Trabalho 1 de Otimização

Carlos Alberto Pedroso Junior

Dezembro 2020

1 Introdução

Podemos definir o problemas de otimização sendo aquele que visa determinar os pontos extremos de uma função, seja ele mínimo, e então os problemas serão de minimização, ou máximo, problemas de maximização. Otimização é algo extremamente comum em nosso cotidiano, sendo aplicável nos mais diversos tipos e formas de problemas. Os exemplos vão desde os mais simples, de decidir como pegar um caminho mais curto para chegar ao trabalho, escolher os produtos corretos no supermercado para diminuir o preço da compra; até problemas mais complexos, como minimizar custos de uma produção em uma fábrica, determinar o melhor conjunto de rotas para uma transportadora, definir a melhor dieta com certos tipos de produtos, entre outros.

Segundo [3], o problema de otimização pode ser dividido em duas categorias: os com variáveis contínuas e os com variáveis discretas. Nos problemas contínuos, busca-se normalmente por um conjunto de números reais ou mesmo uma função. Nos problemas de variáveis contínuas (chamados combinatória), procura-se por um objeto como um inteiro, uma permutação ou grafo de um conjunto finito (ou possivelmente enumerável). Um problema de otimização com variáveis discretas é conhecido como um problema de otimização combinatória, podendo ser resolvido através de programação linear. Desta forma, segundo [3], a programação linear tem papel fundamental na teoria de otimização pois lida com a combinação de problemas sendo aplicada de forma satisfatória.

A otimização abrange diversas áreas do conhecimento, sendo utilizada para resolver uma variedade de problemas, atribuídos à computação, a biologia, a economia, etc. Seja qual for o problema e a área, o primeiro passo rumo à otimização do problema está em encontrar uma modelagem matemática adequada. A otimização necessita uma função matemática para atuar e, principalmente em sistemas dinâmicos, a modelagem não é tarefa simples. A modelagem do problema é representada pela função objetivo, também conhecida como *fitness* ou aptidão. Essa é a função que desejamos minimizar ou maximizar, conforme o problema apresentado. Neste trabalho, estamos objetivando minimizar os custos de uma determinada empresa através do escalonamento de tarefas entre máquinas. Nesse sentido, definimos o escalonamento de tarefas com o objetivo de encontrar uma atribuição que minimize o custo para completar um conjunto de tarefas.

2 Problema

Seguindo as informações do trabalho proposto, define-se o problema de escalonamento de tarefas em máquinas em uma empresa. Sendo composto de um conjunto de tarefas T que devem ser executadas por um conjunto de máquinas M, podendo ser próprias ou alugadas. Além disso, cada tarefa consome uma quantidade de horas h_t e as máquinas têm um custo c_m de reais por hora e tem um tempo máximo u_m em que podem ser utilizadas e executando um subconjunto s_m de tarefas. O problema está em minimizar os custos e executar todas as tarefas atribuídas, de forma a encontrar sempre o custo mínimo entre as máquinas, sejam elas alugadas ou próprias. Na literatura, este problema normalmente define-se como *schedule machine optimization* [2], em que dado um determinado conjunto de tarefas e máquinas, planejamos encontrar o ótimo da execução de todas as tarefas, gastando o mínimo possível. Logo, o escalonamento de tarefas em máquinas consiste em encontrar uma atribuição das tarefas às máquinas de modo a minimizar o máximo de custo de operações, ou seja, encontrar um escalonamento que minimize o tempo para completar todas as tarefas.

Os formatos de entrada para o problema declara dois números representando $n = |T|$ e $l = |M|$, seguidos de n números correspondendo as quantidades de horas (h_t) de cada tarefa. Consideramos também que as tarefas são enumeradas de 1 a n, logo após temos l conjunto de dois números representando os custos de operação (c_m) e o tempo máximo de uso (u_m) de cada máquina m. Em seguida, temos para cada máquina m, o tamanho $s_m = |S_m|$, e s_m índices representando o conjunto S_m. Definindo também que as máquinas estão enumeradas de 1 a l.

3 Modelagem

A modelagem visa criar um modelo matemático para explicar e compreender as formas de resolver o problema proposto satisfazendo todas as condições e restrições atribuídas. Dessa forma, para melhor modelar o problema, foi criada uma variável x_{mt}, onde m representa as horas das máquinas e t as tarefas que ela executa. A função objetivo de minimização min de custos e as restrições são definidas e explicadas abaixo:

$$min. : \sum_{m=1}^{l} c_m.(\sum_{t=1}^{n}) \tag{1}$$

$$sujeito : \sum_{t=1}^{s_m} x_{mt} \leq u_m \ \forall m \ \epsilon \ M \ (1)$$

$$\sum_{m=1}^{l} x_{mt} = h_t \ (2)$$

$$\sum_{t=1}^{n} h_t \leq \sum_{m=1}^{l} u_m \ (3)$$

$$x_{mt} \geq 0 \ (4)$$

$$h_t \geq 0 \ \forall t \ \epsilon \ T \ (5)$$

$$u_m \geq 0 \ \forall m \ \epsilon \ M \ (6)$$

$$S_m \geq 0 \ \forall m \ \epsilon \ M \ (7)$$

A Equação 1 define a função objetivo para minimização dos custos e execução de todas as tarefas. O primeiro somatório representa o cálculo de todas as máquinas dadas como entrada, o segundo somatório calcula todas as tarefas dadas na entrada multiplicado pelo custo obtido na entrada para cada máquina. Desta forma, tenta-se minimizar os custos primeiramente nas máquinas de menor custo ou se o custo for 0. As restrições da modelagem são descritas abaixo da equação, onde são definidas conforme o problema foi apresentado.

A restrição (1) refere-se à soma das horas do conjunto de tarefas que deve ser menor ou igual a quantidade de horas disponíveis para cada máquina do conjunto de máquinas, definindo assim que as máquinas não podem ultrapassar seus limites de horas para execução das tarefas. A restrição (2) determina que o somatório do conjunto de máquinas deve ser igual às horas das tarefas de entrada. A restrição (3) determina que o total de horas das tarefas deve ser menor do que o total de horas disponíveis no conjunto de máquinas. A restrição (4) define que a variável x_{mt} deve ser maior ou igual a 0, garantindo a não utilização de valores negativos de entrada. A restrição (5) garante que o tempo de consumo pode ser maior ou igual a 0, visto que as tarefas sempre demandaram um tempo de consumo. A restrição (6) indica que o tempo máximo de execução sempre será maior ou igual a 0. A restrição (7) estabelece que conjunto de tarefas deve ser sempre maior ou igual a 0, isso justifica-se pelo fato de todas tarefas e máquinas exigirem horas para execução. O estabelecimento dessas restrições tem como objetivo melhorar a modelagem e definir limites para a função objetivo, além de delimitar a implementação e entrada do programa. Algumas das restrições propostas como ((3), (5), (6) e (7)) são direcionadas para a entrada do programa buscando evitar problemas na obtenção dos valores de entrada e alcançando um melhor funcionamento.

3.1 Exemplos

os exemplos descritos abaixo objetivam validar a modelagem proposta, através da atribuição da função objetivo e suas restrições. Desta maneira, aplicamos a modelagem nos exemplos 1 e 2 descritos no trabalho e no exemplo 3 desenvolvido para teste e validação. As entradas estão explicitas da seguinte maneira: primeiro têm-se as tarefas n e as máquinas l, horas das tarefas h_n, os custos c_n e o tempo máximo de execução u_n e o número de tarefas por máquina s_m. Segundo, temos a aplicação da função objetivo de minimização min e as restrições $Sujeito$. E terceiro, temos o resultado obtido, em que temos uma matriz com as linhas representadas pelas máquinas e as colunas representada pelas tarefas executadas, abaixo temos os custos de execução.

Exemplo 1:

2 2
10
10
100 20
20 10
2
1
2
1
1

$$min. = (100 * x_{1_1} + 100 * x_{1_2} + 50 * x_{2_1} + 50 * x_{2_2})$$

$$Sujeito:$$

$$x_{1_1} + x_{1_2} \leq 20$$

$$x_{2_1} + x_{2_2} \leq 10 \ (1)$$

$$x_{1_1} + x_{1_2} = 10$$

$$x_{2_1} + x_{2_2} = 10 \ (2)$$

$$x_{1_1} \geq 0, \ x_{1_2} \geq 0, \ x_{2_1} \geq 0, \ x_{2_2} \geq 0; \ (4)$$

Resultados exemplo 1:

10.0 0.0
0.0 10.0
1500.0

Exemplo 2:

2 3
10
20
0 5
0 10
100 50
2
1
2
1
2
2
1
2

$$min. = (0 * x_{1_1} + 0 * x_{1_2} + 0 * x_{2_1} + 0 * x_{2_2} + 100 * x_{3_1} + 100 * x_{3_2})$$

$$Sujeito:$$
$$x_{1_1} + x_{1_2} \leq 5$$
$$x_{2_1} + x_{2_2} \leq 10$$
$$x_{3_1} + x_{3_2} \leq 50 \ (1)$$
$$x_{1_1} + x_{2_1} + x_{3_1} = 10$$
$$x_{2_1} + x_{2_2} + x_{3_2} = 20 \ (2)$$
$$x_{1_1} \geq \ 0, \ x_{1_2} \geq \ 0, \ x_{2_1} \geq \ 0, \ x_{2_2} \geq \ 0, \ x_{3_1} \geq \ 0, \ x_{3_2} \geq \ 0; \ (4)$$

Resultado exemplo 2:
0.0 5.0
0.0 10.0
10.0 5.0
1500.0

Exemplo 3:
4 4
10
20
15
5
0 20
50 10
100 20
50 10
4
1
2
3
4
4
1
2
3
4
4
1
2
3
4
4
1
2
3
4

$$min. = (0*x_{1_1}+0*x_{1_2}+0*x_{1_3}+0*x_{1_4}+50*x_{2_1}+50*x_{2_2}+50*x_{2_3}+50*x_{2_4}+100*x_{3_1}+100*x_{3_2}+100*x_{3_3}+100*x_{3_4}+50*x_{4_1}+50*x_{4_2}+50*x_{4_3}+50*x_{4_4})$$

$$sujeito:$$

$$x_{1_1}+x_{1_2}+x_{1_3}+x_{1_4} \leq 20$$
$$x_{2_1}+x_{2_2}+x_{3_3}+x_{2_4} \leq 10$$
$$x_{3_1}+x_{3_2}+x_{3_3}+x_{3_4} \leq 20$$
$$x_{4_1}+x_{4_2}+x_{4_3}+x_{4_4} \leq 10 \ (1)$$
$$x_{1_1}+x_{1_2}+x_{1_3}+x_{1_4} = 10$$
$$x_{2_1}+x_{2_2}+x_{3_3}+x_{2_4} = 20$$
$$x_{3_1}+x_{3_2}+x_{3_3}+x_{3_4} = 15$$
$$x_{4_1}+x_{4_2}+x_{4_3}+x_{4_4} = 5 \ (2)$$
$$x_{1_1} \geq 0, \ x_{1_2} \geq 0, \ x_{1_3} \geq 0, \ x_{1_4} \geq 0,$$
$$x_{2_1} \geq 0, \ x_{2_2} \geq 0, \ x_{2_3} \geq 0, \ x_{2_4} \geq 0,$$
$$x_{3_1} \geq 0, \ x_{3_2} \geq 0, \ x_{3_3} \geq 0, \ x_{3_4} \geq 0,$$
$$x_{4_1} \geq 0, \ x_{4_2} \geq 0, \ x_{4_3} \geq 0, \ x_{4_4} \geq 0; \ (4)$$

Resultado exemplo 3:
0.0 0.0 15.0 5.0
0.0 10.0 0.0 0.0
10.0 0.0 0.0 0.0
0.0 10.0 0.0 0.0
2000.0

4 Implementação

O modelo foi implementado na linguagem Python 2.7 [4], nativa dos sistemas Linux. Foi instalada e utilizada a biblioteca do lpsolve versão 5.5 [1]. Toda a implementação foi realizada no sistema Ubuntu Mate 20.04 LTS. Para utilizar o lpsolve juntamente com o Python, foi necessário seguir alguns passos e instruções descritas na documentação do lpsolve. Entretanto, cabe destacar que muitas informações não constam na documentação oficial, desta forma, foi preciso buscar em outras fontes, como *stack overflow*, *github* entre outros. Seguindo as recomendações do enunciado, foi desenvolvido um programa que usa a entrada via terminal e converte do modelo de entrada da função objetivo usada pela biblioteca do lpsove. Dessa forma, dentro do programa realiza-se uma chamada da biblioteca, onde após convertidos os valores gera-se a saída com o resultado para o problema. A construção dos vetores e matrizes que compõem o lpsolve ocorre conforme a entrada obtida através do *script* fazendo-se necessário convertê-las para adaptação ao formato lpsolve. Isso acontece pela forma como o lpsolve é construído, necessitando de um conjunto de atribuições para realização dos cálculos. O modelo de entrada do lpsolve pode ser observado abaixo.

$$[obj, x, duals] = lpsolve([f], [a], [b], [e], [vlb], [vub], [xint])$$

Informações complementares e maiores detalhes sobre essa construção do resolvedor podem ser encontrados na documentação oficial da biblioteca [1], onde constam diversos exemplos, informações e instruções para aplicação de vários problemas.

4.1 Execução do *script*

No arquivo (tar.gz) está contido um *script* em shell *makefile.sh* para instalação das variáveis de ambiente, também consta um *inscript* em Python com o nome (*tarefas.py*) para execução do resolvedor. Além de três exemplos (**exemplo1.txt, exemplo2.txt e exemplo3.txt**), que são os mesmos apresentados neste documento, os instaláveis da biblioteca e o arquivo README.txt com instruções para instalação do lpsolve. Os exemplos podem ser passados como entrada para facilitar a execução, ou caso prefira, o *tarefas.py* também recebe as entradas diretamente no terminal, basta executar e digitar as entradas. [1] [2] Para executar o *sripT* passando o arquivo de entrada basta digitar no terminal:

- python2.7 tarefas.py < exemplo1.txt

Observação: Caso não queira instalar o lpsolve no sistema operacional, apenas execute o *script* de nome *markefile.sh* e passe as entradas por arquivo que funcionará.

References

[1] LpSolve. Web access statistics. http://lpsolve.sourceforge.net/5.5/Python.htm.

[2] Joon-Yung Moon, Kitae Shin, and Jinwoo Park. Optimization of production scheduling with time-dependent and machine-dependent electricity cost for industrial energy efficiency. *The International Journal of Advanced Manufacturing Technology*, 68(1-4):523–535, 2013.

[3] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[4] Python. Web access statistics. hhttps://www.python.org/download/releases/2.7/.

[1] A execução direto no terminal necessita de instalação do lpsolve no sistema operacional

[2] sudo apt install lp-solve