Modelos Determinísticos de Investigação Operacional

Engenharia Informática

Universidade do Minho

Novembro 2020

DRONE

Alexandre Costa - A78890

Luís Pereira - A77667

Rúben Rodrigues - A80960

Sara Marques - A89477

Ricardo Gomes – A93785

Conteúdo

Introdução

Este projeto surge no domínio da unidade curricular Modelos Determinísticos de Investigação Operacional com o objetivo de estender a capacidade de analisar problemas complexos, desenvolver modelos e interpretar as respetivas soluções. O problema em questão advém da necessidade de um veículo não tripulado inspecionar linhas de transporte de energia elétrica em alta tensão para verificar se há vegetação a interferir com as linhas.

Desenvolvimento do modelo

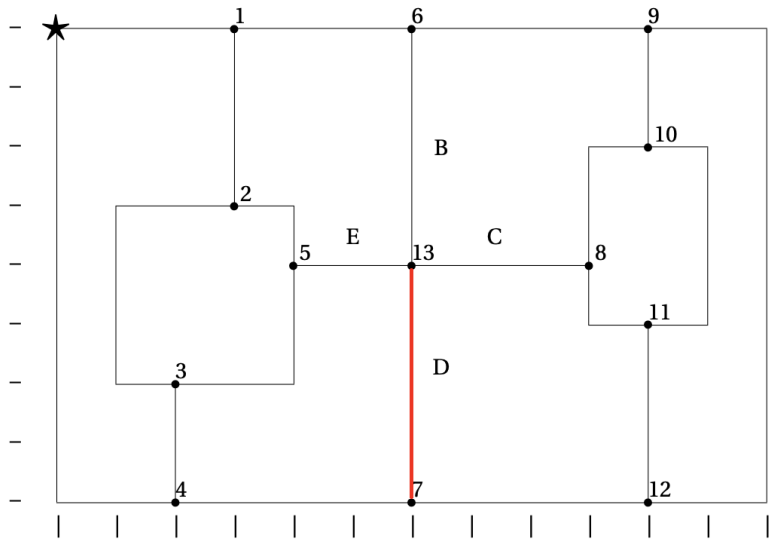
Muitos problemas práticos de Investigação Operacional podem ser expressos recorrendo a programação linear. Este problema é um excelente exemplar para aplicação deste modelo.

Definição do problema

Um problema vulgarmente discutido em Investigação Operacional é o fluxo de redes. Qualquer empresa que tem a necessidade de transportar uma determinada mercadoria de A para B, tem intenção de o realizar ao menor custo possível e no menor espaço de tempo possível, desde que cumpra as suas obrigações para com os clientes.

Neste caso concreto o problema de fluxo trata a inspeção de linhas de transporte de energia elétrica com recurso a um veículo aéreo não tripulado. O drone pode percorrer as linhas de alta tensão em ambos os sentidos e cada vez que termina uma linha pode reposicionar-se através do ar de modo a percorrer a distância mais curta. O veículo inicia o seu percurso no depósito assinalado por uma estrela e regressa ao mesmo após terminar a inspeção. Por simplificação, este problema é esquematizado sobre a forma de um grafo, sendo as arestas usadas para expressar as linhas de transporte de energia. O caminho mais curto para o reposicionamento do drone é habitualmente designada por distância euclidiana.

A figura seguinte apresenta o mapa de linhas de alta tensão. Este mapa é analisado com mais detalhe de seguida.



Análise do problema

O problema passa por minimizar a distância total percorrida num mapa de linhas de transporte de energia elétrica. O mapa reformulado de acordo com os critérios de remoção de arestas apresenta assinalado a vermelho a alteração realizada. Tanto a aresta D como o vértice 7 serão ignorados. O número de aluno utilizado para o efeito é 93785.

Uma primeira análise evidencia que todos os vértices considerados têm grau ímpar e ligam-se entre si por arestas não orientadas. Este facto leva a que se possa tirar vantagem de alterar o sentido do percurso numa mesma aresta caso seja a hipótese menos custosa no quadro final. Uma vez que as arestas são perpendiculares em relação a si próprias, as distâncias euclidianas entre a maioria dos vértices têm o mesmo custo que as próprias linhas em inspeção.

Solução do modelo

Como este problema consiste em minimizar a distância total percorrida sobre um grafo optou-se por utilizar a conservação de fluxo e restrições de integralidade e não negatividade de modo a obter soluções ideais.

Sendo este um problema de programação linear, o modelo matemático apenas contém restrições lineares.

Variáveis de decisão

Tratando-se de um problema de minimização, cujo objetivo final é encurtar o percurso a realizar, o custo dependerá sempre do número de vezes que é necessário percorrer as linhas de alta tensão, das eventuais distâncias euclidianas para reposicionamento e dos respetivos custos.

Dito isto, foram formalizados dois tipos de variáveis:

C_{ij} - representa o número de vezes a percorrer o caminho que liga os vértices i a j;

E_{ij} - representa o número de vezes a percorrer a linha reta que liga os vértices i a j;

Função objetivo

Tendo-se já decidido a estratégia a seguir, podemos então definir a função objetivo como:

$$\text{Min } \Sigma\, c_{ij} \,.\, C_{ij} + c_{ij} \,.\, E_{ij}\,,\ (i,j) \in G$$

- c_{ij} : custo da aresta, em centímetros, que liga os vértices i e j
- C_{ij} : número de vezes que a aresta, entre os vértices i e j, é atravessada
- E_{ij} : número de vezes que a linha reta, entre os vértices i e j, é realizada
- G : conjunto de todas as arestas da rede

Restrições (lineares)

As seguintes restrições implementadas regem o correto funcionamento do programa:

•Restrições de integralidade:

estas restrições garantem que a solução contém um conjunto de valores válidos, que no problema em mãos serão valores inteiros positivos.

• Restrições de não negatividade:

estas restrições garantem que todas as arestas dos grafos são utilizadas (≥1) e garantem que os caminhos através das distâncias euclidianas possam ser ou não utilizadas (≥0).

• Restrições de conservação de fluxo:

estas restrições são necessárias para se poder formar caminhos. Garantem que os vértices são balanceados, uma vez que forçam a soma do número de vezes que se entra num vértice ser igual à soma do número de vezes que se sai do mesmo.

Modelo final

O modelo matemático final é definido como:

$$\text{Min } \Sigma\, c_{ij} \,.\, C_{ij} + c_{ij} \,.\, E_{ij}\,, \qquad (i, j) \in G$$

$$\text{s.a } C_{ij} \geq 1, \qquad C_{ij} \in N^{+}$$

$$E_{ij} \geq 0, \qquad C_{ij} \in N^{+}_{0}$$

$$\Sigma\, C_{kj} + E_{kj} = \Sigma\, C_{ji} + E_{ji}\,, \qquad \{k \mid (k, i) \in G\},\ \{j \mid (i, j) \in G\}$$

Ficheiro de input

De seguida, é apresentado o ficheiro de input definido de acordo com o modelo implementado.

```
/*
   0=S,1=A,2=B,3=C,4=D,5=E,6=F,8=G,9=H,10=I,11=J,12=K,13=L
*/
/* Objective function */
min:
4 Csa + 4 Cas + 10 Csd + 10 Cds + 3 Cab + 3 Cba + 3 Caf + 3 Cfa + 6 Cbc + 6 Ccb +
2 Cbe + 2 Ceb + 2 Ccd + 2 Cdc + 4 Cce + 4 Cec + 2 Cel + 2 Cle + 8 Cdk + 8 Ckd +
4 Cfl + 4 Clf + 4 Cfh + 4 Chf + 3 Clg + 3 Cgl + 12 Chk + 12 Ckh + 2 Chi + 2 Cih +
3 Cig + 3 Cgi + 5 Cij + 5 Cji + 2 Cgj + 2 Cjg + 4 Cjk + 4 Ckj +
6.08 Eac + 6.08 Eca + 8.06 Ead + 8.06 Eda + 4.12 Eae + 4.12 Eea + 7.21 Eag + 7.21 Ega +
7.28 Eai + 7.28 Eia + 8.60 Eaj + 8.60 Eja + 10.63 Eak + 10.63 Eka + 5.00 Eal + 5.00 Ela +
3.16 Ebc + 3.16 Ecb + 5.10 Ebd + 5.10 Edb + 1.41 Ebe + 1.41 Eeb + 4.24 Ebf + 4.24 Efb +
6.08 Ebg + 6.08 Egb + 7.62 Ebh + 7.62 Ehb + 7.07 Ebi + 7.07 Eib + 7.28 Ebj + 7.28 Ejb +
8.60 Ebk + 8.60 Ekb + 3.16 Ebl + 3.16 Elb + 2.83 Ece + 2.83 Eec +
7.21 Ecf + 7.21 Efc + 7.28 Ecg + 7.28 Egc + 10.00 Ech + 10.00 Ehc +
8.94 Eci + 8.94 Eic + 8.06 Ecj + 8.06 Ejc + 8.25 Eck + 8.25 Ekc + 4.47 Ecl + 4.47 Elc +
4.47 Ede + 4.47 Eed + 8.94 Edf + 8.94 Efd + 8.06 Edg + 8.06 Egd +
11.31 Edh + 11.31 Ehd + 10.00 Edi + 10.00 Eid + 8.54 Edj + 8.54 Ejd +
5.66 Edl + 5.66 Eld + 4.47 Eef + 4.47 Efe + 7.21 Eeh + 7.21 Ehe +
6.32 Eei + 6.32 Eie + 6.08 Eej + 6.08 Eje + 7.21 Eek + 7.21 Eke +
5.00 Efg + 5.00 Egf + 4.47 Efi + 4.47 Eif + 6.40 Efj + 6.40 Ejf +
8.94 Efk + 8.94 Ekf + 4.12 Egh + 4.12 Ehg + 2.24 Egi + 2.24 Eig +
1.41 Egj + 1.41 Ejg + 4.12 Egk + 4.12 Ekg + 5.00 Ehj +
5.00 Ejh + 8.00 Ehk + 8.00 Ekh +5.66 Ehl + 5.66 Elh +
3.00 Eij + 3.00 Eji + 6.00 Eik + 6.00 Eki + 4.47 Eil + 4.47 Eli +
4.12 Ejl + 4.12 Elj + 5.66 Ekl + 5.66 Elk;
```

```
/* Garantir que cada aresta e visitada pelo menos 1 vez */

Csa + Cas >= 1;
Csd + Cds >= 1;
Cab + Cba >= 1;
Caf + Cfa >= 1;
Cbc + Ccb >= 1;
Cbe + Ceb >= 1;
Ccd + Cdc >= 1;
Cce + Cec >= 1;
Cel + Cle >= 1;
Cdk + Ckd >= 1;
Cfl + Clf >= 1;
Cfh + Chf >= 1;
Clg + Cgl >= 1;
Chk + Ckh >= 1;
Chi + Cih >= 1;
Cig + Cgi >= 1;
Cij + Cji >= 1;
Cgj + Cjg >= 1;
Cjk + Ckj >= 1;
```

```
/*Caminhos pelo ar */

Eac + Eca >= 0;Ead + Eda >= 0;
Eae + Eea >= 0;Eag + Ega >= 0;
Eai + Eia >= 0;Eaj + Eja >= 0;
Eak + Eka >= 0;Eal + Ela >= 0;

Ebc + Ecb >= 0;Ebd + Edb >= 0;
Ebe + Eeb >= 0;Ebf + Efb >= 0;
Ebg + Egb >= 0;Ebh + Ehb >= 0;
Ebi + Eib >= 0;Ebj + Ejb >= 0;
Ebk + Ekb >= 0;Ebl + Elb >= 0;

Ece + Eec >= 0;Ecf + Efc >= 0;
Ecg + Egc >= 0;Ech + Ehc >= 0;
Eci + Eic >= 0;Ecj + Ejc >= 0;
Eck + Ekc >= 0;Ecl + Elc >= 0;

Ede + Eed >= 0;Edf + Efd >= 0;
Edg + Egd >= 0;Edh + Ehd >= 0;
Edi + Eid >= 0;Edj + Ejd >= 0;
Edl + Eld >= 0;

Eef + Efe >= 0;Eeh + Ehe >= 0;
Eei + Eie >= 0;Eej + Eje >= 0;
Eek + Eke >= 0;

Efg + Egf >= 0;Efi + Eif >= 0;
Efj + Ejf >= 0;Efk + Ekf >= 0;

Egh + Ehg >= 0;Egi + Eig >= 0;
Egj + Ejg >= 0;Egk + Ekg >= 0;

Ehj + Ejh >= 0;Ehk + Ekh >= 0;
Ehl + Elh >= 0;

Eij + Eji >= 0;Eik + Eki >= 0;
Eil + Eli >= 0;

Ejl + Elj >= 0;
```

```
/* Decisoes de percurso para cada aresta */
ToS: Cas + Cds -Csa -Csd =0;
ToA: Csa + Cba + Cfa + Eca + Eda + Eea + Ega +Eia + Eja + Eka + Ela - Cas - Cab - Caf - Eac - Ead - Eae - Eag - Eai - Eaj - Eak - Eal =0;
ToB: Cab + Ceb + Ccb +Ecb +Edb + Eeb + Efb + Egb + Ehb + Eib + Ejb + Ekb + Elb - Cba - Cbe - Cbc - Ebc - Ebd - Ebe - Ebf - Ebg - Ebh - Ebi - Ebj - Ebk - Ebl =0;
ToC: Cbc + Cdc + Cec + Eac +Ebc +Eec +Efc + Egc + Ehc+ Eic + Ejc + Elc - Ccb - Ccd - Cce - Eca - Ecb - Ece - Ecf - Ecg - Ech - Eci - Ecj - Ecl =0;
ToD: Csd + Ccd + Ckd + Ead + Ebd + Eed + Efd + Egd + Ehd + Eid + Ejd + Eld - Cds - Cdc - Cdk - Eda - Edb - Ede - Edf - Edg - Edh - Edi - Edj - Edl =0;
ToE: Cbe + Cce + Cle + Eae +Ebe +Ece + Ede + Efe + Ehe + Eie + Eje + Eke - Ceb - Cec - Cel - Eea - Eeb - Eec - Eeb - Eef - Eeh - Eei - Eej - Eek =0;
ToF: Caf + Chf + Clf +Ebf +Ecf +Edf +Eef +Egf +Eif +Ejf +Ekf - Cfa - Cfh - Cfl - Efb - Efc - Efd - Efe - Efg - Efi - Efj - Efk =0;
ToG: Clg + Cjg +Cig +Eag +Ebg +Ecg +Edg +Efg +Ehg +Eig +Ejg +Ekg - Cgl - Cgj - Cgi - Ega - Egb - Egc - Egd - Egf - Egh - Egi - Egj - Egk =0;
ToH: Cfh +Cih +Ckh +Ebh +Ech +Edh +Eeh +Egh +Ejh +Ekh +Elh  - Chf - Chi - Chk - Ehb - Ehc - Ehd - Ehe - Ehg - Ehj - Ehk - Ehl =0;
ToI: Chi +Cgi +Cji +Eai +Ebi +Eci +Edi +Eei +Efi +Egi +Eji +Eki +Eli - Cih - Cig - Cij - Eia - Eib - Eic - Eid - Eie - Eif - Eig - Eij - Eik - Eil =0;
ToJ: Cgj +Cij +Ckj +Eaj +Ebj +Ecj +Edj +Eej +Efj +Egj +Ehj +Eij +Elj - Cjg - Cji - Cjk - Eja - Ejb - Ejc - Ejd - Eje - Ejf - Ejg - Ejh - Eji - Ejl =0;
ToK: Cjk +Chk +Cdk +Eak +Ebk +Eck +Eek +Efk +Egk +Ehk +Eik +Elk - Ckj - Ckh - Ckd - Eka - Ekb - Ekc - Eke - Ekf - Ekg - Ekh - Eki - Ekl =0;
ToL: Cgl +Cfl +Cel + Eal +Ebl +Ecl +Edl +Ehl +Eil +Ejl +Ekl - Cgl - Cfl - Cel - Eal - Ebl - Ecl - Edl - Ehl - Eil - Ejl - Ekl =0;
```

```
/* Integrity restrictions */

Int Csa , Cas,
Csd , Cds, Cab , Cba, Caf , Cfa,
Cbc , Ccb, Cbe , Ceb, Ccd , Cdc,
Cce , Cec, Cel , Cle, Cdk , Ckd,
Cfl , Clf, Cfh , Chf, Clg , Cgl,
Chk , Ckh, Chi , Cih, Cig , Cgi,
Cij , Cji, Cgj , Cjg, Cjk , Ckj;


Int Eac,Eca,Ead,Eda,Eae,
Eea,Eag,Ega,Eai,Eia,Eaj,Eja,Eak,Eka,Eal,Ela;

Int Ebc , Ecb, Ebd , Edb,
Ebe , Eeb, Ebf , Efb,
Ebg , Egb, Ebh , Ehb,
Ebi , Eib, Ebj , Ejb,
Ebk , Ekb, Ebl , Elb,

Ece , Eec, Ecf , Efc,
Ecg , Egc, Ech , Ehc,
Eci , Eic, Ecj , Ejc,
Eck , Ekc, Ecl , Elc,

Ede , Eed, Edf , Efd,
Edg , Egd, Edh , Ehd,
Edi , Eid, Edj , Ejd, Edl , Eld,

Eef , Efe, Eeh , Ehe,
Eei , Eie, Eej , Eje, Eek , Eke,

Efg , Egf, Efi , Eif,
Efj , Ejf, Efk , Ekf,

Egh , Ehg, Egi , Eig,
Egj , Ejg, Egk , Ekg,

Ehj , Ejh, Ehk , Ekh, Ehl , Elh,

Eij , Eji, Eik , Eki, Eil , Eli,

Ejl , Elj,

Ekl , Elk;
```

Ficheiro de output

Utilizando o método simplex, o lpsolve atribui os seguintes resultados como sendo ótimos:

Objective | Constraints | Sensitivity

Variables	MILP ...	MILP ...	MILP ...	MILP ...	re...
	103.4	102.24	99	98.41	98.41
Clg	1	1	2	2	2
Cih	1	0	1	2	2
Csa	1	1	1	1	1
Cds	1	1	1	1	1
Cab	1	1	1	1	1
Caf	1	1	1	1	1
Cfa	1	0	1	1	1
Cbc	1	1	1	1	1
Ceb	0	1	1	1	1
Ccd	1	1	1	1	1
Cdc	0	1	1	1	1
Cce	1	1	1	1	1
Cel	1	1	1	1	1
Ckd	1	1	1	1	1
Cfl	1	1	1	1	1
Chf	1	1	1	1	1
Chk	0	0	0	1	1
Cgi	0	0	1	1	1
Cji	1	0	0	1	1
Cgj	1	1	1	1	1
Cjk	1	2	2	1	1
Ckj	0	0	0	1	1
Ebe	0	0	0	1	1
Cas	0	0	0	0	0
Csd	0	0	0	0	0
Cba	0	0	0	0	0
Ccb	0	0	0	0	0
Cbe	1	1	1	0	0
Cec	0	0	0	0	0
Cle	0	0	0	0	0
Cdk	1	0	0	0	0
Clf	0	0	0	0	0
Cfh	0	1	0	0	0
Cgl	0	0	0	0	0

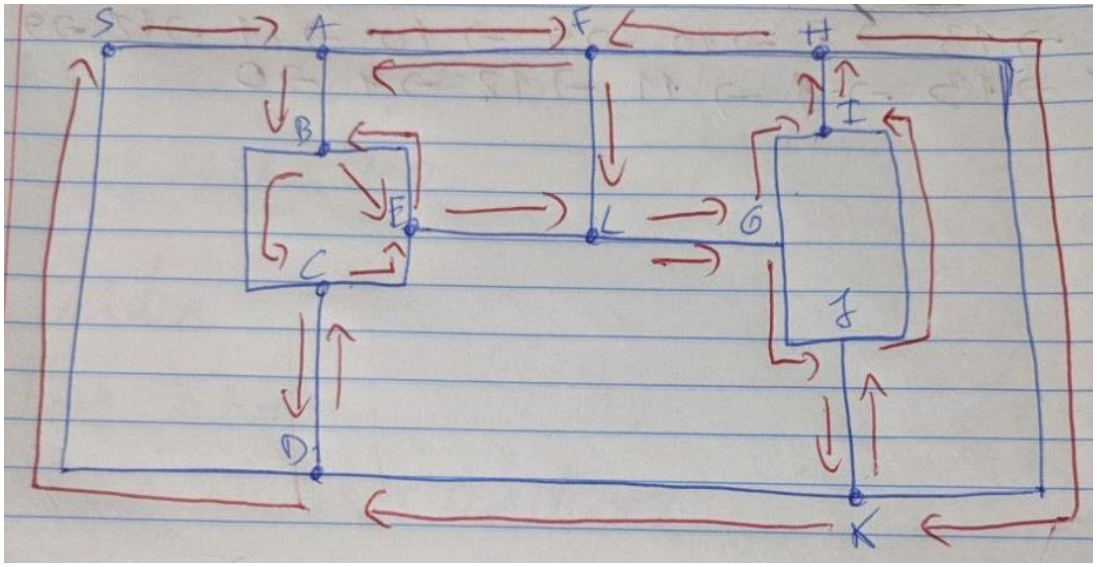
Validação do modelo

Na figura seguinte observa-se a tabela utilizada para o cálculo dos reposicionamentos aéreos.

		x	3	3	2	2	4	6	6	9	10	10	10	10	6
		y	8	5	2	0	4	8	0	4	8	6	3	0	4
			1	2	3	4	5	6	7	8	9	10	11	12	13
3	8	1	0.00		6.08	8.06	4.12			7.21		7.28	8.60	10.63	5.00
3	5	2		0.00	3.16	5.10	1.41	4.24		6.08	7.62	7.07	7.28	8.60	3.16
2	2	3	6.08	3.16	0.00		2.83	7.21		7.28	10.00	8.94	8.06	8.25	4.47
2	0	4	8.06	5.10		0.00	4.47	8.94		8.06	11.31	10.00	8.54		5.66
4	4	5	4.12	1.41	2.83	4.47	0.00	4.47			7.21	6.32	6.08	7.21	
6	8	6		4.24	7.21	8.94	4.47	0.00		5.00		4.47	6.40	8.94	
6	0	7							0.00						
9	4	8	7.21	6.08	7.28	8.06		5.00		0.00	4.12	2.24	1.41	4.12	
10	8	9		7.62	10.00	11.31	7.21			4.12	0.00		5.00	8.00	5.66
10	6	10	7.28	7.07	8.94	10.00	6.32	4.47		2.24		0.00	3.00	6.00	4.47
10	3	11	8.60	7.28	8.06	8.54	6.08	6.40		1.41	5.00	3.00	0.00		4.12
10	0	12	10.63	8.60	8.25		7.21	8.94		4.12	8.00	6.00		0.00	5.66
6	4	13	5.00	3.16	4.47	5.66					5.66	4.47	4.12	5.66	0.00

Comparando a tabela com o mapa de linhas de alta tensão é possível verificar que foram ignorados os custos cuja aresta a inspecionar correspondia ao custo dito euclidiano.

A solução ótima obtida pelo lpsolve leva-nos à análise da figura seguinte



O caminho ideal será então:

S -> A -> F -> A -> B -> C -> D -> C -> E -> B

-> E -> L -> G -> J -> k -> J -> I -> H -> F ->

L -> G -> I -> H -> k -> d -> S

Com um custo total de 98.41.

Podemos afirmar que a solução ótima é admissível uma vez que cumpre com os requisitos definidos. Todas as arestas são percorridas pelo menos uma vez, o caminho pelo ar é utilizado de forma a reduzir o custo total e as restantes restrições verificadas no ficheiro de output são também verdadeiras. A solução ótima foi utilizada para descrever um percurso a seguir, podendo existir outros circuitos. Outras possíveis soluções passariam por ter um custo total igual.

Discussão

Neste capítulo serão apresentadas as dificuldades encontradas, bem como a justificação de determinadas decisões.

Num contexto de programação linear, procura-se primeiro encontrar o modelo que melhor descreve o problema, quer seja de maximização ou minimização. Neste caso em concreto deparámo-nos com um problema de minimização de custos. Como a solução tinha de passar necessariamente por percorrer todas as arestas do mapa, a primeira variável de decisão ideal seria o número de vezes que cada aresta era atravessada.

Mas este problema tem uma nuance, o drone que realiza a inspeção das linhas pode simplesmente sobrevoar o mapa e realizar o deslocamento mais curto que o permitia reposicionar-se. Aqui encontrámos a primeira dificuldade. O número de vértices possíveis de reposicionamento gerava um elevado número de restrições. Decidimos então remover todas as restrições de distâncias euclidianas cujo custo igualava o custo das próprias linhas. Esta situação gera-se quando as arestas entre cada par de vértices é um segmento de linha reta. A distância euclidiana levou à necessidade de se definir a segunda e última variável de decisão.

Esta abordagem leva a que a solução ótima aparente utilizar apenas um deslocamento pelo ar. Era provável que utilizando todas as distâncias euclidianas surgissem mais deslocamentos pelo ar, uma vez que as arestas podiam já ter sido examinadas e não necessitassem de nova inspeção.

Conclusão

Programação Linear é um campo da otimização por várias razões. Problemas de investigação operacional podem ser expressos como modelos de programação linear, para determinar a solução de, por exemplo, o fluxo de redes de transportes.

Este projeto modela uma variante destes problemas em que é necessário percorrer todas as arestas de um mapa orientado sobre um grafo, minimizando o seu custo.

Após uma análise detalhada do problema, decidiu-se a definição matemática do modelo sobre o grafo que melhor se enquadrava.

Seguidamente, com recurso ao lpsolve obteve-se a solução ótima do problema.

O modelo apresentado modela qualquer problema do mundo real, desde que contemple que o mapa possa ser sobrevoado.

Referências

Valério de Carvalho J.M., “Programação Linear - modelos - Investigação Operacional”, 2 de outubro de 2020