# Ukulelelala

*Autor: Aster Santana*

*Julho 2020*

Esta é a documentação do estudo de caso Ukulelelala, que faz parte do projeto [Learning Mip](https://mip-master.github.io/learning_mip/) mantido por Mip Master.

## Conceitos

* Modelagem com variáveis de decisão inteiras
* Notação de somatório
* Restrições Se-Então
* Restrições Big-M
* Complemento de uma variável binária
* Arquivos LP

## Enunciado do problema

Um amigo da Mr. Mip, Ted, tem uma empresa no ramo musical, uma fábrica de ukulele. A marca de Ted, *Ukulelelala* (que pode ser pronunciada cantando: Ukulele-la♬-la♫), virou um grande sucesso devido à excelente qualidade dos ukuleles produzidos por ele, e também por conta dos novos investimentos que Ted fez em divulgação.

Toda semana, Ted despacha toda sua produção para 7 grandes revendedores parceiros. Só que recentemente a demanda tem crescido tanto que Ted não tem conseguido produzir o suficiente para atender a todos os pedidos. Já que Ted não tem permissão para aumentar o preço dos ukuleles (por razões contratuais), toda semana, ele precisa decidir quantas unidades despachar para cada pedido recebido.

A tabela abaixo mostra os dados da semana passada, quando a fábrica conseguiu processar ukuleles, abaixo da demanda total, que era de unidades.

|  |  |  |  |
| --- | --- | --- | --- |
| ID do Revendedor | Preço Unitário no Atacado ($) | Qtd. do Pedido | Qtd. Despachada |
| R01 | 47,00 | 230 | 50 |
| R02 | 65,00 | 150 | 135 |
| R03 | 70,00 | 270 | 270 |
| R04 | 68,00 | 90 | 90 |
| R05 | 46,00 | 190 | 0 |
| R06 | 78,00 | 55 | 55 |
| R07 | 55,00 | 120 | 50 |

Foi um prazer para o Mr. Mip ajudar seu amigo escrevendo um algoritmo que usa MIP para chegar aos números vistos na coluna “Qtd. Despachada”.

À primeira vista, esses números podem não parecer razoáveis. Você provavelmente esperava que a ordem do R02 fossem satisfeita integralmente, enquanto que a ordem do R01 não. Esse seria o caso se o objetivo fosse apenas maximizar lucro. Nesse caso, Mr. Mip poderia ter simplesmente ordenar os revendedores por preço de venda e despachar primeiro os pedidos dos clientes que pagam mais até esgotar o estoque.

Contudo, Ted tem um contrato assinado com cada revendedor. E um dos termos do contrato diz que Ted pagará uma multa de 20 vezes o valor do preço de atacado de um ukulele se não conseguir despachar no mínimo 50 unidade para aquele revendedor na semana em questão.

Vamos ver como Mr. Mip solucionou o problema de Ted usando MIP.

## Formulação

A seguir estão as três etapas principais que o Mr. Mip tipicamente segue ao escrever uma formulação.

### Variáveis de decisão

Um truque de Mr. Mip para definir as variáveis de decisão é pensar sobre quais são as incógnitas do problema. Por exemplo, o número de ukuleles disponível para despache é conhecido, nesse caso. Da mesma forma, o preço de cada unidade a ser vendida para cada revendedor também é conhecido, está na coluna “Preço Unitário no Atacado”.

Para quais revendedores Ted irá despachar ukuleles nesse periodo? Isso não sabemos, e precisa ser decidido. Uma vez que Ted escolheu despachar para um revendedor, quantas unidades ele deve despachar? Isso também precisa ser determinado. Seguindo esse raciocínio, Mr. Mip definiu dois conjuntos de variáveis de decisão.

*Variáveis de decisão:*

assume o valor se Ted despacha para o revendedor , do contrário , para todo

o número de ukuleles despachados para revendedor , para todo

Perceba que é uma variável binária (que somente assume os valores 0 ou 1) enquanto que é uma variável inteira. Não faria sentido definir como contínua já que é impossível despachar ukuleles, por exemplo.

Para simplificar notação, vamos usar em vez de ao escrever restrições e a função objetivo. Similarmente, usaremos em vez de .

### Restrições

No melhor cenário, Ted satisfaria todos os pedidos, já que quanto mais ukuleles ele vende, mais lucro ele tem (vamos modelar isso na função objetivo). Isso significa que a otimização deve tentar incrementar cada variável o máximo possível.

Contudo, Ted não tem recursos suficientes. Sendo assim, Mr. Mip deverá modelar o fato de que a quantidade despachada não pode exceder unidades. Ele consegue isso com a seguinte restrição:

Restrições – Disponibilidade máxima:

Utilizando a notação do somatório, essa inequação é equivalente a seguinte:

A seguir, temos que considerar o fato que, se Ted despacha menos de 50 ukuleles para um revendedor ele deve pagar uma multa equivalente ao valor de 20 unidades. Mr. Mip perguntou a Ted se ele poderia considerar o despacho de menos de unidades, porém Ted disse que não.

Então, por exemplo, ou Ted despacha ao menos unidades para R01, ou ele não despacha qualquer unidade e paga a taxa de . Mr. Mip tem um artifício inteligente e muito útil para modelar esse tipo de requisito.

Restrições – Se houver despacho, faça-o com pelo menos :

Essa desigualdade diz que, **se** , ou seja, se Ted despacha pelo menos uma unidade para o revendedor , **então** deve ser no mínimo , isso significa que Ted deve despachar ao menos unidades àquele revendedor. É claro que precisamos de uma restrição dessa para cada revendedor, como veremos na formulação final.

Essa é uma restrição do tipo **“se-então”** que vale a pena anotar. Mr. Mip utiliza desse artifício com frequência e tenho certeza de que você o usará também. Aqui está uma estrutura genérica para referência:

Enunciado: se , então ;

Formulação: ;

Hipóteses: é binário e é uma constante.

O que acontece se ? Nesse caso, pode assumir qualquer valor não negativo. Mas não é o que Mr. Mip quer. Se , ou seja, se Ted não despacha qualquer unidade ao revendedor , então deve ser zero também. Mesmo estando explícito na definição de e , a otimização (ou o computador, se quiser pensar dessa forma) pode não obedecer a essa regra a menos que seja colocado na forma de uma restrição. Então, como podemos modelar “se , então deve ser zero”?

Talvez a sua primeira ideia seja a seguinte restrição:

Mas existe um problema com essa restrição. Ela faz o que queremos, mas também faz algo que não queremos. Você consegue identificar?

Quando , essa restrição força a ser zero, como desejado. Mas quando , isso restringe a ser menor ou igual a 1, o que é inaceitável. Veja como Mr. Mip resolve esse problema.

Restrições – Se não despacha, então não despacha:

Perceba que é o pedido do revendedor . Quando , essa restrição ainda força a ser zero. E quando , ela restringe a ser menor ou igual a , que também é algo que queremos. Afinal, Ted não pode despachar mais do que a quantidade pedida. Então, essa restrição modela duas condições ao mesmo tempo. Novamente, precisaremos de uma restrição como essa para cada revendedor, como veremos na formulação final.

Esse tipo de restrição em que a variável é binária, nesse caso, é usada para “desativar” outra variável, nesse caso, é uma restrição “se-então”. No entanto, esse aqui é um caso particular (ainda que bem comum) conhecida por restrição de **Big-M**.

Aqui está uma versão genérica para referência:

Enunciado: se , então ;

Formulação: ;

Hipóteses: é binário, é não negativo, e é uma constante suficientemente grande.

Na maioria dos casos, um arbitrariamente grande resulta em uma formulação correta. No entanto, é muito recomendado escolher o menor possível, já que muito grande pode comprometer significantemente o desempenho do *solver*.

### Objetivo

O objetivo desse problema é maximizar o lucro total, que é o faturamento total de vendas menos o total de multas por pedidio não suprido.

Para formular o faturamento do revendedor , simplesmente multiplicamos , a quantidade despachada, pelo preço de venda unitária do revendedor :

A formulação da multa total é um pouco mais trabalhosa. Mr. Mip formulou a multa paga para o revendedor da seguinte maneira:

Perceba que ela funciona exatamente como é de se esperar. Se , ou seja, Ted despacha para o revendedor , a quantidade no segundo parênteses é zero, o que implica .

Por outro lado, se , ou seja, Ted não despacha para o revendedor , então a quantidade no segundo parênteses é 1, nesse caso, , precisamente a multa que Ted supostamente pagaria nesse caso.

Para entender o truque de Mr. Mip, você deve pensar na variável binária como um interruptor, o modo “ligado” é e o modo desligado é . Sendo assim, podemos também pensar em como um interruptor, que faz o contrário de . Ou seja, está ligado/desligado quando está desligado/ligado.

A close up of a stereo

Description automatically generated

Em outras palavras, é o complemento de .

Já que o faturamento total é a soma dos faturamentos proveniente de cada revendedor, e a multa total é a soma das multas proveniente de cada revendedore, Mr. Mip formulou a função objetivo da seguinte maneira.

Objetivo:

Onde:

Onde é o preço de varejo em que Ted vende ao revendedor .

### Formulação final

Juntando as partes, Mr. Mip chegou a seguinte formulação:

Formulação final:

Onde é o preço de varejo e é a demanda do revendedor .

## Implementação e otimização

Abaixo, temos a implementação da formulação utilizando *gurobipy*. O código está disponível no repositório “Mip Master” no GitHub [[link](https://github.com/mip-master/learning_mip/tree/master/ukulelelala)].



Perceba que todos os dados são definidos fora do modelo de otimização. Dessa forma Mr. Mip não precisará mudar o modelo quando Ted o enviar os dados da semana seguinte. Na realidade, uma vez que Ted validar a solução, Mr. Mip modificará o código para ler os dados de um arquivo que Ted poderá atualizar sem tocar no código.

A saída do código é a seguinte:

y = **{**1**:** 50**,** 2**:** 135**,** 3**:** 270**,** 4**:** 90**,** 5**:** 0**,** 6**:** 55**,** 7**:** 50**}**

faturamento 43185.0

multa 920.0

A partir dai, Mr. Mip preencheu a coluna “Qtd. Despachada” da tabela acima.

## Desafie-se

1. Quando Mr. Mip mostrou as soluções de seu programa, Ted não gostou que iria receber somente ukuleles. Isso porque era um cliente novo e ele gostaria de despachar pelo menos do pedido. Como você modificaria/estenderia a formulação de Mr. Mip para abordar esse novo requisito? Implemente o novo modelo para observar o faturamento cai para .
2. Mais tarde, Ted percebeu que ele deveria também considerar o custo de transporte. Especificamente, existe um custo fixo de despacho para cada revendedor conforme a seguir:

Como você poderia modificar/estender mais uma vez a formulação para adequá-la à esse novo requisito? Implemente e resolva o novo modelo para concluir que o rendimento aumentará para e a multa total também aumentará para .

1. Adicione o comando mdl**.**write**(**'ukulelelala.lp'**)**ao seu código em qualquer lugar após mdl.optimize() para gerar um arquivo de texto com uma versão de seu modelo que é fácil de ler. O arquivo será salvo no mesmo diretório de seu *script*. Esse arquivo é chamado de “arquivo LP” do seu modelo, e é bastante útil para efetuar o *debug* de sua implementação. Além do mais, os *solvers* podem ler e solucionar o modelo a partir de um arquivo LP. Digamos que você tenha dois *solvers* MIP e quer ver qual deles soluciona seu modelo mais rapidamente. Então, você pode implementar o modelo com um *solver*, gerar o arquivo LP e solucionar o modelo com o outro *solver* sem ter que implementar o modelo novamente.

## Resumo

1. Problemas reais podem ter requisitos inesperados, como o despacho de no mínimo ou zero.
2. Para definir as variáveis de decisão temos que identificar as incógnitas do problema. Pergunte-se “quais são as decisões que eu deveria tomar para solucionar esse problema?” Nesse exemplo, deveríamos perguntar: Para quais revendedores Ted deve despachar? Se ele decidir despachar para algum revendedor, quantas unidades ele deveria despachar para cada um deles?
3. Uma formulação MIP pode ter múltiplos conjuntos de variáveis de decisão, e de diferentes tipos. Esse exemplo foi formulado com um conjunto de variáveis binárias e inteiras.
4. As restrições “se-então” são úteis e podem assumir várias formas.
5. Ao usar a restrição Big-M, que é um caso especial das restrições “se-então”, não se deixe enganar pelo nome “big”, que significa “grande” em inglês. Na realidade, quanto menor o *M*, melhor.
6. Toda variável binária, digamos , tem como complemento, que é . Então, quando está “ligado”, está “desligado”.
7. Quando um requerimento precisa ser formulado para vários itens do mesmo tipo, como a quantidade mínima de despacho para os sete revendedores nesse exemplo, geralmente é mais fácil formular para um e depois generalizar para os demais.
8. O arquivo LP é uma boa ferramenta para conferir se sua implementação está correta e para exportar o modelo em um formato que todos os *solvers* entendem.
9. Ao implementar uma formulação, é de boa prática manter todos os dados separados do modelo de otimização.
10. É fácil acomodar novos requisitos ou estender uma formulação já existente. Essa é uma importante característica do MIP que não se encontra em outras técnicas como heurísticas e programação dinâmica.