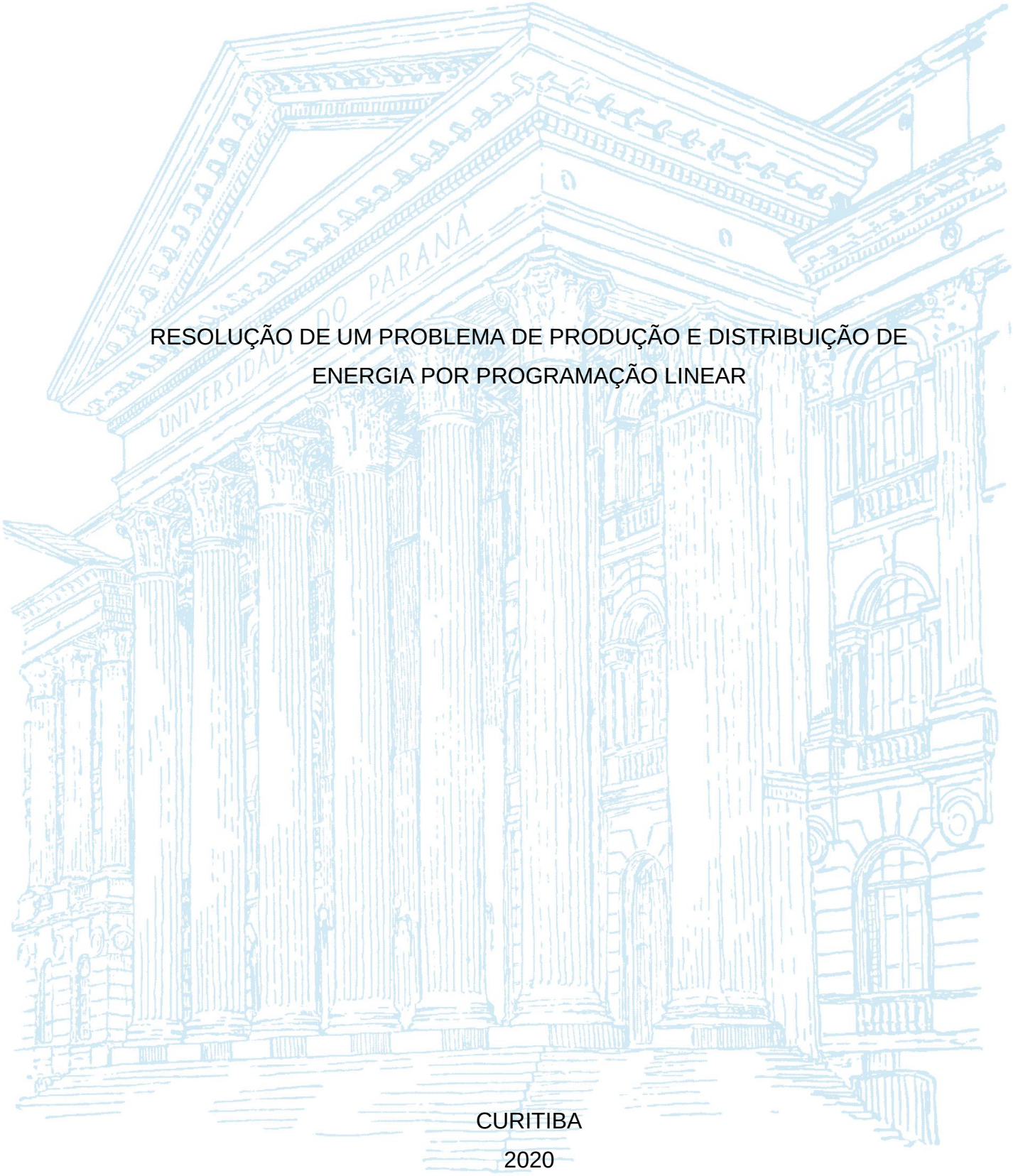
UNIVERSIDADE FEDERAL DO PARANÁ

FERNANDO ZANUTTO MADY BARBOSA



RESOLUÇÃO DE UM PROBLEMA DE PRODUÇÃO E DISTRIBUIÇÃO DE ENERGIA POR PROGRAMAÇÃO LINEAR

CURITIBA

2020

FERNANDO ZANUTTO MADY BARBOSA

RESOLUÇÃO DE UM PROBLEMA DE PRODUÇÃO E DISTRIBUIÇÃO DE ENERGIA POR PROGRAMAÇÃO LINEAR

Artigo apresentado como requisito parcial à conclusão da disciplina de Otimização/Tópicos em Algoritmos, Setor de Informática, Universidade Federal do Paraná.

Professor: Prof. Dr. André Luiz Pires Guedes

CURITIBA

2020

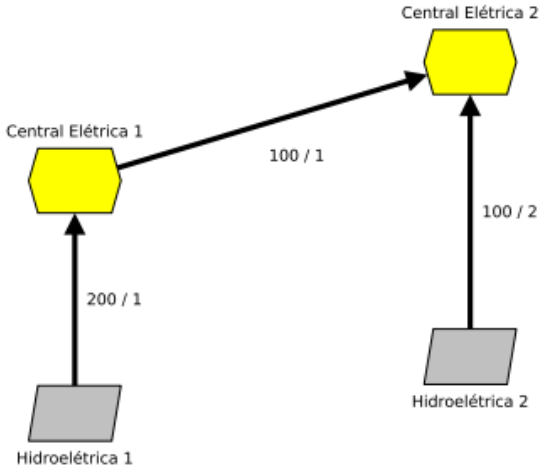
1 INTRODUÇÃO

O problema sugerido pelo orientador consiste em uma rede de produção e distribuição de energia. Há certa quantidade de hidroelétricas disponíveis na rede, as quais devem transmitir, total ou parcialmente, sua produção de energia às centrais elétricas. Para realizar tal transmissão utiliza-se uma ligação direcional, chamada aqui de arco. Uma hidroelétrica só pode enviar energia (para uma ou mais central), nunca receber. As centrais elétricas podem receber ou transmitir de/para outra central, ou receber de uma hidroelétrica. Quanto aos arcos, estes têm uma capacidade e custo por unidade de transmissão, separados respectivamente por "/".

Detalhes técnicos:

- A produção da hidroelétrica é calculada pelo produto de sua eficiência energética pela vazão do rio escolhida.
- O custo de produção da hidroelétrica é calculado pelo produto de seu custo, por unidade energética, pela vazão escolhida.
- O rio tem a mesma vazão máxima para todas as hidroelétricas.
- O custo de transmissão de um arco é calculado pelo produto de seu custo, por unidade energética, pela energia transmitida.

FIGURA 1 – EXEMPLO DE REDE ELÉTRICA



Fonte: Prof. Dr. André Guedes (2020)

Uma resposta viável para o problema deve atender as seguintes restrições:

- Produção da hidroelétrica é limitada por sua capacidade
- A vazão escolhida de uma hidroelétrica não pode ser maior que a vazão máxima do rio
- Um arco não pode transmitir uma quantidade energética maior que sua capacidade
- A demanda de uma central deve ser atendida
- A saída energética de uma hidroelétrica não pode ser maior que sua produção

1.1 VARIÁVEIS DO PROBLEMA

Neste capítulo apresentam-se as varáveis que serão usadas para posterior modelagem do problema. Caso o leitor se confunda sobre a notação usada pelo autor é fortemente recomendado que volte a este capítulo.

Variáveis gerais:

- h = quantidade de hidroelétricas
- L = quantidade de centrais elétricas
- a = quantidade de arcos

Variáveis referentes às hidroelétricas ($1 \leq i \leq h$):

- c_i = custo de produção por unidade da hidroelétrica i
- m_i = capacidade máxima de produção da hidroelétrica i
- v_i = vazão escolhida para a hidroelétrica i
- f_i = eficiência energética da hidroelétrica i
- $outputs_i$ = lista dos arcos que saem da hidroelétrica i

Variáveis referentes às centrais elétricas ($1 \leq i \leq L$):

- d_i = demanda energética da central i
- $inputs_i$ = lista dos arcos que entram na central i
- $outputs_i$ = lista dos arcos que saem da central i

Variáveis referentes aos arcos ($1 \leq i \leq a$):

- c'_i = custo, por unidade, da transmissão pelo arco i
- e_i = energia transmitida pelo arco i
- E_i = capacidade máxima de transmissão do arco i

Das variáveis descritas acima, apenas as vazões v_i de cada hidroelétrica e as energias e_i de cada arco não são dados pela entrada, portanto o problema da rede será construído e resolvido a partir dessas variáveis não fixas.

1.2 RESPOSTA ÓTIMA PARA O PROBLEMA

O problema descrito neste artigo será resolvido de acordo com as técnicas de Programação Linear. Nesta área, uma das principais preocupações é encontrar uma solução ótima para o problema. Na rede elétrica descrita, a solução ótima é aquela que respeita as restrições e o custo total é mínimo. Felizmente, diversos autores descrevem algoritmos muito concisos que calculam de forma eficaz a solução ótima. O algoritmo que será usado nos próximos capítulos é o Simplex. Contudo, neste artigo não iremos explicar por que o método Simplex funciona ou como, apenas vamos utilizá-lo. Para um leitor curioso recomenda-se o livro Understanding and Using Linear Programming, de Jiří Matoušek, Bernd Gärtner.

O Simplex que utilizaremos é a função *linprog* da biblioteca scipy da linguagem *Python*. Dada a função custo, criamos um vetor contendo os coeficientes das variáveis. No exemplo da Figura 1 o vetor será c = [-1, 4]. Para as restrições, criamos um vetor de vetores contendo os coeficientes do lado esquerdo da inequação, portanto A = [[-3, 1], [1, 2]]. Note que a segunda inequação foi multiplicada por -1 para termos o sinal ≤, de acordo com o padrão de uso. Criamos outro vetor com os limitantes de cada inequação, ou seja, b = [6, 4]. Para cada variável precisamos de seu limitante superior e inferior, portanto x_0_bounds = (None, None), pois não há limitante, e x_1_bounds = (-3, None).

FIGURA 2 – EXEMPLO DE USO DO SIMPLEX

$$\begin{aligned} \min_{x_0, x_1} \quad & -x_0 + 4x_1 \\ \text{such that} \quad & -3x_0 + x_1 \leq 6, \\ & -x_0 - 2x_1 \geq -4, \\ & x_1 \geq -3. \end{aligned}$$

```
>>> c = [-1, 4]
>>> A = [[-3, 1], [1, 2]]
>>> b = [6, 4]
>>> x0_bounds = (None, None)
>>> x1_bounds = (-3, None)
>>> from scipy.optimize import linprog
>>> res = linprog(c, A_ub=A, b_ub=b, bounds=[x0_bounds, x1_bounds])
```

FONTE: Documentação da biblioteca scipy (2020)

Se há uma solução ótima para o problema, esta estará contida em res.x, como na Figura 3.

Figura 3 – EXEMPLO DE RESPOSTA DO SIMPLEX

```
>>> print(res)
     con: array([], dtype=float64)
     fun: -21.99999984082494 # may vary
 message: 'Optimization terminated successfully.'
     nit: 6 # may vary
   slack: array([3.89999997e+01, 8.46872439e-08] # may vary
  status: 0
 success: True
       x: array([ 9.99999989, -2.99999999]) # may vary
```

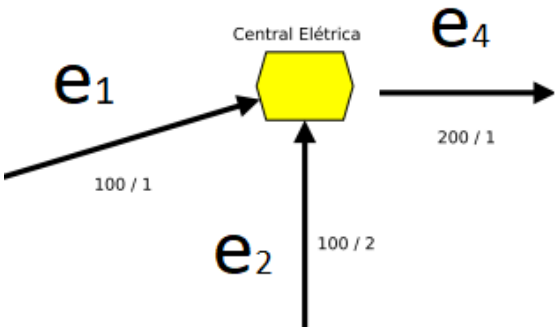
FONTE: Documentação da biblioteca scipy (2020)

2 MODELAGEM DAS RESTRIÇÕES

É necessário traduzir as restrições descritas no capítulo 1.1 para a notação adotada para posterior uso do simplex. As três primeiras equações são facilmente traduzidas, porém as duas últimas necessitam um olhar mais atento.

A inequação 4 assegura que a demanda da central será atendida. A subtração da energia total de entrada pela energia total de saída determina o valor “que fica” na central elétrica, que deverá ser ao menos o valor da demanda de energia. E para determinar os valores de entrada de energia da central checamos a lista *inputs*, se um determinado arco pertencer à lista então a energia deste arco deve ser somada à energia total de entrada. A mesma relação ocorre com os valores de saída de energia da central, se um arco pertencer à lista *outputs* da central, então o valor deste deve ser somado à energia total de saída.

FIGURA 4 – RESTRIÇÕES DE CENTRAIS

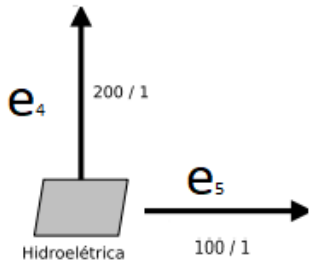


FONTE: O autor (2020).

Na Figura 4, a central elétrica tem a lista de *inputs* = $[1, 2]$ e lista de *outputs* = $[4]$. Portanto o valor de energia dentro da central será $(e_1 + e_2) - (e_4)$, que deverá ser maior ou igual à demanda energética da central.

A inequação 5 assegura que uma hidroelétrica não transmitirá mais do que produz, ou seja, a energia total de saída deve ser menor ou igual à produção elétrica. Os valores de saída da hidroelétrica são determinados checando a lista de *outputs*, se um arco pertencer à lista, este deve ser somado à energia total de saída.

FIGURA 5 – RESTRIÇÕES DE HIDROELÉTRICAS



FONTE: O autor (2020).

Na Figura 5, a hidroelétrica tem a lista de *outputs* = [4, 5]. Portanto o valor da energia total de saída será $(e_4 + e_5)$, que deverá ser menor ou igual à produção da hidroelétrica.

QUADRO 1 – RESTRIÇÕES TRADUZIDAS PARA INEQUAÇÕES

Inequação 1	$P_i \leq M_i$	Produção da hidroelétrica é limitada por sua capacidade	$1 \leq i \leq h$
Inequação 2	$v_i \leq R$	A vazão escolhida de uma hidroelétrica não pode ser maior que a vazão máxima do rio	$1 \leq i \leq h$
Inequação 3	$e_i \leq E_i$	Um arco não pode transmitir uma quantidade energética maior que sua capacidade	$1 \leq i \leq a$
Inequação 4	$\sum e_x - \sum e_y \geq d_i$	A demanda de uma central deve ser atendida	$x \in inputs_i$ $y \in outputs_i$ $1 \leq i \leq L$
Inequação 5	$\sum e_y \leq P_i$	A saída energética de uma hidroelétrica não pode ser maior que sua produção	$y \in outputs_i$ $1 \leq i \leq h$

FONTE: O autor (2020).

2.1 FUNÇÃO DE CUSTO

Como parte de achar uma solução ótima para o problema da rede elétrica, é necessário minimizar o custo total. O custo total é formado pelo custo de produção energética de cada hidroelétrica somado ao custo de transmissão de cada arco.

Minimizar: $\sum_{i=1}^{h} c_i * v_i + \sum_{i=1}^{a} c_i^{'} * e_i$

3 FORMATO DE ENTRADA E SAÍDA

A primeira linha contém três números, representando h, L e R, seguidos de h conjuntos de 3 números representando as capacidades de produção (m_i), as eficiências energéticas (f_i) e os custos de produção (c_i). Logo após, temos L números representando as demandas das centrais elétricas (d_i). Depois destes temos a descrição da rede. Temos h + L blocos um para cada vértice da rede (h

hidroelétricas e L centrais). Cada bloco começa com um numero de vizinhos de saída (n_i) seguido de n_i triplas representando o índice da central para onde vai a conexão ($t_{i,j}$), a capacidade da conexão ($E_{i,j}$) e o custo ($c'_{i,j}$). Como descrito abaixo.

QUADRO 2 – FORMATO DE ENTRADA

H	L	R	//variáveis gerais
m_1	f_i	c_i	//dados da hidroelétrica 1
…	…	…	
m_h	f_h	c_h	//dados da hidroelétrica h
d_1			//demanda da central 1
…	…	…	
d_L			//demanda da central h
n_1			//número de vizinhos da hidroelétrica 1
$t_{1,1}$	$E_{1,1}$	$c'_{1,1}$	//1º vizinho, capacidade e custo
…	…	…	
t_{1,n_1}	E_{1,n_1}	c'_{1,n_1}	
…	…	…	
n_h			
$t_{h,1}$	$E_{h,1}$	$c'_{h,1}$	
…	…	…	
t_{h,n_h}	E_{h,n_h}	c'_{h,n_h}	
n_{h+1}			//número de vizinhos da central 1
$t_{h+1,1}$	$E_{h+1,1}$	$c'_{h+1,1}$	//1º vizinho, capacidade e custo
…	…	…	
n_{h+L}			
$t_{h+L,1}$	$E_{h+L,1}$	$c'_{h+L,1}$	
…	…	…	
$t_{h+L,n_{h+L}}$	$E_{h+L,n_{h+L}}$	$c'_{h+L,n_{h+L}}$	

FONTE: O autor (2020).

Saída: valores da produção de cada hidroelétrica (Pi), um por linha, na ordem em que foram declaradas na entrada e uma linha para cada arco (também na ordem da entrada) com a carga transmitida por ele.

4. UM EXEMPLO DE REDE ELÉTRICA

Neste capítulo vamos modelar uma rede dadas as informações de entrada da Figura 6.

FIGURA 6 – EXEMPLO DE ENTRADA

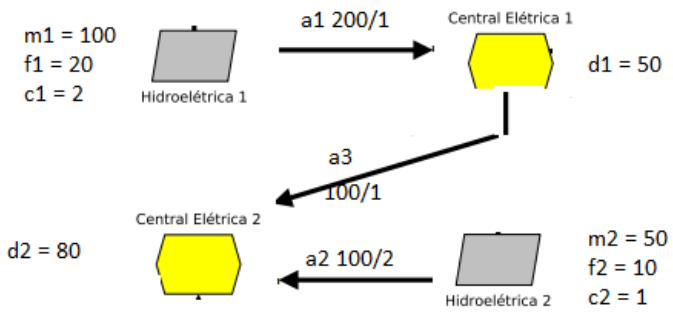
Entrada:

```
2 2 10
100 20 2
50 10 1
50
80
1
1 200 1
1
2 100 2
1
2 100 1
0
```

Fonte: Prof. Dr. André Guedes (2020)

Segundo os dados da entrada de exemplo do Quadro 2, rede construída será como na Figura 7.

FIGURA 7 – REDE RESULTANTE



FONTE: O autor (2020).

De acordo com o Quadro 1 de restrições e as variáveis dadas na Figura 6, a rede deve atender às seguintes restrições:

Inequação 1:

- $P_1 = f_1 * v_1 \leq M_1 \therefore v_1 \leq 5$
- $P_2 = f_2 * v_2 \leq M_2 \therefore v_2 \leq 5$

Inequação 2:

- $v_1, v_2 \leq R$ = 10

Inequação 3:

- $e_1 \leq 200$
- $e_2 \leq 100$
- $e_3 \leq 100$

Inequação 4:

- Referente à central 1: $e_1 - e_3 \geq 50 \therefore -e_1 + e_3 \leq -50$
- Referente à central 2: $e_3 + e_2 \geq 80 \therefore -e_2 - e_3 \leq -80$

Inequação 5:

- Referente à hidroelétrica 1: $e_1 \leq P_1 = 20 * v_1 \therefore -20v_1 + e_1 \leq 0$
- Referente à hidroelétrica 2: $e_2 \leq P_2 = 10 * v_2 \therefore -10v_2 + e_2 \leq 0$

A função de custo, como já comentamos, varia conforme as escolhas de vazões e de energias transmitidas, portanto:

Função custo: $2v_1 + 1v_2 + 1e_1 + 2e_2 + 1e_3$

Considerando $(v_1, v_2, e_1, e_2, e_3) = (x_0, x_1, x_2, x_3, x_4)$, os vetores explicados na Seção 1.2 serão:

- C = [2, 1, 1, 2, 1]
- A = [[-20, 0, 1, 0, 0], [0, -10, 0, 1, 0], [0, 0, -1, 0, 1], [0, 0, 0, -1, -1]]
- b = [0, 0, -50, -80]
- x_0_bounds = [0, min(5, 10)]
- x_1_bounds = [0, min(5, 10)]

- x_2_bounds = [0, 200]
- x_3_bounds = [0, 100]
- x_4_bounds = [0, 100]

Considere min(5, 10) como uma função que retorna valor mais restringente, no caso 5. Após inserir tais vetores no *Simplex*, a solução ótima será:

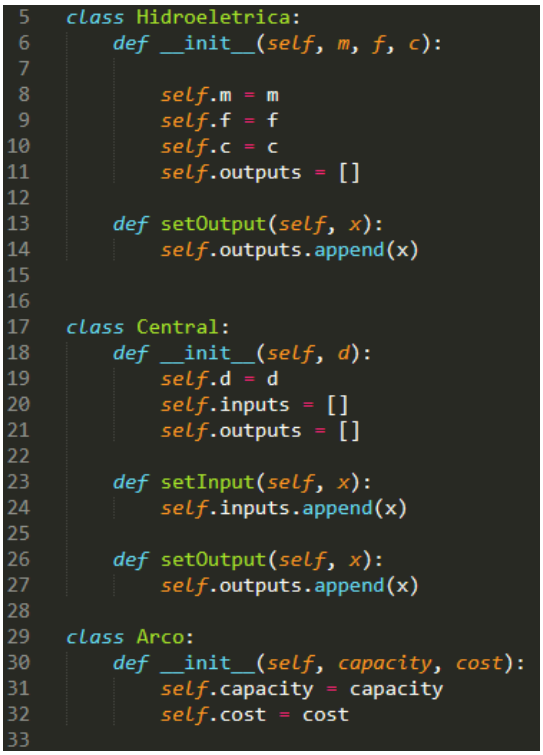
- 80
- 50
- 80
- 50
- 30

5. IMPLEMENTAÇÃO

5.1 CLASSES

Como a implementação do problema de rede será feita na linguagem Python, utilizaremos algumas características da Programação Orientada a objetos. Na Figura 7 foi criada uma classe para cada objeto real do problema, ou seja, hidroelétricas, centrais elétricas e arcos, com seus atributos. As listas *inputs* e *outputs* serão incrementadas posteriormente com o índice dos arcos, com as funções *setInput* e *setOutput*.

FIGURA 8 – CLASSES IMPLEMENTADAS



```
5   class Hidroeletrica:
6       def __init__(self, m, f, c):
7
8           self.m = m
9           self.f = f
10          self.c = c
11          self.outputs = []
12
13      def setOutput(self, x):
14          self.outputs.append(x)
15
16
17  class Central:
18      def __init__(self, d):
19          self.d = d
20          self.inputs = []
21          self.outputs = []
22
23      def setInput(self, x):
24          self.inputs.append(x)
25
26      def setOutput(self, x):
27          self.outputs.append(x)
28
29  class Arco:
30      def __init__(self, capacity, cost):
31          self.capacity = capacity
32          self.cost = cost
33
```

FONTE: O autor (2020).

5.2 CONVERSÃO DA ENTRADA

Neste capítulo convertemos a entrada dada como no Quadro 2 para uma lista de hidroelétricas, centrais e arcos. Na linha 142 e 143 da figura 9 apenas pegamos os dados da entrada padrão STDIN e inserimos linha a linha na lista *inputLines*, de forma que cada elemento de *inputLines* é uma *string* equivalente a uma linha de dados. Nas linhas 146 a 149, a primeira linha de entrada é separada em variáveis do tipo *string* h, l, R que transformamos em variáveis inteiras com “int()”. Nas linhas 151 a 153 são criadas listas que terão como elementos os objetos Hidroeletrica, Central e Arco.

FIGURA 9 – LENDO A PRIMEIRA LINHA

```
142  data = sys.stdin.read()#reading from stdin
143  inputLines = data.splitlines()#splitting the data into a string
144
145
146  h, l, R = inputLines[0].split()
147  h = int(h)
148  l = int(l)
149  R = int(R)
150
151  listHidro = []
152  listCentral = []
153  listArco = []
154
155  convertData(h, l, R, listHidro, listCentral, listArco)
```

FONTE: O autor (2020)

Na linha 155, é chamada a função *convertDat.* A cada execução do loop da linha 36 na Figura 10, criamos um objeto Hidroeletrica e adicionamos na lista referente. O loop executará h vezes, já que existe um bloco de h linhas com os dados m_i, f_i, c_i. Lembrando que o loop não é executado quando x = h+1. Na linha 43 usamos a variável *indexLine,* para facilitar no controle de qual linha será lida. Como ainda não lemos a linha h+1, *indexLine* terá esse valor.

A cada execução do loop da linha 45, criamos um objeto Central e adicionamos na lista referente. O loop executará L vezes, já que existe um bloco de L linhas com o dado d_i. Para verificar se o número de execução está correto, basta ver que (*indexLine* + L – 1) - (indexLine) + 1 = L. Na linha 49 atualizamos a variável *indexLine,* para começar o próximo loop no valor correto.

O loop da linha 51 verifica os vizinhos de cada hidroelétrica. Como há h hidroelétricas, o loop 51 executa h vezes. Primeiro pegamos a primeira informação, o número de vizinhos (n_i) da hidroelétrica x. O loop da linha 54 executará n_i vezes, já que cada linha refere-se a um vizinho. No loop interno criamos um arco (por onde passará a energia), adicionamos uma saída à hidroelétrica x e adicionamos uma entrada à central *inCentral*. O comando “len(listArco)” retorna o a quantidade de arcos, já que o arco que criamos é o último da lista, seu índex é igual ao tamanho da lista de arcos. Lembrando que as listas começam com o índice zero, portanto é necessário adicionar -1 às linhas 60 e 61.

O loop da linha 64 é muito semelhante ao loop 51, com exceção que executará L vezes, já que há L centrais. Fazemos o mesmo processo, adicionamos uma saída à central x e uma entrada à central *Incentral*.

FIGURA 10 – FUNÇÃO “convertData”

```
35  def convertData(h, l, R, listHidro, listCentral, listArco):
36      for x in range(1, h + 1):
37          linha = inputLines[x].split()
38          mi = int(linha[0])
39          fi = int(linha[1])
40          ci = int(linha[2])
41          listHidro.append(Hidroeletrica(mi, fi, ci))
42
43      indexLine = h + 1
44
45      for x in range(indexLine,  indexLine + l):
46          di = int(inputLines[x])
47          listCentral.append(Central(di))
48
49      indexLine += l
50
51      for x in range(1, h+1):
52          ni = int(inputLines[indexLine])
53          indexLine += 1
54          for y in range(indexLine, indexLine + ni):
55              linha = inputLines[y].split()
56              inCentral = int(linha[0])
57              capacity = int(linha[1])
58              cost = int(linha[2])
59              listArco.append(Arco(capacity, cost))
60              listCentral[inCentral-1].setInput(len(listArco))
61              listHidro[x-1].setOutput(len(listArco))
62          indexLine += ni
63
64      for x in range(1, l+1):
65          ni = int(inputLines[indexLine])
66          indexLine += 1
67          for y in range(indexLine, indexLine + ni):
68              linha = inputLines[y].split()
69              inCentral = int(linha[0])
70              capacity = int(linha[1])
71              cost = int(linha[2])
72              listArco.append(Arco(capacity, cost))
73              listCentral[inCentral-1].setInput(len(listArco))
74              listCentral[x-1].setOutput(len(listArco))
75          indexLine += ni
```

FONTE: O autor (2020)

5.3 CRIANDO A ENTRADA PARA O SIMPLEX

5.3.1 RESTRIÇÕES

Como vimos na Seção 4 é necessário criar um vetor de vetores A formado pelas Inequações 5 (referente às hidroelétricas) e Inequações 4 (referente às centrais), nesta ordem. Portanto o tamanho de A será h + L, já que cada hidroelétrica e cada central tem sua restrição.

FIGURA 11 – EQUAÇÕES DA HIDROELÉTRICA

```
92  def equationsHidro(aux):
93      arcoSize = len(listArco)
94      resultEQ = []
95      emptyList = [0] * h
96      for x in range(1, arcoSize + 1):
97          if(x in aux.outputs):
98              resultEQ.append(1)
99          else:
100             resultEQ.append(0)
101
102     emptyList[listHidro.index(aux)] = - aux.f
103     emptyList.extend(resultEQ)
104     return emptyList
```

FONTE: O autor (2020)

A função *equationsHidro* da figura 11, recebe um objeto Hidroeletrica como parâmetro e retorna sua equação restritiva em forma de vetor, como em [-20, 0, 1, 0, 0] na Seção 4. O loop da linha 96 checa se um arco x pertence ao conjunto de saídas da hidroelétrica aux, caso pertença, seu coeficiente deverá ser 1, senão 0. Assim, temos os coeficientes de $e_1, e_2, \dots, e_a$ na lista *resultEQ*. Lembrando que precisamos da primeira parte da inequação, como em $-20v_1 + e_1 \leq 0$. O coeficiente -20 (oposto da eficiência da hidroelétrica 1) deve ser adicionado na posição correta. Fazemos isso criando uma lista vazia *emptyList* de tamanho h e adicionando o oposto da eficiência da hidroelétrica aux no índex i, sendo i igual à posição de aux na lista listHidro. Concatenando a lista *emptyList* (não mais vazia) à lista *resultEQ*, temos uma restrição que será adicionada à A.

QUADRO 3 – COEFICIENTS DE UMA HIDROELÉTRICA

Lista	*emptyList*					*resultEQ*		
Índice	v_1	.. .	v_i	...	v_h	e_1	...	e_a
Coeficientes da hidroelétrica aux	0	0	$-F_i$	0	0	$C_{e_j} = j \in aux.outputs \; ? \; 1 : 0$		

FONTE: O autor (2020)

A função *equationsCentral* da Figura 12 recebe um objeto Central como parâmetro e retorna sua equação restritiva em forma de vetor, como em [0, 0, 0, -1, -1] na seção 4. O loop da linha 81 checa se um arco x pertence ao conjunto de saídas ou entradas da central aux. Caso pertença às entradas, seu coeficiente será -1, caso pertença às saídas, seu coeficiente será 1 e caso não pertença a nada, será 0. Retornamos uma lista vazia (tamanho h) concatenada à lista de coeficientes *resultEQ* .

QUADRO 4 – COEFICIENTES DE CADA

Lista	*emptyList*					*resultEQ*		
Índice	v_1	.. .	v_i	...	v_h	e_1	...	e_a
Coeficientes da central aux	0	0	0	0	0	$C_{e_j} = -1, se\ j \in aux.inputs$ $C_{e_j} = 1, se\ j \in aux.outputs$ $senão, C_{e_j} = 0,$		

FONTE: O autor (2020)

FIGURA 12 – EQUAÇÕES DA CENTRAL

```
77  def equationsCentral(aux):
78      arcoSize = len(listArco)
79      resultEQ = []
80      emptyList = [0] * h
81      for x in range(1, arcoSize + 1):
82          if(x in aux.inputs):
83              resultEQ.append(-1)
84          elif (x in aux.outputs):
85              resultEQ.append(1)
86          else:
87              resultEQ.append(0)
88
89      emptyList.extend(resultEQ)
90      return emptyList
```

FONTE: O autor (2020)

Utilizando as funções *equationsHidro* e *equationsCentral*, adicionamos as restrições de todas as hidroelétricas e centrais, nesta ordem, na função *constraints.*

FIGURA 13 – FUNÇÃO “constraints”

```
106 def constraints(listHidro, listCentral):
107     A = []
108     for x in listHidro:
109         A.append(equationsHidro(x))
110     for x in listCentral:
111         A.append(equationsCentral(x))
112     return A
```

FONTE: O autor (2020)

5.3.2 LIMITANTES DAS VARIÁVEIS

Como vimos na Seção 4, para cada hidroelétrica i, sua vazão v_i tem os limitantes superiores R e m_i/f_i. E para cada arco i, a energia e_1 transmitida por este é limitada pela capacidade E_i do arco. A função *bounds* da Figura 14, adiciona tais restrições em um vetor chamado *bounds*.

FIGURA 14 – FUNÇÃO “bounds”

```
114  def bounds(listHidro, listArco):
115      bounds = []
116      for x in listHidro:
117          xi_bounds = (0, min(x.m/x.f, R))
118          bounds.append(xi_bounds)
119      for x in listArco:
120          xi_bounds = (0, x.capacity)
121          bounds.append(xi_bounds)
122      return bounds
```

FONTE: O autor (2020)

5.3.3 LIMITANTES DAS INEQUAÇÕES E FUNÇÃO CUSTO

Na Figura 15, utilizamos a função *vector* para retornar um vetor c contendo os coeficientes da função custo. A função *results* retorna um vetor b contendo os limitantes superiores de cada inequação, na ordem que foram adicionados em A.

FIGURA 15 – FUNÇÕES “vector” e “results”

```
124  def vector(listHidro, listArco):
125      c = []
126      for x in listHidro:
127          c.append(x.c)
128      for x in listArco:
129          c.append(x.cost)
130      return c
131
132  def results(listHidro, listCentral):
133      b = []
134      for x in listHidro:
135          b.append(0)
136      for x in listCentral:
137          b.append(-x.d)
138      return b
```

FONTE: O autor (2020)

6. USANDO O SIMPLEX

Depois de calcular todos os dados necessários aos simplex utilizamos a função linprog, que retorna uma classe com os campos descritos na Figura 3. Na linha 163 alteramos o vetor x tipo Array para uma lista resultList para poder modificar seus campos. Nas linhas 164 a 166, calculamos a produção da hidroelétrica i ($P_i = v_i * f_i$) e arredondamos os valores de saída para inteiros. A linha 169 é responsável por transformar cada valor de resultList em uma string e colocá-los linha a linha na saída padrão STDOUT.

FIGURA 16 – GERANDO A SAÍDA DO PROBLEMA

```
155  convertData(h, l, R, listHidro, listCentral, listArco)
156
157  c = inSimplex.vector(listHidro, listArco)
158  A = inSimplex.constraints(listCentral, h, listArco, listHidro)
159  b = inSimplex.results(listHidro, listCentral)
160  bounds = inSimplex.bounds(listHidro, listArco, R)
161
162  res = linprog(c, A_ub=A, b_ub=b, bounds=bounds)
163  resultList = res.x.tolist()
164  for x in range(0, h):
165      resultList[x] = int(resultList[x]*listHidro[x].f)
166  for x in range(h, len(resultList)):
167      resultList[x] = int(resultList[x])
168
169  print('\n'.join(map(str, resultList)))
```

FONTE: O autor (2020)

REFERÊNCIAS

MATOUŠEK, Jiří; GÄRTNER, Bernd. **Understanding and Using Linear Programming**. [*S. l.*]: Springer, 2007.

THE SCIPY COMMUNITY. SciPy.org: **scipy.optimize.linprog**. [*S. l.*], 23 jul. 2020. Disponível em: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html#scipy.optimize.linprog. Acesso em: 1 set. 2020.