Trabalho 1 - Otimização

Nicolas Dencker De Marco - GRR20171605

Dezembro de 2020

1. Introdução

No trabalho é apresentado um problema de escalonamento de tarefas em máquinas. Como descrito pelo professor na descrição do trabalho, o problema é:

Uma empresa tem um conjunto de tarefas T para serem executadas em um determinado período. Para a execução destas tarefas a empresa conta com um conjunto M de máquinas. Cada tarefa $t \in T$ consome uma quantidade de horas de cpu, Ht. Cada máquina $m \in M$ tem um custo de operação de Cm reais por hora, tem um tempo máximo Um em horas que pode ser usada, e pode executar um subconjunto de tarefas $Sm \subseteq T$. Queremos minimizar os custos e executar todas as tarefas.

2. Modelagem

Nesta seção é apresentado a modelagem usada para conseguirmos resolver o problema. Primeiramente, vamos definir a equação a ser minimizada, ou seja, a equação do custo de operação Ct.

Para isso, podemos perceber que cada máquina possui seu custo por hora e o número de horas máximo que podemos utilizá-la para executar nossas tarefas, tendo como objetivo executar todas as tarefas no seu tempo determinado, não necessariamente esse tempo chegará ao tempo máximo de uso das máquinas. Dessa forma podemos determinar que o Custo de Operação é representado pela soma das horas que a máquina executa suas tarefas Hl multiplicado pelo custo da sua utilização Cl, ou seja,

$$Ct = \sum Cl \, . \, Hl$$

Tendo $l = 1$ indo até o número de máquinas.

Após isso precisamos definir as Restrições para efetuarmos a minimização.

1. Tempo máximo de uso de uma Máquina.
2. Tempo para execução de uma tarefa
3. O tempo deve ser não-negativo

Sendo assim, partirmos para a 1a Restrição:

O tempo de uso *Hl* de uma máquina $l \in M$ será igual ao somatório do tempo usado para cada tarefa $t \in T$, que pertence ao subconjunto de tarefas que podem executar na máquina *l* ($Sl \subseteq T$), ser executada *Hl,t* , ou seja,

$$Hl = \sum Hl,\ t$$

Sendo o somatório do tempo utilizado para executar todas as tarefas que pertencem ao subconjunto *Sl*. Logo o tempo máximo de uso *Ul* >= *Hl*

Segunda Restrição:

O tempo de execução *Ht* de uma tarefa $t \in T$ é equivalente ao tempo total que a tarefa é executada em cada máquina, ou seja,

$$Ht = \sum Hl,\ t$$

Onde o somatório se inicia em *l = 1* e vai até o número de máquinas, caso *j* não pertença à Sl o tempo *Hl,t = 0.*

Terceira Restrição:

O tempo deve ser não-negativo, ou seja, *Hl,t* >= *0* para qualquer $l \in M$ ou $t \in T$.

3. Resolução

Para resolver o problema apresentado através da modelagem demonstrada acima, utilizei a linguagem de programação Python3, e a lib SciPy para resolver utilizando o método simplex.

1. Recebimento de valores

Como início, é inputado as variáveis a serem usadas no programa, após isso, recebo os inputs no mesmo formato dos exemplos na descrição do trabalho. Começo recebendo o número de tarefas e o número de máquinas. Para cada tarefa eu recebo a quantidade de horas que a tarefa ocupa para ser executada. Para cada máquina: eu recebo as informações de custo e tempo máximo que a máquina pode ser utilizado; eu recebo as tarefas que cada máquina pode executar.

2. Definição do vetor Custo

Além disso, para cada máquina e para cada tarefa que pode ser executada pela máquina eu envio para um array o custo daquela máquina. Sendo o tamanho do vetor será o número de tarefas multiplicado pelo número de máquinas.

3. Definição dos vetores A_eq e B_eq da Equação de Tempo ocupado pelas Tarefas

Tendo essas variáveis definidas eu parto para a definição da equação da segunda restrição. Identificando, se para cada tarefa que pode ser executada de cada máquina existe a tarefa 1, depois 2, 3, até o número de tarefas definido no primeiro input.

Multiplicando as variáveis do lado esquerdo da equação por 1, caso exista e por 0 caso não exista e adicionando no vetor A_eq (lado esquerdo). O vetor B_eq (lado direito), receberá para cada iteração o valor de horas usado pela tarefa em específico.

4. Definição dos vetores A_ineq e B_ineq da Inequação de Tempo máximo utilizado das máquinas

Definimos nesse passo a primeira restrição. Iterando uma quantidade equivalente ao número de máquinas, multiplicamos uma variável à esquerda por 1 para cada tarefa que a máquina executa, e por 0 para cada execução de tarefa que é feita por outras máquinas. Adicionando ao vetor A_ineq (lado esquerdo) os valores 0 ou 1 para cada tarefa, e ao vetor B_ineq (lado direito) o tempo máximo que cada máquina, que está sendo iterada, pode executar tarefas.

5. Envio para a lib calcular a solução utilizando o método Simplex.

Apenas enviamos os vetores Custo, A_eq, B_eq, A_ineq e B_ineq na função *linprog()* da lib. Recebemos o resultado e o armazenamos na variável *Resultado*.

6. Printar o resultado (Saída)

Por fim printamos o resultado no formato solicitado nos exemplos da descrição do trabalho, formatando e mudando o tipo para alcançarmos o resultado esperado.

4. Exemplos

Nesse campo será utilizado o primeiro exemplo informado na descrição do trabalho. Seguindo os passos exibidos na terceira seção, primeiro recebemos os valores inputados, que nesse caso seriam:

Exemplo 1)

2 2
10
10
100 20
50 10
2
1
2
1
1

Tendo os valores definidos podemos fazer o passo 2, construir o vetor de Custos, ficando:

[100, 100, 50, 0]

No passo 3, definimos:

A_eq = [[1, 0, 1, 0], [0, 1, 0, 0]]

B_eq = [10, 10]

No passo 4:

A_ineq = [[1, 1, 0], [0, 0, 1]]
B_ineq = [20, 10]

No passo 5, recebemos o resultado, contendo um objeto, entre elas o resultado da minimização com key = *fun*, e um array contendo o tempo utilizado em cada tarefa para cada máquina, key = *x,* nesse caso:

fun: 1500.0
x: [0, 10, 10, 0]

Formatando as informações no passo 6, temos como saída:

0.0 10.0
10.0 0.0
1500.0

Exemplo 2)

Outro exemplo que pode ser dado é um exemplo onde não é possível achar uma solução factível.
Com a modelagem apresentada para resolução desse problema, qualquer exemplo onde o tempo somado de execução de todas tarefas for maior que o tempo disponível somado por todas as máquinas é um exemplo onde não é possível encontrar uma solução factível.

Um caso de solução não factível, tendo o exemplo acima como base, seria:

2 2
10
10
100 10
50 5
2
1
2
1
1

onde a soma das horas 10 + 10 = 20 < 10 + 5 = 15

Retornando assim no passo 5, uma mensagem de fracasso.

Printado no passo 6:

Phase 1 of the simplex method failed to find a feasible solution. The pseudo-objective function evaluates to 5.0e+00 which exceeds the required tolerance of 1e-09 for a solution to be considered 'close enough' to zero to be a basic solution. Consider increasing the tolerance to be greater than 5.0e+00. If this tolerance is unacceptably large the problem may be infeasible.