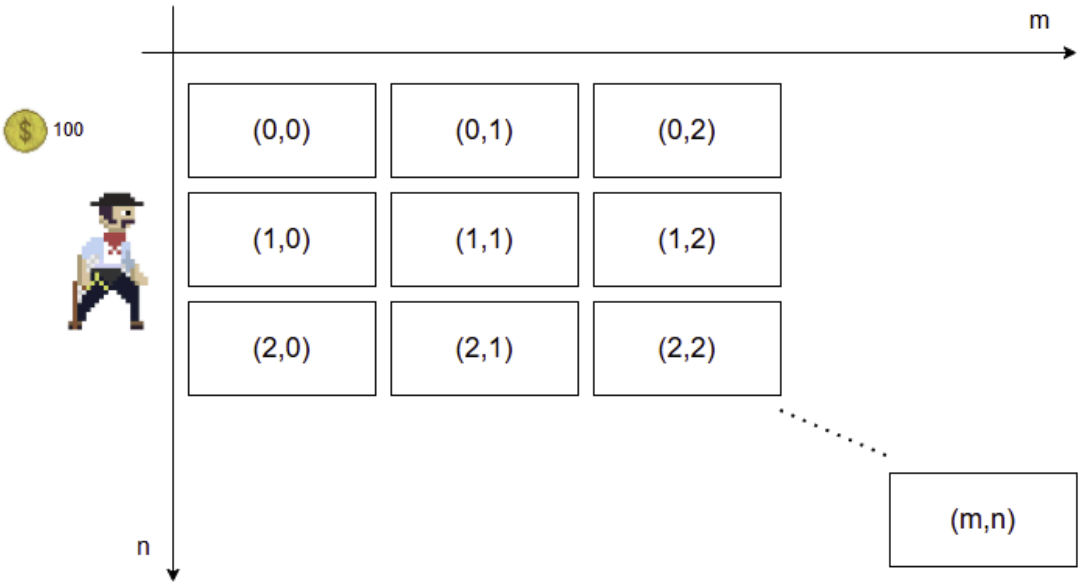
Este documento serve como relatório do projeto *"Raiders of the Lost Ark"* da disciplina de *Inteligência Artificial Aplicada*, ano letivo 2025/2026, do curso Mestrado em Engenharia Informática da Universidade de Évora.

Professor: António Anjos

Alunos: Fábio Macarrão (57098), Vitor Costa (70323)

Interpretação

O objetivo deste projeto é criar um jogo, ou aplicação, que manipula/interprete a probabilidade de ocorrência de variáveis aleatórias em função de informação obtida. Em suma, um tesouro é escondido numa matriz de dimensão m x n, e cabe ao jogador aferir, qual a célula da matriz com maior probabilidade de conter o tesouro dadas as ações tomadas anteriormente – cada ação tem um custo associado que consome pontos do jogador, ou dinheiro.



O utilizador tem ao seu dispor as seguintes ações:

- Utilizar uma das três "tecnologias" disponíveis para avaliar as probabilidades locais (por célula);
- Cavar um local para obter o tesouro – só pode ser executado uma vez;
- revelar o tesouro - para debug apenas;
- alterar o valor da fiabilidade dos sensores utilizados - feature;
- reiniciar o jogo.

As tecnologias que tem ao seu dispor avaliam a probabilidade de uma determinada célula da matriz conter a *arca perdida* P(A), o output é função da distância da célula onde é aplicada a tecnologia para a posição do tesouro escondido, as tabelas seguintes relacionam estes valores:

Ground Penetration Radar:

distance	STRONG	MODERATE	WEAK	NONE
0	0,75	0,15	0,07	0,03
1	0,1	0,7	0,15	0,05
2	0,05	0,15	0,65	0,15
≥3	0,02	0,08	0,15	0,75

Magnetic Survey:

distance	HIGH	MEDIUM	LOW
0	0,7	0,2	0,1
1	0,15	0,65	0,2
≥2	0,1	0,25	0,65

Visual Inspection:

distance	SUSPICIOUS	NORMAL
0	0,6	0,4
≥1	0,2	0,8

A distância entre células é determinada pelo método das distâncias de Manhattan, isto é, da célula (0,0) a (2,2) são atravessadas quatro células → d=4. Assim, as tabelas em cima referem-se à probabilidade condicional de cada aparelho em função da posição da *arca perdida*, admitindo que a arca se encontra em (2,2) então o resultado esperado dos três sensores GPR, MAG e VIS em (0,0) seria NONE, LOW e NORMAL respectivamente..

No exemplo de cima, a ação de utilizar o GPR em (0,0) quando a arca se encontra em (2,2), com o resultado de NONE, reduz a probabilidade de a arca se encontrar neste espaço e aumenta a nossa certeza em relação aos restantes, ou pelo menos aumenta a probabilidade de a arca lá se encontrar.

Exemplo 3x3 do enunciado

Para uma grid 3x3 com a arca escondida na célula (1,1), determinar a probabilidade para cada célula conter a arca dadas as seguintes observações:

1. GPR em (0,0) com resultado MODERATE:

Da tabela de especificações do GPR sabemos que $P(A_{00}|GPR_{00})=0,15$, a probabilidade de existir uma arca em (0,0), é moderada → 15%; da mesma

tabela aferimos que é mais provável existir uma arca numa zona próxima (d=1), para duas células de distância a probabilidade cai para 15% novamente e para valores superiores esta é 3%. Do teorema de Bayes: P(A|GPR) = [P(GPR|A) / P(GPR)].P(A); segundo o método de inferência por enumeração: P(A|GPR) = α . P(GPR=MODERATE | A=a).P(A=a), e com recurso a uma folha de cálculo:

Area(m,n)	d	P(GPR=MOD|A_{mn})	P(GPR=MOD|A_{mn}).P(A=a)	L1Norm
(0,0)	0	0.15	0.01666665	0.06696428571
(0,1)	1	0.7	0.0777777	0.3125
(0,2)	2	0.15	0.01666665	0.06696428571
(1,0)	1	0.7	0.0777777	0.3125
(1,1)	2	0.15	0.01666665	0.06696428571
(1,2)	3	0.08	0.00888888	0.03571428571
(2,0)	2	0.15	0.01666665	0.06696428571
(2,1)	3	0.08	0.00888888	0.03571428571
(2,2)	4	0.08	0.00888888	0.03571428571
sum	-	2.24	0.24888864	1

2. MAG em (1,0) com resultado HIGH:

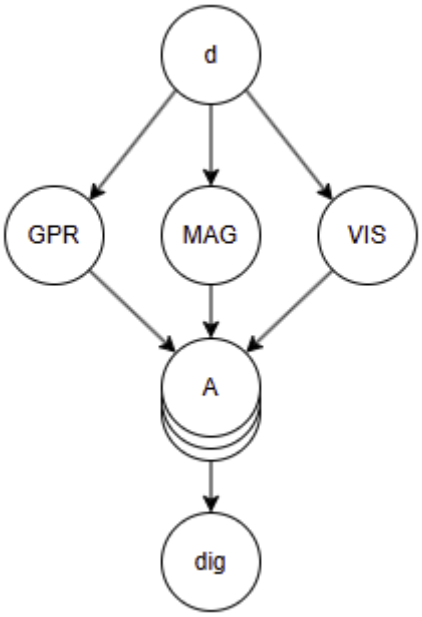
Primeiro determinamos as distâncias entre a célula escolhida e restantes, retiramos o valor de P(MAG=HIGH|A_{mn}) da tabela de Magnetic Survey e procedemos de igual modo numa folha de cálculo.

Area(m,n)	d	P(MAG=HIG|A_{mn})	P(MAG=HIG|A_{mn}).P(A=a)	L1Norm
(0,0)	1	0.15	0.01666665	0.09090909091
(0,1)	2	0.1	0.0111111	0.06060606061
(0,2)	3	0.1	0.0111111	0.06060606061
(1,0)	0	0.7	0.0777777	0.4242424242
(1,1)	1	0.15	0.01666665	0.09090909091
(1,2)	2	0.1	0.0111111	0.06060606061
(2,0)	1	0.15	0.01666665	0.09090909091
(2,1)	2	0.1	0.0111111	0.06060606061
(2,2)	3	0.1	0.0111111	0.06060606061
sum	-	1.65	0.18333315	1

Assim a probabilidade da célula (1,1) conter a arca pode ser determinada por P(A_{11}) = P(A_{11}|GPR_{00},MAG_{00}) = P(A_{11}|GPR_{00}=MODERATE).P(A_{11}|MAG_{10}=HIGH) = 0.15 *0.15 / 0.6665 =

0.033758 → 3.4%. → este valor induz o jogador em erro, realçando o fator de fiabilidade dos dispositivos eletrônicos, considerado mais à frente.

Modelo Bayesiano

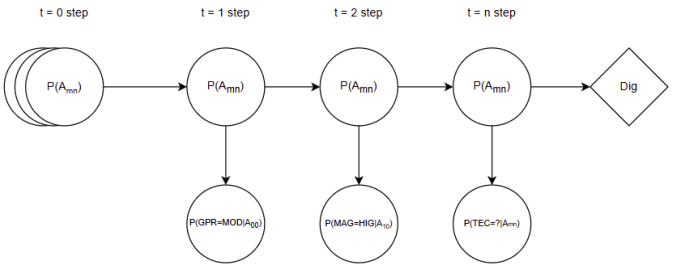


Considerações:

- d = {0,1,2,3,4,5,...,(m-1+n-1)}
- Temos as CPT(GPR|d), CPT(MAG|d), CPT(VIS|d) disponíveis na página 2.
- GPR _||_ MAG _||_ VIS | A
- P(Dig,A,G,M,V) = P(Dig|A).P(A|G,M,V).P(M).P(G).P(V)
- P(Dig) = <P(dig|a).P(a) P(dig|~a).P(~a)>

Visto como cadeia de Markov:

$P(a) \propto \alpha.P(e_1|a).P(e_2|a)P(a_0)$



Detalhes de Implementação

A interface do utilizador deste projeto foi implementada com a biblioteca tkinter, é importante referir a definição da classe *TreasureGame* onde são inicializadas as constantes ou parâmetros do jogo como o budget inicial do jogador, o tamanho da matriz/tabuleiro de jogo, flags de configuração e debug, bem como dicionários com os valores de probabilidade condicional dos sensores. O método *setup_ui* invoca uma série de recursos da biblioteca do tkinter para implementar as necessidades do projeto e faz uso de outras que passamos a apresentar:

- **Lista de tecnologias disponíveis**: Tkinter radio buttons e dicionários de tecnologias e custos associados, bem como dicionários python com as

probabilidades condicionadas por distância, a tecnologia “ground perforation radar” vem por defeito, cabe ao jogador selecionar outra.

- **Histórico de Ações:** Tkinter Text object para mostrar valores adicionados a variável do tipo set. Também é utilizado um dicionário para armazenar o histórico de ações em cada área, representadas como botões do tkinter;
- **Cálculo de probabilidade de presença da arca perdida:** o método *cell_click* determina a distância ao tesouro escondido pelo método *numpy.cityblock* com a posição da arca e apresenta o valor da leitura executada pelo sensor de acordo com as tabelas CPT definidas anteriormente, executa o método *update_probabilities* onde é feita a normalização (normalização L1) em função das probabilidades totais e posteriormente atualiza o UI com o método *update_grid_display;*
- **Heatmap:** A cor de fundo de cada botão é definida de acordo com o valor da probabilidade P(A) em cada célula/botão - definimos uma variável que codifica o valor da cor de fundo para cada botão em hex color code;
- **Revelar a arca:** É definida uma flag para controlar a visualização da arca no tabuleiro, esta é atualizada pelo método *toggle_treasure* e é utilizada para definir o estilo aplicado ao botão com o valor a true, preenchendo o fundo com a cor amarela e o texto “`!!! TESOURO !!!`”;
- **Fator de fiabilidade tecnológica (game feature):** Admitindo que o sensor não tem falhas na medição então não ocorre a situação do exemplo 3x3, em que MAG_{10}=HIGH, o valor medido pelo sensor seria MODERATE → 0.15 x 0.7 /0.6665 = 15.8%, e poderíamos aferir com maior certeza as restantes regiões a explorar. A nossa implementação faz uso do método *choices* da biblioteca *random*, onde a fiabilidade de funcionamento de um dispositivo é tida em consideração em função do parâmetro weights, assim para sensores perfeitos definimos um novo dicionário para armazenar as probabilidades de uma saída do sensor em função da distância, esta opção está identificada como sensores perfeitos e pode ser selecionada pelo jogador no UI;
- **Tamanho de tabuleiro customizável:** A dimensão do tabuleiro de jogo pode ser definida inicialmente ao passar como argumentos as linhas e colunas respectivamente, por exemplo:

```
> python HiddenArkGame.py 10 10
```

/!\ os valores de m e n afetam o desempenho geral da aplicação uma vez que o método de inferência por enumeração utiliza a probabilidade de todas as células no cálculo das probabilidades P(A).