UNIVERSIDADE DO MINHO
MIEI

MODELOS DETERMINÍSTICOS DE INVESTIGAÇÃO OPERACIONAL

Incêndios Florestais

Grupo:
Afonso Sousa - a74196
João Nunes - a82300
Jorge Cardoso - a75876
Luís Braga - a82088
Luís Martins - a82298

Braga, Portugal
25 de Abril de 2019

Conteúdo

Capítulo 1

Questão 1

1.1 Resposta A

Parâmetros:

n - Dimensão de uma matriz quadrada;
vN - Matriz (de dimensão n*n) com o custo de ir para norte de um certo nodo;
vS - Matriz (de dimensão n*n) com o custo de ir para sul de um certo nodo;
vE - Matriz (de dimensão n*n) com o custo de ir para este de um certo nodo;
vW - Matriz (de dimensão n*n) com o custo de ir para oeste de um certo nodo.

A título de exemplo, representando todas as matrizes acima referidas com n=7 segundo as indicações dadas chega-se ao seguinte grafo:

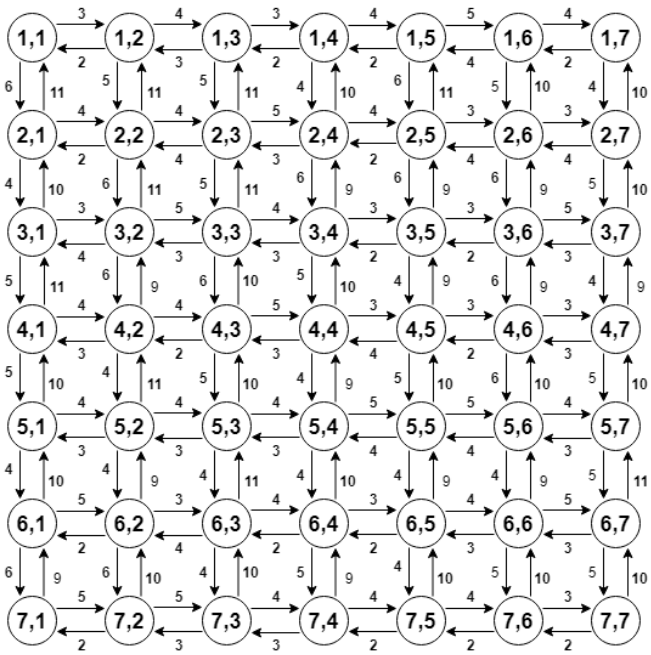

Figura 1.1: Grafo de uma matriz 7x7.

Variáveis de decisão:

M - Matriz (de dimensão n*n) com os tempos de ignição em cada nodo;

Função objetivo:

$$Max\ z = \sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij}$$

Na função objetivo pretende-se maximizar o tempo de ignição em cada célula, pelo que é necessário efetuar o somatório dos tempos de ignição das células.

Sujeito a:

$$R1 : \forall_{2\leq i\leq n}(\forall_{1\leq j\leq n}(M_{i-1j} - M_{ij} \leq vN_{ij}))$$

$$R2 : \forall_{1\leq i\leq n-1}(\forall_{1\leq j\leq n}(M_{i+1j} - M_{ij} \leq vS_{ij}))$$

$$R3 : \forall_{1\leq i\leq n}(\forall_{2\leq j\leq n}(M_{ij-1} - M_{ij} \leq vW_{ij}))$$

$$R4 : \forall_{1\leq i\leq n}(\forall_{1\leq j\leq n-1}(M_{ij+1} - M_{ij} \leq vE_{ij}))$$

$$R5 : M_{11} = 0$$

Nas restrições do problema, é necessário restringir a propagação do fogo apenas aos nodos adjacentes ao nodo de ignição, neste caso o nodo (1,1) ou o nodo 1. O fogo apenas se pode propagar em 4 direções diferentes, para o norte, sul, este e oeste, sem propagações diagonais. Na determinação do tempo de ignição de uma célula, aceder-se-á aos valores do tempo de ignição da estrutura relativa aos custos em cada direção(vN, vS, vE, vW), sendo depois a estrutura M preenchida.

1.2 Resposta B

Parâmetros:

n - Dimensão de uma matriz quadrada;
vN - Matriz (de dimensão n*n) com o custo de ir para norte de um certo nodo;
vS - Matriz (de dimensão n*n) com o custo de ir para sul de um certo nodo;
vE - Matriz (de dimensão n*n) com o custo de ir para este de um certo nodo;
vW - Matriz (de dimensão n*n) com o custo de ir para oeste de um certo nodo.

Variáveis de decisão:

xN - Matriz (de dimensão n*n) com o número de caminhos de que faz parte o arco de ir para norte de um certo nodo;

xS - Matriz (de dimensão n*n) com o número de caminhos de que faz parte o arco de ir para sul de um certo nodo;

xE - Matriz (de dimensão n*n) com o número de caminhos de que faz parte o arco de ir para este de um certo nodo;

xW - Matriz (de dimensão n*n) com o número de caminhos de que faz parte o arco de ir para oeste de um certo nodo;

Função objetivo:

$$Min\ z = \sum_{i=1}^{n}\sum_{j=1}^{n}(vN_{ij} * xN_{ij} + vS_{ij} * xS_{ij} + vE_{ii} * xE_{ij} + vW_{ij} * xW_{ij})$$

Depois da passagem para o modelo dual do exercício *1.a* obteve-se a função objectivo descrita acima. A função visa minimizar o somatório das somas dos produtos entre o tempo que o fogo demora a propagar-se para os quatro pontos cardeais e o número de caminhos de que faz parte o arco que se direcciona para o respectivo ponto cardeal, a partir de uma determinada célula.

Sujeito a:

$$R1 : xS_{11} + xE_{11} = n * n - 1$$

$$R2 : xS_{1n} + xW_{1n} - xE_{1(n-1)} - xN_{2n} = -1$$

$$R3 : xN_{n1} + xE_{n1} - xS_{(n-1)1} - xW_{n2} = -1$$

$$R4 : xN_{nn} + xW_{nn} - xS_{(n-1)n} - xE_{n(n-1)} = -1$$

$$R5 : \forall_{2\leq j\leq n-1}(xS_{1j} + xE_{1j} + xW_{1j} - xN_{2j} - xW_{1(j+1)} - xE_{1(j-1)} = -1)$$

$$R6 : \forall_{2\leq j\leq n-1}(xN_{nj} + xE_{nj} + xW_{nj} - xS_{(n-1)j} - xW_{n(j+1)} - xE_{n(j-1)} = -1)$$

$$R7 : \forall_{2\leq i\leq n-1}(xN_{i1} + xS_{i1} + xE_{i1} - xN_{(i+1)1} - xS_{(i-1)1} - xW_{i2} = -1)$$

$$R8 : \forall_{2\leq i\leq n-1}(xN_{in} + xS_{in} + xW_{in} - xN_{(i+1)n} - xS_{(i-1)n} - xE_{i(n-1)} = -1)$$

$$R9 : \forall_{2\leq i\leq n-1}(\forall_{2\leq j\leq n-1}((xN_{ij}+xE_{ij}+xW_{ij}+xS_{ij})-(xN_{(i+1)j}+xS_{(i-1)j}+xW_{i(j+1)}+xE_{i(j-1)}) = -1))$$

Tendo em conta as limitações dadas pelo contexto do problema e a passagem do modelo primal para dual, formularam-se as restrições supracitadas. Todas estas restrições pretendem delimitar o número de destinos que podem ser alcançados a partir de cada arco, para o efeito dividiram-se as restrições por número de entradas/saídas. Temos, por isso, os cantos (2 acessos), as bordas, sem os cantos(3 acessos) e o interior do grafo(4 acessos).

Começou-se pelos cantos do grafo (figura 1.1). O nodo 1 é caso especial, pois o fogo tem lá início, consequentemente, a partir deste nodo o fogo alcançará todos os outros, portanto a soma de xS e xE será a igual a 48 (R1). Nos restantes nodos de semelhante tipo não se sabe o valor ótimo concreto, no entanto, podem-se formular as restrições uma vez que se perde um destino quando o fogo alcança um nodo (R2, R3 e R4), ficando a formula igualada a -1. Na verdade, este tipo de pensamento pode ser aplicado não só nos nodos das bordas do grafo (R5, R6, R7 e R8) como também nos nodos interiores (R9).

1.3 Resposta C

A partir da instância em anexo e recorrendo ao software *IBM ILOG CPLEX*, adaptando o modelo às especificações do programa, foi gerado o seguinte código:

```
int n=...;
int vN[1..n][1..n]=...;
int vS[1..n][1..n]=...;
int vW[1..n][1..n]=...;
int vE[1..n][1..n]=...;
//int C[1..n][1..n];


dvar float+ M[1..n][1..n];

maximize sum (i in 1..n, j in 1..n) M[i][j];

subject to {
    Restricao1: forall (i in 2..n, j in 1..n) M[i-1][j] - M[i][j] <= vN[i][j];
    Restricao2: forall (i in 1..n-1, j in 1..n) M[i+1][j] - M[i][j] <= vS[i][j];
    Restricao3: forall (i in 1..n, j in 2..n) M[i][j-1] - M[i][j] <= vW[i][j];
    Restricao4: forall (i in 1..n, j in 1..n-1) M[i][j+1] - M[i][j] <= vE[i][j];
    Restricao5: M[1][1] == 0;
}
```

Figura 1.2: Modelo Primal

Soluções ótimas primal através da resolução do modelo primal:

Após se fazer o ficheiro .mod com o modelo primal mostrado na figura anterior, obteram-se as seguintes soluções ótimas primal:

```
// solution (optimal) with objective 1197
// Quality There are no bound infeasibilities.
// There are no reduced-cost infeasibilities.
// Maximum Ax-b residual              = 0
// Maximum c-B'pi residual            = 0
// Maximum |x|                       = 47
// Maximum |slack|                   = 17
// Maximum |pi|                      = 49
// Maximum |red-cost|                = 0
// Condition number of unscaled basis = 2,2e+01
//

M = [[0
             3 7 10 14 19 23]
             [6 8 12 14 18 21 24]
             [10 13 17 20 23 26 29]
             [15 19 23 25 27 30 33]
             [20 23 27 29 32 36 38]
             [24 29 31 33 36 40 43]
             [30 33 35 38 40 44 47]];
```

Figura 1.3: Soluções ótimas primal

Passos sobre a resolução das soluções ótimas dual através do modelo primal

Feito o modelo primal, modificou-se o ficheiro .ops da sua configuração, alterando o seu run e fazendo export do LP na próxima execução. Em seguida, foi feito run da configuração com esse ficheiro .ops alterado e gerou-se um ficheiro .lp. Nesse ficheiro, tinha a função objetivo com o somatório e as restrições extendidas (para um n = 7) do modelo primal.

Por consequência, ao fazer-se "NomedaRestrição".dual obtem-se um elemento de uma das quatro matrizes que se teria que obter como as soluções ótimas dual. Assim, obteve-se as soluções ótimas dual da seguinte maneira:

```
xn[2][1] = c1.dual;
xn[2][2] = c2.dual;
xn[2][3] = c3.dual;
xn[2][4] = c4.dual;
xn[2][5] = c5.dual;
xn[2][6] = c6.dual;
xn[2][7] = c7.dual;
```

Figura 1.4: Alguns exemplos do preenchimento da matriz xn

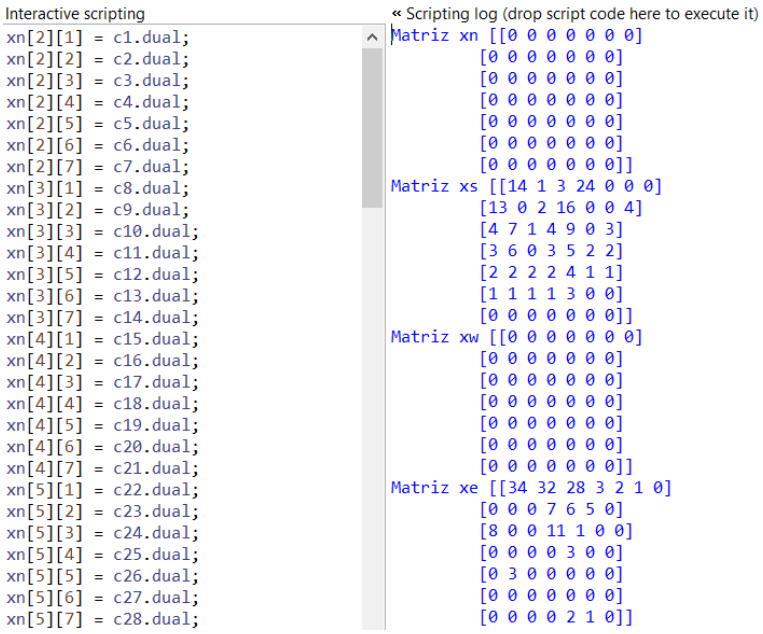

Figura 1.5: Soluções Ótimas Dual

1.4 Resposta D

Partindo da instância em anexo, e tal como na anterior, recorrendo ao software *IBM ILOG CPLEX*, uma vez adaptado às especificações do programa, obteu-se o seguinte código:

```
int n=...;
int vN[1..n][1..n]=...;
int vS[1..n][1..n]=...;
int vW[1..n][1..n]=...;
int vE[1..n][1..n]=...;
//int C[1..n][1..n];

dvar int+ xN[1..n][1..n];
dvar int+ xS[1..n][1..n];
dvar int+ xW[1..n][1..n];
dvar int+ xE[1..n][1..n];

minimize sum (i in 1..n, j in 1..n) (vN[i][j]*xN[i][j] + vS[i][j]*xS[i][j] +
                                     vW[i][j]*xW[i][j] + vE[i][j]*xE[i][j]);

subject to {
    Restricao1: xS[1][1] + xE[1][1] == n*n-1;

    Restricao2: forall(i in 2..n-1, j in 2..n-1) (xN[i][j]+xE[i][j]+xW[i][j]+xS[i][j])-(xN[i+1][j]+xS[i-1][j]+xW[i][j+1]+xE[i][j-1]) == -1;

    Restricao3: xS[1][n] + xW[1][n] - xE[1][n-1] - xN[2][n] == -1;
    Restricao4: xN[n][1] + xE[n][1] - xS[n-1][1] - xW[n][2] == -1;
    Restricao5: xN[n][n] + xW[n][n] - xS[n-1][n] - xE[n][n-1] == -1;

    Restricao6: forall(j in 2..n-1) xS[1][j]+xE[1][j]+xW[1][j]-xN[2][j]-xW[1][j+1]-xE[1][j-1] == -1;
    Restricao7: forall(j in 2..n-1) xN[n][j]+xE[n][j]+xW[n][j]-xS[n-1][j]-xW[n][j+1]-xE[n][j-1] == -1;
    Restricao8: forall(i in 2..n-1) xN[i][1]+xS[i][1]+xE[i][1]-xN[i+1][1]-xS[i-1][1]-xW[i][2] == -1;
    Restricao9: forall(i in 2..n-1) xN[i][n]+xS[i][n]+xW[i][n]-xN[i+1][n]-xS[i-1][n]-xE[i][n-1] == -1;
}
```

Figura 1.6: Modelo Dual

Soluções ótimas primal através da resolução do modelo primal:

Utilizando o modelo dual, da figura anterior, as seguinte soluções ótimas do dual foram obtidas:

```
// solution (optimal) with objective 1197
// Quality Incumbent solution:
// MILP objective                                  1,1970000000e+03
// MILP solution norm |x| (Total, Max)             2,94000e+02  3,30000e+01
// MILP solution error (Ax=b) (Total, Max)         0,00000e+00  0,00000e+00
// MILP x bound error (Total, Max)                 0,00000e+00  0,00000e+00
// MILP x integrality error (Total, Max)           0,00000e+00  0,00000e+00
// MILP slack bound error (Total, Max)             0,00000e+00  0,00000e+00
//

xN = [[0
             0 0 0 0 0 0]
             [0 0 0 0 0 0 0]
             [0 0 0 0 0 0 0]
             [0 0 0 0 0 0 0]
             [0 0 0 0 0 0 0]
             [0 0 0 0 0 0 0]
             [0 0 0 0 0 0 0]];
xS = [[15 1 2 24 0 0 0]
             [14 0 1 19 0 0 1]
             [12 0 0 4 12 0 0]
             [5 4 0 3 5 2 2]
             [4 0 2 2 4 1 1]
             [1 1 1 1 3 0 0]
             [0 0 0 0 0 0 0]];
xW = [[0 0 0 0 0 0 0]
             [0 0 0 0 0 0 0]
             [0 0 0 0 0 0 0]
             [0 0 0 0 0 0 0]
             [0 0 0 0 0 0 0]
             [0 0 0 0 0 0 0]];
xE = [[33 31 28 3 2 1 0]
             [0 0 0 4 3 2 0]
             [1 0 0 14 1 0 0]
             [6 1 0 0 6 3 0]
             [0 3 0 0 0 0 0]
             [2 0 0 0 0 0 0]
             [0 0 0 0 2 1 0]];
```

Figura 1.7: Soluções ótimas dual

Passos sobre a resolução das soluções ótimas primal através do modelo dual

Tal como no exercício anterior, os passos dados foram exatamente os mesmos, primeiro modificou-se o ficheiro .ops da configuração, fazendo um export para LP. De seguida executou-se e foi criado um novo ficheiro com a extensão .lp, de seguida foram copiadas as partes mais relevante deste ficheiro para um ficheiro mod, tendo o cuidado de adaptar as restrições e função objetivo de modo a cumprir com a sintaxe do software.

De seguida, em cada restrição acrescentou-se o .dual em frente do nome gerado para cada restrição, usando para tal o scripting log, uma vez que ao acrescentar o .dual à frente, cada restrição irá ser calculada usando o método primal, contudo a solução obtida não corresponde à solução real do método primal, por razões desconhecidas ao grupo de trabalho, pelo que ao executar apenas parte do script, exemplificado na figura 1.9, a matriz obtida irá ter o seguinte formato:

```
// solution (optimal) with objective 1197
Matriz M [[5 -3 -7 -9 -13 -16 -8]
         [-12 -15 -18 -21 -14 -18 0]
         [0 0 0 0 0 0 0]
         [0 0 0 0 0 0 0]
         [0 0 0 0 0 0 0]
         [0 0 0 0 0 0 -38]
         [-15 -19 -19 -24 -28 -33 0]]
```

Figura 1.8: Soluções ótimas Primal

```
M[1][1] = Restricao1.dual;
M[2][2] = c2.dual;
M[2][3] = c3.dual;
M[2][4] = c4.dual;
M[2][5] = c5.dual;
M[2][6] = c6.dual;
M[3][2] = c7.dual;
```

Figura 1.9: Parte do script para o preenchimento da matriz M

Capítulo 2

Questão 2

2.1 Resposta A

Parâmetros:

n - Dimensão de uma matriz quadrada;
vN - Matriz (de dimensão n*n) com o custo de ir para norte de um certo nodo;
vS - Matriz (de dimensão n*n) com o custo de ir para sul de um certo nodo;
vE - Matriz (de dimensão n*n) com o custo de ir para este de um certo nodo;
vW - Matriz (de dimensão n*n) com o custo de ir para oeste de um certo nodo.
b - Número de recursos;
Δ - Tempo de retardamento;

Variáveis de decisão:

M - Matriz (de dimensão n*n) com os tempos de ignição em cada nodo;

$$\mathrm{x}_{ij} = \begin{cases} 1 & \text{, se o recurso foi alocado para a célula } \quad x_{ij} \quad , i,j \in \{1,...,n\} \\ 0 & \text{, caso contrário} \end{cases}$$

Função objetivo:

$$Max\ z = \sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij}$$

Na função objetivo pretende-se maximizar o tempo de ignição em cada célula, tendo em conta o número de recursos disponíveis, pelo que é necessário efetuar o somatório dos tempos de ignição das células.

Sujeito a:

$$R1 : \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} \leq b$$

$$R2 : \forall_{2\leq i\leq n}(\forall_{1\leq j\leq n}(M_{i-1j} - M_{ij} \leq vN_{ij} + \Delta * x_{ij}))$$

$$R3 : \forall_{1\leq i\leq n-1}(\forall_{1\leq j\leq n}(M_{i+1j} - M_{ij} \leq vS_{ij} + \Delta * x_{ij}))$$

$$R4 : \forall_{1\leq i\leq n}(\forall_{2\leq j\leq n}(M_{ij-1} - M_{ij} \leq vW_{ij} + \Delta * x_{ij}))$$

$$R5 : \forall_{1\leq i\leq n}(\forall_{1\leq j\leq n-1}(M_{ij+1} - M_{ij} \leq vE_{ij} + \Delta * x_{ij}))$$

$$R6 : M_{11} = 0$$

Devido à semelhança com o problema *1.a* as limitações do atual são semelhantes (R2, R3, R4, R5 e R6) com o acréscimo de "$+\Delta * x_{ij}$"em cada uma delas, porque o tempo de propagação do fogo a partir de cada célula é adiado com adição da constante de retardamento Δ se um recurso for alocado para o nodo.

O número total de *b* recursos também deverá ser delimitado, pelo que não deverá ser ultrapassado, para o efeito formulou-se R1.

2.2 Resposta B

Para o problema contextualizado em *2.b*, com Δ e *b* segundo o anexo (Δ=*b*=8), obteve-se a seguinte solução:

```
// solution (optimal) with objective 2391

M = [[0
             11 23 34 46 51 55]
             [14 24 36 46 50 53 56]
             [26 37 42 46 49 52 57]
             [39 43 47 51 53 56 59]
             [44 47 51 55 58 62 64]
             [48 53 55 59 62 66 69]
             [54 57 59 63 66 70 73]];
x = [[1 1 1 1 0 0 0]
             [1 1 1 0 0 0 0]
             [1 0 0 0 0 0 0]
             [0 0 0 0 0 0 0]
             [0 0 0 0 0 0 0]
             [0 0 0 0 0 0 0]
             [0 0 0 0 0 0 0]];
```

Figura 2.1: Solução para o problema descrito em *2.b*.

Depois de examinada a solução (figura 2.2), pode inferir-se que aumentou em relação à solução de *1.a* (figura 1.3), o que seria de esperar devido ao uso de recursos e ao efeito retardante que estes possuem.

É também possível afirmar que os recursos tendem a situar-se em volta do foco de incêndio (1,1), como se pode ver na figura acima na matriz *x*, o que faz com que o tempo total de propagação seja máximo.

2.3 Resposta C

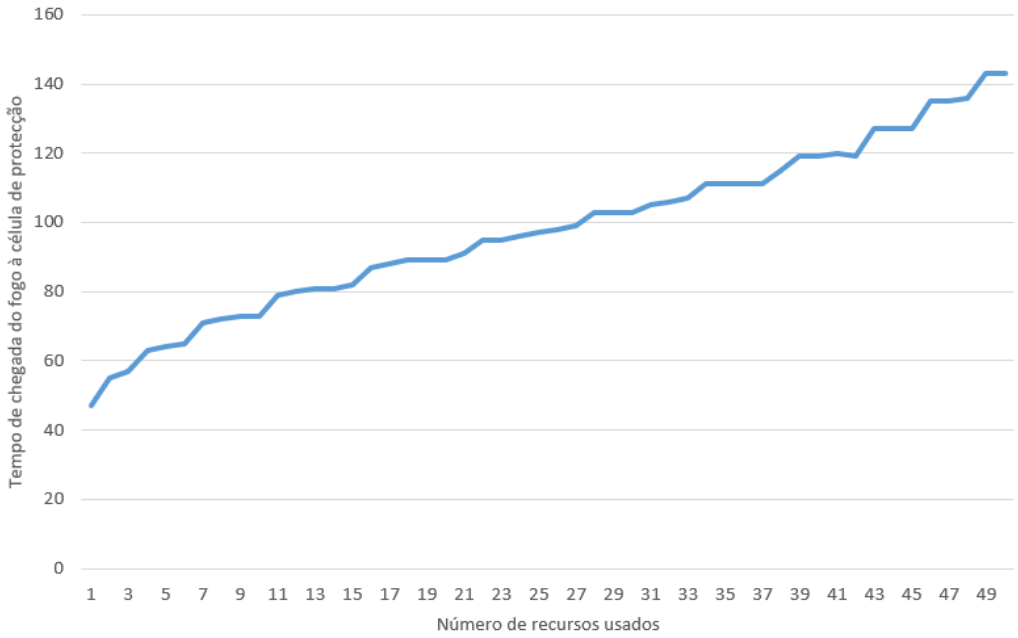

Figura 2.2: Gráfico representativo do tempo que o fogo demora a chegar até (7,7) em função do número de recursos.

Através da análise do gráfico acima (figura 2.2) é possível concluir que o tempo que o incêndio demora a chegar até à célula de protecção (7,7) tende a aumentar com a quantidade de recursos disponíveis. Comportamento este que está de acordo com o previsto, pois, se são alocados cada vez mais recursos, o tempo de propagação até (7,7) tem ,forçosamente, de aumentar.

Capítulo 3

Questão 3

3.1 Resposta A

Parâmetros:

n - Dimensão da matriz, 7;
b - Número de recursos disponíveis, 8;
time - tempo de simulação de incêndio, 12;
Delta - valor de retardamento temporal dos recursos, 8;
vN[1..n][1..n] - Matriz com o custo de ir para Norte desde (x,y);
vS[1..n][1..n] - Matriz com o custo de ir para Sul desde (x,y);
vE[1..n][1..n] - Matriz com o custo de ir para Este desde (x,y);
vW[1..n][1..n] - Matriz com o custo de ir para Oeste desde (x,y).
probIgni[1..n][1..n] - Matriz de probabilidades de ignição na célula (i,j)

Variáveis de decisão:

equipas[1..n][1..n] - Número de recursos colocados na célula (x,y), binária
ardido[1..n][1..n][1..n][1..n] - Para uma ignição em (i,j), quais células (x,y) arderam, binária;
Tempos[1..n][1..n][1..n][1..n] - Para uma ignição em (i,j) indica os tempos de propagação até (x,y), não negativa.

Função objetivo:

$$Min\ z = \sum_{i=1}^{n}\sum_{j=1}^{n} probIgni_{ij} \sum_{i=1}^{n}\sum_{j=1}^{n} ardido_{ijxy}$$

Na função objetivo pretende-se minimizar o valor esperado da área ardida. Para tal temos de ponderar a probabilidade de cada célula entrar em ignição, juntamente com o respetivo número de células ardidas.

Sujeito a:

R1: Máximo de recursos

$$\sum_{i=1}^{n}\sum_{j=1}^{n} equipas_{xy} == b$$

R2: Tempos de propagação

$$\forall_{1\leq i\leq n}(\forall_{1\leq j\leq n}(\forall_{2\leq x\leq n}((\forall_{1\leq y\leq n} Tempos_{ij(x-1)y} \leq Tempos_{ijxy} + vN_{xy} + \Delta * equipas_{xy}))))$$

$$\forall_{1\leq i\leq n}(\forall_{1\leq j\leq n}(\forall_{1\leq x\leq n-1}(\forall_{1\leq y\leq n} Tempos_{ij(x+1)y} \leq Tempos_{ijxy} + vS_{xy} + \Delta * equipas_{xy}))))$$

$$\forall_{1\leq i\leq n}(\forall_{1\leq j\leq n(}\forall_{1\leq x\leq n}(\forall_{2\leq y\leq n} Tempos_{ijx(y-1)} \leq Tempos_{ijxy} + vW_{xy} + \Delta * equipas_{xy}))))$$

$$\forall_{1\leq i\leq n}(\forall_{1\leq j\leq n}(\forall_{1\leq x\leq n}(\forall_{1\leq y\leq n-1} Tempos_{ijx(y+1)} \leq Tempos_{ijxy} + vE_{xy} + \Delta * equipas_{xy}))))$$

R3: Os tempos nos focos de ignição são zero

$$\forall_{1\leq i\leq n}(\forall_{1\leq j\leq n}(Tempos_{ijij} == 0))$$

R4: Células ardidas estão a menos de *tempo* de distância do ponto de ignição

$$\forall_{1\leq i\leq n(}\forall_{1\leq j\leq n-1}(\forall_{1\leq x\leq n(}\forall_{1\leq y\leq n}(ardido_{ijxy} \leq (time - Tempos_{(ijxy/}time))))$$

3.2 Resposta B

Para o tempo limite de 12, as localizações para os 8 recursos permitidos são as seguintes células:

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Como a atribuição de probabilidades é feita segundo plano em que as duas variáveis i e j são decrescentes, é de esperar que haja uma tendência a termos mais equipamentos no canto superior esquerdo do mapa por ter maior probabilidade de ignição.

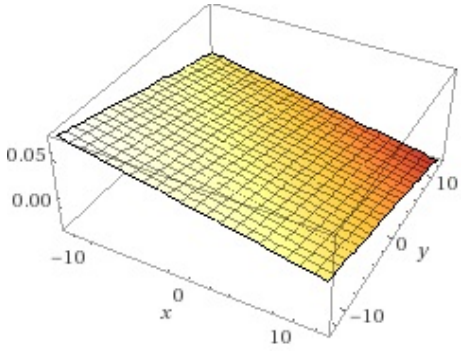

Figura 3.1: Plano da função que atribui as probabilidades de ignição às células da matriz

3.3 Resposta C

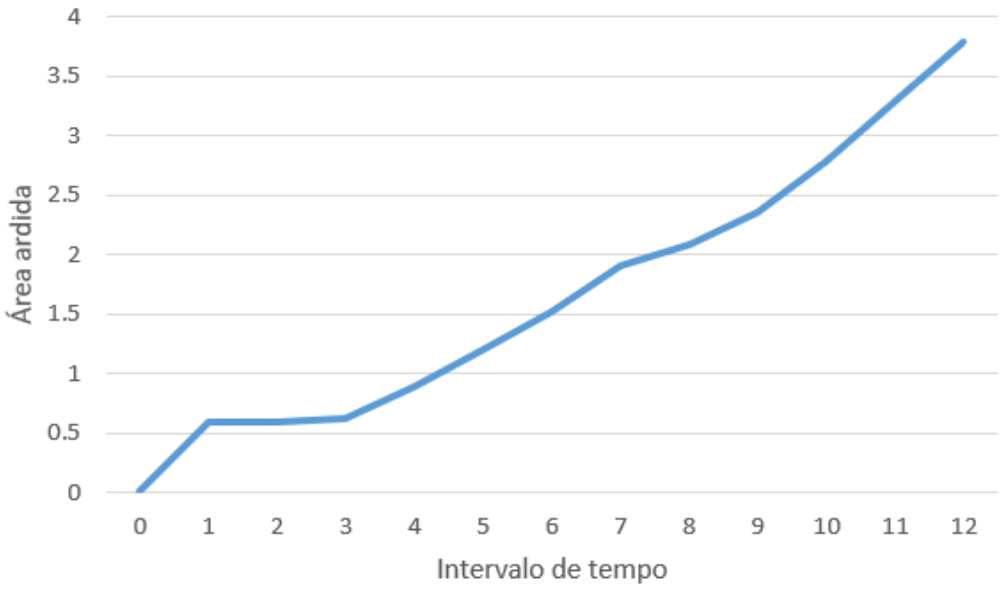

Figura 3.2: Gráfico com a variação de células ardidas segundo o tempo de incêndio

Tal como previsto, o número de células ardidas é tão maior quanto o tempo em que o incêndio deflagra, e é também visível o efeito positivo dos equipamentos alocados, que fizeram com que a área ardida cresça de forma logarítmica.