Otimização: Distribuição de Energia

Chrystopher Naves Bravos

Resumo

Este artigo consiste na modelação de uma equação linear para o problema de distribuição de energia entre centrais. Através de métodos simplistas a solução encontrada gera um resultado satisfatório e rápido para o problema proposto.

1 Introdução

A modelação de problemas matemáticos afim de encontrar um resultado ótimo é uma tarefa que exige atenção e prática, mas que gera melhorias em processos cotidianos em escala residencial e industrial. O problema de distribuição de energia nos apresenta uma junção de produção de energia e distribuição através de uma malha energética, com o propósito de baratear o custo dos processos e atingir as demandas necessárias.

2 Descrição do modelo

Inicialmente é necessário apresentar uma legenda para todas as variáveis utilizadas, de forma a facilitar o compreendimento do leitor.

h_i : Hidrelétrica i $\quad i = \{1, 2, ..., h\}$
P_i : Produção na hidrelétrica i $\quad i = \{1, 2, ..., h\}$
F_i : Eficiência energética na hidrelétrica i $\quad i = \{1, 2, ..., h\}$
V_i : Vazão da hidrelétrica i $\quad i = \{1, 2, ..., h\}$
M_i : Capacidade de produção da hidrelétrica i $\quad i = \{1, 2, ..., h\}$
Ch_i : Custo total de produção da hidrelétrica i $\quad i = \{1, 2, ..., h\}$
Ch'_i : Custo de produção por vazão na hidrelétrica i $\quad i = \{1, 2, ..., h\}$
I_i : Central elétrica i $\quad i = \{1, 2, ..., I\}$
n_i : Número de conexões saindo de i $\quad i = \{1, 2, ..., h + I\}$
$w_{i,j}$: Capacidade de transmissão do arco i, j $\quad i = \{1, 2, ..., h + I\} \quad j = \{1, 2, ..., n_i\}$
$w'_{i,j}$: Quantidade a ser transmitida pelo arco i, j $\quad i = \{1, 2, ..., h + I\} \quad j = \{1, 2, ..., n_i\}$
$Cl_{i,j}$: Custo total de transmissão do arco i, j $\quad i = \{1, 2, ..., h + I\} \quad j = \{1, 2, ..., n_i\}$
$Cl'_{i,j}$: Custo de transmissão do arco por quantidade i, j $\quad i = \{1, 2, ..., h + I\} \quad j = \{1, 2, ..., n_i\}$
E_i : Soma de todos $w'_{i,j}$: que entram na central i $\quad i = \{1, 2, ..., I\}$
E_i : Soma de todos $w'_{i,j}$: que saem da central i $\quad i = \{1, 2, ..., I\}$
R : Vazão máxima das hidroelétricas

Tendo isto, podemos iniciar a ideia por trás do modelo criado.
Como temos um problema de minimização de custo, é quase que automática a ideia de que nossa função objetivo deve conter a soma de todos os custos gerados. As variáveis que nos geram esse custo são Ch_i e $Cl_{i,j}$, porém elas apenas nos fornecem os custos totais de produção e transmissão, respectivamente, sendo assim precisamos gerar variáveis "reais", ou seja, valores incógnitos que serão multiplicados pelos seus respectivos custos. Como $Ch_i = Ch'_i \times V_i$ e $Cl_{i,j} = Cl'_{i,j} \times w'_{i,j}$, generalizando para o sistema todo, temos:

$$\sum_{i=1}^{h} Ch'_i \times V_i + \sum_{i=1}^{h+I} \sum_{j=1}^{n_i} Cl'_{i,j} \times w'_{i,j} \tag{1}$$

2.1 Equações e inequações

Tendo nossa função objetivo, agora podemos buscar os limitantes de nossas variáveis. Pelo problema proposto algumas inequações são facilmente notadas:

$$P_i <= M_i \tag{2}$$
$$V_i <= R \tag{3}$$
$$w'_{i,j} <= w_{i,j} \tag{4}$$

Como $P_i = F_i \times V_i$ então em (2) temos $F_i \times V_i <= M_i$.
O problema da modelagem, no entanto, está em garantir que toda a demanda seja atendida e assegurar o controle da energia produzida no sistema. Ao observarmos o problema de fluxo em uma rede visto em Matousek and Gärtner (2007) vemos que o problema é parecido: fazer com que todos os dados enviados de um lado cheguem ao outro lado da melhor forma possível, a diferença para o nosso problema é que de um nodo para outro existe uma perda de conteúdo, ou seja, parte do que foi recebido fica no nodo para atender a demanda. Dessa forma podemos utilizar o mesmo método de Gartner fazendo uma pequena alteração.
Pelo problema do fluxo em rede temos que tudo que entra em um nodo (E_i) tem que ser igual á tudo que sai (S_i), logo $E_i - S_i = 0$. Ao adicionarmos o fator demanda (D_i), tudo o que sai do nodo deve ser igual a tudo que entrou, descontado da demanda atendida, logo temos que $E_i - D_i = S_i \implies E_i - S_i = D_i$.
Porém esta expressão aplica-se somente as centrais elétricas, que recebem e enviam energia, portanto também precisamos de uma equação que atenda as hidrelétricas, que somente enviam energia. Neste caso temos a mesma situação do fluxo de rede, ou seja, tudo que foi produzido deve sair, sendo assim fica fácil de ver que a expressão $F_j \times V_j - S_j = 0$, com $j = \{1, 2, ..., h\}$, atende ao sistema.

Dessa maneira temos o LP:

$$\min_{s.t.} \sum_{i=1}^{h} Ch'_i \times V_i + \sum_{i=1}^{h+l} \sum_{j=1}^{n_i} CI'_{i,j} \times w'_{i,j}$$
$$F_i \times V_i <= M_i$$
$$V_i <= R$$
$$w'_{i,j} <= w_{i,j}$$
$$E_i - S_i = D_i$$
$$V_j - S_j = 0$$

3 Implementação

Com o programa linear já modelado agora basta realizar a implementação do modelo. Para tal tarefa foi-se utilizado a linguagem Python, que possui uma sintaxe fácil e apresenta um bom desempenho, além de contar com a biblioteca *SciPy*, para a resolução das matrizes e vetores equacionais através do método simplex.

3.1 A biblioteca *SciPy*

A biblioteca *SciPy* é, de acordo com os Developers (2020), "um ecossistema baseado em Python de software de código aberto para matemática, ciências e engenharia". A biblioteca conta com várias plataformas tais como NumPy e Pandas.
Neste trabalho especificamente foi-se utilizado a função *linprog()*, pertencente a biblioteca *scipy.optmize*, para a resolução do problema. A função possuí este escopo:

```
scipy.optimize.linprog(c, A_ub=None, b_ub=None, A_eq=None, b_eq=None,
                       bounds=None, method='some-method', callback=None,
                       options=None)
```

onde *c* é a função objetivo; *A_ub* é a matriz de inequações limitadas superiormente; *b_ub* é o vetor dos limitantes de *A_ub*; *A_eq* é a matriz de equações; *b_eq* é o vetor dos limitantes de *A_eq*; *bounds* é a lista de tuplas dos limitantes superior e inferior de cada variável; *method* é o método a ser utilizado, seguido por flags adicionais.

3.2 O programa

O código implementado segue a ordem de entrada dos valores. Nos primeiros loops são atribuídos as hidrelétricas, com seus respectivos atributos, a uma lista de hidrelétricas e as centrais elétricas, também com seus respectivos atributos, a uma lista de centrais. A "mágica"ocorre na atribuição dos arcos, onde é criada uma matriz $Dist\_h$ de $h + I$ linhas por I colunas, onde $Dist\_h_{0,0}$ contém os dados de uma ligação entre o primeiro elemento e a primeira central, $Dist\_h_{2,5}$ contém os dados de uma ligação entre o terceiro elemento e a sexta central, e assim por diante, caso não exista ligação é atribuído o valor zero. Dessa forma basta acessar a linha do elemento para saber todas as ligações que saem e basta acessar a coluna para saber todos as ligações que entram.
Os loops seguintes já contém as atribuições das matrizes e vetores que irão ser utilizados por *linprog()*, são loops simples, porém alguns detalhes são importantes para uma melhor compreensão.
O vetor *obj*, que corresponde a função objetivo, tem tamanho $h + numero\ de\ arcos$, o que resulta no número de variáveis do sistema.
A matriz de equações é dividida em duas partes: coeficientes da diferença $E_i - S_i$ e coeficientes da diferença $V_j - S_j$.
O vetor *bds*, que corresponde ao vetor de tuplas limitantes, é preenchido da seguinte maneira: variáveis V_i são limitadas superiormente por R, variáveis $w'_{i,j}$ são limitadas superiormente por $w_{i,j}$, e todas são limitadas inferiormente por zero.
Após isto, as matrizes e vetores gerados são adicionados a função *linprog()*, onde um resultado ótimo é calculado pelo método simplex. O vetor resultado é tratado, ou seja, vazões são multiplicadas pela eficiências, e o resultado final é gerado na ordem em que foi recebido.

4 Exemplo

Para melhor entendimento vamos gerar as matrizes e vetores *Dist_h*, *obj*, *A_eq*, *b_eq*, *A_ineq*, *b_ineq* e *bds* para uma 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
```

Temos então:

$$Dist\_h = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$obj = \begin{bmatrix} 2 & 1 & 1 & 2 & 1 \end{bmatrix}$$

$$A\_eq = \begin{bmatrix} 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 1 \\ 20 & 0 & -1 & 0 & 0 \\ 0 & 10 & 0 & -1 & 0 \end{bmatrix}$$

$$b\_eq = \begin{bmatrix} 50 & 80 & 0 & 0 \end{bmatrix}$$

$$A\_ineq = \begin{bmatrix} 20 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \end{bmatrix}$$

$$b\_ineq = \begin{bmatrix} 100 & 50 \end{bmatrix}$$

$$bds = \begin{bmatrix} (0, 10) & (0, 10) & (0, 200) & (0, 200) & (0, 100) \end{bmatrix}$$

Que nos gera uma saída:

80
50
80
50
30

Referências

S. Developers. Scipy.org, 2020. https://www.scipy.org/, acesso em 07/09/2020.

J. Matousek and B. Gärtner. *Understanding and Using Linear Programming*. Springer, Berlin, Heidelberg, 2007. doi: https://doi.org/10.1007/978-3-540-30717-4.