Universidade da Beira Interior
Departamento de Informática

Inteligência Artificial: *Projeto Prático*

Elaborado por:

41266 — Diogo Castanheira Simões
43464 — Cristiano Miguel Abrantes Santos

Docente:

Professor Doutor Luís Filipe Barbosa de Almeida Alexandre

22 de dezembro de 2021

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

IA Inteligência Artificial

UC Unidade Curricular

Capítulo 1

Introdução

O projeto apresentado foi desenvolvido no âmbito da Unidade Curricular (UC) de Inteligência Artificial (IA) como um projeto introdutório a esta área. Este Capítulo descreve os objetivos delineados para a sua implementação, bem como a organização do documento.

1.1 Objetivos

Este projeto prático tem por objetivo criar a inteligência de um *robot* que tem como o seu mundo virtual o piso de um supermercado, o qual percorre o mapa sob o controlo do utilizador e terá de ser capaz de responder às mesmas em qualquer momento da simulação, pelo que é necessário implementar funções que sejam capazes de obedecer a tais requisitos.

1.2 Organização do Documento

De modo a refletir o trabalho feito, este documento encontra-se estruturado da seguinte forma:

1. No primeiro capítulo — **Introdução** — é apresentado o projeto, o enquadramento do mesmo, a constituição do grupo de trabalho, a enumeração dos objetivos delineados para a conclusão do mesmo e a respetiva organização do documento.

2. No segundo capítulo — **Desenvolvimento e Implementação** — são apresentadas e descritas as escolhas, os algoritmos pensados e métodos utilizados na implementação dos mesmos, para responder às perguntas solicitadas no enunciado.

3. No terceiro capítulo — **Reflexão Crítica e Problemas Encontrados** — denota-se a divisão de tarefas pelos elementos do grupo expondo os pontos fortes, fracos, oportunidades e ameaças ao trabalho desenvolvido.

4. No quarto capítulo — **Conclusões e Trabalho Futuro** — apresenta-se uma reflexão do trabalho e conhecimentos adquiridos ao longo do desenvolvimento do projeto prático e um contrabalanço com a possibilidade de existirem objetivos não alcançados e que se podem explorar no futuro.

1.3 Ferramentas Utilizadas

São descritas em seguida as ferramentas e tecnologias utilizadas:

1. Biblioteca *NetworkX* [1] — biblioteca do *Python* [2] utilizada para criar e manipular as estruturas de dados utilizadas (grafos);

2. *GitHub* [3]– plataforma utilizada para a hospedagem do repositório contendo o código fonte do projeto referido neste documento;

3. *Python* [2] — linguagem de programação utilizada para o desenvolvimento do projeto.

4. *Visual Studio Code* [4] — editor de texto utilizado para a escrita do código-fonte do projeto;

Capítulo 2

Desenvolvimento e Implementação

2.1 Escolhas de Implementação

Antes de implementar os algoritmos para satisfazer as respostas às perguntas colocadas, foi necessário definir a estrutura de dados utilizada para a manipulação do mundo virtual e dos objetos nele contidos.

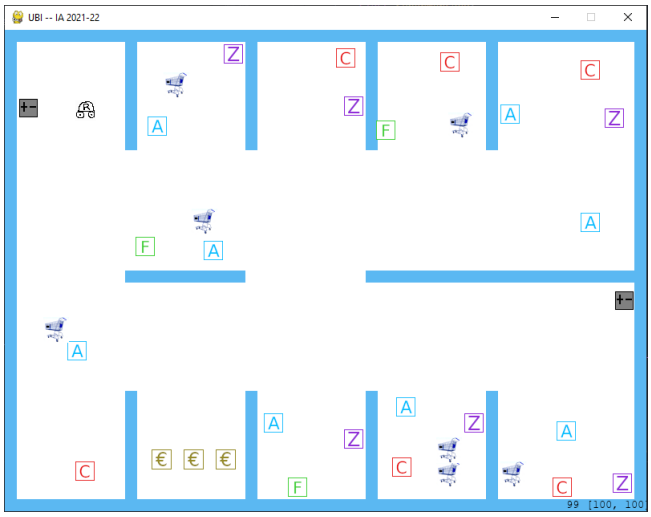

Figura 2.1: Mundo virtual original utilizado para a implementação do projeto.

Desta forma, além da escolha do uso de classes estáticas para gerir os objetos que o dispositivo robótico encontra no seu mundo virtual (Figura 2.3), a solução passou por usar grafos, os quais permitem representar de forma bastante eficiente os dados recolhidos. Para este fim, foi utilizada a biblioteca *NetworkX* [1] para a criação e gestão dos mesmos. Dois grafos foram utilizados neste âmbito:

1. Grafo *zoneMap* (Figura 2.2) — armazena informações sobre as zonas visitadas e a sua ligação.

2. Grafo *infoMap* — realiza o mapeamento do mundo e vai aramzenado todos os objetos contidos nas diferentes zonas

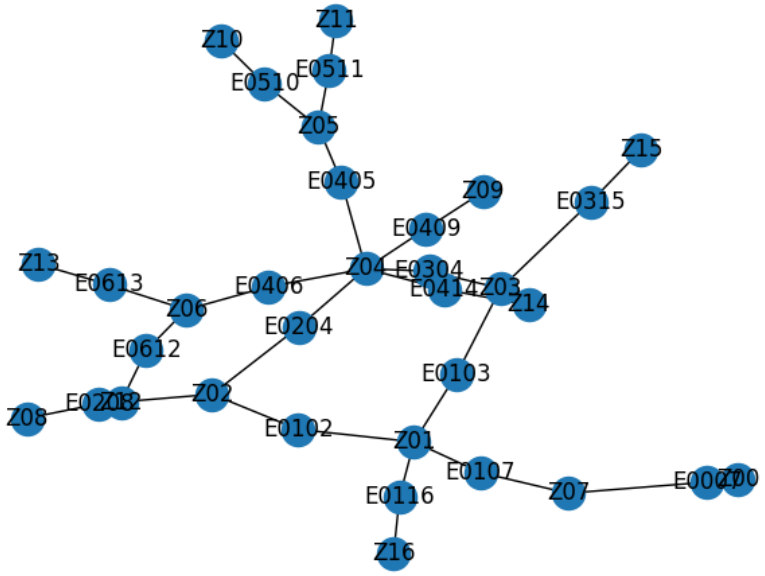

Figura 2.2: Grafo *zoneMap*: cada nodo representa uma zona visitada pelo *robot* (codificado com `Z`) e uma entry que conecta duas divisões (codificado com `E`). Este grafo representa o mapa total original fornecido para o projeto.

Por análise do grafo da Figura 2.2, podemos constatar que se trata de um grafo não dirigido, composto por nodos e arestas, sendo que os nodos guardam informações acerca das zonas. Os objetos contidos nessas mesmas zonas são guardados no grafo *infoMap*.

Em termos de organização do código-fonte do trabalho prático, optou-se por se utilizar classes, as quais se enumeram seguidamente:

1. *Zone* — Classe usada para armazenar e manipular dados de Zona.

2. *Object* — armazena os objectos encontrados pelo *robot*, sem repetição em listas, tratando diretamente da resposta à pergunta 1 (subsecção 2.2.1) e auxiliando as restantes classes a gerir os seus dados;

3. *Robot* — realiza a gestão dos dados inerentes ao *robot*, em particular a velocidade e a bateria, bem como a sua relação com o tempo;

4. *Environment* — principal classe do programa na qual a informação relativa ao supermercado é atualizada conforme as informações dadas pelo *robot*;

5. *Utils* — coleta um conjunto de funções auxiliares, como por exemplo a *pathDescription()* e a *timeToStr()* necessárias para as respostas às perguntas 3 e 5 respetivamente. No fundo contém funções de troca de variáveis e descrição de texto;

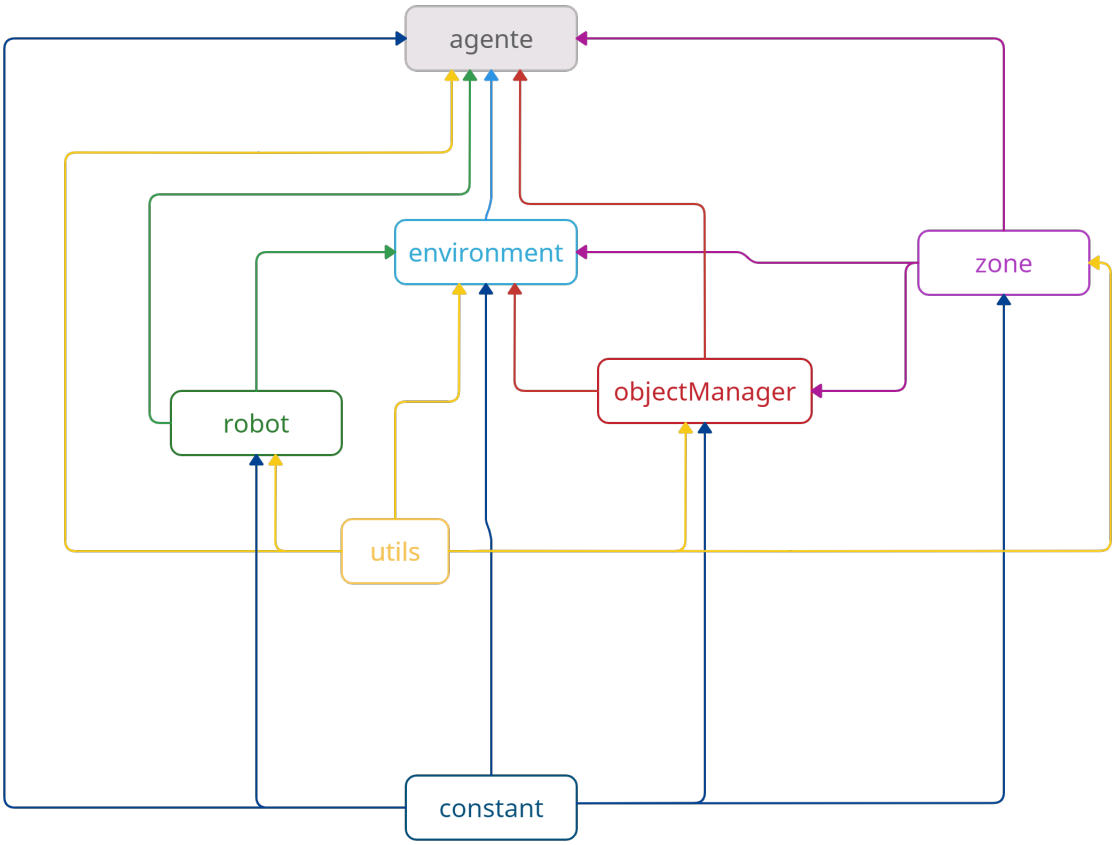

Figura 2.3: Organizaçao dos ficheiros do projeto e consequentemente, das respetivas classes.

2.2 Detalhes de Implementação

2.2.1 Pergunta 1

Enunciado: *Qual foi a penúltima pessoa do sexo feminino que viste?*

A classe *Objects* fornece o método `getPenultSawFemale()`, que devolve o nome da penúltima pessoa que o *robot* encontrou. Este método depende da prévia invocação da função `add()` por parte da classe `Objects`, encarregue de atualizar corretamente as pessoas encontradas.

A verificação do sexo é realizada através do método `sexCheck()`, que recebe como *input* o nome da pessoa observada pelo *robot*, e procura esse nome na base de dados encontrada em [5], contida no ficheiro `res/nomesfeminino.up.csv`. Se o nome estiver contido no ficheiro, a função retorna a *string* `'female'`, caso contrário retorna `'male'`. Apesar de todos os nomes femininos presentes no mundo virtual fornecido terminarem com a letra 'a', foi optada uma solução mais completa de modo a diminuir a probabilidade de falsos negativos.

Caso o *robot* não tenha encontrado pelo menos duas pessoas no momento da pergunta, é devolvida uma mensagem pré-definida de erro pela seguinte constante:

```
1 ERROR_NOT_ENOUGH_FEMALES = "Não foram avistadas pelo
  ↪  menos 2 pessoas do sexo feminino até ao momento"
```

2.2.2 Pergunta 2

Enunciado: *Em que tipo de zona estás agora?*

A resposta a esta pergunta é dada pela função `resp2()` do ficheiro *agente.py*, utilizando a função `getCurrentZone()` da classe *Environment* e a função `getType()` da classe *Zone*. É assim devolvido um código que indica o tipo da sala onde o *robot* se encontra no momento. Estas encontram-se definidas da seguinte forma:

```
1 @staticmethod
2 def getCurrentZone() -> Zone:
3     return Enviroment._zones[Enviroment._currentZone]
```

```
1 @staticmethod
2 def getType(self) -> str:
3     if not self._zoneType:
4         raise Zone.ZoneNotDefinedException(
          ↪  Zone.ERROR_ZONE_NOT_DEFINED)
5     return self._zoneType
```

Por fim, a resposta é dada por:

```
Enviroment.getCurrentZone().getType()}
```

De notar os seguintes aspetos:

1. Se o número da zona está no intervalo $[1, 6]$, então devolve o código correspondente a um corredor;
2. Se o número da zona for o 7, então é devolvido o código correspondente à entrada do supermercado.
3. A zona das caixas corresponde também à saída do supermercado, pelo que quando solicitada essa zona, aparece no ecrã a String: "SAIDA".

2.2.3 Pergunta 3

Enunciado: *Qual o caminho para a papelaria?*

Para esta pergunta, o algoritmo implementado retorna uma lista com o nome dos nodos do grafo `zoneMap` correspondentes ao caminho mais curto desde a posição atual do *robot* até à papelaria.

Para tal, é adicionado ao grafo `zoneMap` um nodo representativo do *robot* e criadas as respetivas arestas para todos os seus vizinhos. De seguida, o algoritmo A^* é aplicado sobre o grafo recorrendo ao método `astar_path` da biblioteca *NetworkX*, tendo como retorno uma lista de nodos contendo o caminho mais curto desde do nodo *robot* até ao nodo papelaria.

Por fim, o método `pathDescription()` da classe `Utils` formata a lista de nodos para texto corrente de modo a facilitar a leitura do utilizador.

2.2.4 Pergunta 4

Enunciado: *Qual a distância até ao talho?*

Para esta questão, apesar de não ter sido pedido o caminho, optou-se por se procurar o talho seguindo o *path* do grafo. A função `resp4()` do ficheiro *agente.py* contém o algoritmo delineado para a resposta.

À semelhança da pergunta 3, o *robot* é temporariamente adicionado ao grafo *zoneMap*. O algoritmo A^* é aplicado sobre o grafo recorrendo ao método `astar_path_length` da biblioteca *NetworkX* para a zona do talho, se a mesma já tiver sido visitada pelo *robot*. Consultando a documentação da biblioteca acima referida, sabemos que o `astar_path_length` retorna o comprimento do caminho

mais curto entre o nodo inicial (*robot*) e o nodo final (talho) usando o algoritmo A^*.

A informação é devolvida com o seguinte formato:
`Resposta : A Distância até ao talho = <distância>`.

2.2.5 Pergunta 5

> **Enunciado:** *Quanto tempo achas que demoras a ir de onde estás até à caixa?*

A resposta a esta pergunta assemelha-se à da pergunta 3 (secção 2.2.3), sendo tratada pela função `resp5()` do ficheiro *agente.py*. O *robot* é adicionado temporariamente ao grafo *zoneMap* e é determinado o caminho mais curto até às caixas.

Com base na distância determinada, é estimado o tempo que demorará a chegar às caixas com recurso ao método `predictTimeFromDistance()` da classe *Robot*. Esta classe calcula a cada nova posição do *robot* a variação da velocidade ao longo do tempo sob a forma de uma regressão linear a fim de se poder retirar o tempo necessário a percorrer uma certa distância.

Ora, a distância é a área sob a curva da função $v(t)$ (velocidade em função do tempo), a qual é determinada por **regressão linear**. Sendo esta "curva" uma reta temos que a área sob a curva é a área de um trapézio, tal que:

$$d = \frac{v_f + v_i}{2}\Delta t \tag{2.1}$$

Rearranjando a igualdade da expressão anterior, foi possível chegar à formula usada, sendo ela a seguinte:

$$\Delta t = \frac{2d}{v_f + v_i} \tag{2.2}$$

O tempo determinado é formatado pela função `timeToStr()` da classe *Utils* de forma a apresentar a estimativa em segundos e milissegundos.

2.2.6 Pergunta 6

> **Enunciado:** *Quanto tempo achas que falta até ficares com metade da bateria que tens agora?*

Para a resolução deste problema, tomamos primeiramente como objetivo representar de forma proxima os dados fornecidos. Por este motivo, após uma breve analise concluimos que seria preferivel o uso de um algoritmo baseado numa curva de regressão não linear.

Com base na bateria desejada, é estimado o tempo que demorará a chegar ao valor dado com recurso ao método `predictTimeFromBattery()` da classe *Robot*. Esta classe calcula a cada frame a variação do tempo desde o último momento em que a bateria atingiu os 100% e a bateria atual na forma de uma regressão não linear a fim de se poder prevêr o tempo necessário até que a bateria atinja o valor previamente indicado.

Tomando em consideração os eixos, Bateria (representado pelo Y) e Duração do Tempo (representado pelo X), foram recolhidos os pontos e guardados numa lista. Sobre eles foi aplicada uma Regressão Não Linear com recurso à função `curve_fit()` proveniente da biblioteca *Scipy* com o objetivo de optimizar a curva.

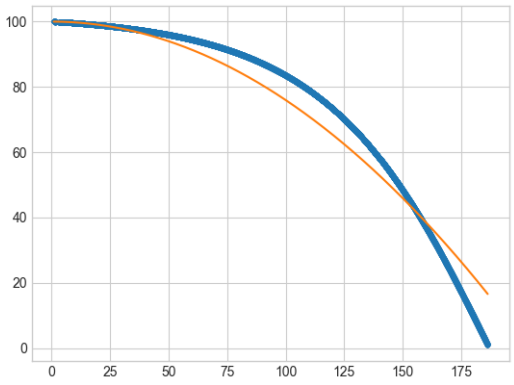

Figura 2.4: Gráfico da curva obtida comparando o algoritmo de Regressão Não Linear (cor laranja) com os pontos recolhidos (cor azul)

2.2.7 Pergunta 7

> **Enunciado:** *Qual é a probabilidade da próxima pessoa a encontrares ser uma criança?*

Para responder a esta pergunta, tomámos particular atenção ao seguinte excerto do enunciado do projeto:

> *Os clientes podem ou não trazer um carrinho de compras. Alguns adultos (clientes) trazem crianças com eles.*

Tal significa que os eventos **não são independentes**, o que invalida por princípio a aplicação direta de uma probabilidade condicionada. Neste sentido, optou-se

por uma **probabilidade condicionada baseada numa Rede *Bayesiana*** (Figura 2.5).

Seja então:

- A: Adulto
- B: Carrinho
- C: Criança

Com base nesta Rede *Bayesiana*, sabemos que a probabilidade procurada é a seguinte:

$$P(C) = \frac{P(C, B)}{P(B)} \tag{2.3}$$

A probabilidade $P(B)$ pode ser determinada diretamente pela divisão entre o número de carrinhos e o número total de "objetos" (adultos, crianças e carrinhos). Contudo, a probabilidade $P(C, B)$ é dada pela equação (2.4).

$$P(C, B) = \sum_{X \in \{V,F\}} P(B, A, C) = P(B, A, C) + P(B, \neg A, C) \tag{2.4}$$

Por conseguinte, as duas probabilidades necessárias são dadas pela respetiva marginalização:

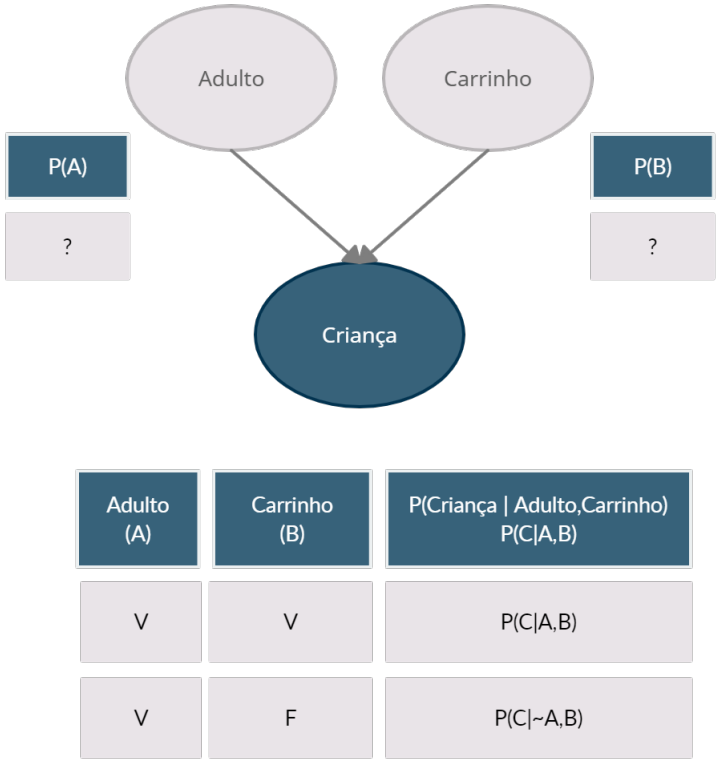

Figura 2.5: Rede *Bayesiana* que correlaciona os seguintes eventos: zona conter adultos, zona conter carrinho e zona conter crianças.

- $P(B, A, C) = P(A) \cdot P(B) \cdot P(C|A, B)$;
- $P(B, \neg A, C) = P(\neg A) \cdot P(B) \cdot P(C|\neg A, B)$.

Onde $P(\neg A) = 1 - P(A)$.

Utilizando a tabela fornecida pelo professor no enunciado, podemos obter o resultado direto das seguintes probabilidades:

- $P(C|A, B) = 0.8$;
- $P(C|\neg A, B) = 0.1$.

O método `getProbabilityOfNextPersonBeAChild()` da classe *Environment* realiza esta série de cálculos a fim de encontrar uma solução à equação (2.4).

2.2.8 Pergunta 8

> **Enunciado:** *Qual é a probabilidade de encontrar um adulto numa zona se estiver lá uma criança mas não estiver lá um carrinho?*

Por falta de indicação em contrário no enunciado do projeto, consideramos que estes são eventos **independentes**, pelo que pode ser aplicada a **probabilidade condicionada**.

O método `getProbabilityOfAdultInZoneIfThereIsChildButNoShopcar()` da classe *Environment* realiza este cálculo, respondendo à equação (2.5):

$$P(A|C, \neg B) = \frac{P(A \cap C \cap \neg B)}{P(C \cap \neg B)} \tag{2.5}$$

Onde:

- $P(A)$: probabilidade de encontrar um adulto na zona;
- $P(B)$: probabilidade de encontrar um carrinho na zona.
- $P(C)$: probabilidade de encontrar uma criança na zona.

Tendo em conta que, por exemplo, $P(C \cap \neg B) = (\text{zonas com C } \cap \neg \text{B})/(\text{total zonas visitadas pelo robo ate ao momento})$, o cálculo destas probabilidades é direto, sendo por fim retornado o resultado da divisão da equação (2.5).

2.3 Conclusões

No presente capítulo foram apresentados os passos e os métodos necessários ao desenvolvimento do projeto prático e do funcionamento do mesmo.

Desta forma, através do conteúdo exposto neste capítulo, encontra-se a apresentação do projeto desenvolvido e o funcionamento do mesmo, mas também a contextualização das ferramentas utilizadas conforme listadas na Secção 1.3.

Capítulo 3

Reflexão Crítica e Problemas Encontrados

3.1 Objetivos Propostos vs. Alcançados

A Tabela 3.1 expõe os objetivos propostos inicialmente para o projeto e identifica quais foram alcançados totalmente ou parcialmente, e quais não foram bem sucedidos.

Objetivo proposto	**Alcançado?**
Pergunta 1	✓
Pergunta 2	✓
Pergunta 3	✓
Pergunta 4	✓
Pergunta 5	✓
Pergunta 6	✓
Pergunta 7	✓
Pergunta 8	✓
Documentação do código	✓

Tabela 3.1: Objetivos propostos e respetiva indicação de sucesso.
Legenda. ✓ Totalmente alcançado; – Não alcançado.

3.2 Divisão de Tarefas

Para a gestão e divisão das tarefas que delineiam o projeto, foi realizada uma reunião inicial onde o foco incidiu na definição das metas por cada membro do

grupo de forma balanceada. A referida gestão é apresentada na tabela 3.2. A escolha baseou-se no equilíbrio da dificuldade e temas das questões.

Por seu turno, o relatório e a apresentação foram divididos pelas diferentes partes que cada membro implementou, assim como pelos restantes Capítulos adicionais. A revisão final do relatório foi, contudo, conjunta a fim de garantir a sua coerência.

Tarefas	**Diogo Simões**	**Cristiano Santos**
Pergunta 1		✓
Pergunta 2	✓	
Pergunta 3	✓	
Pergunta 4	✓	
Pergunta 5		✓
Pergunta 6	✓	
Pergunta 7		✓
Pergunta 8		✓
Relatório	✓	✓
Apresentação	✓	✓

Tabela 3.2: Distribuição de tarefas pelos elementos do grupo.

3.3 Problemas Encontrados

A implementação dos diferentes algoritmos necessários para responder às diversas perguntas colocadas levou a que fossem encontrados alguns problemas, estando os mais notáveis resumidos na Tabela 3.3, incluindo as soluções encontradas para os ultrapassar.

3.4 Reflexão Crítica

Nesta secção expõe-se uma reflexão crítica sobre o trabalho, destacando os pontos fortes, os pontos fracos e algumas observações que mereçam destaque.

Destacam-se como **pontos fortes**:

1. Forte estruturação dos tipos de dados com recurso a classes;
2. Utilização dos grafos para aumento significativo da eficiência;
3. Implementação de um método altamente eficiente para estimar a duração da bateria e prever a velocidade.

Problema	**Solução**
Determinação da relação entre a bateria, velocidade e tempo	Utilização de regressões lineares e não-lineares para as respetivas estimativas
Localização imprecisa dos objetos	Determinar a direção do *robot* de forma a estimar a localização real do objeto

Tabela 3.3: Problemas encontrados durante o desenvolvimento do projeto e respetivas soluções.

Contudo, alguns **pontos fracos** são de realçar, nomeadamente:

1. O código-fonte final, incluindo documentação, é denso e de difícil navegação;
2. As funções lineares que relacionam a bateria e a velocidade com o tempo não representam fielmente a variação que ocorre ao longo do tempo (dedução nas observações abaixo);
3. Várias funções não foram otimizadas a fim de melhorar o consumo de recursos e/ou de aumentar a eficiência temporal.

Por fim, anotam-se as seguintes **observações** sobre a realização do projeto:

1. O uso de grafos não permite obter o caminho exato que o *robot* deve fazer pelo mundo até determinada sala, sendo por conseguinte imprecisa a distância calculada para lá chegar;
2. O método de estimação da bateria do *robot* é impreciso uma vez que o comportamento real destas funções não se revela linear. Acreditamos que possam revelar um comportamento exponencial (a sua implementação não seria de todo complicada, uma vez que a derivada e a primitiva de uma exponencial é ela própria.
3. Na primeira pergunta, só nos apercebemos na fase final do projeto, poderíamos ter implementado uma árvore de decisão. Treinando a árvore e fornecendo critérios de decisão, o *robot* ficaria mais apto para distinguir nomes do sexo feminino e masculino.

Capítulo 4

Conclusões e Trabalho Futuro

4.1 Conclusões Principais

Este projeto permitiu-nos adquirir um melhor conhecimento acerca da linguagem *Python*, como também nos permitiu evoluir em temas recorrentes na Unidade Curricular de Inteligência Artificial. Foi uma maneira inteligente de aplicar os algoritmos lecionados na parte teórica desta cadeira, nomeadamente na parte dos algoritmos de pesquisa de caminhos e nas probabilidades relacionadas com a construção de redes *Bayesianas*.

O projeto encorajou também uma constante pesquisa, no que diz respeito à busca de soluções para as perguntas solicitadas, fazendo com que os alunos se deparassem com várias abordagens para o mesmo problema escolhendo a que achavam mais adequada. Promoveu também o exercicio de raciocinio.

Este projeto revelou como a IA tem atualmente aplicações interessantes na área dos *robots* domésticos de limpeza, os quais têm tido uma procura crescente no mercado.

4.2 Trabalho Futuro

No futuro, a nossa abordagem em relação ao projeto passaria por refazer certas respostas, nomeadamente em termos de organização do código. Em vez de termos alguns algortimos nas funcões de resposta do ficheiro *agente.py*, optaríamos por criar uma função à parte, e chamá-la apenas no respetivo local.

Seria também nossa intenção estudar os melhores métodos de otimização de cada função a fim de evitar processos potencialmente redundantes.

Bibliografia

[1] N. developers, "NetworkX — NetworkX documentation," 2020, [Online] https://networkx.org/. Último acesso a 20 de dezembro de 2021.

[2] P. S. Foundation, "Welcome to Python.org," 2021, [Online] https://www.python.org/. Último acesso a 22 de dezembro de 2021.

[3] I. GitHub, "GitHub," 2021, [Online] https://github.com/. Último acesso a 22 de dezembro de 2021.

[4] Microsoft, "Visual Studio Code - Code Editing. Redefined," 2021, [Online] https://code.visualstudio.com/. Último acesso a 22 de dezembro de 2021.

[5] I. dos Registos e do Notariado, "Registo de nomes (feminino) - dados.gov.pt - portal de dados abertos da administração pública," Aug 2021. [Online]. Available: https://dados.gov.pt/pt/datasets/nomesfeminino/