abordagens automatizadas previamente exploradas.

Importante destacar que a solução adotada baseia-se na metodologia apresentada em [1], a qual foi aprimorada com melhorias pontuais para adequar-se às modificações introduzidas com a evolução do algoritmo de Aprendizado por Reforço empregado neste estudo.

Para concluir, a Tabela 3.1 apresenta uma análise comparativa que destaca as características dos estudos analisados, ressaltando as inovações e vantagens do trabalho aqui proposto.

Tabela 3.1: Comparação entre trabalhos relacionados e o presente estudo

Estudo	**Abordagem / Método**	**Contexto / Ambiente**
ZHU, Y.; ZHANG, [38]	Algoritmo A* aprimorado com múltiplos fatores ambientais, baseado em imagens de satélite	Incêndios florestais e ambientes abertos de grande escala
HABOURY [39]	Aprendizado supervisionado híbrido clássico-quântico (FiLM + rede quântica)	Evacuação veicular em cidades após terremotos, ambiente dinâmico e incerto
SRIKANTH [40]	Aprendizado por Reforço com sensorística embarcada (LiDAR, sonar)	Incêndios estruturais com baixa visibilidade (robótica móvel)
XU, L.; HUANG, [41]	Otimização por Colônia de Formigas aprimorada (IACO)	Ambientes fechados, como supermercados, com monitoramento em tempo real
AGNIHOTRI, A [42]	Combinação de A* com KNN para pré-seleção de saídas	Ambientes fechados com diferentes plantas baixas
SHARMA, J [43]	Deep Q-Learning com Transfer Learning	Ambientes fechados simulados como grafos e obstáculos
HAN, L.; GUO, H. [44]	Algoritmo de Dijkstra com agrupamento e controle de tempo para evitar congestionamentos	Ambientes fechados com grande fluxo de pessoas
ROSA, A. C. [1]	Aprendizado por Reforço profundo com *transfer learning* e interface manual para construção de grafos a partir de plantas baixas	Ambientes fechados com grafos desenhados manualmente
Trabalho Proposto	Aprendizado por Reforço profundo com inferência para múltiplos cenários e interface manual para construção de grafos a partir de plantas baixas	Ambientes fechados com grafos desenhados manualmente, com obstáculos dinâmicos (fogo e congestionamento)

4 ARQUITETURA DA PROPOSTA

A proposta deste trabalho é dar continuidade à pesquisa realizada em [1], que propôs o uso de Aprendizado por Reforço Profundo com *transfer learning* para identificar, em tempo real, as saídas de emergência em casos de incêndio em um edifício utilizando sua planta baixa. No entanto, algumas limitações foram identificadas, especialmente quanto à necessidade de empregar Aprendizado por Reforço Profundo em um ambiente relativamente simples. Além disso, o modelo apresentava dificuldades em cenários com grafos maiores e mais complexos.

Para superar essas limitações e reforçar a aplicabilidade do Aprendizado por Reforço nesse contexto, este trabalho propõe uma abordagem aprimorada, introduzindo um ambiente de aprendizado mais dinâmico. A nova abordagem considera fatores críticos, como gargalos e nós atingidos pelo fogo, demonstrando que o Aprendizado por Reforço pode ser eficaz mesmo em cenários mais complexos e realistas. Além disso, enquanto o estudo anterior utilizava plantas baixas mais simples e com menor número de obstáculos, esta pesquisa amplia a complexidade do ambiente, permitindo a aplicação do modelo em plantas maiores e mais detalhadas, tornando a solução mais escalável e próxima de cenários reais.

A metodologia utilizada em [1] consistia no uso de uma rede MLP (*Multilayer Perceptron*) para representar o modelo de Aprendizado por Reforço Profundo. O estudo explorava principalmente a técnica de *transfer learning*, onde um agente era previamente treinado para aprender todas as rotas de fuga em um ambiente sem bloqueios. Em um momento de necessidade real, essa técnica permitia que, em poucos segundos ou minutos, o agente fosse ajustado para evitar os nós atingidos pelo fogo; porém, a cada modificação do ambiente, o agente precisava ser retreinado.

Diferentemente de [1], este estudo propõe uma versão aprimorada da abordagem anterior, utilizando uma rede neural profunda e estruturada para capturar melhor as complexidades do ambiente. Essa melhoria adiciona camadas e ajusta os hiperparâmetros utilizados, buscando otimizar o desempenho do agente no processo de Aprendizado por Reforço. Além disso, a ampliação do ambiente de teste para plantas baixas maiores e mais complexas permite avaliar a escalabilidade da abordagem e sua viabilidade em cenários reais de evacuação.

Uma das principais inovações proposta, em comparação com o apresentado em [1], é a introdução de uma maior complexidade no ambiente. Nessa abordagem, simulamos gargalos por meio de agentes virtuais que se movimentam pelo grafo, representando o alto fluxo de pessoas em determinados nós. Dessa forma, o agente treinado aprende a evitar nós congestionados e, sempre que possível, optar por caminhos alternativos que promovam uma evacuação mais eficiente.

Uma ressalva importante é que os agentes virtuais, que simulam o fluxo elevado de pessoas, não são treinados e não possuem inteligência própria. Eles se movimentam pelo grafo de acordo com critérios pré-estabelecidos, funcionando apenas como obstáculos dinâmicos que simulam gargalos. Assim, o foco do trabalho é o treinamento de um único agente, que aprende a evitar esses nós congestionados e a escolher caminhos alternativos, enquanto os agentes virtuais atuam de maneira fixa e determinística.

Para o desenvolvimento do código de Aprendizado por Reforço Profundo, foi utilizada principalmente

a linguagem de programação Python [53], devido à sua ampla adoção na comunidade científica e suporte a diversas bibliotecas para aprendizado de máquina. A implementação foi baseada na biblioteca PyTorch [54], uma estrutura de código aberto utilizada para computação em tensores e construção de redes neurais, oferecendo flexibilidade e eficiência no treinamento de modelos de aprendizado profundo.

4.1 ARQUITETURA GERAL

A arquitetura proposta é apresentada na Figura 4.1, composta por quatro camadas principais: apresentação, aplicação, persistência e inteligência. Por meio dessa estrutura, o usuário pode cadastrar o grafo correspondente à imagem de uma planta baixa do ambiente, utilizando a interface front-end desenvolvida. Ao submeter esse projeto à API, os dados são armazenados nos serviços de persistência. Com esses dados, o usuário pode iniciar o treinamento de um agente de aprendizado, cujo modelo resultante (pesos treinados) é salvo para uso posterior de inferência, conforme as configurações definidas.

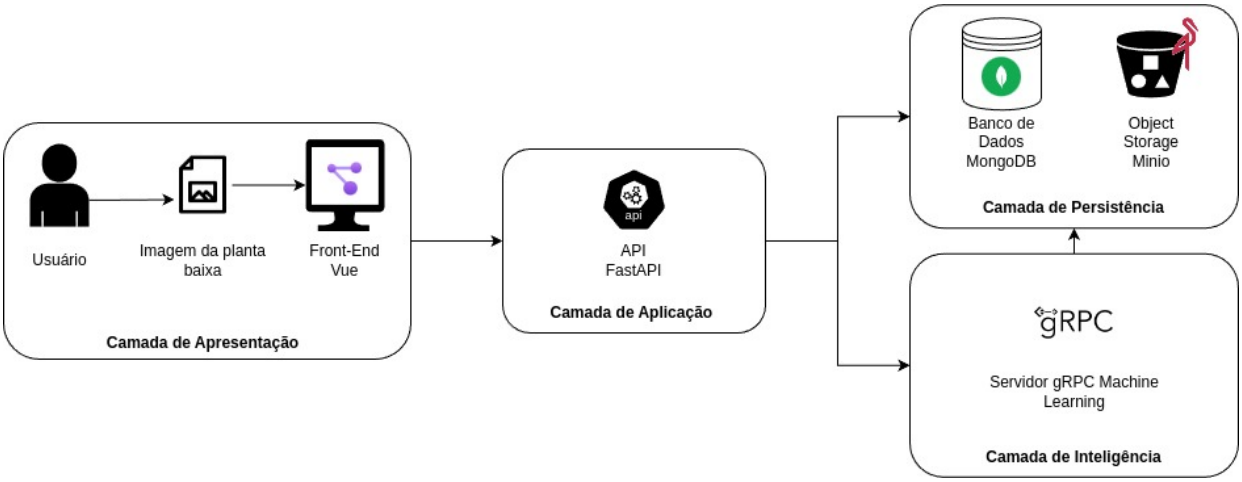

Figura 4.1: EvacuAI - Arquitetura Proposta.

O código-fonte da interface front-end está disponível em [55], o código da API pode ser acessado em [56] e o servidor gRPC responsável pelo treinamento do agente está disponível em [57]. Os serviços empregados na camada de persistência foram executados em *containers* Docker [58] e não possuem instâncias públicas acessíveis, estando disponíveis apenas em suas implementações oficiais.

4.1.1 Camada de Apresentação

A camada de apresentação é responsável pela interação do usuário com o sistema, composta por um front-end desenvolvido em Vue.js [59]. O Vue.js é um framework para construção de interfaces de usuário, que se destaca por sua simplicidade, reatividade e facilidade de integração com outras bibliotecas. Essa interface permite tanto o cadastro de um novo projeto com sua planta baixa correspondente, quanto o treinamento e a execução de inferências em projetos previamente criados.

Para a hospedagem e disponibilização da interface front-end, foi utilizada a plataforma Vercel vercel, que permite o deploy contínuo de aplicações web estáticas ou dinâmicas. Sua utilização se deve ao fato de

ser facilmente integrável com repositórios Git. A aplicação desenvolvida pode ser acessada em [60].

É importante observar que a API, que será apresentada na próxima Subseção, não está hospedada publicamente e, portanto, para realizar testes completos utilizando o front-end, é necessário hospedar localmente a API e indicar seu caminho no campo `API Endpoint` da interface.

A Figura 4.2 apresenta a página inicial da interface front-end, onde o usuário pode iniciar a criação, inferência ou treinamento de projetos.

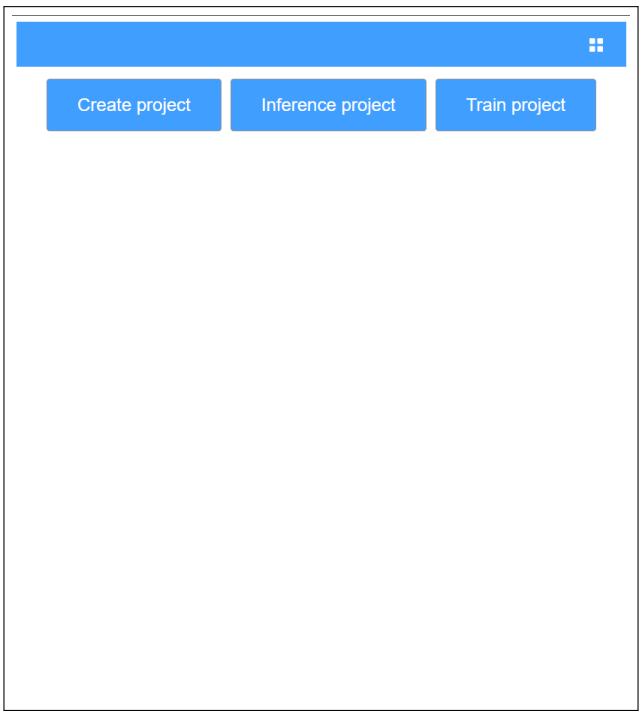

Figura 4.2: Página inicial do front-end.

4.1.1.1 Criar um novo projeto

Para o desenvolvimento da funcionalidade de criação de um novo projeto, foi utilizada principalmente a biblioteca v-network-graph [61], sendo desenvolvida para Vue.js e voltada para a visualização e edição interativa de grafos diretamente no navegador. Ela permite que nós e arestas sejam criados, movidos e configurados de forma dinâmica, sendo especialmente útil para a modelagem de plantas baixas em forma de grafo. Essa biblioteca facilita tanto a usabilidade quanto a flexibilidade na construção do grafo pelo usuário.

A criação do grafo é realizada manualmente. Portanto, ao enviar uma imagem da planta baixa, o usuário deve desenhar o grafo na interface web, indicando os nós e as arestas. É responsabilidade do usuário informar corretamente todas as conexões entre os nós e indicar os nós que representam saída. Esse processo está ilustrado na Figura 4.3.

No momento em que o usuário submete um novo projeto, é realizada uma requisição HTTP do tipo `POST` para a rota `/create` da API. Nessa requisição, são enviados tanto a imagem da planta baixa quanto as informações do grafo, extraídas pela biblioteca v-network-graph. Além disso, são incluídos dados adi-

cionais do projeto, como o nome e outros metadados relevantes.

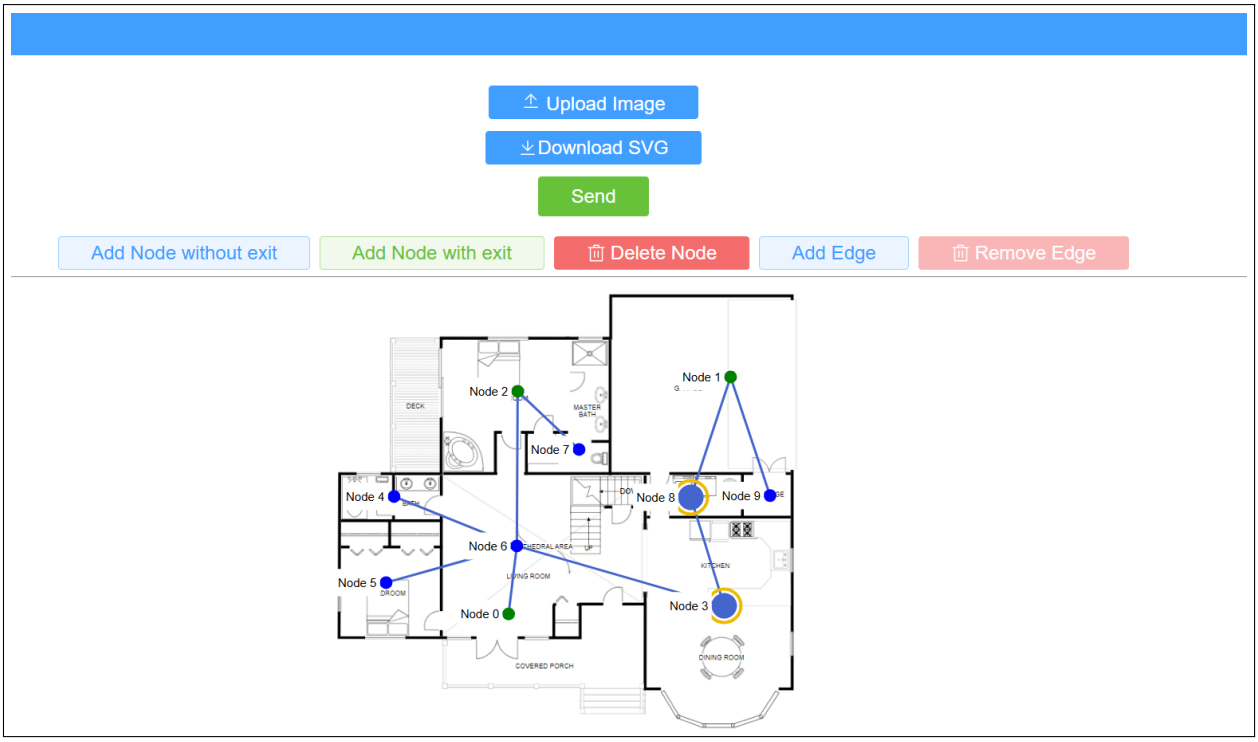

Figura 4.3: Página para o usuário desenhar o grafo referente a planta baixa.

Neste trabalho, todos os nós dos grafos são identificados por rótulos numéricos únicos. As arestas são bidirecionais, ou seja, há conectividade em ambos os sentidos entre nós adjacentes.

4.1.1.2 Treinamento de um projeto

Após a criação de um projeto com o grafo correspondente à planta baixa do ambiente, é possível realizar o treinamento de um agente de aprendizado por reforço nesse cenário. Para isso, foi implementada uma página que permite ao usuário selecionar o projeto desejado e fornecer os hiperparâmetros do treinamento. Um mesmo projeto pode ter diversos treinamentos diferentes, sendo útil principalmente para testes de diferentes hiperparâmetros. Cada treinamento será armazenado separadamente e com um identificador único.

Portanto, quando o usuário submete um novo treinamento, uma requisição HTTP do tipo `POST` é realizada para a rota `/train_project` da API. O corpo da requisição deve conter um objeto `JSON` com os campos necessários, como o project_id e os hiperparâmetros necessários. Caso os hiperparâmetros não sejam especificados, será utilizado os valores padrão para iniciar o processo de treinamento.

O treinamento é realizado de forma assíncrona. Isso significa que, ao enviar a requisição de treinamento, um identificador único (ID) é gerado e retornado ao usuário. Quando o treinamento for concluído, os pesos do modelo treinado serão salvos na camada de persistência. A partir desse momento, o usuário poderá utilizar o ID para realizar inferências com o modelo treinado.

A Figura 4.4 ilustra o processo necessário para iniciar o treinamento de um agente no ambiente/projeto

selecionado.

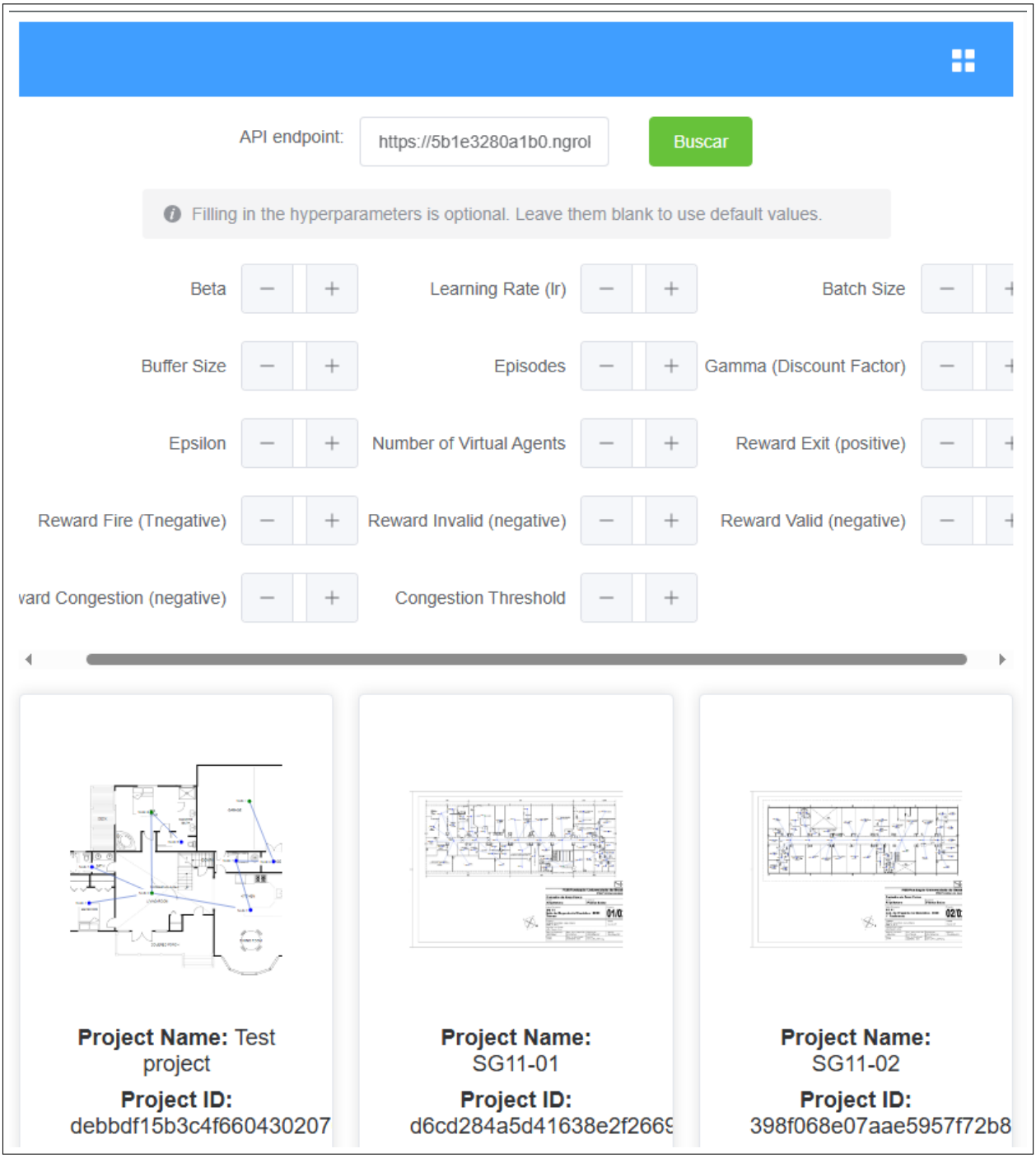

Figura 4.4: Página para iniciar o treinamento de um agente de aprendizagem.

4.1.1.3 Inferência de um projeto

Após a criação e o treinamento de um projeto, é possível realizar inferências utilizando os pesos de um treinamento específico. Para isso, o usuário deve selecionar o projeto e o identificador único (*ID*) do treinamento desejado. A inferência pode ser configurada conforme o objetivo específico da análise, sendo

possível definir parâmetros opcionais, como um nó inicial, nó em chamas e os nós congestionados. Com essas informações, o sistema executa o agente no ambiente configurado e retorna os caminhos percorridos durante a inferência. Caso nenhum nó inicial seja especificado, será feita a inferência do agente a partir de todos os nós do grafo, exceto os nós de saída.

Para a listagem de todos os projetos cadastrados na interface web, é utilizada a rota `GET` da API `/project`. Para exibir ao usuário todos os identificadores únicos (*IDs*) de treinamentos associados a um determinado projeto, utiliza-se a rota `GET /train_ids/{project_id}`. Por fim, para que o usuário visualize os resultados da inferência na interface, é utilizada a rota `GET /inference`.

Esse processo de inferência é ilustrado sequencialmente nas Figuras 4.5, 4.6 e 4.7.

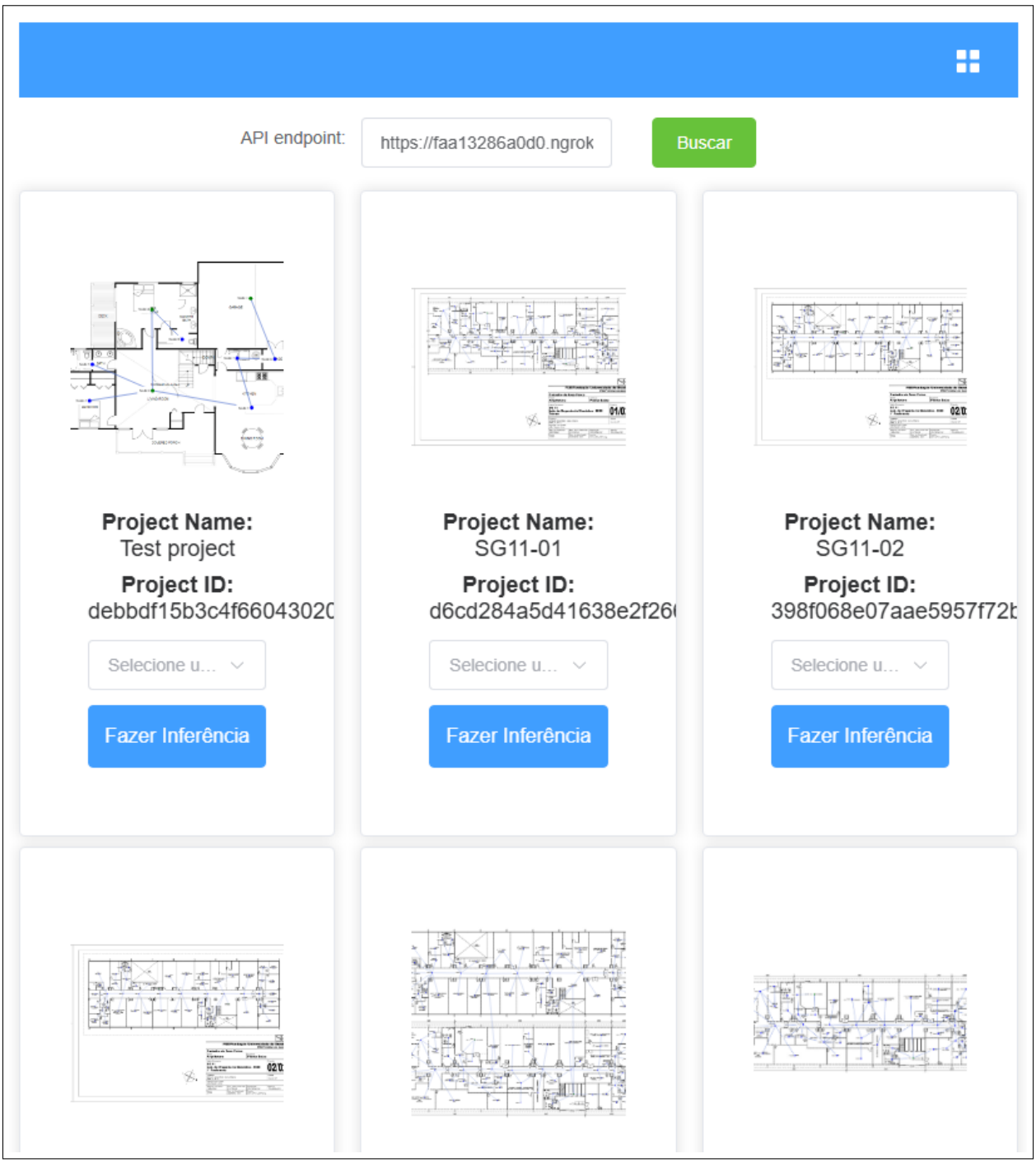

Figura 4.5: Página para escolha do projeto e identificador de treinamento na inferência.

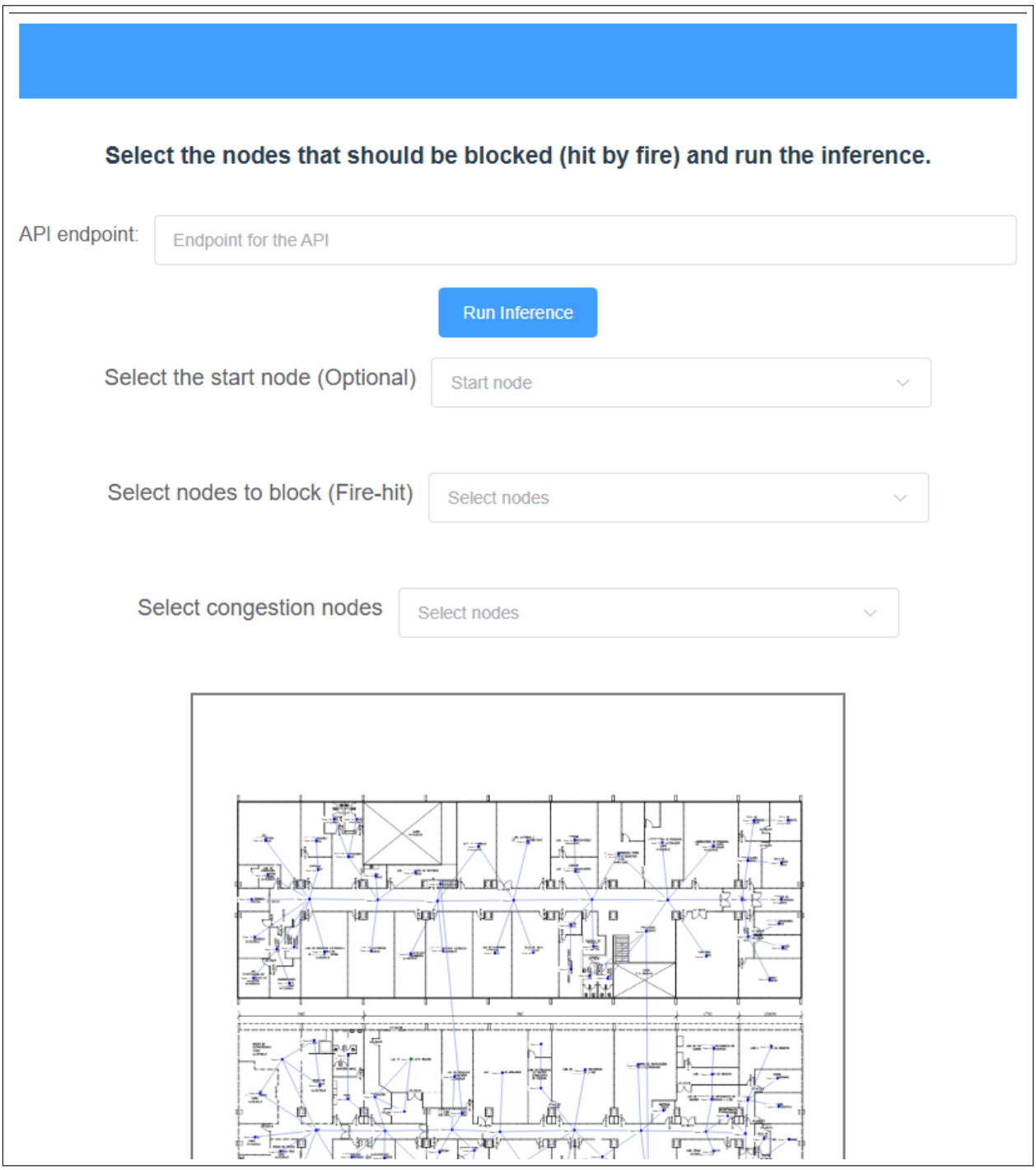

Figura 4.6: Página para a configuração do ambiente de inferência.

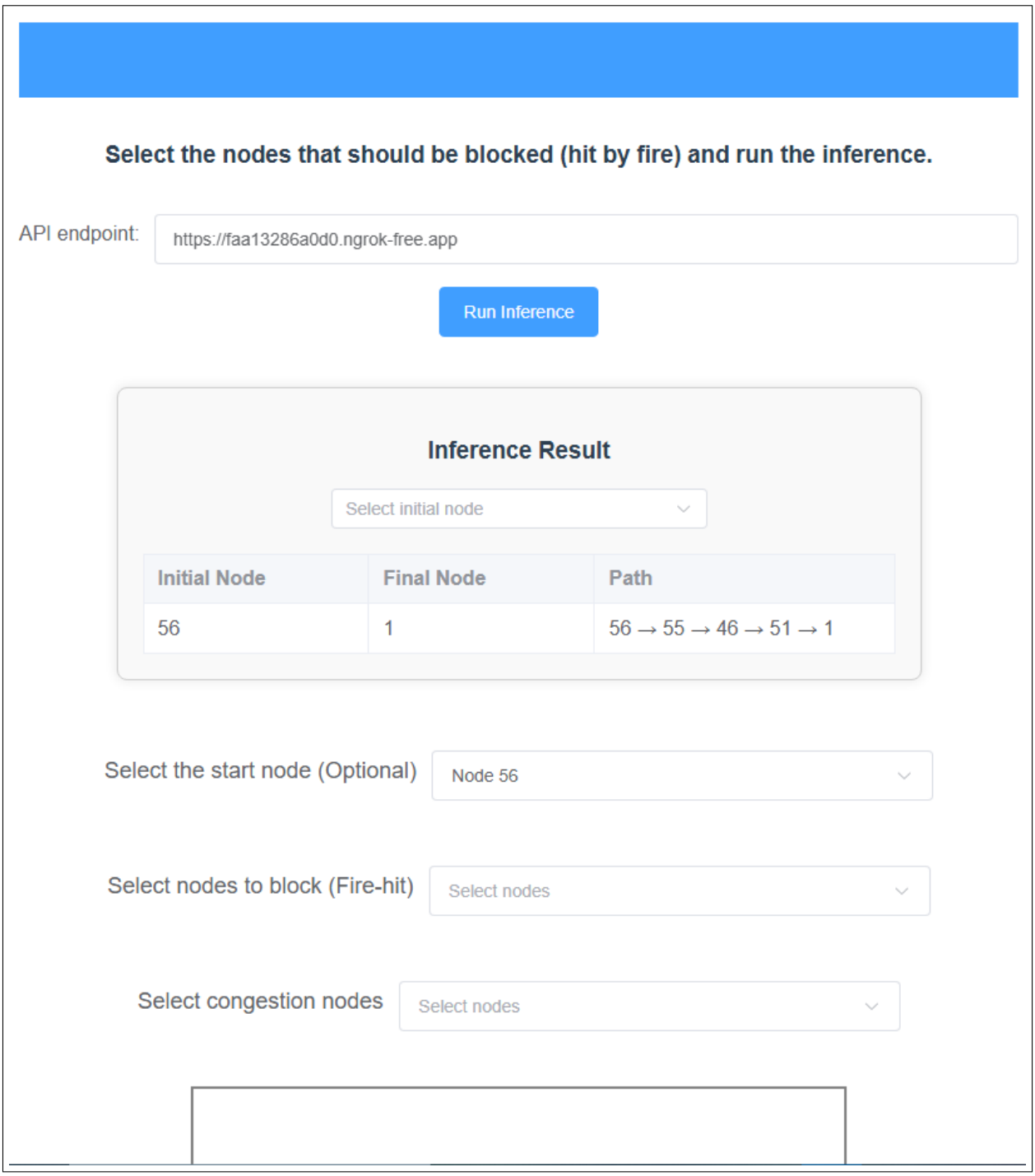

Figura 4.7: Página para visualização da inferência conforme a configuração escolhida.

4.1.2 Camada de Aplicação

A camada de aplicação é composta por uma API (*Application Programming Interface*) REST desenvolvida com FastAPI [62], um *framework* para a construção de APIs com a linguagem de programação Python, baseada nos padrões do OpenAPI.

Essa API se comunica com todas as camadas da arquitetura proposta. Ela é utilizada pelo front-end

para buscar e salvar informações, pela camada de persistência para armazenar e resgatar os dados dos projetos, e pela camada de inteligência para iniciar treinamentos e realizar inferências.

Para atender a essas funcionalidades, a API expõe rotas específicas que dão suporte completo à integração entre as camadas. As principais rotas disponíveis são:

- /create `[POST]`: cria um novo projeto;
- /inference `[GET]`: retorna o resultado de uma inferência;
- /projects `[GET]`: lista todos os projetos cadastrados;
- /train_ids/project_id `[GET]`: retorna todos os train_ids (IDs de treinamento) associados a um projeto;
- /train_project `[POST]`: inicia o treinamento de um projeto.

Essas rotas e sua organização podem ser visualizadas na Figura 4.8.

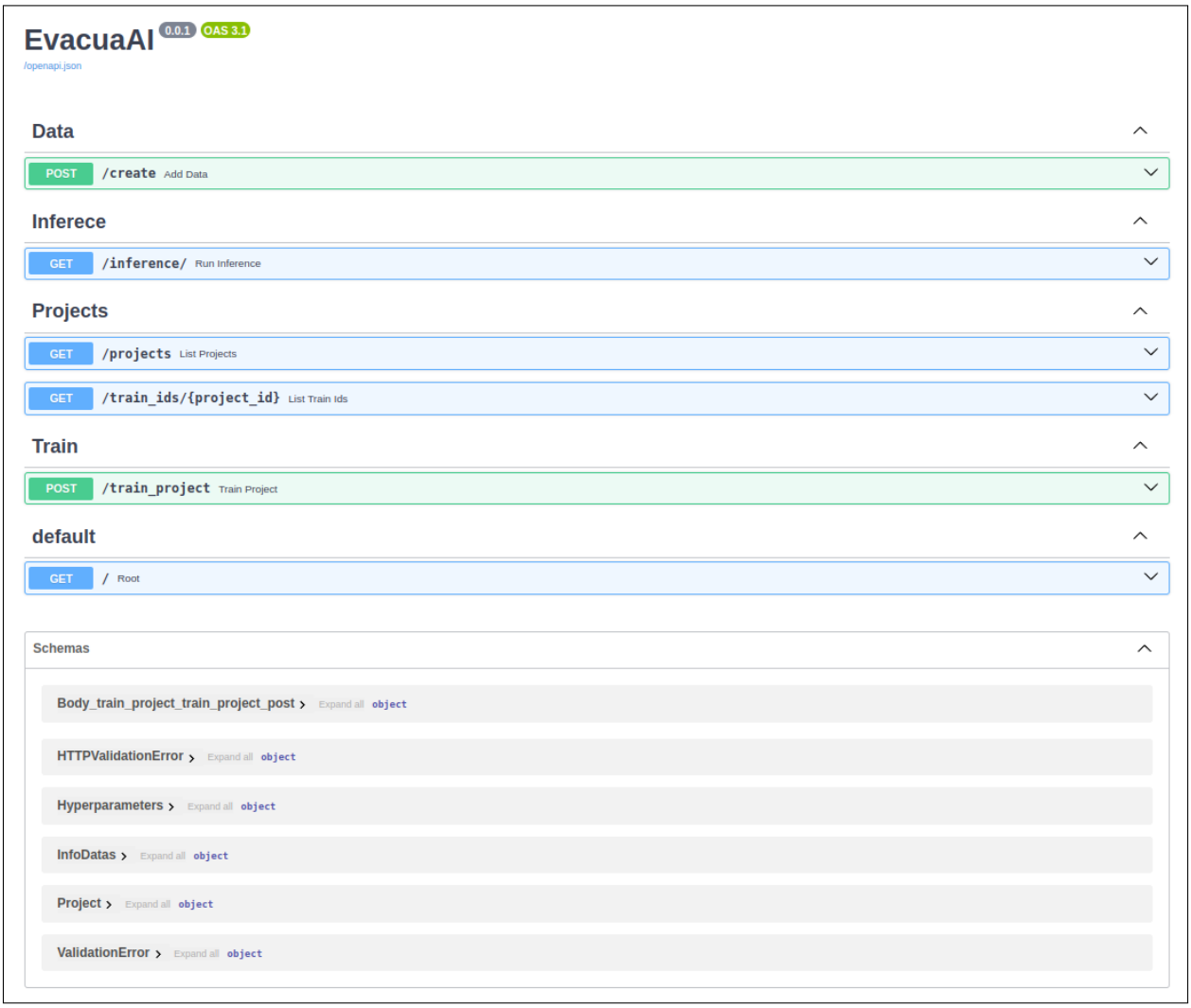

Figura 4.8: Documentação da API desenvolvida.

Para a inicialização da API, é necessário estabelecer conexões tanto com o banco de dados MongoDB, que será utilizado para o armazenamento dos dados, quanto com o serviço de *Object Storage*, responsável por salvar e recuperar informações relacionadas aos projetos. Para isso, foram utilizadas as bibliotecas Python pymongo e minio. Essas conexões são abertas durante a inicialização da API e fechadas no momento da finalização. Esses serviços serão explicados na próxima subseção.

Além disso, é preciso estabelecer conexão com o servidor gRPC da camada de inteligência para possibilitar o início e a inferência dos projetos. Para essa comunicação, foram utilizadas as bibliotecas grpcio e grpcio-tools.

4.1.2.1 `/create`

A rota `/create` é utilizada pelo *front-end* para a criação de um novo projeto. Para isso, é necessário o envio dos seguintes dados: *project_name*, *image*, *nodes*, *edges* e *layouts*.

Os campos *nodes*, *edges* e *layouts* referem-se às estruturas criadas quando o usuário constrói um grafo utilizando a biblioteca NetworkX do VueJs. O campo *nodes* representa os nós do grafo, o campo *edges* representa as arestas, e o campo *layouts* define a disposição espacial dos nós para fins de visualização, como o posicionamento em um plano bidimensional.

Na criação do projeto, foi implementada uma função para o cálculo da distância entre nós adjacentes, utilizando as informações contidas no campo *layouts*. O cálculo da distância é feito de forma simples, considerando a raiz quadrada da soma dos quadrados das diferenças entre as coordenadas x e y de dois nós, conforme a Equação 4.1.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{4.1}$$

Uma observação importante é que, neste trabalho, não é utilizada a escala real da planta do edifício, uma vez que não foi realizada nenhuma inspeção presencial para obtenção das medidas físicas e nenhuma extração dessas informações da planta baixa foi realizada. Dessa forma, as distâncias calculadas estão relacionadas apenas às posições em *pixels* da imagem da planta baixa. Esta limitação representa um dos desafios a serem enfrentados caso se deseje aplicar o projeto em um ambiente com escala real, o que não é considerado nesta etapa do desenvolvimento.

O Listing 1 do Apêndice I.2 mostra como foi implementada, em Python, a rota para a criação de um projeto.

4.1.2.2 `/train_project`

Esta é outra rota crucial para o projeto, pois é por meio dela que se pode iniciar o treinamento de um projeto. A partir do momento em que um projeto é criado, ele já pode ser treinado.

Para realizar o treinamento, é necessário informar qual projeto será utilizado através do campo `project`, assim como os hiperparâmetros através do campo `hyperparameters` que serão aplicados nesse treina-

mento. Todos os hiperparâmetros são opcionais; caso o usuário não os envie, serão utilizados os valores padrão.

Quando o usuário envia o `project_id` do experimento, esse valor é enviado com os hiperparâmetros recebidos para o servidor gRPC. Esse servidor é responsável por acessar a camada de persistência e obter as informações necessárias para realizar o treinamento. O Listing 2 do Apêndice I.2 mostra como foi implementada em Python a rota para o treinamento de um projeto.

4.1.2.3 `/inference`

Após a realização do treinamento, é possível fazer a inferência de um projeto em uma versão específica do treino, informando na rota os campos `project_id`, `version` (versão do treinamento) e `state` (caso deseje especificar um nó inicial). Caso o campo `state` não seja informado, a inferência será realizada em todos os nós. Além disso, é necessário enviar o campo `fire_nodes`, caso existam nós de fogo, e o campo `congestion_nodes`, caso existam nós de congestionamento.

Com esses dados, é enviado uma requisição para o servidor gRPC, que retorna os caminhos para a configuração do ambiente criada. O Listing 3 do Apêndice I.2 mostra como foi implementada em Python a rota para a inferência de um projeto.

4.1.2.4 `/projects` e `/train_ids/project_id`

Essas duas rotas são rotas auxiliares de leitura, criadas apenas para retornar ao *front-end* e usuários informações armazenadas na camada de persistência. A rota `/projects` retorna todos os projetos anteriormente criados, e a rota `/train_ids/project_id` retorna todos os IDs (`train_ids`) de treinamento que já foram realizados para cada projeto.

O Listing 4 do Apêndice I.2 mostra como foi implementada em Python a rota para a listagem de um projeto, enquanto o Listing 5 do Apêndice I.2 mostra como foi implementada em Python a rota para a listagem de IDs de treinamento de um projeto.

4.1.3 Camada de Persistência

A camada de persistência é composta por dois componentes principais: um banco de dados não relacional MongoDB [63] e um sistema de armazenamento de objetos MinIO [64].

O MongoDB é um banco de dados orientado a documentos, baseado no modelo NoSQL, que armazena os dados em formato JSON (ou BSON internamente). Ele é especialmente indicado para aplicações que exigem flexibilidade na estrutura dos dados, alto desempenho e escalabilidade horizontal. Neste sistema, são armazenadas as informações relacionadas aos projetos.

Os dados persistidos no banco de dados incluem:

- project_id: identificador único do projeto;

- project_name: nome do projeto informado pelo usuário;
- image_s3_uri: caminho no MinIO para a imagem da planta baixa com o grafo desenhado;
- graph_s3_uri: caminho no MinIO para o arquivo JSON contendo as informações do grafo, como nós, arestas e distâncias;
- create_time: data de criação do projeto;
- experiment_s3_uri: caminho para os pesos treinados do último experimento do projeto (caso exista);
- update_time: data da última atualização do projeto.

A Figura 4.9 ilustra a estrutura dos documentos conforme visualizada no MongoDB Compass, uma ferramenta gráfica oficial fornecida pela MongoDB Inc. [63] que permite a visualização, consulta e manipulação de dados armazenados no banco.

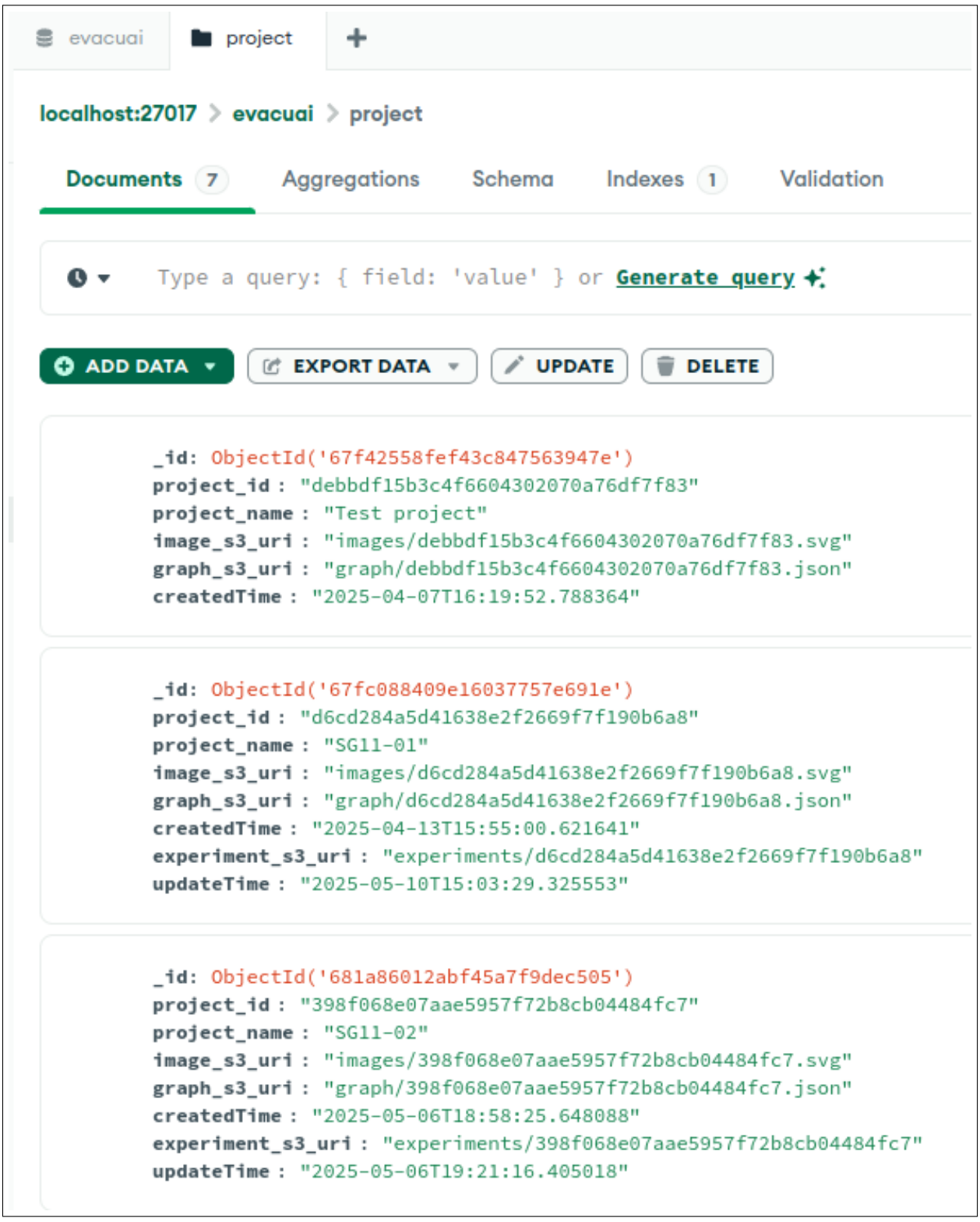

Figura 4.9: Documentos no MongoDB Compass.

O MinIO é um sistema de armazenamento de objetos compatível com a API do Amazon S3, projetado para armazenar grandes volumes de dados não estruturados, como arquivos e modelos. Ele é leve, de alto desempenho e amplamente utilizado em aplicações que demandam escalabilidade e integração com

sistemas distribuídos.

Neste trabalho, o bucket do MinIO foi organizado em três diretórios principais: *graph*, *experiments* e *images*, com funções distintas:

- *graph*: armazena arquivos JSON que contêm as informações do grafo, como nós, arestas, distâncias entre os nós e metadados relevantes para o treinamento e construção do ambiente. Cada arquivo recebe como nome o *project_id*.
- *experiments*: armazena os resultados dos treinamentos, incluindo o peso do modelo treinado (formato `.pth`), os logs do processo (formato `.pkl`) e a configuração utilizada no treinamento (formato `.json`). A estrutura de diretórios segue o padrão `/project_id/train_id`.
- *images*: guarda as imagens dos projetos no formato SVG, com o grafo desenhado sobre a planta baixa. O nome de cada imagem SVG corresponde ao *project_id*.

Essa organização facilita o acesso e o versionamento, mantendo os arquivos de cada projeto agrupados de forma lógica e eficiente dentro do ambiente de armazenamento. Essa estrutura pode ser visualizada na Figura 4.10.

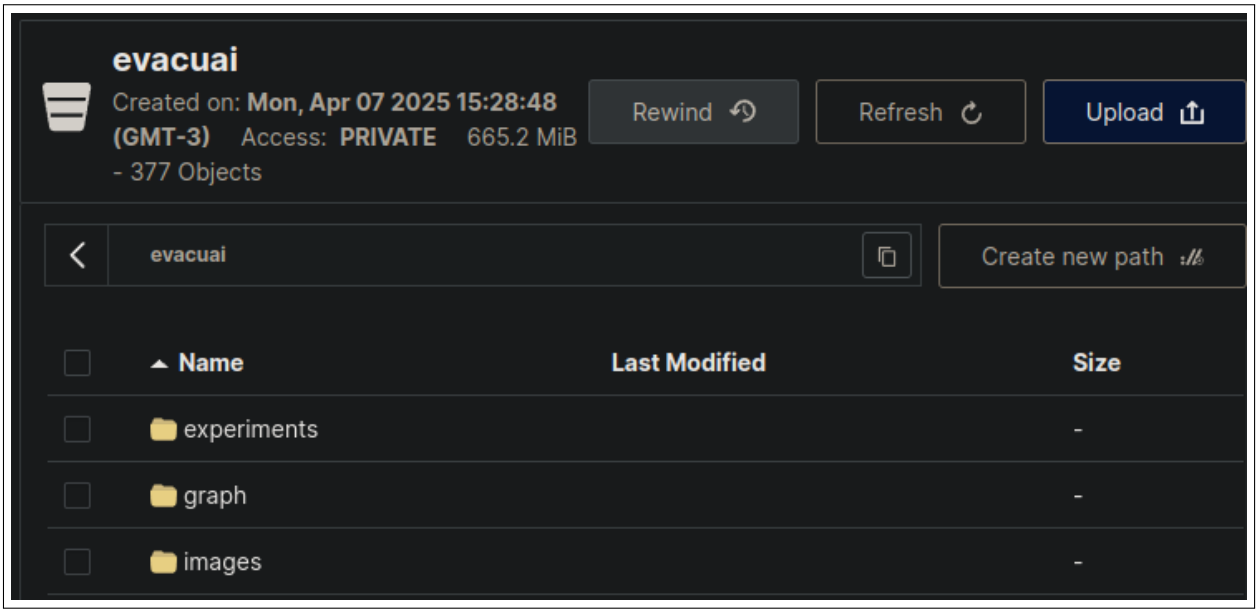

Figura 4.10: Estrutura de arquivos no *object storage*.

Para todos os experimentos que serão apresentados, tanto o MongoDB quanto o MinIO foram executados como containers Docker, utilizando suas imagens oficiais. Dessa forma, não há qualquer dado público disponível, uma vez que todos os dados utilizados foram gerados localmente no ambiente de testes.

4.1.4 Camada de Inteligência

A camada de inteligência é responsável pelo treinamento do agente de Aprendizado por Reforço. Os detalhes de sua implementação serão apresentados na próxima Seção. Para facilitar a comunicação entre

essa camada e a API, optou-se pelo uso do gRPC [65], um framework de chamada de procedimento remoto (RPC) de alto desempenho. Sendo assim, um servidor gRPC pode ser executado em uma máquina com maior capacidade computacional e receber requisições de treinamento e inferência a partir de seu cliente, o qual é a API REST apresentada anteriormente.

O Listagem 6 do Apêndice I.3 apresenta a definição do serviço gRPC utilizado para comunicação entre a API REST e a camada de inteligência. O serviço 'ReinforcementLearning' possui três métodos principais: 'TrainModel', 'Inference' e 'GetRewardPath', responsáveis, respectivamente, por iniciar o treinamento, realizar a inferência e calcular a recompensa de um caminho proposto.

4.2 MODELAGEM DO AMBIENTE DE APRENDIZAGEM

A geração de um grafo a partir de uma planta baixa é um desafio complexo. Neste trabalho, utilizamos o mesmo processo desenvolvido em [1], que implementou uma interface web permitindo ao usuário enviar imagens nos formatos PNG, JPEG ou JPG e, a partir delas, desenhar um grafo sobre a imagem da planta baixa. Recomenda-se a leitura do trabalho [1] para uma compreensão mais aprofundada do processo.

No entanto, iterar diretamente sobre um grafo pode ser desafiador devido à forma como suas estruturas de dados são organizadas, o que dificulta o acesso sequencial e a modificação de seus elementos. Para contornar essa limitação, optou-se por utilizar a matriz de adjacência do grafo como ambiente. Essa abordagem permite uma iteração mais simples e eficiente, além de possibilitar a atribuição de recompensas (*rewards*) a cada interação do agente. Com base na matriz de adjacência do grafo, é possível identificar se o agente se moveu para um estado conectado ao seu atual (adjacente), o que facilita a definição de recompensas adequadas.

Adicionalmente, essa matriz armazena informações sobre as distâncias entre os nós em *pixels*, conforme definido anteriormente na Equação 4.1, permitindo que o agente seja treinado não apenas para evitar estados indesejados, mas também para sempre buscar o menor caminho, em conjunto com outros critérios de otimização definidos.

A teoria de grafos, que fundamenta essas abordagens, pode ser encontrada na Seção 2.6.

Dentro do ambiente, são criados agentes virtuais que não possuem inteligência própria e se movimentam de acordo com critérios pré-estabelecidos. Esses agentes simulam gargalos, contribuindo para que o agente de treinamento aprenda a evitar nós congestionados e, quando possível, opte por caminhos alternativos até a saída.

A cada episódio de treinamento, há uma probabilidade de que agentes virtuais de congestionamento sejam inseridos no ambiente. Quando essa inserção ocorre, os agentes são posicionados em nós localizados entre o estado atual do agente de treinamento e os nós de saída, garantindo que o agente, ao longo do treinamento, seja exposto a diversas combinações de movimentação possíveis ao tomar suas decisões. Dessa forma, o agente aprende a entender, com base nas configurações do ambiente, qual o caminho até a saída é mais vantajoso, evitando trajetórias que possam levar a situações de congestionamento.

Os congestionamentos considerados neste trabalho são fixos em cada episódio e não apresentam vari-

ações ou movimentações adicionais, sendo tratados como obstáculos estáticos.

Além dos agentes virtuais, os nós atingidos pelo fogo são inseridos como obstáculos adicionais. Diferentemente do trabalho [1], em que o agente precisava ser retreinado sempre que um nó em chamas é alterado, recebendo apenas uma recompensa altamente negativa como forma de penalização, neste estudo, os nós afetados pelo fogo são simulados ao longo de todo o treinamento com uma determinada probabilidade.

Esses nós de fogo são explicitamente informados na entrada da rede neural, o que permite que o agente aprenda a evitá-los não apenas por meio da penalização, mas também pela representação do estado. Dessa forma, a rede neural é capaz de identificar e desconsiderar trajetórias que envolvem nós em chamas, mesmo alterando o ambiente, sem a necessidade de retreinamento.

Neste trabalho, os nós de fogo são representados como uma lista, embora durante o treinamento apenas um nó seja selecionado por vez. No entanto, essa estrutura foi projetada para possibilitar, em trabalhos futuros, a consideração de múltiplos pontos de chamas simultâneos em um mesmo episódio dentro do ambiente.

As configurações do ambiente incluem: o grafo que representa a planta baixa, a lista de nós de saída, o número máximo de agentes virtuais e a definição dos hiperparâmetros de treinamento.

4.2.0.1 Parâmetros e Hiperparâmetros

Diversos parâmetros e hiperparâmetros são utilizados no treinamento do agente, sendo todos passíveis de configuração a cada novo treinamento.

Os parâmetros são variáveis que definem o ambiente e a configuração geral do experimento, mas não afetam diretamente o processo de aprendizado da política. São eles:

- Grafo: Estrutura formada por nós e arestas que representa a planta baixa do ambiente. (Obrigatório)
- Nós de saída: Conjunto de nós do grafo que representam as saídas do ambiente. (Obrigatório)
- Número de agentes virtuais: Quantidade máxima de agentes virtuais utilizados para simular congestionamento. Caso não se deseje simular congestionamento, o valor padrão é 0. (Opcional)

Os hiperparâmetros são variáveis definidas antes do treinamento que controlam o comportamento do algoritmo de aprendizado e influenciam diretamente o desempenho do modelo. São eles:

- Learning Rate (lr): Taxa de aprendizado da rede neural. Define o quanto os pesos são ajustados em cada atualização. (Padrão: 1e-2)
- Batch Size: Tamanho do lote de experiências amostradas da memória para atualização da rede. (Padrão: 240)
- Iterações: Número total de iterações que o agente realizará dentro do ambiente. (Padrão: 6000)
- Gamma (γ): Fator de desconto que controla a importância das recompensas futuras. (Padrão: 0.9)

- Epsilon (ϵ): Probabilidade de o agente explorar novas ações ao invés de explorar o conhecimento atual. (Padrão: 0.7)
- Recompensa Saída: Valor de recompensa a ser recebido quando o agente chega em uma saída, esse valor sempre será positivo. (Padrão: 10.000)
- Recompensa Fogo: Valor de recompensa a ser recebido quando o agente passa em um nó atingido por fogo, esse valor sempre será negativo. (Padrão: -10.000)
- Recompensa Inválido: Valor de recompensa a ser recebido quando o agente escolhe um nó que não é adjacente ao estado atual, esse valor sempre será negativo. (Padrão: -5.000)
- Recompensa Válido: Valor de recompensa a ser recebido quando o agente escolhe um nó adjacente ao seu estado atual, esse valor sempre será negativo. (Padrão: -1)
- Recompensa Congestionamento: Valor de recompensa a ser recebido quando o agente em um nó atingido por congestionamento, esse valor sempre será negativo. (Padrão: -300)

As recompensas foram definidas como hiperparâmetros. No entanto, recomenda-se que apenas uma situação resulte em recompensa positiva: quando o agente escolhe como próximo estado um nó de saída. Os demais casos, o agente deve ser penalizado. O objetivo é reforçar de forma clara o propósito final do experimento. Ainda assim, outras combinações de recompensas podem ser testadas para fins de avaliação e comparação.

4.2.1 Definição dos estados e ações

No início de cada episódio de treinamento, o agente recebe um estado inicial, que pode ser um nó previamente definido ou um nó selecionado aleatoriamente. Além disso, é definido se haverá inserção de agentes virtuais de congestionamento ou de algum nó em chamas durante o episódio.

Caso o estado inicial seja escolhido aleatoriamente, é mantida uma contagem de quantas vezes cada nó foi utilizado como ponto de partida. Com base nessa contagem, os nós menos visitados são priorizados de forma balanceada nas próximas escolhas aleatórias. Essa estratégia força o processo de exploração e explotação a ocorrer de maneira mais equilibrada entre todos os nós do grafo.

Cada episódio é composto por diversas interações distintas com o ambiente e é finalizado assim que o agente alcança um nó de saída. Ao longo de todo o episódio, as configurações dos agentes virtuais e os nós afetados pelo fogo permanecem inalterados. Apenas o estado do agente é atualizado a cada iteração, para refletir sua movimentação e as ações tomadas durante o treinamento.

A taxa de exploração (*epsilon*) é ajustada a cada iteração com o ambiente e determina a estratégia utilizada para escolher a próxima ação do agente. Se um valor aleatório p for menor que *epsilon*, a ação será escolhida de forma aleatória (exploração), respeitando a restrição de vizinhança do nó atual, ou seja, só poderá ser escolhido como próximo estado um nó adjacente ao estado atual. Caso contrário, a ação será determinada pela rede neural (explotação). O Algoritmo 3 ilustra o processo de tomada de decisão baseado na escolha entre exploração e explotação.

Caso a estratégia definida for a de explotação, a rede neural recebe como entrada um vetor que representa o estado atual do agente e retorna, como saída, um vetor representando cada nó do grafo. A ação é então escolhida com base no maior desses valores, utilizando a função *argmax*, que seleciona o índice do nó correspondente ao valor de ação mais alto indicado pela rede.

A entrada da rede neural representa o estado atual do agente, sendo um tensor que encapsula informações sobre o nó atual, os estados finais, os nós atingidos pelo fogo e os congestionamentos, definidos pela quantidade de agentes virtuais em cada nó do ambiente. O Algoritmo 3 mostra que a entrada da rede neural é definida por meio da função *state_to_vector*, que define o estado atual do agente com base nas características presentes no ambiente.

O estado atual é representado por um vetor cuja dimensão é 4 vezes o número total de nós no grafo, pois ele é composto por quatro partes distintas: nó atual, nós de saída, nós atingidos pelo fogo e congestionamento. As duas primeiras partes do vetor do estado atual indicam onde o agente está e onde ele deve chegar, enquanto as duas últimas representam os obstáculos que podem interferir em seu caminho.

- Nó atual: A primeira parte do vetor indica a posição atual do agente no grafo. Essa representação utiliza *one-hot encoding*, ou seja, apenas o valor que correspondente ao nó atual será 1, enquanto os demais serão 0.
- Nós de saída: A segunda parte do vetor é fixa e identifica os nós de saída. Cada nó de saída recebe o valor 1, enquanto os demais permanecem 0. Essa informação guia o agente para alcançar seu objetivo final.
- Nós atingidos pelo fogo: A terceira parte do vetor indica os nós que estão em chamas. Se um nó for afetado pelo fogo, ele recebe o valor 1, caso contrário, recebe 0.
- Definição de Congestionametos: A última parte do vetor representa a densidade de agentes virtuais em cada nó do grafo, que é considerada um gargalo quando o número de agentes em um nó ultrapassa um valor limite (*congestion_threshold*) previamente estabelecido. A densidade de agentes é normalizada para o intervalo [0,1], levando em consideração o número total de agentes presentes no ambiente. Dessa forma, a normalização permite uma comparação uniforme da carga de cada nó, com valores próximos de 1 indicando nós mais congestionados e valores próximos de 0 indicando nós menos congestionados.

A entrada da rede neural será o vetor definido na Equação 4.2. Essa estrutura fornece ao agente informações essenciais para a tomada de decisão, permitindo que ele identifique sua posição e o melhor caminho seguro até uma saída.

$$\mathbf{s_t} = [\text{nó atual}, \text{nós de saída}, \text{nós em chamas}, \text{nós congestionados}] \tag{4.2}$$

Quando um agente executa uma ação, é necessário verificar se essa ação leva a um nó adjacente, pois, embora na exploração seja sempre escolhido um nó adjacente, isso nem sempre ocorre na explotação. Caso o próximo estado corresponda a um nó adjacente, verifica-se se ele representa um nó em chamas, um nó

congestionado ou mesmo um nó de saída; caso contrário, o agente recebe uma penalização para entender que aquele nó não é adjacente ao estado atual.

Com base nessa verificação, uma recompensa é atribuída ao agente, e o estado do agente é atualizado para definição do próximo estado. A transição para o novo estado s_{t+1} é determinada pela ação escolhida, pois o agente só se moverá para um novo estado se a ação conduzir a um nó adjacente ao estado atual. Caso contrário, o agente permanecerá no estado atual s_t até que uma ação válida, que o mova para um nó adjacente, seja tomada. A transição de estados é demonstrada na Equação 4.3.

$$s_{t+1} = \begin{cases} \text{ação}, & \text{se a ação levar o agente a um nó adjacente} \\ s_t, & \text{caso contrário, o agente permanece no estado atual} \end{cases} \tag{4.3}$$

Algorithm 3 Exploração-Exploitation no Ambiente

Require: Entrada: $iteration$ ▷ Número total de iterações de treinamento
Require: Entrada: adj_mat ▷ Matriz de adjacência do grafo
Require: Entrada: $graph$ ▷ Objeto representando o grafo
Require: Entrada: $state_to_vector(state)$ ▷ Função para converter um estado inteiro em vetor
Require: Entrada: net ▷ Rede neural para aprendizado da política
Require: Entrada: $state$ ▷ Estado atual do agente
Ensure: Saída: $action$ ▷ Ação selecionada pelo agente

for $i \leftarrow 0$ **to** $iteration - 1$ **do**
    $\epsilon \leftarrow \exp\left(-\frac{i}{i/2}\right)$
    $p \leftarrow \text{random}(0, 1)$
    if $p \leq \epsilon$ **then**
        $vizinhos \leftarrow$ lista de nós v conectados a $state$ em adj_mat
        $action \leftarrow \text{random_choice}(vizinhos)$
    else
        $vector_state \leftarrow state_to_vector(\text{integer}(state))$
        $tensor_state \leftarrow \text{Tensor}(vector_state)$
        $action \leftarrow net.act(tensor_state)$ ▷ Escolhendo ação com valor máximo
    end if
end for
return $action$

4.2.2 Função de Recompensa

Sempre que um agente executa uma ação, seja por exploração ou explotação, é realizada uma avaliação para determinar a recompensa correspondente. Uma variável fundamental da função de recompensa é a recompensa imediata, denotada por r_t, atribuída assim que o agente escolhe uma ação.

Neste trabalho, a recompensa imediata é calculada com base em diversos critérios, sendo eles:

- Se a ação escolhida leva a um nó adjacente ao estado atual e, se sim, a distância entre esses nós.
- Se a ação resulta em um nó bloqueado, seja por congestionamento ou por fogo.

- Se a ação conduz o agente diretamente a um nó de saída.

A recompensa final é a soma das pontuações atribuídas a esses critérios. O cálculo segue as seguintes regras:

1. Se a ação leva a um nó atingido por fogo, a recompensa é definida pela variável *reward_fire* que por padrão é 10.000.
2. Se o próximo nó não estiver em chamas, mas não for adjacente ao estado atual, a recompensa é definida pela variável *reward_invalid* que por padrão é -5.000.
3. Se a ação leva a um nó adjacente, que não está em chamas e não é um nó de saída, a recompensa definida pela variável *reward_valid* que por padrão é -1 * a distância entre os dois nós.
4. Se a ação leva a um nó adjacente que é uma saída, a recompensa é definida pela variável *reward_exit* que por padrão é em 10.000.

Além disso, há um ajuste baseado no congestionamento do nó de destino. Caso a quantidade de agentes no nó ultrapasse o limite de congestionamento, um fator de penalização é aplicado. O congestionamento é definido pelo *Congestion Ratio* (CR), conforme a equação 4.4.

$$CR = \frac{\max(0, N_{dest} - T_{cong})}{\max(1, N_{total})} \tag{4.4}$$

onde:

- N_{dest} é o número de agentes no nó de destino.
- T_{cong} é o limite de congestionamento.
- N_{total} é o número total de agentes no ambiente, garantindo que nunca haja divisão por zero.

A penalização final por congestionamento é calculada conforme a equação 4.5.

$$R_{cong} = -\textit{reward_congestion} \times CR \tag{4.5}$$

onde R_{cong} representa a penalização de congestionamento. Essa penalização é somada à recompensa anteriormente estabelecida para refletir as recompensas em ambientes congestionados.

A escolha por recompensas parametrizadas, em vez de valores fixos, se dá pela possibilidade de ajustar o comportamento do agente conforme os objetivos do experimento realizado. Por exemplo, é possível penalizar mais fortemente o congestionamento ou o fogo e diminuir a penalização de movimentos válidos (que levam a nós adjacentes).

Neste trabalho, cada movimentação válida, ou seja, quando o agente está em um estado atual e seu próximo estado é adjacente, gera uma recompensa negativa proporcional à distância entre esses dois nós.

Assim, em grafos maiores, onde os trajetos até a saída são mais longos, o agente acumula penalizações mesmo ao seguir caminhos corretos.

O Algoritmo 4 apresenta o cálculo da recompensa imediata. Nele, a penalização por atravessar um nó em chamas é definida pela variável *reward_fire*; essa penalidade pode se tornar pequena e até mesmo irrelevante diante da recompensa negativa acumulada ao longo do caminho em grafos maiores, fazendo com que o agente considere atravessar o fogo como uma estratégia aceitável ou até mesmo não entenda o objetivo do obstáculo. Para evitar esse comportamento, *reward_fire* deve ser ajustada proporcionalmente ao tamanho do grafo.

Algorithm 4 Cálculo da Recompensa

Require: $state$: Estado atual do agente
Require: $action$: Ação tomada pelo agente
Require: $graph$: Grafo representando o ambiente
Require: $exit_nodes$: Conjunto de nós de saída
Require: $fire_nodes$: Conjunto de nós em chamas
Require: $congestion_threshold$: Limite para considerar congestionamento
Require: $agent_positions$: Lista de posições dos agentes
Require: $reward_exit$: Recompensa de saída
Require: $reward_fire$: Recompensa de fogo
Require: $reward_invalid$: Recompensa de movimento inválido
Require: $reward_valid$: Recompensa de movimento válido
Require: $reward_congestion$: Recompensa de congestionamento
Ensure: $reward$: Recompensa atribuída ao agente
Ensure: $done$: Indica se o episódio terminou
Ensure: $connected$: Indica se os nós são adjacentes

$new_state \leftarrow action$
$reward \leftarrow 0$
$done \leftarrow$ False
$connected \leftarrow$ **True** se new_state for vizinho de $state$, **senão False**
if $new_state \in fire_nodes$ **then**
    $reward \leftarrow reward_fire$
else if $connected =$ False **then**
    $reward \leftarrow reward_invalid$
else if $new_state \in exit_nodes$ **then**
    $reward \leftarrow reward_exit$
    $done \leftarrow$ True
else
    $dist \leftarrow reward_valid \times$ distância entre $state$ e new_state
    $reward \leftarrow -dist$
end if
$num_agents_new_node \leftarrow$ contagem de agentes em new_state
$total_agents \leftarrow \max(1,$ número total de agentes$)$
if $num_agents_new_node \geq congestion_threshold$ **then**
    $CR \leftarrow \frac{\max(0, num_agents_new_node - congestion_threshold)}{total_agents}$
    $reward \leftarrow reward - (reward_congestion \times CR)$
end if
return $reward, done, connected$

A função de recompensa utilizada neste trabalho irá considerar apenas a recompensa imediata demonstrada. No entanto, ao calcular os *targets* da *Q-Network*, além da recompensa imediata, é considerada a recompensa futura, levando em consideração a potencial evolução do estado do ambiente e as ações subsequentes; isso será demonstrado na subseção 4.2.4. Isso permite que o agente aprenda não apenas com as recompensas imediatas, mas também com as consequências de suas ações a longo prazo.

4.2.3 Arquitetura da Rede Neural (*Q-Network*)

A rede neural construída neste trabalho é composta por seis camadas totalmente conectadas (*fully connected*), com cinco camadas ocultas. Para promover maior estabilidade durante o treinamento, foram inseridas camadas de normalização e *dropout*. A normalização é aplicada em cada camada oculta para reduzir a variância na distribuição dos dados de entrada, o que contribui para uma convergência mais estável. Já o *dropout* funciona como uma técnica de regularização, prevenindo o *overfitting*, que também pode ocorrer em Aprendizado por Reforço, principalmente em ambientes simulados ou com pouca variação de estados, nos quais a rede pode memorizar trajetórias específicas em vez de generalizar.

A inicialização dos pesos é realizada por meio do método *Kaiming Uniform*, voltado para funções de ativação do tipo *Leaky ReLU*, resultando em uma melhor propagação dos gradientes e aceleração do processo de aprendizado. A Tabela 4.1 apresenta a arquitetura detalhada da rede neural adotada.

A entrada da rede neural é determinada pelo tamanho do estado de observação, que corresponde ao número total de nós multiplicado por 4, conforme detalhado na Subseção 4.2.1. Essa estrutura de entrada é composta por 4 partes distintas: a primeira representa o nó atual, a segunda representa os nós de saída, a terceira representa os nós atingidos por fogo e a quarta representa os congestionamentos. A saída da rede é composta pelo número de ações que o agente pode executar, que, neste caso, é igual à quantidade de nós presentes no ambiente.

Tabela 4.1: Arquitetura da Rede Neural proposta.

Camada	**Tamanho da Entrada**	**Tamanho da Saída**
Entrada	obs_size	512
Fully Connected (FC1)	512	512
Layer Normalization	512	512
Dropout (p=0.2)	512	512
Fully Connected (FC2)	512	256
Layer Normalization	256	256
Dropout (p=0.2)	256	256
Fully Connected (FC3)	256	128
Layer Normalization	128	128
Dropout (p=0.2)	128	128
Fully Connected (FC4)	128	64
Layer Normalization	64	64
Dropout (p=0.1)	64	64
Fully Connected (FC5)	64	32
Fully Connected (FC6 - Saída)	32	$n_actions$

No trabalho [1], foi utilizada a função de ativação ReLU, enquanto neste estudo, optou-se pela função

de ativação Leaky ReLU. A principal diferença entre elas é que a ReLU só permite a ativação para entradas positivas, enquanto a Leaky ReLU permite uma ativação para entradas negativas, mas com um pequeno valor negativo, conhecido como "*negative slope*". Neste estudo, o valor do "*negative slope*"foi escolhido como 0,01.

A função de ação (*act*) da rede neural é responsável por determinar qual nó o agente deve escolher como próximo estado, com base no estado de observação de entrada recebido. Como a saída da rede possui dimensão igual à quantidade de nós do grafo, cada valor de saída representa a probabilidade associada à escolha de um determinado nó. Então, a função de ação seleciona o nó com o maior valor dentre as saídas da rede, utilizando a operação *argmax*, o que indica a decisão mais favorável segundo a política aprendida pelo agente.

O código completo da rede neural pode ser visualizado no Listing 7 do Apêndice I.3, que demonstra toda a implementação, desde a definição das camadas totalmente conectadas até a função de ação que utiliza o *argmax* para selecionar a ação com maior valor.

4.2.4 Atualização da *Q-Network* e *Q-Network* Alvo

A rede neural *Q-Network* é construída conforme descrito na Subseção 4.2.3, sendo responsável por aproximar a função de valor de ação durante o treinamento do agente, enquanto a rede neural *Q-Network* Alvo é inicialmente criada como uma cópia da *Q-Network*.

A *Q-Network* é atualizada a cada iteração do agente com o ambiente, buscando melhorar a estimativa da função de valor. Já a *Q-Network* Alvo é uma cópia congelada da *Q-Network*, para melhorar a estabilidade no processo de aprendizado, importante para o cálculo dos *targets* nas atualizações. A rede alvo é atualizada periodicamente, com base na *Q-Network*, de modo a garantir que os *targets* utilizados nas atualizações da rede principal não variem excessivamente, promovendo um treinamento mais estável.

A memória de experiência (*replay buffer*) é uma estrutura de armazenamento que guarda as transições e experiências do agente com o ambiente ao longo de todo o treinamento. Com um tamanho fixo pré-definido, sua principal função é armazenar um conjunto de experiências utilizadas para a atualização das redes neurais *Q-Network* e *Q-Network* Alvo. Essa memória permite que o agente aprenda de maneira mais eficiente, utilizando experiências passadas para melhorar suas estimativas de valor e promover um aprendizado mais estável.

A memória de experiência é uma estrutura de armazenamento composta por tuplas que armazenam as transições realizadas pelo agente. Cada tupla contém o vetor de estado atual do agente, a ação tomada, a recompensa recebida, o indicador de término da época (*done*), e o vetor de estado resultante no próximo estado, conforme definido na equação 4.6. Os vetores de estado (s_t) são definidos conforme a equação 4.2, apresentada na Subseção 4.2.1.

$$\mathbf{transition} = (\mathrm{s}_t, \mathrm{action}, \mathrm{reward}, \mathrm{done}, \mathrm{s}_{t+1}) \tag{4.6}$$

Para a atualização das redes neurais, sempre é utilizada uma amostra de tamanho máximo igual ao valor predefinido de *batch_size*, que determina a quantidade de transições a serem processadas por vez.

Caso o número de transições armazenadas na memória de experiência seja menor que *batch_size*, todas as transições disponíveis serão utilizadas. Caso contrário, uma seleção aleatória de transições será feita, mantendo o tamanho definido pelo *batch_size*.

Os *targets* representam os valores-alvo utilizados para atualizar a *Q-Network*, servindo como uma estimativa do retorno esperado para cada ação. Esses valores-alvo consideram tanto a recompensa imediata previamente calculada quanto uma expectativa da recompensa futura utilizando as transições de estado, ajustada pelo fator de desconto γ (gamma). O cálculo desses valores é realizado a partir da *Q-Network* Alvo, que faz uma estimativa dos valores de Q para os próximos estados. Esses *targets* são utilizados no cálculo da função de perda (*loss*) durante o treinamento, conforme descrito anteriormente na equação 2.12.

Por outro lado, os valores de ação Q (*action Q-values*) representam a estimativa da qualidade de cada ação em um determinado estado. Eles são calculados diretamente pela *Q-Network*. Esses valores são utilizados para a atualização da rede, pois são comparados com os *targets* durante o cálculo da função de perda (*loss*).

Tanto os *targets* quanto os valores de ação Q são calculados a partir das transições selecionadas da memória de experiência, que contém o histórico das iterações do agente com o ambiente.

O *loss*, portanto, é calculado utilizando a função de erro quadrático médio (*Mean Squared Error - MSE Loss*), que é uma métrica comum para quantificar a diferença entre os valores previstos e os valores reais. Neste contexto, os valores previstos são os valores de ação Q, calculados pela *Q-Network*, e os valores reais são os *targets*, que representam a estimativa do retorno esperado para cada ação, considerando a recompensa imediata e a projeção da recompensa futura descontada. O *MSE Loss* é calculado pela média dos quadrados das diferenças entre os valores de ação Q e os *targets*, conforme a equação 4.7.

$$\text{Loss} = \frac{1}{N}\sum_{i=1}^{N}\left(Q(s_i, a_i) - \textit{Target}_i\right)^2 \tag{4.7}$$

Por fim, a *Q-Network* alvo é atualizada periodicamente, com base em uma frequência de atualização previamente definida. A cada interação com o ambiente, verifica-se se o número da iteração atual é múltiplo da frequência estabelecida, por meio da operação de módulo. Caso o restante da divisão seja igual a zero, os parâmetros da *Q-Network* alvo são atualizados para refletir o estado atual da *Q-Network* principal. Essa estratégia de atualização periódica contribui para a estabilidade do treinamento, evitando oscilações bruscas nas estimativas de valor ao longo do processo de aprendizado.

O Algoritmo 5 demonstra todos os passos realizados para as atualizações das redes durante o treinamento.

Algorithm 5 Atualização da *Q-Network* e *Q-Network* Alvo

Require: $states_tensor$: Estados atuais do agente
Require: $actions_tensor$: Ações tomadas pelo agente
Require: $rewards_tensor$: Recompensas recebidas
Require: $dones_tensor$: Indicadores de término de episódio
Require: new_states_tensor: Novos estados resultantes das ações
Require: $batch_size$: Tamanho do lote de amostras
Require: γ: Fator de desconto
Require: $target$: *Q-Network* Alvo
Require: net: *Q-Network*
Ensure: $loss$: Perda calculada para atualização das redes

Passo 1: Calcule os valores de ação Q para os estados atuais usando a *Q-Network*
  $Q_{\text{current}} \leftarrow net(states_tensor)$
Passo 2: Calcule os valores de ação Q da *Q-Network* Alvo para os novos estados
  $Q_{\text{target}} \leftarrow target(new_states_tensor)$
Passo 3: Para cada transição, calcule o *target* usando a recompensa e o valor futuro projetado
  $target_i \leftarrow rewards_tensor_i + \gamma \cdot \max(Q_{\text{target},i})$
Passo 4: Calcule a perda (*loss*) como a diferença quadrática média entre os valores de ação Q e os *targets*
  $loss \leftarrow \frac{1}{batch_size} \sum_{i=1}^{batch_size} (Q_{\text{current},i}(s_i, a_i) - target_i)^2$
Passo 5: Realize a atualização da *Q-Network* principal utilizando a otimização do *loss* calculado
  optimizer.step()
if $episode \bmod update_frequency = 0$ **then**
  Atualize a *Q-Network* Alvo:
    $target \leftarrow net$
end if
  return $loss$

4.2.5 Obstáculos

Dois obstáculos foram considerados neste trabalho: nós atingidos por fogo e congestionamento. No trabalho [1], os nós em chamas não eram representados explicitamente no estado do agente, ou seja, a entrada da rede neural não continha nenhuma indicação da presença desse tipo de obstáculo. Como consequência, o agente aprendia a evitar esses nós apenas com base na alta penalização recebida ao acessar esses estados. Com isso, existia a necessidade de retreinamento sempre que novos nós fossem atingidos por fogo.

Neste trabalho, os nós em chamas são informados diretamente no estado do agente, como descrito na subseção 4.2.2. O objetivo dessa abordagem é permitir que o agente aprenda, desde o início, a lidar com diferentes combinações e configurações de obstáculos, evitando a necessidade de retreinamento a cada alteração no ambiente.

No início de cada época, é definido se haverá ou não nós em chamas e/ou nós em congestionamento. É importante observar que diversos grafos possuem nós críticos, ou seja, nós que, se forem escolhidos como fogo, bloqueiam completamente o nó inicial, impedindo qualquer rota segura de evacuação. Portanto, nesse trabalho foi definido que um nó crítico não pode ser selecionado como nó em chamas; esses nós são identificados no início do treinamento e excluídos da lista de possíveis nós de fogo, embora ainda possam

ser considerados como nós de congestionamento.

Para a configuração do ambiente de aprendizado, são considerados quatro possíveis cenários para cada episódio:

- Cenário 0: Nenhum obstáculo é inserido (sem fogo e sem congestionamento).
- Cenário 1: Apenas nós em chamas são inseridos.
- Cenário 2: Apenas nós em congestionamento são inseridos.
- Cenário 3: São inseridos tanto nós em chamas quanto nós em congestionamento.

O Algoritmo 6 apresenta como é realizada a seleção dos obstáculos no ambiente, incluindo nós de fogo e de congestionamento. A cada início de um episódio, um desses cenários é selecionado aleatoriamente. A partir da escolha do cenário, os nós de fogo e de congestionamento são sorteados aleatoriamente quando necessário.

Para garantir que o agente aprenda primeiramente o caminho ideal até a saída, o treinamento é iniciado forçando pelo menos duas execuções com o Cenário 0 para cada nó inicial, antes de permitir a exploração de ambientes com obstáculos.

Além disso, para promover uma exploração justa dos diferentes cenários existentes dentro do ambiente, é mantido um registro da frequência com que cada nó foi escolhido como fogo ou congestionamento, a partir de um nó inicial. Portanto, quando é feita a seleção dos obstáculos, é priorizado os nós que foram menos utilizados em épocas anteriores, garantindo uma distribuição mais equilibrada e uma cobertura mais completa das combinações de obstáculos ao longo do treinamento.

Além do cálculo dos nós críticos, são também pré-calculados os 5 menores caminhos possíveis de cada nó até todas as saídas disponíveis no ambiente. Essas informações são armazenadas em um dicionário chamado *all_paths* que associa cada nó do grafo a uma lista de caminhos viáveis, representados como sequências de nós.

No Algoritmo 6, a variável *all_paths* armazena todos os caminhos pré-calculados a partir do nó de origem (*state*). Um desses caminhos é selecionado aleatoriamente para a definição do obstáculo de fogo, enquanto outro caminho distinto, se disponível, é utilizado para a definição de obstáculos de congestionamento. Essa estratégia procura simular obstáculos em diferentes caminhos prováveis que o agente poderia utilizar para evacuação, permitindo a geração de cenários variados e distintos. Além disso, ao restringir a seleção dos obstáculos aos nós dos caminhos possíveis entre o nó inicial do agente e as saídas, evita-se a seleção desnecessária de obstáculos em nós que não fazem parte dos caminhos ideais do agente.

Algorithm 6 Geração de cenário a partir de um nó inicial

Require: $state$ ▷ Nó inicial
Ensure: Dicionário com `path`, `fire_nodes` e `congestion_nodes`

if $state \notin$ all_paths **or** all_paths$[state] = \emptyset$ **then**
    return vazio
end if
Escolher aleatoriamente um `path` em `all_paths`[state]
Inicializar listas vazias: `fire_nodes` e `congestion_nodes`
Incrementar contador de visitas para $state$
Definir $scenario_type$:
if visitas ≤ 2 **then**
    $scenario_type \leftarrow 0$
else
    Escolher aleatoriamente entre $\{0, 1, 2, 3\}$
end if
Definir número de agentes virtuais entre 0 e $max_virtual_agents$
Escolher `fire_path` e `congestion_path` (diferentes, se possível)
if $scenario_type$ inclui fogo **then**
    Escolher nó intermediário em `fire_path` (excluindo saídas e críticos)
    Adicionar nó à lista `fire_nodes` e atualizar contador
end if
if $scenario_type$ inclui congestionamento **then**
    Escolher nó intermediário em `congestion_path` (excluindo saídas e fogo)
    Ajustar número mínimo de agentes virtuais
    Adicionar múltiplos nós à lista `congestion_nodes` e atualizar contador
end if
Atualizar posições de agentes e nós de fogo
return resultado

5 RESULTADOS

Este capítulo apresenta os resultados obtidos em dois experimentos: o primeiro realizado em um ambiente menor, com apenas um piso, e o segundo em um ambiente maior, com dois pisos. Os resultados evidenciam uma evolução em relação ao trabalho de [1], destacando a capacidade do modelo proposto de operar em ambientes mais complexos.

5.1 AMBIENTE COMPUTACIONAL

Todos os experimentos descritos neste trabalho foram realizados em uma máquina com processador *AMD Ryzen 5 3500U*, placa de vídeo integrada *NVIDIA GeForce MX250*, 8 GB de memória RAM DDR4 com frequência de 2400 MHz.

Vale destacar que os tempos de treinamento e inferência estão diretamente relacionados às limitações do hardware utilizado.

5.2 EXPERIMENTO 1

No primeiro experimento, foi utilizada a planta baixa do primeiro piso do laboratório SG11 da Universidade de Brasília. A planta baixa e o grafo correspondente a essa planta podem ser visualizados na Figura 5.1. Vale ressaltar que, neste edifício, existem apenas duas saídas de emergência, o que indica que o dimensionamento do edifício não atende de forma ideal às exigências de segurança. A Figura 5.2 é a representação apenas do grafo desse ambiente.

Conforme as definições apresentadas na Subseção 2.6.2, o grafo em questão possui ordem 64, tamanho 80 e densidade aproximada de 0,0397, caracterizando um grafo esparso. Na Figura 5.2, os nós destacados em verde representam as saídas de emergência, enquanto os nós em azul correspondem a ambientes internos que não são pontos de saída. Especificamente, os nós 0 e 1 foram definidos como os pontos de saída deste grafo. É importante destacar que os nós representam não apenas os cômodos fechados, mas também os corredores, pois estes integram os caminhos das rotas de fuga. Além disso, janelas não foram consideradas como saídas de emergência nesta modelagem.

5.2.1 Configurações do Experimento

Os hiperparâmetros adotados no experimento 1 estão apresentados na Tabela 5.1.

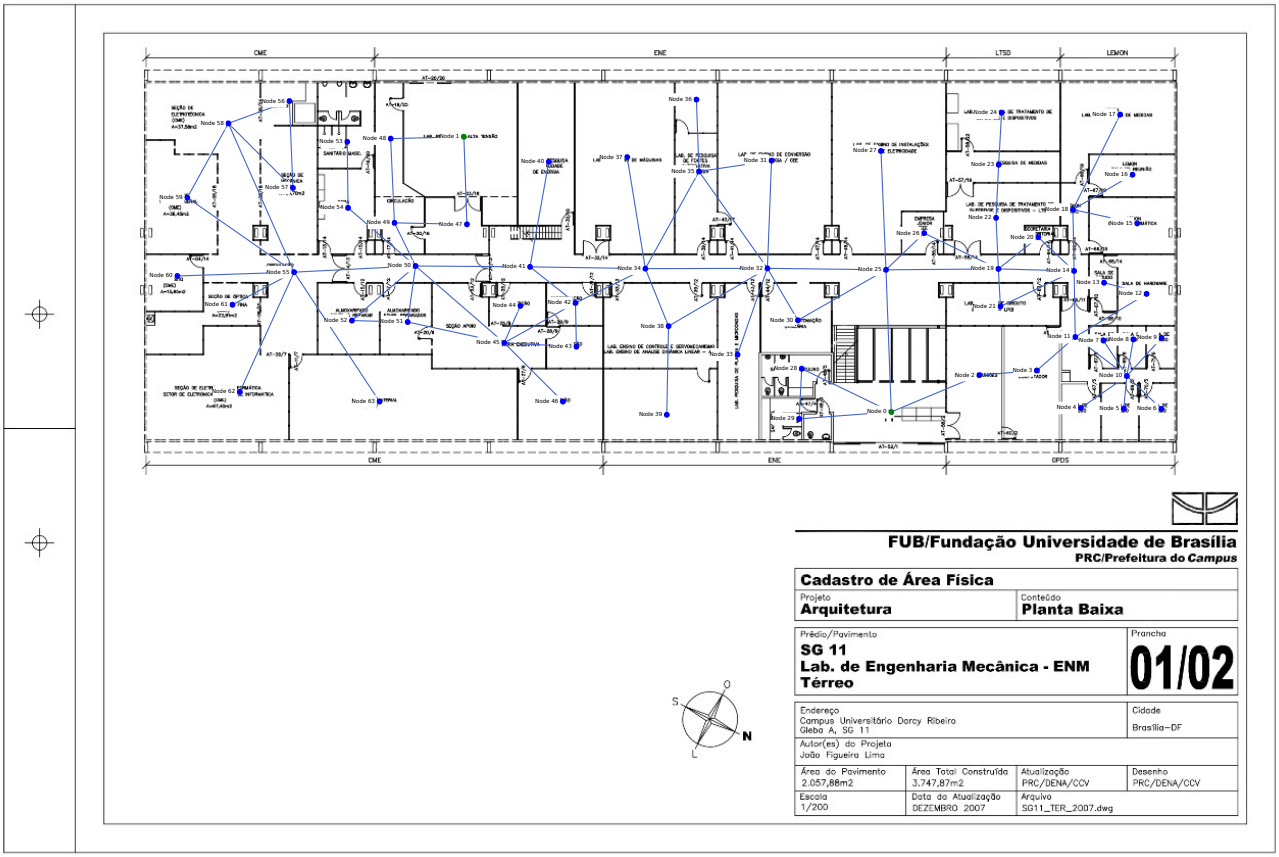

Figura 5.1: Grafo e planta baixa do laboratório SG11 da Universidade de Brasília (UnB). O arquivo original, em alta resolução, está disponível em [3].

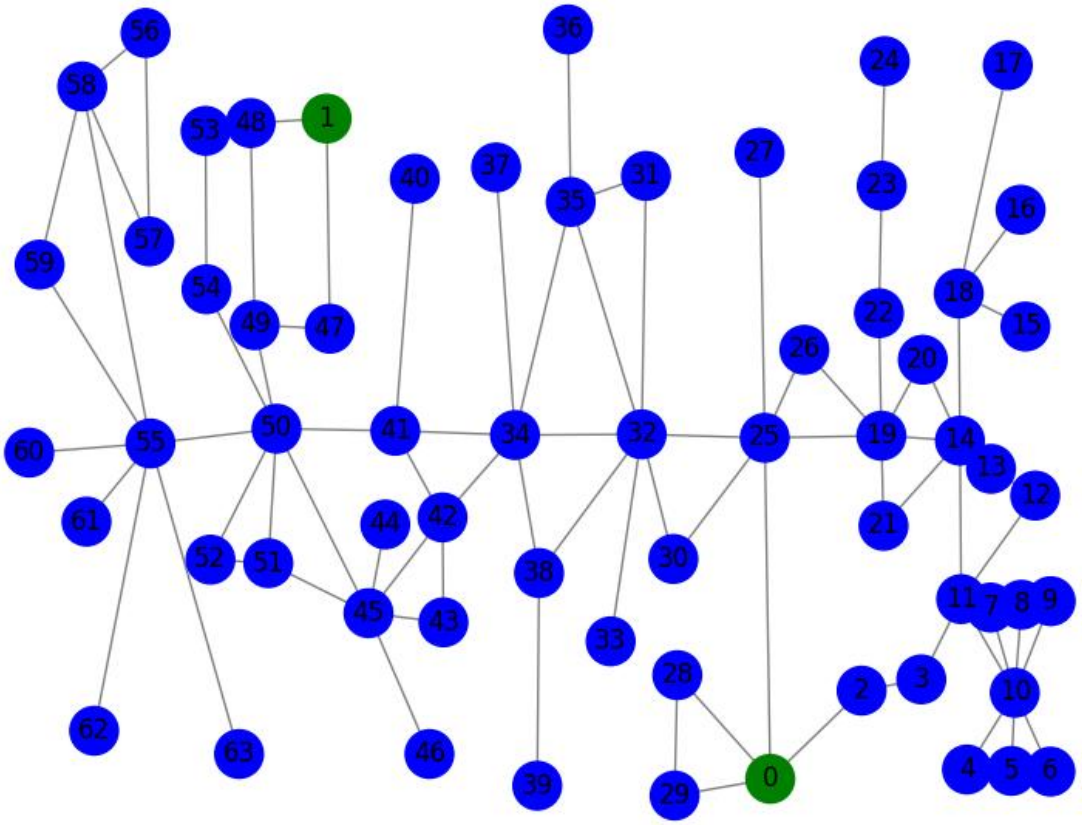

Figura 5.2: Grafo correspondente à planta baixa do laboratório SG11 da Universidade de Brasília (UnB).

Tabela 5.1: Hiperparâmetros utilizados no Experimento 1.

Hiperparâmetro	**Valor**
Beta	1
Learning Rate	1e-2
Batch Size	240
Buffer Size	4000
Iterações	56000
Gamma	0.9
Epsilon	0.7
Número de agentes virtuais	20
Threshold de congestionamento	5
Recompensa de saída	10000
Recompensa de fogo	-10000
Recompensa de movimento inválido	-5000
Recompensa de movimento válido	-1 * distância
Recompensa de congestionamento	-3000 * taxa de congestionamento

As métricas de avaliação utilizadas no treinamento foram: a função de perda, a recompensa média, o desvio padrão da recompensa, a proporção entre exploração e explotação, a quantidade de passos percorridos em cada episódio e a perda MSE.

O treinamento foi realizado permitindo a seleção aleatória do nó inicial do agente a cada término de trajeto bem-sucedido, bem como a escolha aleatória dos nós em chamas e dos nós com congestionamento.

5.2.2 treinamento

O treinamento do agente, considerando cenários com ocorrência de nós em chamas e/ou congestionamentos, teve duração total de 54 minutos. Nesse período, foram executadas aproximadamente 80.000 iterações, distribuídas em 10.305 episódios no ambiente simulado.

Os resultados da relação entre exploração e explotação estão apresentados na Figura 5.3. Observa-se que, ao longo das iterações com o ambiente, a taxa de exploração diminui progressivamente, enquanto a de explotação aumenta. Esse comportamento é resultado da estratégia adotada para controlar a exploração, conforme descrito na Subseção 2.3.1 e implementada no Algoritmo 3, apresentado na Subseção 4.2.1.

Os resultados referentes à quantidade de passos (iterações) necessários para que o agente conclua um episódio estão apresentados na Figura 5.4. Observa-se que, ao longo do treinamento, o agente aprende progressivamente os melhores caminhos para cada nó inicial, exigindo um número menor de passos para alcançar a saída.

Os resultados referentes à média das recompensas recebidas a cada iteração com o ambiente estão apresentados na Figura 5.5, enquanto a variância das recompensas é exibida na Figura 5.6. Observa-se que, com o passar do tempo, o agente aprende a maximizar suas recompensas, ao mesmo tempo em que reduz a variabilidade dos valores recebidos.

Por fim, o resultado da perda (*loss*) MSE está apresentado na Figura 5.7. Esse resultado é bastante

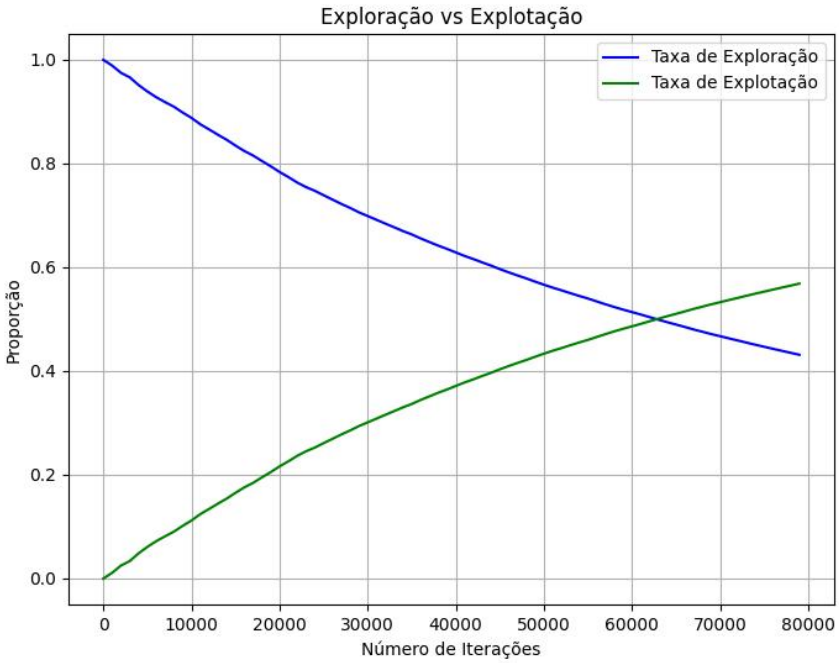

Figura 5.3: Experimento 1 - Relação entre as taxas de exploração e explotação ao longo do treinamento.

significativo, pois mostra que a função de perda escolhida (erro quadrático médio (MSE)), descrita na Subseção 4.2.4 na Equação 4.7, diminui a cada iteração do agente com o ambiente, indicando que o agente está de fato aprendendo seu objetivo final.

5.2.3 Visualização das Rotas Aprendidas

Nesta seção, são apresentados exemplos de trajetórias aprendidas pelo agente após o processo de treinamento, demonstrando sua capacidade de adaptação a diferentes configurações do ambiente. Serão demonstrados cenários em que o agente precisa encontrar o melhor caminho até a saída, sem conhecimento prévio da posição dos obstáculos. Também serão abordados casos específicos, como a presença de congestionamentos, a ocorrência de um nó atingido por fogo, e, por fim, um ambiente mais complexo que combina simultaneamente congestionamento e um nó em chamas.

5.2.3.1 Cenário 1 - Rotas sem Obstáculos

Este cenário representa a configuração mais simples para o processo de aprendizado, sendo o trajeto ideal entre a posição inicial do agente e a saída do edifício. Nele, não há a presença de congestionamentos ou outros obstáculos, permitindo que o agente tenha total liberdade para escolher sua rota de evacuação. Assim, o único critério de decisão adotado pelo agente é a busca pelo caminho mais curto, levando-o a selecionar a saída mais próxima com a menor distância percorrida.

Todas as rotas calculadas para cada nó do grafo individualmente podem ser consultadas na Tabela 1 do

Figura 5.4: Experimento 1 - Resultados referente a quantidade de passos necessários para concluir um episódio.

Apêndice I.1. Observa-se que o agente foi capaz de aprender o caminho até a saída a partir de todos os nós do ambiente. Isso demonstra que uma das principais etapas do objetivo foi atingida: entender quais são os nós de saída e identificar que eles representam o objetivo final a ser alcançado.

Para ilustrar esse comportamento, foram considerados cinco nós distintos, localizados em diferentes regiões do grafo: os nós 4, 17, 35, 46, 58 e 62. A Tabela 5.2 apresenta os nós iniciais e os respectivos caminhos percorridos pelo agente até alcançar um dos nós de saída.

Tabela 5.2: Experimento 1 - Caminhos percorridos pelo agente em um ambiente sem obstáculos, com as respectivas recompensas (Subseção 4.2.2) e distâncias aproximadas (Subseção 4.1.2.1).

Nó Inicial	**Caminho Percorrido**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
4	$4 \rightarrow 10 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$	9912	124
17	$17 \rightarrow 18 \rightarrow 14 \rightarrow 19 \rightarrow 25 \rightarrow 0$	9863	190
35	$35 \rightarrow 32 \rightarrow 25 \rightarrow 0$	9909	144
46	$46 \rightarrow 45 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	9877	156
58	$58 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	9844	188
62	$62 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	9856	176

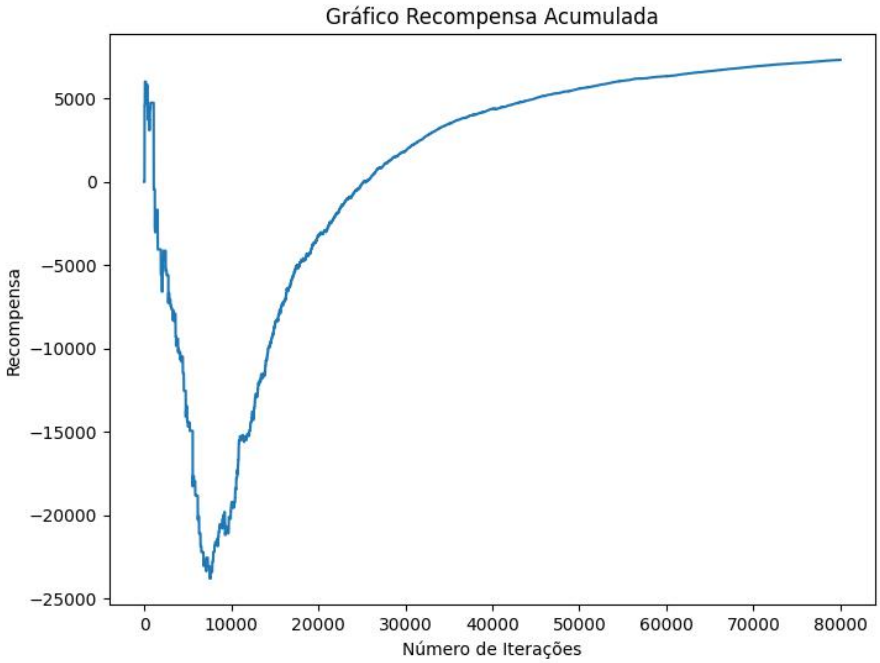

Figura 5.5: Experimento 1 - Resultados referente a média das recompensas.

A Figura 5.8 demonstra que o agente foi capaz de identificar caminhos eficientes em termos de distância e recompensa acumulada, priorizando o caminho mais curto até um nó de saída. Esses resultados mostram que, mesmo com a variação nas posições iniciais, o agente identificou caminhos que minimizavam a distância total percorrida, entendendo o objetivo principal do treinamento em um cenário onde não existem obstáculos.

Além disso, nota-se que a recompensa acumulada tende a ser maior quando a distância até a saída é menor, reforçando a correlação entre o critério de recompensa adotado e a eficiência da trajetória. Essa relação indica que o agente aprendeu a associar trajetos mais curtos a um melhor desempenho, reconhecendo a distância como um fator relevante a ser considerado.

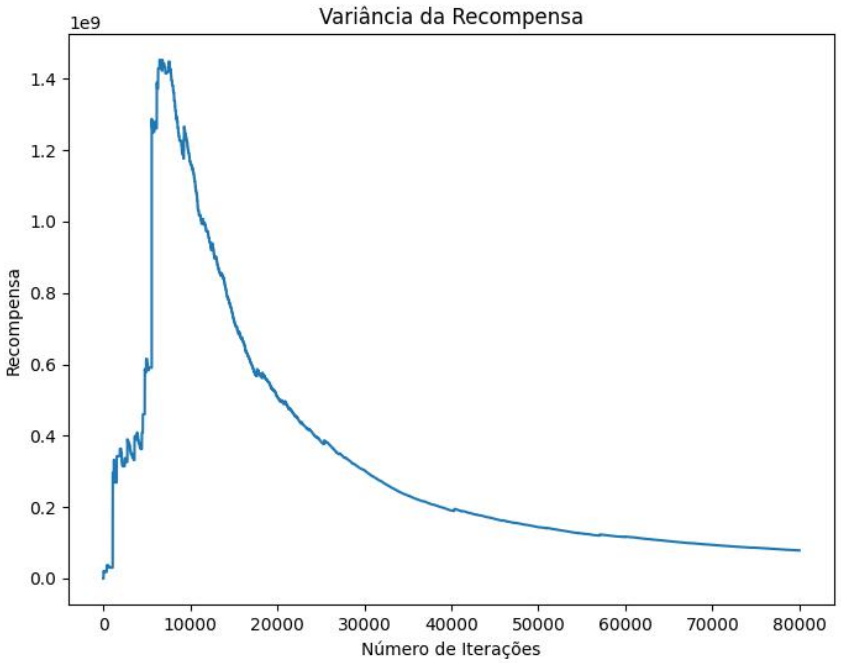

Figura 5.6: Experimento 1 - Resultados referente a variância das recompensa.

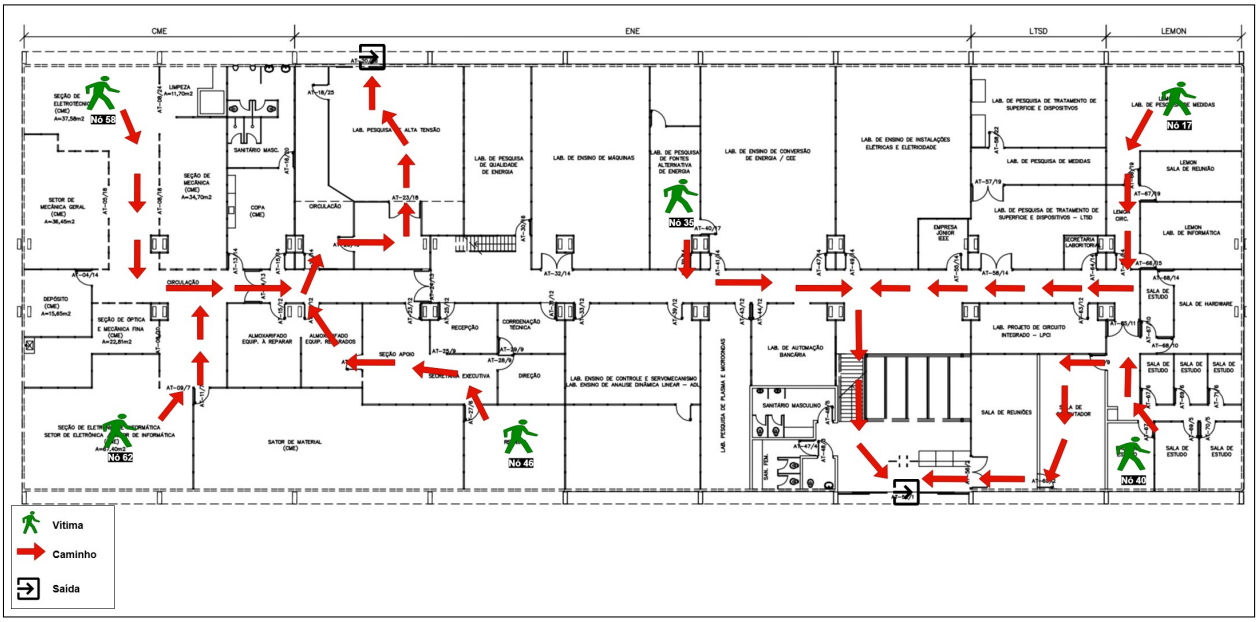

Figura 5.8: Experimento 1 - Resultado do agente em um ambiente sem obstáculos.

5.2.3.2 Cenário 2 - Rotas com congestionamento

Como o agente foi treinado considerando os cenários com presença de fogo, congestionamento e a combinação de ambos, é possível simular um ambiente contendo apenas congestionamento. A análise desse cenário é relevante para a pesquisa, pois ajuda a entender de forma clara como o agente responde

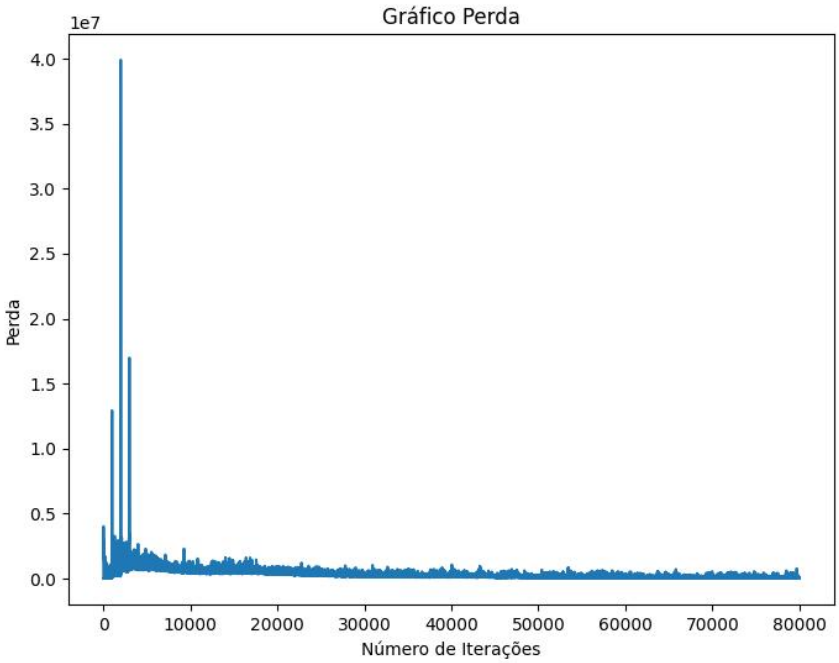

Figura 5.7: Experimento 1 - Resultados referente a perda das recompensas.

especificamente a esse tipo de obstáculo, sem a interferência de outros fatores, como o fogo.

Nesse cenário foram escolhidos dois nós para simulação de congestionamento no ambiente, sendo eles os nós 3 e 47, ambos com 6 agentes virtuais representando as áreas de congestionamento. Esses nós tiveram papel importante no cenário anterior (Subseção 5.2.3.1), sendo utilizados como parte das rotas de fuga mais curtas.

Com a adição desses congestionamentos, o objetivo desta etapa é avaliar a capacidade de adaptação do agente com mudanças nas condições do ambiente, avaliando se ele consegue evitar os nós congestionados e, ao mesmo tempo, encontrar caminhos eficientes e seguras até a saída.

É importante destacar que os cenários com congestionamento em diferentes nós foram simulados separadamente, mas serão apresentados em conjunto para facilitar a otimização e a análise comparativa dos resultados.

Para essa comparação, foram feitas duas simulações distintas:

- Primeira análise – Caminhos anteriores forçados: Nesta simulação, embora o ambiente tenha sido modificado com a introdução de nós congestionados, o agente foi obrigado a seguir os mesmos caminhos utilizados no cenário anterior, sem obstáculos (Subseção 5.2.3.1). O objetivo é analisar o impacto direto das penalizações na recompensa total ao manter as rotas previamente aprendidas, sem adaptação. Os resultados estão apresentados na Tabela 5.3.
- Segunda análise – Inferência livre com base no novo ambiente: Aqui, o agente teve liberdade para decidir seus caminhos com base na nova configuração do ambiente, levando em conta as penalizações associadas aos nós congestionados. O objetivo é avaliar se o modelo aprendeu a evitar rotas

penalizadas, ajustando suas decisões de forma inteligente. Os trajetos inferidos pelo agente nessa simulação estão apresentados na Tabela 5.4.

Tabela 5.3: Experimento 1 - Recompensas e distâncias ao forçar o agente a repetir os mesmos caminhos utilizados no ambiente sem congestionamento, agora com congestionamento nos nós 3 e 47.

Nó Inicial	**Caminho Percorrido**	**Nó congestionado**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
4	$4 \rightarrow 10 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$	3	9412	124
17	$17 \rightarrow 18 \rightarrow 14 \rightarrow 19 \rightarrow 25 \rightarrow 0$	3	9863	190
35	$35 \rightarrow 32 \rightarrow 25 \rightarrow 0$	3	9909	144
46	$46 \rightarrow 45 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	47	9377	156
58	$58 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	47	9344	188
62	$62 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	47	9356	176

Tabela 5.4: Experimento 1 - Recompensas e distâncias dos caminhos efetivamente escolhidos pelo agente ao enfrentar o ambiente com congestionamento nos nós 3 e 47.

Nó Inicial	**Caminho Percorrido**	**Nó congestionado**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
4	$4 \rightarrow 10 \rightarrow 11 \rightarrow 14 \rightarrow 19 \rightarrow 25 \rightarrow 0$	3	9856	198
17	$17 \rightarrow 18 \rightarrow 14 \rightarrow 19 \rightarrow 25 \rightarrow 0$	3	9863	190
35	$35 \rightarrow 32 \rightarrow 25 \rightarrow 0$	3	9909	144
46	$46 \rightarrow 45 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$	47	9872	156
58	$58 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$	47	9840	188
62	$62 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$	47	9852	176

Ao comparar os resultados das duas tabelas, observa-se que, ao evitar os nós congestionados, o agente percorreu distâncias um pouco maiores, ou até mesmo iguais, porém obteve recompensas acumuladas significativamente superiores. Essa diferença de recompensa reforça que a política aprendida não se limita a buscar o caminho mais curto em termos de distância, mas sim a rota com melhor custo-benefício, considerando os fatores dinâmicos do ambiente.

Os nós 17 e 35 mantiveram seus caminhos inalterados, pois seus caminhos ideais (sem a presença de obstáculos) já não envolviam nenhum dos nós congestionados. Dessa forma, esse cenário não possui impacto sobre o agente quando ele inicia a partir desses nós.

Portanto, considerando os cinco nós iniciais avaliados nesse cenário, a média das recompensas obtidas quando o agente foi forçado a seguir os caminhos antigos (Tabela 5.3) foi de aproximadamente 9677 pontos. Por outro lado, quando o agente teve liberdade para escolher novas rotas (Tabela 5.4), a média de recompensa foi de cerca de 9864 pontos, representando um aumento médio de aproximadamente 1,9%. Embora a diferença numérica possa parecer irrelevante, ela é significativa dentro do contexto da função de recompensa utilizada, principalmente considerando que o agente precisou percorrer distâncias maiores para evitar os congestionamentos e maximizar sua recompensa acumulada. Esses resultados demonstram que o agente aprendeu a realizar o custo-benefício entre a distância percorrida e penalizações por obstáculos no ambiente.

A Figura 5.22 apresenta os caminhos ideais inferidos pelo agente para o cenário com congestionamento, mostrando as rotas escolhidas para evitar os nós penalizados.

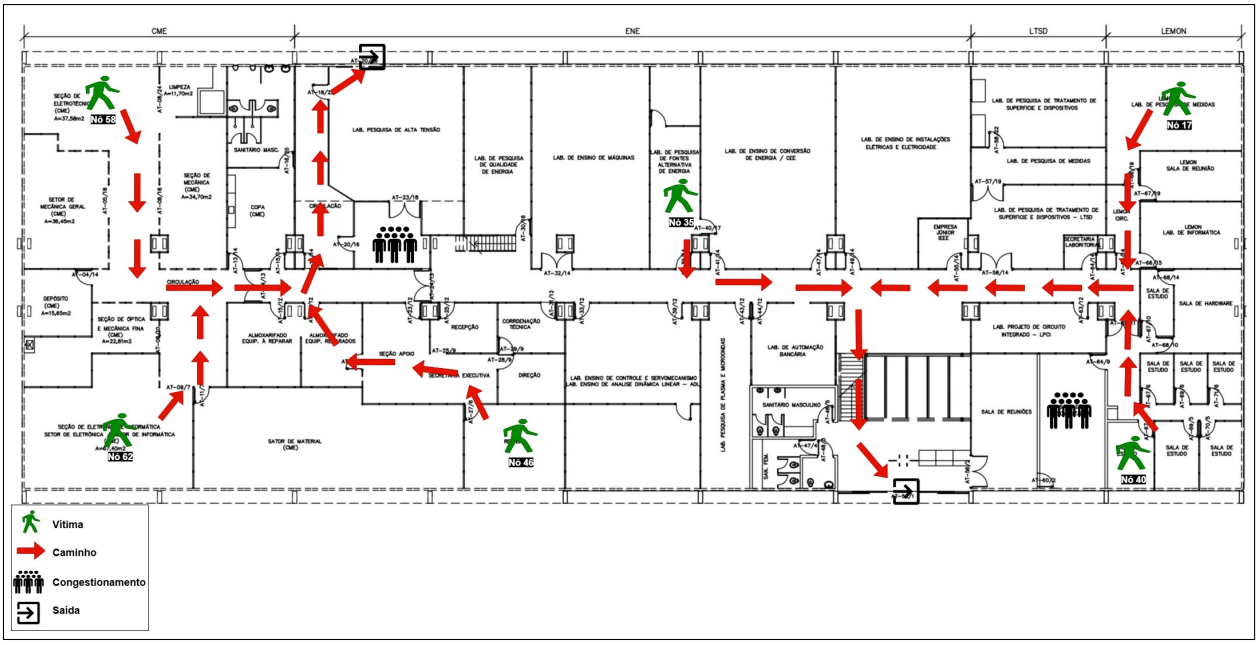

Figura 5.9: Experimento 1 - Resultado do agente ao enfrentar congestionamento nos nós 3 e 17.

5.2.3.3 Cenário 3 - Rotas com nós em chamas

De forma semelhante ao cenário anterior, onde o objetivo foi avaliar a capacidade do agente de aprender situações com apenas congestionamento, este novo cenário pretende analisar o comportamento do agente na presença exclusiva de nós em chamas. Serão realizadas duas inferências distintas, a primeira considerando o nó 25 como obstáculo de fogo, e a segunda com o nó 49. O objetivo é verificar se o agente consegue adaptar suas rotas de fuga ao evitar essas regiões perigosas, mesmo na ausência de congestionamento.

A escolha dos nós 25 e 49 como obstáculos de fogo não foi aleatória, ambos os nós desempenham um papel importante nas rotas de fuga, sendo frequentemente utilizados nos caminhos mais curtos até a saída, como demonstrado na Subseção 5.2.3.1. A presença de fogo nesses nós inviabiliza o uso de alguns caminhos dentro do ambiente, exigindo que o agente adapte sua política de decisão e busque alternativas viáveis para alcançar a saída com segurança e a melhor recompensa acumulada possível.

Ao configurar o nó 25 como um nó em chamas e realizar a comparação com o cenário inicial apresentado na Subseção 5.2.3.1, observa-se que os nós iniciais 17 e 35 precisam recalcular seus caminhos até a saída, uma vez que seus caminhos anteriores dependiam da passagem pelo nó 25, agora configurado como um obstáculo. Por outro lado, os nós 4, 46, 62 e 58 não apresentam alterações em seus caminhos, pois suas rotas de evacuação anteriormente calculadas não incluíam o nó 25.

Para comparação, foram feitas duas simulações distintas:

- Primeira análise – Caminhos anteriores forçados: Nesta simulação, embora o ambiente tenha sido modificado com a introdução de nós com presença de fogo, o agente foi obrigado a seguir os mesmos caminhos utilizados no primeiro cenário, sem obstáculos (Subseção 5.2.3.1). O objetivo é analisar o impacto direto das penalizações na recompensa total ao manter as rotas previamente aprendidas, sem adaptação. Os resultados estão apresentados na Tabela 5.5.
- Segunda análise – Inferência livre com base no novo ambiente: Aqui, o agente teve liberdade para decidir seus caminhos com base na nova configuração do ambiente, levando em conta as penalizações associadas aos nós com presença de fogo. O objetivo é avaliar se o modelo aprendeu a evitar rotas penalizadas, ajustando suas decisões de forma inteligente. Os trajetos inferidos pelo agente nessa simulação estão apresentados na Tabela 5.6.

Tabela 5.5: Experimento 1 - Recompensas e distâncias ao forçar o agente a repetir os mesmos caminhos utilizados no ambiente sem nós em chamas, agora com fogo no nó 25.

Nó Inicial	**Caminho Percorrido**	**Nó com fogo**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
17	$17 \rightarrow 18 \rightarrow 14 \rightarrow 19 \rightarrow 25 \rightarrow 0$	25	-93	190
35	$35 \rightarrow 32 \rightarrow 25 \rightarrow 0$	25	-45	144

Tabela 5.6: Experimento 1 - Recompensas e distâncias dos caminhos efetivamente escolhidos pelo agente ao enfrentar o ambiente com fogo no nó 25.

Nó Inicial	**Caminho Percorrido**	**Nó com fogo**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
17	$17 \rightarrow 18 \rightarrow 14 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$	25	9868	168
35	$35 \rightarrow 34 \rightarrow 41 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	25	9823	210

Percebe-se que a penalização aplicada quando o agente é forçado a atravessar um nó em chamas é extremamente severa, resultando em recompensas negativas nos casos testados (Tabela 5.5). Esse resultado demonstra que a função de recompensa foi eficaz em desencorajar o agente a escolher caminhos que passem por esse tipo de obstáculo, por meio de penalizações.

Um detalhe interessante a ser observado é o comportamento do agente em relação ao nó inicial 17. No cenário sem fogo (Subseção 5.2.3.1), o agente percorreu uma distância maior (190) e obteve uma recompensa acumulada de aproximadamente 9863. Já no cenário com o nó 25 em chamas, o agente escolheu um caminho mais curto (168) e alcançou uma recompensa acumulada ligeiramente superior (9868).

Essa pequena diferença entre as recompensas totais sugere que, quando diferentes caminhos oferecem recompensas finais semelhantes, o agente tende a não considerar essas variações como significativas. Isso indica que o modelo foi capaz de aprender a evitar penalizações severas, como as causadas por atravessar nós em chamas, mas que pequenas flutuações positivas ou negativas na recompensa final não influenciam de forma relevante sua escolha, dados os hiperparâmetros definidos neste experimento.

A Figura 5.10 apresenta as trajetórias inferidas pelo agente para cada nó inicial considerado no cenário com o nó 25 em chamas.

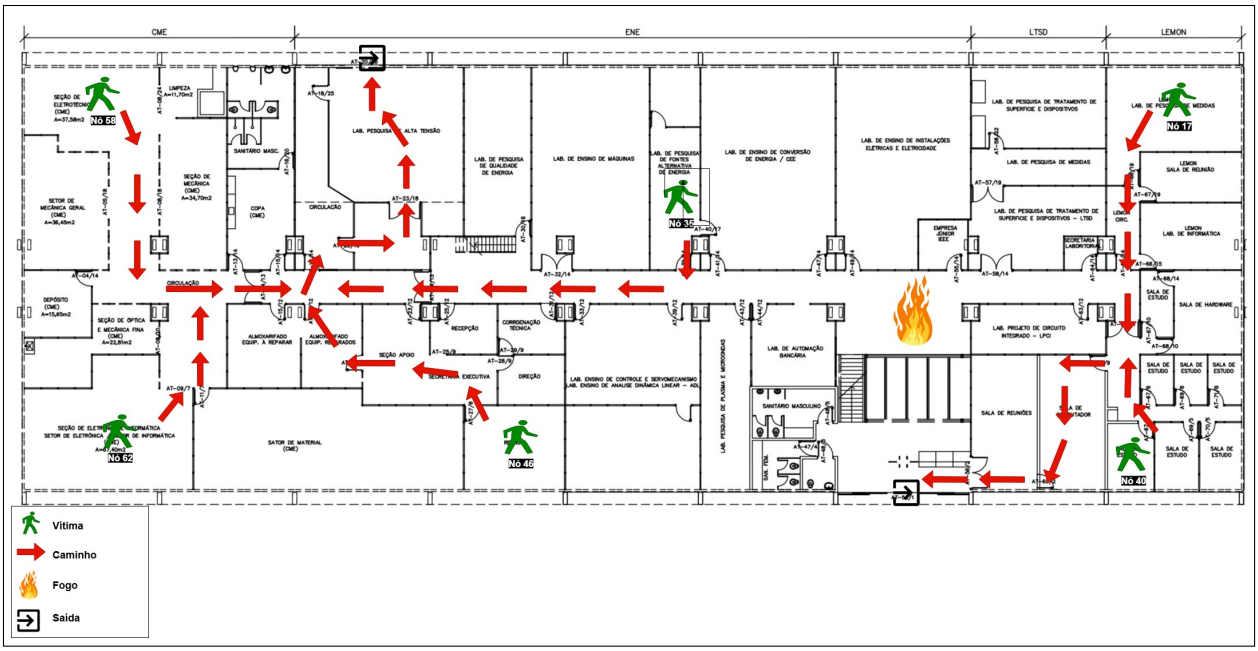

Figura 5.10: Experimento 1 - Resultado do agente ao enfrentar um ambiente com fogo no nó 25.

As mesmas comparações serão realizadas agora considerando o nó 49 como nó em chamas. Nesse novo cenário, os nós iniciais 46, 58 e 62 são afetados diretamente, uma vez que todos os caminhos ideais calculados no cenário inicial (Subseção 5.2.3.1) envolviam a passagem pelo nó 49. Dessa forma, o objetivo é analisar como o agente adapta suas decisões para evitar o nó penalizado e ainda encontrar caminhos seguros até a saída.

A Tabela 5.7 apresenta os resultados da simulação em que o agente foi forçado a seguir os mesmos caminhos ideais calculados anteriormente na Subseção 5.2.3.1, mesmo com a presença de fogo no nó 49. Já a Tabela 5.8 mostra as trajetórias efetivamente inferidas pelo agente ao se deparar com o ambiente com obstáculo de fogo.

Tabela 5.7: Experimento 1 - Recompensas e distâncias ao forçar o agente a repetir os mesmos caminhos utilizados no ambiente sem nós em chamas, agora com fogo no nó 49.

Nó Inicial	**Caminho Percorrido**	**Nó com fogo**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
46	$46 \rightarrow 45 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	49	-104	156
58	$58 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	49	-136	188
62	$62 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	49	-124	176

Tabela 5.8: Experimento 1 - Recompensas e distâncias dos caminhos efetivamente escolhidos pelo agente ao enfrentar o ambiente com fogo no nó 49.

Nó Inicial	**Caminho Percorrido**	**Nó com fogo**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
46	$46 \rightarrow 45 \rightarrow 42 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$	49	9814	239
58	$58 \rightarrow 55 \rightarrow 50 \rightarrow 41 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$	49	9710	343
62	$62 \rightarrow 55 \rightarrow 50 \rightarrow 41 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$	49	9722	331

Assim como no experimento anterior, as penalizações recebidas quando o caminho percorrido pelo agente passa por um nó em chamas são bastante severas, tornando inviável para o agente escolher caminhos que incluam o nó 49. Como consequência, o agente foi incentivado a buscar caminhos significativamente mais longos, porém mais seguros e com maiores recompensas acumuladas. Esse comportamento, assim como o anterior, reforça a eficácia da política aprendida na priorização de caminhos com menor risco, ainda que exista um aumento considerável na distância percorrida.

A Figura 5.11 apresenta as trajetórias inferidas pelo agente para cada nó inicial considerado no cenário com o nó 49 em chamas.

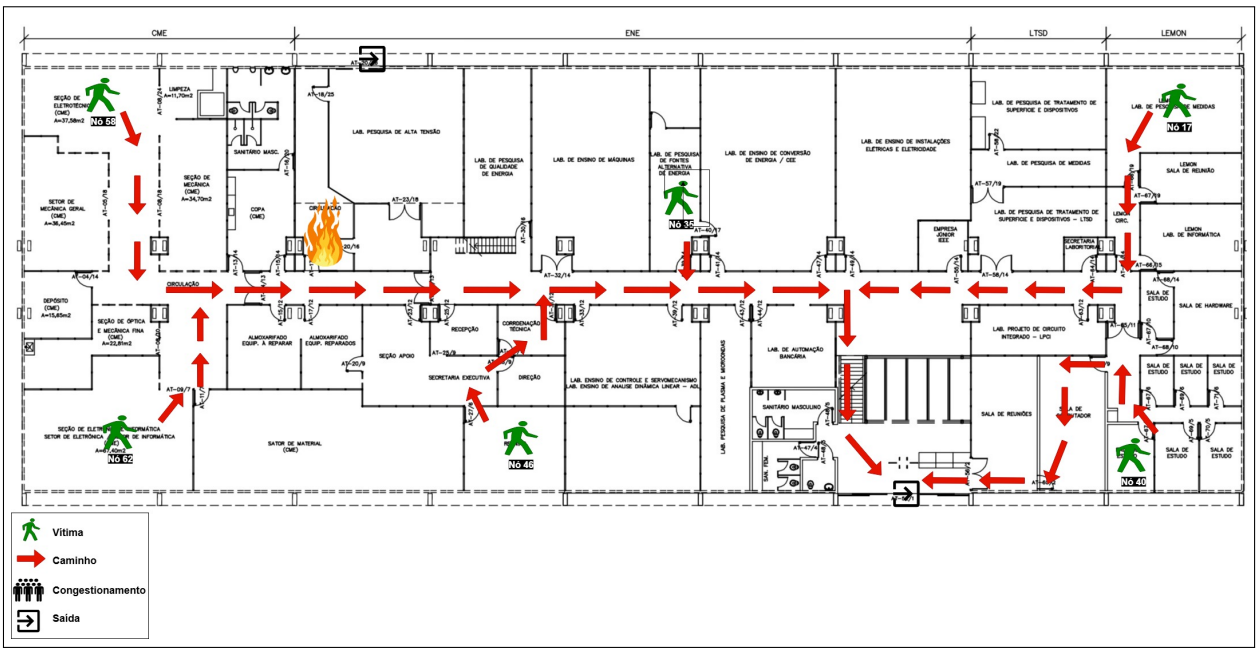

Figura 5.11: Experimento 1 - Resultado do agente ao enfrentar um ambiente com fogo nó 49.

5.2.3.4 Cenário 4 - Rotas com nós em chamas e congestionamento

Por fim, esse cenário apresenta a combinação dos dois tipos de obstáculos abordados neste trabalho, sendo a presença simultânea de nós em chamas e de congestionamentos. O objetivo dessa análise é avaliar a capacidade do agente de lidar com múltiplos obstáculos de naturezas diferentes, adaptando sua política de decisão para encontrar caminhos seguros até as saídas, com a melhor recompensa acumulada, mesmo em ambientes perigosos.

Assim como no cenário anterior, este será dividido em duas etapas experimentais distintas. Na primeira configuração, o nó 25 será definido como nó em chamas, enquanto o nó 47 será tratado como nó congestionado. Na segunda configuração, o nó 49 será o nó em chamas e o nó 41 representará o congestionamento.

A escolha dos nós para representar os obstáculos de fogo e congestionamento não foi aleatória. Tanto o nó 25 quanto o nó 47 (na primeira configuração), assim como o nó 49 e o nó 41 (na segunda configuração), fazem parte das principais rotas de fuga em um cenário livre de obstáculos (Subseção 5.2.3.1). Dessa forma, a escolha desses obstáculos força o agente a recalcular seus caminhos ideais, levando em consideração a nova configuração do ambiente.

Na primeira configuração deste experimento combinado, os nós iniciais 17 e 35 são os mais impactados, uma vez que seus caminhos ideais no cenário sem obstáculos (Subseção 5.2.3.1) passavam diretamente pelos nós 25 (agora em chamas) e 47 (agora congestionado). A Tabela 5.9 apresenta a recompensa acumulada caso o agente fosse forçado a seguir os mesmos caminhos do cenário sem obstáculos, enquanto a Tabela 5.10 mostra os novos caminhos inferidos pelo agente, considerando simultaneamente a presença de fogo e congestionamento.

Tabela 5.9: Experimento 1 - Recompensas e distâncias ao forçar o agente a repetir os mesmos caminhos utilizados no ambiente sem obstáculos, agora com fogo no nó 25 e congestionamento no nó 47.

Nó Inicial	**Caminho Percorrido**	**Nó com fogo**	**Nó congestionado**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
17	$17 \rightarrow 18 \rightarrow 14 \rightarrow 19 \rightarrow 25 \rightarrow 0$	25	47	-93	190
35	$35 \rightarrow 32 \rightarrow 25 \rightarrow 0$	25	47	-45	144

Tabela 5.10: Experimento 1 - Recompensas e distâncias dos caminhos efetivamente escolhidos pelo agente ao enfrentar o ambiente com fogo no nó 25 e congestionamento no nó 47.

Nó Inicial	**Caminho Percorrido**	**Nó com fogo**	**Nó congestionado**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
17	$17 \rightarrow 18 \rightarrow 14 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$	25	47	9868	168
35	$35 \rightarrow 34 \rightarrow 41 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$	25	47	9818	209

Na segunda configuração experimental, os nós iniciais mais afetados são 46, 58 e 62, onde os caminhos em um ambiente sem obstáculos incluíam o nó 49 (agora em chamas) e o nó 41 (agora congestionado). Novamente, a Tabela 5.11 apresenta os resultados forçando o agente a seguir os caminhos do ambiente sem obstáculos, e a Tabela 5.12 mostra os caminhos efetivamente escolhidos pelo agente no novo ambiente proposto.

Tabela 5.11: Experimento 1 - Recompensas e distâncias ao forçar o agente a repetir os mesmos caminhos utilizados no ambiente sem obstáculos, agora com fogo no nó 49 e congestionamento no nó 41.

Nó Inicial	**Caminho Percorrido**	**Nó com fogo**	**Nó congestionado**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
46	$46 \rightarrow 45 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	49	41	-104	156
58	$58 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	49	41	-136	188
62	$62 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	49	41	-124	176

Tabela 5.12: Recompensas e distâncias dos caminhos efetivamente escolhidos pelo agente ao enfrentar o ambiente com fogo no nó 49 e congestionamento no nó 41.

Nó Inicial	**Caminho Percorrido**	**Nó com fogo**	**Nó congestionado**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
46	$46 \rightarrow 45 \rightarrow 42 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$	49	41	9814	239
58	$58 \rightarrow 55 \rightarrow 50 \rightarrow 45 \rightarrow 42 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$	49	41	9692	361
62	$62 \rightarrow 55 \rightarrow 50 \rightarrow 45 \rightarrow 42 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$	49	41	9704	349

Com os resultados apresentados nas Tabelas 5.10 a 5.12, entende-se que o agente aprendeu a evitar, tanto os nós congestionados quanto os nós em chamas. Em ambas as configurações, percebe-se que, ao ser forçado a seguir os caminhos da Subseção 5.2.3.1, onde não existia nenhum obstáculo no ambiente, o agente acumulou recompensas extremamente baixas e negativas, devido à penalização por passar por nós em chamas e congestionados. Por outro lado, ao ter liberdade para recalcular suas rotas, o agente foi capaz de encontrar caminhos alternativos que, embora apresentem distâncias maiores, resultam em recompensas mais altas.

Esse comportamento reforça a capacidade de generalização e adaptação da política aprendida, demonstrando que o agente não apenas memoriza rotas fixas, mas consegue tomar decisões com base nas condições dinâmicas do ambiente. O equilíbrio entre evitar penalizações severas e minimizar a distância percorrida fica claro nas escolhas feitas pelo agente quando encontra um ambiente com múltiplos obstáculos.

A Figura 5.12 apresenta os caminhos inferidos pelo agente na primeira configuração, com fogo no nó 25 e congestionamento no nó 47. Já a Figura 5.13 mostra os trajetos escolhidos na segunda configuração, com fogo no nó 49 e congestionamento no nó 41.

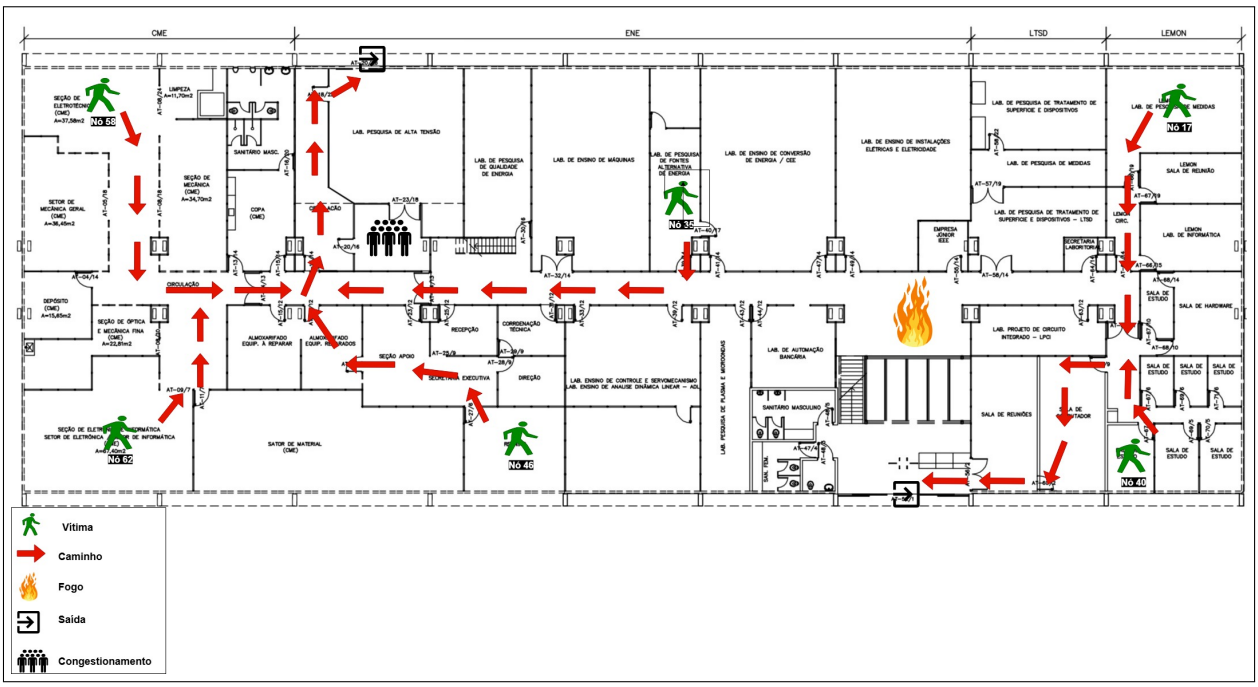

Figura 5.12: Experimento 1 - Resultado do agente ao enfrentar um ambiente com fogo no nó 25 e congestionamento no nó 47.

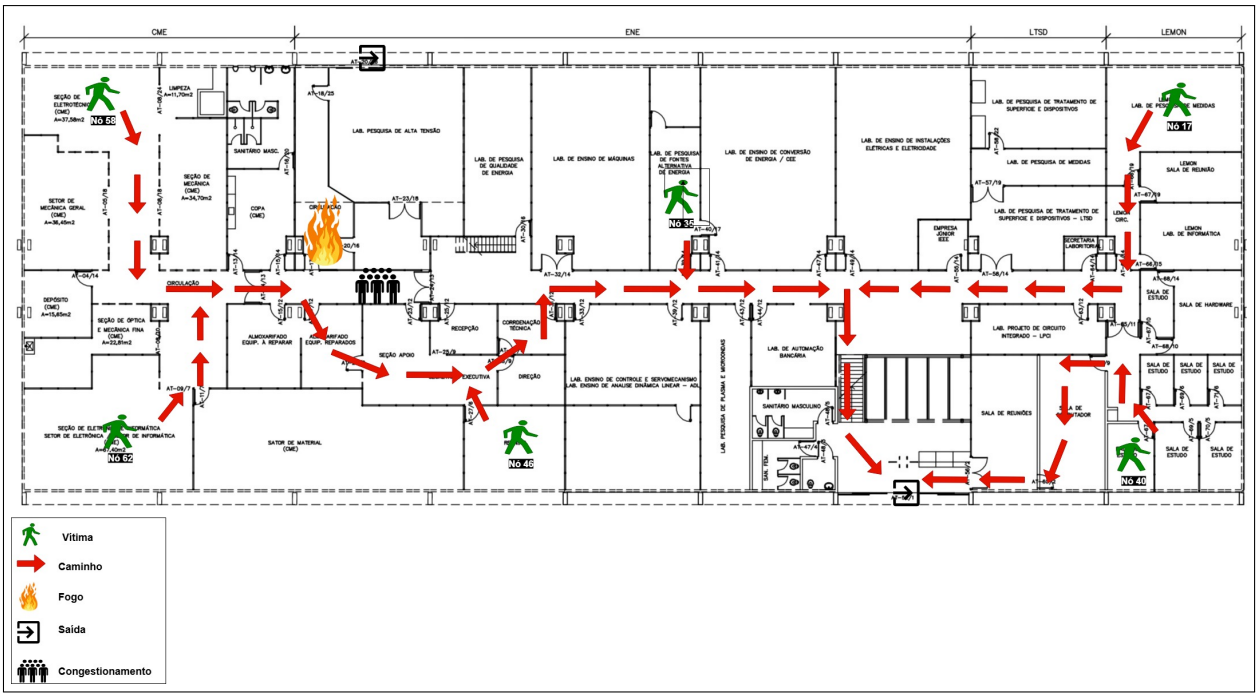

Figura 5.13: Experimento 1 - Resultado do agente ao enfrentar um ambiente com fogo no nó 49 e congestionamento no nó 41.

5.3 EXPERIMENTO 2

No segundo experimento, foi utilizada uma planta baixa combinada do primeiro piso e do segundo piso do laboratório SG11 da Universidade de Brasília. A planta baixa e o grafo correspondente a essa planta podem ser visualizados na Figura 5.14. Vale ressaltar que, neste edifício, existem apenas duas saídas de emergência, o que indica que o dimensionamento do edifício não atende de forma ideal às exigências de segurança. A Figura 5.15 é a representação apenas do grafo desse ambiente e apresenta de forma mais clara como o ambiente está representado.

Conforme as definições apresentadas na Subseção 2.6.2, o grafo do experimento 2 possui ordem 119, tamanho 142 e densidade aproximada de $0{,}0202$, caracterizando um grafo esparso. Na Figura 5.15, os nós destacados em verde representam as saídas de emergência, enquanto os nós em azul correspondem a ambientes internos que não são pontos de saída. Especificamente, os nós 0 e 1 foram definidos como os pontos de saída deste grafo. É importante destacar que os nós representam não apenas os cômodos fechados, mas também os corredores, pois estes participam do trajeto das rotas de fuga. Além disso, janelas não foram consideradas como saídas de emergência nesta modelagem.

5.3.1 Configurações do Experimento

Os hiperparâmetros adotados no Experimento 2 estão apresentados na Tabela 5.13. Nota-se que a penalização aplicada aos nós em chamas foi maior do que no experimento anterior. Isso ocorre porque as distâncias entre os nós, medidas em *pixels*, não foram normalizadas. Como neste experimento as distâncias até a saída são maiores, foi necessário aumentar a penalização associada ao fogo, de modo a manter sua influência proporcional sobre a decisão do agente.

Tabela 5.13: Hiperparâmetros utilizados no Experimento 2.

Hiperparâmetro	**Valor**
Beta	1
Learning Rate	1e-2
Batch Size	240
Buffer Size	4000
Iterações	140000
Gamma	0.9
Epsilon	0.7
Número de agentes virtuais	20
Threshold de congestionamento	5
Recompensa de saída	10000
Recompensa de fogo	-15000
Recompensa de movimento inválido	-5000
Recompensa de movimento válido	-1 * distância
Recompensa de congestionamento	-3000 * taxa de congestionamento

Além disso, em comparação com o experimento 1, observa-se que o número de iterações do agente dentro do ambiente será consideravelmente maior. Isso ocorre porque o grafo utilizado no experimento 2

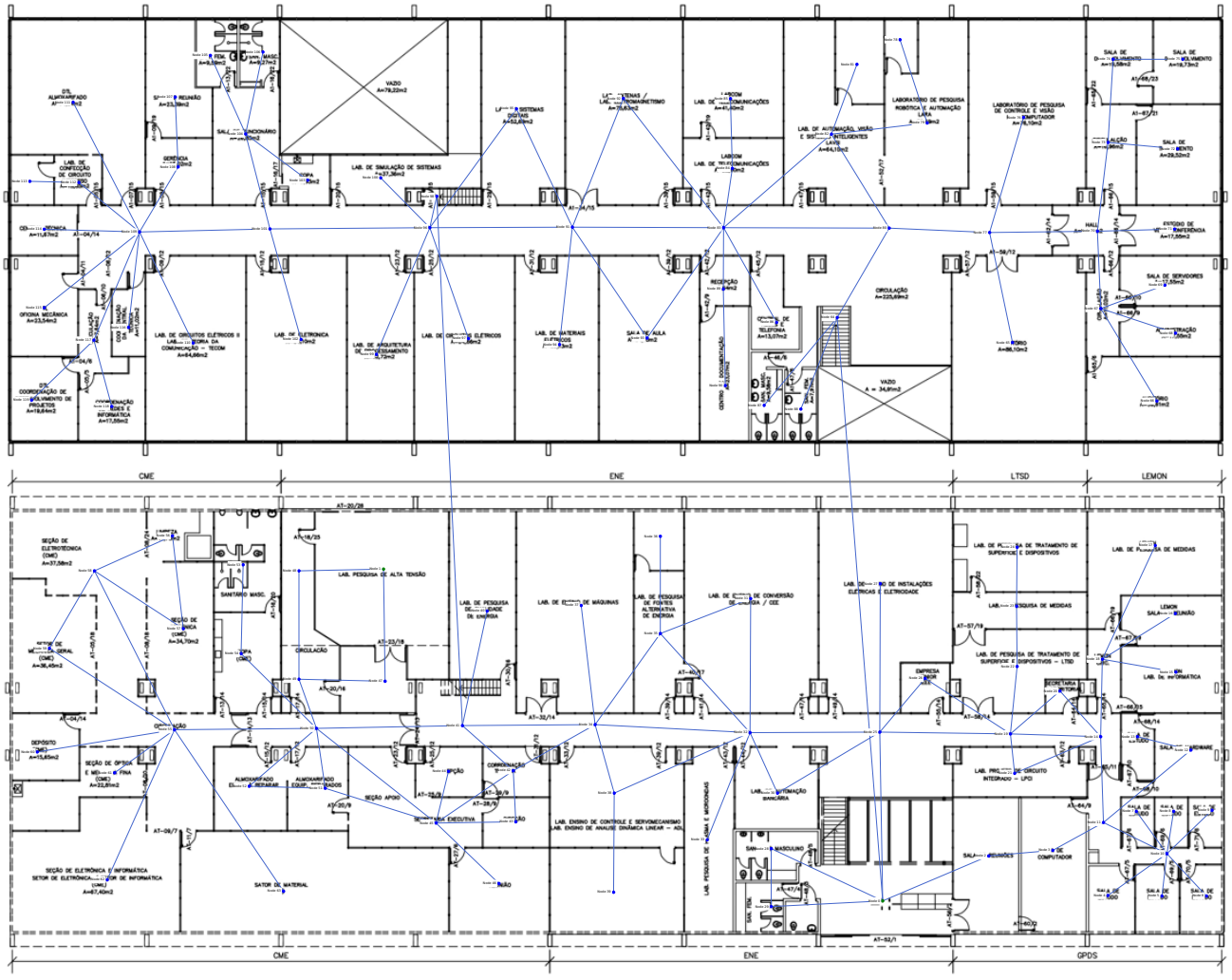

Figura 5.14: Grafo e planta baixa dos dois pisos do laboratório SG11 da Universidade de Brasília (UnB). O arquivo original, em alta resolução, está disponível em: [4].

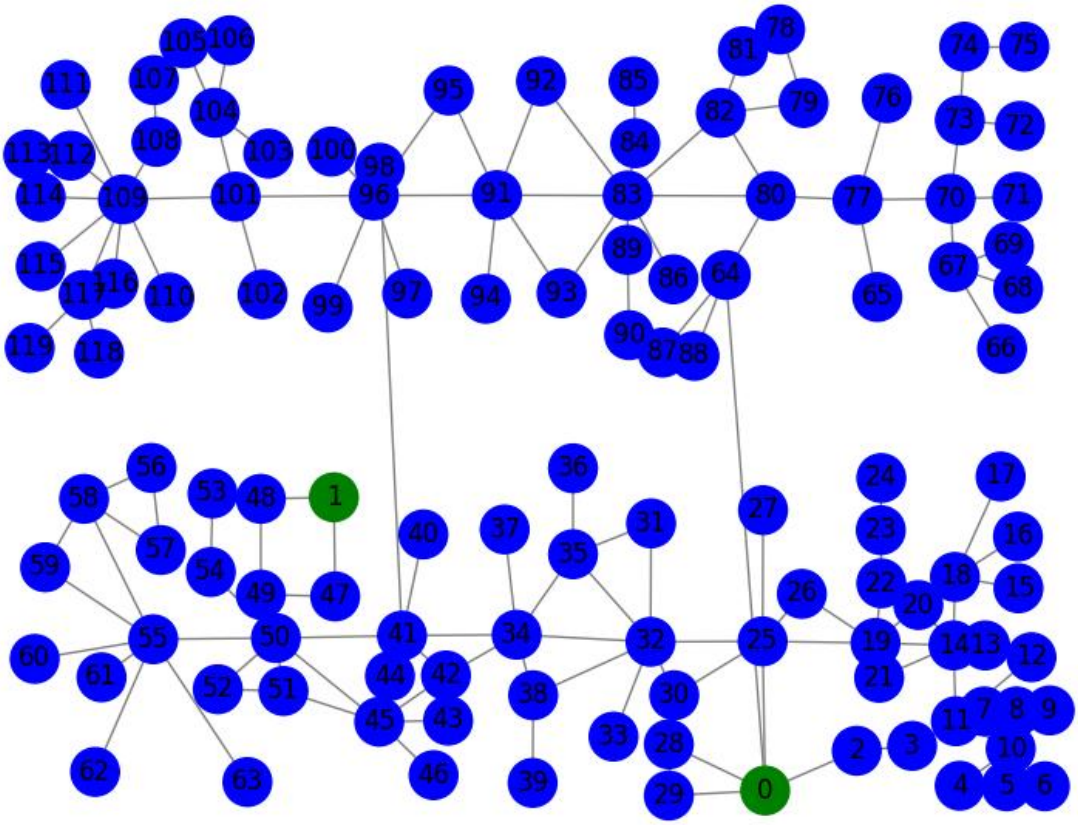

Figura 5.15: Grafo correspondente à planta baixa dos dois pisos do laboratório SG11 da Universidade de Brasília (UnB).

possui um tamanho significativamente maior, exigindo mais iterações até que o agente consiga compreender todas as configurações possíveis para cada um dos nós.

Assim como no experimento anterior, as métricas de avaliação utilizadas no treinamento foram: a função de perda, a recompensa média, o desvio padrão da recompensa, a proporção entre exploração e explotação, a quantidade de passos percorridos em cada episódio e a perda MSE.

O treinamento foi realizado permitindo a seleção aleatória do nó inicial do agente a cada término de trajeto bem-sucedido, bem como a escolha aleatória dos nós em chamas e dos nós com congestionamento.

5.3.2 treinamento

O treinamento do agente, com os obstáculos de fogo e de congestionamentos, teve duração total de 2 horas, 35 minutos e 29 segundos. Durante esse processo, o agente executou aproximadamente 140.000 iterações, distribuídas ao longo de 12.235 episódios no ambiente simulado.

Os resultados da relação entre exploração e explotação estão apresentados na Figura 5.16. Observa-se que, ao longo das interações com o ambiente, a taxa de exploração diminui progressivamente, enquanto a de explotação aumenta. Esse comportamento é resultado da estratégia adotada para controlar a exploração, conforme descrito na Subseção 2.3.1 e implementada no Algoritmo 3, apresentado na Subseção 4.2.1.

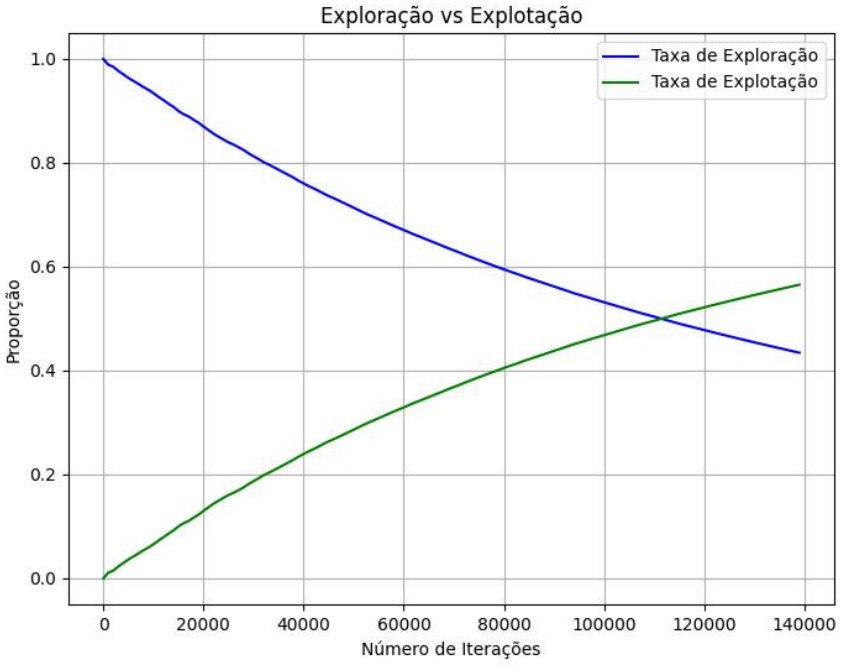

Figura 5.16: Relação entre as taxas de exploração e explotação ao longo do treinamento no Experimento 1.

Os resultados referentes à quantidade de passos (iterações) necessários para que o agente conclua um episódio estão apresentados na Figura 5.17. Observa-se que, ao longo dos episódios, o agente aprende progressivamente os melhores caminhos, exigindo um número menor de passos para alcançar a saída.

Os resultados referentes à média das recompensas recebidas a cada iteração com o ambiente estão

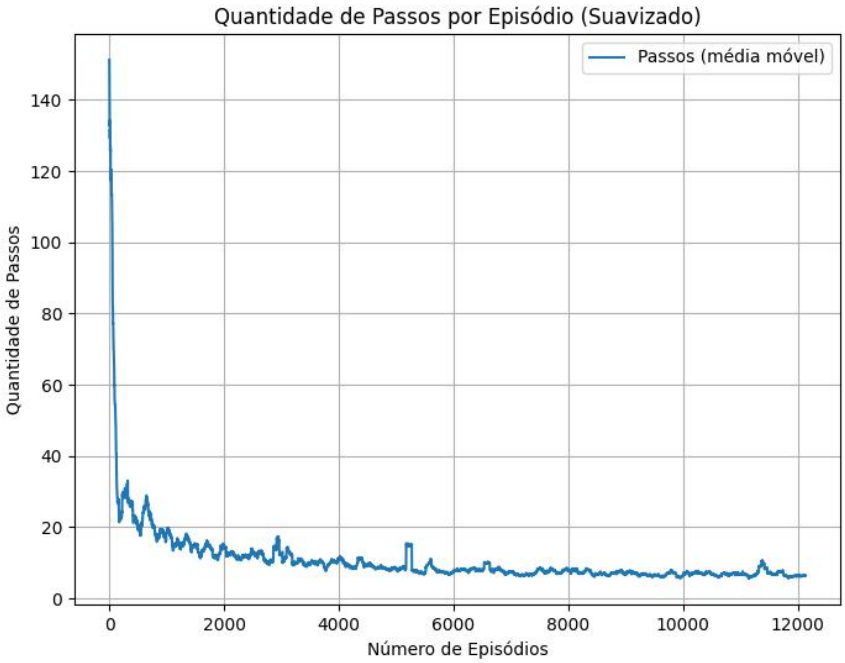

Figura 5.17: Resultados referente a quantidade de passos necessários para concluir um episódio no Experimento 2.

apresentados na Figura 5.18, enquanto a variância das recompensas é exibida na Figura 5.19. Observa-se que, com o passar do tempo, o agente aprende a maximizar suas recompensas, ao mesmo tempo em que reduz a variabilidade dos valores recebidos.

Por fim, os resultados da perda (*loss*) estão apresentados na Figura 5.20. Esse resultado é bastante significativo, pois mostra que a função de perda escolhida, o erro quadrático médio (MSE), descrita na Subseção 4.2.4 na Equação 4.7, diminui a cada iteração do agente com o ambiente, indicando que ele está de fato aprendendo seu objetivo final.

5.3.3 Visualização das Rotas Aprendidas

Nesta seção, são apresentados exemplos de trajetórias aprendidas pelo agente após o processo de treinamento, demonstrando sua capacidade de adaptação a diferentes configurações do ambiente. Serão demonstrados cenários em que o agente precisa encontrar a melhor rota até a saída, sem inserção de obstáculos. Também serão abordados casos onde serão inseridos os obstáculos de congestionamento, obstáculo de fogo e, por fim, um cenário combinado com os dois obstáculos acontecendo simultaneamente.

5.3.3.1 Cenário 1 - Rotas sem Obstáculos

Este cenário representa a configuração mais simples para o processo de aprendizado, sendo o trajeto ideal entre a posição inicial do agente e a saída do edifício. Nele, não há a presença de congestionamentos ou outros obstáculos, sendo assim, o agente tem total liberdade para escolher sua rota de evacuação. Assim,

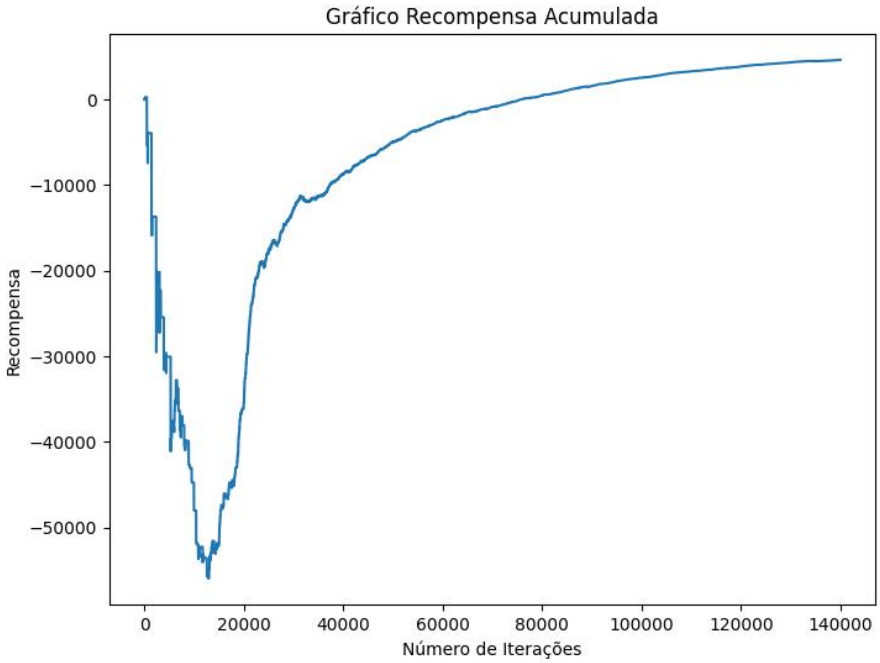

Figura 5.18: Resultados referente a média das recompensas do Experimento 2.

o único critério de decisão adotado pelo agente é a busca pelo caminho mais curto, levando-o a selecionar a saída mais próxima com a menor distância percorrida.

Todas as rotas calculadas para cada nó do grafo individualmente podem ser consultadas na Tabela 2 do Apêndice I.1. Observa-se que o agente foi capaz de aprender o caminho até a saída a partir de todos os nós do ambiente. Isso demonstra que uma das principais etapas do objetivo foi atingida: entender quais são os nós de saída e identificar que eles representam o objetivo final a ser alcançado.

Para ilustrar esse cenário, foram considerados cinco nós distintos, localizados em diferentes regiões do grafo: 66, 73, 92, 105 e 119. Esses nós foram escolhidos por estarem situados no piso superior, enquanto, no experimento anterior, foram analisados nós do piso inferior. A Tabela 5.14 apresenta os nós iniciais e os respectivos caminhos percorridos pelo agente até alcançar um dos nós de saída.

Tabela 5.14: Experimento 2 - Caminhos percorridos pelo agente em um ambiente sem obstáculos, com as respectivas recompensas (Subseção 4.2.2) e distâncias aproximadas (Subseção 4.1.2.1).

Nó Inicial	**Caminho Percorrido**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
66	$66 \rightarrow 67 \rightarrow 70 \rightarrow 77 \rightarrow 80 \rightarrow 64 \rightarrow 0$	9799	436
73	$73 \rightarrow 70 \rightarrow 77 \rightarrow 80 \rightarrow 64 \rightarrow 0$	9838	397
92	$92 \rightarrow 83 \rightarrow 80 \rightarrow 64 \rightarrow 0$	9825	410
105	$105 \rightarrow 104 \rightarrow 101 \rightarrow 96 \rightarrow 91 \rightarrow 83 \rightarrow 80 \rightarrow 80 \rightarrow 64 \rightarrow 0$	9634	601
119	$119 \rightarrow 117 \rightarrow 109 \rightarrow 101 \rightarrow 96 \rightarrow 91 \rightarrow 83 \rightarrow 80 \rightarrow 80 \rightarrow 64 \rightarrow 0$	9573	662

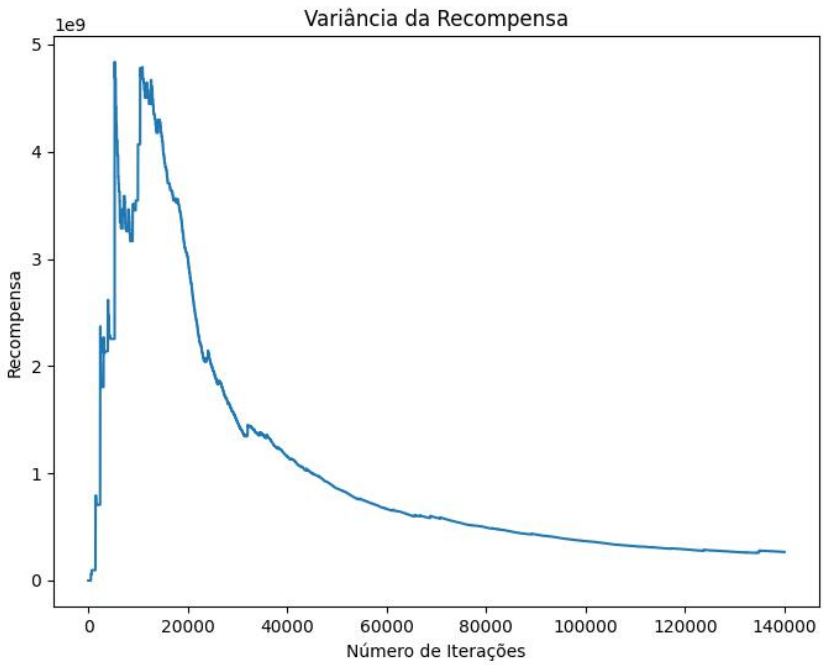

Figura 5.19: Resultados referente a variância das recompensas do Experimento 2.

A Figura 5.21 ilustra visualmente os caminhos aprendidos pelo agente a partir dos nós escolhidos anteriormente. Observa-se que o agente foi capaz de identificar caminhos eficientes em termos de distância e recompensa acumulada. Interessante notar que nessa simulação, todos os nós utilizam caminhos que levam a saída pelo nó 0.

Além disso, como já observado no experimento 1, a recompensa acumulada tende a ser maior quando a distância até a saída é menor, reforçando a correlação entre o critério de recompensa adotado e a eficiência da trajetória. Essa relação indica que o agente aprendeu a associar trajetos mais curtos a um melhor desempenho, reconhecendo a distância como um fator relevante a ser considerado.

5.3.3.2 Cenário 2 - Rotas com congestionamento

Como o agente foi treinado considerando os cenários com presença de fogo, congestionamento e a combinação de ambos, é possível simular um ambiente contendo apenas congestionamento. A análise desse cenário é relevante para a pesquisa, pois ajuda a entender de forma mais clara como o agente responde especificamente a esse tipo de obstáculo, sem a interferência de outros fatores, como o fogo.

Nesse cenário 2 foram escolhidos dois nós para simulação de congestionamento no ambiente, sendo eles os nós 64 e 91, ambos com 6 agentes virtuais representando as áreas de alta densidade. Esses nós tiveram papel importante no cenário anterior (Subseção 5.3.3.1), sendo utilizados como parte dos caminhos ideais calculados.

Com a adição desses congestionamentos, o objetivo desta etapa é avaliar a capacidade de adaptação do agente com mudanças nas condições do ambiente, avaliando se ele consegue evitar os nós congestionados

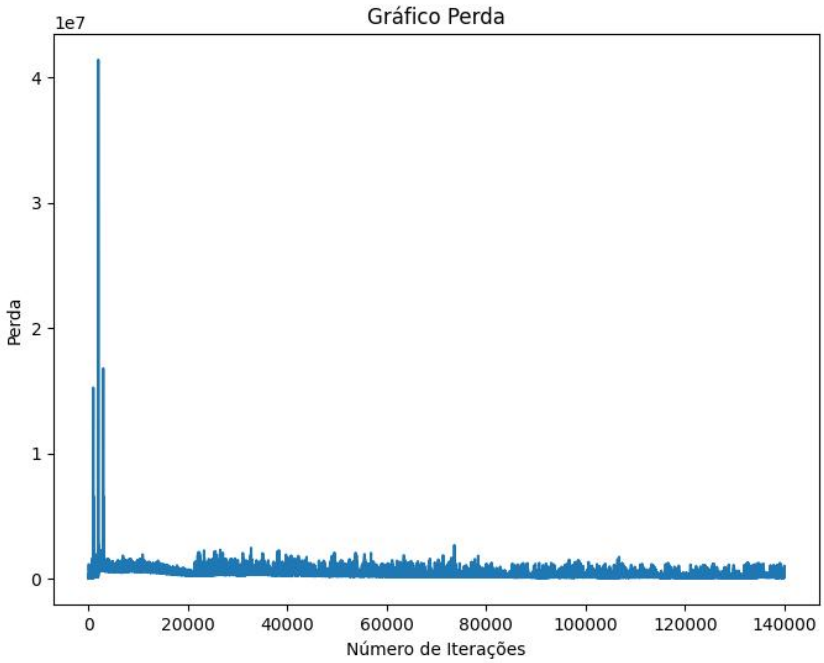

Figura 5.20: Resultados referente a perda das recompensas do Experimento 2.

e, ao mesmo tempo, encontrar rotas eficientes e seguras até a saída.

É importante destacar que os cenários com congestionamento em diferentes nós foram simulados separadamente, mas serão apresentados em conjunto para facilitar a otimização e a análise comparativa dos resultados. Também será adotada a mesma abordagem utilizada no experimento anterior, apresentada na Subseção 5.2.3.2, com a apresentação dos resultados de duas simulações distintas para fins de comparação:

- Primeira análise – Caminhos anteriores forçados: Nesta simulação, embora o ambiente tenha sido modificado para a inserção de nós congestionados, o agente foi obrigado a seguir os mesmos caminhos utilizados no cenário anterior, sem obstáculos (Subseção 5.2.3.2). O objetivo é analisar o impacto direto das penalizações na recompensa total ao manter as rotas anteriormente aprendidas. Os resultados estão apresentados na Tabela 5.15.
- Segunda análise – Inferência livre com base no novo ambiente: Aqui, o agente teve liberdade para decidir seus caminhos com base na nova configuração do ambiente, levando em conta as penalizações associadas aos nós congestionados. O objetivo é avaliar se o modelo aprendeu a evitar rotas penalizadas, ajustando suas decisões de forma inteligente. Os trajetos inferidos pelo agente nessa simulação estão apresentados na Tabela 5.16.

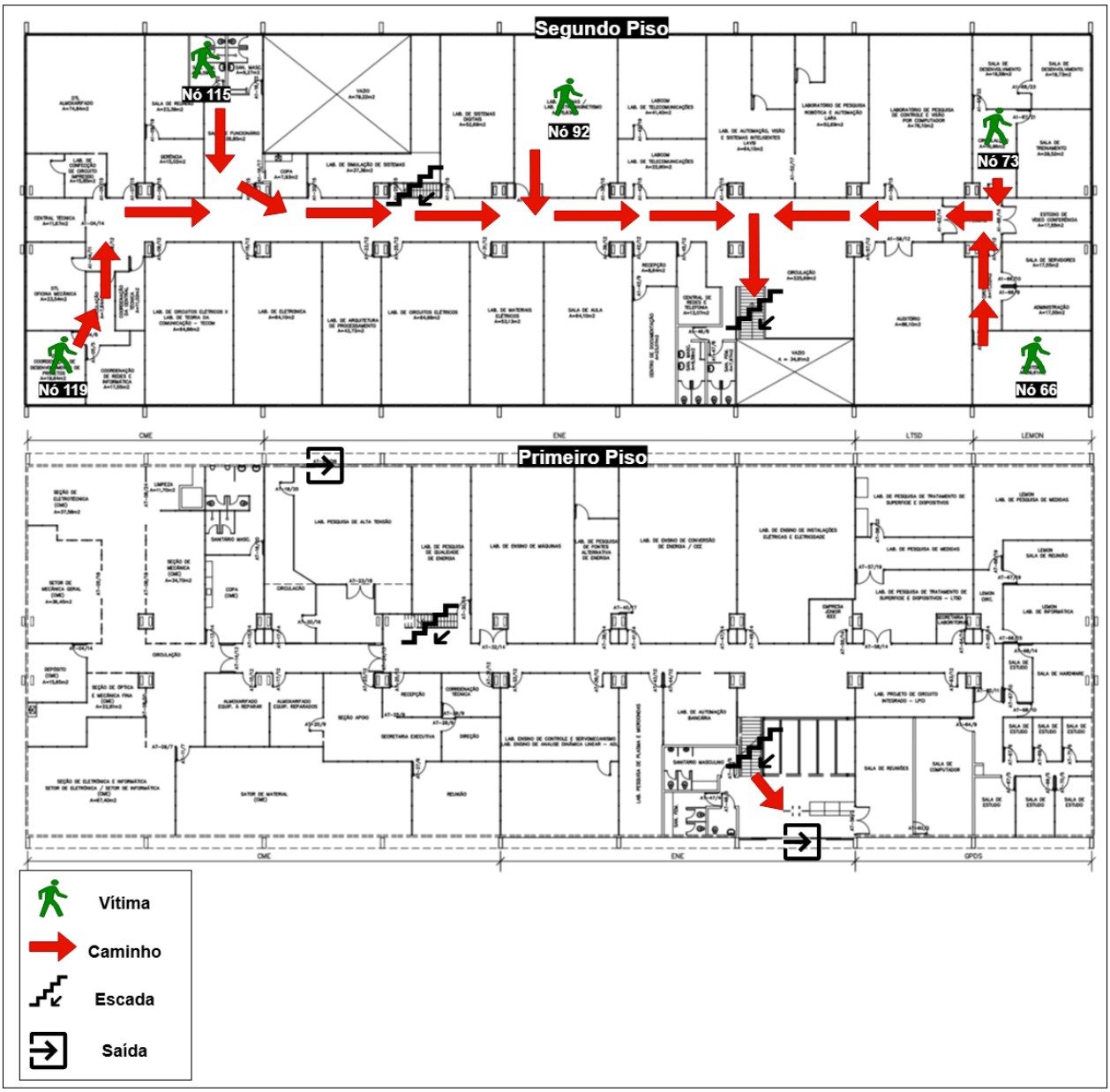

Figura 5.21: Experimento 2 - Resultado do agente em um ambiente sem obstáculos.

Tabela 5.15: Experimento 2 - Recompensas e distâncias ao forçar o agente a repetir os mesmos caminhos utilizados no ambiente sem congestionamento, agora com congestionamento nos nós 64 e 91.

Nó Inicial	**Caminho Percorrido**	**Nó congestionado**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
66	66 → 67 → 70 → 77 → 80 → 64 → 0	64	9299	436
73	73 → 70 → 77 → 80 → 64 → 0	64	9338	397
92	92 → 83 → 80 → 64 → 0	64	9325	410
105	105 → 104 → 101 → 96 → 91 → 83 → 80 → 80 → 64 → 0	91	9134	601
119	119 → 117 → 109 → 101 → 96 → 91 → 83 → 80 → 80 → 64 → 0	91	9073	662

Tabela 5.16: Experimento 2 - Recompensas e distâncias dos caminhos efetivamente escolhidos pelo agente ao enfrentar o ambiente com congestionamento nos nós 64 e 91.

Nó Inicial	**Caminho Percorrido**	**Nó congestionado**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
66	66 → 67 → 70 → 77 → 80 → 64 → 0	64	9299	436
73	73 → 70 → 77 → 80 → 64 → 0	64	9338	397
92	92 → 83 → 80 → 64 → 0	64	9325	410
105	105 → 104 → 101 → 96 → 98 → 41 → 50 → 49 → 47 → 1	91	9519	525
119	119 → 117 → 109 → 101 → 96 → 98 → 41 → 50 49 → 47 → 1	91	9459	586

Comparando as Tabelas 5.15 e 5.16, observa-se que, em alguns nós iniciais, a recompensa acumulada final é menor quando o agente percorre os mesmos caminhos do cenário anterior, agora em um ambiente com congestionamento. Isso indica que, nesses casos, quando o agente tem a liberdade para encontrar o caminho alternativo ideal, o agente optou por realizar desvios que resultaram em uma melhor recompensa total. Por outro lado, há situações em que o agente manteve o caminho original, mesmo com a presença de congestionamento, por avaliar que, mesmo com a presença desse obstáculo, esse caminho ainda é mais vantajoso do que realizar um desvio. Esses comportamentos demonstram que o agente é capaz de ponderar entre manter um caminho ou desviá-lo, com base no custo-benefício da recompensa acumulada.

Embora o agente receba penalizações por passar por nós congestionados, ele aprendeu a avaliar o custo-benefício de suas decisões. Quando o caminho alternativo representa um aumento significativo no tamanho do trajeto, o agente opta por enfrentar o congestionamento, desde que isso proporcione uma saída, em sua opinião, mais eficiente. Esse comportamento é observado nos caminhos escolhidos a partir dos nós 66, 73 e 92, que percorrem a mesma rota com ou sem o congestionamento no nó 64, indicando que,

mesmo penalizado, esse ainda é o caminho mais eficiente em termos de recompensa acumulada e distância percorrida no entendimento do agente.

Neste trabalho, optou-se por atribuir uma penalidade negativa baixa aos nós congestionados, visando permitir que o agente aprenda a evitá-los apenas quando for vantajoso em termos da recompensa acumulada aproximada total, e não de forma absoluta. Essa abordagem é o contrário da recompensa para os nós atingidos por fogo, nos quais a penalização escolhida foi severa, forçando o agente a evitá-los independentemente da distância envolvida.

Uma observação interessante é que para os nós 105 e 119, observa-se que os caminhos alternativos escolhidos, que evitam o nó congestionado 91, passam a utilizar o nó 1 como saída. No entanto, esses novos caminhos apresentam recompensas significativamente menores em comparação com os resultados do cenário sem congestionamento, apresentados na Tabela 5.14. Esse resultado reforça a decisão inicial do agente no cenário 1, em que o caminho que leva à saída pelo nó 0 foi considerado mais vantajoso. Assim, fica evidente que, na ausência de congestionamento, o agente avaliou corretamente que o caminho que leva a saída do nó 0 oferecia maior recompensa acumulada e, portanto, era a opção mais eficiente.

A Figura 5.22 apresenta os caminhos ideais inferidos pelo agente para o cenário com congestionamentos, evidenciando as rotas escolhidas para evitar ou não os nós penalizados.

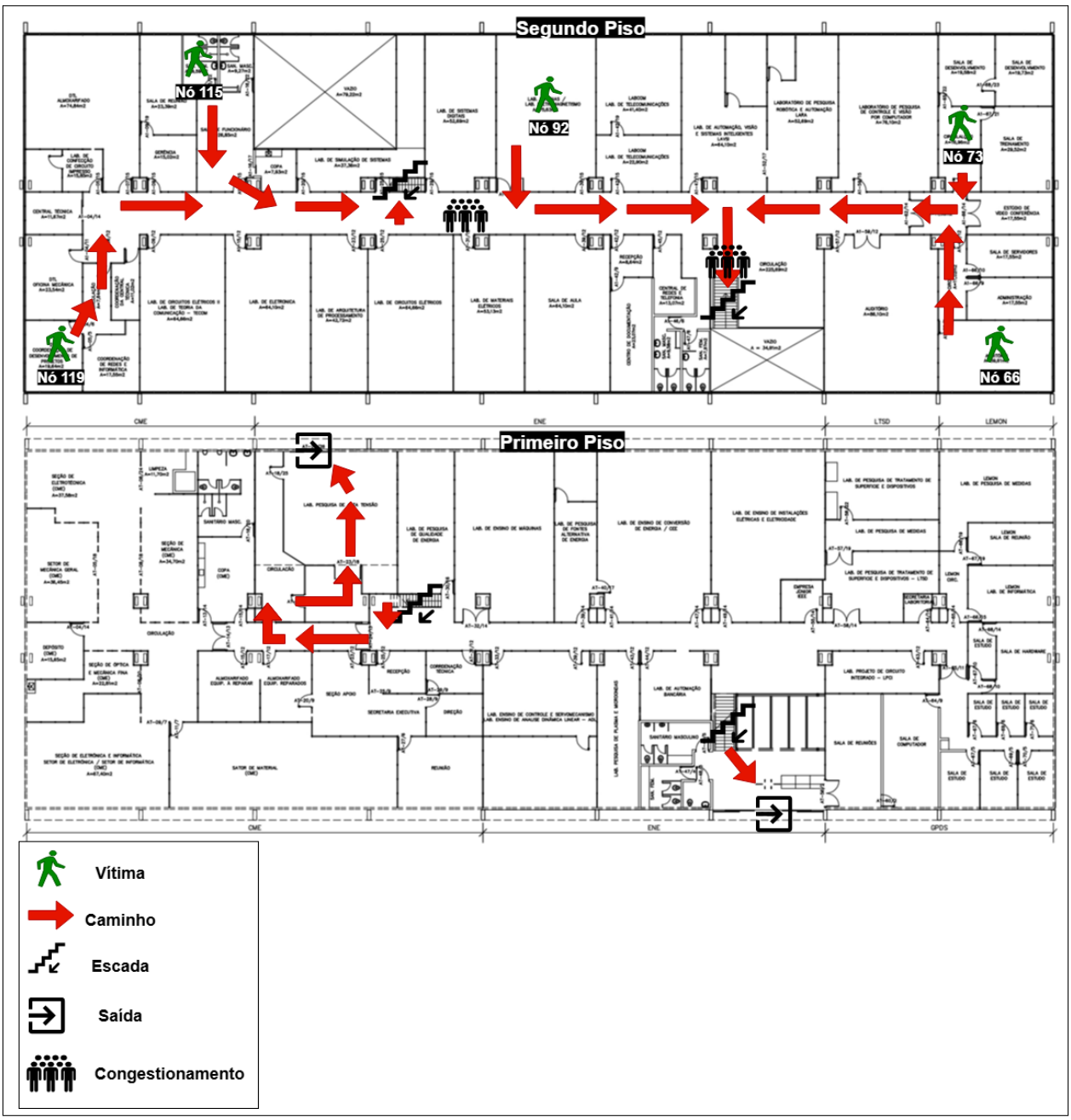

Figura 5.22: Experimento 2 - Resultado do agente ao enfrentar congestionamento nos nós 64 e 91.

5.3.3.3 Cenário 3 - Rotas com nós em chamas

De forma semelhante ao cenário anterior, cujo objetivo foi avaliar a capacidade do agente de aprender em situações com presença exclusiva de congestionamentos, este novo cenário foi elaborado para analisar o comportamento do agente diante apenas de nós em chamas.

O nó 64 foi utilizado em todos os caminhos apresentados no cenário 1 deste experimento (Subseção 5.3.3.1), e, portanto, foi escolhido como o nó em chamas nesta nova simulação. A escolha pretende avaliar como o agente reage quando um nó fundamental dos caminhos ideais passa a representar uma ameaça severa, exigindo que ele busque caminhos alternativos para alcançar o seu objetivo final com segurança.

Para comparação, foram feitas duas simulações distintas:

- Primeira análise – Caminhos anteriores forçados: Nesta simulação, embora o ambiente tenha sido modificado com a introdução de nós com presença de fogo, o agente foi obrigado a seguir os mesmos caminhos utilizados no primeiro cenário, sem obstáculos (Subseção 5.2.3.2). O objetivo é analisar o impacto direto das penalizações na recompensa total ao manter as rotas anteriormente aprendidas. Os resultados estão apresentados na Tabela 5.17.
- Segunda análise – Inferência livre com base no novo ambiente: Aqui, o agente teve liberdade para decidir seus caminhos com base na nova configuração do ambiente, levando em conta as penalizações associadas aos nós com presença de fogo. O objetivo é avaliar se o modelo aprendeu a evitar rotas penalizadas, ajustando suas decisões de forma inteligente. Os trajetos inferidos pelo agente nessa simulação estão apresentados na Tabela 5.18.

Tabela 5.17: Experimento 2 - Recompensas e distâncias ao forçar o agente a repetir os mesmos caminhos utilizados no ambiente sem presença de fogo, agora com fogo no nó 64

Nó Inicial	**Caminho Percorrido**	**Nó com fogo**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
66	66 $\rightarrow$ 67 $\rightarrow$ 70 $\rightarrow$ 77 $\rightarrow$ 80 $\rightarrow$ 64 $\rightarrow$ 0	64	-5158	436
73	73 $\rightarrow$ 70 $\rightarrow$ 77 $\rightarrow$ 80 $\rightarrow$ 64 $\rightarrow$ 0	64	-5120	397
92	92 $\rightarrow$ 83 $\rightarrow$ 80 $\rightarrow$ 64 $\rightarrow$ 0	64	-5132	410
105	105 $\rightarrow$ 104 $\rightarrow$ 101 $\rightarrow$ 96 $\rightarrow$ 91 $\rightarrow$ 83 $\rightarrow$ 80 $\rightarrow$ 80 $\rightarrow$ 64 $\rightarrow$ 0	64	-5324	601
119	119 $\rightarrow$ 117 $\rightarrow$ 109 $\rightarrow$ 101 $\rightarrow$ 96 $\rightarrow$ 91 $\rightarrow$ 83 $\rightarrow$ 80 $\rightarrow$ 80 $\rightarrow$ 64 $\rightarrow$ 0	64	-5384	662

Tabela 5.18: Experimento 2 - Recompensas e distâncias dos caminhos efetivamente escolhidos pelo agente ao enfrentar o ambiente presença de fogo no nó 64

Nó Inicial	**Caminho Percorrido**	**Nó com fogo**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
66	$66 \rightarrow 67 \rightarrow 70 \rightarrow 77 \rightarrow 880 \rightarrow 83 \rightarrow 91 \rightarrow 96 \rightarrow 98 \rightarrow 41 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	64	9314	731
73	$73 \rightarrow 70 \rightarrow 77 \rightarrow 80 \rightarrow 83 \rightarrow 91 \rightarrow 96 \rightarrow 98 \rightarrow 41 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	64	9352	692
92	$92 \rightarrow 91 \rightarrow 96 \rightarrow 98 \rightarrow 41 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	64	9545	500
105	$105 \rightarrow 104 \rightarrow 101 \rightarrow 96 \rightarrow 98 \rightarrow 41 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	64	9519	525
119	$119 \rightarrow 117 \rightarrow 109 \rightarrow 101 \rightarrow 96 \rightarrow 98 \rightarrow 41 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	64	9459	586

Percebe-se que a penalização aplicada quando o agente é forçado a atravessar um nó em chamas é extremamente severa, resultando em recompensas altamente negativas nos casos testados (Tabela 5.17). Esse resultado demonstra que a função de recompensa foi eficaz em desencorajar o agente a escolher caminhos que envolvem esse tipo de obstáculo.

Uma observação relevante diz respeito à distinção entre evitar um nó quando necessário e evitá-lo a todo custo. No cenário apresentado na Subseção 5.3.3.2, ao inserir um congestionamento no nó 64, o agente considerou que, para os nós iniciais 66, 73 e 92, era mais vantajoso manter o caminho original, mesmo com penalização por passar por um nó congestionado, por entender que seguir um caminho alternativo não valeria a pena. No entanto, neste cenário com fogo no nó 64, o comportamento do agente muda significativamente, pois ele evita completamente qualquer caminho que passe por um nó em chamas, mesmo quando a alternativa representa um aumento considerável na distância percorrida.

Esse comportamento fica evidente ao perceber que os caminhos alternativos escolhidos dos nós 66 e 73 são mais longos. Para o nó 92, embora o caminho alternativo também seja maior, a diferença em relação ao caminho original é pequena, sendo medida apenas em poucos *pixels*. Isso explica por que, no cenário anterior, o agente ainda optava pelo caminho congestionado; porém, diante de um obstáculo mais crítico como o fogo, ele altera sua estratégia para garantir maior segurança, mesmo ao custo de maior deslocamento.

A Figura 5.23 apresenta as trajetórias inferidas pelo agente para cada nó inicial considerado no cenário com o nó 64 em chamas.

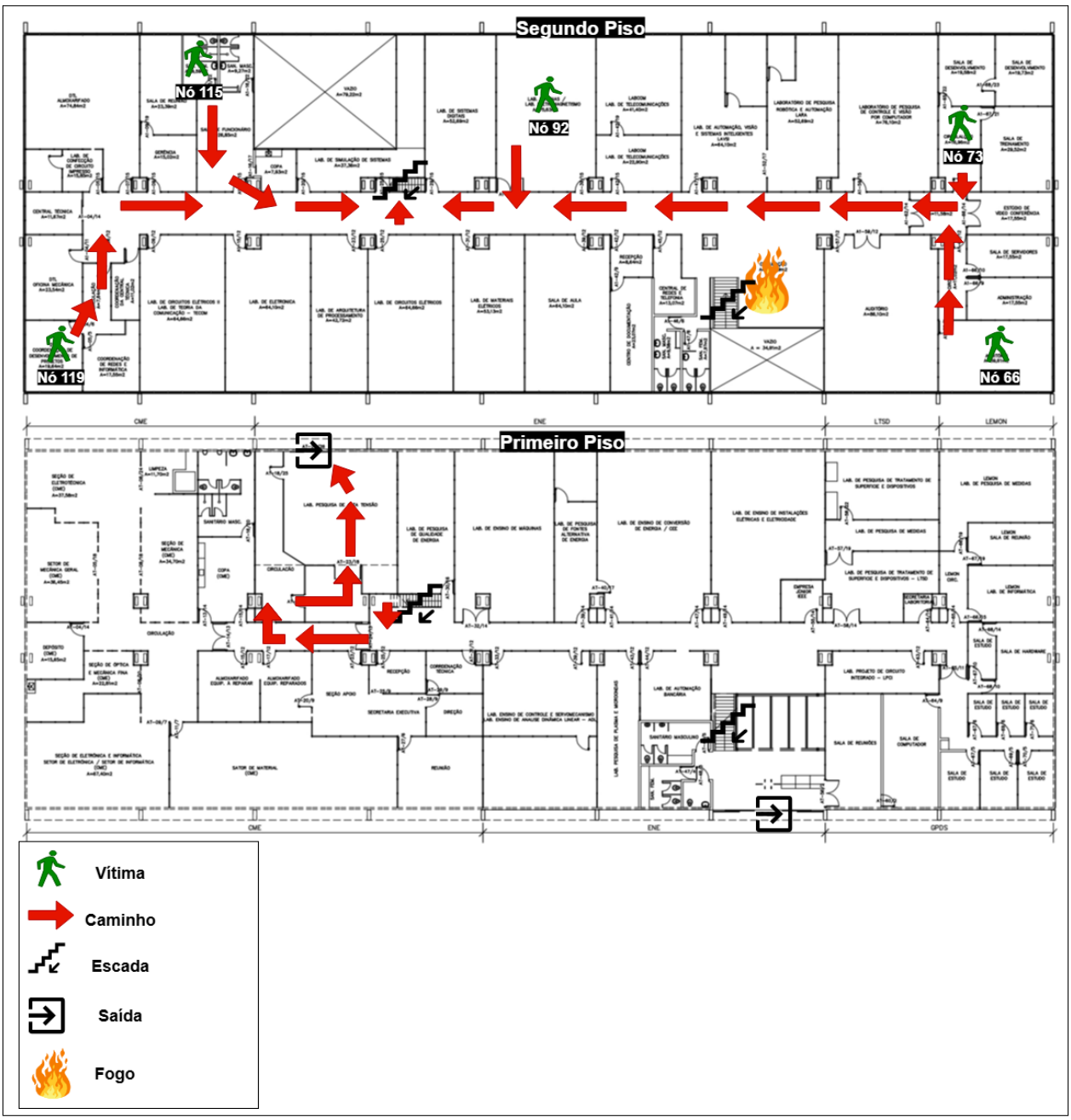

Figura 5.23: Resultado do Experimento 2 - Resultado do agente ao enfrentar um ambiente com fogo no nó 64.

5.3.3.4 Cenário 4 - Rotas com nós em chamas e congestionamento

Por fim, o Cenário 4 apresenta a combinação dos dois tipos de obstáculos abordados neste trabalho, sendo a presença simultânea de nós em chamas e de congestionamentos. O objetivo dessa análise é avaliar a capacidade do agente de lidar com múltiplos obstáculos de naturezas diferentes, adaptando sua política de decisão para encontrar caminhos viáveis e com o melhor retorno possível, mesmo em ambientes altamente perigosos.

Dando continuidade à análise do cenário anterior, em que foi observado o impacto da presença de fogo em um nó crítico (nó 64) para as rotas de fuga com saída pelo nó 0, este novo experimento introduz uma

complexidade adicional. Além de manter o foco de incêndio no nó 64, será inserido um congestionamento no nó 50, o qual é utilizado por todas as rotas alternativas calculadas anteriormente. O objetivo é avaliar como o agente se comporta diante da obstrução simultânea de seu caminho principal por fogo e do seu caminho alternativo por congestionamento.

Para simular o congestionamento, foram inseridos seis agentes virtuais no nó 50, enquanto o nó 64 foi mantido como ponto de foco de incêndio. A Tabela 5.19 apresenta os resultados obtidos nesse cenário combinado de obstáculo por fogo e congestionamento.

Tabela 5.19: Recompensas e distâncias dos caminhos efetivamente escolhidos pelo agente do experimento 2 ao enfrentar o ambiente com presença de fogo no nó 64 e congestionamento no nó 50

Nó Inicial	**Caminho Percorrido**	**Nó com fogo**	**Nó congestionado**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
66	$66 \rightarrow 67 \rightarrow 70 \rightarrow 77 \rightarrow 880 \rightarrow 83 \rightarrow 91 \rightarrow 96 \rightarrow 98 \rightarrow 41 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$	64	50	9260	807
73	$73 \rightarrow 70 \rightarrow 77 \rightarrow 80 \rightarrow 83 \rightarrow 91 \rightarrow 96 \rightarrow 98 \rightarrow 41 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$	64	50	9299	768
92	$92 \rightarrow 91 \rightarrow 96 \rightarrow 98 \rightarrow 41 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$	64	50	9491	576
105	$105 \rightarrow 104 \rightarrow 101 \rightarrow 96 \rightarrow 98 \rightarrow 41 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$	64	50	9465	602
119	$119 \rightarrow 117 \rightarrow 109 \rightarrow 101 \rightarrow 96 \rightarrow 98 \rightarrow 41 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$	64	50	9405	662

Com base nos resultados apresentados na Tabela 5.19, observa-se que, ao se deparar com bloqueios simultâneos em dois de seus caminhos principais, o agente foi capaz de identificar um terceiro caminho para uma saída segura. Diferente do cenário anterior, apresentado na Subseção 5.3.3.3, em que todas as rotas inferidas utilizavam o nó 1 como saída, neste cenário o agente optou por caminhos alternativos que levam ao nó 0. Em todos os cinco nós iniciais considerados, o agente avaliou ser mais vantajoso evitar o nó 50 congestionado e buscar uma nova rota até uma das saídas. Isso sugere que, mesmo que as rotas que levam ao nó 0 sejam mais longas, elas ainda apresentam melhor custo-benefício em termos de segurança e eficiência diante dos obstáculos simulados.

Para fins de comparação e para fundamentar a decisão tomada pelo agente, a Tabela 5.20 apresenta os resultados obtidos caso ele fosse forçado a seguir pelo caminho congestionado, tal como no cenário anterior, passando obrigatoriamente pelo nó 50 até alcançar o nó de saída 1. Os valores de recompensa acumulada mostram que, ao adotar esses caminhos, a recompensa acumulada pelo agente é significativamente inferior àquela obtida no cenário apresentado na Tabela 5.19, em que ele teve liberdade para desviar

e buscar caminhos alternativos. Esses resultados reforçam a efetividade da política aprendida, que foi capaz de identificar trajetos mais vantajosos mesmo diante de obstáculos simultâneos.

Tabela 5.20: Experimento 2 - Recompensas e distâncias ao forçar o agente a repetir os mesmos caminhos utilizados no ambiente do cenário 3, mas com fogo no nó 64 e congestionamento no nó 50.

Nó Inicial	**Caminho Percorrido**	**Nó com fogo**	**Nó congestionado**	**Recompensa Aproximada** ($r \times 10^3$)	**Distância Aproximada** ($d \times 10^3$)
66	$66 \rightarrow 67 \rightarrow 70 \rightarrow 77 \rightarrow 880 \rightarrow 83 \rightarrow 91 \rightarrow 96 \rightarrow 98 \rightarrow 41 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	64	50	8814	731
73	$73 \rightarrow 70 \rightarrow 77 \rightarrow 80 \rightarrow 83 \rightarrow 91 \rightarrow 96 \rightarrow 98 \rightarrow 41 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	64	50	8852	692
92	$92 \rightarrow 91 \rightarrow 96 \rightarrow 98 \rightarrow 41 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	64	50	9045	500
105	$105 \rightarrow 104 \rightarrow 101 \rightarrow 96 \rightarrow 98 \rightarrow 41 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	64	50	9019	525
119	$119 \rightarrow 117 \rightarrow 109 \rightarrow 101 \rightarrow 96 \rightarrow 98 \rightarrow 41 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$	64	50	8959	586

A Figura 5.24 apresenta as trajetórias inferidas pelo agente para cada nó inicial considerado no cenário com o nó 64 em chamas e o nó 50 congestionado.

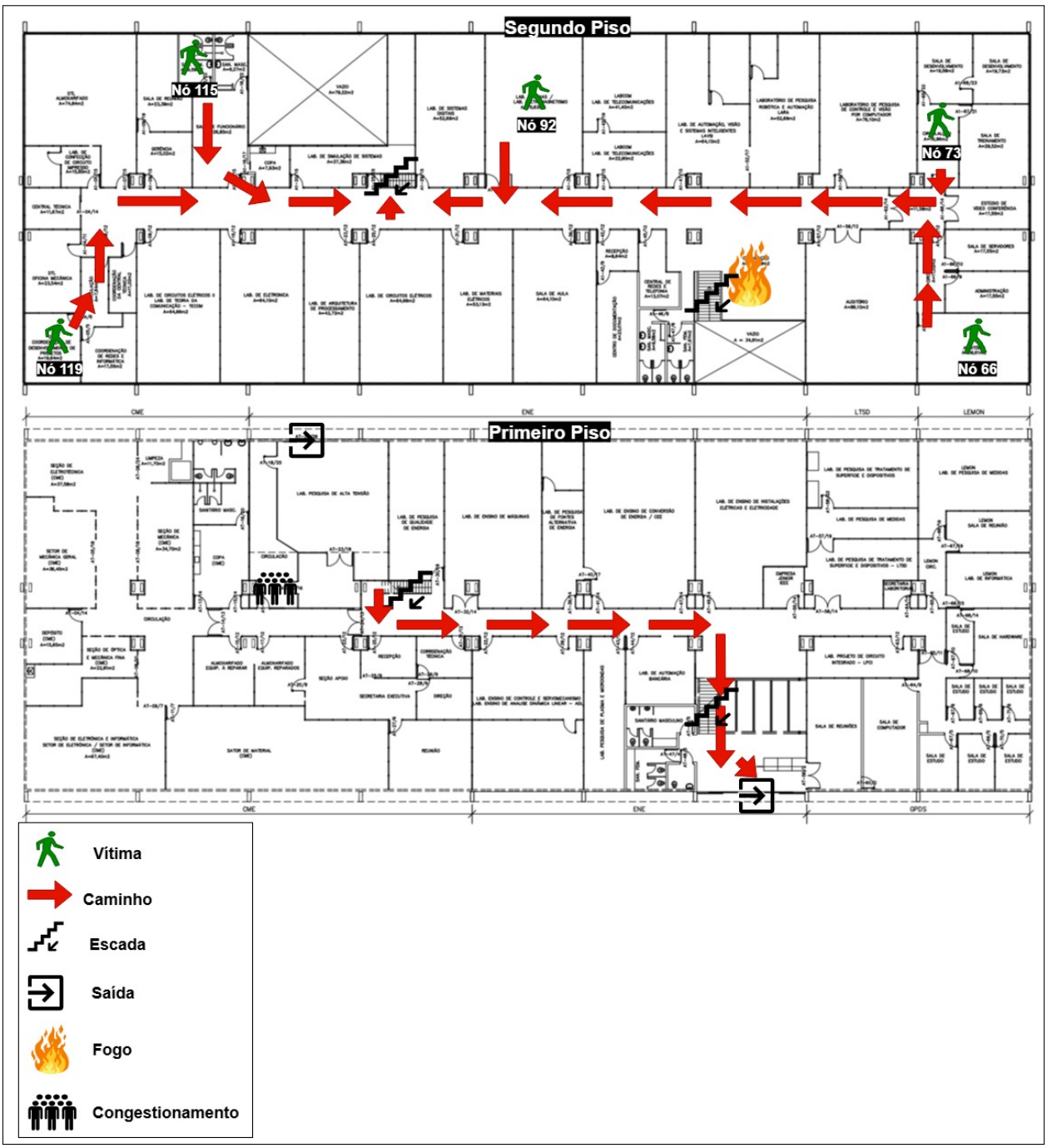

Figura 5.24: Resultado do Experimento 2 - Resultado do agente ao enfrentar um ambiente com fogo no nó 64 e congestionamento no nó 50.

5.4 AVALIAÇÃO DA TOPOLOGIA DE SAÍDAS NOS CENÁRIOS DE EVACUAÇÃO

Um dos aspectos que merece atenção é a quantidade de saídas de emergência e a presença de nós críticos na planta do Laboratório SG11. Nos dois experimentos apresentados, consideraram-se apenas duas saídas de emergência e duas escadas que conectam os dois pavimentos. Contudo, diversos setores do laboratório contêm nós críticos, isto é, nós que, se bloqueados por fogo, impedem completamente o

acesso de uma vítima localizada neles, até o exterior por meio de um caminho seguro. Essa limitação estrutural compromete a eficiência da evacuação, pois a inexistência de caminhos alternativos deixa certas áreas suscetíveis ao isolamento em cenários de emergências.

Na planta do Experimento 2 (Figura 5.14), identificam-se 40 nós críticos em um total de 119, o que representa aproximadamente 34% do grafo, um número considerável. Entre esses nós, destaca-se o nó 96, que, se obstruído, isola 22 outros nós (do 97 ao 119). O nó 101 também se mostra particularmente relevante, pois sua obstrução compromete diretamente o acesso de 18 nós (do 102 ao 119). Outro ponto de atenção é o corredor formado pelos nós 80, 77 e 70, utilizado como caminho de passagem por múltiplos caminhos do grafo; assim, um foco de incêndio em qualquer um desses nós é suficiente para bloquear diversos caminhos simultaneamente. Por fim, o nó 50 atua como um ponto central para os nós de 55 a 63, representando um gargalo crítico no pavimento inferior do grafo em caso de incêndio.

Considerando essas vulnerabilidades citadas, este trabalho sugere algumas melhorias no laboratório SG11 da Universidade de Brasília. A primeira é o reforço da proteção dos principais nós críticos, por meio da instalação de sensores de fumaça e temperatura, isolamento térmico e *sprinklers* reforçados nos nós 96, 101, 80, 77, 70 e 50. Além disso, recomenda-se o aumento da quantidade de caminhos alternativos, com a criação de novas saídas de emergência em áreas atualmente atendidas apenas por caminhos que passam por gargalos, bem como a abertura de corredores ou portas corta-fogo que permitam contornar esses pontos críticos.

Por fim, sugere-se a realização periódica de simulações computacionais de evacuação, utilizando agentes inteligentes e cenários com focos de incêndio variados, a fim de verificar se as alterações implementadas de fato melhoram o desempenho das rotas de fuga. Essas medidas aumentam a resiliência da planta do laboratório, minimizando o risco em casos de incêndio e otimizando o tempo de evacuação em emergências.

É importante destacar que esta análise possui caráter preliminar e superficial, uma vez que considera exclusivamente a topologia representada pela imagem das plantas baixas transformada em grafo. Elementos arquitetônicos e estruturais relevantes, como a largura dos corredores, a presença e dimensões de janelas que possam funcionar como saídas de emergência, características da planta elétrica, materiais utilizados e demais aspectos construtivos não foram levados em consideração neste estudo. Assim, as sugestões aqui apresentadas se baseiam apenas nos critérios computacionais e topológicos considerados na modelagem e simulação proposta.

5.5 COMPARAÇÃO COM TRABALHOS RELACIONADOS

O trabalho de [1] propôs uma abordagem baseada em Aprendizado por Reforço com *transfer learning* para resolver o problema de evacuação em ambientes internos, considerando apenas o obstáculo de fogo. Uma das limitações apontadas naquele estudo era a dificuldade de lidar com grafos grandes, sendo que a maior planta baixa analisada continha apenas 35 nós.

Neste trabalho, essa limitação foi superada por meio de mudanças na forma como o ambiente é representado, permitindo a utilização de grafos significativamente maiores sem comprometer a escalabilidade ou a viabilidade do treinamento.

Em relação ao tempo de treinamento, os valores apresentados em [1] eram consideravelmente menores do que os apresentados aqui. O maior tempo de treinamento no trabalho anterior foi de aproximadamente 7 minutos (Projeto 3), utilizando o mesmo ambiente computacional. No entanto, a abordagem adotada em [1] realizava o treinamento do agente inicialmente em um ambiente sem obstáculos, visando encontrar o menor caminho até a saída. Quando surgia a necessidade de incorporar obstáculos, o modelo era retreinado por meio de *transfer learning*.

Neste trabalho, é proposto uma estratégia diferente, onde o agente é treinado desde o início, considerando os obstáculos como parte da entrada da rede neural. Com isso, não é necessário retreiná-lo para cada nova configuração de obstáculos (como inserção de fogo e/ou congestionamento), o que torna o modelo mais flexível e adequado a aplicações em tempo real, mesmo que à custa de tempos de treinamento mais longos, inicialmente.

Para fins de comparação, foi utilizado o notebook de treinamento no Google Colab [66], indicado no trabalho original [1], para simular o ambiente do Projeto 1 desenvolvido neste trabalho, conforme descrito na Subseção 5.2.

Foram utilizados os hiperparâmetros recomendados pelo trabalho [1], conforme apresentados na Tabela 5.21. Conforme a recomendação do estudo, o número de iterações (*episodes*) é proporcional à quantidade de arestas do grafo. Assim, para este experimento, foi utilizado um total de 72.000 iterações.

Tabela 5.21: Hiperparâmetros utilizados conforme o trabalho [1]

Hiperparâmetro	**Valor**
beta (β)	1
Taxa de aprendizado (lr)	1×10^{-2}
Tamanho do batch ($batchsize$)	240
Tamanho do buffer ($buffer_size$)	4.000
Número de iterações ($episodes$)	72.000
Fator de desconto (γ)	0,99
Epsilon (ϵ)	0,5

Como era de se esperar, devido ao tamanho do grafo, os resultados obtidos foram ruins. O agente não conseguiu aprender nenhum caminho até as saídas, mesmo na ausência de obstáculos. A Figura 5.25 apresenta a recompensa acumulada durante o treinamento, enquanto a Figura 5.26 exibe a evolução da função de perda.

Esses resultados já demonstram evoluções significativas em relação ao trabalho anterior, pois, mesmo com grafos maiores e cenários mais complexos, a abordagem aqui proposta foi capaz de lidar com obstáculos e realizar inferências de rotas sem a necessidade de retreinamento a cada alteração no ambiente. Diferentemente da solução anterior, que exigia a aplicação de *transfer learning* a cada modificação e apresentava dificuldades para lidar com grafos com maior número de nós e arestas, a metodologia atual mostrou-se mais robusta e escalável.

Apesar de o Experimento 2 apresentado neste trabalho ser ainda mais complexo, não foi necessária a comparação com o mesmo, uma vez que os resultados obtidos com o Experimento 1 já demonstraram um desempenho ruim na abordagem anterior. Dessa forma, a limitação da solução proposta em [1] ficou

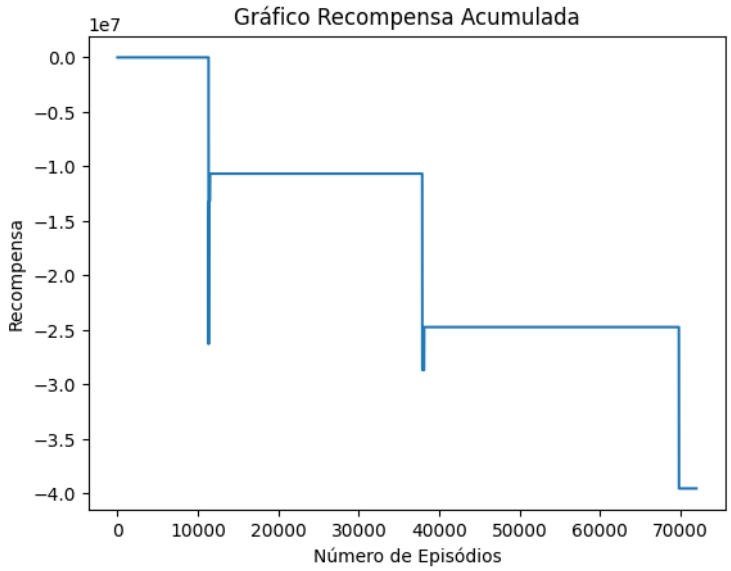

Figura 5.25: Resultados referente a média das recompensas do Experimento 1 utilizando o algoritmo do trabalho [1].

evidente já no projeto menos complexo.

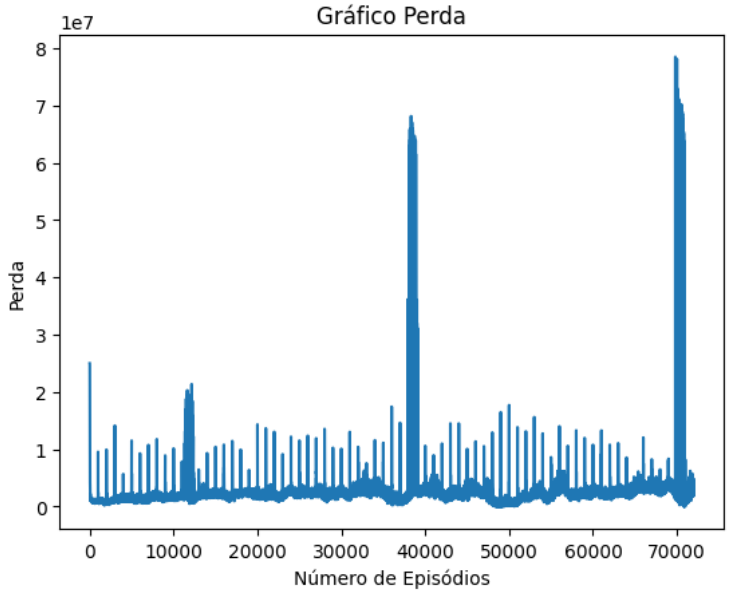

Figura 5.26: Resultados de perda do Experimento 1 utilizando o algoritmo do trabalho [1].

6 CONCLUSÃO

Incêndios estruturais representam um grave problema de segurança em todo o mundo. Uma das principais dificuldades em situações de incêndio é a definição de rotas de evacuação eficientes dentro de edifícios. Este trabalho busca contribuir com esse problema por meio da análise inteligente de rotas de fuga, com a planta baixa do edifício e os focos de incêndio previamente conhecidos.

Para isso, foi adotada uma abordagem baseada em Aprendizado por Reforço Profundo, como continuação do trabalho desenvolvido em [1], utilizando grafos e suas respectivas matrizes adjacentes como ambiente de simulação para o agente. Diferentemente da técnica de *transfer learning* utilizada no trabalho anterior, esta pesquisa propõe incorporar os obstáculos diretamente na definição do estado do agente, permitindo que a rede neural aprenda as possíveis combinações de ambientes já durante o treinamento inicial. Assim, é eliminada a necessidade de retreinamento sempre que o ambiente for alterado, aumentando a flexibilidade e a eficiência da solução proposta.

Além de treinar o agente para percorrer o ambiente representado por um grafo, partindo de um nó inicial até alcançar uma das saídas disponíveis, este trabalho introduz dois tipos de obstáculos: nós em chamas e nós congestionados. O objetivo é que o agente aprenda a evitar completamente os nós em chamas, devido ao alto risco que uma vítima corre ao passar por essa zona, e a evitar os nós congestionados sempre que possível, buscando rotas alternativas mais seguras e eficientes. Para simular o congestionamento, foram inseridos agentes virtuais fixos no início de cada época de treinamento, representando o acúmulo de pessoas tentando evacuar por determinados caminhos. Essa simulação visa refletir situações reais em que o fluxo elevado de vítimas em um mesmo ponto pode comprometer a evacuação. Assim, o agente é incentivado a tomar decisões inteligentes com base não apenas na distância até a saída, mas também na presença de obstáculos ao longo do trajeto.

Para possibilitar todo o experimento, foi desenvolvida uma arquitetura composta por quatro camadas principais. A camada de apresentação é representada por um front-end, no qual o usuário pode enviar a imagem da planta baixa, desenhar o grafo correspondente, iniciar o treinamento para essa planta e visualizar os resultados obtidos pelo modelo treinado. A camada de aplicação é composta por uma API REST, responsável por gerenciar a comunicação entre a camada de apresentação (usuário) e a camada de inteligência, que realiza os treinamentos e as inferências do agente de aprendizagem. Além disso, a camada de aplicação atua como intermediária entre as demais camadas e a camada de persistência, onde são armazenados dados como a imagem da planta baixa, as informações do grafo, os pesos do modelo, entre outros, permitindo análises posteriores.

Com a arquitetura desenvolvida, foram realizados dois experimentos distintos. O primeiro utilizou a planta baixa do primeiro piso do laboratório SG11 da Universidade de Brasília, considerada de porte intermediário. O segundo experimento envolveu a combinação dos dois pisos do mesmo laboratório, caracterizando uma planta baixa de grande porte. Em ambos os cenários, os resultados obtidos foram satisfatórios, pois o agente conseguiu aprender a política de evacuação proposta, compreendendo que seu objetivo final é alcançar uma saída, evitar nós atingidos por fogo a todo custo e, sempre que possível, contornar áreas

congestionadas.

Este trabalho apresenta algumas limitações que devem ser consideradas. A primeira refere-se à dificuldade em definir recompensas fixas que funcionem de maneira eficiente para todos os cenários possíveis, dada a variabilidade das plantas e das situações simuladas. A segunda limitação está relacionada à infraestrutura, pois não foi possível disponibilizar a arquitetura para testes públicos, restringindo seu uso a ambientes locais. A terceira limitação diz respeito à ausência de testes em ambientes reais, já que as validações foram realizadas exclusivamente em simulações, o que pode gerar imprecisões, especialmente na representação das distâncias entre os nós do grafo, que correspondem aos ambientes dos edifícios. Além disso, existe a dependência do usuário para desenhar manualmente o grafo sobre a planta baixa, o que pode introduzir erros ou inconsistências.

Apesar dessas limitações, o presente trabalho demonstra que o Aprendizado por Reforço vai além de aplicações em jogos ou simulações virtuais, sendo uma ferramenta promissora para apoiar a tomada de decisão em cenários complexos de evacuação, contribuindo para a segurança e eficiência nesses contextos.

Como continuidade deste trabalho, diversas melhorias podem ser estudadas para melhorar o sistema proposto. Entre as principais direções futuras, destacam-se:

- Integração com ambientes reais: Validar a abordagem em cenários físicos reais, utilizando sensores e sistemas de monitoramento para atualização dinâmica dos focos de incêndio e congestionamentos.
- Automação na geração dos grafos: Desenvolver métodos automáticos para converter plantas baixas em grafos, reduzindo a dependência do usuário na criação manual.
- Aprimoramento das funções de recompensa: Investigar funções de recompensa dinâmicas e adaptativas que considerem diferentes tipos de riscos e prioridades, possibilitando maior flexibilidade e personalização conforme o ambiente e cenário.
- Treinamento multiagente: Estender o modelo para múltiplos agentes simultâneos, simulando evacuações em grupo e estudando comportamentos cooperativos ou competitivos em emergências.
- Melhoria da infraestrutura de testes: Disponibilizar a arquitetura em nuvem para testes remotos e facilitar o acesso de usuários e pesquisadores, ampliando a validação e uso prático do sistema.

Este trabalho traz contribuições relevantes para a área de segurança em ambientes fechados, especialmente no contexto da segurança cibernética aplicada a sistemas de emergência e automação predial. Além disso, o sistema desenvolvido contribui para o avanço de *smart buildings* e *smart cities*, oferecendo soluções integradas para ambientes inteligentes e seguros.

REFERÊNCIAS BIBLIOGRÁFICAS

1 ROSA, A. C. *Análise de rotas de fuga em ambientes utilizando aprendizado por reforço. 2022. 71 f., il. Trabalho de Conclusão de Curso*. Brasília: [s.n.].

2 Wikipédia. *Mapa de Königsberg no tempo de Euler mostrando o layout real das sete pontes*. 2025. <https://pt.wikipedia.org/wiki/Sete_pontes_de_K%C3%B6nigsberg>. Acesso em: 26 maio 2025.

3 ROSA, A. C. F. *EvacuAI-RL: Código-fonte do Servidor gRPC para Treinamento do Agente*. 2025. <https://github.com/logannas/EvacuAI-RL/blob/main/experiments/sg11-01/f51115e94ca6442aa61fc99b41856580.svg>. Acesso em: jul. 2025.

4 ROSA, A. C. F. *EvacuAI-RL: Código-fonte do Servidor gRPC para Treinamento do Agente*. 2025. <https://github.com/logannas/EvacuAI-RL/blob/main/experiments/sg11-completo/990c6a85aef3adffe85242f3e6f4d7e1.svg>. Acesso em: jul. 2025.

5 Sprinkler Brasil. *Notícias de incêndios estruturais batem recorde em 2024 e chegam ao maior volume da série histórica*. 2024. Acessado em: 3 mar. 2025. Disponível em: <https://sprinklerbrasil.org.br/imprensa/noticias-de-incendios-estruturais-batem-recorde-em-2024-e-chegam-ao-maior-volume-da-serie-historica/>.

6 ABNT/CB-024 - Comitê Brasileiro de Segurança Contra Incêndio. 2024. Disponível em: <https://www.abntcb24.com.br/index.html>.

7 SAíDAS de emergência em edifícios. 2001. Disponível em: <https://www.cnmp.mp.br/portal/images/Comissoes/DireitosFundamentais/Acessibilidade/NBR_9077_Sa%C3%ADdas_de_emerg%C3%AAncia_em_edif%C3%ADcios-2001.pdf>.

8 SUTTON, R. S.; BARTO, A. G. Chapter 1.7 - introduction, early history of reinforcement learning. In: ____. *Reinforcement learning: An introduction*. [S.l.]: The MIT Press, 2018. p. 13–22.

9 ROSA, A. C.; FALQUEIRO, M. C.; BONACIN, R.; MENDONçA, F. L. L. de; FILHO, G. P. R.; GONçALVES, V. P. Evacuai: An analysis of escape routes in indoor environments with the aid of reinforcement learning. *Sensors*, v. 23, n. 21, 2023. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/23/21/8892>.

10 SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. Second edition. Cambridge, Massachusetts, London, England: The MIT Press, 2018. This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 2.0 Generic License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/2.0/. ISBN 978-0-262-19398-6.

11 MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, Pergamon Press plc, Society for Mathematical Biology, v. 52, n. 1/2, p. 99–115, 1990. Reprinted from Bulletin of Mathematical Biophysics, Vol. 5, pp. 115-133 (1943). Disponível em: <https://www.cs.cmu.edu/~./epxing/Class/10715/reading/McCulloch.and.Pitts.pdf>.

12 GUGERTY, L. Newell and simon's logic theorist: Historical background and impact on cognitive modeling. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, v. 50, n. 9, p. 880–884, 2006. Original work published 2006.

13 RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, v. 323, p. 533–536, 1986.

14 ALPAYDıN, E. *Introduction to Machine Learning*. 4th. ed. Cambridge, MA: MIT Press, 2020. 1–A p. Section 1.1: What Is Machine Learning? ISBN 9780262358064.

15 MAHESH, B. Machine learning algorithms - a review. *International Journal of Science and Research (IJSR)*, v. 9, n. 1, 2020. ISSN 2319-7064. Licensed Under Creative Commons Attribution CC BY. Disponível em: <https://www.researchgate.net/publication/344717762_Machine_Learning_Algorithms_-A_Review>.

16 SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. Second edition. Cambridge, Massachusetts, London, England: The MIT Press, 2018. This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 2.0 Generic License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/2.0/. ISBN 978-0-262-19398-6.

17 SUTTON, R. S.; BARTO, A. G. Elements of reinforcement learning. In: *Reinforcement Learning: An Introduction*. 2nd. ed. Cambridge, Massachusetts, London, England: The MIT Press, 2018. cap. 1.3, p. 6–7.

18 SUTTON, R. S.; BARTO, A. G. Elements of reinforcement learning. In: *Reinforcement Learning: An Introduction*. 2nd. ed. Cambridge, Massachusetts, London, England: The MIT Press, 2018. cap. 3.1, p. 47–53.

19 SUTTON, R. S.; BARTO, A. G. Elements of reinforcement learning. In: *Reinforcement Learning: An Introduction*. 2nd. ed. Cambridge, Massachusetts, London, England: The MIT Press, 2018. cap. 3.2, p. 53–54.

20 SUTTON, R. S.; BARTO, A. G. Elements of reinforcement learning. In: *Reinforcement Learning: An Introduction*. 2nd. ed. Cambridge, Massachusetts, London, England: The MIT Press, 2018. cap. 2.2, p. 27–28.

21 SUTTON, R. S.; BARTO, A. G. Elements of reinforcement learning. In: *Reinforcement Learning: An Introduction*. 2nd. ed. Cambridge, Massachusetts, London, England: The MIT Press, 2018. cap. 3.5, p. 58–62.

22 SUTTON, R. S.; BARTO, A. G. Elements of reinforcement learning. In: *Reinforcement Learning: An Introduction*. 2nd. ed. Cambridge, Massachusetts, London, England: The MIT Press, 2018. cap. 3.6, p. 66–67.

23 SUTTON, R. S.; BARTO, A. G. Elements of reinforcement learning. In: *Reinforcement Learning: An Introduction*. 2nd. ed. Cambridge, Massachusetts, London, England: The MIT Press, 2018. cap. 6.5, p. 131–133.

24 FACE, H. *Deep Q-Learning Algorithm*. 2022. <https://huggingface.co/learn/deep-rl-course/unit3/deep-q-algorithm>. Acesso em: 2 jun. 2025.

25 TENSORFLOW. *Introdução ao Aprendizado por Reforço com o TensorFlow Agents*. 2023. <https://www.tensorflow.org/agents/tutorials/0_intro_rl?hl=pt-br>. Acesso em: 2 jun. 2025.

26 GOIáS, C. de Bombeiros Militar do Estado de. *Nota Técnica nº 11/2022 – Saídas de Emergência*. 2022. Acesso em: 26 abr. 2025. Disponível em: <https://www.bombeiros.go.gov.br/wp-content/uploads/2021/03/NT-11_2022-Saidas-de-Emergencia.pdf>.

27 Autodesk. *AutoCAD 2025: Software de Projeto e Desenho*. 2025. <https://www.autodesk.com/br/products/autocad/overview>. Acessado em 15 de abril de 2025.

28 COMMENTARII Academiae Scientiarum Imperialis Petropolitanae. Petropoli: Academia Scientiarum, 1736. v. 8. Acesso em: 26 maio 2025. Disponível em: <https://archive.org/details/commentariiacade08impe/page/128/mode/2up>.

29 GOMES, P. C. R. *Grafos: conceitos fundamentais, algoritmos e aplicações*. Blumenau, SC: Editora do Instituto Federal Catarinense, 2022. Capítulo 1: Conceitos fundamentais, Seção 1.2: Grafos, p. 8. Acesso em: 26 maio 2025. ISBN 978-65-88089-12-5. Disponível em: <https://editora.ifc.edu.br/wp-content/uploads/sites/39/2022/11/Grafos-versao-final.pdf>.

30 DIESTEL, R. *Graph Theory*. 2nd. ed. Berlin: Springer, 2000. Capítulo 1: The Basics. Acesso em: 26 maio 2025. Disponível em: <https://web.xidian.edu.cn/zhangxin/files/20020101_111456.pdf>.

31 MELO, G. S. d. *Introdução à Teoria dos Grafos*. 35 p. Dissertação (Mestrado Profissional em Matemática), 2014.

32 STEEN, M. van. *5.4.1 Dijkstra's algorithm*. [s.n.], 2010. Version (January 2010). In *An Introduction to Graph Theory and Complex Networks* (Vol. January 2010). Disponível em: <https://pages.di.unipi.it/ricci/book-watermarked.pdf>.

33 STEEN, M. van. *5.4.2 The Bellman-Ford algorithm*. [s.n.], 2010. Version (January 2010). In *An Introduction to Graph Theory and Complex Networks* (Vol. January 2010). Disponível em: <https://pages.di.unipi.it/ricci/book-watermarked.pdf>.

34 FOEAD, D.; GHIFARI, A.; KUSUMA, M. B.; HANAFIAH, N.; GUNAWAN, E. A systematic literature review of a* pathfinding. *Procedia Computer Science*, v. 179, p. 507–514, 2021. ISSN 1877-0509. 5th International Conference on Computer Science and Computational Intelligence 2020. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050921000399>.

35 MORINA, V.; RAFUNA, R. A comparative analysis of pathfinding algorithms in npc movement systems for computer games. In: . [S.l.: s.n.], 2023.

36 PERMANA, S.; BINTORO, K.; ARIFITAMA, B.; SYAHPUTRA, A. Comparative analysis of pathfinding algorithms a *, dijkstra, and bfs on maze runner game. *IJISTECH (International Journal Of Information System Technology)*, v. 1, p. 1, 05 2018.

37 RAFIQ, A.; KADIR, T. A. A.; IHSAN, S. N. Pathfinding algorithms in game development. *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, v. 769, n. 1, p. 012021, feb 2020. Disponível em: <https://dx.doi.org/10.1088/1757-899X/769/1/012021>.

38 ZHU, Y.; ZHANG, G.; CHU, R.; XIAO, H.; YANG, Y.; WU, X. Research on escape route planning analysis in forest fire scenes based on the improved a* algorithm. *Ecological Indicators*, v. 166, p. 112355, 2024. ISSN 1470-160X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1470160X24008124>.

39 HABOURY, N.; KORDZANGANEH, M.; SCHMITT, S.; JOSHI, A.; TOKAREV, I.; ABDALLAH, L.; KURKIN, A.; KYRIACOU, B.; MELNIKOV, A. *A supervised hybrid quantum machine learning solution to the emergency escape routing problem*. 2023. Disponível em: <https://arxiv.org/abs/2307.15682>.

40 SRIKANTH, H. *Reinforcement Learning Based Escape Route Generation in Low Visibility Environments*. 2024. Disponível em: <https://arxiv.org/abs/2406.07568>.

41 XU, L.; HUANG, K.; LIU, J.; LI, D.; CHEN, Y. F. Intelligent planning of fire evacuation routes using an improved ant colony optimization algorithm. *Journal of Building Engineering*, v. 61, p. 105208, 2022. ISSN 2352-7102. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2352710222012141>.

42 AGNIHOTRI, A.; FATHI-KAZEROONI, S.; KAYMAK, Y.; ROJAS-CESSA, R. Evacuating routes in indoor-fire scenarios with selection of safe exits on known and unknown buildings using machine learning. In: *2018 IEEE 39th Sarnoff Symposium*. [S.l.: s.n.], 2018. p. 1–6.

43 SHARMA, J.; ANDERSEN, P.-A.; GRANMO, O.-C.; GOODWIN, M. Deep q-learning with q-matrix transfer learning for novel fire evacuation environment. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, v. 51, n. 12, p. 7363–7381, 2021.

44 HAN, L.; GUO, H.; ZHANG, H.; KONG, Q.; ZHANG, A.; GONG, C. An efficient staged evacuation planning algorithm applied to multi-exit buildings. *ISPRS International Journal of Geo-Information*, v. 9, n. 1, p. 46, 2020. Disponível em: <https://doi.org/10.3390/ijgi9010046>.

45 HöNIG, W.; LI, J.; KOENIG, S. *Overview of Multi-Agent Path Finding (MAPF)*. 2020. Model AI 2020 Assignments: A Project on Multi-Agent Path Finding (MAPF). Disponível em: <https://ics.uci.edu/~svenk/project-p/material/overview.pdf>.

46 WEERD, R. Van der. *Predicting plausible escape routes using reinforcement learning and graph representations*. Tese (Doutorado), 07 2022.

47 YAMADA, M.; WANG, X.; YAMASAKI, T. Graph structure extraction from floor plan images and its application to similar property retrieval. In: *2021 IEEE International Conference on Consumer Electronics (ICCE)*. [S.l.: s.n.], 2021. p. 1–5.

48 LIFULL HOME'S Dataset. [Online]. Available: https://www.nii.ac.jp/dsc/idr/en/. Disponível em: <https://www.nii.ac.jp/dsc/idr/en/>.

49 KALERVO, A.; YLIOINAS, J.; HÄIKIÖ, M.; KARHU, A.; KANNALA, J. Cubicasa5k: A dataset and an improved multi-task model for floorplan image analysis. *CoRR*, abs/1904.01920, 2019. Disponível em: <http://arxiv.org/abs/1904.01920>.

50 LU, Y.; TIAN, R.; LI, A.; WANG, X.; LóPEZ, J. L. G. del Castillo y. Cubigraph5k: Organizational graph generation for structured architectural floor plan dataset. In: *Proceedings of CAADRIA 2021*. [S.l.: s.n.], 2021.

51 WANG, X.-Y.; YANG, Y.; ZHANG, K. Customization and generation of floor plans based on graph transformations. *Automation in Construction*, v. 94, p. 405–416, 2018. ISSN 0926-5805. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0926580517311251>.

52 PIZARRO, P. N.; HITSCHFELD, N.; SIPIRAN, I.; SAAVEDRA, J. M. Automatic floor plan analysis and recognition. *Automation in Construction*, v. 140, p. 104348, 2022. ISSN 0926-5805. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0926580522002217>.

53 Python Software Foundation. *Python Programming Language*. 2025. Acessado em: 3 mar. 2025. Disponível em: <https://www.python.org/>.

54 PyTorch Team. *PyTorch: An Open Source Machine Learning Framework*. 2025. Acessado em: 3 mar. 2025. Disponível em: <https://pytorch.org/>.

55 Anna Carolina Logan. *Front Graph Path: Código-fonte da Interface Front-end*. 2025. <https://github.com/logannas/front-graphPath>. Acesso em: 17 jul. 2025.

56 Anna Carolina Ferreira Rosa. *EvacuAI-API: Código-fonte da API Rest*. 2025. <https://github.com/logannas/EvacuAI-API>. Acesso em: 17 jul. 2025.

57 Anna Carolina Ferreira Rosa. *EvacuAI-RL: Código-fonte do Servidor gRPC para Treinamento do Agente*. 2025. <https://github.com/logannas/EvacuAI-RL>. Acesso em: 17 jul. 2025.

58 Docker. *Docker: Empowering App Development for Developers*. 2025. <https://www.docker.com/>. Acessado em: 19 jul. 2025.

59 Evan You and Vue.js Contributors. *Vue.js - The Progressive JavaScript Framework*. 2024. Acesso em: 21 maio 2025. Disponível em: <https://vuejs.org/>.

60 Anna Carolina Ferreira Rosa. *Front Graph Path: Interface Web para Treinamento e Inferência de Rotas em Grafos*. 2025. Acesso em: 17 jul. 2025. Disponível em: <https://front-graph-path.vercel.app/>.

61 DASH14. *v-network-graph: Interactive Network Graph Visualization Component for Vue 3*. 2023. Acesso em: 21 maio 2025. Disponível em: <https://dash14.github.io/v-network-graph/>.

62 RAMíREZ, S. *FastAPI*. 2018. <https://fastapi.tiangolo.com/>. Acesso em: 22 maio 2025.

63 MongoDB, Inc. *MongoDB – The Developer Data Platform*. 2009. <https://www.mongodb.com/>. Acesso em: 22 maio 2025.

64 MinIO, Inc. *MinIO – High Performance Object Storage*. 2015. <https://min.io/>. Acesso em: 22 maio 2025.

65 The gRPC Authors. *gRPC*. 2024. <https://grpc.io/>. Available at: <https://grpc.io/>.

66 ROSA, A. C. *Google Colaboratory - GraphDQN*. 2023. <https://colab.research.google.com/drive/1OtWE2lqF_oyCO0o_PG_QCot2C52RfI4J#scrollTo=ZxeMudpTKxBg>. Acesso em: jun. 2025.

I.1 CAMINHOS CALCULADOS ATÉ AS SAÍDAS SEM OBSTÁCULOS

Nó Inicial	Nó Final	Caminho
2	0	$2 \rightarrow 0$
3	0	$3 \rightarrow 2 \rightarrow 0$
4	0	$4 \rightarrow 10 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
5	0	$5 \rightarrow 10 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
6	0	$6 \rightarrow 10 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
7	0	$7 \rightarrow 10 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
8	0	$8 \rightarrow 10 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
9	0	$9 \rightarrow 10 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
10	0	$10 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
11	0	$11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
12	0	$12 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
13	0	$13 \rightarrow 12 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
14	0	$14 \rightarrow 19 \rightarrow 25 \rightarrow 0$
15	0	$15 \rightarrow 18 \rightarrow 14 \rightarrow 19 \rightarrow 25 \rightarrow 0$
16	0	$16 \rightarrow 18 \rightarrow 14 \rightarrow 19 \rightarrow 25 \rightarrow 0$
17	0	$17 \rightarrow 18 \rightarrow 14 \rightarrow 19 \rightarrow 25 \rightarrow 0$
18	0	$18 \rightarrow 14 \rightarrow 19 \rightarrow 25 \rightarrow 0$
19	0	$19 \rightarrow 25 \rightarrow 0$
20	0	$20 \rightarrow 19 \rightarrow 25 \rightarrow 0$
21	0	$21 \rightarrow 19 \rightarrow 25 \rightarrow 0$
22	0	$22 \rightarrow 19 \rightarrow 25 \rightarrow 0$
23	0	$23 \rightarrow 22 \rightarrow 19 \rightarrow 25 \rightarrow 0$
24	0	$24 \rightarrow 23 \rightarrow 22 \rightarrow 19 \rightarrow 25 \rightarrow 0$
25	0	$25 \rightarrow 0$
26	0	$26 \rightarrow 25 \rightarrow 0$
27	0	$27 \rightarrow 25 \rightarrow 0$
28	0	$28 \rightarrow 0$
29	0	$29 \rightarrow 0$
30	0	$30 \rightarrow 25 \rightarrow 0$
31	0	$31 \rightarrow 32 \rightarrow 25 \rightarrow 0$
32	0	$32 \rightarrow 25 \rightarrow 0$
33	0	$33 \rightarrow 32 \rightarrow 25 \rightarrow 0$
34	0	$34 \rightarrow 32 \rightarrow 25 \rightarrow 0$
35	0	$35 \rightarrow 32 \rightarrow 25 \rightarrow 0$
36	0	$36 \rightarrow 35 \rightarrow 32 \rightarrow 25 \rightarrow 0$

Nó Inicial	Nó Final	Caminho
37	0	$37 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$
38	0	$38 \rightarrow 32 \rightarrow 25 \rightarrow 0$
39	0	$39 \rightarrow 38 \rightarrow 32 \rightarrow 25 \rightarrow 0$
40	0	$40 \rightarrow 41 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$
41	0	$41 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$
42	0	$42 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$
43	0	$43 \rightarrow 42 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$
44	1	$44 \rightarrow 45 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$
45	1	$45 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$
46	1	$46 \rightarrow 45 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$
47	1	$47 \rightarrow 1$
48	1	$48 \rightarrow 1$
49	1	$49 \rightarrow 48 \rightarrow 1$
50	1	$50 \rightarrow 49 \rightarrow 48 \rightarrow 1$
51	1	$51 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$
52	1	$52 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$
53	1	$53 \rightarrow 54 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$
54	1	$54 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$
55	1	$55 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$
56	1	$56 \rightarrow 58 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$
57	1	$57 \rightarrow 56 \rightarrow 58 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$
58	1	$58 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$
59	1	$59 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$
60	1	$60 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$
61	1	$61 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$
62	1	$62 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$
63	1	$63 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 48 \rightarrow 1$

Tabela 1: Caminhos do Experimento 1

Nó Inicial	Nó Final	Caminho
2	0	$2 \rightarrow 0$
3	0	$3 \rightarrow 2 \rightarrow 0$
4	0	$4 \rightarrow 10 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
5	0	$5 \rightarrow 10 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
6	0	$6 \rightarrow 10 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
7	0	$7 \rightarrow 10 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
8	0	$8 \rightarrow 10 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
9	0	$9 \rightarrow 10 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
10	0	$10 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
11	0	$11 \rightarrow 3 \rightarrow 2 \rightarrow 0$

Nó Inicial	Nó Final	Caminho
12	0	$12 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
13	0	$13 \rightarrow 12 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 0$
14	0	$14 \rightarrow 19 \rightarrow 25 \rightarrow 0$
15	0	$15 \rightarrow 18 \rightarrow 14 \rightarrow 19 \rightarrow 25 \rightarrow 0$
16	0	$16 \rightarrow 18 \rightarrow 14 \rightarrow 19 \rightarrow 25 \rightarrow 0$
17	0	$17 \rightarrow 18 \rightarrow 14 \rightarrow 19 \rightarrow 25 \rightarrow 0$
18	0	$18 \rightarrow 14 \rightarrow 19 \rightarrow 25 \rightarrow 0$
19	0	$19 \rightarrow 25 \rightarrow 0$
20	0	$20 \rightarrow 19 \rightarrow 25 \rightarrow 0$
21	0	$21 \rightarrow 19 \rightarrow 25 \rightarrow 0$
22	0	$22 \rightarrow 19 \rightarrow 25 \rightarrow 0$
23	0	$23 \rightarrow 22 \rightarrow 19 \rightarrow 25 \rightarrow 0$
24	0	$24 \rightarrow 23 \rightarrow 22 \rightarrow 19 \rightarrow 25 \rightarrow 0$
25	0	$25 \rightarrow 0$
26	0	$26 \rightarrow 25 \rightarrow 0$
27	0	$27 \rightarrow 25 \rightarrow 0$
28	0	$28 \rightarrow 0$
29	0	$29 \rightarrow 0$
30	0	$30 \rightarrow 25 \rightarrow 0$
31	0	$31 \rightarrow 32 \rightarrow 25 \rightarrow 0$
32	0	$32 \rightarrow 25 \rightarrow 0$
33	0	$33 \rightarrow 32 \rightarrow 25 \rightarrow 0$
34	0	$34 \rightarrow 32 \rightarrow 25 \rightarrow 0$
35	0	$35 \rightarrow 32 \rightarrow 25 \rightarrow 0$
36	0	$36 \rightarrow 35 \rightarrow 32 \rightarrow 25 \rightarrow 0$
37	0	$37 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$
38	0	$38 \rightarrow 32 \rightarrow 25 \rightarrow 0$
39	0	$39 \rightarrow 38 \rightarrow 32 \rightarrow 25 \rightarrow 0$
40	1	$40 \rightarrow 41 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
41	1	$41 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
42	0	$42 \rightarrow 34 \rightarrow 32 \rightarrow 25 \rightarrow 0$
43	1	$43 \rightarrow 45 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
44	1	$44 \rightarrow 45 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
45	1	$45 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
46	1	$46 \rightarrow 45 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
47	1	$47 \rightarrow 1$
48	1	$48 \rightarrow 1$
49	1	$49 \rightarrow 47 \rightarrow 1$
50	1	$50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
51	1	$51 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
52	1	$52 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$

Nó Inicial	Nó Final	Caminho
53	1	$53 \rightarrow 54 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
54	1	$54 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
55	1	$55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
56	1	$56 \rightarrow 58 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
57	1	$57 \rightarrow 58 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
58	1	$58 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
59	1	$59 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
60	1	$60 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
61	1	$61 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
62	1	$62 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
63	1	$63 \rightarrow 55 \rightarrow 50 \rightarrow 49 \rightarrow 47 \rightarrow 1$
64	0	$64 \rightarrow 0$
65	0	$65 \rightarrow 77 \rightarrow 80 \rightarrow 64 \rightarrow 0$
66	0	$66 \rightarrow 67 \rightarrow 70 \rightarrow 77 \rightarrow 80 \rightarrow 64 \rightarrow 0$
67	0	$67 \rightarrow 70 \rightarrow 77 \rightarrow 80 \rightarrow 64 \rightarrow 0$
68	0	$68 \rightarrow 67 \rightarrow 70 \rightarrow 77 \rightarrow 80 \rightarrow 64 \rightarrow 0$
69	0	$69 \rightarrow 67 \rightarrow 70 \rightarrow 77 \rightarrow 80 \rightarrow 64 \rightarrow 0$
70	0	$70 \rightarrow 77 \rightarrow 80 \rightarrow 64 \rightarrow 0$
71	0	$71 \rightarrow 70 \rightarrow 77 \rightarrow 80 \rightarrow 64 \rightarrow 0$
72	0	$72 \rightarrow 73 \rightarrow 70 \rightarrow 77 \rightarrow 80 \rightarrow 64 \rightarrow 0$
73	0	$73 \rightarrow 70 \rightarrow 77 \rightarrow 80 \rightarrow 64 \rightarrow 0$
74	0	$74 \rightarrow 73 \rightarrow 70 \rightarrow 77 \rightarrow 80 \rightarrow 64 \rightarrow 0$
75	0	$75 \rightarrow 74 \rightarrow 73 \rightarrow 70 \rightarrow 77 \rightarrow 80 \rightarrow 64 \rightarrow 0$
76	0	$76 \rightarrow 77 \rightarrow 80 \rightarrow 64 \rightarrow 0$
77	0	$77 \rightarrow 80 \rightarrow 64 \rightarrow 0$
78	0	$78 \rightarrow 79 \rightarrow 82 \rightarrow 80 \rightarrow 64 \rightarrow 0$
79	0	$79 \rightarrow 82 \rightarrow 80 \rightarrow 64 \rightarrow 0$
80	0	$80 \rightarrow 64 \rightarrow 0$
81	0	$81 \rightarrow 82 \rightarrow 80 \rightarrow 64 \rightarrow 0$
82	0	$82 \rightarrow 80 \rightarrow 64 \rightarrow 0$
83	0	$83 \rightarrow 80 \rightarrow 64 \rightarrow 0$
84	0	$84 \rightarrow 83 \rightarrow 80 \rightarrow 64 \rightarrow 0$
85	0	$85 \rightarrow 84 \rightarrow 83 \rightarrow 80 \rightarrow 64 \rightarrow 0$
86	0	$86 \rightarrow 83 \rightarrow 80 \rightarrow 64 \rightarrow 0$
87	0	$87 \rightarrow 64 \rightarrow 0$
88	0	$88 \rightarrow 64 \rightarrow 0$
89	0	$89 \rightarrow 83 \rightarrow 80 \rightarrow 64 \rightarrow 0$
90	0	$90 \rightarrow 89 \rightarrow 83 \rightarrow 80 \rightarrow 64 \rightarrow 0$
91	0	$91 \rightarrow 83 \rightarrow 80 \rightarrow 64 \rightarrow 0$
92	0	$92 \rightarrow 83 \rightarrow 80 \rightarrow 64 \rightarrow 0$
93	0	$93 \rightarrow 83 \rightarrow 80 \rightarrow 64 \rightarrow 0$

Nó Inicial	Nó Final	Caminho
94	0	94 → 91 → 83 → 80 → 64 → 0
95	0	95 → 91 → 83 → 80 → 64 → 0
96	0	96 → 91 → 83 → 80 → 64 → 0
97	0	97 → 96 → 91 → 83 → 80 → 64 → 0
98	1	98 → 41 → 50 → 49 → 47 → 1
99	0	99 → 96 → 91 → 83 → 80 → 64 → 0
100	0	100 → 96 → 91 → 83 → 80 → 64 → 0
101	0	101 → 96 → 91 → 83 → 80 → 64 → 0
102	0	102 → 101 → 96 → 91 → 83 → 80 → 64 → 0
103	0	103 → 104 → 101 → 96 → 91 → 83 → 80 → 64 → 0
104	0	104 → 101 → 96 → 91 → 83 → 80 → 64 → 0
105	0	105 → 104 → 101 → 96 → 91 → 83 → 80 → 64 → 0
106	0	106 → 104 → 101 → 96 → 91 → 83 → 80 → 64 → 0
107	0	107 → 108 → 109 → 101 → 96 → 91 → 83 → 80 → 64 → 0
108	0	108 → 109 → 101 → 96 → 91 → 83 → 80 → 64 → 0
109	0	109 → 101 → 96 → 91 → 83 → 80 → 64 → 0
110	0	110 → 109 → 101 → 96 → 91 → 83 → 80 → 64 → 0
111	0	111 → 109 → 101 → 96 → 91 → 83 → 80 → 64 → 0
112	0	112 → 109 → 101 → 96 → 91 → 83 → 80 → 64 → 0
113	0	113 → 112 → 109 → 101 → 96 → 91 → 83 → 80 → 64 → 0
114	0	114 → 109 → 101 → 96 → 91 → 83 → 80 → 64 → 0
115	0	115 → 109 → 101 → 96 → 91 → 83 → 80 → 64 → 0
116	0	116 → 109 → 101 → 96 → 91 → 83 → 80 → 64 → 0
117	0	117 → 109 → 101 → 96 → 91 → 83 → 80 → 64 → 0
118	0	118 → 117 → 109 → 101 → 96 → 91 → 83 → 80 → 64 → 0
119	0	119 → 117 → 109 → 101 → 96 → 91 → 83 → 80 → 64 → 0

Tabela 2: Caminhos do Experimento 2

I.2 CÓDIGOS FONTE DA API

I.2.1 Rota `/create`

```
@router.post("")
async def add_data(
    Project: Project,
    minio_client=Depends(get_minio_client),
    mongodb: tuple = Depends(get_mongodb)
):
    _, _, collection = mongodb
```

```
    image_data = re.sub("^", "", Project.image)
    im_bytes = base64.b64decode(image_data)

    nodes = json.loads(Project.nodes)
    edges = json.loads(Project.edges)
    layouts = json.loads(Project.layouts)

    graph_dict = {
        "nodes": nodes,
        "edges": edges,
        "layouts": layouts
    }

    graph_dict = calculate_distance(graph_dict)

    random_str = "".join(random.choice(string.ascii_uppercase + string.digits)
        for _ in range(6))
    str_final = Project.project_name + random_str
    file_id = hashlib.md5(str_final.encode("utf-8")).hexdigest()

    await upload_minio(minio_client, f"images/{file_id}.svg", im_bytes,
        content_type="image/svg+xml")

    json_data = json.dumps(graph_dict)
    json_bytes = json_data.encode('utf-8')
    await upload_minio(minio_client, f"graph/{file_id}.json", json_bytes,
        content_type="application/json")

    configs_project = {
        "project_id": file_id,
        "project_name": Project.project_name,
        "image_s3_uri": f"images/{file_id}.svg",
        "graph_s3_uri": f"graph/{file_id}.json",
        "createdTime": datetime.datetime.now().isoformat(),
    }

    collection.insert_one(configs_project)

    return DatasReturn(project_id=file_id)
```

Listing 1: Rota de criação de projeto na API FastAPI

I.2.2 Rota `/train`

```
@router.post("")
async def train_project(
    info: InfoDatas,
    hyperparameters: Hyperparameters,
    stub=Depends(get_grpc_stub)
```

```
):
    hyp_dict = hyperparameters.dict(exclude_unset=True)
    hyp_params = evacuai_rl_pb2.HypParams(**hyp_dict)

    train_request = evacuai_rl_pb2.TrainRequest(
        hyperparameters=hyp_params,
        project_id=info.project,
    )

    response = stub.TrainModel(train_request)
    return {"model_id": response.model_id}
```

Listing 2: Rota para iniciar treinamento de um projeto

I.2.3 Rota `/inference`

```
@router.get("/")
async def run_inference(
    project_id: str,
    version: str,
    state: Optional[int] = None,
    fire_nodes: List[int] = Query(default=[]),
    congestion_nodes: List[int] = Query(default=[]),
    stub=Depends(get_grpc_stub)
):
    inference_request = evacuai_rl_pb2.InferenceRequest(
        project_id=project_id,
        version=version or "",
        previous_state=state if state is not None else 0,
        fire_nodes=fire_nodes,
        agents_positions = congestion_nodes
    )

    response = stub.Inference(inference_request)

    response_dict = MessageToDict(
        response,
        preserving_proto_field_name=True,
    )
    for prediction in response_dict.get("predictions", []):
        if "last_node" not in prediction:
            path = prediction.get("path", [])
            prediction["last_node"] = path[-1] if path else None

    return {"predictions": response_dict["predictions"]}
```

Listing 3: Rota de inferência via gRPC

I.2.4 Rota `/projects`

```python
@router.get("")
async def list_projects(
    minio_client=Depends(get_minio_client),
    mongodb: tuple = Depends(get_mongodb)
):
    _, _, collection = mongodb
    projects = collection.find()

    project_list = []

    for project in projects:
        project_id = project["project_id"]
        project_name = project["project_name"]
        image_path = project.get("image_s3_uri")
        graph_path = project.get("graph_s3_uri")

        image_bytes = await download_minio(minio_client, image_path)
        image_base64 = base64.b64encode(image_bytes).decode("utf-8")

        graph_bytes = await download_minio(minio_client, graph_path)
        graph_str = graph_bytes.decode("utf-8")
        graph = json.loads(graph_str)

        project_list.append({
            "project_id": project_id,
            "project_name": project_name,
            "graph_nodes": graph['nodes'],
            "image_base64": f"{image_base64}"
        })

    return JSONResponse(content=project_list)
```

Listing 4: Rota para listagem de projetos

I.2.5 Rota `/train_ids/{project_id}`

```python
@router.get("/{project_id}")
async def list_train_ids(
    project_id: str,
    minio_client: Minio = Depends(get_minio_client),
):
    prefix = f"experiments/{project_id}/"
    objects = await list_files(minio_client, prefix)

    train_ids = set()

```

```
    for obj in objects:
        parts = obj.object_name.split("/")
        if len(parts) >= 3:
            train_id = parts[2]
            train_ids.add(train_id)

    return JSONResponse(content=sorted(list(train_ids)))
```

Listing 5: Rota para listagem de versões de treinamento

I.3 CÓDIGOS DA CAMADA DE INTELIGÊNCIA

I.3.1 Servidor Grpc

```
syntax = "proto3";

package rl;

service ReinforcementLearning {
    rpc TrainModel (TrainRequest) returns (TrainResponse);
    rpc Inference (InferenceRequest) returns (InferenceResponse);
    rpc GetRewardPath (GetRewardPathRequest) returns (GetRewardPathResponse);
}

message TrainRequest {
    string project_id = 1;
    optional string transfer_learning_version = 2;
    HypParams hyperparameters = 3;
}

message TrainResponse {
    string model_id = 1;
}

message InferenceRequest {
    string project_id = 1;
    string version = 2;
    int32 previous_state = 3;
    repeated int32 fire_nodes = 4;
    repeated int32 agents_positions = 5;
}

message GetRewardPathRequest {
    string project_id = 1;
    string version = 2;
    repeated int32 path = 3;
    repeated int32 fire_nodes = 4;
    repeated int32 agents_positions = 5;
```

```
}

message Path {
    int32 init_node = 1;
    int32 last_node = 2;
    repeated int32 path = 3;
}

message InferenceResponse {
    repeated Path predictions = 1;
}

message GetRewardPathResponse {
    float reward = 1;
}

message HypParams {
    optional float beta = 1;
    optional float lr = 2;
    optional int32 batch_size = 3;
    optional int32 buffer_size = 4;
    optional int32 episodes = 5;
    optional float gamma = 6;
    optional float epsilon = 7;
    optional int32 congestion_threshold = 8;
    optional int32 num_virtual_agents = 9;
    optional float reward_exit = 10;
    optional float reward_fire = 11;
    optional float reward_invalid = 12;
    optional float reward_valid = 13;
    optional float reward_congestion = 14;
}
```

Listing 6: Definição do serviço gRPC para aprendizado por reforço

I.3.2 Rede Neural para Seleção de Ações

```
import torch as T
import torch.nn as nn
import torch.nn.functional as F

class Network(nn.Module):
    def __init__(self, obs_size: int, n_actions: int) -> None:
        super(Network, self).__init__()

        self.fc1 = nn.Linear(obs_size, 512)
        self.ln1 = nn.LayerNorm(512)
        self.dropout1 = nn.Dropout(p=0.2)

```

```
        self.fc2 = nn.Linear(512, 256)
        self.ln2 = nn.LayerNorm(256)
        self.dropout2 = nn.Dropout(p=0.2)

        self.fc3 = nn.Linear(256, 128)
        self.ln3 = nn.LayerNorm(128)
        self.dropout3 = nn.Dropout(p=0.2)

        self.fc4 = nn.Linear(128, 64)
        self.ln4 = nn.LayerNorm(64)
        self.dropout4 = nn.Dropout(p=0.1)

        self.fc5 = nn.Linear(64, 32)
        self.fc6 = nn.Linear(32, n_actions)

        self.device = T.device("cuda" if T.cuda.is_available() else "cpu")
        self.to(self.device)

        self._init_weights()

    def _init_weights(self):
        for layer in [self.fc1, self.fc2, self.fc3, self.fc4, self.fc5, self.fc6]:
            nn.init.kaiming_uniform_(layer.weight, nonlinearity="leaky_relu")
            nn.init.zeros_(layer.bias)

    def forward(self, x: T.Tensor) -> T.Tensor:
        x = F.leaky_relu(self.ln1(self.fc1(x)), negative_slope=0.01)
        x = self.dropout1(x)
        x = F.leaky_relu(self.ln2(self.fc2(x)), negative_slope=0.01)
        x = self.dropout2(x)
        x = F.leaky_relu(self.ln3(self.fc3(x)), negative_slope=0.01)
        x = self.dropout3(x)
        x = F.leaky_relu(self.ln4(self.fc4(x)), negative_slope=0.01)
        x = self.dropout4(x)
        x = F.leaky_relu(self.fc5(x), negative_slope=0.01)
        x = self.fc6(x)
        return x

    def act(self, state: T.Tensor) -> int:
        state = state.to(self.device)
        actions = self.forward(state)
        action = T.argmax(actions).item()
        return action
```

Listing 7: Implementação da rede neural utilizada na política de decisão