

# Tópicos em Otimização Combinatória

## Atividade 1

Adolfo Aires Schneider

Ana Paula da Silva

Adivair Santana Ramos

01 de Abril, 2022

MO824A

Par completação do relatório, procure descrever o problema  
Tópicos em Otimização Combinatória *sendo estudado.*

2

## 1 Fomulaçao

Considerando as seguintes variáveis de interesse:

- $X_{plf}$  sendo a quantidade de toneladas do produto  $p$  fabricado pela máquina  $l$  na fábrica  $f$
- $Y_{pjf}$  sendo a quantidade de toneladas transportada do produto  $p$  para o cliente  $j$  da fábrica  $f$ .

Com as variáveis definidas, é apresentado abaixo a formulação de Programação Linear desenvolvida, abrangendo o objetivo, as características e restrições descritas pelo enunciado da atividade 1.

$$\text{Min: } \sum_{p \in P} \sum_{l \in L} \sum_{f \in F} X_{plf} P_{plf} + \sum_{p \in P} \sum_{j \in J} \sum_{f \in F} Y_{pjf} T_{pjf} \quad (1)$$

$$\text{Sujeito a: } \sum_{p \in P} X_{plf} \leq C_{lf} \quad \forall l \in L, \forall f \in F \quad (2)$$

$$\sum_{l \in L} \sum_{p \in P} r_{mpl} X_{plf} \leq R_{mf} \quad \forall m \in M, \forall f \in F \quad (3)$$

$$\sum_{f \in F} Y_{pjf} = D_{jp} \quad \forall j \in J, \forall p \in P \quad (4)$$

$$\sum_{l \in L} X_{plf} = \sum_{j \in J} Y_{pjf} \quad \forall p \in P, \forall f \in F \quad (5)$$

$$X_{plf} \geq 0 \quad \forall p \in P, \forall l \in L, \forall f \in F \quad (6)$$

$$Y_{pjf} \geq 0 \quad \forall p \in P, \forall j \in J, \forall f \in F \quad (7)$$

A função objetivo, descrita pela Equação 1, tem como objetivo minimizar o custo total de produção e transporte das demandas dos produtos. A Equação 1 é composta pela soma das toneladas produzidas multiplicado pelo seu respectivo custo e pelo custo de transporte das toneladas transportadas de cada fabrica.

Para limitar o espaço de soluções para conter apenas soluções factíveis, de acordo com o enunciado, um conjunto de 6 restrições foram criadas:

- As restrições em (2) garantem que a quantidade produzida do produto não exceda a capacidade disponível de produção de cada máquina.
- As restrições em (3) garantem que a quantidade de matéria-prima utilizada não exceda a quantidade de matéria-prima disponível nas fábricas.
- As restrições em (4) garantem que as demandas dos clientes sejam atendidas.
- As restrições em (5) garantem que a quantidade de cada produto transportado seja igual ao total produzido em cada fábrica, para evitar produção desnecessária. *ativar o total produzido com o total transportado*.
- Pôr fim, as restrições em (6), (7) garantem que as variáveis de decisão sejam positivas.

## 2 Experimentos

Para a avaliação e validação do modelo criado, foi utilizado o resovedor comercial GUROBI com uma licença acadêmica. A implementação do modelo foi feita através da sua API em Python, dado sua maior facilidade para testes. A escolha do método para solução do modelo LP contínuo foi configurada no GUROBI como automática.

A geração das instâncias de testes também foi feita em Python, utilizando a biblioteca numpy para gerar os valores randômicos de forma uniforme.

Na execução dos experimentos computacionais, foi utilizado um computador com 16GB de memória ram, processador Intel(R) Core(TM) i5-9300H CPU de 2.40GHz e sistema operacional Windows 11.

*Descrição das instâncias?*

## 3 Resultados e Análise

Os resultados dos testes foram utilizados para criar a Tabela 1, onde são exibidos os valores das variáveis obtidos de forma randômica na geração dos testes, o custo da solução ótima e o tempo de execução do GUROBI em cada instância. Como mencionado, GUROBI conseguiu encontrar a solução ótima em todas as instâncias de forma eficiente, não sendo necessário realizar mais testes por mais tempo.

Table 1: Resultados obtidos.

Clientes	Fábricas	Máquinas	Materiais	Produtos	Custo	Tempo (s)
100	148	10	7	6	1.844660000e+05	0.60
200	310	9	5	6	3.652166500e+05	3.87
300	333	7	9	10	8.308868831e+05	11.29
400	524	5	9	7	8.734571333e+05	20.10
500	606	7	5	6	9.196780770e+05	24.78
600	653	9	6	5	9.185151667e+05	32.84
700	760	7	6	10	2.172018884e+06	246.40
800	1116	7	7	6	1.466434259e+06	221.24
900	1569	7	8	8	2.185608973e+06	436.19
1000	1603	7	8	10	3.052295421e+06	838.78

Durante a execução, o GUROBI imprime algumas informações sobre o processo de solução, inclusive os métodos que estão sendo utilizados. Na menor instância, foi indicado que o método utilizado foi o Dual Simplex. Já nos demais, foi utilizado o Método dos Pontos Interiores.

O tempo de execução foi crescente em relação ao número de clientes e o número de fábricas, porém não em uma escala linear, visto que o aumento de tamanho nas instâncias maiores representam um aumento maior de tempo em relação ao mesmo aumento de clientes nas instâncias menores.

porque isso ocorre?  
 tamanho do modelo vs. parâmetros.  
 número de variáveis e restrições?

# Relatório Atividade 01 - MO824

Aluno

Athyrson Machado Ribeiro

01 de abril de 2022

Código do Curso: 3

## Introdução

### Descrição do problema

Uma companhia possui  $F$  fábricas para atender a demanda de  $J$  clientes. Cada fábrica pode escolher dentre  $L$  máquinas e  $M$  tipos de matéria-prima para produzir  $P$  tipos de produtos. A companhia precisa desenvolver um plano de produção e transporte com o objetivo de minimizar os custos totais. Mais especificamente, a companhia deve determinar a quantidade de cada tipo de produto a ser produzida em cada máquina de cada fábrica e a quantidade que deve ser transportada de cada produto partindo de cada fábrica para cada consumidor. Os parâmetros do problema encontram-se abaixo:

- $D_{j,p}$  = demanda do cliente  $j$ , em toneladas, do produto  $p$
- $r_{m,p,l}$  = quantidade de matéria-prima  $m$ , em toneladas, necessária para produzir uma tonelada do produto  $p$  na máquina  $l$ ;
- $R_{m,f}$  = quantidade de matéria-prima  $m$ , em toneladas, disponível na fábrica  $f$ ;
- $C_{l,f}$  = capacidade disponível de produção, em toneladas, da máquina  $l$  na fábrica  $f$ ;
- $p_{p,l,f}$  = custo de produção por tonelada do produto  $p$  utilizando a máquina  $l$  na fábrica  $f$ ;
- $t_{p,f,j}$  = custo de transporte por tonelada do produto  $p$  partindo da fábrica  $f$  até o cliente  $j$ ;

### Modelo matemático

O objetivo da companhia é encontrar uma solução que, de acordo com a quantidade de clientes, fabricas, máquinas, matérias-primas e produtos, seja a que apresente os menores custos totais de transporte e produção, ao mesmo tempo que satisfaça as demandas dos clientes.

### Função objetivo

Temos então duas variáveis de decisão nesse problema,  $X_{p,l,f}$ , que é a quantidade de produtos  $p$  produzidos utilizando a máquina  $l$  na fábrica  $f$  e  $Y_{p,f,j}$  que é a quantidade de produtos  $p$  produzidos na fábrica  $f$  que devem ser entregues ao cliente  $j$ .

A função objetivo do problema é então:

$$\min \left( \sum_{p=1}^{|P|} \sum_{l=1}^{|L|} \sum_{f=1}^{|F|} X_{p,l,f} \cdot p_{p,l,f} + \sum_{p=1}^{|P|} \sum_{f=1}^{|F|} \sum_{j=1}^{|J|} Y_{p,f,j} \cdot t_{p,f,j} \right)$$

Onde o primeiro termo se refere a soma dos custos de produção para todos os produtos produzidos pela companhia e o segundo termo se refere a soma dos custos de entrega de todos os produtos entregues pela companhia. Sendo  $|P|$  a quantidade de tipos diferentes de produtos que a companhia produz;  $|L|$  a quantidade de tipos diferentes de máquinas que a companhia possui;  $|F|$  a quantidade de fábricas existentes na companhia e  $|J|$  a quantidade de possíveis clientes que a companhia atende.

## Restrições

A primeira restrição do problema é que a somatória de todos os produtos que saem para transporte partindo de todas as fábricas, para todo produto  $p$  e todo cliente  $j$ , deve ser maior ou igual à demanda do cliente  $j$  pelo produto  $p$ .

$$\sum_{f=1}^{|F|} X_{p,f,j} \geq D_{j,p} \quad \forall p \quad \forall j$$

A segunda restrição é que a somatória de todos os produtos produzidos pela fábrica  $f$  multiplicados pela respectiva quantidade de matéria-prima  $m$  necessária para produzir tal produto  $p$  na máquina  $l$  deve ser maior ou igual à quantidade  $R_{m,f}$  de matéria-prima  $m$  disponível na fábrica  $f$ , para toda fábrica  $f$  e toda matéria-prima  $m$ .

$$\sum_{p=1}^{|P|} \sum_{l=1}^{|L|} X_{p,l,f} \cdot r_{mpl} \leq R_{m,f} \quad \forall f \quad \forall m$$

Sendo  $|L|$  a quantidade total de tipos de máquinas que a companhia possui.

A terceira restrição é que a somatória de todos os produtos que são produzidos em uma máquina  $l$  na fábrica  $f$  deve ser menor ou igual a capacidade de produção  $C_{l,f}$  da máquina  $l$  na fábrica  $f$ , para toda máquina  $l$  e toda fábrica  $f$ .

$$\sum_{p=1}^{|P|} X_{p,l,f} \leq C_{l,f} \quad \forall l \quad \forall f$$

A quarta restrição é que a somatória da quantidade de um produto  $p$  produzida em todas as máquinas numa fábrica  $f$  seja igual a quantidade desse produto  $p$  que sai dessa fábrica  $f$  para todos os clientes, para toda fábrica  $f$  e todo produto  $p$ .

$$\sum_{l=1}^{|L|} X_{p,l,f} = \sum_{j=1}^{|J|} Y_{p,f,j} \quad \forall p \quad \forall f$$

A quinta restrição é que os valor de  $X_{p,l,f}$  e  $Y_{p,f,j}$  seja maior ou igual a 0 para todo produto  $p$ , máquina  $l$ , fábrica  $f$  e cliente  $j$ .

$$X_{p,l,f} \geq 0 \quad \forall p \quad \forall l \quad \forall f$$

$$Y_{p,f,j} \geq 0 \quad \forall p \quad \forall f \quad \forall j$$

*Descrição instâncias e ambiente computacional?*

## Resultados

Como demonstrado na Tabela 1, para cada instância do modelo foi possível obter o de número de variáveis. Porém o número total de restrições, o custo e o tempo para se encontrar a solução ótima só pode ser obtido para instâncias com até 500 clientes. Não foi possível executar a função `model.optimize()` para instâncias com mais de 500 clientes.

Table 1: Um nome qualquer

Clientes	Variáveis	Restrições	Custo	Tempo (s)
100	101760	33200	259328	0.738
200	393872	66130	511546	4.373
300	878832	99330	762582	10.355
400	1555792	132530	1011760	43.454
500	2420704	165460	1264267	100.578
600	3480864	-//-	-//-	-//-
700	4733024	-//-	-//-	-//-
800	6170736	-//-	-//-	-//-
900	7806096	-//-	-//-	-//-
1000	9625408	-//-	-//-	-//-

O código foi feito em Python 3 e executado numa máquina com as seguintes especificações: Intel Core i3-4005U 64 bits e RAM 4GB.

## Conclusões

De acordo com os dados demonstrados na tabela foram plotados dois gráficos, Figuras 1 e 2. No gráfico da figura 1 se observa um crescimento mais drástico do tempo de execução conforme se aumenta a quantidade de clientes, valor esse que também influencia diretamente na quantidade de fábricas. Essa curva de crescimento se assemelha a uma curva de uma função exponencial. No

*para afirmar isso seria importante se sustentar com uma regressão*

gráfico da figura 2 se observa um crescimento mais acentuado do valor da solução ótima em relação ao número de clientes. A curva de crescimento lembra uma curva de uma função linear.

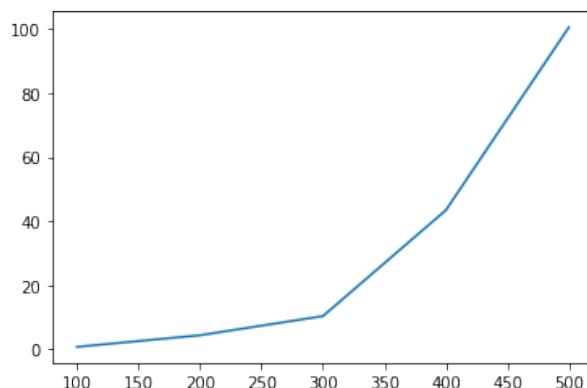


Figure 1: Figura 1: gráfico do número de clientes versus o tempo necessário para se encontrar a solução ótima do problema em cada instância - Figura de autoria própria

→ a gente só cita a origem quando a fonte é de terceiros.

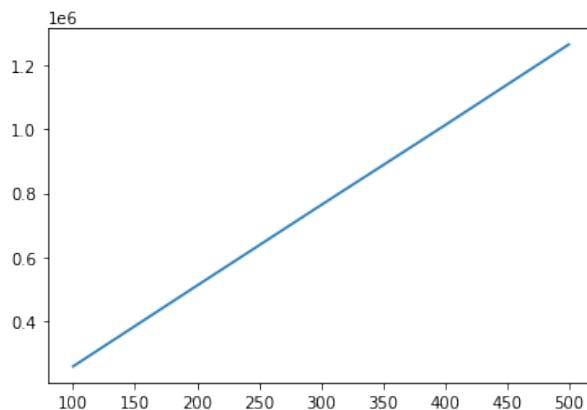


Figure 2: Figura 2: gráfico do número de clientes versus o valor da solução ótima do problema ~~Figura de autoria própria~~

→ na verdade deve ser polinomial

Inferimos então, com os dados obtidos na Tabela 1 e Figuras 1 e 2, que aumentar a quantidade de clientes gera um aumento linear no valor da solução ótima, porém um aumento exponencial no tempo necessário para se encontrar a solução ótima. Isso se deve em parte ao fato de que a função objetivo é da ordem de  $X^2 + Y^2$ , onde  $X$  e  $Y$  são as variáveis de decisão do problema.

→ acho que isso não é verdade

Para complemento do relatório é relevante descrever o problema investigado

## Atividade 1 - Programação Linear

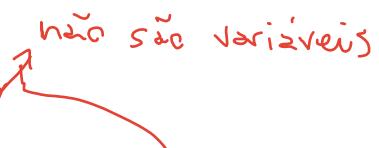
Felipe Domingues RA 171036

Felipe Pereira RA 263808

Fabio Stori RA 196631

### 1. Modelo e Restrições

Considerando as variáveis: quantidade de produto produzido (em toneladas) do tipo  $p$  pela máquina  $l$  na fábrica  $f$  definida por  $Q_{p,l,f}$ , o custo de produção do tipo  $p$  pela máquina  $l$  na fábrica  $f$  definido por  $p_{p,l,f}$ , a quantidade transportada (em toneladas) do produto  $p$  da fábrica  $f$  para o cliente  $j$  definida por  $Q_{t,p,f,j}$ , e o custo de transporte do produto  $p$  da fábrica  $f$  para o cliente  $j$  definido por  $t_{p,f,j}$ .



Desta forma, propomos o seguinte problema de programação linear:

Objetivo: Minimizar

$$\sum_{p=1}^P \sum_{l=1}^L \sum_{f=1}^F Qp_{p,l,f} * p_{p,l,f} + \sum_{p=1}^P \sum_{f=1}^F \sum_{j=1}^J Qt_{p,f,j} * t_{p,f,j} \quad (1.1)$$

Restrições

$$\sum_{p=1}^P Qt_{p,j,f} = D_{j,p} \quad \forall f \in [1, F], j \in [1, J] \quad (1.2)$$

$$\sum_{j=1}^J Qp_{p,f,j} \leq C_{l,f} \quad \forall l \in [1, L], f \in [1, F] \quad (1.3)$$

$$\sum_{p=1}^P \sum_{l=1}^L Qp_{p,f,j} * r_{m,p,l} \leq R_{m,f} \quad \forall m \in [1, M], f \in [1, F] \quad (1.4)$$

$$\sum_{l=1}^L Qp_{p,l,f} = \sum_{j=1}^J Qt_{p,f,j} \quad \forall p \in [1, P], f \in [1, F] \quad (1.5)$$

$$Qp \geq 0 \quad (1.6)$$

$$Qt \geq 0 \quad (1.7)$$

A função objetivo é a minimização dos custos totais (produção e transporte) Eq. 1.1, tal que as restrições acima são definidas por:

- Eq 1.2 - A quantidade de toneladas transportadas  $Qp_{p,l,f}$  deve ser maior ou igual à demanda de todos os clientes  $j$  por todos os produtos  $p$ .
- Eq 1.3 - A quantidade de toneladas produzidas  $Qt_{p,f,j}$  deve ser menor ou igual à capacidade de produção de todas as máquinas  $l$  em todas as fábricas  $f$ .
- Eq 1.4 - A quantidade de matéria-prima necessária em máquinas  $m$  nas fábricas  $f$  deve ser menor ou igual à quantidade disponível de matéria-prima disponível para a máquina  $m$  na fábrica  $f$ .
- Eq 1.5 - A quantidade de produto transportado deve ser igual à quantidade de produto produzido.
- Eq 1.6 e 1.7 - Restrições de não negatividade.

A nível de código, duas execuções diferentes foram descritas para os conjuntos definidos, tal que uma tolera apenas valores discretos para as variáveis  $Qp$  e  $Qt$ , e a segunda tolerando valores contínuos.

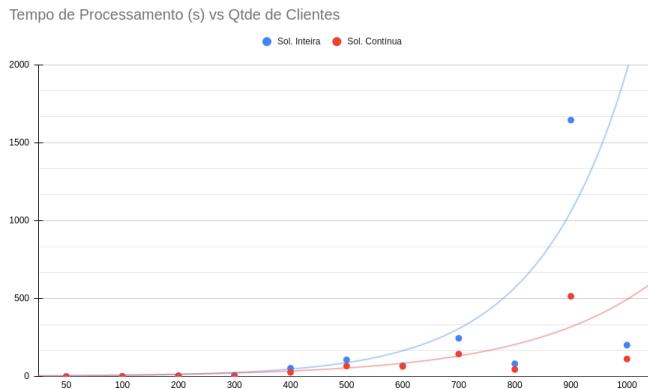


Figura 1: Tempo de execução das instâncias

## 2. Resultados

Os experimentos foram executados em uma máquina Intel(R) Core(TM) i7-7700 CPU 4.2 GHz com 32GB de RAM e sistema operacional Ubuntu Linux 20.04.

O software utilizado foi o Gurobi[1], na plataforma Jupyter[2] executando códigos Python 3[3] com 8 threads habilitadas, sem fixar o método de otimização, deixando o programa selecionar o mais adequado e com o seed para números aleatórios gerado de maneira aleatória, baseado na hora em que o código é executado. As execuções para as instâncias de até 900 clientes ocorreram sem problemas, mas a partir de 1000 algumas instâncias se tornavam muito pesadas, ocupando quase toda a memória disponível e eram encerradas pelo sistema operacional. A instância de 1000 contida nesse relatório foi uma das poucas executadas com sucesso, tendo um custo mais baixo pelos valores gerados aleatoriamente.

Na Figura 1 temos a comparação entre os tempos de execução e seus respectivos custos obtidos na Tabela 1.

Como as instâncias foram geradas?

## 3. Análise e Observações Finais

Nota-se na Figura 1 que a solução do problema com variáveis contínuas é resolvidas mais rapidamente, com exceção de algumas flutuações. O custo para solução contínua é sempre menor ou igual ao custo da solução inteira, Tabela 1. Isso é esperado, uma vez que o conjunto dos números inteiros é um subconjunto dos reais, dessa forma a solução inteira é mais restritiva.

Ainda sobre o tempo de execução, podemos verificar que as soluções contínuas possuem uma tendência de crescimento, relativa à quantidade de clientes, praticamente linear; enquanto esta mesma tendência, para as soluções inteiras, é exponencial, o que pode ser um fator importante para problemas complexos.

→ pelo gráfico que foi plotado não parece linear e, de fato, é polinomial no tamanho da instância.

Relação parâmetros vs. tamanho do modelo?

# Clientes	Custo Mínimo	
	Qt e Qp discretos	Qt e Qp contínuos
50	112644	112644,0
100	241077	241077,0
200	552538	552516,156
300	465028	465010,265
400	1091370	1091356,699
500	1520867	1520846,380
600	1508866	1508786,313
700	1973413	1973348,132
800	1234629	1234596,898
900	2723815	2723747,580
1000	1819523	1819498,496

Tabela 1: Custo mínimo obtido para cada instância.

descreva  
brevemente

Outro ponto importante para o desempenho do algoritmo é a forma como o problema é modelado. Durante a fase de modelagem do problema matemático, fizemos alguns testes que foram muito mais inefficientes do que o modelo final, demonstrando que a forma como definimos a função objetivo e as restrições associadas, tem impacto direto na complexidade do algoritmo e, portanto, na capacidade que os recursos possuem em processar tal algoritmo, assim como o tempo de execução necessário.

Embora haja uma tendência em aumentar o tempo de processamento conforme o número de clientes (e as demais variáveis) aumentavam, a complexidade depende muito dos valores gerados para as demais variáveis, sendo que as instâncias geradas para 1000 clientes que executaram com sucesso tinham um custo computacional menor que o problema gerado para 700 e 900 clientes.

Todos os códigos no formato Jupyter Notebook (Python) assim como os CSVs com os resultados e detalhes de todas as variáveis das instâncias estão disponíveis junto a esse documento.

## Referências

- [1] L. Gurobi Optimization, “Gurobi optimizer reference manual,” 2022.
- [2] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, “Jupyter notebooks – a publishing format for reproducible computational workflows,” in *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (F. Loizides and B. Schmidt, eds.), pp. 87 – 90, IOS Press, 2016.

- [3] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

**MO824A/MC859A – Tópicos em Otimização Combinatória 1S2022**  
**Atividade 1 - Relatório**

Gabriela Arcifa De Resende, Gabriel Bezerra Castelo Branco, Gabriel Gomes De Siqueira

**Primeiro de Abril de 2022**

## Introdução

O seguinte problema de programação linear foi modelado e solucionado utilizando o software Gurobi:

"Uma companhia possui **F fábricas** para atender a demanda de **J clientes**. Cada fábrica pode escolher dentre **L máquinas** e **M tipos de matéria-prima** para produzir **P tipos de produtos**. A companhia precisa desenvolver um plano de produção e transporte com o objetivo de minimizar os custos totais. Mais especificamente, a companhia deve determinar a quantidade de cada tipo de produto a ser produzida em cada máquina de cada fábrica e a quantidade que deve ser transportada de cada produto partindo de cada fábrica para cada consumidor. Os parâmetros do problema encontram-se abaixo:

- $D_{j,p}$  = demanda do cliente  $j$ , em toneladas, do produto  $p$ ;
- $r_{m,p,l}$  = quantidade de matéria-prima  $m$ , em toneladas, necessária para produzir uma tonelada do produto  $p$  na máquina  $l$ ;
- $R_{m,f}$  = quantidade de matéria-prima  $m$ , em toneladas, disponível na fábrica  $f$ ;
- $C_{l,f}$  = capacidade disponível de produção, em toneladas, da máquina  $l$  na fábrica  $f$ ;
- $p_{p,l,f}$  = custo de produção por tonelada do produto  $p$  utilizando a máquina  $l$  na fábrica  $f$ ;
- $t_{p,f,j}$  = custo de transporte por tonelada do produto  $p$  partindo da fábrica  $f$  até o cliente  $j$ ;

## Modelo matemático

apresente o modelo e descreva o significado das variáveis de decisão e das restrições.

→ descrição do modelo, variáveis, parâmetros, restrições, domínios

O modelo matemático obtido considera QP a quantidade a produzir em tonelada do produto  $p$ , utilizando a máquina  $l$ , na fábrica  $F$  e QT a quantidade a transportar em tonelada do produto  $p$ , partindo da fábrica  $f$  até o cliente  $j$ . Desta forma, o custo total mínimo de produção e transporte dos produtos pode ser descrito pela quantidade a produzir (QP) multiplicada pelo custo de produção, para cada tonelada produzida, em cada máquina de cada fábrica, somado da quantidade a transportar (QT) multiplicada pelo custo de transporte, para cada tonelada transportada, de cada fábrica para cada cliente. Para isso, deve-se considerar que a quantidade a produzir em todas as fábricas, para cada cliente e para cada produto deveria ser pelo menos a demanda dos clientes por cada produto. Considerou-se também que para isso a quantidade de matéria-prima disponível nas fábricas deveria ser pelo menos a quantidade de matéria-prima necessária para produzir os produtos multiplicado pela quantidade de produtos a serem produzidos, para cada produto, em cada máquina. Além disso, a capacidade de produção deveria ser pelo menos a quantidade a ser produzida para cada um dos produtos e, por fim, a quantidade a produzir de todas as máquinas deveria ser a mesma quantidade a ser transportada para todos os clientes.

### Variáveis de decisão:

\*abordagem com duas variáveis de custo.

$QP_{p,l,f}$  = quantidade a produzir em tonelada do produto  $p$ , utilizando a máquina  $l$ , na fábrica  $F$ .

*faltav relações  
com as equações do modelo*

$QT_{p,f,j}$  = quantidade a transportar em tonelada do produto p, partindo da fábrica f até o cliente j.

### Parâmetros:

Os parâmetros do problema encontram-se abaixo:

- $D_{j,p}$  = demanda do cliente j, em toneladas, do produto p;
- $r_{m,p,l}$  = quantidade de matéria-prima m, em toneladas, necessária para produzir uma tonelada do produto p na máquina l;
- $R_{m,f}$  = quantidade de matéria-prima m, em toneladas, disponível na fábrica f;
- $C_{l,f}$  = capacidade disponível de produção, em toneladas, da máquina l na fábrica f;
- $p_{p,l,f}$  = custo de produção por tonelada do produto p utilizando a máquina l na fábrica f;
- $t_{p,f,j}$  = custo de transporte por tonelada do produto p partindo da fábrica f até o cliente j;

### Restrições:

$$D_{j,p} \leq \sum_f QT_{p,f,j} \quad \forall_{p,j}$$

$$R_{m,f} \geq \sum_p \sum_l r_{m,p,l} * QP_{p,l,f} \quad \forall_{f,m}$$

$$C_{l,f} \geq \sum_p QP_{p,l,f} \quad \forall_{l,f}$$

$$\sum_l QP_{p,l,f} = \sum_j QT_{p,f,j} \quad \forall_{p,f}$$

### Domínios:

Função objetivo:

$$Q = \sum_p \sum_l \sum_f QP_{p,l,f} \cdot p_{p,l,f} + \sum_p \sum_f \sum_j QT_{p,f,j} \cdot t_{p,f,j}$$

### Experimentos

tabela de resultados contendo, para cada instância: quantidade de variáveis do modelo, quantidade de restrições, custo da solução e tempo de execução  
 -> descrição dos experimentos, configuração da máquina, geração de instâncias, linguagem de programação

Gerou-se 10 instâncias aleatórias variando a quantidade de clientes  $|J| = \{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000\}$  e os demais parâmetros de maneira aleatória e uniforme conforme os seguintes intervalos inteiros:

- $|F| \in [|J|, 2|J|]$  - Fábricas
- $|L| \in [5, 10]$  - Máquinas
- $|M| \in [5, 10]$  - tipos de matéria prima
- $|P| \in [5, 10]$  - tipos de produtos
- $D_{j,p} \in [10, 20]$  - demandas de cliente
- $r_{m,p,l} \in [1, 5]$  - quantidades de matéria prima necessária
- $R_{m,f} \in [800, 1000]$  - quantidades de matéria prima disponível
- $C_{l,f} \in [80, 100]$  - capacidades de produção
- $p_{p,l,f} \in [10, 100]$  - custos de produção
- $t_{p,f,j} \in [10, 20]$  - custos de transporte

Ambiente computacional?

O problema foi resolvido utilizando o solver Gurobi na linguagem Python 3. A máquina

Instância (J)	Quantidade de variáveis	Quantidade de restrições	Custo da solução	Tempo de execução
100	[1e+00, 5e+00]	[0e+00, 0e+00]	2.76982e+05	2.01 s
200	[1e+00, 5e+00]	[0e+00, 0e+00]	3.03176e+05	16.68 s
300	[1e+00, 5e+00]	[0e+00, 0e+00]	7.30718e+05	82.7 s
400	[1e+00, 5e+00]	[0e+00, 0e+00]	6.36156e+05	327.4s
500	[1e+00, 5e+00]	[0e+00, 0e+00]	1.226226e+06	993.9 s
600	[1e+00, 5e+00]	[0e+00, 0e+00]	1.107189e+06	4789.5 s
700	[1e+00, 5e+00]	[0e+00, 0e+00]	1.710741e+06	18235.8 s

### Análise

O tempo de execução aumenta de forma exponencial dada a complexidade de resolução de problemas lineares pelo Gurobi, para domínios com  $J \geq 800$  o tempo de execução é excedido de forma que um computador convencional não é capaz de chegar na solução.

→ na verdade polinomial

relação tamanho do modelo vs tempo  
Custo da Solução vs. números de clientes

For completeness, you should include the problem's description

UNIVERSIDADE ESTADUAL DE CAMPINAS  
INSTITUTE OF COMPUTING

TOPICS IN COMBINATORIAL OPTIMIZATION  
MO824

## EXERCISE 1

STUDENT: Gian Franco Joel Condori Luna  
RA: 234826

### OBJETIVO

The objective of this activity is to model a linear programming problem and the solution of this model using the Gurobi software

### DEVELOPMENT OF EXERCISE

#### 1.- Mathematical model

##### 1.1 Objetive Function

$$\min z = \sum_{p \in P} \sum_{f \in F} \left( \sum_{l \in L} x_{p,f,l} p_{p,l,f} + \sum_{j \in J} y_{p,f,j} t_{p,f,j} \right)$$

The variable "x" represents the quantity in tons of the product "p" that a machine "l" produces in the factory "f" for the client "j".

The variable "y" represents the quantity in transport of the product "p" starting from the factory "f" to the client "j"

##### 1.2 Restrictions

$$D_{j,p} \leq \sum_{f \in F} y_{p,f,j} \quad \forall p \in P \quad \forall j \in J.$$

$$\sum_{l \in L} x_{p,l,f} = \sum_{j \in J} y_{p,f,j} \quad \forall p \in P \quad \forall f \in F.$$

$$R_{m,f} \geq \sum_{p \in P} \sum_{l \in L} x_{p,l,f} r_{m,p,l} \quad \forall f \in F \quad \forall m \in M$$

$$C_{l,f} \geq \sum_{p \in P} x_{p,l,f} \quad \forall l \in L \quad \forall f \in F.$$

$$x_{p,l,f} \geq 0 \quad \forall p \in P \quad \forall l \in L \quad \forall f \in F$$

$$y_{p,l,j} \geq 0 \quad \forall p \in P \quad \forall l \in L \quad \forall j \in J.$$

You haven't explained your model.

### 1.3 Result

Instancia J	X variables	Y variables	Demand	Prod_equals_transp	Material	Capacity	Objetive func	Time Ejecucion (seconds)
100	4060	81200	700	812	580	580	1254910.0	1.22
200	9765	279000	1000	1395	1395	1953	1793768.0	3.66
300	39830	1707000	3000	5690	4552	3983	5409044.0	19.77
400	20610	916000	2000	2290	4122	4122	3593297.0	11.82
500	44750	2237500	2500	4475	6265	8950	4504440.0	29.56
600	26796	2296800	3600	3828	3828	4466	6438896.0	30.08
700	70488	5482400	5600	7832	5874	8811	10044034.0	139.15
800	104184	10418400	7200	13023	14470	11576	12993386.0	412.42
900	45720	8229600	5400	9144	10668	7620	9711961.0	490.7
1000	46160	9232000	8000	9232	5770	5770	14290822.0	475.86

### 1.2 Data analysis

Ambiente Competitivo?  
Relações tamanho do modelo vs tempo?  
Custo vs número clientes?

When carrying out several tests we were able to verify that in a first attempt it did not manage to execute all the test data, it only stayed at 300 or maximum 500, so we used a computer with more RAM (16 RAM) and we managed to finish all the tests that the teacher requested.

Based on the table we can see that if there are J=100 clients the execution time is low (1.2 seconds) we can say that up to the instance J=600 the time is relatively low (30.08) but when we are in the instance J=700 there is a takeoff of almost five times more than the previous instance (139.15) that is where we can appreciate GUROBI's ability to solve problems that require many mathematical operations.

This is where the objective function is defined in the code:

```
# Variables declaration
```

```

x = [ [ [ model.addVar(lb=0.0, ub=GRB.INFINITY, obj=C_p[p][l][f] ,
vtype=GRB.INTEGER, name=f"x[{p}][{l}][{f}]") for f in range(F) ] for l
in range(L) ] for p in range(P) ]

y = [ [ [ model.addVar(lb=0.0, ub=GRB.INFINITY, obj=C_t[p][f][j] ,
vtype=GRB.INTEGER, name=f"y[{p}][{f}][{j}]") for j in range(J) ] for f
in range(F) ] for p in range(P) ]

```

Here we define the constraints:

```

# RESTRICTIONS

model.addConstrs((sum([y[p][f][j] for f in range(F)]) == D[j][p] for p
in range(P) for j in range(J)),name="demand")

model.addConstrs((sum(x[p][l][f] for l in range(L)) - sum(y[p][f][j]
for j in range(J)) == 0 for f in range(F) for p in
range(P)),name="prod_equals_transp")

model.addConstrs((sum([x[p][l][f] for l in range(L) for p in range(P)]) <=
R[m][f] for f in range(F) for m in range(M)),name="material")

model.addConstrs((sum([x[p][l][f] for p in range(P)]) <= C[l][f] for f
in range(F) for l in range(L)),name="capacity")

```

I use this code in python to be able to verify that my variables have the correct dimensions:

```

for i in range(1,11):

    J, F, L, M, P, D, r, R, C, C_p, C_t = g.gen(i*100)

    print("===== INICIO =====")

    print("Instância = ",J, F, L, M, P)

    print("D => ", numpy.shape(D))

    print("r => ", numpy.shape(r))

```

```

print("R => ", numpy.shape(R))

print("C => ", numpy.shape(C))

print("C_p => ", numpy.shape(C_p))

print("C_t => ", numpy.shape(C_t))

print("-----")

s.solve(J, F, L, M, P, D, r, R, C, C_p, C_t)

print("===== FIN =====")

```

This can be submitted as a separate file

**ANEXO**

#### **Result of the execution of the program in python gurobi**

```

=====
INICIO
=====
Instância = 100 116 5 5 7
D => (100, 7)
r => (5, 7, 5)
R => (5, 116)
C => (5, 116)
C_p => (7, 5, 116)
C_t => (7, 116, 100)

-----
Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 2672 rows, 85260 columns and 190820 nonzeros
Model fingerprint: 0xaab893b6
Variable types: 0 continuous, 85260 integer (0 binary)
Coefficient statistics:
    Matrix range [1e+00, 1e+00]
    Objective range [1e+01, 1e+02]
    Bounds range      [0e+00, 0e+00]
    RHS range        [1e+01, 1e+03]
Found heuristic solution: objective 747546.00000
Presolve removed 580 rows and 0 columns
Presolve time: 0.26s
Presolved: 2092 rows, 85260 columns, 170520 nonzeros
Variable types: 0 continuous, 85260 integer (0 binary)
Found heuristic solution: objective 772716.00000

```

Root relaxation: objective 1.254910e+06, 7174 iterations, 0.58 seconds (0.30 work units)

	Nodes	Current Node	Objective Bounds			Work
	Expl	Unexpl	Obj	Depth	IntInf	Incumbent BestBd Gap   It/Node Time
*	0	0	0	1254910.0000	1254910.00	0.00% - 1s

Explored 1 nodes (7174 simplex iterations) in 1.22 seconds (0.86 work units)  
Thread count was 8 (of 8 available processors)

Solution count 3: 1.25491e+06 772716 747546

Optimal solution found (tolerance 1.00e-04)

Best objective 1.254910000000e+06, best bound 1.254910000000e+06, gap 0.0000%

===== FIN =====

===== INICIO =====

Instância = 200 279 7 5 5

D => (200, 5)

r => (5, 5, 7)

R => (5, 279)

C => (7, 279)

C\_p => (5, 7, 279)

C\_t => (5, 279, 200)

Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)

Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 5743 rows, 288765 columns and 626355 nonzeros

Model fingerprint: 0x7396817c

Variable types: 0 continuous, 288765 integer (0 binary)

Coefficient statistics:

Matrix range [1e+00, 1e+00]

Objective range [1e+01, 1e+02]

Bounds range [0e+00, 0e+00]

RHS range [1e+01, 1e+03]

Found heuristic solution: objective 1041047.0000

Presolve removed 1395 rows and 0 columns

Presolve time: 1.21s

Presolved: 4348 rows, 288765 columns, 577530 nonzeros

Variable types: 0 continuous, 288765 integer (0 binary)

Found heuristic solution: objective 1094267.0000

Root simplex log...

Iteration	Objective	Primal Inf.	Dual Inf.	Time
11687	1.9670600e+06	1.629288e+04	0.000000e+00	5s
15733	1.7937680e+06	0.000000e+00	0.000000e+00	6s

Root relaxation: objective 1.793768e+06, 15733 iterations, 3.66 seconds (2.07 work units)

Nodes		Current Node		Objective Bounds		Work			
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
*	0	0	0		1793768.0000	1793768.00	0.00%	-	6s

Explored 1 nodes (15733 simplex iterations) in 6.43 seconds (4.06 work units)  
Thread count was 8 (of 8 available processors)

Solution count 3: 1.79377e+06 1.09427e+06 1.04105e+06

Optimal solution found (tolerance 1.00e-04)

Best objective 1.793768000000e+06, best bound 1.793768000000e+06, gap 0.0000%

===== FIN =====

===== INICIO =====

Instância = 300 569 7 8 10

D => (300, 10)

r => (8, 10, 7)

R => (8, 569)

C => (7, 569)

C\_p => (10, 7, 569)

C\_t => (10, 569, 300)

---

Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)

Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 17225 rows, 1746830 columns and 3812300 nonzeros

Model fingerprint: 0x54c95b7b

Variable types: 0 continuous, 1746830 integer (0 binary)

Coefficient statistics:

Matrix range [1e+00, 1e+00]

Objective range [1e+01, 1e+02]

Bounds range [0e+00, 0e+00]

RHS range [1e+01, 1e+03]

Found heuristic solution: objective 3186812.0000

Presolve removed 4552 rows and 0 columns (presolve time = 5s) ...

Presolve removed 4552 rows and 0 columns

Presolve time: 5.44s

Presolved: 12673 rows, 1746830 columns, 3493660 nonzeros

Variable types: 0 continuous, 1746830 integer (0 binary)

Found heuristic solution: objective 3364526.0000

Deterministic concurrent LP optimizer: primal and dual simplex

Showing first log only...

Root simplex log...

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	3.3351060e+06	1.709125e+03	1.260644e+12	15s
141469	5.4090440e+06	0.000000e+00	0.000000e+00	19s

Concurrent spin time: 0.02s

Solved with primal simplex

Root relaxation: objective 5.409044e+06, 141469 iterations, 9.01 seconds (2.97 work units)

Nodes	Current Node	Objective	Bounds	Work					
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
*	0	0	0		5409044.0000	5409044.00	0.00%	-	19s

Explored 1 nodes (141469 simplex iterations) in 19.77 seconds (15.61 work units)

Thread count was 8 (of 8 available processors)

Solution count 3: 5.40904e+06 3.36453e+06 3.18681e+06

Optimal solution found (tolerance 1.00e-04)

Best objective 5.409044000000e+06, best bound 5.409044000000e+06, gap 0.0000%

===== FIN =====

===== INICIO =====

Instância = 400 458 9 9 5

D => (400, 5)

r => (9, 5, 9)

R => (9, 458)

C => (9, 458)

C\_p => (5, 9, 458)

C\_t => (5, 458, 400)

Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)

Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 12534 rows, 936610 columns and 2058710 nonzeros

Model fingerprint: 0x8720ba59

Variable types: 0 continuous, 936610 integer (0 binary)

Coefficient statistics:

Matrix range [1e+00, 1e+00]

Objective range [1e+01, 1e+02]

Bounds range [0e+00, 0e+00]

RHS range [1e+01, 1e+03]

Found heuristic solution: objective 2072624.0000

Presolve removed 3955 rows and 0 columns

Presolve time: 2.88s

Presolved: 8579 rows, 936610 columns, 1880735 nonzeros

Variable types: 0 continuous, 936610 integer (0 binary)

Found heuristic solution: objective 2211262.0000

Root simplex log...

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	2.3933504e+08	4.715451e+06	0.000000e+00	6s
15842	3.6321860e+06	5.158475e+04	0.000000e+00	10s
21578	3.5932970e+06	0.000000e+00	0.000000e+00	12s

Root relaxation: objective 3.593297e+06, 21578 iterations, 6.09 seconds (2.81 work units)

Nodes	Current Node	Objective	Bounds	Work					
Expl	Unexpl	Obj	Depth	IntlInf	Incumbent	BestBd	Gap	It/Node	Time
*	0	0	0	3593297.0000	3593297.00	0.00%	-	11s	

Explored 1 nodes (21578 simplex iterations) in 11.82 seconds (9.53 work units)  
Thread count was 8 (of 8 available processors)

Solution count 3: 3.5933e+06 2.21126e+06 2.07262e+06

Optimal solution found (tolerance 1.00e-04)

Best objective 3.593297000000e+06, best bound 3.593297000000e+06, gap 0.0000%

===== FIN =====

===== INICIO =====

Instância = 500 895 10 7 5

D => (500, 5)

r => (7, 5, 10)

R => (7, 895)

C => (10, 895)

C\_p => (5, 10, 895)

C\_t => (5, 895, 500)

Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)

Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 22190 rows, 2282250 columns and 4877750 nonzeros

Model fingerprint: 0xeafdd8f1

Variable types: 0 continuous, 2282250 integer (0 binary)

Coefficient statistics:

Matrix range [1e+00, 1e+00]

Objective range [1e+01, 1e+02]

Bounds range [0e+00, 0e+00]

RHS range [1e+01, 1e+03]

Found heuristic solution: objective 2619057.0000

Presolve removed 5384 rows and 0 columns (presolve time = 5s) ...

Presolve removed 5384 rows and 0 columns

Presolve time: 7.31s

Presolved: 16806 rows, 2282250 columns, 4608550 nonzeros

Variable types: 0 continuous, 2282250 integer (0 binary)

Found heuristic solution: objective 2829883.0000

Deterministic concurrent LP optimizer: primal and dual simplex  
Showing first log only...

Root simplex log...

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	7.6702900e+05	1.748063e+04	1.481978e+11	22s
158434	4.0640166e+06	1.049820e+03	2.601233e+11	26s
351682	4.3819012e+06	3.834180e+02	8.278577e+10	32s
429523	4.4587591e+06	1.759514e+02	1.460430e+11	36s
532392	4.5023649e+06	1.920674e+01	1.029599e+07	41s
552248	4.5044400e+06	0.000000e+00	0.000000e+00	45s

Concurrent spin time: 0.02s

Solved with primal simplex

Root relaxation: objective 4.504440e+06, 552248 iterations, 29.56 seconds (7.31 work units)

Nodes	Current Node	Objective	Bounds	Work					
Expl	Unexpl	Obj	Depth	IntlInf	Incumbent	BestBd	Gap	It/Node	Time

*	0	0	0	4504440.0000	4504440.00	0.00%	-	45s
---	---	---	---	--------------	------------	-------	---	-----

Explored 1 nodes (552248 simplex iterations) in 45.77 seconds (23.81 work units)  
Thread count was 8 (of 8 available processors)

Solution count 3: 4.50444e+06 2.82988e+06 2.61906e+06

Optimal solution found (tolerance 1.00e-04)

Best objective 4.504440000000e+06, best bound 4.504440000000e+06, gap 0.0000%

===== FIN =====

===== INICIO =====

Instância = 600 638 7 6 6

D => (600, 6)

r => (6, 6, 7)

R => (6, 638)

C => (7, 638)

C\_p => (6, 7, 638)

C\_t => (6, 638, 600)

Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)

Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 15722 rows, 2323596 columns and 4807968 nonzeros

Model fingerprint: 0x837ea99f

Variable types: 0 continuous, 2323596 integer (0 binary)

Coefficient statistics:

Matrix range [1e+00, 1e+00]  
Objective range [1e+01, 1e+02]  
Bounds range [0e+00, 0e+00]  
RHS range [1e+01, 1e+03]  
Found heuristic solution: objective 3824102.0000  
Presolve removed 3828 rows and 0 columns (presolve time = 5s) ...  
Presolve removed 3828 rows and 0 columns  
Presolve time: 7.21s  
Presolved: 11894 rows, 2323596 columns, 4647192 nonzeros  
Variable types: 0 continuous, 2323596 integer (0 binary)  
Found heuristic solution: objective 3999479.0000

Deterministic concurrent LP optimizer: primal and dual simplex  
Showing first log only...

Root simplex log...

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	4.1033910e+06	2.944375e+03	1.920905e+12	22s
138067	6.3114180e+06	0.000000e+00	1.027192e+06	25s
202374	6.4388960e+06	0.000000e+00	0.000000e+00	29s

Concurrent spin time: 0.02s

Solved with primal simplex

Root relaxation: objective 6.438896e+06, 202374 iterations, 14.93 seconds (4.32 work units)

Nodes	Current Node	Objective					Bounds			Work
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time	
*	0	0	0	6438896.0000	6438896.00	0.00%	-	29s		

Explored 1 nodes (202374 simplex iterations) in 30.08 seconds (20.79 work units)  
Thread count was 8 (of 8 available processors)

Solution count 3: 6.4389e+06 3.99948e+06 3.8241e+06

Optimal solution found (tolerance 1.00e-04)  
Best objective 6.438896000000e+06, best bound 6.438896000000e+06, gap 0.0000%

===== FIN =====

===== INICIO =====

Instância = 700 979 9 6 8

D => (700, 8)

r => (6, 8, 9)

R => (6, 979)

C => (9, 979)

C\_p => (8, 9, 979)

C\_t => (8, 979, 700)

---

Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)

Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 28117 rows, 5552888 columns and 11528704 nonzeros

Model fingerprint: 0xb9e4f6a0

Variable types: 0 continuous, 5552888 integer (0 binary)

Coefficient statistics:

Matrix range [1e+00, 1e+00]

Objective range [1e+01, 1e+02]

Bounds range [0e+00, 0e+00]

RHS range [1e+01, 1e+03]

Found heuristic solution: objective 5848963.0000

Presolve removed 4895 rows and 0 columns (presolve time = 5s) ...

Presolve removed 5619 rows and 0 columns (presolve time = 10s) ...

Presolve removed 5619 rows and 0 columns (presolve time = 15s) ...

Presolve removed 5619 rows and 0 columns

Presolve time: 19.48s

Presolved: 22498 rows, 5552888 columns, 11124136 nonzeros

Variable types: 0 continuous, 5552888 integer (0 binary)

Found heuristic solution: objective 6244403.0000

Deterministic concurrent LP optimizer: primal simplex, dual simplex, and barrier

Showing barrier log only...

Root barrier log...

Elapsed ordering time = 5s

Elapsed ordering time = 10s

Elapsed ordering time = 15s

Elapsed ordering time = 20s

Elapsed ordering time = 25s

Elapsed ordering time = 30s

Elapsed ordering time = 35s

Elapsed ordering time = 40s

Elapsed ordering time = 45s

Ordering time: 47.67s

Barrier performed 0 iterations in 135.37 seconds (57.43 work units)

Barrier solve interrupted - model solved by another algorithm

Concurrent spin time: 0.55s

Solved with primal simplex

Root relaxation: objective 1.004403e+07, 700610 iterations, 96.67 seconds (13.18 work units)

Nodes	Current Node	Objective	Bounds	Work					
Expl	Unexpl	Obj	Depth	IntLnf	Incumbent	BestBd	Gap	It/Node	Time
*	0	0	0		1.004403e+07	1.0044e+07	0.00%	-	137s

Explored 1 nodes (700610 simplex iterations) in 139.15 seconds (53.38 work units)  
 Thread count was 8 (of 8 available processors)

Solution count 3: 1.0044e+07 6.2444e+06 5.84896e+06

Optimal solution found (tolerance 1.00e-04)  
 Best objective 1.004403400000e+07, best bound 1.004403400000e+07, gap 0.0000%  
 ===== FIN =====  
 ===== INICIO =====  
 Instância = 800 1447 8 10 9  
 D => (800, 9)  
 r => (10, 9, 8)  
 R => (10, 1447)  
 C => (8, 1447)  
 C\_p => (9, 8, 1447)  
 C\_t => (9, 1447, 800)

Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)  
 Thread count: 4 physical cores, 8 logical processors, using up to 8 threads  
 Optimize a model with 46269 rows, 10522584 columns and 22087008 nonzeros  
 Model fingerprint: 0xb1741133  
 Variable types: 0 continuous, 10522584 integer (0 binary)

Coefficient statistics:

Matrix range [1e+00, 1e+00]  
 Objective range [1e+01, 1e+02]  
 Bounds range [0e+00, 0e+00]  
 RHS range [1e+01, 1e+03]

Found heuristic solution: objective 7623229.0000  
 Presolve removed 0 rows and 0 columns (presolve time = 5s) ...  
 Presolve removed 0 rows and 0 columns (presolve time = 10s) ...  
 Presolve removed 13023 rows and 0 columns (presolve time = 15s) ...  
 Presolve removed 14470 rows and 0 columns (presolve time = 20s) ...  
 Presolve removed 14470 rows and 0 columns (presolve time = 25s) ...  
 Presolve removed 14470 rows and 0 columns (presolve time = 30s) ...  
 Presolve removed 14470 rows and 0 columns (presolve time = 37s) ...  
 Presolve removed 14470 rows and 0 columns  
 Presolve time: 39.64s  
 Presolved: 31799 rows, 10522584 columns, 21045168 nonzeros  
 Variable types: 0 continuous, 10522584 integer (0 binary)  
 Found heuristic solution: objective 8104147.0000

Deterministic concurrent LP optimizer: primal simplex, dual simplex, and barrier  
 Showing barrier log only...

Root barrier log...

Elapsed ordering time = 5s  
Elapsed ordering time = 10s  
Elapsed ordering time = 15s  
Elapsed ordering time = 24s  
Elapsed ordering time = 25s  
Elapsed ordering time = 30s  
Elapsed ordering time = 35s  
Elapsed ordering time = 40s  
Elapsed ordering time = 45s  
Elapsed ordering time = 50s  
Elapsed ordering time = 55s  
Elapsed ordering time = 60s  
Elapsed ordering time = 65s  
Elapsed ordering time = 70s  
Elapsed ordering time = 75s  
Elapsed ordering time = 80s  
Elapsed ordering time = 85s  
Elapsed ordering time = 90s  
Elapsed ordering time = 95s  
Elapsed ordering time = 100s  
Elapsed ordering time = 105s  
Elapsed ordering time = 110s  
Elapsed ordering time = 115s  
Elapsed ordering time = 120s  
Elapsed ordering time = 125s  
Elapsed ordering time = 130s  
Elapsed ordering time = 135s  
Elapsed ordering time = 140s  
Elapsed ordering time = 145s  
Elapsed ordering time = 150s  
Elapsed ordering time = 155s  
Elapsed ordering time = 160s  
Elapsed ordering time = 165s  
Elapsed ordering time = 170s  
Elapsed ordering time = 175s  
Ordering time: 177.71s

Barrier statistics:

AA' NZ : 1.052e+07  
Factor NZ : 6.736e+07 (roughly 5.0 GB of memory)  
Factor Ops : 3.121e+11 (roughly 12 seconds per iteration)  
Threads : 2

Barrier performed 0 iterations in 383.80 seconds (147.93 work units)  
Barrier solve interrupted - model solved by another algorithm

Concurrent spin time: 64.10s (can be avoided by choosing Method=3)

Solved with primal simplex

Root relaxation: objective 1.299339e+07, 111622 iterations, 322.04 seconds (70.18 work units)

Nodes	Current Node	Objective	Bounds	Work	
Expl	Unexpl	Obj	Depth IntInf	Incumbent BestBd Gap	It/Node Time
*	0 0	0	1.299339e+07	1.2993e+07 0.00%	- 410s

Explored 1 nodes (111622 simplex iterations) in 412.42 seconds (147.59 work units)  
Thread count was 8 (of 8 available processors)

Solution count 3: 1.29934e+07 8.10415e+06 7.62323e+06

Optimal solution found (tolerance 1.00e-04)

Best objective 1.299338600000e+07, best bound 1.299338600000e+07, gap 0.0000%

===== FIN =====

===== INICIO =====

Instância = 900 1524 5 7 6

D => (900, 6)

r => (7, 6, 5)

R => (7, 1524)

C => (5, 1524)

C\_p => (6, 5, 1524)

C\_t => (6, 1524, 900)

Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)

Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 32832 rows, 8275320 columns and 16870680 nonzeros

Model fingerprint: 0x8cb1c60d

Variable types: 0 continuous, 8275320 integer (0 binary)

Coefficient statistics:

Matrix range [1e+00, 1e+00]

Objective range [1e+01, 1e+02]

Bounds range [0e+00, 0e+00]

RHS range [1e+01, 1e+03]

Found heuristic solution: objective 5633760.0000

Presolve removed 0 rows and 0 columns (presolve time = 6s) ...

Presolve removed 9144 rows and 0 columns (presolve time = 10s) ...

Presolve removed 10668 rows and 0 columns (presolve time = 15s) ...

Presolve removed 10668 rows and 0 columns (presolve time = 20s) ...

Presolve removed 10668 rows and 0 columns (presolve time = 25s) ...

Presolve removed 10668 rows and 0 columns (presolve time = 30s) ...

Presolve removed 10668 rows and 0 columns

Presolve time: 30.04s  
Presolved: 22164 rows, 8275320 columns, 16550640 nonzeros  
Variable types: 0 continuous, 8275320 integer (0 binary)  
Found heuristic solution: objective 5860726.0000

Deterministic concurrent LP optimizer: primal simplex, dual simplex, and barrier  
Showing barrier log only...

Root barrier log...

Ordering time: 0.09s

Barrier statistics:

AA' NZ : 8.275e+06

Factor NZ : 8.579e+07 (roughly 4.0 GB of memory)

Factor Ops : 3.892e+11 (roughly 17 seconds per iteration)

Threads : 2

Iter	Objective		Residual			Compl	Time
	Primal	Dual	Primal	Dual	Time		
0	2.17047079e+08	5.59380735e+06	4.34e+02	5.05e+02	8.45e+01	150s	
1	2.47169058e+07	1.51893113e+07	3.33e+01	1.48e+02	7.06e+00	173s	
2	9.13809164e+06	1.51962327e+07	9.05e-01	4.45e+01	6.01e-01	204s	
3	8.71513252e+06	1.06444539e+07	2.66e-03	1.02e+01	1.30e-01	227s	
4	9.04323956e+06	9.85142054e+06	8.24e-04	3.77e+00	5.38e-02	259s	
5	9.25098580e+06	9.76144158e+06	3.24e-03	1.55e+00	3.29e-02	288s	
6	9.29880341e+06	9.74966001e+06	1.19e-02	1.42e+00	2.91e-02	314s	
7	9.35117884e+06	9.73386051e+06	6.39e-02	1.20e+00	2.47e-02	338s	
8	9.40016313e+06	9.72630339e+06	1.02e-01	1.03e+00	2.11e-02	359s	
9	9.45433115e+06	9.71435516e+06	1.03e-01	8.59e-01	1.69e-02	381s	
10	9.51275122e+06	9.71141456e+06	8.11e-02	2.72e-01	1.24e-02	407s	
11	9.57809930e+06	9.71183147e+06	5.52e-02	1.70e-01	8.32e-03	430s	
12	9.63961935e+06	9.71275901e+06	2.41e-02	4.94e-02	4.50e-03	452s	
13	9.67986522e+06	9.71241714e+06	3.66e-01	2.16e-02	2.54e-03	472s	

Barrier performed 13 iterations in 484.53 seconds (187.87 work units)  
Barrier solve interrupted - model solved by another algorithm

Concurrent spin time: 1.58s

Solved with dual simplex

Root relaxation: objective 9.711961e+06, 937668 iterations, 420.92 seconds (65.17 work units)

Nodes	Current Node	Objective	Bounds	Work					
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time

\* 0 0 0 9711961.0000 9711961.00 0.00% - 488s

Explored 1 nodes (937668 simplex iterations) in 490.73 seconds (124.90 work units)  
Thread count was 8 (of 8 available processors)

Solution count 3: 9.71196e+06 5.86073e+06 5.63376e+06

Optimal solution found (tolerance 1.00e-04)

Best objective 9.711961000000e+06, best bound 9.711961000000e+06, gap 0.0000%

===== FIN =====

===== INICIO =====

Instância = 1000 1154 5 5 8

D => (1000, 8)

r => (5, 8, 5)

R => (5, 1154)

C => (5, 1154)

C\_p => (8, 5, 1154)

C\_t => (8, 1154, 1000)

---

Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)

Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 28772 rows, 9278160 columns and 18787120 nonzeros

Model fingerprint: 0xbac6002e

Variable types: 0 continuous, 9278160 integer (0 binary)

Coefficient statistics:

Matrix range [1e+00, 1e+00]

Objective range [1e+01, 1e+02]

Bounds range [0e+00, 0e+00]

RHS range [1e+01, 1e+03]

Found heuristic solution: objective 8389227.0000

Presolve removed 0 rows and 0 columns (presolve time = 5s) ...

Presolve removed 0 rows and 0 columns (presolve time = 10s) ...

Presolve removed 4616 rows and 0 columns (presolve time = 15s) ...

Presolve removed 5770 rows and 0 columns (presolve time = 21s) ...

Presolve removed 5770 rows and 0 columns (presolve time = 25s) ...

Presolve removed 5770 rows and 0 columns (presolve time = 32s) ...

Presolve removed 5770 rows and 0 columns

Presolve time: 33.35s

Presolved: 23002 rows, 9278160 columns, 18556320 nonzeros

Variable types: 0 continuous, 9278160 integer (0 binary)

Found heuristic solution: objective 8674864.0000

Deterministic concurrent LP optimizer: primal simplex, dual simplex, and barrier

Showing barrier log only...

Root barrier log...

Elapsed ordering time = 5s

Elapsed ordering time = 10s  
 Elapsed ordering time = 15s  
 Elapsed ordering time = 20s  
 Elapsed ordering time = 26s  
 Elapsed ordering time = 30s  
 Elapsed ordering time = 35s  
 Elapsed ordering time = 40s  
 Elapsed ordering time = 45s  
 Elapsed ordering time = 50s  
 Elapsed ordering time = 55s  
 Elapsed ordering time = 60s  
 Elapsed ordering time = 65s  
 Elapsed ordering time = 70s  
 Elapsed ordering time = 75s  
 Elapsed ordering time = 80s  
 Elapsed ordering time = 85s  
 Elapsed ordering time = 90s  
 Elapsed ordering time = 95s  
 Elapsed ordering time = 100s  
 Elapsed ordering time = 105s  
 Elapsed ordering time = 110s  
 Elapsed ordering time = 115s  
 Elapsed ordering time = 120s  
 Elapsed ordering time = 125s  
 Elapsed ordering time = 130s  
 Elapsed ordering time = 135s  
 Elapsed ordering time = 140s  
 Elapsed ordering time = 145s  
 Elapsed ordering time = 150s  
 Elapsed ordering time = 155s  
 Ordering time: 156.01s

Barrier statistics:

AA' NZ : 9.278e+06  
 Factor NZ : 4.256e+07 (roughly 4.0 GB of memory)  
 Factor Ops : 1.381e+11 (roughly 8 seconds per iteration)  
 Threads : 2

Iter	Objective		Residual			Compl Time
	Primal	Dual	Primal	Dual	Time	
0	2.11316935e+08	8.35431646e+06	2.83e+02	5.19e+02	7.45e+01	326s
1	7.10973223e+07	1.79227374e+07	8.23e+01	2.98e+02	2.21e+01	335s
2	1.89969427e+07	2.29426041e+07	8.18e+00	9.19e+01	2.74e+00	344s
3	1.32137847e+07	2.13784712e+07	5.47e-01	4.24e+01	6.50e-01	353s
4	1.28539778e+07	1.74452097e+07	4.57e-04	2.23e+01	2.87e-01	364s
5	1.31228700e+07	1.45927547e+07	3.97e-04	6.03e+00	8.96e-02	376s
6	1.32986512e+07	1.43928719e+07	2.88e-04	4.13e+00	6.61e-02	387s
7	1.33913167e+07	1.43474401e+07	4.55e-05	3.45e+00	5.73e-02	397s

8	1.34208576e+07	1.43420610e+07	7.21e-04	3.19e+00	5.50e-02	406s
9	1.34356345e+07	1.43237590e+07	7.08e-03	2.98e+00	5.29e-02	414s
10	1.35861855e+07	1.43118820e+07	1.88e-01	2.60e+00	4.35e-02	423s
11	1.37020013e+07	1.43015584e+07	2.63e-01	2.23e+00	3.61e-02	438s
12	1.37713203e+07	1.42943787e+07	2.49e-01	1.81e+00	3.12e-02	459s

Barrier performed 12 iterations in 465.66 seconds (155.96 work units)  
 Barrier solve interrupted - model solved by another algorithm

Concurrent spin time: 1.38s

Solved with dual simplex

Root relaxation: objective 1.429082e+07, 1046619 iterations, 391.31 seconds (67.00 work units)

Nodes	Current Node	Objective	Bounds	Work					
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
*	0	0	0		1.429082e+07	1.4291e+07	0.00%	-	472s

Explored 1 nodes (1046619 simplex iterations) in 475.86 seconds (113.67 work units)  
 Thread count was 8 (of 8 available processors)

Solution count 3: 1.42908e+07 8.67486e+06 8.38923e+06

Optimal solution found (tolerance 1.00e-04)  
 Best objective 1.429082200000e+07, best bound 1.429082200000e+07, gap 0.0000%  
 ===== FIN =====

Por completude, faltou incluir a descrição do problema

## MO824 - Atividade 1

Ieremies Romero

### Formulation

#### Objective Function

Our objective, as said in the activity, is to minimize the costs of production and shipment of goods from a set of factories to another set of clients. Since we have cost associated with each of those operations, we will utilize two set of variables:  $x_{p,l,f}$  to represent the amount (in tons) of product  $p \in P$  produced by machine  $l \in L$  at factory  $f \in F$  and  $y_{p,f,j}$  to represent the amount (in tons) of product  $p \in P$  transported from factory  $f \in F$  to client  $j \in J$ .

That way, we can describe our objective function as

$$\min z = \sum_{p \in P} \sum_{f \in F} \left( \sum_{l \in L} x_{p,f,l} p_{p,f,l} + \sum_{j \in J} y_{p,f,j} t_{p,f,j} \right).$$

#### Restrictions

First, we need to ensure every demand is satisfied, which means the amount of each product transported for each client should be at least greater than the client's demand. In ~~another~~ terms, we have

$$D_{p,j} \leq \sum_{f \in F} y_{p,f,j} \quad \forall p \in P \quad \forall j \in J.$$

Then, we have to ensure the amount of products produced should match the amount transported from each factory.

$$\sum_{l \in L} x_{p,f,l} = \sum_{j \in J} y_{p,f,j} \quad \forall p \in P \quad \forall f \in F.$$

Lastly, we have to be able to produce ~~X~~ said products, which means, we must have the materials required and the capacity.

$$R_{m,f} \geq \sum_{p \in P} \sum_{l \in L} x_{p,f,l} r_{m,p,l} \quad \forall f \in F \quad \forall m \in M$$

$$C_{f,l} \geq \sum_{p \in P} x_{p,f,l} \quad \forall l \in L \quad \forall f \in F.$$

To complete our restrictions, since we modeled each variable as the amount produced, they should be positive.

$$x_{p,l,f} \geq 0 \quad \forall p \in P \quad \forall l \in L \quad \forall f \in F$$

$$y_{p,l,j} \geq 0 \quad \forall p \in P \quad \forall l \in L \quad \forall j \in J.$$

### Final program

Combining everything, we have our Linear Program as follows.

$$\min z = \sum_{p \in P} \sum_{f \in F} \left( \sum_{l \in L} x_{p,l,f} p_{p,f,l} + \sum_{j \in J} y_{p,f,j} t_{p,f,j} \right)$$

subject to

$$\sum_{f \in F} y_{p,f,j} \geq D_{p,j} \quad \forall p \in P \quad \forall j \in J.$$

$$\sum_{l \in L} x_{p,f,l} - \sum_{j \in J} y_{p,f,j} = 0 \quad \forall p \in P \quad \forall f \in F.$$

$$\sum_{p \in P} \sum_{l \in L} x_{p,f,l} r_{m,p,l} \leq R_{m,f} \quad \forall f \in F \quad \forall m \in M$$

$$\sum_{p \in P} x_{p,f,l} \leq C_{f,l} \quad \forall l \in L \quad \forall f \in F.$$

$$x_{p,l,f} \geq 0 \quad \forall p \in P \quad \forall l \in L \quad \forall f \in F$$

$$y_{p,l,j} \geq 0 \quad \forall p \in P \quad \forall l \in L \quad \forall j \in J.$$

*RC*

## Results

Running our model using Gurobi 9.2 solver in a machine with a Ryzen 1800x octacore at 3.6GHz with 16GB of ram and 16GB of swap (which we saw being used). We obtained the following results.

As of work time, we can see in the Figure 1 that we have an exponential relationship between number of client and work/running time.

2

→ na verdade é polinomial,  
tudo depende bastante  
da estrutura da matriz de  
coeficientes.

Table 1: Result of running 10 random instances.

# Clients	# Variables	# Constraints	Result cost	Execution time	Work
100	146880	4128	11005953	0.8959	0.8408
200	282880	6984	20663983	1.9750	2.0171
300	1378600	12040	22862187	8.7141	10.3167
400	2400830	21023	52976044	18.7725	20.8948
500	4558284	28428	79913943	38.5863	39.2341
600	5070720	27684	67383596	52.1588	43.8014
700	5857758	37248	119142323	64.8838	51.0371
800	9353070	43124	118680540	183.6494	82.7664
900	9546845	37947	74096075	314.5878	143.8355
1000	11612258	44927	97121488	203.0819	95.4361

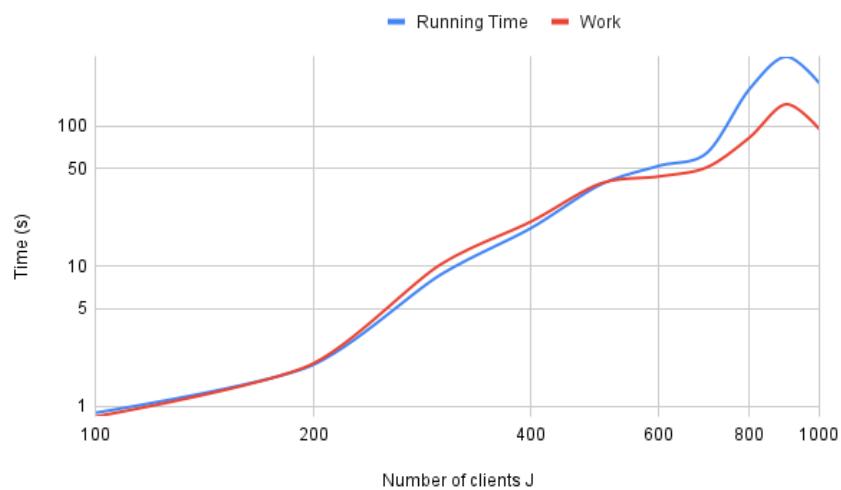


Figure 1: Plotting in logarithm scale of execution time by number of clients.

We do note the number of variables is proportional to the number of clients and we did not put the non-negativity constraints in the total number of constraints.

Also, we can note some weirdness in our data: the randomness of the instances made the instances with 700 and 800 clients have out of proportion result value, compared to its peers; we also note the execution time of the instance with 900 clients, which came out to be a lot slower than others. We do believe that anomaly maybe caused by some external factors involving the machine.

→ estrutura da instância

relação tempo vs. tamanho do modelo ?  
relação custo vs. número de clientes

# Atividade 1 - MC 859 A

## José Antonio Mauad Leis - RA 219061

### 1. Introdução do Problema

O problema proposto por essa atividade visava buscar a solução de um problema de modelagem linear com o software Gurobi. O problema consistia em descobrir as quantidades de cada tipo de produto a ser produzida em cada máquina de cada fábrica e a quantidade que deve ser transportada de cada produto partindo de cada fábrica para cada consumidor, tendo os números de fábricas F, clientes J, máquinas L, matérias-prima M e produtos P.

### 2. Modelo Matemático

Com os parâmetros passados no enunciado da atividade, podemos formar o modelo matemático padrão para esse problema. Começando pelas duas variáveis que queremos buscar:

X: Quantidade de produtos ~~X~~ do tipo p, produzidos ~~por~~ pelas máquinas l, na fábrica f.

Y: Quantidade de produtos ~~X~~ do tipo p, transportados ~~para~~ para o cliente j, pela fábrica f.

Como queremos minimizar o total desses valores, podemos tomar a soma deles como variável para minimizar.

Minimize

soma

subject to:

$$X, Y \geq 0 \quad (1)$$

$$\sum_{l,p} X \bullet r \leq R \quad \begin{matrix} \text{índices?} \\ \text{domínios?} \end{matrix} \quad (2)$$

$$soma \geq \sum_{p,l,f} \sum X \bullet p + \sum_{p,l,f} t \bullet Y \quad (3)$$

$$\sum_f Y = D \quad (4)$$

$$\sum_p X \leq C \quad (5)$$

$$\sum_f Y = \sum_l X \quad (6)$$

Primeiramente a restrição (1) diz respeito sobre a condição de ambos os valores das variáveis serem maiores que 0. A restrição (2) mostra que a quantidade de matéria-prima utilizada deve ser menor que o total disponível. A restrição (3) mostra que o custo total do processo é a soma do custo da produção com o ~~de~~ transporte. A restrição (4) mostra que o total de papéis transportados devem ser iguais à demanda dos clientes. A restrição (5) mostra que o total de papéis produzidos não pode ser maior que a capacidade da máquina, e por fim a restrição (6) diz que o total transportado por uma fábrica deve ser igual ao total produzido por ela.

*geração de instâncias? ambiente computacional?*

### 3. Resultados

Abaixo estão os resultados calculados das instâncias com o número respectivo de clientes. Para os casos de 700, 800, 900 e 1000, não foi possível concluir os cálculos devido a um erro de falta de memória (**GurobiError**: Out of memory).

Clientes	Variáveis	Custo
100	74121	152051.0
200	494495	422264.0
300	1355712	729878.0
400	1899649	976032.0
500	2807208	1378919.0
600	3246720	906012.0
700	5277087	-
800	-	-
900	-	-
1000	-	-

*tempo?*

### 4. Análise

As instâncias de valores mais baixos foram resolvidas de maneira ótima, porém para valores de clientes mais altos ocorreram problemas possivelmente de hardware ou de elaboração do código. Abaixo também seguem análises gráficas relacionando o número de clientes com o número de variáveis e o custo. Podemos observar que o custo cresceu linearmente de acordo com o aumento de clientes.

*Relação tempo vs. tamanho do modelo?*

### Clientes x Variáveis

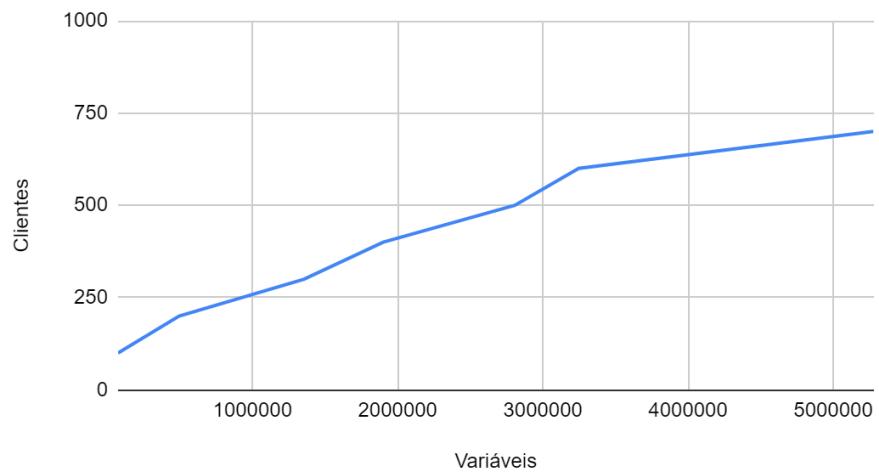


Imagen 1: gráfico de clientes x variáveis

### Clientes versus Custo

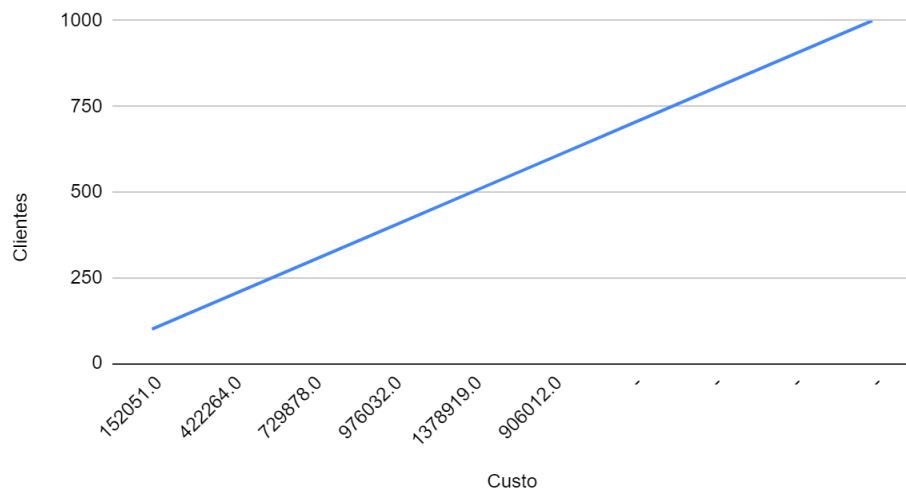


Imagen 2: gráfico de clientes x custo

## Atividade 1 - MO824A/MC859A

João Victor Aquino Batista - RA: 199798

Ítalo Fernandes Gonçalves - RA: 234990

Este relatório tem como objetivo apresentar os resultados obtidos na atividade 1 da matéria MO824A/MC859A.

### Definição Formal do Problema

Dado o problema de produção de um produto P por um conjunto fábricas que visam suprir uma demanda de clientes, temos em vista que será necessário minimizar os custos totais, ou seja, devemos minimizar a expressão que representa a soma dos custos de produção e de transporte. Para o problema em questão podemos definir as seguintes variáveis:

- $D_{j,p}$  sendo a demanda do cliente ( $j$ ), em toneladas, do produto ( $p$ );
- $r_{m,p,l}$  sendo a quantidade de matéria-prima  $m$ , em toneladas, necessária para produzir uma tonelada do produto ( $p$ ) na máquina ( $l$ );
- $R_{m,f}$  sendo a quantidade de matéria-prima ( $m$ ), em toneladas, disponível na fábrica ( $f$ );
- $C_{l,f}$  sendo a capacidade disponível de produção, em toneladas, da máquina ( $l$ ) na fábrica ( $f$ );
- $p_{p,l,f}$  sendo o custo de produção por tonelada do produto ( $p$ ) utilizando a máquina ( $l$ ) na fábrica ( $f$ );
- $t_{p,f,j}$  sendo o custo de transporte por tonelada do produto ( $p$ ) partindo da fábrica  $f$  até o cliente ( $j$ );

Sendo:

- $P$  a quantidade de produtos
- $F$  a quantidade de fábricas
- $L$  a quantidade de máquinas de cada fábrica
- $J$  a quantidade de clientes e
- $M$  a quantidade de matéria-prima disponível

### Formulação Matemática

Além das variáveis do problema, foram criadas as seguintes variáveis contínuas de decisão, que serão necessárias para aplicação ao software GUROBI:

- $X_{p,l,f}$  sendo a variável contínua que representa se foi produzido o produto ( $p$ ) na máquina ( $l$ ) da fábrica ( $f$ );
- $Y_{p,f,j}$  sendo a variável contínua que representa se foi transportando o produto ( $p$ ) da máquina ( $l$ ) e da fábrica ( $f$ );

Tendo isso em vista a definição do problema, podemos representá-lo matematicamente com a seguinte formulação:

Minimizar:

$$\sum_{p}^P \sum_{l}^L \sum_{f}^F p_{plf} * X_{plf} + \sum_{p}^P \sum_{f}^F \sum_{j}^J t_{pfj} * Y_{pfj} \quad (1)$$

sujeito a:

$$D_{jp} \leq \sum_f^F Y_{pfj}, \forall_p \forall_j \quad (2)$$

$$R_{mf} \geq \sum_p^P \sum_l^L X_{plf} * r_{mpl}, \forall_f \forall_m \quad (3)$$

$$C_{lf} \geq \sum_p^P X_{plf}, \forall_l \forall_f \quad (4)$$

$$\sum_l^L X_{plf} = \sum_j^J Y_{pfj}, \forall_p \forall_f \quad (5)$$

*Não-negatividade?*

A equação (1) expressa a função objetivo do problema, dividida em duas partes: o custo de produção representado pela variável tridimensional  $p_{plf}$  multiplicada pela variável de decisão contínua  $X_{plf}$  de mesmos índices, e o custo de transporte do produto representado pela variável tridimensional  $t_{pfj}$  multiplicada pela variável de decisão contínua  $Y_{pfj}$  de mesmos índices, resultando assim nos custos totais do problema, estando sujeito às restrições de (2) a (5).

A restrição em (2) garante que toda demanda de um cliente ( $j$ ) seja satisfeita com a ideia de que se o produto foi entregue, então ele foi produzido. A restrição (3) garante para cada fábrica que seja usada somente a quantidade de matéria-prima disponível. A restrição (4) garante que a capacidade de produção não seja excedida, levando em consideração cada máquina ( $l$ ) de cada fábrica ( $f$ ). A restrição em (5) garante que se o produto foi produzido ele vai ser entregue e dessa forma entra em conformidade com a restrição em (2).

## Geração das instâncias

Após definido o modelo matemático, 10 instâncias foram geradas:

- $J = \{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000\}$
- $|F| \in [|J|, 2|J|]$

- $|L| \in [5, 10]$
- $|M| \in [5, 10]$
- $|P| \in [5, 10]$
- $D_{pj} \in [10, 20]$
- $r_{mpl} \in [1, 5]$
- $R_{mf} \in [800, 1000]$
- $C_{lf} \in [80, 100]$
- $P_{plf} \in [10, 100]$
- $t_{pfj} \in [10, 20]$

Tais instâncias foram geradas com um fator randômico e podem ser encontradas neste link: <https://github.com/fernandesitalo/pesquisa-operacional-unicamp/tree/main/instances>, assim como o programa em python utilizado para geração das mesmas. A implementação do modelo foi feita na linguagem de programação *python* na versão 3.9.7, por sua facilidade de uso e praticidade provido pela biblioteca *gurobipy*. O computador utilizado para execução das instâncias tem as seguintes configurações: 1<sup>a</sup> geração de Intel® Core™ i7-1165G7 (4 núcleos, caché de 12MB, até 4.7GHz), 16GB de memória RAM com sistema operacional ubuntu 20.04. As execuções foram limitadas a 30 minutos.

## Resultados e Discussões

A tabela 1 apresenta os resultados obtidos de cada instância. A primeira coluna indica a instância, as próximas cinco colunas indicam a sua configuração e as três últimas colunas representam a quantidade de variáveis de decisão da instância, o custo ótimo encontrado pelo software Gurobi e o tempo gasto para processar a instância.

**Tabela 1 - Configuração e resultados obtidos por instância.**

	J	F	L	M	P	Quantidade de variáveis de decisão $ P  *  L  *  F  +  P  *  F  *  J $	Custo Ótimo	Tempo (s)
<b>Inst. 1</b>	100	160	9	6	7	122080	2.1036e+05	0.607
<b>Inst. 2</b>	200	280	8	10	10	582400	6.08e+05	3.636
<b>Inst. 3</b>	300	598	8	9	9	1657656	8.08963e+05	20.592
<b>Inst. 4</b>	400	400	9	6	6	981600	7.3654e+05	7.55
<b>Inst. 5</b>	500	510	7	9	8	2068560	1.2497e+06	29.721
<b>Inst. 6</b>	600	791	8	9	19	9137632	1.8288e+06	105.121
<b>Inst. 7</b>	700	946	7	7	6	4012932	1.2897e+06	75.677

<b>Inst. 8</b>	800	828	5	7	8	5332320	2.0463e+06	126.147
<b>Inst. 9</b>	900	1350	9	8	5	6135750	1.3514e+06	202.875
<b>Inst. 10</b>	1000	1050	7	5	5	5286750	1.54763e+06	158.242

J é o número de clientes, F é o número de fábricas, L é o número de máquinas, M é a quantidade de matéria prima disponível e P é a quantidade de tipos de produtos existentes.

Ao analisarmos as instâncias geradas, pode-se notar o aumento "natural" (salvo algumas exceções) no valor do custo e no tempo que o problema foi resolvido, já que um aumento no número de clientes implica em um maior número de produtos fabricados e em um maior custo, consequentemente. Entretanto, em alguns casos, mesmo com o aumento no número de clientes e fábricas, temos um custo menor do que uma instância com valores menores. Pode-se dizer que isso tem relação direta com o número de produtos demandados. Por exemplo: na instância 6 temos um caso com 19 produtos e um custo total de 1.8288e+06. Já na instância 7, mesmo possuindo um maior número de clientes, por somente 6 produtos serem demandados, temos um custo menor. Também podemos notar o mesmo fato entre as instâncias 8, 9 e 10. Por ter menos produtos demandados, o custo da instância 9 e 10 é menor que o da 8. Além disso, podemos traçar um paralelo também entre o custo e a quantidade de máquinas. Por possuir mais máquinas, a instância 9 pode produzir a mesma quantidade de produtos que a instância 10, com um custo menor. *Excelente discussão*

Junto aos apontamentos apresentados, levamos em conta o fator randômico na geração das instâncias, que influencia diretamente na complexidade de suas soluções, isto é, na quantidade de variáveis de decisão. Outro ponto que pode ser observado, é que o tempo limite considerado foi de 30 minutos, porém o Gurobi apresentou um alto desempenho, gastando menos de 4 minutos no pior caso (instância 9) considerando ser um problema complexo pela quantidade de variáveis de decisão criadas em cada instância.

O código desenvolvido pode ser encontrado no link: <https://github.com/fernandesitalo/pesquisa-operacional-unicamp> assim como o *input* e *output* das instâncias.

## Referências

- Gurobi Optimizer Reference Manual Disponível em: <<https://www.gurobi.com/documentation/9.5/refman/index.html>>. Acesso em: 27 de março de 2022.
- Notas de aula e material disponibilizado pelo professor.

# Atividade 1

Grupo 8

Lucas Costa De Oliveira RA: 182410, Lucas De Oliveira Silva RA: 220715

01 de Abril de 2022

MO824A/MC859A – Tópicos em Otimização Combinatória

*Por completude, descrever o problema que está sendo resolvido*

## Modelo Matemático

Como objetivo da Atividade 1 temos como finalidade minimizar os custos de produção e transporte a cada cliente  $j$ , de cada produto  $p$ , feito por cada máquina  $l$  de cada fábrica  $f$  usando a matéria-prima  $m$ . Como parâmetros inteiros temos:

$D_{j,p}$  = demanda do cliente  $j$ , em toneladas, do produto  $p$

$r_{m,p,l}$  = quantidade de matéria-prima  $m$ , em toneladas, necessária para produzir uma tonelada do produto  $p$  na máquina  $l$

$R_{m,f}$  = quantidade de matéria-prima  $m$ , em toneladas, disponível na fábrica  $f$

$C_{l,f}$  = capacidade disponível de produção, em toneladas, da máquina  $l$  na fábrica  $f$

$p_{p,l,f}$  = custo de produção por tonelada do produto  $p$  utilizando a máquina  $l$  na fábrica  $f$

$t_{p,f,j}$  = custo de transporte por tonelada do produto  $p$  partindo da fábrica  $f$  até o cliente  $j$

Para resolver esse problema decidimos criar 2 classes de variáveis reais na modelagem:

$X_{p,l,f}$  = quantidade de toneladas do produto  $p$ , feita pela máquina  $m$ , na fábrica  $f$

$Y_{p,f,j}$  = quantidade de toneladas do produto  $p$ , transportado pela fábrica  $f$ , para o cliente  $j$

Como objetivo temos que minimizar a função objetiva  $\text{custo total}$  dada por:

$$F(X, Y) = \sum_p \sum_l \sum_f X_{p,l,f} p_{p,l,f} + \sum_p \sum_j \sum_f Y_{p,f,j} t_{p,f,j}$$

Para resolver esse problema de PL estabelecemos as seguintes restrições às nossas variáveis:

$$(1) \sum_f Y_{p,f,j} = D_{j,p} \quad \forall p, j$$

$$(2) \sum_p \sum_l X_{p,l,f} r_{m,p,l} \leq R_{m,f} \quad \forall m, f$$

$$(3) \sum_p X_{p,l,f} \leq C_{l,f} \quad \forall l, f$$

$$(4) \sum_l X_{p,l,f} = \sum_j Y_{p,f,j} \quad \forall p, f$$

$$(5) X_{p,l,f} \geq 0, \quad Y_{p,f,j} \geq 0 \quad \forall p, f, j, l$$

Que podem ser traduzidas como:

- (1) A soma da quantidade total transportada pelas fábricas  $f$  do produto  $p$  para o cliente  $j$  deve ser igual a demanda do produto  $p$  do cliente  $j$  para todo produto  $p$  e cliente  $j$

- (2) A soma da quantidade produzida dos produtos p por cada máquina l utilizando a matéria prima m na fábrica f não pode ser maior que a quantidade de matéria prima m disponível na fábrica f para toda matéria prima m e toda fábrica f.
- (3) A soma da quantidade dos produtos p produzidos pela máquina l da fábrica f deve ser menor ou igual a capacidade de produção da máquina l da fábrica f para toda máquina l e fábrica f.
- (4) A soma da quantidade produzida pelas máquinas l do produto p da fábrica f deve ser igual a soma da quantidade transportada aos cliente j do produto p da fábrica f para todo produto p e fábrica f.
- (5) Os valores de toneladas produzidas e transportadas não podem ser negativos

## Resultados

### Experimentos

Todos os experimentos foram realizados em um computador com as seguintes configurações: Intel Core i5-7300HQ com 2.50GHz e 8GB de memória RAM

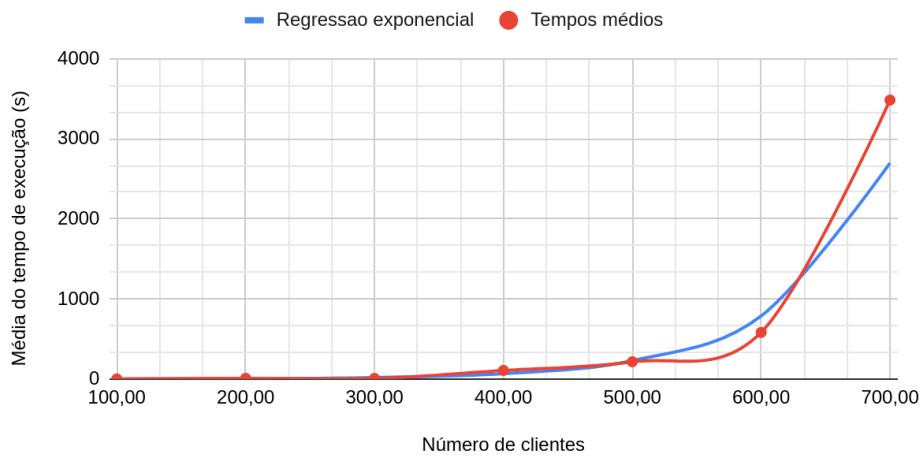
As instâncias com até 500 clientes foram executadas com 4 threads, e as demais execuções usaram apenas uma para limitar o uso de memória RAM. As três últimas instâncias geradas ( $J = 800, 900$  e  $1000$ ) possuem muitos parâmetros (acima de 10 milhões de variáveis) e portanto não foi possível executá-las, mesmo com a restrição no número de threads devido à falta de memória.

Cada instância foi gerada com o auxílio de um script python e cada teste foi repetido 4 vezes e a média dos tempos de execução foi calculado.

→ não precisava pois  
o algoritmo é determinístico.

Número de clientes	Número de variáveis do modelo	Quantidade de restrições	Custo da solução ótima	Média do tempo de execução (s)
100	190.050	4.620	311.037,400	1,65
200	744.150	9.260	621.677,040	9,44
300	762.195	9.954	665.836,039	8,44
400	2.944.350	18.540	1.231.227,750	107,83
500	4.176.350	21.540	1.544.624,111	216,08
600	3.023.790	19.908	1.318.032,045	582,62
700	9.545.700	34.080	2.152.147,455	3.483,51
800	11.704.700	37.080	-	-
900	14.063.700	40.080	-	-
1000	16.622.700	43.080	-	-

## Tempo médio de execução para uma instância aleatória com o número de clientes fixo



## Análise

Apesar de ser um problema facilmente enunciável e conceitualmente simples, a formulação em PL deu origem a modelos grandes demais com relativamente poucos clientes (800 pra cima). Isso pode ser justificado pela escolha da parametrização pelo número de clientes, já que o número de variáveis e restrições é  $O(J^2)$  (pois  $F \in [J, 2J]$ ). *Faltou considerar 'P' e 'L'*

O gasto excessivo de memória que impossibilitou a execução de testes grandes pode ser concluído através da análise dos tempos de execução, que tomaram aproximadamente a forma de uma exponencial com relação a  $J$ . *na verdade o custo é polinomial*

Do teste 200 para o 300 houve uma pequena queda de tempo, mesmo com o aumento do número de variáveis. Pela natureza aleatória dos testes esse tipo de comportamento é esperado, pois uma determinada instância (mesmo que maior) pode produzir um politopo mais complexo dependendo da natureza das restrições. *Muito bom!*

O custo de transporte e produção são limitados a constantes, mas mesmo com um número quadrático de variáveis não houve um aumento correspondente quadrático do custo ótimo das instâncias. O crescimento foi aproximadamente de ordem linear com relação a  $J$ . Isso vem do fato de que a demanda total de cada novo cliente, e portanto o custo total, é da ordem de  $O(1)$  já que os custos máximos de produção/transporte são constantes assim como a demanda de cada cliente por cada produto. *Ótimo!*

Número de clientes	100	200	300	400	500	600	700	800	900	1000
1ª execução	1,64	9,52	8,45	107,50	218,44	588,09	3488,92	-	-	-
2ª execução	1,64	9,44	8,39	108,12	213,73	578,76	3490,99	-	-	-
3ª execução	1,65	9,36	8,48	107,70	215,56	583,74	3460,25	-	-	-
4ª execução	1,65	9,43	8,45	108,01	216,60	579,90	3493,87	-	-	-
Média	1,65	9,44	8,44	107,83	216,08	582,62	3483,51	-	-	-

Tempo de execução (s) de cada teste de cada instância

Por completude, é bom descrever o problema que está sendo resolvida.

## Atividade 1

Lucas Guesser Targino da Silva - RA: 203534

Lucas Pedroso Cantanhede - RA: 202058

Luiz Fernando Bueno Rosa - RA: 221197

April 1, 2022

# 1 Modelo Matemático

## 1.1 Variáveis do Problema

- $F$ : Número de fábricas das companhias.
- $J$ : Número de clientes das companhias.
- $L$ : Número de máquinas das fábricas.
- $M$ : Número de matérias-primas das fábricas.
- $P$ : Número de tipos de produtos produzidos pelas fábricas.
- $D_{j,p}$ : Demanda do cliente  $j$ , em toneladas, do produto  $p$ .
- $r_{m,p,l}$ : Quantidade de matéria-prima  $m$ , em toneladas, necessária para produzir uma tonelada do produto  $p$  na máquina  $l$ .
- $R_{m,f}$ : Quantidade de matéria-prima  $m$ , em toneladas, disponível na fábrica  $f$ .
- $C_{l,f}$ : Capacidade disponível de produção, em toneladas, da máquina  $l$  na fábrica  $f$ .
- $p_{p,l,f}$ : Custo de produção, por tonelada, do produto  $p$  utilizando a máquina  $l$  na fábrica  $f$ .
- $t_{p,f,j}$ : Custo de transporte, por tonelada, do produto  $p$  partindo da fábrica  $f$  até o cliente  $j$ .

## 1.2 Variáveis de Decisão

- $x_{p,l,f}$ : quantidade, em toneladas, do produto  $p$  produzida na máquina  $l$  da fábrica  $f$ .
- $y_{p,f,j}$ : quantidade, em toneladas, do produto  $p$  transportada da fábrica  $f$  para o cliente  $j$ .

### 1.3 Objetivo

Minimizar:

$$\sum_p \sum_l \sum_f p_{p,l,f} x_{p,l,f} + \sum_p \sum_f \sum_j t_{p,f,j} y_{p,f,j} \quad (1)$$

Sujeito a:

$$D_{j,p} = \sum_f y_{p,f,j} \quad \forall p \forall j \quad (2)$$

$$R_{m,f} \geq \sum_p \sum_l r_{m,p,l} x_{p,l,f} \quad \forall m \forall f \quad (3)$$

$$C_{l,f} \geq \sum_p x_{p,l,f} \quad \forall l \forall f \quad (4)$$

$$\sum_l x_{p,l,f} = \sum_j y_{p,f,j} \quad \forall p \forall f \quad (5)$$

$$x_{p,l,f}, y_{p,f,j} \geq 0 \quad \forall p \forall l \forall f \forall j \quad (6)$$

A notação  $\forall i$  acima (podendo  $i$  ser  $p, l, f, m, j$ ) significa que a restrição se aplica a todos os valores do domínio discreto de  $i$ . Por exemplo, se o domínio de  $f$  for  $\{1, 2, 3\}$ , então a condição se aplica para  $f = 1$ ,  $f = 2$  e  $f = 3$ . Esse modelo ainda não usa [1].

Iremos explicar, brevemente, o que cada restrição significa:

1. A função objetivo 1 é calculada como sendo a soma do custo de produção (somatório à esquerda) e o custo de transporte (somatório à direita). O custo de produção é calculado como uma soma variando para cada produto  $p$ , máquina  $l$  e fábrica  $f$  da quantidade produzida, em toneladas, vezes o custo por tonelada  $p_{p,l,f}$ . O custo de transporte é calculado como uma soma variando para cada produto  $p$ , fábrica  $f$  e cliente  $j$  da quantidade transportada, em toneladas, vezes o custo de transporte  $t_{p,f,j}$ . Tendo em vista que queremos o menor custo possível, temos que o objetivo é minimizar o máximo possível o resultado dessa função.
2. A restrição 2 é a restrição da demanda onde restringimos que a demanda solicitada ( $D_{j,p}$ ) por qualquer cliente  $j$  de um produto  $p$ , deve ser igual a somatória da quantidade de  $p$  de produtos produzidos e transportados para esse cliente  $j$  em todas as fábricas.
3. A restrição 3 é a restrição de produção de um produto  $p$  na fábrica  $f$  onde  $r_{m,p,l}$  é a quantidade em toneladas do material  $m$  necessário para produzir uma tonelada de  $p$  na máquina  $l$  e  $R_{m,f}$  é a quantidade de  $m$  na fábrica  $f$ .
4. A restrição 4 é a restrição da capacidade produtiva. Restringimos que a capacidade disponível, em toneladas, de produção de qualquer máquina  $l$  em uma fábrica  $f$  deve igual ou superior a somatória da quantidade, em toneladas, produzida de produtos  $p$  por essa mesma máquina  $l$  na fábrica  $f$ .
5. A restrição 5 é a restrição de compatibilidade entre o que foi produzido e que foi transportado. Nela consideramos que existe uma igualdade da somatória da quantidade, em toneladas, produzida em todas as fábricas de qualquer produto  $p$  em determinada máquina  $m$  com a somatória da quantidade, em toneladas, transportada para todos os clientes desse mesmo produto  $p$  produzido pela máquina  $m$ .
6. A restrição 6 é a restrição de não-negatividade, considerando que nem  $x_{p,l,f}$  nem  $y_{p,f,j}$  podem ser negativos independentes dos valores de  $p, l, f$  e  $j$ .

*Excelente descrição  
do modelo*

## 1.4 Geração das instâncias

Conforme solicitado no enunciado, geramos 10 instâncias aleatórias variando a quantidade de clientes  $|J| = 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000$  e os demais parâmetros do enunciado, apresentados em 1.1, de maneira aleatória e uniforme seguindo as restrições dos coeficientes abaixo:

- $|F| \in [|J|, 2|J|]$
- $|L| \in [5, 10]$
- $|M| \in [5, 10]$
- $|P| \in [5, 10]$
- $D_{j,p} \in [10, 20]$
- $r_{m,p,l} \in [1, 5]$
- $R_{m,f} \in [800, 1000]$
- $C_{l,f} \in [80, 100]$
- $p_{p,l,f} \in [10, 100]$
- $t_{p,f,j} \in [10, 20]$

O módulo de geração de números aleatórios empregado foi `numpy.random` da biblioteca `numpy` [2]. Utilizou-se a semente de números aleatórios 17558175 para gerar, de forma que tenha-se as mesmas instâncias de problemas em diferentes execuções do código.

## 1.5 Configuração da Máquina

O problema foi executado num ideapad S145 81S90005BR: Lenovo IdeaPad S145 Notebook Intel Core i5-8265U (6MB Cache, 1.6GHz), 8GB DDR4-SDRAM, 460 GB SSD, Intel UHD Graphics 620.

O sistema operacional foi o Fedora 35 executando o Python 3.7.12 e Gurobi Optimizer v9.5.1rc2.

## 1.6 Resultados

Instância(J)	Qtde. de variáveis	Qtde. de restrições	Custo da Solução	Tempo de execução(s)
100	100320	3422	253714.11620693645	3.263497
200	668817	9698	567490.8153045126	18.304874
300	1441176	18141	744867.1786728669	60.564301
400	1909440	16070	994991.7809754083	97.239349
500	1771000	13700	790455.3006384401	50.536251
600	3981420	20454	1929256.3805292659	222.136828

Devido a limitações de recursos computacionais (memória), não foi possível rodar instâncias geradas com  $J \geq 700$ . Porém, os dados coletados já são suficientes para o propósito de aprendizado do trabalho.

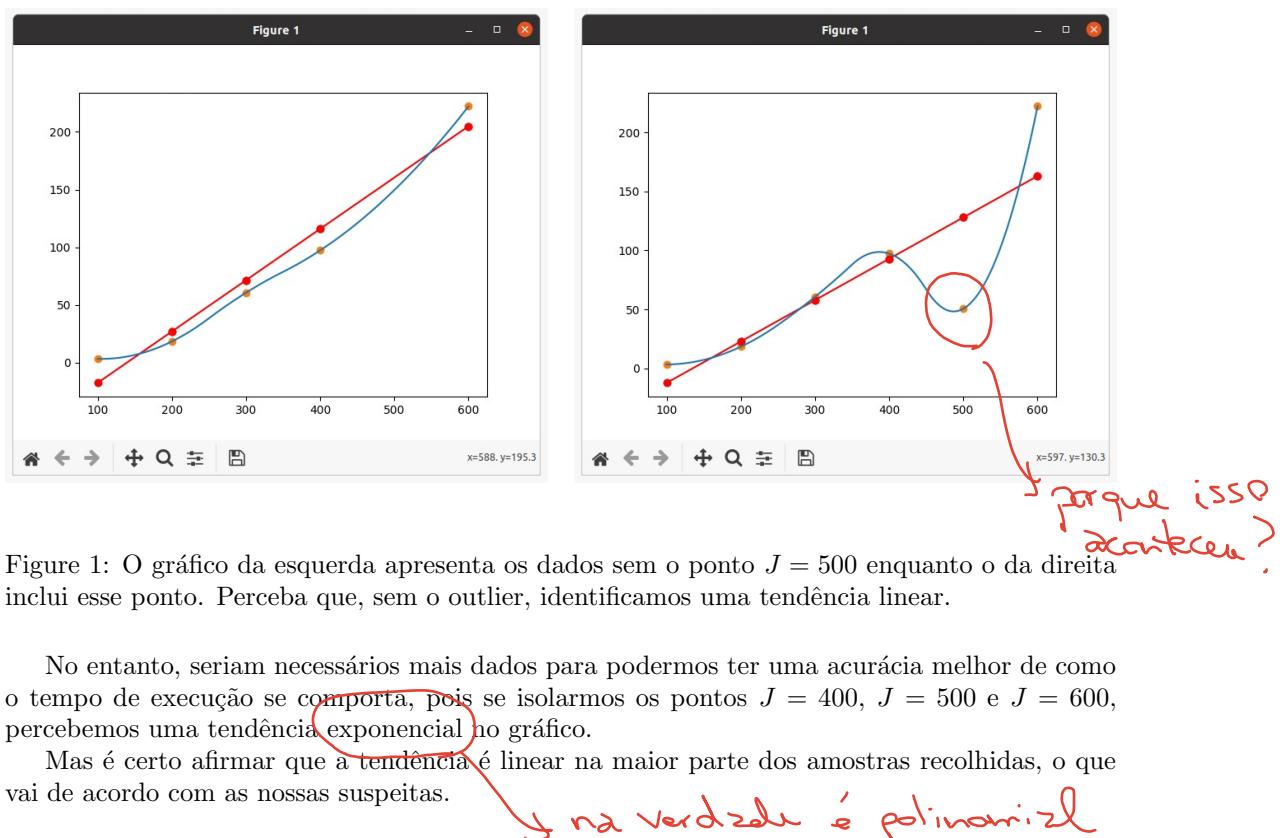
## 1.7 Análise

### 1.7.1 Tempo de execução

A partir dos dados da tabela anterior, utilizamos uma regressão linear para estudar o crescimento do tempo de execução. A suspeita inicial é que encontrariamos um crescimento polinomial, isto é, que cresce em  $O(n^k)$ .

Mas, primeiramente, observe na tabela 1.6 de resultados acima a instância  $J = 500$ . É o primeiro caso que possa apontar um comportamento não-linear da nossa função, então um experimento interessante seria plotar os nossos tempos de execução ora levando em conta  $J = 500$  ora não e tentar estimar visualmente o crescimento da nossa função.

Os gráficos a seguir são uma plotagem do tempo de execução em relação ao número de clientes  $|J|$  juntamente com uma estimativa linear desses dados, gerada a partir do algoritmo de regressão linear.



### 1.7.2 Minimização de custos

Podemos realizar as mesmas análises em relação à minimização da função objetivo. Hipotetizamos que, como a função objetivo é linear, então a sua minimização deve ser proporcional ao aumento da demanda (número dos clientes).

Conseguimos averiguar pelos dois gráficos anteriores que o crescimento da minimização da função objetivo possui a mesma tendência que o crescimento do tempo de execução, o que confirma nossa hipótese.

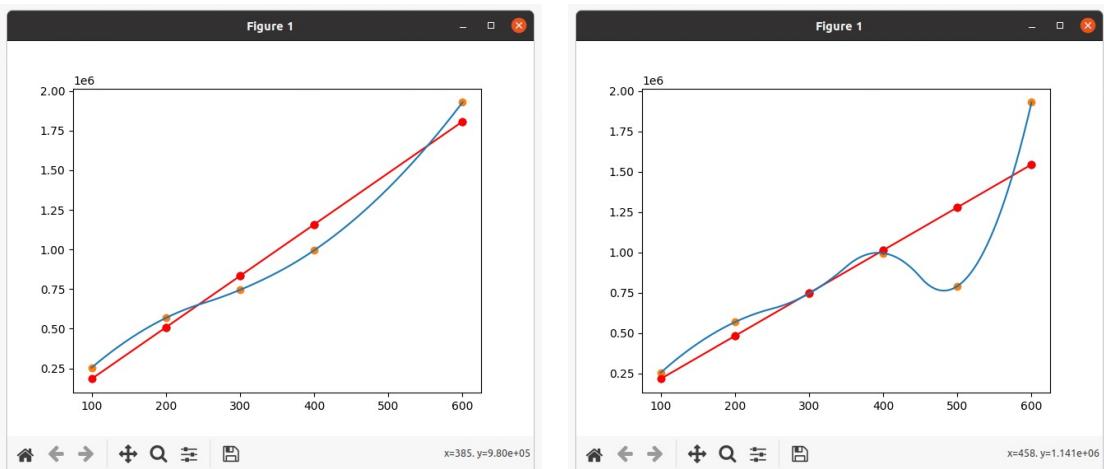


Figure 2: O gráfico da esquerda apresenta os dados sem o ponto  $J = 500$  enquanto o da direita inclui esse ponto. Perceba que, sem o outlier, identificamos, novamente, uma tendência linear.

## 2 Referências

### ~~References~~

- [1] Gurobi Optimization. Gurobi optimization. <https://www.gurobi.com/>. [acessado em 2022-03-30].
- [2] NumPy Developers. Numpy documentation. <https://numpy.org/doc/stable/index.html>. [acessado em 2022-04-01].

# MO824 - Tópicos em Otimização Combinatória

Problema de produção

Equipe 10

Luiz Gustavo S. Aguiar, Matheus Cesar Nunes

RAs: 240499, 203373

01 de Abril de 2022

## Formulação

### Variáveis

- $Q_{p,l,f}$  = Quantidade produzida do produto  $p$  utilizando a máquina  $l$  na fábrica  $f$ .
- $W_{p,f,j}$  = Quantidade transportada do produto  $p$  partindo da fábrica  $f$  para o cliente  $j$

### Função objetivo

$$MIN z = \sum_{p \in P} \sum_{f \in F} \sum_{l \in L} Q_{p,l,f} \cdot p_{p,l,f} + \sum_{p \in P} \sum_{f \in F} \sum_{j \in J} W_{p,f,j} \cdot t_{p,f,j}$$

### Restrições

$$D_{j,p} = \sum_{f \in F} W_{p,f,j} \quad \forall j \forall p \quad (1)$$

$$R_{m,f} \geq \sum_{p \in P} \sum_{l \in L} Q_{p,l,f} \cdot r_{m,p,l} \quad \forall m \forall f \quad (2)$$

$$C_{l,f} \geq \sum_{p \in P} Q_{p,l,f} \quad \forall l \forall f \quad (3)$$

$$\sum_{l \in f} Q_{p,l,f} = \sum_{j \in J} W_{p,f,j} \quad \forall p \forall f \quad (4)$$

não-negatividade?

### Resultados

Descrição do modelo?

Segue abaixo duas tabelas que apresentam os dados inseridos e obtidos das 10 instâncias propostas.

Forma de geração das instâncias?

ambiente computacional?

Table 1: Parâmetros gerados

Instâncias	Clientes	Fábricas	Máquinas	Matérias-primas	Produtos
1	100	102	8	6	8
2	200	351	8	8	6
3	300	317	5	5	9
4	400	665	6	6	6
5	500	623	5	9	8
6	600	1048	9	8	6
7	700	1226	6	6	6
8	800	924	9	8	9
9	900	1469	5	8	5
10	1000	1333	7	6	7

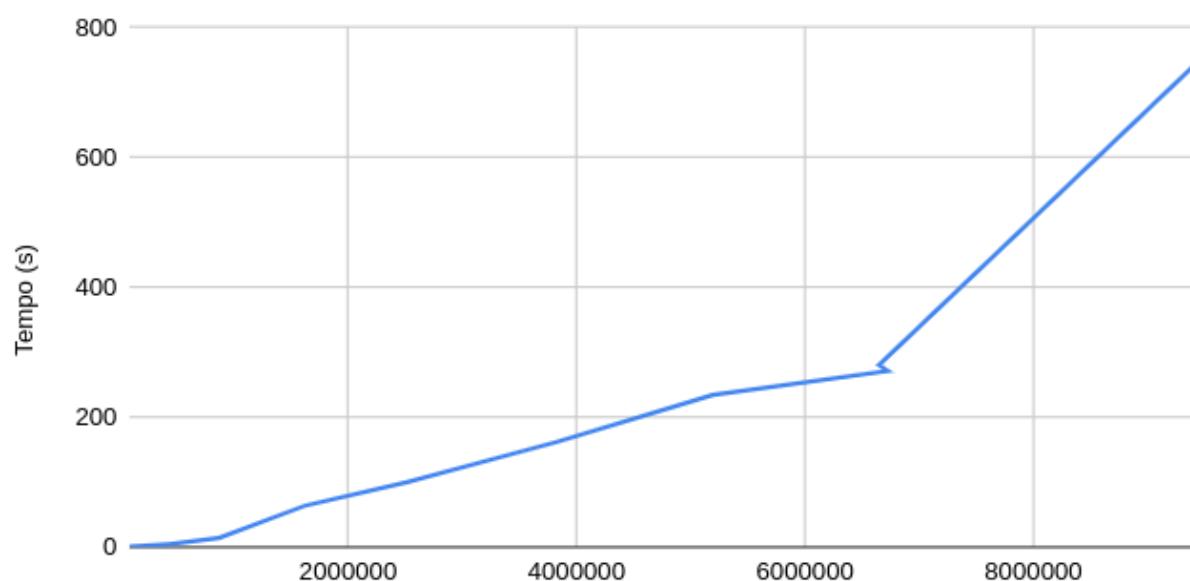
Table 2: Resultados

Instâncias	Variáveis	Restrições	Otimizador	Custo	Tempo (s)
1	88128	3044	-	6716 simplex iterations	0.65
2	438048	8922	-	13846 simplex iterations	4.34
3	870165	8723	-	66411 simplex iterations	14.14
4	1619940	14370	primal and dual simplex	1352601 simplex iterations	63.56
5	2516920	17706	primal and dual simplex	693986 simplex iterations	100.14
6	3829392	27704	dual simplex and barrier	485110 simplex iterations	162.08
7	5193336	26268	dual simplex and barrier	740400 simplex iterations	234.36
8	6727644	31224	dual simplex and barrier	1018803 simplex iterations	270.97
9	6647225	30942	dual simplex and barrier	3175866 simplex iterations	280.54
10	9396317	33660	dual simplex and barrier	171463 simplex iterations	739.63

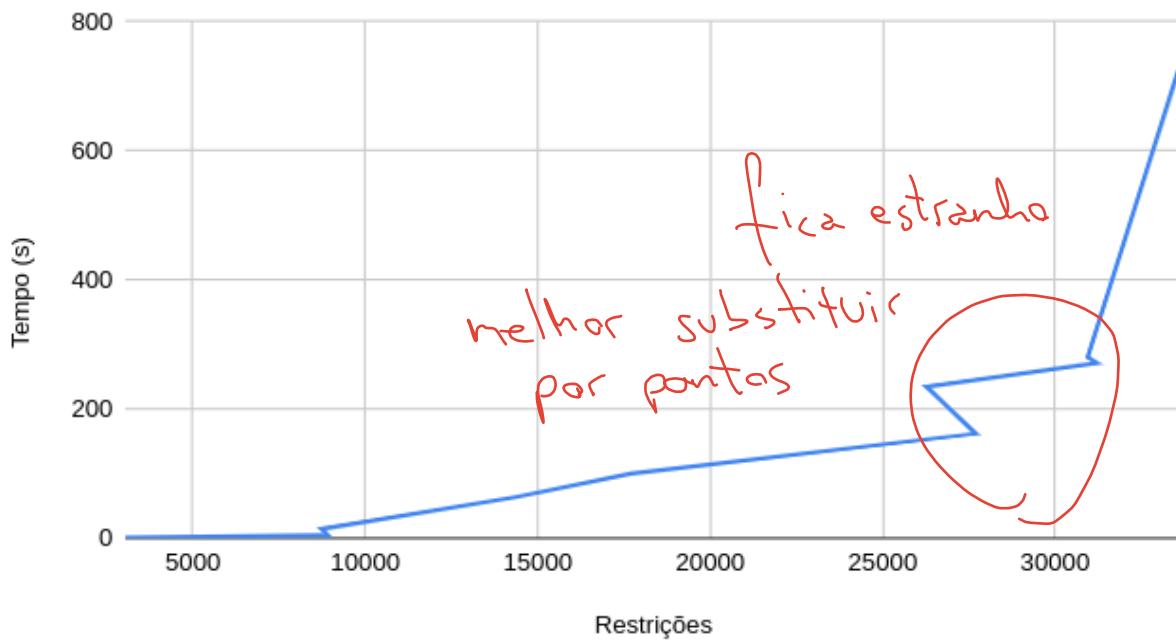
Análises

Custo da Solução?

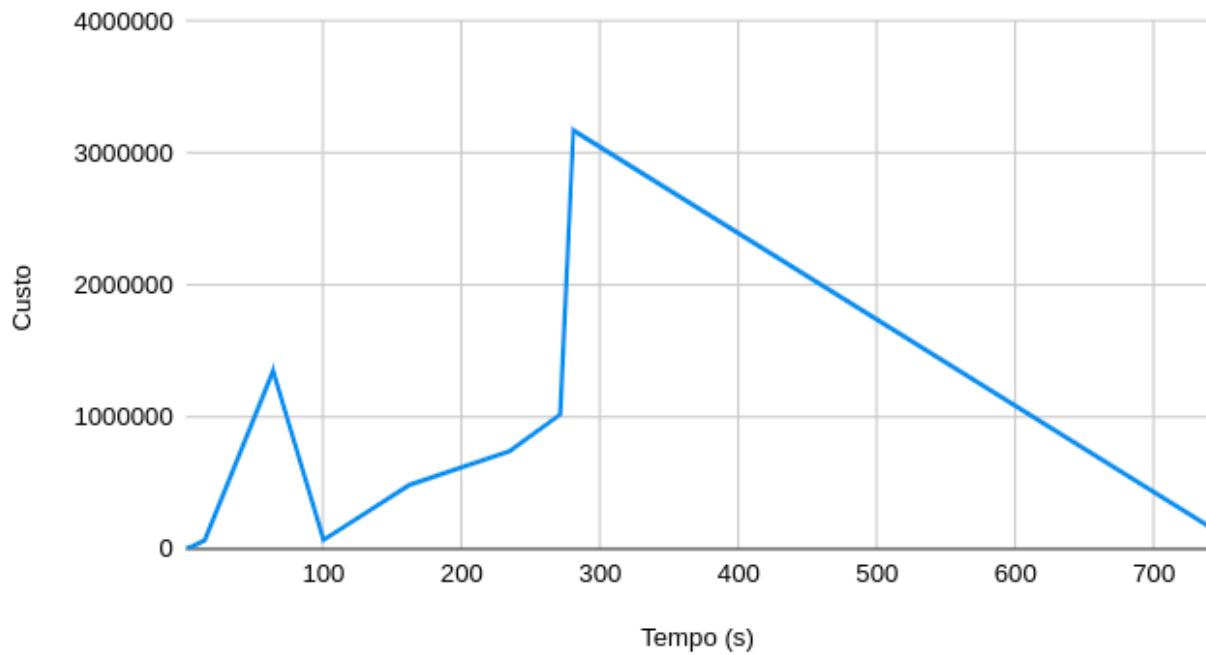
Variáveis x Tempo



Restrições x Tempo



Custo x Tempo



A partir dos valores gerados e dos gráficos podemos notar uma tendência de crescimento exponencial do tempo em função das restrições e variáveis. Note que no gráfico Restrições × Tempo há pontos onde temos menos restrições e um tempo maior, isto indica que mais fatores

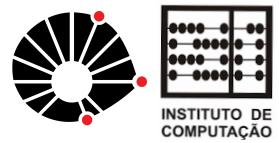
→ na verdade é polinomial

são levados em consideração para explicar o tempo dentre eles está a matriz de coeficientes que pode prejudicar a convergência e portanto aumentar o tempo de execução do algoritmo. Outro ponto interessante que vale observar é o gráfico Custo × Tempo, podemos ver que o tempo não depende muito da quantidade de iterações do simplex visto que da nona para a décima instância há uma diferença da ordem de  $3 \cdot 10^6$  iterações a mais na nona e o tempo de execução é cerca de 2,5 vezes maior na décima.

# MO824 | Atividade 1

Nome Paulo Pacitti RA 185447

Nome Pedro Mota RA 185853



Excelente relatório, parabéns!

## 1. Introdução

O objetivo desta atividade consiste na modelagem em programação linear e a solução desse modelo utilizando o *software* Gurobi do seguinte problema:

Uma companhia possui  $F$  fábricas para atender a demanda de  $J$  clientes. Cada fábrica pode escolher dentre  $L$  máquinas e  $M$  tipos de matéria-prima para produzir  $P$  tipos de produtos. A companhia precisa desenvolver um plano de produção e transporte com o objetivo de minimizar os custos totais. Mais especificamente, a companhia deve determinar a quantidade de cada tipo de produto a ser produzida em cada máquina de cada fábrica e a quantidade que deve ser transportada de cada produto partindo de cada fábrica para cada consumidor. Os parâmetros do problema encontram-se abaixo:

- $D_{j,p}$  = demanda do cliente  $j$ , em toneladas, do produto  $p$ ;
- $r_{m,p,l}$  = quantidade de matéria-prima  $m$ , em toneladas, necessária para produzir uma tonelada do produto  $p$  na máquina  $l$ ;
- $R_{m,f}$  = quantidade de matéria-prima  $m$ , em toneladas, disponível na fábrica  $f$ ;
- $C_{l,f}$  = capacidade disponível de produção, em toneladas, da máquina  $l$  na fábrica  $f$ ;
- $p_{p,l,f}$  = custo de produção por tonelada do produto  $p$  utilizando a máquina  $l$  na fábrica  $f$ ;
- $t_{p,l,j}$  = custo de transporte por tonelada do produto  $p$  partindo da fábrica  $f$  até o cliente  $j$ ;

## 2. Modelo

Para modelar o problema apresentado, usou-se duas variáveis de decisão:

$$X_{plf} : \text{toneladas do produto } p \text{ produzidas pela máquina } l \text{ na fábrica } f$$
$$Y_{pfj} : \text{toneladas do produto } p \text{ transportadas da fábrica } f \text{ para o cliente } j$$

A grandeza a ser minimizada neste problema, é o custo total da companhia. O custo total é a soma do custo da produção e custo de transporte dos produtos. O custo da produção pode ser calculado multiplicando as toneladas produzidas ( $X_{plf}$ ) pelo custo de produção de cada tonelada ( $p_{p,l,f}$ ). Da mesma forma, o custo de transporte pode ser calculado multiplicando as toneladas transportadas ( $Y_{pfj}$ ) pelo custo de transporte de cada tonelada ( $t_{p,l,j}$ ). A Equação 1 expressa de forma matemática o custo total da companhia.

$$\sum_{p} \sum_{l} \sum_{f} X_{plf} \cdot p_{p,l,f} + \sum_{p} \sum_{l} \sum_{f} Y_{pfj} \cdot t_{p,l,j} \quad (1)$$

As Equações 2 a 5 a seguir descrevem respectivamente as restrições de demanda dos clientes, material disponível para produção, capacidade de produção e igualdade entre toneladas produzidas e transportadas.

$$\sum_f Y_{pfj} = D_{j,p} \quad \forall_{p,j} \quad (2)$$

$$\sum_p \sum_l \sum_f X_{plf} \cdot r_{m,p,l} \leq R_{m,f} \quad \forall_{m,f} \quad (3)$$

$$\sum_p X_{plf} \leq C_{l,f} \quad \forall_{l,f} \quad (4)$$

$$\sum_l X_{plf} = \sum_j Y_{pfj} \quad \forall_{p,f} \quad (5)$$

*não-negatividade?*

### 3. Resultados

Implementamos o modelo construído para o problema com a biblioteca `gurobipy`, implementação do *software Gurobi* para a linguagem de programação Python (Código anexo ao fim do relatório). A máquina utilizada para execução foi um desktop Windows 10 com processador Intel Core i5-9400F (6 cores e 6 threads) e 16GB DDR4 de memória RAM. O código implementado executa o modelo 10 vezes com  $J = \{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000\}$  e os demais parâmetros de forma aleatória seguindo os seguintes intervalos inteiros:

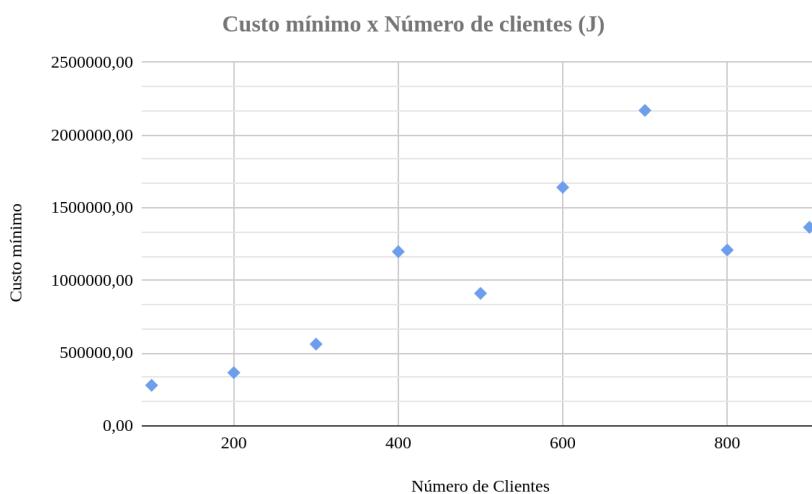
- $F \in [|J|, 2|J|]$
- $L \in [5, 10]$
- $M \in [5, 10]$
- $P \in [5, 10]$
- $D_{j,p} \in [10, 20]$
- $r_{m,p,l} \in [1, 5]$
- $R_{m,f} \in [800, 1000]$
- $C_{l,f} \in [80, 100]$
- $p_{p,l,f} \in [10, 100]$
- $t_{p,l,j} \in [10, 20]$

Executamos nosso *script* 3 vezes: a primeira obtivemos o custo ótimo até  $J = 1000$ , porém a segunda e última vez o processo foi interrompido em  $J = 700$  e  $J = 900$  respectivamente, devido a problemas de falta de memória. Apesar da primeira execução ter completado seu objetivo, somente nas duas últimas incluímos o *log* do número de variáveis e restrições utilizadas por cada iteração. A **Tabela 1** apresenta os dados dessa última execução (*log* de execução ao final do relatório).

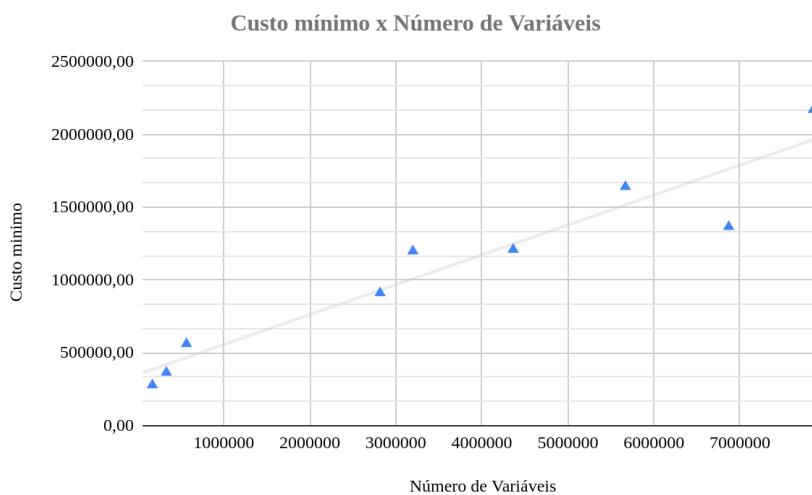
<b>J</b>	<b>Restrições</b>	<b>Variáveis</b>	<b>Tempo (s)</b>	<b>Custo ótimo (mínimo)</b>
100	4772	169488	0,94	278806,20
200	6006	331614	1,6	366206,60
300	8268	565488	4,96	561865,22
400	23550	3198380	50,84	1198865,00
500	20594	2816892	42,18	911287,25
600	31300	5668992	120,09	1640404,58
700	33736	7853700	191,08	2170361,80
800	24501	4364555	81,37	1209754,62
900	30255	6870525	143,18	1365627,367
1000*	-	-	-	-

**Tabela 1:** tabela com os dados de execução do modelo para cada valor de J iterado.

\*Execução não chegou ao fim devido à falta de memória



**Gráfico 1:** relação entre o custo mínimo e o número de clientes.



**Gráfico 2:** relação entre o custo mínimo e o número de variáveis.

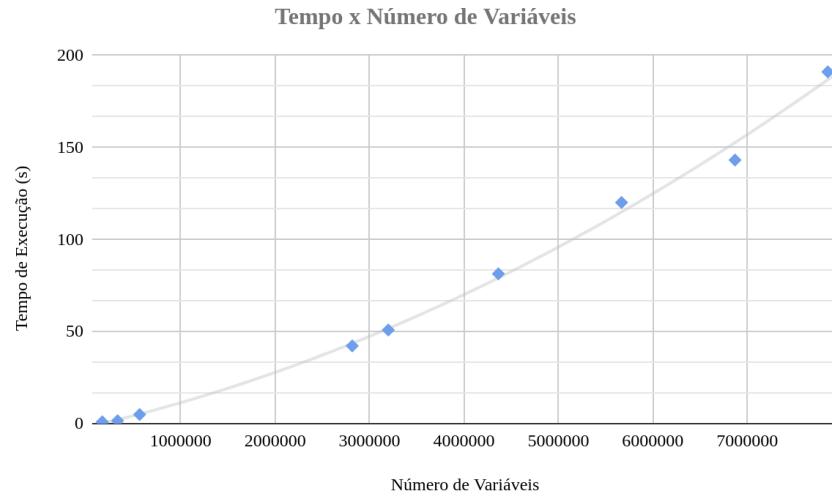


Gráfico 3: relação entre o tempo de execução e o número de variáveis.

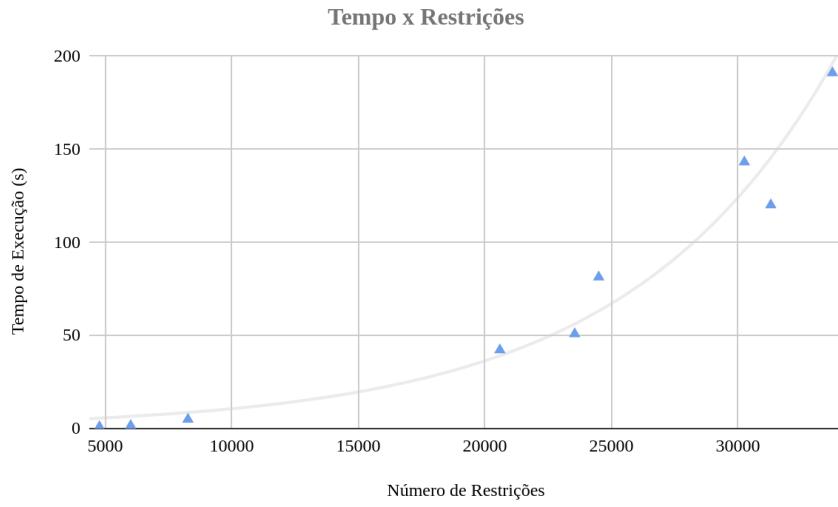


Gráfico 4: relação entre o tempo de execução e o número de restrições.

### 3. Análise

Realizando uma pesquisa sobre o uso do Gurobi em diversas linguagens, o processo de otimização tem a mesma performance em memória e processamento em todas as linguagens, pois a biblioteca é apenas uma camada por cima de uma implementação da linguagem C. Porém, a construção do modelo (variáveis e restrições), depende das estruturas de dados implementadas na linguagem de uso, podendo usar mais ou menos memória de acordo com a implementação utilizada. Outros fatores ~~que~~ em que a linguagem de uso afeta a performance é se esta usa *garbage collector* ou não, afetando na eficiência do uso da memória.

O Gráfico 1 mostra que é difícil determinar o crescimento do custo mínimo usando apenas o número de clientes J. A partir de  $J = 400$  o custo mínimo não mostra um padrão claro de crescimento para as próximas instâncias. Porém, analisando o Gráfico 2 vê-se que o custo mínimo cresce linearmente com o número de variáveis do modelo. Ambos os comportamentos são esperados. Analisando a função objetivo da Equação 1 vemos que o custo mínimo depende linearmente das variáveis  $X_{plf}$  e  $Y_{pfj}$ , o que explica o comportamento do Gráfico 2. Porém, as variáveis  $X_{plf}$  e  $Y_{pfj}$  não dependem exclusivamente de J. Seu tamanho depende também de L, F e P que são gerados aleatoriamente para cada instância, o que pode explicar o comportamento do Gráfico 1.

Quanto ao tempo de execução para cada instância vê-se claramente que o tempo de execução cresce junto com o tamanho do modelo. O Gráfico 3 mostra que o tempo para resolver uma instância cresceu polinomialmente com o número de variáveis, apesar de ter uma tendência linear. E o Gráfico 4 mostra que o tempo de execução cresce exponencialmente com o número de restrições.

*Excelente análise!*

#### 4. Conclusão

Desta análise podemos concluir que o custo mínimo depende linearmente das variáveis do modelo. Além disso, vemos que o tempo para resolver uma instância do modelo depende polinomialmente da quantidade de variáveis e exponencialmente da quantidade de restrições.

*↳ polinomialmente também*

Pode deixar os logs em um arquivo separado

## Anexo I: código fonte do script

```
import gurobipy as gp
from gurobipy import GRB
import random

def production_minimization(J):

    F = random.randint(J, 2*J)

    L = random.randint(5, 10)
    M = random.randint(5, 10)
    P = random.randint(5, 10)

    Djp = [[random.randint(10, 20) for p in range(P)] for j in range(J)]
    rmpl = [[[random.randint(1, 5) for l in range(L)] for p in range(P)] for m in range(M)]
    Rmf = [[random.randint(800, 1000) for f in range(F)] for m in range(M)]
    Clf = [[[random.randint(80, 100) for f in range(F)] for l in range(L)] for m in range(M)]
    pplf = [[[random.randint(10, 100) for f in range(F)] for l in range(L)] for p in range(P)]
    tpfj = [[[random.randint(10, 20) for j in range(J)] for f in range(F)] for p in range(P)]

    model = gp.Model("atividade1")

    # Xplf[p][l][f]: toneladas do produto p produzido pela maquina l na fabrica f
    Xplf = model.addVars(P, L, F, lb=0, vtype=GRB.CONTINUOUS, name="Xplf")

    # Ypfj[p][f][j]: toneladas do produto p transportadas da fabrica f para o cliente j
    Ypfj = model.addVars(P, F, J, lb=0, vtype=GRB.CONTINUOUS, name="Ypfj")
    model.update()

    # Restricao de demanda: o transporte deve ser igual a demanda
    for p in range(P):
        for j in range(J):
            model.addConstr(
                (gp.quicksum(Ypfj[p, f, j] for f in range(F)) == Djp[j][p])
            )

    # Restricao de material: material disponivel deve ser maior ou igual ao material utilizado
    for m in range(M):
        for f in range(F):
            model.addConstr(
                (gp.quicksum(Xplf[p, l, f]*rmpl[m][p][l] for l in range(L) for p in range(P)) <= Rmf[m][f])
            )

    # Restricao de capacidade: quantidade produzida deve ser menor ou igual a capacidade de producao
    for l in range(L):
        for f in range(F):
            model.addConstr(
                (gp.quicksum(Xplf[p, l, f] for p in range(P)) <= Clf[l][f]))
    
    # Restricao de producao e tranporte: a quantidade de produto transportado deve ser igual a quantidade de produtos produzidos
    # Non zero
    for p in range(P):
        for f in range(F):
            model.addConstr(
                (gp.quicksum(Xplf[p, l, f] for l in range(L)) ==
                 gp.quicksum(Ypfj[p, f, j] for j in range(J))))
    
    objective_func = gp.quicksum(gp.quicksum(gp.quicksum(Xplf.sum(p, l, f) * pplf[p][l][f] for p in range(P)) for l in range(L)) for f in range(F)) +
                    gp.quicksum(gp.quicksum(gp.quicksum(Ypfj.sum(p, f, j) * tpfj[p][f][j] for p
```

```

        in range(P)) for f in range(F)) for j in range(J))

#objective_func = Xplf.prod(pplf) + Ypfj.prod(tpfj)
model.setObjective(objective_func, GRB.MINIMIZE)
model.update()
model.optimize()
model.write('atividade.lp')

if model.Status == GRB.OPTIMAL:
    print("\nJ: " + str(J))
    print("{M: " + str(M) + ", F: " + str(F) +
          ", L: " + str(L) + ", P: " + str(P) + "}")
    model.printStats()
    print("Production cost + Transportation cost = " +
          str(model.getObjective().getValue()))

return

def main():
    for j in range(100, 1001, 100):
        production_minimization(j)

main()

```

## Anexo II: output da execução do script

```
Set parameter Username
Academic license - for non-commercial use only - expires 2023-03-18
Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)
Thread count: 6 physical cores, 6 logical processors, using up to 6 threads
Optimize a model with 4772 rows, 169488 columns and 405504 nonzeros
Model fingerprint: 0xc7eb8ab3
Coefficient statistics:
  Matrix range [1e+00, 5e+00]
  Objective range [1e+01, 1e+02]
  Bounds range [0e+00, 0e+00]
  RHS range [1e+01, 1e+03]

Concurrent LP optimizer: primal simplex, dual simplex, and barrier
Showing barrier log only...

Presolve time: 0.35s
Presolved: 4772 rows, 169488 columns, 405504 nonzeros

Ordering time: 0.15s

Barrier statistics:
  AA' NZ : 1.890e+05
  Factor NZ : 1.104e+06 (roughly 80 MB of memory)
  Factor Ops : 6.276e+08 (less than 1 second per iteration)
  Threads : 4

          Objective             Residual
Iter      Primal       Dual     Primal     Dual    Compl   Time
  0  2.75472007e+08  0.00000000e+00  1.62e+04  0.00e+00  4.16e+03   1s
  1  2.97139493e+07 -1.69246266e+06  1.75e+03  4.26e-14  4.57e+02   1s
  2  6.60872533e+05 -4.62790158e+05  1.09e+00  4.26e-14  6.70e+00   1s
  3  4.37009247e+05  2.28005149e+05  9.95e-14  4.26e-14  1.22e+00   1s

Barrier performed 3 iterations in 0.94 seconds (0.39 work units)
Barrier solve interrupted - model solved by another algorithm

Solved with primal simplex
Solved in 35972 iterations and 0.95 seconds (0.49 work units)
Optimal objective 2.788062000e+05

J:100
{M: 6, F: 176, L: 7, P: 9}

Statistics for modelatividade1_copy:
  Linear constraint matrix : 4772 Constrs, 169488 Vars, 405504 NZs
  Matrix coefficient range : [ 1, 5 ]
  Objective coefficient range : [ 10, 100 ]
  Variable bound range : [ 0, 0 ]
  RHS coefficient range : [ 10, 1000 ]
Production cost + Transportation cost = 278806.2
Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)
Thread count: 6 physical cores, 6 logical processors, using up to 6 threads
Optimize a model with 6006 rows, 331614 columns and 719298 nonzeros
Model fingerprint: 0x349a1570
Coefficient statistics:
  Matrix range [1e+00, 5e+00]
  Objective range [1e+01, 1e+02]
  Bounds range [0e+00, 0e+00]
  RHS range [1e+01, 1e+03]

Concurrent LP optimizer: primal simplex, dual simplex, and barrier
Showing barrier log only...

Presolve time: 0.52s
Presolved: 6006 rows, 331614 columns, 719298 nonzeros
```

Ordering time: 0.37s

Barrier statistics:

AA' NZ : 3.516e+05  
Factor NZ : 1.435e+06 (roughly 150 MB of memory)  
Factor Ops : 9.046e+08 (less than 1 second per iteration)  
Threads : 4

Iter	Objective		Residual		Compl	Time
	Primal	Dual	Primal	Dual		
0	4.50621514e+08	0.00000000e+00	2.21e+04	0.00e+00	3.58e+03	1s
1	4.32126986e+07	-2.03245264e+06	2.11e+03	4.26e-14	3.48e+02	1s
2	8.11217314e+05	-4.74432627e+05	3.15e-12	2.84e-14	3.84e+00	1s
3	5.60063988e+05	3.18379029e+05	1.81e-13	3.55e-14	7.22e-01	1s
4	5.04522210e+05	3.41907495e+05	1.75e-13	2.84e-14	4.86e-01	2s

Barrier performed 4 iterations in 1.59 seconds (0.92 work units)  
Barrier solve interrupted - model solved by another algorithm

Solved with primal simplex

Solved in 60741 iterations and 1.60 seconds (0.87 work units)  
Optimal objective 3.662066000e+05

J:200

{M: 5, F: 267, L: 7, P: 6}

Statistics for modelatividade1\_copy:

Linear constraint matrix : 6006 Constrs, 331614 Vars, 719298 NZs  
Matrix coefficient range : [ 1, 5 ]  
Objective coefficient range : [ 10, 100 ]  
Variable bound range : [ 0, 0 ]  
RHS coefficient range : [ 10, 1000 ]

Production cost + Transportation cost = 366206.60000000003

Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)

Thread count: 6 physical cores, 6 logical processors, using up to 6 threads

Optimize a model with 8268 rows, 565488 columns and 1230768 nonzeros

Model fingerprint: 0x05570350

Coefficient statistics:

Matrix range [1e+00, 5e+00]  
Objective range [1e+01, 1e+02]  
Bounds range [0e+00, 0e+00]  
RHS range [1e+01, 1e+03]

Concurrent LP optimizer: primal simplex, dual simplex, and barrier  
Showing barrier log only...

Presolve time: 0.91s

Presolved: 8268 rows, 565488 columns, 1230768 nonzeros

Ordering time: 1.32s

Barrier statistics:

AA' NZ : 6.098e+05  
Factor NZ : 2.005e+06 (roughly 250 MB of memory)  
Factor Ops : 1.417e+09 (less than 1 second per iteration)  
Threads : 4

Iter	Objective		Residual		Compl	Time
	Primal	Dual	Primal	Dual		
0	9.78060431e+08	0.00000000e+00	3.37e+04	0.00e+00	4.63e+03	3s
1	1.17597724e+08	-3.78760990e+06	4.03e+03	4.26e-14	5.63e+02	3s
2	1.23169075e+06	-8.22433224e+05	8.84e-12	4.26e-14	3.60e+00	3s
3	8.58373909e+05	3.87271046e+05	1.57e-13	4.26e-14	8.26e-01	3s
4	7.30080572e+05	4.88225006e+05	2.66e-13	2.84e-14	4.24e-01	3s
5	7.07516726e+05	5.18667347e+05	3.87e-13	2.84e-14	3.31e-01	3s
6	6.88467346e+05	5.34000548e+05	3.67e-13	2.84e-14	2.71e-01	4s
7	6.61152906e+05	5.42317259e+05	4.59e-13	2.84e-14	2.08e-01	4s

8	6.23058613e+05	5.46875532e+05	1.46e-12	2.84e-14	1.34e-01	4s
9	6.05105222e+05	5.54301623e+05	1.66e-12	2.84e-14	8.91e-02	4s
10	5.93981008e+05	5.58581013e+05	2.73e-12	2.84e-14	6.21e-02	4s
11	5.79210713e+05	5.60344267e+05	3.11e-12	2.84e-14	3.31e-02	4s
12	5.70547600e+05	5.61381452e+05	3.78e-12	2.84e-14	1.61e-02	4s
13	5.65429193e+05	5.61749022e+05	3.42e-12	2.84e-14	6.46e-03	4s
14	5.63102703e+05	5.61838504e+05	1.91e-11	2.84e-14	2.22e-03	5s
15	5.62061517e+05	5.61862904e+05	5.86e-11	2.84e-14	3.48e-04	5s
16	5.61876767e+05	5.61865131e+05	9.33e-09	4.26e-14	2.04e-05	5s

Barrier performed 16 iterations in 4.94 seconds (3.10 work units)  
 Barrier solve interrupted - model solved by another algorithm

Solved with primal simplex  
 Solved in 143647 iterations and 4.96 seconds (2.53 work units)  
 Optimal objective 5.618652233e+05

J:300  
 {M: 9, F: 308, L: 6, P: 6}

Statistics for modelatividade1\_copy:  
 Linear constraint matrix : 8268 Constrs, 565488 Vars, 1230768 NZs  
 Matrix coefficient range : [ 1, 5 ]  
 Objective coefficient range : [ 10, 100 ]  
 Variable bound range : [ 0, 0 ]  
 RHS coefficient range : [ 10, 1000 ]  
 Production cost + Transportation cost = 561865.2233333334  
 Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)  
 Thread count: 6 physical cores, 6 logical processors, using up to 6 threads  
 Optimize a model with 23550 rows, 3198380 columns and 6819040 nonzeros  
 Model fingerprint: 0x9b8968a2  
 Coefficient statistics:  
 Matrix range [1e+00, 5e+00]  
 Objective range [1e+01, 1e+02]  
 Bounds range [0e+00, 0e+00]  
 RHS range [1e+01, 1e+03]

Concurrent LP optimizer: primal simplex, dual simplex, and barrier  
 Showing barrier log only...

Presolve time: 5.93s  
 Presolved: 23550 rows, 3198380 columns, 6819040 nonzeros

Ordering time: 3.55s

Barrier performed 0 iterations in 50.80 seconds (8.32 work units)  
 Barrier solve interrupted - model solved by another algorithm

Solved with primal simplex  
 Solved in 518868 iterations and 50.84 seconds (14.21 work units)  
 Optimal objective 1.198865000e+06

J:400  
 {M: 6, F: 782, L: 9, P: 10}

Statistics for modelatividade1\_copy:  
 Linear constraint matrix : 23550 Constrs, 3198380 Vars, 6819040 NZs  
 Matrix coefficient range : [ 1, 5 ]  
 Objective coefficient range : [ 10, 100 ]  
 Variable bound range : [ 0, 0 ]  
 RHS coefficient range : [ 10, 1000 ]  
 Production cost + Transportation cost = 1198865.0  
 Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)  
 Thread count: 6 physical cores, 6 logical processors, using up to 6 threads  
 Optimize a model with 20594 rows, 2816892 columns and 5867136 nonzeros  
 Model fingerprint: 0x857d0f40  
 Coefficient statistics:

```
Matrix range      [1e+00, 5e+00]
Objective range   [1e+01, 1e+02]
Bounds range     [0e+00, 0e+00]
RHS range        [1e+01, 1e+03]
```

Concurrent LP optimizer: primal simplex, dual simplex, and barrier  
Showing barrier log only...

Presolve time: 5.40s  
Presolved: 20594 rows, 2816892 columns, 5867136 nonzeros

Ordering time: 3.33s

Barrier performed 0 iterations in 42.14 seconds (9.35 work units)  
Barrier solve interrupted - model solved by another algorithm

Solved with primal simplex  
Solved in 389449 iterations and 42.18 seconds (9.20 work units)  
Optimal objective 9.112872559e+05

J:500  
{M: 6, F: 926, L: 7, P: 6}

Statistics for modelatividade1\_copy:  
Linear constraint matrix : 20594 Constrs, 2816892 Vars, 5867136 NZs  
Matrix coefficient range : [ 1, 5 ]  
Objective coefficient range : [ 10, 100 ]  
Variable bound range : [ 0, 0 ]  
RHS coefficient range : [ 10, 1000 ]  
Production cost + Transportation cost = 911287.2558608059  
Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)  
Thread count: 6 physical cores, 6 logical processors, using up to 6 threads  
Optimize a model with 31300 rows, 5668992 columns and 11934720 nonzeros  
Model fingerprint: 0xec8c0b22  
Coefficient statistics:  
Matrix range [1e+00, 5e+00]  
Objective range [1e+01, 1e+02]  
Bounds range [0e+00, 0e+00]  
RHS range [1e+01, 1e+03]

Concurrent LP optimizer: primal simplex, dual simplex, and barrier  
Showing barrier log only...

Presolve removed 0 rows and 0 columns (presolve time = 10s) ...  
Presolve removed 0 rows and 0 columns (presolve time = 11s) ...  
Presolve time: 15.80s  
Presolved: 31300 rows, 5668992 columns, 11934720 nonzeros

Ordering time: 7.41s

Barrier performed 0 iterations in 120.00 seconds (18.52 work units)  
Barrier solve interrupted - model solved by another algorithm

Solved with primal simplex  
Solved in 1068041 iterations and 120.09 seconds (37.39 work units)  
Optimal objective 1.640404582e+06

J:600  
{M: 8, F: 1036, L: 8, P: 9}

Statistics for modelatividade1\_copy:  
Linear constraint matrix : 31300 Constrs, 5668992 Vars, 11934720 NZs  
Matrix coefficient range : [ 1, 5 ]  
Objective coefficient range : [ 10, 100 ]  
Variable bound range : [ 0, 0 ]  
RHS coefficient range : [ 10, 1000 ]  
Production cost + Transportation cost = 1640404.5816765402

```

Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)
Thread count: 6 physical cores, 6 logical processors, using up to 6 threads
Optimize a model with 33736 rows, 7853700 columns and 16208700 nonzeros
Model fingerprint: 0x00e4e486
Coefficient statistics:
  Matrix range [1e+00, 5e+00]
  Objective range [1e+01, 1e+02]
  Bounds range [0e+00, 0e+00]
  RHS range [1e+01, 1e+03]

```

```

Concurrent LP optimizer: primal simplex, dual simplex, and barrier
Showing barrier log only...

```

```

Presolve removed 0 rows and 0 columns (presolve time = 6s) ...
Presolve removed 0 rows and 0 columns (presolve time = 15s) ...
Presolve removed 0 rows and 0 columns (presolve time = 15s) ...
Presolve time: 22.07s
Presolved: 33736 rows, 7853700 columns, 16208700 nonzeros

```

```
Ordering time: 10.50s
```

```

Barrier performed 0 iterations in 190.95 seconds (26.90 work units)
Barrier solve interrupted - model solved by another algorithm

```

```

Solved with primal simplex
Solved in 2070353 iterations and 191.08 seconds (88.05 work units)
Optimal objective 2.170361804e+06

```

```
J:700
{M: 9, F: 1114, L: 5, P: 10}
```

```

Statistics for modelatividade1_copy:
  Linear constraint matrix : 33736 Constrs, 7853700 Vars, 16208700 NZs
  Matrix coefficient range : [ 1, 5 ]
  Objective coefficient range : [ 10, 100 ]
  Variable bound range : [ 0, 0 ]
  RHS coefficient range : [ 10, 1000 ]
Production cost + Transportation cost = 2170361.8043837855
Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)
Thread count: 6 physical cores, 6 logical processors, using up to 6 threads
Optimize a model with 24501 rows, 4364555 columns and 8971885 nonzeros
Model fingerprint: 0xd4044cab
Coefficient statistics:
  Matrix range [1e+00, 5e+00]
  Objective range [1e+01, 1e+02]
  Bounds range [0e+00, 0e+00]
  RHS range [1e+01, 1e+03]

```

```

Concurrent LP optimizer: primal simplex, dual simplex, and barrier
Showing barrier log only...

```

```

Presolve removed 0 rows and 0 columns (presolve time = 5s) ...
Presolve time: 9.14s
Presolved: 24501 rows, 4364555 columns, 8971885 nonzeros

```

```
Ordering time: 9.95s
```

```

Barrier performed 0 iterations in 81.32 seconds (18.94 work units)
Barrier solve interrupted - model solved by another algorithm

```

```

Solved with primal simplex
Solved in 557893 iterations and 81.37 seconds (16.45 work units)
Optimal objective 1.209754621e+06

```

```
J:800
{M: 5, F: 1079, L: 9, P: 5}
```

```

Statistics for modelatividade1_copy:
  Linear constraint matrix : 24501 Constrs, 4364555 Vars, 8971885 NZs
  Matrix coefficient range : [ 1, 5 ]
  Objective coefficient range : [ 10, 100 ]
  Variable bound range : [ 0, 0 ]
  RHS coefficient range : [ 10, 1000 ]
Production cost + Transportation cost = 1209754.6210526316
Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)
Thread count: 6 physical cores, 6 logical processors, using up to 6 threads
Optimize a model with 30255 rows, 6870525 columns and 14006175 nonzeros
Model fingerprint: 0xf7b8a01f
Coefficient statistics:
  Matrix range      [1e+00, 5e+00]
  Objective range   [1e+01, 1e+02]
  Bounds range      [0e+00, 0e+00]
  RHS range         [1e+01, 1e+03]

Concurrent LP optimizer: primal simplex, dual simplex, and barrier
Showing barrier log only...

Presolve removed 0 rows and 0 columns (presolve time = 5s) ...
Presolve removed 0 rows and 0 columns (presolve time = 12s) ...
Presolve removed 0 rows and 0 columns (presolve time = 15s) ...
Presolve time: 19.20s
Presolved: 30255 rows, 6870525 columns, 14006175 nonzeros

Ordering time: 13.29s

Barrier performed 0 iterations in 143.06 seconds (32.07 work units)
Barrier solve interrupted - model solved by another algorithm

Solved with primal simplex
Solved in 861201 iterations and 143.18 seconds (25.57 work units)
Optimal objective 1.365627367e+06

J:900
{M: 5, F: 1515, L: 7, P: 5}

Statistics for modelatividade1_copy:
  Linear constraint matrix : 30255 Constrs, 6870525 Vars, 14006175 NZs
  Matrix coefficient range : [ 1, 5 ]
  Objective coefficient range : [ 10, 100 ]
  Variable bound range : [ 0, 0 ]
  RHS coefficient range : [ 10, 1000 ]
Production cost + Transportation cost = 1365627.3670731708
Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)
Thread count: 6 physical cores, 6 logical processors, using up to 6 threads
Optimize a model with 51424 rows, 15255040 columns and 31265280 nonzeros
Model fingerprint: 0x66aa1782
Coefficient statistics:
  Matrix range      [1e+00, 5e+00]
  Objective range   [1e+01, 1e+02]
  Bounds range      [0e+00, 0e+00]
  RHS range         [1e+01, 1e+03]

Concurrent LP optimizer: primal simplex, dual simplex, and barrier
Showing barrier log only...

Presolve removed 0 rows and 0 columns (presolve time = 22s) ...
Presolve removed 0 rows and 0 columns (presolve time = 30s) ...
Presolve removed 0 rows and 0 columns (presolve time = 32s) ...
Presolve removed 0 rows and 0 columns (presolve time = 37s) ...
Presolve removed 0 rows and 0 columns (presolve time = 54s) ...
Presolve removed 0 rows and 0 columns (presolve time = 55s) ...
Presolve time: 79.05s
Presolved: 51424 rows, 15255040 columns, 31265280 nonzeros

Elapsed ordering time = 5s

```

Elapsed ordering time = 10s  
Ordering time: 28.26s

Barrier performed 0 iterations in 250.56 seconds (77.09 work units)  
Optimization exhausted available memory

# Atividade 1

## Modelagem de Problema de Programação Linear

Grupo 13

Rafael Sartori Martins Dos Santos,  
Rebecca Moreira Messias,  
Renan Fernando Franco da Silva

Março de 2022

MC859/MO824

## 1 Objetivo

Utilizaremos o software Gurobi[1] para solucionar um problema[3] comum, a minimização de custos de uma companhia, que tanto fábrica quanto distribui vários tipos de produtos em diversos lugares, considerando limitações como capacidade de produção de diferentes máquinas e quantidade de diferentes matérias-primas disponível em cada fábrica, para diferente número de clientes em diferentes situações de demanda e custo.

## 2 Descrição do problema

A companhia possui um conjunto de fábricas  $F$ , cada uma com diferentes tipos de máquinas  $L$  e de matéria-prima  $M$  capazes de produzir tipos de produtos  $P$ . Ela atende clientes  $J$  e deve idealmente suprir a demanda de cada cliente de forma a minimizar os custos totais.

Cada cliente  $j$  possui uma demanda  $D_{j,p}$  por um tipo de produto  $p$  em tonelada. A companhia conhece o custo de transporte  $t_{p,f,j}$  deste produto saindo da fábrica  $f$  até chegar à  $j$ , bem como o custo de produção  $p_{p,l,f}$  e quantidade  $r_{m,p,l}$  de matéria-prima  $m$  consumida ao utilizar a máquina  $l$  na fábrica  $f$ , todos considerando uma tonelada de produto. Há fatores limitantes, no entanto: para cada fábrica  $f$ , há a capacidade de produção  $C_{l,f}$  da máquina  $l$  e a quantidade  $R_{m,f}$  de matéria-prima  $m$  disponível, ambos em tonelada.

## 3 Desenvolvimento da solução

A fim de obter o custo mínimo em diferentes cenários, consideramos o número de clientes partindo de 100 e crescendo de 100 em 100 até atingir 1000. Tendo fixo  $|J|$ , tomamos valores aleatorizados para  $|F|$  (entre  $|J|$  e  $2|J|$ ),  $|L|$  (entre 5 e 10),  $|M|$  (entre 5 e 10) e  $|P|$  (entre 5 e 10), bem como para  $D_{j,p}$  (entre 10 e 20),  $r_{m,p,l}$  (entre 1 e 5),  $R_{m,f}$  (entre 800 e 1000),  $C_{l,f}$  (entre 80 e 100),  $p_{p,l,f}$  (entre 10 e 100) e, por fim,  $t_{p,f,j}$  (entre 10 e 20). As 10 instâncias geradas se encontram em: <https://drive.google.com/file/d/1Yfg7A5zlIwA6vhP36ANyhZOhlmxbyIcR/view?usp=sharing>

Optamos por reduzir o problema através da obtenção das quantidades de produto produzido em cada máquina para cada fábrica e a quantidade que seria transportada de cada fábrica

ao cliente. Conseguimos juntar, com essas duas variáveis, todas as outras características do problema.

### 3.1 Formulação do problema

Temos como objetivo:

$$\text{minimizar} \quad \sum_{p \in P} \sum_{l \in L} \sum_{f \in F} p_{p,l,f} \cdot x_{p,l,f} + \sum_{f \in F} \sum_{p \in P} \sum_{j \in J} t_{p,f,j} \cdot y_{f,p,j} \quad (1)$$

$$\text{sujeito a} \quad \sum_{f \in F} y_{f,p,j} = D_{j,p} \quad \text{para } j \in J, p \in P, \quad (2)$$

$$\sum_{p \in P} x_{p,l,f} \leq C_{l,f} \quad \text{para } l \in L, f \in F, \quad (3)$$

$$\sum_{p \in P} \sum_{l \in L} r_{m,p,l} \cdot x_{p,l,f} \leq R_{m,f} \quad \text{para } m \in M, f \in F, \quad (4)$$

$$\sum_{l \in L} x_{p,l,f} = \sum_{j \in J} y_{f,p,j} \quad \text{para } f \in F, p \in P, \quad (5)$$

$$x_{p,l,f} \geq 0 \quad \text{para } p \in P, l \in L, f \in F, \quad (6)$$

$$y_{f,p,j} \geq 0 \quad \text{para } f \in F, p \in P, j \in J \quad (7)$$

Onde:

- $x_{p,l,f}$  é a quantidade de produto  $p$  em toneladas produzido pela máquina  $l$  na fábrica  $f$ , e
- $y_{f,p,j}$ , a quantidade de produto  $p$  em toneladas que cada fábrica  $f$  vai enviar do produto  $p$  para o cliente  $j$

são as variáveis que criamos para a solução do problema.

Conseguimos essas equações através da análise do problema. Nomeamos a quantidade de produto  $p$  produzido pela máquina  $l$  na fábrica  $f$  de  $x_{p,l,f}$ , de forma a conseguirmos obter quanto seria consumido de matéria-prima, que seria a multiplicação dessa quantidade produzida por  $r_{m,p,l}$ , cuja soma de todos esses fatores (para todo  $l$ ) deve ser menor ou igual à quantidade de matéria-prima disponível na fábrica,  $R_{m,f}$ . Obtemos assim a ??. Também é claro que a quantidade produzida deve ser positiva ou zero, chegando, então, a ??, e que a soma da quantidade produzida para todo cliente  $j$  em cada máquina  $l$  em uma fábrica  $f$  não pode exceder sua capacidade, então temos a ??.

De forma a suprirmos a demanda de cada cliente, nomeamos a quantidade de produto  $p$  produzido por todas as máquinas da fábrica  $f$  de modo a suprir a demanda do cliente  $j$  de  $y_{f,p,j}$ . Assim, podemos notar que ao somar todos  $y_{f,p,j}$  dado uma fábrica  $f$  e um produto  $p$ , teremos toda a produção para a fábrica, que é equivalente a soma de  $x_{p,l,f}$  para todo  $l$  dado mesmo  $f$  e  $p$ , então obtemos a ?? e, por definição, ?? Da mesma forma que em  $x_{p,l,f}$ , podemos concluir que só é possível enviar quantidades positivas ou nulas (deixar de enviar) de produtos a clientes, e então obtemos a ??.

## 4 Resultados

*Ambiente computacional?*

Com essas condições, executamos a solução do problema através do Gurobi utilizando a linguagem Python[2] para produzir o *script* de execução, obtendo os resultados:

$ J $	Tempo de execução (s)	Número de variáveis	Número de Restrições <sup>1</sup>	Custo total
100	0,27	136080	4300	210903,0
200	2,29	621001	10975	483576,7
300	2,61	736801	9000	548957,8
400	4,83	1320201	14300	848565,5
500	28,91	3432486	20007	1069404,0
600	43,04	4567676	24625	1278210,9
700	63,90	5433777	28592	1705573,0
800	45,89	4081807	21660	1483670,7
900	48,96	4720261	23256	1387113,5
1000	516,30	10412101	37050	1829328,8

Table 1: Resultado e estatísticas da execução

Foi possível obter o resultado para todas as instâncias, sem exceder o limite de tempo, memória ou obtermos uma instância inviável. Também podemos notar que o tempo de execução não tem uma relação bem definida com o número de clientes, mas sim com o número de variáveis do sistema, como podemos notar pela ??, que possui característica próxima de exponencial, porém faltam dados para determinar com melhor precisão a curva que melhor descreve esse comportamento.

<sup>1</sup>Não inclui as restrições de não negatividade

*polinomial*

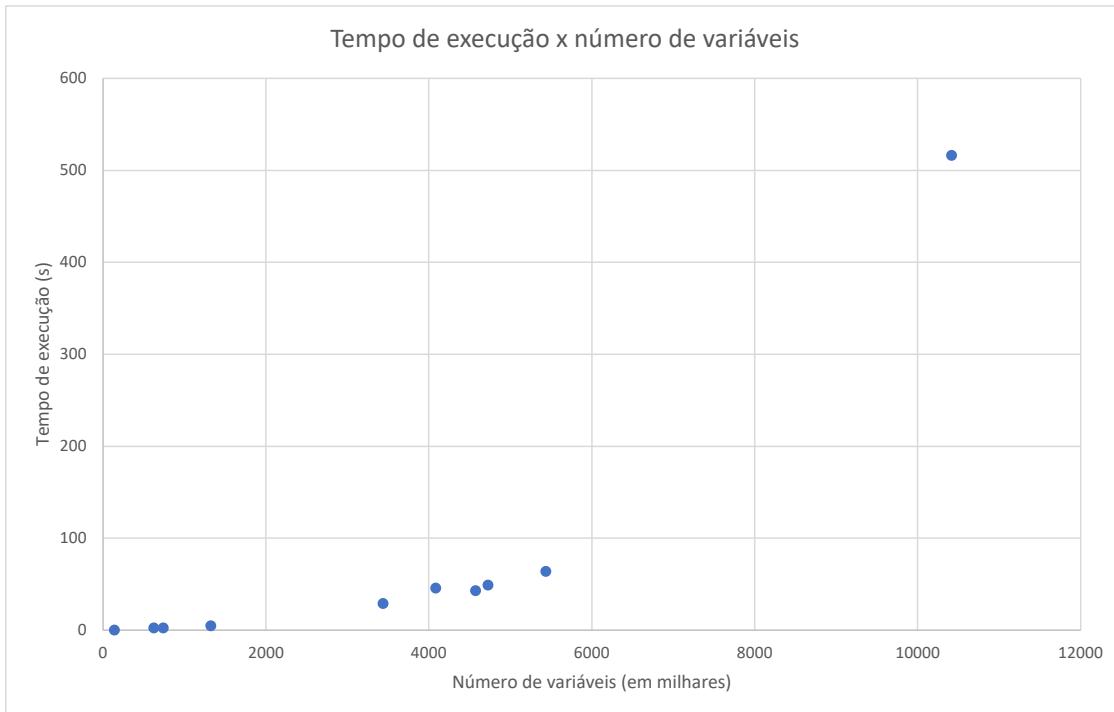


Figure 1: Relação entre tempo de execução e número de variáveis

## 5 Conclusão

Gurobi é um software bastante complexo e muito otimizado, a solução desse problema envolve um número muito grande de variáveis, o que tornaria a solução por métodos não otimizados extremamente demorado, senão impossível. A utilização de Python para a escrever e executar o *script* auxiliou a reduzir tempo de desenvolvimento e de correção de erros, além de facilitar a colaboração.

Vale a pena notar, no entanto, que mesmo utilizando Gurobi, a escolha da linguagem Python tornou o uso de memória volátil pela nossa solução um grande problema. Precisamos de algumas tentativas e algumas modificações (por exemplo, deixar de colocar em cache as instâncias para fazê-las sob demanda) para conseguirmos então obter a solução para as 3 maiores instâncias nos computadores pessoais do grupo.

Pode-se concluir, também, que o tempo de execução é proporcional, mesmo que não linearmente, ao número de variáveis do problema.

## References

- [1] Gurobi documentation. <https://www.gurobi.com/documentation/9.5/>. Accessed: 2022-04-01.
- [2] Python documentation. <https://docs.python.org/3/>. Accessed: 2022-04-01.
- [3] Fábio Luiz Usberti and Celso Cavellucci. Linear programming problems.

# Atividade 1

Sandro Henrique Uliana Catabriga - RA 219597

April 2, 2022

Por comodidade do relatório, é relevante descrever o problema sendo estudado.

## 1 Formulação do Problema

O modelo descrito a seguir contém as seguintes variáveis de decisão:  $x_{plf}$ , que indica a quantidade (em toneladas) do produto  $p$  produzida utilizando a máquina  $l$  da fábrica  $f$ , e  $y_{plf}$ , que indica a quantidade (em toneladas) do produto  $p$  produzida na fábrica  $f$  transportada para o cliente  $j$ . Portanto, nesse modelo separamos as quantidades produzidas das quantidades transportadas.

$$(PL) \quad \text{minimize} \sum_{p \in P} \sum_{l \in L} \sum_{f \in F} (x_{plf} \times p_{plf}) + \sum_{p \in P} \sum_{f \in F} \sum_{j \in J} (y_{pfj} \times t_{pfj}) \\ \text{sujeito a} \quad \sum_{f \in F} y_{pfj} = D_{jp} \quad \forall p \in P, \forall j \in J \quad (1)$$

$$\sum_{p \in P} \sum_{l \in L} (x_{plf} \times r_{mpl}) \leq R_{mf} \quad \forall f \in F, \forall m \in M \quad (2)$$

$$\sum_{p \in P} x_{plf} \leq C_{lf} \quad \forall l \in L, \forall f \in F \quad (3)$$

$$\sum_{j \in J} y_{pfj} = \sum_{l \in L} x_{plf} \quad \forall p \in P, \forall f \in F \quad (4)$$

$$x_{plf} \geq 0 \quad \forall p \in P, \forall l \in L, \forall f \in F \quad (5)$$

$$y_{pfj} \geq 0 \quad \forall p \in P, \forall f \in F, \forall j \in J \quad (6)$$

Na função objetivo queremos minimizar os custos de produção e os custos de transporte. A restrição (1) garante que a quantidade do produto  $p$  transportada até o cliente  $j$ , considerando todas as fábricas, irá atender sua demanda. A restrição (2) garante que

a quantidade de matéria-prima  $m$  utilizada para produzir os produtos na fábrica  $f$ , considerando todas as suas máquinas, não será maior que o limite disponível. A restrição (3) limita a quantidade de produtos produzidos na máquina  $l$  da fábrica  $f$  à sua capacidade  $C_{lf}$ . A restrição (4) faz com que a quantidade produzida do produto  $p$  pela fábrica  $f$ , considerando todas as suas máquinas, seja igual a quantidade do produto  $p$  que ela transportou, considerando todos os clientes. Por fim, as restrições (5) e (6) são relativas aos domínios das variáveis.

## 2 Experimentos e Análise

*Critério de geração de instâncias?*

Para os experimentos foi utilizada uma máquina com chip Apple M1 e 8GB de memória RAM. Foi utilizada a API em C++ do software Gurobi, executando em apenas uma thread (por padrão o Gurobi utiliza mais de uma, se disponível).

A Tabela 1 apresenta um resumo das instâncias geradas. Os parâmetros que mais afetam a quantidade de variáveis do modelo são J e F, que foram crescentes em todas as instâncias com exceção da 9, onde F diminuiu em relação à 8.

Instância	J	F	L	M	P
1	100	110	6	8	10
2	200	285	10	9	6
3	300	517	8	8	8
4	400	545	9	8	6
5	500	681	10	6	5
6	600	753	9	5	8
7	700	974	6	9	9
8	800	1286	7	9	10
9	900	1117	6	8	5
10	1000	1559	6	10	8

Tabela 1: Tabela de instâncias

A Tabela 2 apresenta algumas informações e resultados das execuções. É possível notar que quanto maior a quantidade de variáveis, maior a quantidade de restrições. Em

## *Excelente análise*

um primeiro momento podemos pensar que quanto mais variáveis, mais clientes e consequentemente mais itens a serem produzidos, e isso aumentaria o custo. Os resultados, sem apresentarem qualquer surpresa, confirmam esse cenário esperado, onde o custo aumenta conforme o número de variáveis aumenta.

O tempo de execução também segue esse comportamento esperado. O crescimento da quantidade de variáveis e da quantidade de restrições faz com que o tempo de execução também aumente. Vale a pena destacar que o tempo é muito mais sensível ao aumento de variáveis que o custo. Se o número de variáveis aumenta, o tempo tende a crescer significativamente mais do que o custo.

Instância	Dados		Solução	
	Nº variáveis	Nº restrições	Custo	Tempo (s)
1	116600	3640	319038	0.43
2	359100	8325	360960	1.56
3	1273888	14808	725911	8.46
4	1337430	14935	734454	8.65
5	1736550	16801	759301	64
6	3668616	21366	1461000	200
7	6188796	29676	1958820	645
8	10378020	41436	2440400	2396
9	5060010	25723	1387850	747
10	12546832	45416	2453860	2856

Tabela 2: Tabela de resultados

# Atividade 1

Tiago de Paula Alves (187679)  
 tiagodepalves@gmail.com

Vinicius da Silva (206734)  
 v206734@dac.unicamp.br

1 de abril de 2022

## 1 Modelo

### 1.1 Descrição do Problema

Uma companhia possui  $F$  fábricas para atender a demanda de  $J$  clientes. Cada fábrica pode escolher dentre  $L$  máquinas e  $M$  tipos de matéria-prima para produzir  $P$  tipos de produtos. A companhia precisa desenvolver um plano de produção e transporte com o objetivo de minimizar os custos totais. Mais especificamente, a companhia deve determinar a quantidade de cada tipo de produto a ser produzida em cada máquina de cada fábrica e a quantidade que deve ser transportada de cada produto partindo de cada fábrica para cada consumidor.

### 1.2 Parâmetros

#### 1.2.1 Dimensões

Esses valores são necessariamente inteiros, já que medem contagem de itens.

$J$  quantidade de clientes;

$F$  quantidade de fábricas;

$L$  quantidade de máquinas em cada fábrica;

$M$  quantidade de tipos de matéria-prima;

$P$  quantidade de tipos de produtos;

#### 1.2.2 Parâmetros de Restrição

Valores usados nas restrições do problema e podem ser reais.

$D_{j,p}$  demanda do cliente  $j$ , em toneladas, do produto  $p$ ;

$R_{m,f}$  quantidade de matéria-prima  $m$ , em toneladas, disponível na fábrica  $f$ ;

$C_{l,f}$  capacidade disponível de produção, em toneladas, da máquina  $l$  na fábrica  $f$ ;

#### 1.2.3 Parâmetros de Relação de Variáveis

Relação entre matéria-prima e produto e entre produtos, clientes e custo. Também podem ser reais.

$r_{m,p,l}$  quantidade de matéria-prima  $m$ , em toneladas, necessária para produzir uma tonelada do produto  $p$  na máquina  $l$ ;

$p_{p,l,f}$  custo de produção por tonelada do produto  $p$  utilizando a máquina  $l$  na fábrica  $f$ ;

$t_{p,f,j}$  custo de transporte por tonelada do produto  $p$  partindo da fábrica  $f$  até o cliente  $j$ ;

### 1.3 Variáveis de Decisão

As variáveis desse problema podem assumir quaisquer valores reais, considerando as restrições abaixo.

$x_{p,l,f}$  toneladas produzidas de  $p$  na máquina  $l$  da fábrica  $f$ ;

$y_{p,f,j}$  toneladas transportadas de  $p$  da fábrica  $f$  para o cliente  $j$ ;

### 1.4 Restrições

#### Não-negatividade

$$\begin{aligned} x_{p,l,f} &\geq 0 && \text{para toda produção de } p \text{ na máquina } l \text{ da fábrica } f \\ y_{p,f,j} &\geq 0 && \text{para toda transporte de } p \text{ da fábrica } f \text{ para o cliente } j \end{aligned}$$

#### Atendimento às demandas dos clientes

$$\sum_{f=1}^F y_{p,f,j} = D_{j,p} \quad \text{para todo cliente } j \text{ e produto } p$$

#### Limite de matéria-prima disponível

$$\sum_{p=1}^P \sum_{l=1}^L r_{m,p,l} x_{p,l,f} \leq R_{m,f} \quad \text{para toda matéria-prima } m \text{ e fábrica } f$$

#### Capacidade de produção

$$\sum_{p=1}^P x_{p,l,f} \leq C_{l,f} \quad \text{para toda máquina } l \text{ na fábrica } f$$

#### Equivalência de produção e transporte

$$\sum_{l=1}^L x_{p,l,f} = \sum_{j=1}^J y_{p,f,j} \quad \text{para toda produção de } p \text{ na fábrica } f$$

### 1.5 Função Objetivo

O objetivo final da otimização é minimizar o custo total, garantindo a demanda dos clientes. O custo é separado nos seguintes tipos:

$$\text{custo de produção} = \sum_{f=1}^F \sum_{p=1}^P \sum_{l=1}^L x_{p,l,f} p_{p,l,f}$$

$$\text{custo de transporte} = \sum_{f=1}^F \sum_{p=1}^P \sum_{j=1}^J y_{p,f,j} t_{p,f,j}$$

Assim, o objetivo é minimizar:

$$\text{custo total} = \text{custo de produção} + \text{custo de transporte}$$

$$= \sum_{f=1}^F \sum_{p=1}^P \left( \sum_{l=1}^L x_{p,l,f} p_{p,l,f} + \sum_{j=1}^J y_{p,f,j} t_{p,f,j} \right)$$

## 2 Experimentos

*Forma de geração de instâncias?*

A partir da escolha de  $J$ , a instância é gerada com valores aleatórios seguindo as restrições do enunciado. Tudo foi feito em Python 3, inclusive a implementação do modelo, usando o Gurobi.

### 2.1 Resultados

Os resultados a seguir foram obtidos em um Macbook Pro M1 com 16 GB de memória RAM.

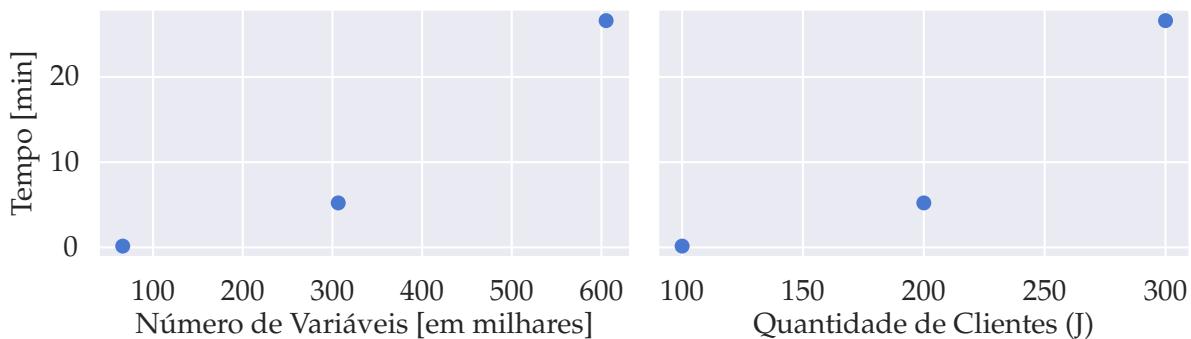
$J$	$F$	$L$	$M$	$P$	Vars.	Restrs.	Iters.	Tempo de Execução	Custo Final
100	126	5	8	5	66 150	2768	12753	8.9 s	160 268.5
200	299	5	8	5	306 475	6382	58689	5 min 13 s	309 782.7
300	393	8	8	5	605 220	9753	93678	26 min 37 s	455 172.4
400	—	—	—	—	—	—	—	—	—
500	—	—	—	—	—	—	—	—	—
600	—	—	—	—	—	—	—	—	—
700	—	—	—	—	—	—	—	—	—
800	—	—	—	—	—	—	—	—	—
900	—	—	—	—	—	—	—	—	—
1000	—	—	—	—	—	—	—	—	—

*Por algum motivo ficou bem lento na máquina que vocês usaram.*

### 3 Análise dos Resultados

*Da próxima vez, podem usar instâncias menores para concluir a análise dos experimentos.*

Figura 1: Tempo de execução em relação a quantidade de clientes e o número de variáveis de otimização.



Infelizmente, não conseguimos executar todos os experimentos. Um dos possíveis problemas é que o número de variáveis no modelo, que é dado por:

$$\text{número de variáveis} = PLF + PFJ = PF(L + J)$$

Considerando os limites da instância no enunciado, temos que

$$\begin{aligned} 5 \cdot J(5 + J) &\leq \text{número de variáveis} \leq 10 \cdot 2J(10 + J) \\ 5J^2 + 25J &\leq \text{número de variáveis} \leq 20J^2 + 200J \end{aligned}$$

Portanto, o número de variáveis cresce de forma quadrática com o parâmetro  $J$ . Considerando que, no melhor dos casos, o tempo de execução depende linearmente do número de variáveis, então o tempo também cresce de forma quadrática.