

UNIVERSIDADE DO MINHO

DEPARTAMENTO DE INFORMÁTICA

Modelos Determinísticos de Investigação Operacional

Eduardo Barbosa (a83344) Filipe Monteiro (a80229)
Bárbara Cardoso (a80453) Bruno Martins (a80410)
Márcio Sousa (a82400)

9 de Dezembro de 2018

Conteúdo

1	Questão 1	2
1.1	a)	2
1.2	c)	2
1.3	d)	4
2	Questão 2	4
2.1	a)	4
2.2	b)	5
2.3	c)	7
3	Questão 3	7
3.1	a)	7
3.2	b)	8
3.3	c)	8

1 Questão 1

Considere-se um grafo, G , que representa a rede de propagação de um incêndio florestal, sendo que A é o conjunto de arestas de G e V é o conjunto dos nodos. A cardinalidade de V , o número de nodos, é n . Cada um dos vértices corresponde a uma área que é possível ser atingida pelo fogo.

1.1 a)

Seja c_{ij} , $\forall (i, j) \in A$, o custo da aresta (i, j) e x_z , $\forall z \in V$, o tempo que o fogo demorou a chegar ao nodo z , um modelo possível é:

Função objectivo:

$$\text{Max } z = \sum x_i, \forall i \in V$$

Sujeito a:

$$x_j \leq x_i + c_{ij}, j > i$$

$$x_i \geq 0$$

$$(i, j) \in A$$

Traduzindo o modelo apresentado no enunciado para um grafo orientado com diferentes pesos associados a cada aresta, foi definida uma variável de decisão x_i que representa o tempo que o fogo demora a propagar-se desde o nodo 1 até um nodo i que varia entre 1 e 49. Desta forma, é possível deduzir que o tempo do fogo chegar a um determinado nodo i do grafo, corresponde à soma do tempo que demora a chegar a um nodo adjacente j com o tempo que demora a chegar de j a i .

b) Para determinar o dual correspondente ao problema em questão baseamo-nos no modelo primal da alínea a. Para isso seguimos um conjunto de passos para executar a conversão do qual resultou o seguinte modelo:

$$c_{ij} := \text{Custo da aresta (i,j)}$$

$$x_{ij} := \text{Numero de caminhos que usam a aresta (i,j)}$$

Função objectivo:

$$\text{Min } z = \sum c_{ij} * x_{ij}, \forall (i, j) \in A$$

Sujeito a:

$$\sum a_{ik} - a_{kj} \geq 1, (i, k) \in A \wedge (k, j) \in A$$

$$c_{ij} \geq 0$$

1.2 c)

Modelo primal no OPL:

```

// Nº de nodos
int n = ...;

//Nº de arcos
int n_arcos = ...;

//Set de
{int} Nodos = asSet(1..n);

tuple Arco{
    int i;
    int j;
}

{Arco} Arcos with i in Nodos, j in Nodos = ...;
// Tempo que demorou a chegar ao nodo i
int c_total[Arcos] = ...;

// Variável de Decisão
dvar int x[Nodos];

// Minimizar o tempo do fogo chegar a todos os nodos
//maximize max(i in R) x[i];
maximize sum (i in 1..n) x[i];

subject to {

    forall (<i,j> in Arcos) x[j] <= x[i] + c_total[<i,j>];
    x[1] == 0;

}

```

Solução do primal (nodo, tempo):

1	0
2	5
3	11
4	16
5	20
6	24
7	28
8	4
9	10
10	16
11	18
12	11
13	15
14	21
15	10
16	14
17	19
18	22
19	17
20	19
21	25
22	16
23	20
24	24
25	28
26	22
27	24
28	29
29	21
30	26
31	30
32	32
33	26
34	30
35	34
36	26
37	30
38	34
39	37
40	32
41	36
42	38
43	32
44	36
45	38
46	41
47	36
48	40
49	44

1.3 d)

Modelo dual no OPL:

```
// Nº de nodos
int n = ...;

//Set de
{int} Nodos = asSet(1..n);

tuple Arco{
    int i;
    int j;
}

{Arco} Arcos with i in Nodos, j in Nodos = ...;
// Tempo que demorou a chegar ao nodo i
int c_total[Arcos] = ...;

// Variável de Decisão
dvar int y[Nodos][Nodos];

// Minimizar o tempo do fogo chegar a todos os nodos
minimize sum (<i,j> in Arcos) c_total[<i,j>]*y[i][j];

subject to {

    forall (<i,j> in Arcos) sum(<i,k> in Arcos) y[i][k] - sum(<k,j> in Arcos) y[k][j] >= 1;

    forall(i,j in Nodos) y[i][j] >= 0;
}
```

O resultado obtido foi um erro relativo à versão utilizada que não suportava a quantidade de variáveis produzidas pelo modelo.

2 Questão 2

Para um grafo, igual ao da questão 1, pretende-se agora que o instante de chegada de um fogo a uma determinada célula seja o mais tarde possível aplicando recursos limitados a outras células retardando assim o incêndio.

2.1 a)

Seja $y_i, \forall i \in V$, a utilização de recurso na célula i , Δ a taxa de retardamento quando um recurso é aplicado a um nodo, x_i o custo de chegar a um nodo i desde o nodo de ignição, x_j o custo de chegar ao nodo j a partir de um nodo de ignição e $c_{ij} \forall (i, j) \in A$ o custo da aresta (i, j) , um modelo possível é:

Função objectivo:

$$\text{Max } z = x_p, p := \text{nodo escolhido}$$

Sujeito a:

$$\begin{aligned} x_j &\leq x_i + (\Delta * y_i) + c_{ij}, j > i \\ x_i &\geq 0 \\ y_i &= \{0, 1\} \\ y_i &\leq b \\ (i, j) &\in A \end{aligned}$$

Traduzindo o modelo apresentado anteriormente, observamos que o custo de chegada ao nodo destino é a soma de chegar ao nodo anterior i com o custo de ir de i para j . Quando a variável y_i é igual a 1 significa que vai ser aplicado um retardamento na célula i e que o custo de chegar ao nodo j vai aumentar. Visto que a função é de maximização, garantimos que retardamos o máximo possível a chegada do fogo ao nodo p . Em conjunto com as restrições é garantido que o custo de chegar a qualquer nodo não é negativo e que só é utilizado um recurso por célula e que os recursos utilizados não excedem os existentes.

2.2 b)

A passagem do modelo descrito na alínea anterior para o software de modelação *OPL* resultou no seguinte:

```
// Nº de nodos
int n = ...;
// Nº de arcos
int n_arcos = ...;
// Nº de recusos máximo a poder alocar
int b = ...;
// Constante de retardação
int delta = ...;
// Nodo a proteger o máximo possível
int p = ...;
// Nodo de ignição
int inicio = ...;

{int} Nodos = asSet(1..n);

tuple Arco{
    int i;
    int j;
}

{Arco} Arcos with i in Nodos, j in Nodos = ...;
// Tempo que demorou a chegar ao nodo i
int c_total[Arcos] = ...;

dvar boolean y[1..n];
dvar int x[Nodos];

maximize x[p];

subject to {
    forall (<i,j> in Arcos) x[j] <= x[i] + (y[i]*delta) + c_total[<i,j>];

    sum (i in Nodos) y[i] <= b;

    forall (i in Nodos) x[i]>= 0;

    x[inicio] == 0;
}
```

Figura 1: Modelo representado no OPL

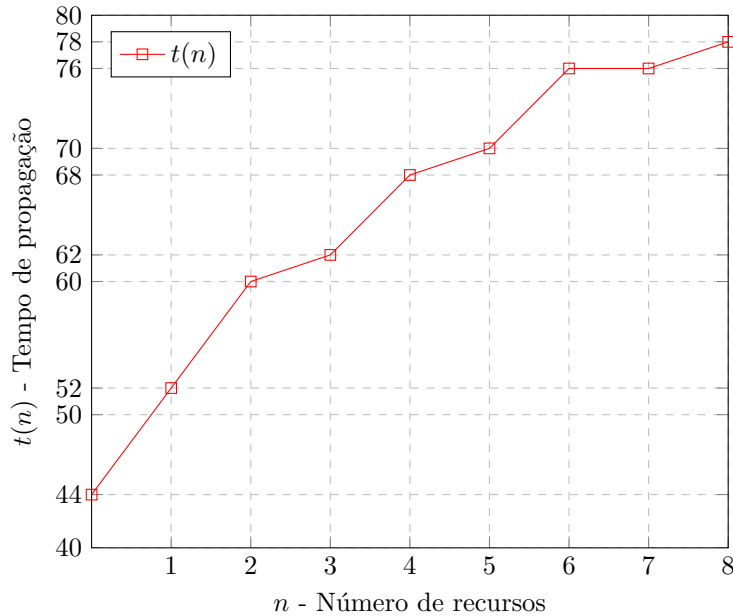
A execução deste modelo num grafo de 49 nodos com ignição no nodo nº 1 e a proteção do nodo nº 49 resultou no seguinte:

X	Valores		Y	Valores
1	0		1	1
2	13		2	1
3	27		3	0
4	32		4	0
5	36		5	0
6	42		6	0
7	46		7	0
8	12		8	1
9	26		9	0
10	32		10	0
11	33		11	0
12	27		12	1
13	39		13	1
14	47		14	0
15	26		15	0
16	30		16	0
17	35		17	0
18	38		18	0
19	41		19	0
20	47		20	0
21	53		21	0
22	32		22	0
23	36		23	0
24	40		24	0
25	44		25	0
26	40		26	1
27	52		27	0
28	57		28	0
29	36		29	0
30	42		30	0
31	46		31	0
32	49		32	0
33	52		33	0
34	56		34	0
35	60		35	0
36	41		36	0
37	45		37	0
38	50		38	0
39	54		39	0
40	58		40	0
41	62		41	0
42	64		42	1
43	42		43	0
44	47		44	0
45	52		45	0
46	58		46	0
47	62		47	0
48	66		48	1
49	78		49	0

Figura 2: Tabela resultante da execução

Podemos retirar da tabela *Y* os nodos em que foram utilizados recursos, 1, 2, 8, 12, 13, 28, 42, 48, e da tabela *X* retiramos os custos de chegar a cada nodo após a utilização dos recursos.

2.3 c)



Como podemos observar no gráfico, a utilização de recursos não é linear, logo a utilização de recursos em diferentes nodos tem como resultado retardamentos diferentes, nem sempre muito grandes.

3 Questão 3

Para um grafo igual ao da questão 1 e 2, sabendo que existem recursos para o combate ao incêndio, pretende-se agora localizar esses recursos de forma a que a área ardida seja minimizada.

3.1 a)

Seja T_i o tempo de chegada do fogo ao nodo i , R_i se o recurso é usado em i , F_i se o fogo chegou antes do tempo ao nodo i , P_i que representa a probabilidade do nodo i arder e c_{ij} que representa o custo de chegar de i a j . A partir destas variáveis surge o modelo:

Função objectivo:

$$\text{Max } z = \sum_{i=0}^n P_i * F_i$$

Sujeito a:

$$T_0 = 0$$

$$\sum_{i=0}^n R_i = b$$

$$T_j \leq T_i + (\Delta * R_i) + c_{ij}, j > i$$

$$T_i * P_i \leq k, k = \text{tempo}$$

Observando o modelo atentamente vemos que vamos maximizar o nº de nodos que não arderam, sabendo que o tempo de chegada ao nodo de ignição é zero, o nº de recursos usados tem que ser menor que b , o tempo que o fogo demora a chegar a j é o tempo de chegar a i somado ao custo de chegar de i a j retardado ou não se é usado recurso. Os nodos também não ardem se o k for maior que zero.

3.2 b)

A tradução do modelo anteriormente descrito para o OPL resulta no seguinte:

```

1  /*****
2  * OPL 12.8.0.0 Model
3  * Author: Filipe Monteiro
4  * Creation Date: Dec 7, 2018 at 1:03:03 PM
5  *****/
6
7  // Nº de nodos
8  int n = ...;
9  // Nº de recursos disponíveis
10 int b = ...;
11 // Intervalo de tempo
12 float t = ...;
13 // Probabilidade de i arder
14 float prb[1..n]=...;
15 //
16 int delta = ...;
17
18 {int} Nodos = asSet(1..n);
19
20 tuple Arco{
21     int i;
22     int j;
23 }
24
25 {Arco} Arcos with i in Nodos, j in Nodos = ...;
26 // Tempo que demorou a chegar ao nodo i
27 int c_total[Arcos] = ...;
28
29
30 // Usar recurso no nodo i ou não
31 dvar boolean y[1..n];
32 // Se nodo i ardeu antes do tempo
33 dvar boolean z[1..n];
34 // Tempo de chegada ao nodo i
35 dvar int x[1..n];
36
37 maximize sum(i in 1..n) prb[i]* (sum (j in 1..n) z[i]);
38
39 subject to{
40
41     forall(<i,j> in Arcos) x[j] <= x[i] + (delta*y[i]) + c_total[<i,j>];
42     forall(i in Nodos) x[i] <= t;
43     x[1] == 0;
44     sum (i in Nodos) y[i] <= b;
45     forall(i in Nodos) x[i] >= 0;
46 }

```

Figura 3: Modelo representado no OPL

Visto que este modelo não dá resultados que aparentem estar corretos assumimos que o modelo está errado mas fica a tentativa da modelação.


	x y z	<pre> [05556584667776577557876576565556578 76655762565571] [0000000000000000000000000000000000 00000000000000] [111111111111111111111111111111111111 11111111111110] </pre>
---	-------------	---

Figura 4: Resultado do OPL

X representa o tempo.
Y representa o uso do recurso.
Z representa se o nodo ardeu antes do tempo.

3.3 c)

Como na alínea anterior foi obtido um resultado errado não foi possível representar o gráfico.