

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Sistema de Apoio à Decisão para a Evacuação de Emergência de Edifícios

Tiago João Marabuto da Silva Pereira

Mestrado em Engenharia Eletrotécnica e de Computadores

Orientador: Maria Antónia da Silva Lopes e Carravilla

29 de Julho de 2022

Resumo

Num mundo em que os edifícios continuam a ficar cada vez maiores e são capazes de albergar a população equivalente a aldeias e vilas, a forma como lidamos com as evacuações de emergência neles pode ter um enorme impacto quando se trata da vida dos cidadãos no seu interior. Durante uma evacuação, escolher corretamente o melhor caminho para chegar à segurança pode ser uma tarefa difícil. Embora possamos conhecer a planta do edifício, o perigo pode alastrar e dificultar o nosso caminho para a segurança. As instruções tradicionais de evacuação através de sinalética e plantas estáticas não conseguem lidar com a imprevisibilidade natural destas emergências e podem, portanto, dar direções incorretas aos ocupantes levando-os, em última instância, a situações perigosas enquanto deixam o edifício.

Esta dissertação, elaborada no âmbito do segundo ano do Mestrado em Engenharia Eletrotécnica e de Computadores, visa o desenvolvimento de um Sistema de Apoio à Decisão para a Evacuação de Emergência de Edifícios (SAD3E).

O SAD3E é um sistema capaz de agir em situações de incêndio de edifícios e direccionar os seus ocupantes para a saída mais próxima e segura. Este sistema, recebe informações de uma central de deteção de incêndio sobre o estado dos detetores de incêndio do edifício e, tendo em conta a sua localização, direcciona os ocupantes através de sinalética dinâmica de emergência.

Neste documento, primeiramente é feito um estudo sobre as diferentes abordagens ao problema da evacuação de edifícios, bem como um levantamento das tecnologias e conceitos necessários para o desenvolvimento do projeto.

De seguida, apresenta-se a implementação da solução para o problema, explicando cada um dos constituintes do sistema e o seu modo de funcionamento.

Por último, apresentam-se os testes feitos ao SAD3E e os resultados obtidos.

Palavras-chave: Evacuação, Sistema de Apoio à Decisão, Emergência.

Abstract

In a world where buildings keep getting bigger and bigger and are capable of hosting at a time the population equivalent to villages, the way we deal with emergency evacuations in them could have a huge impact when it comes to the lives of the citizens inside. During an evacuation, correctly choosing the best path to get to safety can be a difficult task. Even though we may know the building plan, the hazard can spread and hamper our path to safety. The traditional evacuation instructions through static signs and plans can't cope with the natural unpredictability of these emergencies and can, therefore, give improper directions to the occupants leading evacuees to dangerous situations while leaving the building. This dissertation aims on developing a Decision Support System (DSS) that comes into play whenever an emergency occurs. After collecting data from sensors, fire detectors, video analysis, and knowing, beforehand, the building plan, the system should be able to plan optimal escape routes and communicate them to the evacuees, as well as act on fire contention and extinction devices.

This dissertation, developed in the second year of the Master in Electrical and Computer Engineering, aims to develop a Decision Support System for Emergency Evacuation of Buildings (SAD3E).

The SAD3E is a system capable of acting in situations of building fire and directing its occupants to the nearest and safest exit. This system receives information from a fire detection station about the status of the building's fire detectors and, taking into account their location, directs the occupants through dynamic emergency signaling.

In this document, first a study is made about the different approaches to the problem of building evacuation, as well as a survey of the technologies and concepts needed for the development of the project.

Next, the implementation of the solution to the problem is presented, explaining each of the constituents of the system and how it works.

Finally, the tests made to SAD3E and the results obtained are presented.

Keywords: Evacuation, Decision Support System, Emergency.

Agradecimentos

Primeiramente, quero agradecer à Professora Maria Antónia Carravilla por todo o apoio, motivação e ajuda na elaboração desta dissertação.

À NIBBLE® e em especial ao João o meu muito obrigado pela oportunidade e acompanhamento ao longo de todo o projeto.

À minha família, por serem um suporte fundamental no meu desenvolvimento e sem vocês não seria o homem que sou hoje.

Aos meus amigos, da faculdade, da escola, dos escuteiros, da escalada, do catan, do canyoning, o meu muito obrigado por todo o apoio, gargalhadas e momentos passados.

Por último, à Inês, paciência em pessoa, o meu muito obrigado por estares sempre lá a orientar-me quando, mesmo sem saber, necessito.

A todos, o meu muito obrigado! Sem vocês nada disto seria possível e sem vocês nada disto teria sido tão alegre e risonho.

Tiago Marabuto

“Rockets are cool. There’s no getting around that”

Elon Musk

Conteúdo

Resumo	i
Abstract	iii
Abreviaturas	xv
1 Introdução	1
1.1 Contexto e Motivação	1
1.2 Objetivos	2
1.3 A NIBBLE®	2
1.4 Fases de desenvolvimento	2
1.5 Estrutura da Dissertação	3
2 Revisão da Literatura	5
2.1 Abordagens ao problema de evacuação	5
2.1.1 Abordagem de Filippoupolitis e Gelenbe	6
2.1.2 Abordagem de Fang <i>et al.</i>	7
2.1.3 Abordagem de Pursals e Garzon	8
2.1.4 Abordagem de Cuesta <i>et al.</i>	9
2.1.5 Análise comparativa dos algoritmos apresentados	10
2.2 Grafos	11
2.3 Algoritmos de procura de caminho mínimo	11
2.3.1 Dijkstra	12
2.3.2 Bellman-Ford	13
2.3.3 Floyd-Warshall	14
2.3.4 Comparação dos Algoritmos apresentados	15
2.4 Comparação de formatos de ficheiro para armazenamento de dados	16
2.5 Comparação de <i>toolkits</i> para desenvolvimento de GUIs	17
2.6 MODBUS TCP	18
3 Análise do problema	19
3.1 Cenário de atuação	19
3.2 Estudo de Mercado	20
3.2.1 Tipos de sistemas de iluminação de emergência	20
3.2.2 Eaton - Evacuação de emergência adaptável	21
3.2.3 Implementação futura por parte da NIBBLE®	22
3.3 Requisitos do sistema	22

4	Implementação da Solução	25
4.1	Arquitetura do Sistema	25
4.1.1	Objeto Node	25
4.1.2	Constituintes do sistema	26
4.2	Implementação do algoritmo	27
4.2.1	Construção do grafo	28
4.2.2	Cálculo do caminho mínimo até à saída	29
4.3	Comunicação com a central de deteção de incêndio	31
4.4	Implementação da Interface Gráfica	32
4.4.1	Estrutura de aplicações de DPG	33
4.4.2	Estrutura e Funcionamento da Interface Gráfica	34
4.4.3	Janela inicial	34
4.4.4	Janela de Criação de Projeto	36
4.4.5	Janela de Visualização	38
5	Testes e funcionamento do sistema	43
5.1	Incêndio introduzido pelo utilizador	43
5.2	Resposta a estímulo da central de deteção de incêndio	44
5.3	Resposta do sistema a múltiplos incêndios	46
6	Conclusões e Trabalhos Futuros	49
6.1	Satisfação dos Objetivos	49
6.2	Trabalhos Futuros	50
A	Tabelas de Registos Modbus da central NURIA	51
	Referências	55

Lista de Figuras

2.1	Exemplo de grafo correspondente a um andar de um edifício.	6
2.2	Modelação da incorporação a uma via de evacuação.	9
2.3	(Esq.) Grafo não direcional (Dir.) Grafo direcional	11
3.1	Exemplo de funcionamento de sistema de luminárias do tipo estático [1].	20
3.2	Exemplo de funcionamento de sistema de luminárias do tipo dinâmico [1].	21
3.3	Exemplo de funcionamento de sistema de luminárias do tipo adaptável [1].	21
4.1	Objeto Node	26
4.2	Esquemático ds constituintes do SAD3E.	27
4.3	Exemplo de grafo guardado no formato .json.	28
4.4	(Esq.) Grafo sem incêndio (Dir.) Grafo com incêndio e zona de afetação.	31
4.5	Janela inicial: (Esq) em Windows (Dir) MacOS.	33
4.6	Itens da biblioteca DPG.	34
4.7	Janela inicial da GUI do SAD3E.	35
4.8	Janela de Criação de Projeto do SAD3E com planta do piso 0 do Bloco B do edifício da FEUP.	36
4.9	Exemplo de tentativa de criação de um nó com um nome já existente.	37
4.10	Exemplo de criação de arestas entre nós	37
4.11	(Em cima) Exemplo de grafo (Em baixo) Output da função de impressão do grafo no terminal.	38
4.12	Janela de Visualização.	39
4.13	Janela de Visualização com grafo desenhado.	39
4.14	Janela de Visualização após seleção das saídas do edifício (a vermelho)	40
4.15	Exemplo das luminárias dinâmicas virtuais dos nós c , f e l numa situação de repouso	40
4.16	Janela de adição de fogo	41
4.17	Tabela de informação	42
5.1	Teste ao sistema: colocação das luminárias virtuais.	44
5.2	Teste ao sistema: Visualização de resposta ao sistema com adição de incêndio. . .	44
5.3	Detetores usados para teste do sistema.	45
5.4	Central endereçável de deteção de incêndio NURIA.	45
5.5	Teste ao sistema: colocação das luminárias virtuais.	46
5.6	Teste ao sistema: Inserção de vários incêndios simultâneos.	47

Lista de Tabelas

2.1	Comparação entre os diferentes algoritmos	16
3.1	Tabela de requisitos funcionais.	22
A.1	Registos das saídas digitais dos laços da central NURIA	51
A.2	Funções para o cálculo do registo do dispositivo D em função do Laço	51
A.3	Excerto de registos das saídas digitais dos dispositivos	52
A.4	Funções para o cálculo do estado do alarme do dispositivo D em função do Laço	52
A.5	Excerto de registos das entradas analógicas dos dispositivos	53

Abreviaturas

ND	Nó de Decisão
SAD	Sistema de Apoio à Decisão
SAD3E	Sistema de Apoio à Decisão para a Evacuação de Emergência de Edifícios
GUI	Graphical User Interface
DPG	DearPy Gui

Capítulo 1

Introdução

Neste documento, apresenta-se um Sistema de Apoio à Decisão para a Evacuação de Emergência de Edifícios (SAD3E). O tema desta dissertação foi proposto pela empresa NIBBLE® e o trabalho foi realizado maioritariamente em ambiente empresarial, nas instalações da empresa.

Neste capítulo introdutório, é feita uma breve exposição do tema. Assim, são apresentados o contexto e a motivação do desenvolvimento deste projeto e uma pequena síntese dos objetivos a cumprir. Segue-se uma pequena apresentação da empresa proponente e as fases de desenvolvimento do projeto. Por fim, apresenta-se a estrutura e organização deste documento.

1.1 Contexto e Motivação

Em 2016, mais de 50% da população vivia em zonas urbanas e é esperado que, em 2030, esta percentagem suba para 60%, sendo que uma em cada três pessoas viverá numa cidade com mais de meio milhão de habitantes[2].

Podemos então inferir que a quantidade de pessoas a ocupar edifícios de grandes dimensões irá também aumentar e, com isto, existe uma probabilidade maior de ocorrerem situações de emergência nestes edifícios. Assim, é necessário continuar a melhorar a resposta a estas mesmas situações.

Os métodos tradicionais de apoio à evacuação de emergência e contenção de perigo são demasiado inflexíveis e estáticos, não tendo a capacidade de adaptação à natureza imprevisível de desastres que podem ocorrer. Em termos de incêndios em edifícios, a utilização de sinalética de emergência estática, portas corta-fogo e aspersores pode muitas das vezes levar a que os ocupantes dos edifícios se dirijam, sem saberem, para a zona de perigo, incorrendo num maior risco e a danos materiais que talvez pudessem ser evitados.

Assim, é perceptível a necessidade de desenvolvimento de métodos que ultrapassem as fragilidades dos métodos tradicionais. Um Sistema de Apoio à Decisão para a Evacuação de Emergência de Edifícios (SAD3E) deve ser capaz de atuar em situações de emergência quando os caminhos de evacuação predefinidos estiverem comprometidos. Nesta situação, a tarefa principal do SAD3E é direcionar os ocupantes do edifício minimizando a sua exposição ao risco.

1.2 Objetivos

O objetivo principal desta dissertação passa pelo desenvolvimento de um sistema capaz de atuar em situações de emergência e, mais concretamente, em situações de incêndio. Este sistema deve atuar em conjunto com uma central de deteção de incêndio. A função da central de incêndio é a de fazer a interface entre o sistema e os detetores e atuadores que fazem parte dos sistemas de segurança do edifício.

Para que o sistema consiga atuar adequadamente terá de cumprir os seguintes objetivos:

- calcular automaticamente o melhor caminho de evacuação numa situação de repouso a partir de um qualquer ponto num edifício;
- recalcular as rotas de evacuação com base em informações recebidas de uma central de deteção de incêndio endereçável;
- apresentar ao utilizador através de uma interface gráfica, o estado do sistema, isto é, os locais que apresentem fogos ativos e as rotas a seguir a partir de qualquer ponto do edifício;
- atuar em saídas para indicação do melhor caminho a seguir (sinalética de emergência).

1.3 A NIBBLE®

A NIBBLE® é uma empresa portuguesa com sede na Trofa presente no mercado desde 2004. É especializada na gestão e desenvolvimento de projetos no âmbito da Engenharia Eletrónica, com Investigação & Desenvolvimento internos. As suas atividades principais são os Produtos e Soluções nas áreas da Segurança e Domótica, GPS e Comunicações Móveis, a Iluminação LED e Eletrónica, os quais são referência no mercado nacional.

A Investigação, Desenvolvimento e Inovação fazem parte do ADN da NIBBLE e é estratégico no seu posicionamento no mercado.

A intenção da proposta deste tema de dissertação surgiu como possível futura integração do SAD3E num produto desenvolvido recentemente pela NIBBLE, uma central de deteção de incêndio endereçável, NURIA.

A NURIA é uma central de deteção de incêndio analógica endereçável de última geração. Uma central de deteção de incêndio endereçável tem a capacidade de identificar os diversos dispositivos ligados ao sistema em que se insere, isto é, permite saber exatamente qual o detetor ou dispositivo que foi ativado. Pelo contrário, as centrais convencionais apenas permitem distinguir entre zonas (grupo de dispositivos num mesmo circuito). As vantagens de uma central endereçável passam principalmente pelo tempo poupado em situações de incêndio ao saber exatamente o local onde se encontra o incêndio.^[3]

1.4 Fases de desenvolvimento

A metodologia seguida na realização deste projeto teve as seguintes fases principais:

1. Estudo de abordagens ao problema da evacuação de edifícios

Esta fase foi desenvolvida na Unidade Curricular de Iniciação à Investigação (M.EEC039) na FEUP e foi durante este período que várias abordagens ao problema de evacuação de edifícios foram estudadas de modo a perceber que tipo de soluções já tinham sido estudadas e qual seria a melhor forma de desenvolver este projeto tendo em conta os requisitos e objetivos do mesmo.

2. Estudo sobre Grafos e Algoritmos de caminho mínimo

Esta fase compreende um estudo sobre diversos algoritmos de procura de caminho mínimo a aplicar na solução do problema, bem como um estudo mais aprofundado do conceito de grafos em programação.

3. Elaboração do programa de procura de caminho mínimo

Nesta fase, foi, inicialmente, elaborado um programa de procura de caminho mínimo num grafo. Este programa permitiria a adição de "incêndios" a um grafo representativo do edifício.

4. Desenvolvimento de uma Interface Gráfica (GUI)

A GUI foi desenvolvida com o intuito de criar um método de visualização e verificação do estado do programa, bem como para permitir a criação de grafos representativos dos edifícios partindo de uma planta.

5. Testes e ajustes ao Sistema

Ao longo do desenvolvimento do projeto, o produto foi sendo testado sempre que novas funcionalidades eram desenvolvidas. Primeiramente, era testado o funcionamento dessas mesmas funcionalidades isoladamente e, posteriormente, eram integradas no sistema e eram executados testes ao sistema como um todo.

1.5 Estrutura da Dissertação

Para além da introdução, esta dissertação contém mais quatro capítulos.

No Capítulo 2, é apresentada uma revisão da literatura. Inicialmente, são apresentadas algumas abordagens ao problema da evacuação de edifícios. Seguidamente, são apresentados estudos de alguns conceitos e temas importantes na fase de desenvolvimento do sistema tais como algoritmos de pesquisa de caminho mínimo, complexidade temporal e espacial de algoritmos e Modbus TCP. Por último, não saindo da fase de desenvolvimento, também foram elaborados estudos comparativos de ferramentas de desenvolvimento de Interfaces Gráficas (GUIs) e ainda de formatos de ficheiros para armazenamento de dados do programa.

No Capítulo 3, é apresentada uma análise ao problema. Aqui é descrita uma situação hipotética onde o SAD3E poderia ser chamado a atuar e quais as suas vantagens em comparação com os sistemas atuais. Para finalizar este capítulo, são apresentados os requisitos do sistema.

No Capítulo 4, é descrita a implementação da solução. Primeiramente, descreve-se a arquitetura do sistema. Depois, explica-se a implementação do algoritmo de procura de caminho mínimo e o modo de implementação e funcionamento da interface gráfica. Por último, são feitos testes ao sistema apresentando a sua resposta a um incêndio dentro de um edifício.

No Capítulo 6, são apresentadas as conclusões retiradas do trabalho, bem como os próximos passos e trabalhos futuros. É também feita uma avaliação dos objetivos do projeto.

Capítulo 2

Revisão da Literatura

Ao longo deste capítulo é apresentada a revisão da literatura e de diversos conceitos, importantes para a compreensão do problema e implementação da solução.

Inicialmente são apresentadas diversas abordagens ao problema da evacuação de edifícios. Este estudo foi especialmente relevante na medida em que forneceu contexto sobre o atual conhecimento deste problema bem como apresentou ideias e possíveis soluções para implementação do SAD3E.

Depois, é apresentada uma explicação do conceito de grafos que é indispensável para a representação dos edifícios de modo a que possa ser interpretada pelo programa.

De seguida, foi executado um estudo de algoritmos de procura de caminho mínimo em grafos, com o objetivo de escolher qual o algoritmo que melhor se adequava ao desenvolvimento do programa.

De modo a conseguir armazenar os dados do grafo representativo dos edifícios para o programa, uma comparação entre diversos formatos de ficheiros para armazenar dados relevantes de forma persistente por parte do programa foi feita.

Dentro do tópico de desenvolvimento da GUI, aferiu-se dentre várias ferramentas para desenvolvimento de GUIs, qual o que melhor se adequava ao projeto.

Por último, uma vez que o programa terá de comunicar com uma central de deteção de incêndio é apresentado o protocolo Modbus TCP que servirá para fazer essa comunicação.

2.1 Abordagens ao problema de evacuação

Após revisão da literatura relevante, Haghani [4], categorizou os esforços de investigação na otimização da evacuação em três sub-domínios: conceção arquitetónica e ajustamento de infraestruturas, programação matemática e otimização da planificação do caminho/partida, e modificação do comportamento, formação e instruções ativas. Tendo isto em mente, o pilar central desta dissertação assenta na abordagem da programação matemática e na otimização da planificação do caminho/evacuação escalonada. Esta abordagem assenta geralmente na procura dos melhores caminhos ou *timings* de partida para um edifício ou infraestrutura pré-determinados.

Nesta secção são apresentadas algumas abordagens consideradas relevantes para o estudo do problema de evacuação de edifícios.

2.1.1 Abordagem de Filippoupolitis e Gelenbe

Filippoupolitis e Gelenbe [5] utilizam uma abordagem distribuída utilizando nós de decisão (ND) em todo o edifício e sinais dinâmicos para direcionar os ocupantes para o caminho de evacuação mais curto e seguro, no entanto, têm poucas ou nenhuma considerações relativamente ao número de pessoas que utilizam um único caminho, o que pode levar a um caminho de evacuação sobrelotado, podendo comprometer a segurança dos ocupantes do edifício.

A solução depende da existência de vários ND, instalados no interior do edifício. Estes dispositivos não precisam de ser muito complexos em termos computacionais, uma vez que o seu trabalho requer apenas o cálculo da direção que deve ser seguida por cada evacuante, em direção à melhor saída disponível. Esta direção é comunicada aos indivíduos na sua vizinhança e é feita através de um indicador visual ou por um dispositivo de comunicação sem fios transportado pelos evacuantes.

A fim de implementar este sistema, o plano do edifício era já conhecido e permitiu a construção de um grafo. Na Figura 2.1, podemos ver um exemplo de grafo correspondente a um andar de um edifício.

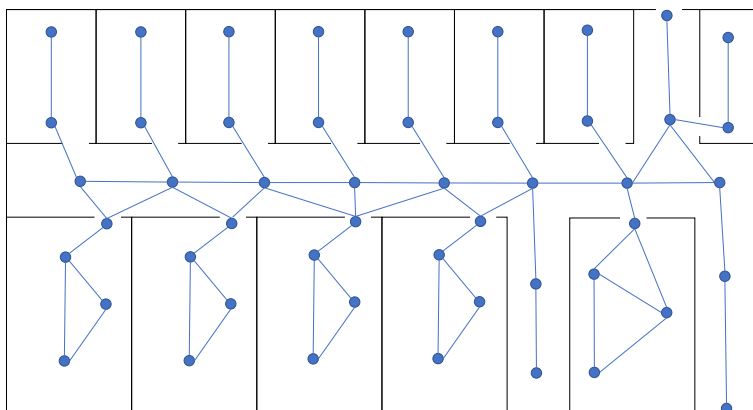


Figura 2.1: Exemplo de grafo correspondente a um andar de um edifício.

Os vértices do gráfico correspondem a locais onde pessoas podem estar reunidas. As ligações entre os vértices (arestas do grafo) representam os caminhos que as pessoas podem seguir. Cada uma destas arestas tem um sensor ligado ao mesmo que monitoriza a intensidade do perigo nessa dada ligação. Para calcular o custo efetivo de cada ligação, a intensidade do perigo é multiplicada pelo comprimento efetivo da ligação. Em situações normais sem perigo na ligação, a intensidade do perigo é fixada em 1 e cresce a partir daí. Ou seja, imaginemos que existe um incêndio de intensidade 50 numa determinada aresta cujo comprimento é de 20 metros, neste caso, o custo

dessa ligação seria de $20 \times 50 = 1000$. Em comparação com as restantes arestas, esta teria um valor muito elevado pelo que o algoritmo desencorajaria a sua utilização.

O SAD proposto poderia ser implementado de uma forma mais centralizada. Nesse caso, o algoritmo funcionaria num dispositivo central de processamento. Este dispositivo teria a função de recolher informação dos sensores, executar o algoritmo e comunicar as decisões resultantes aos respetivos NDs. No entanto, esta abordagem tem inúmeros inconvenientes que têm um impacto negativo na funcionalidade do sistema e podem comprometer o resultado de um procedimento de evacuação. Uma abordagem centralizada não é tolerante a falhas, uma vez que depende de um único ponto de recolha, processamento e transmissão de dados. No caso de existência de uma falha na central de processamento, o sistema de decisão não pode funcionar. Além disso, podem também existir falhas no sistema de comunicação global durante uma emergência. A utilização de comunicações locais dos nós de decisão para informar os evacuantes é mais robusta em comparação com uma solução centralizada que depende da capacidade de comunicar sempre com o centro de processamento.

Mais ainda, o poder computacional necessário para processar todos os dados dos nós de decisão é significativamente maior em comparação com a simples computação dos nós de decisão singulares e, à medida que o tamanho de um edifício (mais andares ou salas) aumenta, o mesmo acontece com os dados a serem processados.

Uma abordagem distribuída também facilita um processo de conceção extensível e escalável. Por outras palavras, os princípios de conceção do sistema distribuído para um determinado tamanho de edifício podem ser escalados para cima ou para baixo para lidar com edifícios maiores ou menores, utilizando os mesmos princípios e blocos de construção.

O algoritmo funciona da seguinte forma. Um ND num determinado vértice u , armazena:

- a distância efetiva (comprimento da aresta $\times$ intensidade do perigo) L de todas as arestas ligadas ao vértice u ;
- para cada nó vizinho n de u , o comprimento efetivo da rota y de n até uma saída e : $L(n, e, y)$;
- o comprimento efetivo do caminho mais curto x , de u até uma saída e : $L(u, e, x)$;
- a sugestão do próximo ND.

Uma vez que assumimos que o sistema já se encontra ativo aquando da ocorrência de algum perigo, isto é, todos os pesos das arestas são a distância real entre os nós, já foi calculado pelos NDs o melhor caminho.

Depois de ser implementado, o sistema de apoio a decisão computa, periodicamente, os pesos das arestas e cada ND calcula qual o próximo ND.

2.1.2 Abordagem de Fang *et al.*

Na abordagem de Fang *et al.* [6], para estudar um processo de evacuação foi utilizado um estádio. O estudo foi conduzido através de uma rede hierárquica direcional, onde cada nó representava

um espaço integrado, tal como bancadas, escadas de ligação de pisos, etc., capaz de acomodar uma certa quantidade de pessoas. Estes nós atuam como destinos intermédios no processo de saída do estádio. As arestas representam os caminhos entre espaços, tais como corredores, passagens de portas, etc.

Este problema de caminho de evacuação reside no campo dos problemas de otimização de caminhos. Neste caso, este problema foi considerado uma otimização multi-objetivo baseada numa rede direcional hierárquica. O problema proposto tem em conta três objetivos principais dentro do processo de evacuação: tempo mínimo de evacuação, distância mínima de evacuação e graus mínimos de congestionamento acumulados.

A abordagem utiliza um Algoritmo de Colónia de Formigas (ACO) uma vez que estes algoritmos têm algumas vantagens em resolver problemas de otimização combinatória, devido à forte capacidade de exploração das formigas.

Os investigadores concentram-se normalmente em duas estratégias principais num algoritmo ACO, nomeadamente a estratégia de pesquisa heurística de uma formiga e o método de atualização de feromonas, uma vez que estas podem afetar a qualidade das soluções e a velocidade de convergência do algoritmo de otimização. O algoritmo desenvolvido diferencia-se do algoritmo ACO nos dois aspetos. Aqui, a estratégia de pesquisa heurística hierárquica pode ser explicada da seguinte forma: uma formiga procura sempre os nós na hierarquia superior a partir do nó em que se encontra atualmente. A estratégia de atualização de feromonas binárias reside no caso em que, numa iteração, existe muita feromona numa determinada ligação. Esta estratégia incentiva as formigas a procurarem melhores soluções minimizando a sobrelotação em alguns caminhos, o que poderia ser problemático para o processo de evacuação.

2.1.3 Abordagem de Pursals e Garzon

Esta abordagem tenta minimizar o tempo de evacuação e concentra-se principalmente na atribuição de ocupantes do edifício a diferentes saídas. A formulação do problema consiste num espaço fechado com k ocupantes, no qual existem n saídas possíveis que estão localizadas de tal forma que não existe interferência nos movimentos dos ocupantes enquanto se dirigem para cada uma delas. Assume-se também que os ocupantes estão dispostos de tal forma que existe um fluxo constante em cada saída. Existe uma rota de evacuação para cada saída e por isso existem n rotas de evacuação de comprimento l_j e com uma área de acesso de a_j .

A evacuação do espaço considera-se terminada quando o último ocupante deixa o edifício. O tempo de evacuação é o tempo que este mesmo ocupante leva para chegar ao exterior do edifício. Todos os ocupantes iniciam a evacuação ao mesmo tempo.

Considerando o fluxo de pessoas em cada saída proporcional à largura da saída (maior largura, maior fluxo de pessoas) podemos alcançar uma evacuação ótima, assegurando que o número de ocupantes alocados a uma saída seja proporcional à relação entre a largura da saída mais pequena e a soma de todas as larguras das saídas no espaço.

Uma vez que o fluxo total para fora do recinto é conhecido, é possível determinar o tempo mínimo de evacuação para o local e o tempo mínimo de evacuação para cada caminho. Isto

permite calcular a distribuição ótima dos ocupantes para cada saída. A alocação ótima é, portanto, dada pelo produto do tempo de evacuação da rota e do fluxo de saída. Esta alocação é crítica, uma vez que atribuir mais ocupantes a uma dada saída poderá aumentar o tempo global de evacuação.

Outra abordagem para modelar o movimento de pessoas passa pela não existência de condição prévia sobre a velocidade e o fluxo do movimento em cada saída. Em vez disso, estas variáveis são simplesmente uma função do número de pessoas que utilizam cada saída e das características geométricas da rota de circulação. A Figura 2.2 mostra os princípios de modelação desta configuração.

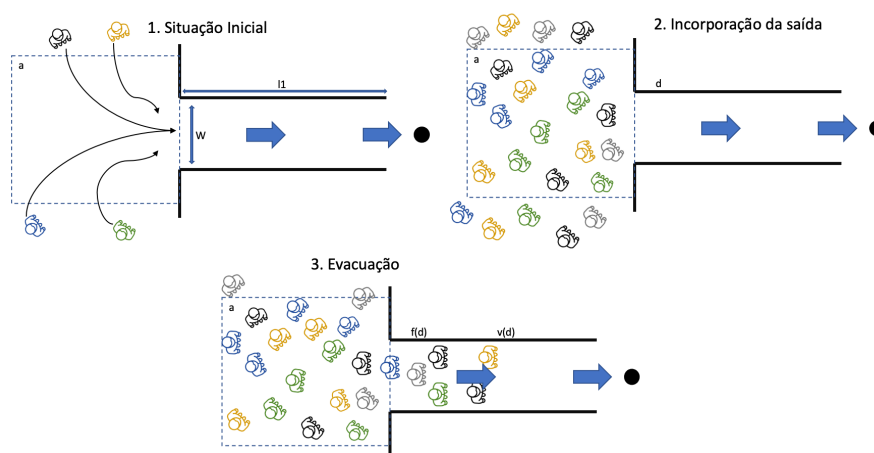


Figura 2.2: Modelação da incorporação a uma via de evacuação.

Com isto em mente, é apresentada a função de evacuação que mostra o tempo de evacuação dado o número de pessoas evacuadas e também a função de evacuação inversa que mostra a quantidade de pessoas evacuadas ao longo do tempo. O algoritmo é então executado tendo em conta os valores mínimos e máximos de pessoas atribuídas às saídas. Estes valores dependem da área de acesso das diferentes saídas. Em seguida, a função de evacuação inversa é calculada para cada saída, ordenando-as tendo em conta os pontos de fluência. Finalmente, para obter a função de evacuação final, as funções de todas as saídas são somadas.

Esta abordagem permite-nos obter resultados muito favoráveis apenas atribuindo pessoas a diferentes saídas, no entanto, o edifício em que se baseia o estudo é muito simplista e amplo, não sendo representativo da maior parte dos edifícios. [7]

2.1.4 Abordagem de Cuesta *et al.*

Este artigo visa apresentar um método de decisão em tempo real para a seleção de rotas de evacuação ótimas, a fim de minimizar os tempos de evacuação e a exposição dos ocupantes ao perigo detetado. O método inclui um modelo matemático, um algoritmo de otimização e um modelo computacional. O algoritmo de otimização baseia-se nas previsões do modelo estocástico de

evacuação que calcula o tempo de evacuação dos ocupantes através de todas as rotas de evacuação disponíveis em cada simulação. O tempo de evacuação de cada ocupante depende do tempo de pré-evacuação, da velocidade de viagem e da distância de viagem. Além disso, o método considera a localização da ameaça e a alteração das condições na disponibilidade de rotas de evacuação.

Este modelo visa fornecer ao utilizador, neste caso, um gestor de emergência, as melhores rotas de evacuação em tempo real. Existe uma GUI que permite ao operador introduzir parâmetros para configurar o modelo.

O modelo recebe uma folha de cálculo com células numeradas que representam a planta do espaço e potenciais rotas de evacuação. Depois, durante a emergência, o utilizador pode introduzir na GUI as áreas do edifício comprometidas pelo perigo. Cada iteração do código verifica as rotas disponíveis para a evacuação.

No artigo, o modelo foi testado num edifício simples, com um grande conjunto de cenários. Do trabalho, concluiu-se que o modelo poderia ser utilizado para apoiar decisões de evacuação durante emergências reais. [8]

2.1.5 Análise comparativa dos algoritmos apresentados

A solução apresentada por Filippoupolitis e Gelenbe [5] apresenta-se como a única que não requer qualquer tipo de intervenção por parte de um utilizador, uma vez que já é um SAD completo. Apresenta-se como uma solução muito robusta uma vez que não está centralizada e depende da rede de NDs. A implementação física do sistema pode ser um problema, uma vez que existe a necessidade de colocar um ND em cada divisão.

Uma vantagem apresentada é que os NDs não precisam de ser muito capazes computacionalmente e que uma solução centralizada necessitaria de um poder de processamento muito mais elevado. Uma vez que este estudo foi realizado em 2008, desde então, a evolução dos computadores foi tal que um sistema centralizado funcionaria provavelmente tão bem e seria provavelmente mais rentável do que a solução atual.

No trabalho de Fang *et al.* foi feita uma comparação entre o algoritmo desenvolvido e três outros algoritmos para avaliar a sua eficiência. Nos testes conduzidos observa-se uma divisão muito mais equilibrada dos ocupantes pelas diferentes saídas do estádio. O principal benefício deste algoritmo quando comparado com outros é a sua capacidade de utilizar plenamente os nós e evitar o congestionamento.

O trabalho de Pursals e Garzon [7] leva em consideração o movimento das pessoas em relação à densidade da ocupação. Isto pode resultar em soluções muito mais fiáveis.

Cuesta *et al.* [8] apresentam alguns testes promissores ao modelo mostrando resultados fiáveis. No entanto, os próprios autores apontam limitações do modelo, tais como o facto de o modelo apenas ter em conta locais com relativamente poucas pessoas e não ter um modo automático de deteção de perigo. No entanto, já foi integrado num protótipo de sistema de apoio à evacuação e testado. A inclusão de monitorização de sensores seria um enorme passo em frente no sistema, uma vez que provavelmente acabaria com a necessidade de iteração com o utilizador durante a evacuação de emergência.

2.2 Grafos

Um grafo é um conjunto finito de nós (também denominados de vértices ou pontos) e um conjunto de ligações (também denominadas de arestas) que conectam pares de nós. Geralmente, está associado a cada aresta um valor, representando, por exemplo, o custo, a capacidade ou o comprimento da ligação.

Os grafos podem ser direcionais quando as arestas apenas tem valor relevante numa determinada direção, isto é a ligação entre um nó A e um nó B tem um valor de x mas a ligação inversa, ou seja, entre o nó B e nó A, não toma nenhum valor. Quando um grafo é não direcional, as arestas são bi-direcionais e o seu valor é constante, independentemente da direção tomada.[9]

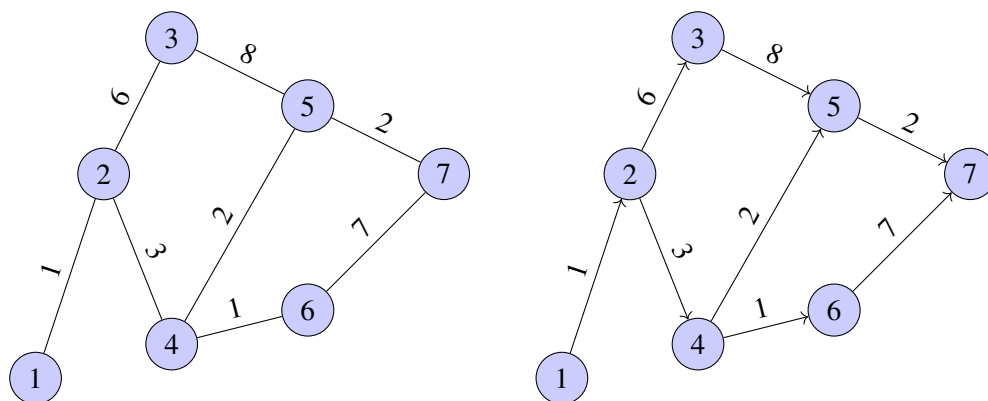


Figura 2.3: (Esq.) Grafo não direcional (Dir.) Grafo direcional

Na Figura 2.3, podemos observar a diferença entre grafos direcionais e não direcional. Por exemplo, partindo do **nó 1** e querendo chegar ao **nó 4**, enquanto que no grafo não direcional existem três caminhos possíveis: **1 -> 2 -> 4**, **1 -> 2 -> 3 -> 5 -> 4** e **1 -> 2 -> 3 -> 5 -> 7 -> 6 -> 4**. No grafo direcional, apenas existe o caminho **1 -> 2 -> 4**.

Em ciência de computadores um grafo é uma estrutura de dados abstrata cujo objetivo é implementar os conceitos de grafo direcional e não-direcional do ramo da teoria de grafos da Matemática. São bastante úteis, por exemplo, para resolver problemas de procura de caminho mínimos e outros problemas de otimização.

2.3 Algoritmos de procura de caminho mínimo

Nesta secção é apresentado um estudo de possíveis algoritmos a usar no desenvolvimento do SAD3E para encontrar os melhores caminhos para evacuação.

Para todos os efeitos, a missão de um SAD3E é a de tentar evacuar as pessoas que se encontram dentro de um edifício cujo interior apresenta algum perigo. Assim, este sistema deverá ser capaz de interpretar a morfologia do edifício e apresentar caminhos seguros para os evacuantes, tendo em conta as zonas do edifício afetadas pelos tais perigos. Uma maneira de representar e armazenar computacionalmente esta morfologia incide na utilização de grafos.

Desta forma, podemos representar um edifícios com os nós que representam zonas do edifício sejam elas salas, corredores, escadas, portas de saída, etc. e as ligações entre si. Estas ligações são uma representação da peso da deslocação de um nó para outro.

Para conseguirmos levar as pessoas em segurança para a saída do edifício é conveniente que o caminho que estas percorram seja o mais curto e seguro. É aqui que entram os algoritmos de escolha de caminho mínimo. A escolha correta destes algoritmos baseada na sua performance para o tipo de problema que aqui apresentamos pode ser fulcral na rápida e segura evacuação de pessoas resultando, em último caso, na possível poupança de vidas.

De seguida serão apresentados três dos mais conhecidos e utilizados algoritmos de procura de caminho mínimo: algoritmos de Dijkstra, Bellman-Ford e Floyd-Warshall[10].

2.3.1 Dijkstra

O algoritmo de Dijkstra calcula o caminho mais curto desde um vértice do grafo para todos os restantes vértices. Correndo nó a nó de modo a determinar a árvore de caminhos ótimos.

- Inicializamos um vetor $D[i]$ que representa a distância do nó inicial V até ao nó V_i . Ou seja, caso o nó V_i seja adjacente ao nó V , $D[i]$ é o peso da aresta. Caso contrário, $D[i] = \text{inf}$.
- Procuramos um nó V_j , adjacente ao nó V com a menor distância ao nó V .
- Procuramos um nó V_k , adjacente a V_j cuja distância ao último é menor. Conclui-se então que $D[k] = \min\{D[k], D[j] + \text{distância de } V_j \text{ a } V_k\}$.
- Repetimos a ação anterior até chegarmos ao nó pretendido.

No Algoritmo 1, está representado o pseudo-código do algoritmo de Dijkstra.

```

input : Grafo  $G = (V, E)$ 
pesos positivos das arestas  $l_e : e \in E$ 
vértice  $s \in V$ 
output: Para todos os vértices  $u$ , alcançáveis partindo de  $s$ ,  $dist(u)$  é a distância do
vértice de partida  $s$  a  $u$ 

for  $u \in V$  do
     $dist(u) = \infty$ 
     $prev(u) = nil$ 
end
 $dist(s) = 0$ 
 $H = makequeue(V)$ 
while  $H$  is not empty do
     $u = deletemin(H)$ 
    for edges  $(u, v) \in E$  do
        if  $dist(v) > dist(u) + l(u, v)$  then
             $dist(v) = dist(u) + l(u, v)$ 
             $prev(v) = u$ 
             $decreasekey(H, v)$ 
        end
    end
end

```

Algoritmo 1: Pseudo-código do Algoritmo de Dijkstra.

2.3.2 Bellman-Ford

O algoritmo de Bellman-Ford calcula o caminho mais curto desde um vértice do grafo para todos os restantes vértices.

- Inicializamos um vetor $D[i]$ que representa a distância do nó inicial V até ao nó V_i .
- Executamos o seguinte ciclo com $n - 1$ iterações (em que n é o número de vértices do grafo):
 - Se $(D[k] > D[j] + \text{peso da aresta que liga } j \text{ a } k)$, então $D[k] = D[j] + \text{peso da aresta que liga } j \text{ a } k$

No Algoritmo 2, está representado o pseudo-código do algoritmo de Bellman-Ford.

```

input : Grafo  $G = (V, E)$ 
pesos das arestas  $l_e : e \in E$  sem ciclos negativos
vértice  $s \in V$ 
output: Para todos os vértices  $u$ , alcançáveis partindo de  $s$ ,  $dist(u)$  é a distância do
vértice de partida  $s$  a  $u$ 

for  $u \in V$  do
|    $dist(u) = \infty$ 
|    $prev(u) = nil$ 
end
 $dist(s) = 0$ 
repeat
|   for  $e \in E$  do
|   |    $dist(v) = \min\{dist(v), dist(u) + l(u, v)\}$ 
|   end
until  $|V| - 1 = 0$ 

```

Algoritmo 2: Pseudo-código do Algoritmo de Bellman-Ford.

2.3.3 Floyd-Warshall

O algoritmo de Floyd-Warshall compara todos os caminhos possíveis entre qualquer par de vértices, encontrando assim o caminho mínimo entre todos os nós.

- Inicializamos uma matriz de distâncias $D[i][j]$ com dimensões $V \times V$ em que V é o número de vértices do grafo. Na sua diagonal colocamos todos os valores a 0 uma vez que a distância de um nó a si próprio é 0.
- De seguida, adicionamos os restantes valores das arestas que já conhecemos, ou seja, caso exista uma aresta $u - v$, $D[u][v] = \text{peso da aresta } u - v$.
- A parte principal do algoritmo depende de um conjunto de ciclos encadeados. Caso a condição $D[i][j] > D[i][k] + D[k][j]$, então $D[i][j] = D[i][k] + D[k][j]$.

O pseudo-código do algoritmo de Floyd-Warshall pode ser visto no Algoritmo 3.

```

input : Grafo  $G = (V, E)$ 
pesos das arestas  $l_e : e \in E$ 
output: Matriz de distâncias mínimas  $D[i][j]$ 

for  $u \in V$  do
   $dist[v][v] = 0$ 
end
for  $edges(u, v) \in E$  do
   $dist[u][v] = l(u, v)$ 
end
for  $k \in V$  do
  for  $i \in V$  do
    for  $j \in V$  do
      if  $dist[i][j] > dist[i][k] + dist[k][j]$  then
         $dist[i][j] = dist[i][k] + dist[k][j]$ 
      end
    end
  end
end

```

Algoritmo 3: Pseudo-código do Algoritmo de Floyd-Warshall.

2.3.4 Comparação dos Algoritmos apresentados

Depois de apresentados os algoritmos de Dijkstra, Bellman-Ford e Floyd-Warshall, é necessário encontrar uma forma de os avaliar e comparar. Para compararmos algoritmos e a sua performance, normalmente recorremos aos conceitos de complexidade temporal e espacial.

2.3.4.1 Complexidade Temporal e Espacial

Normalmente, num algoritmo, os recursos a ter em maior consideração são o espaço ocupado em memória e o tempo despendido na execução do mesmo. Assim, para avaliar estas duas grandezas no que toca a algoritmos e programas, utiliza-se a notação de *Big O*. A letra O foi escolhida por Paul Bachmann que significa *Ordnung*, ou seja, ordem de aproximação [11].

Então, quando dizemos que uma função $T(n) = O(f(n))$ estamos a garantir que a taxa de crescimento da função $T(n)$ é inferior a $f(n)$.

Como exemplo, n^3 cresce mais rapidamente que n^2 , por isso poderíamos dizer que $n^2 = O(n^3)$. $f(n) = n^2$ e $g(n) = 2n^2$ crescem da mesma forma por isso poderíamos dizer que $f(n) = O(g(n))$. No entanto, esta notação está muito dependente do contexto para ser relevante, ou seja, se $g(n) = 2n^2$, $g(n) = O(n^4)$, $g(n) = O(n^3)$ e $g(n) = O(n^2)$ estão todos tecnicamente corretos, no entanto, a designação mais correta e relevante seria a última [12].

2.3.4.2 Análise Comparativa

Uma das principais características que distingue os algoritmos apresentados é a capacidade de calcular caminhos mínimos sabendo que os grafos podem ter arestas com pesos negativos. No entanto, neste caso específico, tal não é aplicável.

Tabela 2.1: Comparação entre os diferentes algoritmos

	Dijkstra	Bellman-Ford	Floyd-Warshall
Complexidade espacial	$O(M)$	$O(M)$	$O(N^2)$
Complexidade temporal	$O(N^2)$	$O(MN)$	$O(N^3)$
Pesos das arestas negativos	×	✓	✓
M é o número de arestas e N é o número de vértices			

Em termos de complexidade temporal, o algoritmo de Dijkstra é o mais eficiente tendo em conta que o número de arestas é significativamente superior ao número de nós.

Em termos de complexidade espacial, quer os algoritmos de Dijkstra e Bellman-Ford utilizariam menos espaço, caso o número de arestas for inferior ao quadrado do número de vértices.

Por estas razões a escolha para o desenvolvimento do algoritmo de escolha de caminho mínimo recaiu no algoritmo de Dijkstra.

2.4 Comparação de formatos de ficheiro para armazenamento de dados

Para efetuar trocas de dados entre sistemas ou programas, pode ser necessário recorrer a ficheiros, no entanto, estes ficheiros podem ser escritos numa imensa panóplia de formatos. Atualmente, existem três principais tipos de formatos utilizados em computação para fazer esta comunicação e são eles JSON, XML e CSV.

JSON tem como significado *JavaScript Object Notation* e é um formato compacto, *open standard*, de troca de dados simples e rápida entre sistemas, especificado por Douglas Crockford em 2000, que tem a particularidade de conseguir ser legível para um humano[13].

Extensible Markup Language (XML) é uma linguagem de marcação e um formato de ficheiro para armazenar, transmitir, e reconstruir dados arbitrários. Define um conjunto de regras para a codificação de documentos num formato que é simultaneamente legível por humanos e por máquinas[14].

Um ficheiro de *Comma Separated Values* (CSV) é um ficheiro de texto delimitado que utiliza uma vírgula para separar valores. Cada linha do ficheiro é um registo de dados. Cada registo consiste num ou mais campos, separados por vírgulas. Um ficheiro CSV armazena normalmente dados tabulares (números e texto) em texto simples, caso em que cada linha terá o mesmo número de campos[15].

Alguns atributos dos formatos são:

- CSV:

- Tamanho de ficheiro muito pequeno relativo aos outros dois formatos;
- Sem sintaxe relevante;
- Não permite hierarquias.
- JSON:
 - Tamanho de ficheiro aproximadamente duas vezes superior ao do CSV;
 - Começa a ser bastante utilizado na indústria;
 - Permite uma escrita e interpretação mais fácil.
- XML:
 - Tamanho de ficheiro aproximadamente três vezes superior ao do CSV, tornando-o no mais pesado;
 - Muito usado mas começa agora a ser ultrapassado pelo JSON;
 - Sintaxe do ficheiro mais complexa que JSON e mais difícil de interpretar.

Visto que o formato JSON é bastante usado na indústria, é intermédio em termos de espaço ocupado no armazenamento e uma vez que é de leitura e interpretação mais fácil, para o desenvolvimento do projeto, nomeadamente na questão de armazenamento persistente de dados da aplicação, foi escolhido este formato.

2.5 Comparação de *toolkits* para desenvolvimento de GUIs

Para o desenvolvimento da GUI foi utilizado o **DearPyGui**, um *toolkit* para Python de fácil uso, dinâmico, acelerado por GPU e *cross-platform* e é baseado na biblioteca de C++ Dear ImGui.

No entanto também foram testados outros *toolkits* para desenvolvimento de GUIs:

- **Tkinter** - apresenta inúmeras vantagens, nomeadamente o facto de estar incluído na biblioteca base de Python, ser muito simples e de fácil aprendizagem. Uma vez que é bastante utilizado está muito bem documentado. Quanto às desvantagens, peca por ser bastante pobre nos componentes disponíveis o que torna o desenvolvimento de algumas aplicações muito complexo ou mesmo impossível. A sua performance também é baixa[16].
- **PyQt5** - ao contrário do Tkinter, o PyQt5 apresenta widgets complexos e permite desenvolver aplicações complexas. Uma vez que é bastante usado na indústria também apresenta vários recursos de aprendizagem e ainda tem uma interface que permite o desenvolvimento de GUIs sem recurso a código através do QtDesigner. No entanto, as desvantagens passam por ter pouca documentação relativa a algumas classes em Python (Qt é maioritariamente usado com C++) e ainda apresenta uma curva de aprendizagem muito íngreme[17].

Concluindo, a escolha recaiu no uso de DearPy Gui pela sua facilidade de desenvolvimento, através de *containers* o que facilita a compreensão do código e debugging e pela complexidade que podemos facilmente obter através de widgets simples.

2.6 MODBUS TCP

MODBUS é um protocolo de mensagens de camada de aplicação e proporciona comunicação cliente/servidor entre dispositivos ligados em diferentes tipos de barramentos ou redes.

É o standard a nível industrial desde 1979 e continua a permitir a comunicação de milhões de dispositivos de automação.

O MODBUS é um protocolo de request/reply e oferece serviços especificados por códigos de função. Os códigos de função MODBUS são elementos de PDUs de request/reply MODBUS. [\[18\]](#)

Este protocolo será utilizado na comunicação do SAD3E com a central de incêndio endereçável NURIA.

Capítulo 3

Análise do problema

Neste capítulo é apresentada uma explicação do atual paradigma das evacuações de emergência, os possíveis problemas que o SAD3E pode resolver e ainda os requisitos definidos para o sistema desenvolvido.

3.1 Cenário de atuação

Imagine-se que a família Fonseca se encontra de férias e decide ir às compras ao grande centro comercial local. Estacionam o carro no parque de estacionamento ao ar livre e deslocam-se à entrada mais próxima para começarem a fazer as suas compras. Passados alguns instantes e depois de já terem percorrido várias lojas no grande centro comercial ouvem um sinal sonoro de incêndio e ouvem ordens nos altifalantes para se manterem calmos e se dirigirem à saída mais próxima. Uma vez que nunca tinham estado naquele edifício e, como tal, não conheciam a sua morfologia, com algum pânico, tentaram dirigir-se à saída pela qual entraram.

Segundo um estudo, apenas 38% das pessoas atentam à sinalética de emergência durante uma evacuação em ambiente pouco familiar. Como tal, os ocupantes do edifício podem ficar confusos e, geralmente, retornam ao local por onde entraram no edifício, o que pode levar a engarrafamentos no processo de evacuação e atrasar o mesmo[1].

Seria numa situação semelhante a esta que o SAD3E seria muito útil. Visto que estamos habituados a que a sinalética de emergência seja estática e com indicações concordantes, ou seja, as indicações não oferecem informação sobre a viabilidade do caminho e não apresentam caminhos a não seguir, em situações de emergência podemos optar por escolher o ponto de saída ou caminho que conhecemos mesmo que este não seja o ideal.

Caso um sistema como o SAD3E estivesse implementado, a sinalética de segurança seria dinâmica e alterar-se-ia mediante as condições e avanço do incêndio no interior do edifício. Isto é, as luminárias iriam ajustar-se, mudando o seu estado, e direcionar os ocupantes do edifício para a saída mais próxima, informando também quais os caminhos bloqueados ou perigo de serem percorridos, através de sinalética de conotação negativa como por exemplo, uma cruz vermelha.

3.2 Estudo de Mercado

Atualmente, no que diz respeito a evacuação de emergência, existem muito poucas soluções no mercado capazes de se ajustar dinamicamente à natureza imprevisível das situações de emergência.

Nesta secção, é apresentada a diferença entre as soluções existentes no mercado no que diz respeito a luminárias de emergência. Depois é descrita uma solução da Eaton e ainda a possível implementação futura por parte da NIBBLE®.

3.2.1 Tipos de sistemas de iluminação de emergência

Segundo a Eaton [1], existem três tipos de sistemas de iluminação de emergência. Esta caracterização assenta na capacidade das luminárias de alterarem o seu estado e comunicarem diferentes informações aos ocupantes do edifício.

O tipo de sistema mais usual e que está presente na maior parte dos edifícios é o sistema estático. Neste sistema, as luminárias apresentam sempre o mesmo estado e não tem capacidade de se alterar. Na Figura 3.1, podemos observar um exemplo do funcionamento deste sistema.



Figura 3.1: Exemplo de funcionamento de sistema de luminárias do tipo estático [1].

Existem também sistemas um pouco mais complexos do que os estáticos. Os sistemas dinâmicos têm a capacidade de alternar entre dois estados, no entanto, só voltam ao estado inicial depois de o sistema ser reinicializado. Por exemplo, imagine-se uma luminária com dois estados: uma cruz vermelha que indica que o caminho está bloqueado e uma seta branca em fundo verde que indica o caminho a seguir. Numa situação de evacuação, o sistema dinâmico seria capaz de alterar o estado da luminária caso o caminho estivesse bloqueado, no entanto, após o desbloqueio desse caminho, o sistema seria incapaz de voltar a alterar o estado da luminária para o estado inicial. Na Figura 3.2, podemos observar um exemplo deste mesmo sistema.



Figura 3.2: Exemplo de funcionamento de sistema de luminárias do tipo dinâmico [1].

Por último, o sistema mais complexo e completo é o sistema adaptável. Neste caso, o sistema é mais completo que o sistema dinâmico, conseguindo alterar o estado das luminárias para o estado inicial. Atentando ao exemplo usado anteriormente, caso o caminho bloqueado ficasse livre, o sistema era capaz de alterar o estado da luminária indicando que o pode ser usado. Por forma a ilustrar este sistema podemos observar a Figura 3.3



Figura 3.3: Exemplo de funcionamento de sistema de luminárias do tipo adaptável [1].

3.2.2 Eaton - Evacuação de emergência adaptável

A Eaton oferece, no seu leque de produtos, uma solução para a evacuação de emergência de edifícios com vários benefícios em relação ao normal paradigma das soluções presentes no mercado.

A solução proposta reside num sistema de evacuação adaptável que utiliza, de entre vários componentes, as luminárias adaptáveis apresentadas na Figura 3.3. Este sistema, quando instalado, pode ser programado com uma variedade de possíveis rotas de saída. Depois, com base na informação obtida através de detetores, circuitos de televisão fechados (CCTV) e outros dispositivos, o sistema pode escolher qual a rota mais rápida e segura para os ocupantes do edifício e orientar os ocupantes em conformidade. Entretanto, um operador do sistema pode aceitar ou rejeitar as recomendações de rota. [19]

Uma das principais limitações do sistema passa pela possibilidade de não conseguir atuar em todos os cenários possíveis de forma autónoma. Ou seja, se quando foi executada a configuração do sistema, não forem adicionadas todas as rotas possíveis, este pode não conseguir ser eficiente na sua tarefa. Pode acontecer de um dos cenários não ter sido pensado pelos instaladores do sistema e, numa situação de emergência, o operador do sistema pode não conseguir dar uma resposta atempada. No entanto, a Eaton afirma que a tecnologia já foi testada extensivamente e que está em conformidade com as normas e regulamentos.

3.2.3 Implementação futura por parte da NIBBLE®

Um dos objetivos da NIBBLE® no futuro é o de desenvolver um produto de sinalética de emergência adaptativa com algumas vantagens em relação aos que já existem no mercado.

A ideia inicial com a proposta de execução deste projeto seria a utilização do SAD3E em conjunto com a central de deteção de incêndio endereçável NURIA e esta sinalética adaptativa. Com estes três constituintes seria possível construir um sistema completo e que proporcionaria evacuações mais seguras e céleres, minimizando os riscos para os indivíduos em processo de evacuação de emergência.

3.3 Requisitos do sistema

Nesta secção são apresentados e descritos os requisitos do sistema. Fundamentalmente, o que é pedido do SAD3E é que seja um sistema autónomo, configurável e de rápida resposta a estímulos de incêndio. Na Tabela 3.1 podemos observar a lista de requisitos funcionais.

Tabela 3.1: Tabela de requisitos funcionais.

Requisitos Funcionais	
Código do Requisito	Descrição do Requisito
RF1	O sistema deve poder ser usado em edifícios com diversas tipologias
RF2	O sistema deve permitir que o utilizador defina quais os caminhos possíveis dentro do edifício
RF3	O sistema deve calcular automaticamente o melhor caminho de evacuação
RF4	O sistema deve ter a capacidade de comunicar com a central de incêndio endereçável
RF5	O sistema deve ser capaz de recalcular os caminhos de evacuação sempre que receber um sinal de alarme da central sem intervenção do utilizador
RF6	O sistema deve ser capaz de recalcular os caminhos de evacuação num período inferior a cinco segundos
RF7	O sistema deve ser capaz de controlar as indicações do melhor caminho a seguir através de sinalética de emergência.
RF - Requisito Funcional RNF - Requisito Não Funcional	

Os requisitos **RF1** e **RF2** asseguram que o sistema é configurável, isto é, asseguram que este pode ser utilizado em qualquer tipo de edifício, independentemente da sua configuração e que o utilizador pode escolher quais os caminhos passíveis de utilização por parte dos ocupantes em caso de evacuação.

Os requisitos **RF3**, **RF4** e **RF5** asseguram a automação do sistema, uma vez que este não necessita de interação com o utilizador para funcionar.

O requisito **RF6** assegura que o sistema tem uma resposta rápida a estímulos de incêndio e o requisito **RF7** garante que o sistema consegue comunicar aos ocupantes do edifício a melhor rota de evacuação.

Capítulo 4

Implementação da Solução

Neste capítulo, é apresentada uma descrição detalhada da implementação da solução. Em primeiro lugar é explanada a arquitetura do sistema e quais os seus constituintes. De seguida, apresentam-se as implementações do algoritmo, da interface gráfica e da comunicação com a central de deteção de incêndio.

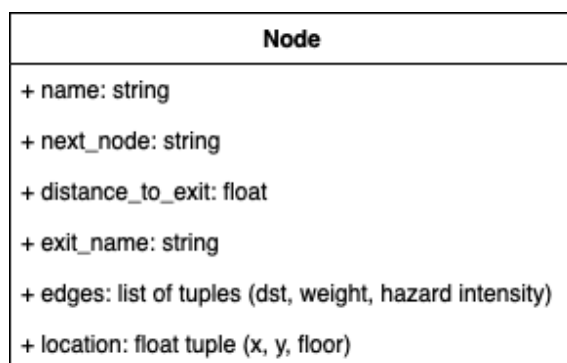
4.1 Arquitetura do Sistema

Depois de estudadas algumas abordagens ao problema de evacuação de edifícios e definidos quais os requisitos e objetivos do sistema é agora altura de abordar a sua organização e constituintes.

O SAD3E desenvolvido baseia-se na solução apresentada por Filippoupolitis e Gelenbe [5]. Esta solução utiliza uma abordagem distribuída através de dispositivos (ND) com capacidade computacional reduzida que estão instalados no edifício. As únicas informações armazenadas nos ND são a sua distância à saída, o comprimento das ligações a eles conectados e qual o nó a dirigir-se em contexto de evacuação. Neste sistema, a monitorização de incêndio ocorre através de sensores nas arestas e, periodicamente, o sistema calcula novamente os novos caminhos a seguir.

4.1.1 Objeto Node

Inspirado pelo sistema proposto por de Filippoupolitis e Gelenbe, optou-se por representar os edifícios por grafos. Cada nó do grafo seria representativo de um espaço ou divisão onde podem estar presentes os ocupantes do edifício. No entanto, ao contrário desta abordagem, esta representação acontece em software, ou seja, os ND não são dispositivos fixos colocados no interior do edifício. São uma classe de código cujos atributos contém informações relevantes. Na Figura 4.1, podemos observar a classe `Node` cujos atributos são semelhantes às informações guardadas pelos ND de Filippoupolitis e Gelenbe.

Figura 4.1: Objeto **Node**.

A descrição de cada atributo da classe `Node` é apresentado a seguir:

- `name` - contém uma *string* com o nome do nó.
- `nex_node` - contém uma *string* com o nome do nó seguinte no caminho para a saída. Isto é, caso um ocupante do edifício estivesse nas imediações do nó deveria dirigir-se para a localização do nó `next_node` e assim sucessivamente, percorrendo a melhor rota até chegar à saída.
- `distance_to_exit` - contém o somatório de todos os pesos das arestas até à saída, ou seja, a distância total da rota de evacuação.
- `exit_name` - contém uma *string* com o nome do nó da saída destacada à rota de evacuação em que o nó está inserido.
- `edges` - é uma lista de *tuples* que contém as informações das arestas ligadas ao nó. O valor `dst` contém o nome do nó conectado; `weight` - comprimento da ligação e ainda `hazard_intensity` que representa a intensidade do perigo existente naquela aresta. Segundo a abordagem utilizada para o desenvolvimento do SAD3E, o valor real da aresta é calculado multiplicando o valor do seu comprimento(`weight`) e a intensidade do perigo na aresta (`hazard_intensity`).
- `location` - apresenta as coordenadas do nó dentro da planta do edifício.

4.1.2 Constituintes do sistema

O SAD3E desenvolvido apresenta três principais constituintes: o algoritmo de procura de caminho mínimo, a interface gráfica (GUI) e a interface de comunicação com a central de deteção de incêndio. Na Figura 4.2, podemos observar os constituintes do sistema e as suas interações

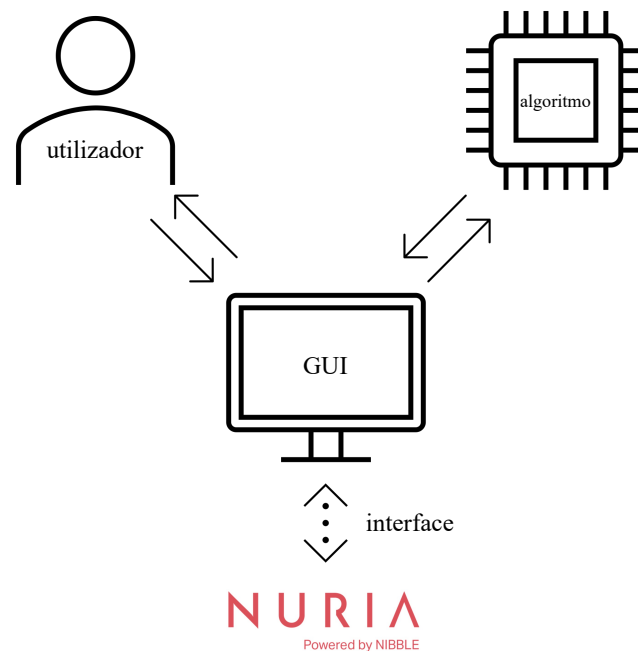


Figura 4.2: Esquemático ds constituintes do SAD3E.

Aqui, o núcleo e base do sistema é sempre o algoritmo de procura de caminho mínimo, isto porque é nele que o funcionamento do sistema assenta. É função do algoritmo calcular e modificar os atributos mutáveis de todos os objetos da classe `Node` presentes no grafo, nomeadamente os atributos `next_node`, `distance_to_exit` e `exit_name`.

Existe ainda uma interface responsável pela comunicação entre a central e a interface gráfica. Neste caso, esta comunicação é feita com recurso ao protocolo Modbus TCP.

Por último, a interface gráfica é responsável por receber os *inputs* do utilizador e apresentar-lhe o estado do sistema. A função da GUI é gerir quando é que o algoritmo deve ser corrido, quer a pedido do utilizador ao adicionar um "incêndio" manualmente na interface ou quando a central de deteção deteta um incêndio.

4.2 Implementação do algoritmo

Nesta secção é aprofundada a implementação do algoritmo de procura de caminho mínimo com o intuito de descobrir quais as melhores rotas de evacuação a partir de qualquer parte do edifício.

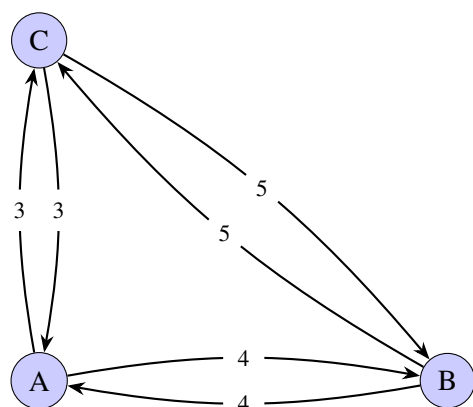
O funcionamento do algoritmo, bem como a sua implementação tem quatro fases principais. Inicialmente, explicar-se-á como é feita a representação com grafos de edifícios e como é que é construído o grafo representativo. Depois, aborda-se o algoritmo de procura de caminho mínimo

e ainda a atribuição das saídas aos diferentes nós do grafo. Por último, explica-se como funciona a inserção de um incêndio no grafo e de que forma é que isso afeta as arestas do grafo e o recálculo da rota de evacuação.

4.2.1 Construção do grafo

No início do programa e para que seja possível correr o algoritmo de pesquisa de caminho mínimo, é necessário proceder à construção do grafo que representa o edifício.

Isto é feito a partir da leitura de um ficheiro do formato `.json` que contém um dicionário de Python com os atributos e informações do grafo. No excerto de código ??, podemos observar um exemplo do texto contido num ficheiro `.json`.



```

1 "A": {
2   "dst": ["B", "C"],
3   "weight": [4, 5],
4   "hazard": [1, 1],
5   "location": [0, 0, 0]
6 },
7 "B": {
8   "dst": ["A", "C"],
9   "weight": [4, 5],
10  "hazard": [1, 1],
11  "location": [4, 0, 0]
12 },
13 "C": {
14  "dst": ["A", "B"],
15  "weight": [3, 5],
16  "hazard": [1, 1],
17  "location": [0, 3, 0]
18 }
  
```

Figura 4.3: Exemplo de grafo guardado no formato `.json`.

Os três nós tem os nomes A, B e C e as arestas conectam os nós em triângulo. A aresta A-B tem comprimento **4**, a aresta B-C tem comprimento **5** e a aresta C-A tem comprimento **3**. Todas elas tem intensidade do fogo 1.

De seguida, é construído um grafo utilizando a função `build_graph()` que recebe como argumento o dicionário com as mesmas características do apresentado no excerto de código ??. A função retorna um objeto do tipo `defaultdict` com a construção do grafo. No excerto de código 4.1 podemos observar o funcionamento da função `build_graph()`.

```

1 from collections import defaultdict
2
3 def build_graph(edges):
  
```

```

4     graph = defaultdict(Node)
5     seen_edges = defaultdict(int)
6     for src in edges:
7         for dst, weight, hazard in zip(edges.get(src).get("dst"),
8                                         edges.get(src).get("weight"),
9                                         edges.get(src).get("hazard")):
10            seen_edges[(src, dst)] += 1
11            # checking for duplicated edge entries
12            if seen_edges[(src, dst)] > 1:
13                continue
14            graph[src].edges.append((dst, weight, hazard))
15            graph[src].location = edges.get(src).get("location")
16            graph[src].name = src
17     return graph

```

Excerto de código 4.1: Função **build_graph()**.

4.2.2 Cálculo do caminho mínimo até à saída

Tal como referido no Capítulo 2, Secção 2.3, o algoritmo de pesquisa de caminho mínimo escolhido foi o algoritmo de Dijkstra. Esta implementação foi feita através de uma função que recebe como argumentos o grafo e qual o nó de partida e retorna a distância a todos os nós do grafo e o caminho para todos eles. No excerto de código 4.2, podemos ver esta função.

```

1 def dijkstra_distance(graph, src_node):
2     dist = dict()
3     prev = dict()
4     for node in graph:
5         # set distance to infinity
6         dist[node] = float('inf')
7         prev[node] = None
8
9     dist[src_node] = 0
10
11     heap = [(dist[src_node], src_node)]
12     seen = set()
13     while heap:
14         node = heapq.heappop(heap)[1]
15         if node not in seen:
16             seen.add(node)
17             for dst, weight, hazard in graph[node].edges:
18                 if dist[dst] > dist[node] + weight * hazard:

```

```

19         dist[dst] = dist[node] + weight * hazard
20         prev[dst] = node
21         heapq.heappush(heap, (dist[dst], dst))
22
23     return dist, prev

```

Excerto de código 4.2: Implementação do Algoritmo de Dijkstra.

De notar que, à semelhança da solução proposta por Filippoupolitis e Gelenbe [5], o peso das arestas para cálculo do caminho mínimo, é feito através da multiplicação da distância real entre os nós pela intensidade do perigo nesse segmento do grafo, ou seja, se a distância entre um **nó A** e um **nó B** é 10 e a intensidade do perigo nessa ligação é 50, então, para o cálculo do caminho, o peso dessa ligação é $10 \times 50 = 500$.

4.2.2.1 Atribuição das saídas

Como é normal em edifícios de grandes dimensões, podem existir várias saídas para o exterior. Então, mais do que encontrar o caminho mais curto e seguro para uma saída, é necessário escolher qual o caminho mais curto de entre as várias saídas e atribuir, assim, a cada nó uma determinada saída. Para esta necessidade foi desenvolvida a função `set_nearest_exit()` que recebe como argumentos o grafo representativo do edifício e uma lista com o nome dos nós que representam as saídas do edifício.

```

1 def set_nearest_exit(graph, exit_array):
2     for node in graph:
3         graph[node].set_distance(float('inf'))
4     for exit in exit_array:
5         distance, prev = dijkstra_distance(graph, exit)
6         for node in graph:
7             path = find_path(prev, node)
8             if graph[node].get_distance() > distance[node]:
9                 graph[node].set_distance(distance[node])
10                graph[node].exit = exit
11                if exit != node:
12                    graph[node].set_next_node(path[1])
13            else:
14                graph[node].set_next_node(path[0])

```

Excerto de código 4.3: Implementação da função de atribuição de saídas.

No excerto de código 4.3 podemos observar a função `set_nearest_exit()` que começa por reinicializar a distância de todos os nós do grafo à respetiva saída. De seguida, para cada nó do grafo, calcula a distância a cada uma das saídas do edifício, guardando apenas a saída mais conveniente. É neste momento que é também definido qual o nó a seguir e a distância à saída.

4.2.2.2 Inserção de incêndio e recálculo das rotas

Em caso de incêndio no interior de um edifício, durante a evacuação, é importante afastar ao máximo os ocupantes da zona de fogo, mesmo que isso os leve a percorrer distâncias maiores. A maneira de conseguir desencorajar o programa a escolher caminhos que percorram as imediações do local afetado pelo incêndio, foi criar uma área de afetação nestas imediações. Por forma a ilustrar este conceito, atente-se a Figura 4.4.

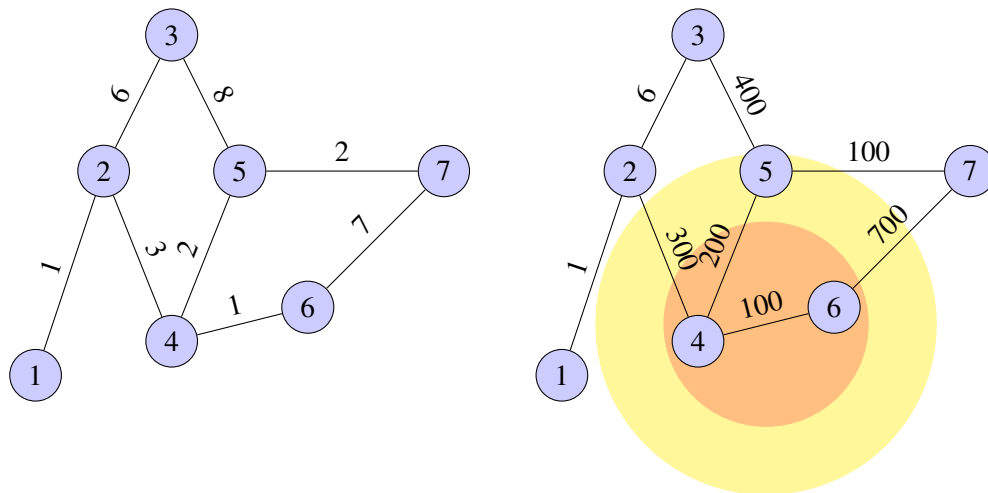


Figura 4.4: (Esq.) Grafo sem incêndio (Dir.) Grafo com incêndio e zona de afetação.

Na Figura 4.4, podemos observar que, não só o peso da aresta onde se encontra o incêndio foi afetada, mas também as que se encontram próximas dela. O que acontece é que a intensidade do incêndio afeta as arestas à sua volta em dois níveis da seguinte forma: todas as arestas ligadas a nós que estejam na área laranja são afetadas com 100% da intensidade do fogo, ao passo que as arestas ligadas aos nós que se encontrem dentro da área laranja são afetadas com 50% da intensidade do fogo.

Tomando o exemplo da figura podemos observar que existe um incêndio de intensidade 100 entre os nós 4 e 6. As arestas conectadas a estes dois nós (únicos nós dentro da área a laranja) são afetadas com 100% da intensidade do incêndio, ou seja, as arestas 4-2, 4-6, 4-5 e 6-7. Já dentro da área amarela apenas temos o nó 5, isto quer dizer que as arestas 5-7 e 5-3 são afetadas em 50% da intensidade.

Calculando o caminho entre os nós 7 e 1, sem o incêndio temos a rota 7 -> 5 -> 4 -> 2 -> 1, ao passo que com incêndio temos a rota 7 -> 5 -> 3 -> 2 -> 1.

4.3 Comunicação com a central de detecção de incêndio

A comunicação com a central de detecção de incêndio é feita através do protocolo Modbus TCP. Numa fase inicial, esta comunicação ocorre com o objetivo de informar o programa sobre quais os dispositivos que se encontram ativos.

Geralmente, numa central de deteção de incêndio, os dispositivos estão conectados em laços. Um laço é um circuito fechado entre a central e vários dispositivos ligados em série. Mediante a sua complexidade, as centrais de deteção de incêndio podem suportar o uso de vários laços, agrupando, assim, vários conjuntos de dispositivos em circuitos independentes.

Inicialmente, o programa pede informação sobre quais os laços que estão ativos e depois quais os detetores que estão conectados dentro dos laços ativos. Depois, a cada 5 segundos, o programa comunica com a central a fim de receber, novamente, o estado dos dispositivos. Caso um dispositivo se encontre em alarme, o programa adicionará um fogo na localização do dispositivo.

A biblioteca de Python usada para esta comunicação foi a **pyModbusTCP**. Para implementação, criou-se um objeto `MB_Client` responsável pela criação de um cliente Modbus. Esta classe contém inúmeros métodos que permitem iniciar e encerrar a conexão com o servidor de Modbus, neste caso a central de deteção de incêndio e ler e escrever *coils* e *holding registers*.

Para além da classe `MB_Client`, como descrito em cima, foram também desenvolvidas funções para ler quais os dispositivos ativos conectados à central e qual o seu estado. Para isso foi consultado o manual de utilização de Modbus da NURIA [20] com o objetivo de descobrir quais os registos e *coils* a ler por forma a obter a informação desejada. Todos os registos acedidos foram feitos com recurso às tabelas presentes em A.

Assim, a primeira função implementada tem como objetivo perceber quais os laços da central de incêndio que se encontram habilitados. Para isso, foram utilizados os registos presentes na Tabela A.1.

De seguida, de modo a saber quais os dispositivos ativos em cada laço, foi implementada uma função que faz um ciclo de procura de todos os possíveis dispositivos conectados nos laços ativos. De modo a aceder aos registos destes dispositivos na central NURIA, é necessário proceder ao cálculo dos mesmos. Para isso, utilizaram-se as funções presentes em A.2 e na Tabela A.3 podemos observar exemplos dos registos percorridos.

Por último, depois de saber quais os dispositivos conectados, é feita uma nova leitura de modo a ler o estado dos dispositivos conectados, isto é, saber se estão em alarme ou não. Neste caso, de modo a calcular os registos em questão, usaram-se as funções presentes em A.4 e na Tabela A.5 podemos observar exemplos dos registos percorridos.

4.4 Implementação da Interface Gráfica

Nesta secção é apresentada a implementação da interface gráfica do SAD3E desenvolvido. Para o seu desenvolvimento, tal como explicado no Capítulo 2, foi utilizado o **DearPyGui**(DPG).

Inicialmente, é explicada a estrutura e o funcionamento das aplicações desenvolvidas com DPG. De seguida, fala-se da estrutura da interface gráfica desenvolvida para o SAD3E bem como do seu modo de funcionamento.

4.4.1 Estrutura de aplicações de DPG

As aplicações de DPG seguem uma determinada estrutura global. Todas elas devem ser capazes de: Criar e Destruir o Contexto; Criar e Exibir o *Viewport* e Configurar e inicializar o DearPyGui.

Ao criar o contexto, a aplicação pode aceder aos vários comando da biblioteca de DPG. Assim, isto deve ser o primeiro passo ao correr a aplicação uma vez que sem ele, a aplicação não irá funcionar.

O *Viewport* é a janela criada pelo sistema operativo. Esta janela é diferente mediante o sistema operativo. Na Figura 4.5, podemos ver as diferenças entre o *Viewport* no sistema operativo MacOS e no sistema operativo Windows.

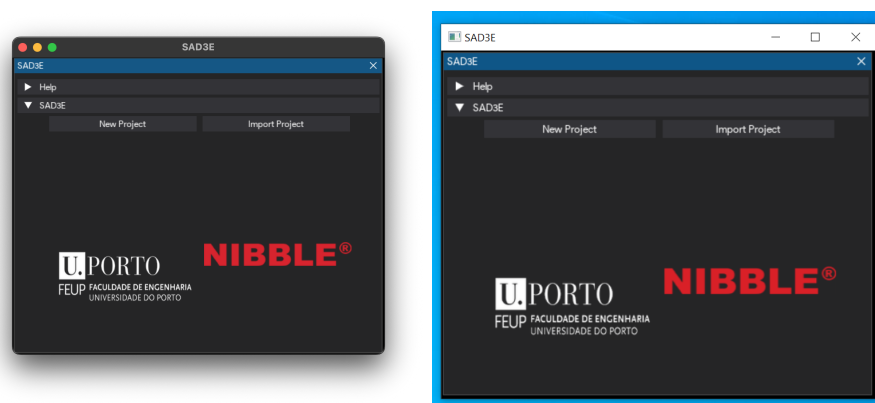


Figura 4.5: Janela inicial: (Esq) em Windows (Dir) MacOS.

Enquanto a aplicação está ativa, o ciclo de renderização é responsável por exibir os itens que nela existem, manter o estado da aplicação e ainda trata da gestão dos `callbacks`. Um `callback` é a função chamada quando o utilizador interage com o item em questão.

O DPG permite que uma janela seja definida como Janela principal. Esta Janela principal aparece no *Viewport* e é sempre renderizada atrás de todas as outras janelas, ou seja, sempre que durante o funcionamento da aplicação se altera o estado de uma janela para que esta esteja visível, ela vai aparecer por cima da janela principal, mesmo tendo sido instanciada primeiro no código fonte da aplicação.

Na sua biblioteca, o DPG conta com vários itens que podem ser organizados em três conjuntos, no entanto estes não são mutuamente exclusivos.

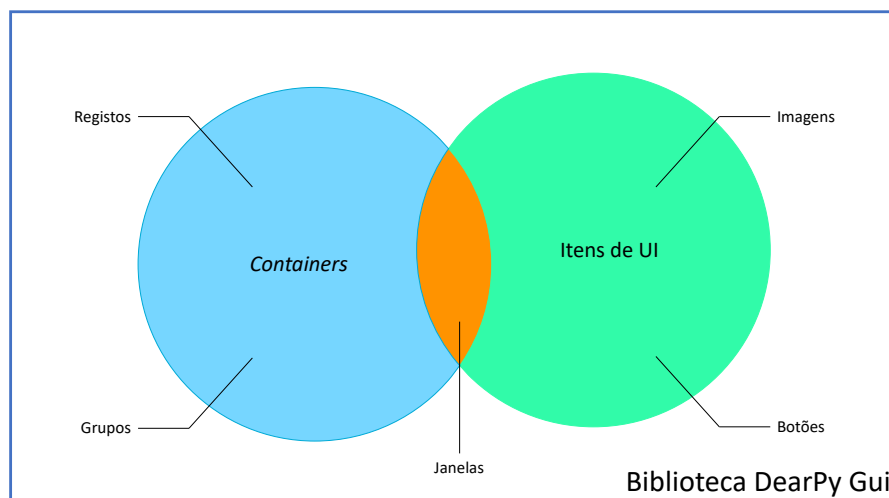


Figura 4.6: Itens da biblioteca DPG.

Os *Containers* são itens que podem conter outros itens ligados a si e os *Itens de UI* são todos os itens que podem ter componentes visuais.

4.4.2 Estrutura e Funcionamento da Interface Gráfica

A interface gráfica desenvolvida apresenta três janelas principais. Uma janela inicial de boas-vindas, uma janela de criação de um novo projeto e ainda uma janela responsável pela importação de um projeto.

Neste sistema, um projeto consiste num diretório ou pasta que contém uma planta de um edifício e o grafo representativo correspondente a essa mesma planta. A aplicação apenas funciona caso os nomes dos ficheiros não sejam alterados, isto é, num diretório de projeto os ficheiros lidos e utilizados pela aplicação serão sempre os ficheiros `graph.json` e `plant.(jpg/jpeg/png)`.

4.4.3 Janela inicial

A janela inicial tem como principal objetivo apresentar a aplicação ao utilizador. Apresenta os logótipos da FEUP e da NIBBLE® e ainda um separador de ajuda que explicará qual o objetivo da aplicação e o seu funcionamento. Apresenta também dois botões que irão abrir as janelas de Criação de Projeto e Visualização de Projeto.

Na Figura 4.7, podemos observar a janela inicial da aplicação.

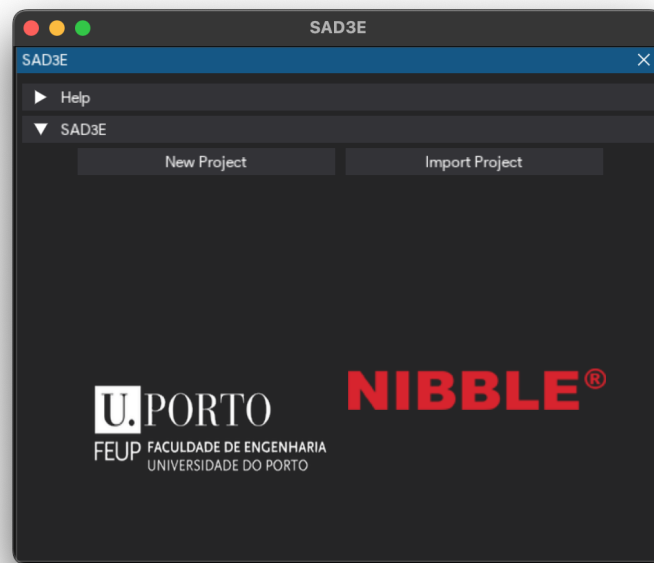


Figura 4.7: Janela inicial da GUI do SAD3E.

Os elementos de DPG, visíveis na Figura 4.7, são dois cabeçalhos colapsáveis, dois botões e duas imagens. No excerto de código 4.4, podemos observar um excerto de código que geraria uma janela semelhante.

```
1 with dpg.window(label="SAD3E", width=500, height=400):
2     with dpg.collapsing_header(label="Help"):
3         dpg.add_text(help_text)
4     with dpg.collapsing_header(label="SAD3E"):
5         with dpg.group(horizontal=True):
6             dpg.add_button(label="New Project",
7                             callback=open_file_dialog)
8             dpg.add_button(label="Import Project",
9                             callback=open_file_dialog)
10        with dpg.group(horizontal=True):
11            dpg.add_image("feup_logo")
12            dpg.add_image("nibble_logo")
```

Excerto de código 4.4: Código fonte da Janela de inicial do SAD3E.

Aqui podemos ver que a programação com DPG é bastante simples. Temos a entrada `with dpg.window()` : que cria a janela propriamente dita e que funciona como *Container* para os restantes elementos. Depois temos os botões e os logótipos que fazem parte de dois grupos distintos. Estes grupos estão, por sua vez, inseridos no *Container* do cabeçalho colapsável.

Na configuração dos botões, podemos observar o argumento `callback`. No caso dos botões o `callback` é responsável pela abertura de uma caixa de diálogo para seleção de ficheiros. No caso do botão `New Project` é pedida uma imagem de uma planta e no caso do botão `Import Project`, um diretório com um projeto já existente. Depois desta seleção são abertas as janelas de Criação ou Visualização de Projeto e é passado como argumento o caminho dos ficheiros selecionados

4.4.4 Janela de Criação de Projeto

Depois de escolhida a planta do edifício, o programa abrirá a janela de criação de projeto. Na Figura 4.8 podemos observar esta mesma janela.



Figura 4.8: Janela de Criação de Projeto do SAD3E com planta do piso 0 do Bloco B do edifício da FEUP.

Aqui podemos observar um exemplo da janela de Criação de Projeto, onde foi carregada uma planta de evacuação do piso 0 Bloco B do edifício da FEUP. Na parte superior da janela podemos também ver uma barra de menus com três itens `Nodes`, `Edges` e `Graph`.

4.4.4.1 O menu `Nodes`

O menu `Nodes` permite ao utilizador desenhar os nós do grafo ou remover nós já adicionados. O modo de adição de nós funciona utilizando o botão direito do rato. Após clicar no local escolhido, uma janela aparece pedindo ao utilizador que escolha o nome do nó. Caso o utilizador insira um nome já existente, um pequeno texto com um aviso aparece na janela como pode ser visto na Figura 4.9.

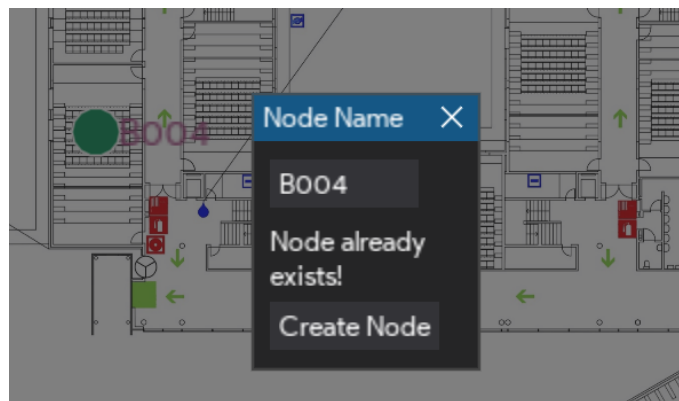


Figura 4.9: Exemplo de tentativa de criação de um nó com um nome já existente.

4.4.4.2 O menu Edges

O menu *Edges* permite ao utilizador desenhar as arestas do grafo e assim decidir quais os caminhos que podem ser percorridos pelos ocupantes do edifício. De notar que o programa não tem qualquer capacidade de interpretação da planta e por isso não pode impedir o utilizador de adicionar arestas em locais que podem não ser acessíveis aos ocupantes. Isto é, caso o utilizador insira uma ligação que atravessasse uma parede, o programa não terá em consideração se isso seria um caminho válido no mundo real. Após clicar no menu, uma janela aparece permitindo ao utilizador desenhar as arestas.

Na Figura 4.10 podemos observar o processo de criação de um aresta.

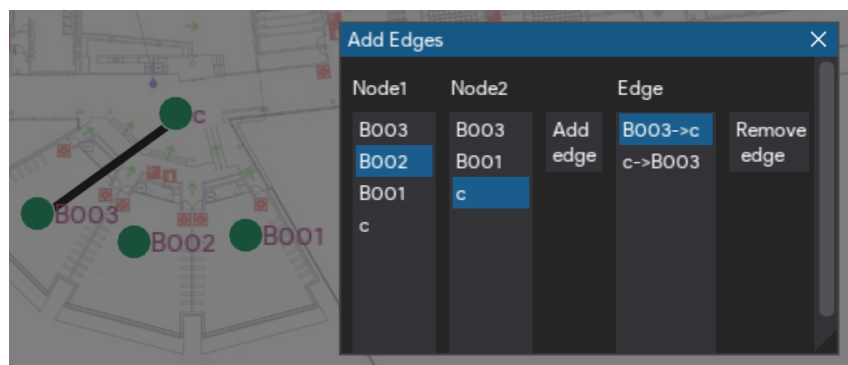


Figura 4.10: Exemplo de criação de arestas entre nós

Neste caso, a aresta entre o nó **B003** e o nó **c** já está criada. Podemos observar que os nós **B002** e **c** estão seleccionados. Caso o utilizador clicasse no botão *Add edge*, a lista de arestas presente no lado direito da janela iria ser atualizada com a nova aresta. De notar que existe também um botão para a remoção de arestas.

4.4.4.3 O menu Graph

Por último, temos o menu `Graph` cuja função é permitir ao utilizador imprimir as características do grafo na consola ou exportar o projeto para um diretório à escolha. A função de imprimir o grafo gera um resultado semelhante ao que podemos observar na Figura 4.11.

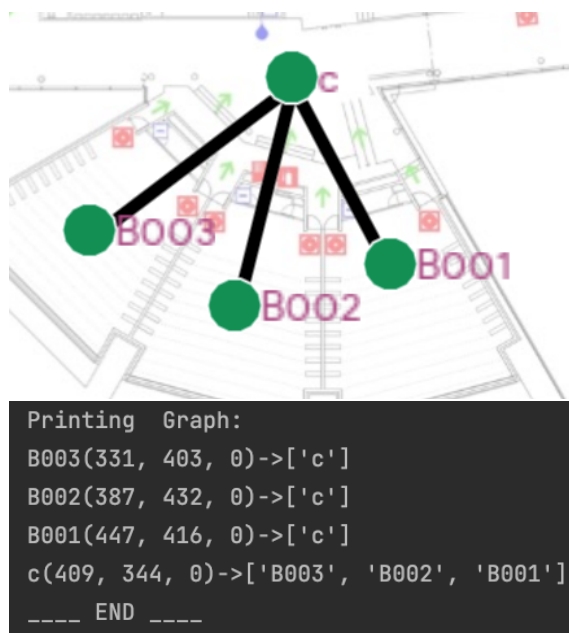


Figura 4.11: (Em cima) Exemplo de grafo (Em baixo) Output da função de impressão do grafo no terminal.

Ao pedir para exportar, o utilizador pode escolher o nome que quer dar ao projeto. Depois é criado um novo diretório com esse nome contendo o ficheiro do grafo e a imagem da planta.

4.4.5 Janela de Visualização

Depois de escolhido o diretório que contém a planta do edifício e o grafo já construído, o programa abrirá a janela de Visualização. É nesta janela que irá correr o SAD3E de forma integrada, ou seja, é nesta janela que ocorre a interação entre o Algoritmo de procura de caminho mínimo, a central de deteção de incêndio e o utilizador.

Logo após a escolha do projeto a importar, uma janela semelhante à observada na Figura 4.8 irá abrir. No entanto, a barra de menus é diferente e apresenta outras funcionalidades que são necessárias para a execução do SAD3E.

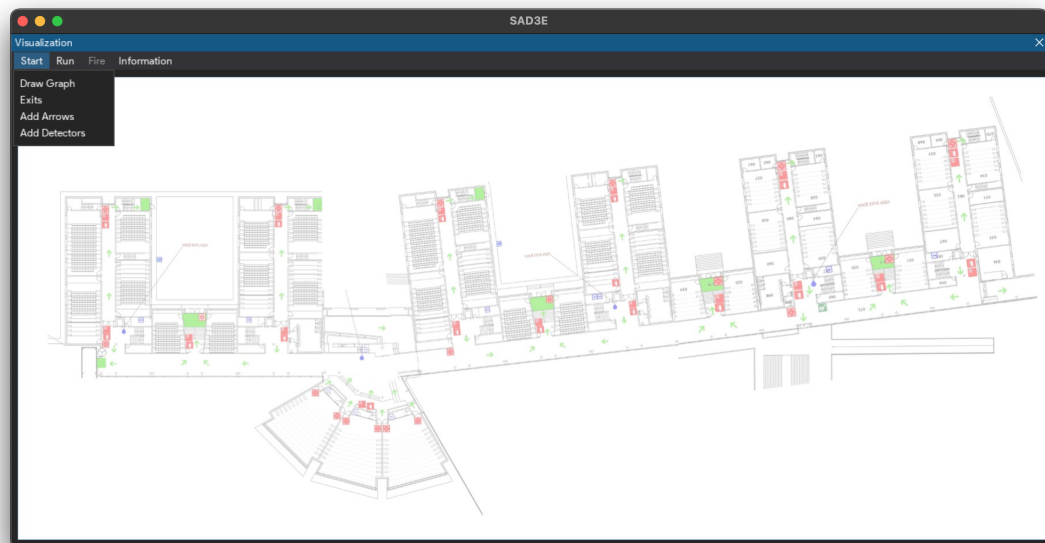


Figura 4.12: Janela de Visualização.

4.4.5.1 O menu Start

Como podemos observar na Figura 4.12, dentro do menu Start temos várias opções para inicializar o SAD3E. A primeira opção, Draw Graph, permite, tal como o nome indica, efetuar o desenho do grafo, podemos observar um exemplo na Figura 4.13.

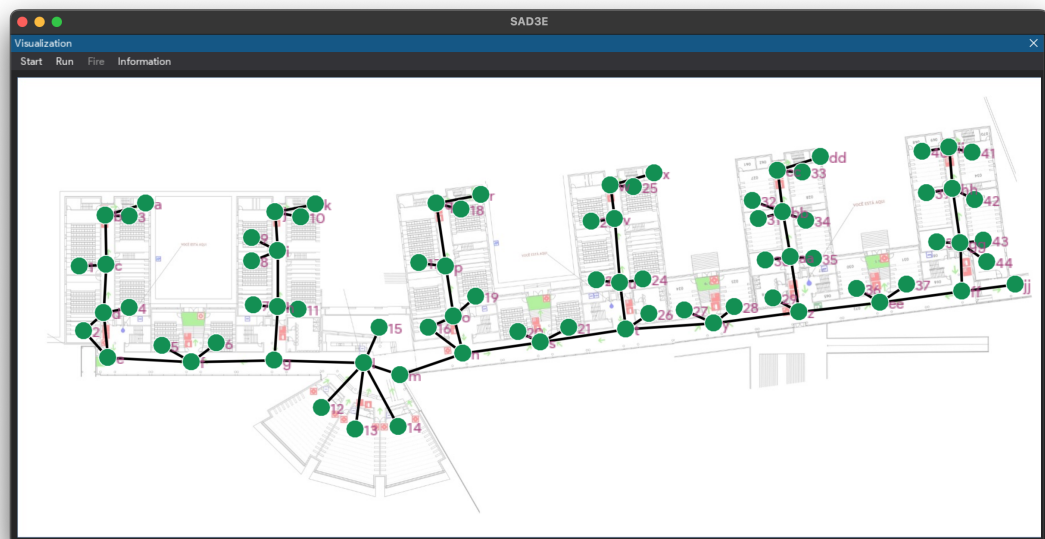


Figura 4.13: Janela de Visualização com grafo desenhado.

Depois, a opção `Exits`, permite ao utilizador escolher, de entre todos os nós do grafo, quais os que representam as saídas do edifício - ao fazê-lo, a cor dos nós escolhidos como saída alterar-se-á para vermelho como podemos observar na Figura 4.14.

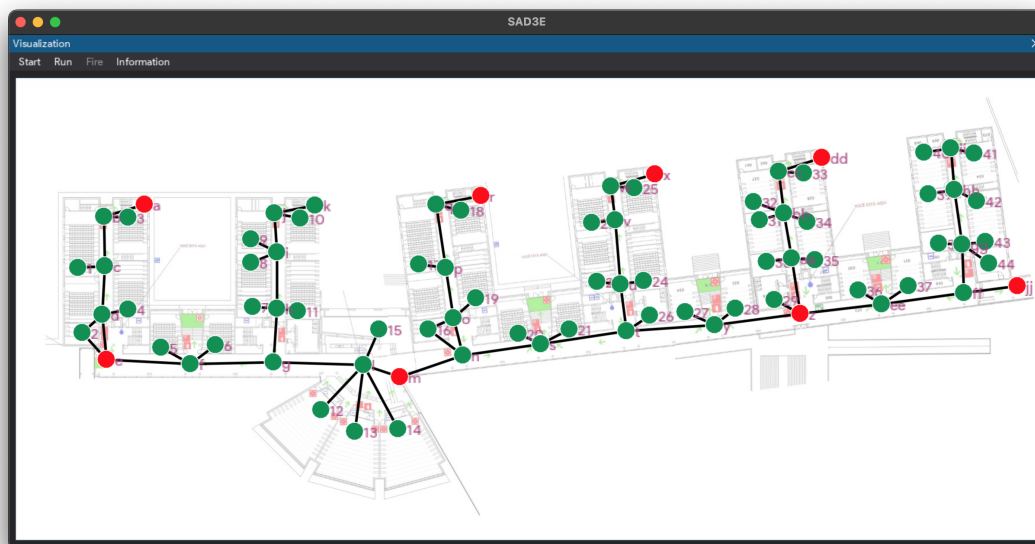


Figura 4.14: Janela de Visualização após seleção das saídas do edifício (a vermelho)

Para efeitos de simulação e uma vez que não foi possível executar testes com luminárias dinâmicas de evacuação, foi criado um método que permite a sua simulação. Assim, a opção `Add Arrows` permite a adição de setas verdes que simulam as luminárias dinâmicas. Estas setas alteram a sua direção tendo em consideração o melhor caminho a ser seguido pelos ocupantes do edifício em caso de evacuação. Podemos observar na Figura 4.15 o exemplo destas "luminárias virtuais" numa situação de repouso em que não existe nenhum fogo ativo no edifício.

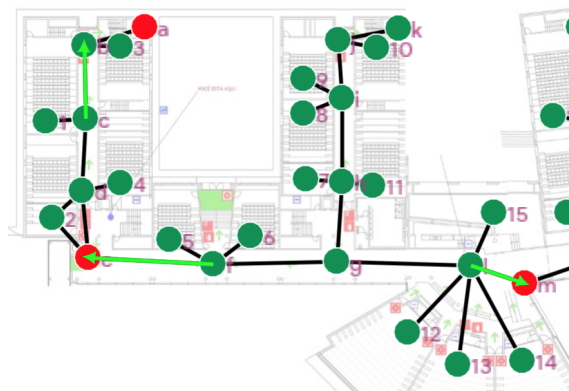


Figura 4.15: Exemplo das luminárias dinâmicas virtuais dos nós **c**, **f** e **l** numa situação de repouso

Por último, a opção `Add Detectors` permite fazer a conexão à central de detecção e informa o utilizador dos detetores que estão ligados. Posteriormente, o utilizador pode assimilar os detetores a arestas do grafo. Aí, o programa inicia o ciclo de leitura do estado dos dispositivos, ou seja, a cada dez segundos o programa comunica com a central e lê o estado de todos os dispositivos a ela conectados, tal como explicado em 4.3.

4.4.5.2 O menu `Run`

O menu `Run` apenas tem duas opções, uma para iniciar a simulação e outra para terminar a simulação. Estas opções apenas ficam ativas após a escolha das saídas por parte do utilizador.

4.4.5.3 O menu `Fire`

O menu `Fire` capacita o utilizador para adicionar ou remover fogos virtuais do sistema. Esta funcionalidade pode ser útil para perceber a resposta do mesmo a diversos cenários. Este menu só fica ativo após o início da simulação.

Para adicionar um fogo, uma pequena janela é aberta. Nesta janela o utilizador pode escolher entre que nós deseja adicionar o fogo e qual a sua intensidade. Esta pequena janela pode ser vista na Figura 4.16.

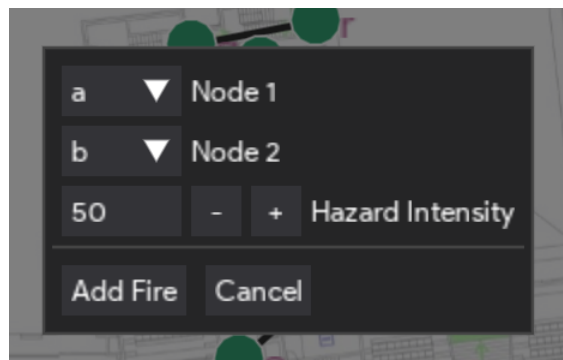


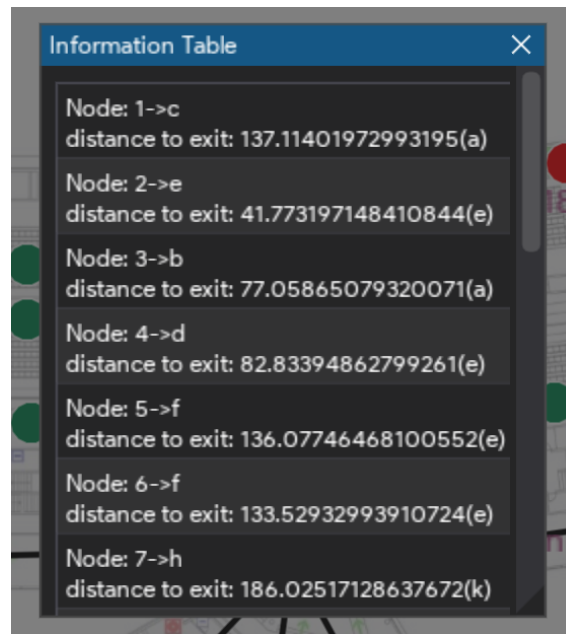
Figura 4.16: Janela de adição de fogo

Ao clicar na opção de remoção de fogo, uma lista colapsável aparece. Aqui estão listados todos os fogos ativos, bem como a sua localização, isto é, em que aresta se encontra, e a sua intensidade. Após a seleção do fogo a remover, um botão `Remove Fire` aparece e clicando nele, o fogo é extinguido e a sua área de afetação também é normalizada.

4.4.5.4 O menu `Information`

O menu `Information` contém uma ligação que abre uma tabela com informações sobre os caminhos a seguir no grafo. Esta tabela apresenta, para todos os nós do grafo, qual a direção a seguir, indicando o nó para o qual os ocupantes se devem deslocar; qual a saída destacada para aquela rota e ainda qual a distância até à saída. Na Figura 4.17 podemos observar uma parte da

tabela de informações relativa ao grafo apresentado em 4.13 quando o sistema está em repouso, ou seja, sem qualquer incêndio ativo.



The image shows a software window titled "Information Table" with a close button (X) in the top right corner. The window contains a list of seven nodes, each with a target exit point and a numerical distance value. The background of the window is dark grey, and the text is white. The nodes are listed vertically, separated by thin horizontal lines. The first node is "Node: 1->c" with a distance of "137.11401972993195(a)". The second is "Node: 2->e" with "41.773197148410844(e)". The third is "Node: 3->b" with "77.05865079320071(a)". The fourth is "Node: 4->d" with "82.83394862799261(e)". The fifth is "Node: 5->f" with "136.07746468100552(e)". The sixth is "Node: 6->f" with "133.52932993910724(e)". The seventh is "Node: 7->h" with "186.02517128637672(k)".

Node	Target	Distance
Node: 1	c	137.11401972993195(a)
Node: 2	e	41.773197148410844(e)
Node: 3	b	77.05865079320071(a)
Node: 4	d	82.83394862799261(e)
Node: 5	f	136.07746468100552(e)
Node: 6	f	133.52932993910724(e)
Node: 7	h	186.02517128637672(k)

Figura 4.17: Tabela de informação

Na tabela da Figura 4.17, podemos observar, por exemplo, na primeira linha que caso um ocupante do edifício estivesse nas imediações do nó **1** se deveria deslocar para o nó **c**. Também podemos ver a distância do nó **1** até à saída **a** que é a saída mais próxima.

Capítulo 5

Testes e funcionamento do sistema

Neste capítulo são apresentados testes ao sistema para demonstrar o seu funcionamento. Inicialmente, irá ser testado o sistema inserindo um incêndio manualmente pelo utilizador por forma a observar a resposta do sistema. Depois, irá ser testada a ligação à central e acionamento de um dos detetores de incêndio. Por último, serão adicionados múltiplos incêndios ao edifício.

5.1 Incêndio introduzido pelo utilizador

Aqui é descrito o processo de inserção de um incêndio manualmente por parte do utilizador. Todos os passos são ilustrados por figuras.

Depois de importado o projeto, neste caso o edifício em estudo é o piso 0 do bloco B do edifício da FEUP, foi desenhado o grafo e adicionado as respetivas saídas.

Depois de adicionadas as saídas, proceder-se-á à escolha dos locais onde estarão as luminárias de emergência dinâmicas. Neste caso, irão ser colocadas em todos os corredores, bem como nas interseções dos mesmos. Podemos ver na Figura 5.1 que foram colocadas luminárias nos nós **c, f, g, i, l, n, p, s, t, v, y, bb, ee, ff e hh**.

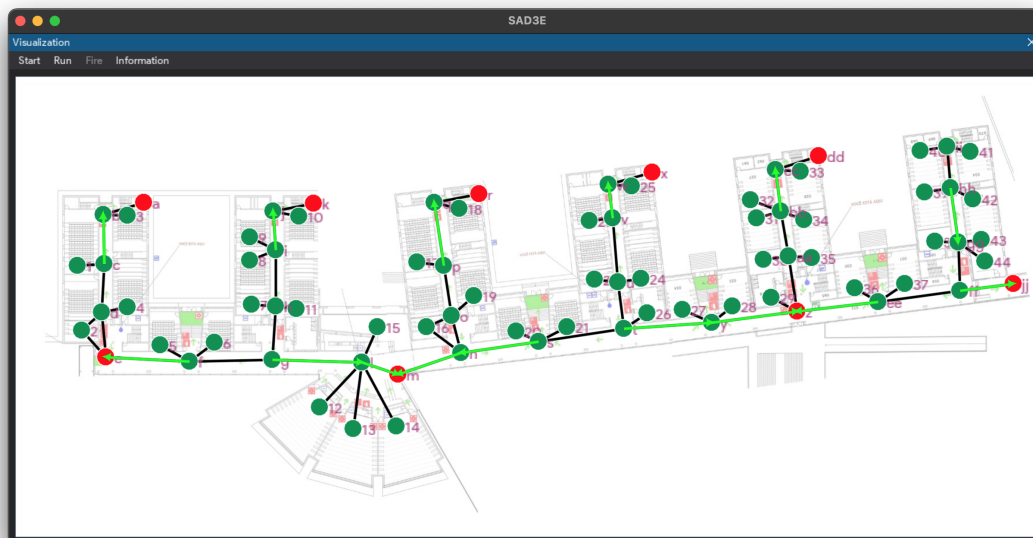


Figura 5.1: Teste ao sistema: colocação das luminárias virtuais.

De seguida podemos observar na Figura 5.2 em pormenor a parte direita do edifício. Foi adicionado um incêndio entre os nós **d** e **e** de intensidade **50**.

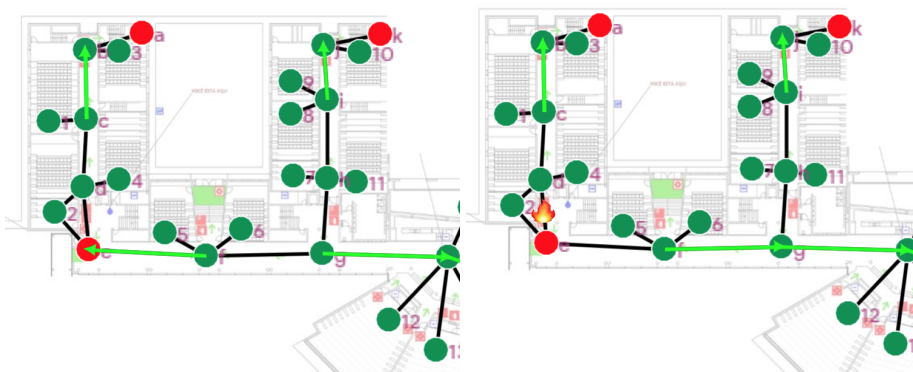


Figura 5.2: Teste ao sistema: Visualização de resposta ao sistema com adição de incêndio.

De notar que a luminária presente no nó **f** alterou o seu estado, passando a dirigir os ocupantes para a saída **m** ao invés de os direccionar para a saída **e** que se encontra mais próxima do incêndio.

5.2 Resposta a estímulo da central de detecção de incêndio

Os testes executados com a central de incêndio endereçável NURIA foram feitos utilizando dois sensores de detecção de incêndio conectados ao laço **1** da central. Na Figura 5.3, podemos ver

uma imagem dos detetores utilizados bem como de um spray usado para simular fumo por forma a ser detetado pelos mesmos. Na Figura 5.4, podemos ver a central NURIA usada.



Figura 5.3: Detetores usados para teste do sistema.



Figura 5.4: Central endereçável de detecção de incêndio NURIA.

Um dos requisitos do sistema é que o tempo de resposta a um alarme seja inferior a 5 segundos, na maior parte das vezes isto acontece, no entanto, por causa da comunicação com a central, estes tempos podem ser bastante elevados. Pelos testes executados e pelo *feedback* oferecido

pela NIBBLE®, o problema não se encontra no SAD3E, mas sim na própria central que é muito inconstante a responder a pedidos de informação.

Assim, o sistema é capaz de comunicar com a central, sabendo quais os dispositivos que estão ligados e qual o seu estado. Para além disso, consegue responder aos estímulos de alarme e fazer o cálculo da rota tendo em conta esses estímulos.

Na Figura 5.5, podemos observar o processo de comunicação com a central. O sistema efetua a ligação e percorre todos os dispositivos disponíveis apresentando o laço em que se encontram e o seu número dentro do laço. Depois o utilizador pode atribuir cada dispositivo disponível a uma aresta. A partir do momento em que o utilizador inicia o programa no menu Run, inicia-se o ciclo de leitura como explicado anteriormente.

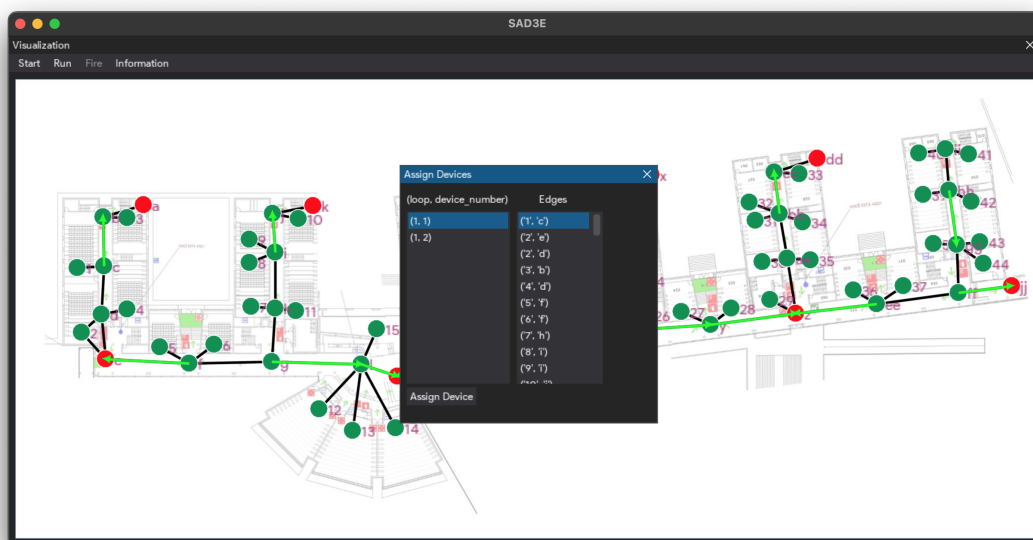


Figura 5.5: Teste ao sistema: colocação das luminárias virtuais.

5.3 Resposta do sistema a múltiplos incêndios

Neste teste, o sistema irá responder à inserção de múltiplos incêndios no mesmo edifício. Aqui, serão introduzidos incêndios nas arestas **d-e**, **l-15** e **y-27**.

Os resultados obtidos podem ser observados na Figura 5.6.

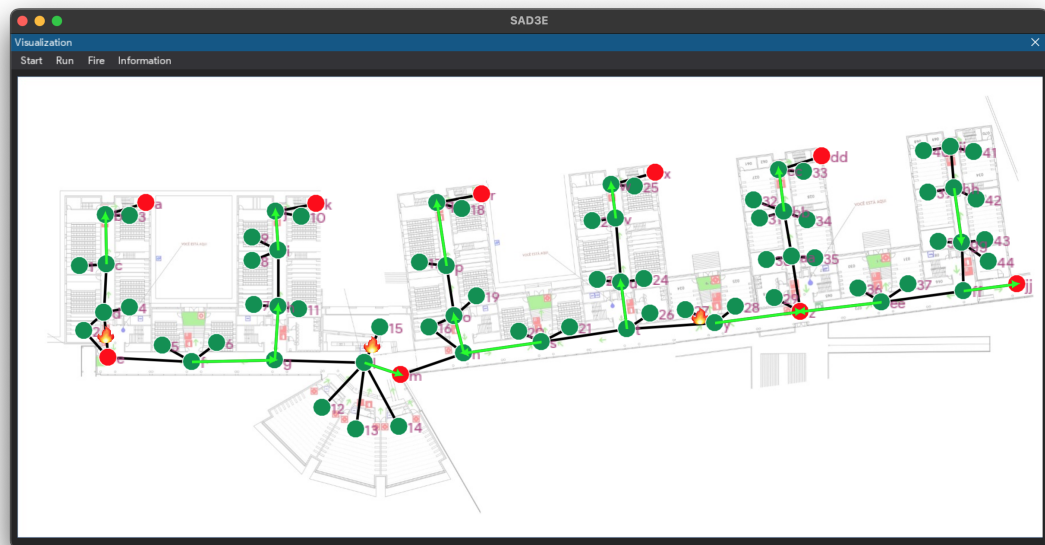


Figura 5.6: Teste ao sistema: Inserção de vários incêndios simultâneos.

Podemos comparar esta resposta com a resposta do sistema da Figura 5.2. A luminária do nó **g** altera o seu estado, redirecionando a rota para a saída **k** uma vez que existe, agora, um incêndio próximo da saída **m**.

Capítulo 6

Conclusões e Trabalhos Futuros

O principal objetivo desta dissertação passava pelo desenvolvimento de um SAD que, conectado a uma central endereçável de deteção de incêndio, seria capaz de atuar em edifícios, em situações de incêndio, fornecendo aos ocupantes informações para que estes tivessem uma evacuação mais segura.

O estudo inicial das diferentes abordagens ao problema da evacuação permitiu uma melhor compreensão do problema ajudando numa conceção inicial da arquitetura do sistema bem como na definição da sua estrutura.

Seguiu-se o início do desenvolvimento do sistema propriamente dito. Aqui, a maior dificuldade passou pela constante evolução da unidade base do sistema, o objeto Node (4.1) e os pequenos ajustes a todo o programa.

Por último, o grande desafio foi o desenvolvimento da interface gráfica. Até então, o contacto com ferramentas de desenvolvimento de interfaces gráficas tinha sido inexistente. Desenvolver uma GUI requer um pensamento quase de previsão, uma vez que o programador tem de pensar numa forma intuitiva para o utilizador interagir com o programa. Para além disso, uma GUI contém inúmeros elementos e gerir o seu aparecimento e interação no ecrã pode ser complexo.

No final, o sistema funcionou dentro do esperado, no entanto, ainda apresenta margem para melhoramento. Nas secções seguintes é feita a avaliação ao cumprimento dos objetivos e, por fim, serão expostos alguns pontos de melhoramento do SAD3E.

6.1 Satisfação dos Objetivos

No que diz respeito à satisfação dos objetivos, o SAD3E é bastante completo. O sistema é capaz de calcular automaticamente o melhor caminho de evacuação quando se inicia o sistema. Quando recebe informações da central de deteção de incêndio ou quando é introduzido um incêndio virtual por parte do utilizador, o SAD3E também é capaz de executar o cálculo da melhor rota e alterar, rapidamente as direções a fornecer aos ocupantes do edifício.

Quanto à demonstração do estado do sistema, isto é, apresentação ao utilizador dos locais onde existam incêndios ativos bem como a apresentação dos caminhos de evacuação a seguir, isto é

cumprido pela interface gráfica, quer através das setas que simulam as luminárias dinâmicas(4.15), quer através da tabela de informações (4.17).

Por último, a atuação em saídas para a indicação do melhor caminho a seguir não foi totalmente implementada, a classe `MB_Client` possui métodos de escrita de *coils* e *holding registers*, no entanto não foram aplicados nem testados uma vez que não existiam dispositivos desse tipo para ligar à central endereçável de deteção de incêndio.

No cômputo geral, os objetivos foram cumpridos e o sistema tem um funcionamento dentro do esperado.

Em termos pessoais, o desenvolvimento deste projeto foi muito frutífero na medida em que as dificuldades encontradas ajudaram a adquirir conhecimentos e competências até então pouco desenvolvidas. Outra mais-valia para a minha vida profissional foi o contacto com o ambiente empresarial deixando-me mais preparado para o futuro.

6.2 Trabalhos Futuros

Existem ainda inúmeras ferramentas e melhorias que podem ser adicionadas ao SAD3E.

Em termos de opções de customização da interface gráfica e de visualização, poderia ser implementada uma funcionalidade que permitisse que o utilizador alterasse a cor dos nós do grafo, das arestas, do nome dos nós, etc. Isto poderia melhorar a visualização dependendo da imagem da planta do edifício usada. O DPG tem, na sua vasta biblioteca de itens, um item de escolha de cores, `dearpygui.add_color_picker()` que poderia ser empregue neste caso.

Outra das funcionalidades que poderia ser adicionada era a de permitir ao utilizador escolher o raio de afetação do fogo na interface gráfica, uma vez que, neste momento, apenas pode ser alterado diretamente no código. Esta pode ser uma questão importante porque a morfologia dos edifícios pode variar bastante e pode ser relevante ter áreas de afetação diferentes mediante o tipo de edifício. O sistema ideal teria, para cada detetor de incêndio, um raio de afetação configurável. Assim, o processo de escolha de melhor caminho de evacuação iria ser melhorado e específico para cada situação.

Para abranger ainda mais as possibilidades de uso em qualquer edifício poderia ser adicionado suporte para edifícios com múltiplos andares. O algoritmo de procura da melhor rota está preparado para utilizar uma coordenada *z* referente ao andar do edifício, no entanto, a interface gráfica não está preparada para tal.

Uma das partes muito importantes para o sistema seria cumprir o objetivo de comunicar com a central de deteção de incêndio e atuar nas luminárias.

Por último, para que o produto pudesse ser disponibilizado no mercado deveriam ser feitos mais testes e melhorar a robustez do sistema como um todo. Ou seja, resolver alguns *bugs* da interface gráfica e garantir que o sistema funciona em quaisquer condições.

Anexo A

Tabelas de Registos Modbus da central NURIA

As seguintes tabelas foram retiradas do manual de Modbus da central de deteção de incêndio endereçável NURIA [20].

Tabela A.1: Registos das saídas digitais dos laços da central NURIA

Registo (dec)	Conteúdo	Resposta/Dados
910	Laço 1 habilitado/desabilitado	1 - Habilitado 0 - Desabilitado
942	Laço 2 habilitado/desabilitado	1 - Habilitado 0 - Desabilitado
974	Laço 3 habilitado/desabilitado	1 - Habilitado 0 - Desabilitado
1006	Laço 4 habilitado/desabilitado	1 - Habilitado 0 - Desabilitado

Tabela A.2: Funções para o cálculo do registo do dispositivo D em função do Laço

	Laço 1	Laço 2	Laço 3	Laço 4
Registo do dispositivo D	$10244+10*(D-1)$	$15324+10*(D-1)$	$20404+10*(D-1)$	$25484+10*(D-1)$

Tabela A.3: Excerto de registos das saídas digitais dos dispositivos

Registo (dec)	Conteúdo	Resposta/Dados
10244	Dispositivo 1 Laço 1 habilitado/desabilitado	1 - Habilitado 0 - Desabilitado
10254	Dispositivo 2 Laço 1 habilitado/desabilitado	1 - Habilitado 0 - Desabilitado
15324	Dispositivo 1 Laço 2 habilitado/desabilitado	1 - Habilitado 0 - Desabilitado
22934	Dispositivo 254 Laço 3 habilitado/desabilitado	1 - Habilitado 0 - Desabilitado

Tabela A.4: Funções para o cálculo do estado do alarme do dispositivo D em função do Laço

	Laço 1	Laço 2	Laço 3	Laço 4
Registo do dispositivo D	$10242+10*(D-1)$	$15322+10*(D-1)$	$20402+10*(D-1)$	$25482+10*(D-1)$

Tabela A.5: Excerto de registos das entradas analógicas dos dispositivos

Registo (dec)	Conteúdo	Resposta/Dados
10242	Estado do alarme do Dispositivo 1 Laço 1	0 - Falha 1 - Normal 2 - Pré-alarme 3 - Alarme 4 - Teste 5 - Desconhecido 6 - Ligado 7 - Desligado 8 - Falha em aberto 9 - Falha em curto-circuito
10252	Estado do alarme do Dispositivo 2 Laço 1	0 - Falha 1 - Normal 2 - Pré-alarme 3 - Alarme 4 - Teste 5 - Desconhecido 6 - Ligado 7 - Desligado 8 - Falha em aberto 9 - Falha em curto-circuito
15322	Estado do alarme do Dispositivo 1 Laço 2	0 - Falha 1 - Normal 2 - Pré-alarme 3 - Alarme 4 - Teste 5 - Desconhecido 6 - Ligado 7 - Desligado 8 - Falha em aberto 9 - Falha em curto-circuito
22932	Estado do alarme do Dispositivo 254 Laço 3	0 - Falha 1 - Normal 2 - Pré-alarme 3 - Alarme 4 - Teste 5 - Desconhecido 6 - Ligado 7 - Desligado 8 - Falha em aberto 9 - Falha em curto-circuito

Referências

- [1] Eaton Electric Ltd. Introduction to adaptive evacuation, 2020.
- [2] United Nations, Department of Economic and Social Affairs, Population Division. The world's cities in 2016 – data booklet, 2016.
- [3] NIBBLE - Engenharia, Lda. NURIA Datasheet, 2019.
- [4] Milad Haghani. Optimising crowd evacuations: Mathematical, architectural and behavioural approaches. *Safety Science*, 128, 8 2020. doi:10.1016/j.ssci.2020.104745.
- [5] Avgoustinos Filippoupolitis e Erol Gelenbe. A distributed decision support system for building evacuation. páginas 323–330, 2009. doi:10.1109/HSI.2009.5091000.
- [6] Zhixiang Fang, Xinlu Zong, Qingquan Li, Qiuping Li, e Shengwu Xiong. Hierarchical multi-objective evacuation routing in stadium using ant colony optimization approach. *Journal of Transport Geography*, 19:443–451, 5 2011. doi:10.1016/j.jtrangeo.2010.10.001.
- [7] Salvador Casadesús Pursals e Federico Garriga Garzón. Optimal building evacuation time considering evacuation routes. *European Journal of Operational Research*, 192:692–699, 1 2009. doi:10.1016/j.ejor.2007.10.004.
- [8] Arturo Cuesta, Orlando Abreu, Adriana Balboa, e Daniel Alvear. Real-time evacuation route selection methodology for complex buildings. *Fire Safety Journal*, 91:947–954, 7 2017. doi:10.1016/j.firesaf.2017.04.011.
- [9] Wikipedia. Graph (abstract data type) — Wikipedia, the free encyclopedia. [https://en.wikipedia.org/wiki/Graph_\(abstract_data_type\)](https://en.wikipedia.org/wiki/Graph_(abstract_data_type)), 2022. [Online; acedido a 13-abril-2022].
- [10] Xiao Zhu Wang. The comparison of three algorithms in shortest path issue. volume 1087. Institute of Physics Publishing, 10 2018. doi:10.1088/1742-6596/1087/2/022011.
- [11] Wikipedia. Big O notation — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Big_O_notation, 2022. [Online; acedido a 13-abril-2022].
- [12] Mark Allen Weiss. *Data Structures and Algorithm Analysis in C*. 2010.
- [13] Wikipedia. JSON — Wikipedia, the free encyclopedia. <https://pt.wikipedia.org/wiki/JSON>, 2022. [Online; acedido a 13-abril-2022].
- [14] Wikipedia. XML — Wikipedia, the free encyclopedia. <https://pt.wikipedia.org/wiki/XML>, 2022. [Online; acedido a 13-abril-2022].

- [15] Wikipedia. CSV — Wikipedia, the free encyclopedia. https://pt.wikipedia.org/wiki/Comma-separated_values, 2022. [Online; acedido a 7-junho-2022].
- [16] Wikipedia. Tkinter — Wikipedia, the free encyclopedia. <http://pt.wikipedia.org/w/index.php?title=Tkinter&oldid=63633740>, 2022. [Online; acedido a 22-junho-2022].
- [17] Qt for Python. https://wiki.qt.io/Qt_for_Python, 2022. [Online; acedido a 22-junho-2022].
- [18] Modbus Organization. Modbus protocol. <https://modbus.org/specs.php>, 2022. [Online; acedido a 7-junho-2022].
- [19] Eaton Electric Ltd. From static to adaptive escape routing. <https://www.eaton.com/gb/en-gb/catalog/emergency-lighting/central-battery-system-adaptive-evacuation.html>. [Online; acedido a 28-junho-2022].
- [20] NIBBLE - Engenharia, Lda. P17040075 - Modbus - Manual de utilização, 2021.