Simplex e Pontos Interiores

Pedro Loes e Felipe Sadock

20/06/2021

O método de otimização Simplex possui a característica de encontrar a solução exata em um problema de programação linear, porém, em problemas de grande dimensão, o algoritmo apresenta alto custo computacional para atingir a convergência. O método de optimização Pontos Interiores apresenta baixo custo computacional em problemas de grande dimensão, mas apresenta resultados que apenas aproximam a solução ótima. O objetivo deste projeto foi ilustrar o funcionamento desses dois algoritmos e compará-los por meio de simulações para identificar suas vantagens e desvantagens.

0. Introdução

- O projeto foi dividido em 4 etapas que objetivam explicar e explorar os métodos de Otimização Simplex e Pontos Interiorescomparando suas performances.

  1. Método Simplex

  - Desenvolvimento de uma explicação geral, descrição do funcionamento matemático do método, exemplo simples resolvido passo a passo e mecânica da convergência ilustrada graficamente.

  2. Método Pontos Interiores

  - Desenvolvimento de uma explicação geral, descrição do funcionamento matemático do método, exemplo simples resolvido passo a passo e mecânica da convergência ilustrada graficamente.

  3. Simulação Simplex e Pontos Interiores

  - Desenvolvimento de simulação de todas as combinações de **100** variáveis com **100** restrições aleatórias e exibição dos resultados ilustrados em gráficos de número de iterações e diferença entre máximos.

  4. Referências Bibliográficas

  - Indicação de todo o material consultado para o desenvolvimento do projeto.

- A biblioteca Scipy da linguagem Python foi utilizada para implementar os algoritmos Simplex e Pontos Interiores. Os gráficos foram produzidos com as bibliotecas Plotly e Seaborn da linguagem Python. O relatório no formato pdf foi produzido no IDE RStudio com a linguagem RMarkdown.

1. Método Simplex

- A ideia básica do simplex é caminhar pelos vértices do politopo de restrições passando de uma base para outra com a utilização de variáveis de folga para encontrar o máximo da função objetivo.

Descrição

- Cada base corresponde a um valor para a função. Um deles é o valor máximo da função **F**. A próxima base será escolhida de forma que o valor da função **F** não seja menor do que o anterior.
- Uma variável é chamada de variável básica de uma equação se entrar nesta equação com um coeficiente de um e não entrar em outro sistema de equações. Se cada equação tem uma variável básica, pode-se dizer que o sistema tem uma base.

Funcionamento

- Para alternar as bases são usadas tabelas. Cada linha da tabela é equivalente a uma equação do sistema. O método consiste em escolher a coluna com um coeficiente positivo de forma iterativa para obter um valor da função objetivo que não seja inferior aos seus antecessores.
- Para os coeficientes positivos da coluna selecionada, observa-se o coeficiente $\boldsymbol{\theta}$ e a linha com valor mínimo é selecionada. Quando não existirem mais coeficientes positivos na linha da função o valor máximo pode ser calculado.

Exemplo

- Para ilustrar o funcionamento do algoritmo foi escolhido um problema de **2º dimensão** representado pelas variáveis x_1 e x_2 e restringido por **3** desigualdades do tipo $\leq$ representadas pela matriz **A** e vetor **b** de valores das restrições.
- **Função Objetivo:**
  - $f(\ x_1,\ x_2\ )\ =\ 3x_1 + 5x_2$
- **Matriz de Restrições:**
  - $A = \begin{bmatrix} 3x_1 + 2x_2 \\ x_1 \\ x_2 \end{bmatrix}$
- **Vetor Resposta:**
  - $b = \begin{bmatrix} 18 \\ 4 \\ 6 \end{bmatrix}$
- **Domínio**:
  - $x_1\ \geq\ 0$
  - $x_2\ \geq\ 0$
- Os passos da evolução do algoritmo em direção a convergência do máximo foram representados no formato de tabelas para ilustrar a caminhada pelas bases geradas com a adição de **3** variáveis de folga (slack) S_1, S_2 e S_3.

- **Iteração 1**
  - Verifica-se que a variável x_2 apresenta menor θ e calcula $f(\ x_1 = 0\ ,\ x_2 = 0\ ) = 0$.
  - A base inicial será $S_1 = 18$, $S_2 = 4$ e $S_3 = 6$.
  - Transformando S_3 para $S_3 = 0$ as bases passam a ser $x_2 = 6$, $S_1 = 6$ e $S2 = 4$.
  - $f(\ x_1 = 0\ ,\ x_2 = 6) = 30$.

x_1	x_2	S_1	S_2	S_3	Base	Θ
3	2	1	0	0	18	18 : 2 = 9
1	0	0	1	0	4	
0	1	0	0	1	6	6 : 1 = 6
3	5	0	0	0	F - 0	
3	0	1	0	-2	6	
1	0	0	1	0	4	
0	1	0	0	1	6	
3	0	0	0	-5	F - 30	

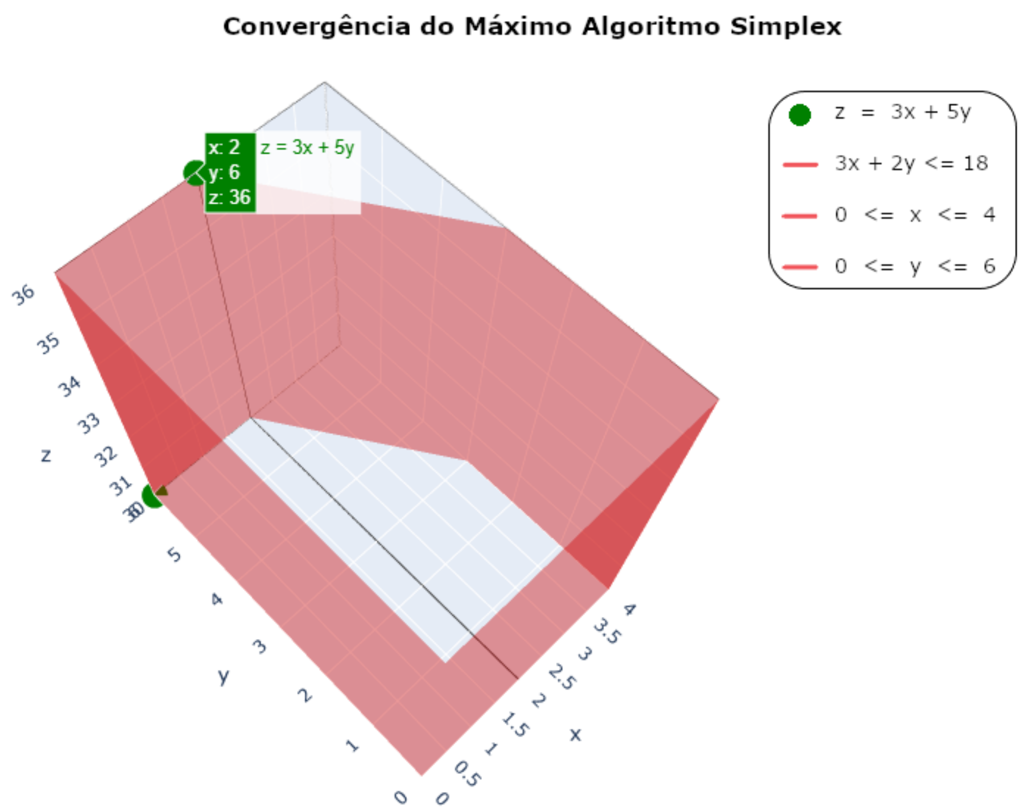
- **Iteração 2**
  - Verifica-se que a variável x_1 apresenta menor θ
  - Transformando a variável S_1 para $S_1 = 0$ as bases passam a ser $x_1 = 2$, $x_2 = 6$ e $S_2 = 2$.
  - $f(\ x_1 = 2\ ,\ x_2 = 6\ ) = 36$.

x_1	x_2	S_1	S_2	S_3	Base	Θ
3	0	1	0	-2	6	6 : 3 = 2
1	0	0	1	0	4	4 : 1 = 4
0	1	0	0	1	6	
3	0	0	0	-5	F - 30	
1	0	1/3	0	-2/3	2	
1	0	0	1	0	4	
0	1	0	0	1	6	
3	0	0	0	-5	F - 30	
1	0	1/3	0	-2/3	2	
0	0	-1/3	1	2/3	2	
0	1	0	0	1	6	
0	0	-1	0	-3	F - 36	

- **Resultado**
  - A primeira e segunda colunas apresentaram valores pivôs na primeira e terceira linhas indicando a posição no vértice **(2,6)** como solução de máximo da função objetivo.

Convergência

- Para ilustrar o caminho do algoritmo Simplex no espaço de busca foi utilizado o pacote Plotly que permitiu a construção de uma figura com 3 dimensões.
- A variável x_1 do problema foi representada pelo eixo **x** do gráfico, a variável x_2 do problema foi representada pelo eixo **y** do gráfico e a imagem da função objetivo foi representada pelo eixo **z** do gráfico.
- O **politopo** de segunda dimensão formado pelas restrições do problema foi representado por **semi-planos** na cor vermelha que se expandem sobre o eixo **z** onde a função objetivo assume valores correspondentes à posição dos vértices do politopo.
- O caminho do algoritmo pelos vértices do politopo para convergência do valor máximo da função objetivo foi representado pelos círculos na cor verde verde.



- O algoritmo inicia no infinito considerando a função objetivo sem restrições.
- 1º Atinge o vértice $\mathbf{x = 0}$ e $\mathbf{y = 6}$ com $\mathbf{f(x,\ y) = 30}$.
- 2º Atinge o vértice $\mathbf{x = 2}$ e $\mathbf{y = 6}$ convergindo para $\mathbf{max\ f(x,\ y) = 36}$.
- O funcionamento do algoritmo animado pode ser visualizado no link:
  - https://colab.research.google.com

2. Método Pontos Interiores

- A ideia principal do algoritmo de Pontos Interiores é caminhar internamente no politopo nas melhores direções sem violar a fronteira de restrições com intuito de convergir rapidamente para um valor próximo do máximo da função objetivo.

Descrição

- Para inicializar o algoritmo é escolhido um vetor X_0 arbitrariamente definido diferente do vetor nulo é uma variável $\gamma \in [0, 1]$ que controla a distância percorrida em cada passo.
- Quando o algoritmo é iniciado é calculada a distância do ponto em relação a cada restrição. A parte mais importante do algoritmo é o uso da álgebra linear para calcular a direção que gera maior crescimento monótono da função objetivo utilizando a técnica de derivada direcional.

Funcionamento

- **Passo 1**:
  - Escolher x^0 diferente do vetor nulo para ser o ponto inicial dentro do politopo e escolher uma variável γ de forma que:
  - $Ax^0 < b \quad e \quad 0 < \gamma < 1$
- **Passo 2**:
  - Calcular a folga, ou seja, a distância do ponto em relação à cada restrição indicando $\mathbf{k = k + 1}$.
  - $v^k \ = \ b - Ax^k \ = \ [v_1^k, v_1^k, ..., v_m^k]^T$
- **Passo 3**:
  - Calcular a matriz diagonal.
  - $D_k \ = \ diag \ [\frac{1}{v_1^k}, \frac{1}{v_2^k}, ..., \frac{1}{v_m^k}]$
- **Passo 4**:
  - Calcular a projeção na fronteira da esfera.
  - $(A^T \ D_K \ D_k \ A) \ d_x^k \ = \ c$
- **Passo 5**:
  - Escalar a direção projetada d_x^k
  - $d_v^k \ = \ -A \ d_x^k \ [\ (dv)_1, \ (dv)_2, \ ..., \ (dv)_m \ ]^T$
- **Passo 6**:
  - Calcular o tamanho da caminhada na direção escolhida.
  - $\alpha \ = \ \gamma \ MAX \ (\ \frac{v_i^k}{(d_v)_i} \ < \ 0 \ )$
- **Passo 7**:
  - Calcular novo ponto interior
  - $x^{k+1} \ = \ x^k - \alpha \, d_x^k$
- **Passo 8**:
  - Verificar parada das interações do algoritmo quando encontrar uma diferença entre a última solução e a anterior inferior à tolerância arbitrariamente definida.
  - $\frac{| \ b^T \ Y^k \ - \ c^T \ x^k \ |}{MAX \ (\ 1 \ | \ c^T \ x^k \ )} < \epsilon$

Exemplo

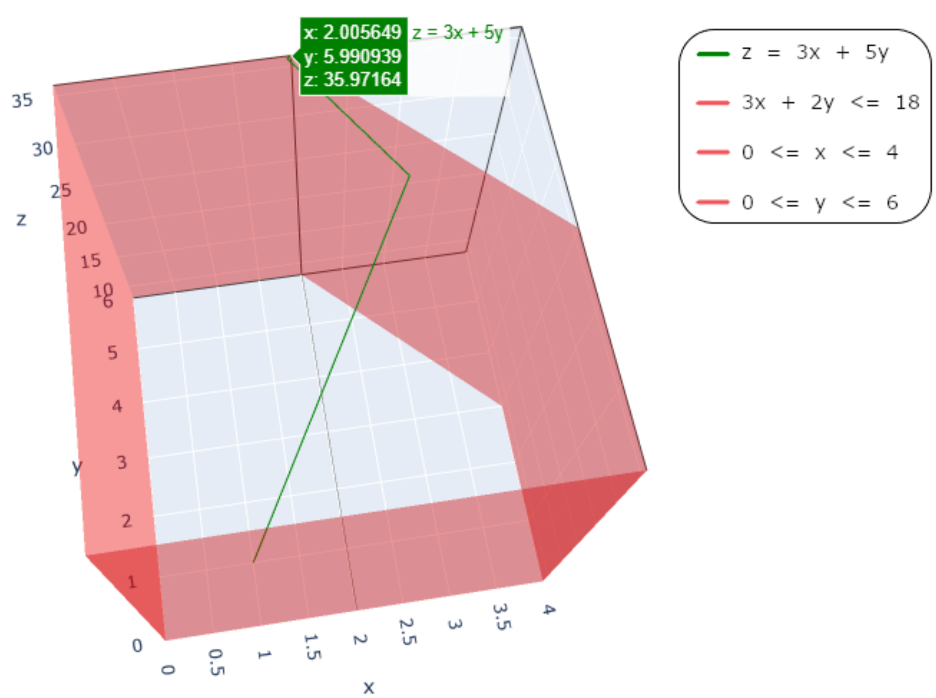
- Foi utilizado o mesmo problema do algoritmo Simplex para comparação das mecânicas e resultados.

It	dv	alpha	(x, y)	F(x, y)
0	-	-	(1, 1)	7
1	(2.4, 4.6)	-0.78	(2.9, 4.6)	31.65
2	(-2, 3.1)	-0.45	(1.97, 5.98)	35.84
3	(2.4, 4.3)	-0.03	(2, 5.99)	35.97

Convergência

- O mesmo gráfico em terceira dimensão foi utilizado para ilustrar a mecânica do algoritmo Pontos Interiores.

Convergência do Máximo Algoritmo Pontos Interiores



- O algoritmo inicia no ponto $x_0 = (1, 1)$ com f(x, y) = 8.
- 1º Caminha para o ponto **x = 2.89** e y = **4.59** com **f(x, y) = 31.65**.
- 2º Caminha para o ponto **x = 1.97** e **y = 5.98** com **f(x, y) = 35.84**.
- 3º Caminha para o ponto **x = 2** e **y = 5.99** convergindo para **max f(x, y) = 35.97**.
- O funcionamento do algoritmo animado pode ser visualizado no link:
  - https://colab.research.google.com

3. Simulação Simplex e Pontos Interiores

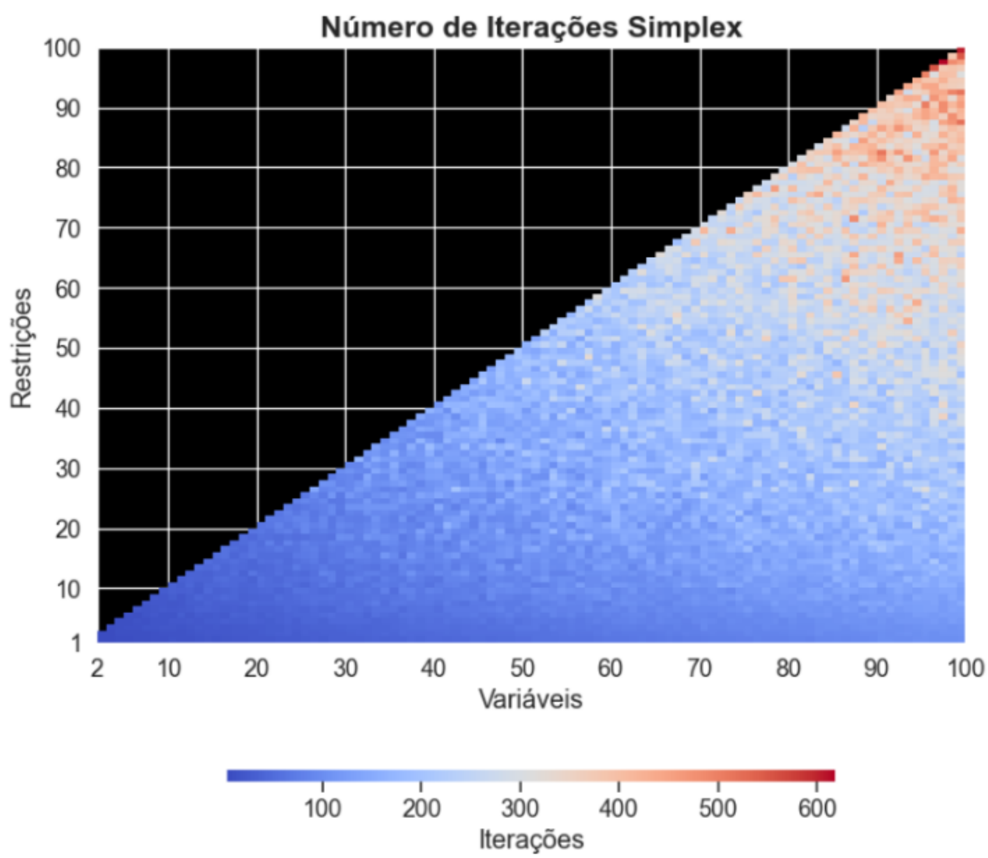
- Para comparar a performance dos dois métodos no que diz respeito ao número de iterações e convergência de problemas com diferentes dimensões, foi desenvolvido uma simulação com **5000** amostras no espaço de dimensões [**2, 100**] variáveis e [**1, 100**] restrições.

Descrição

- As amostras foram aleatoriamente sorteadas nos intervalos [**0, 100**] para as restrições, [**10, 100**] para o lado direito das desigualdades e [**1, 100**] para a função objetivo. Amostras com muito mais restrições do que variáveis apresentaram redundância e ou valor máximo muito próximo de zero, portanto a simulação foi desenhada com um número de restrições menor ou igual ao número de variáveis. A mesma amostra foi utilizada uma vez em cada método.

Mapa de Calor Iterações Simplex

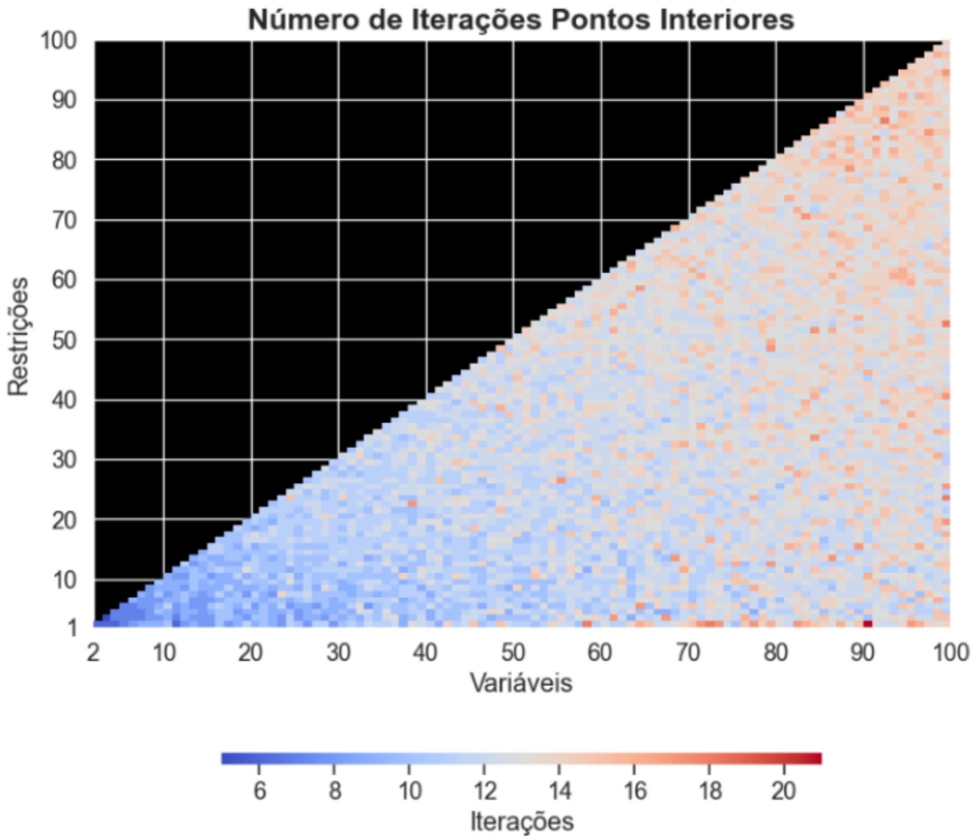
- O eixo **x** foi utilizado para representar o número de variáveis do problema. No eixo **y** foi representado o número de restrições do problema. O **gradiente** das cores vermelho até azul foi utilizado para representar o número de iterações gastas para convergência.



- Dimensões de até **30** variáveis e **30** restrições apresentaram custo inferior a **100** iterações.
- Dimensões de **30** até **60** apresentaram custo entre **100** e **300** iterações.
- Dimensões superiores a **60** apresentaram custo computacional de **300** até **600** iterações.

Mapa de Calor Iterações Pontos Iteriores

- O gráfico do tipo mapa de calor foi produzido para ilustrar os resultados do número de iterações da simulação Pontos Interiores. No eixo **x** foi representado o número de variáveis do problema. No eixo **y** foi representado o número de restrições do problema. O **gradiente** das cores vermelho até azul foi utilizado para representar o número de iterações gastas para convergência.



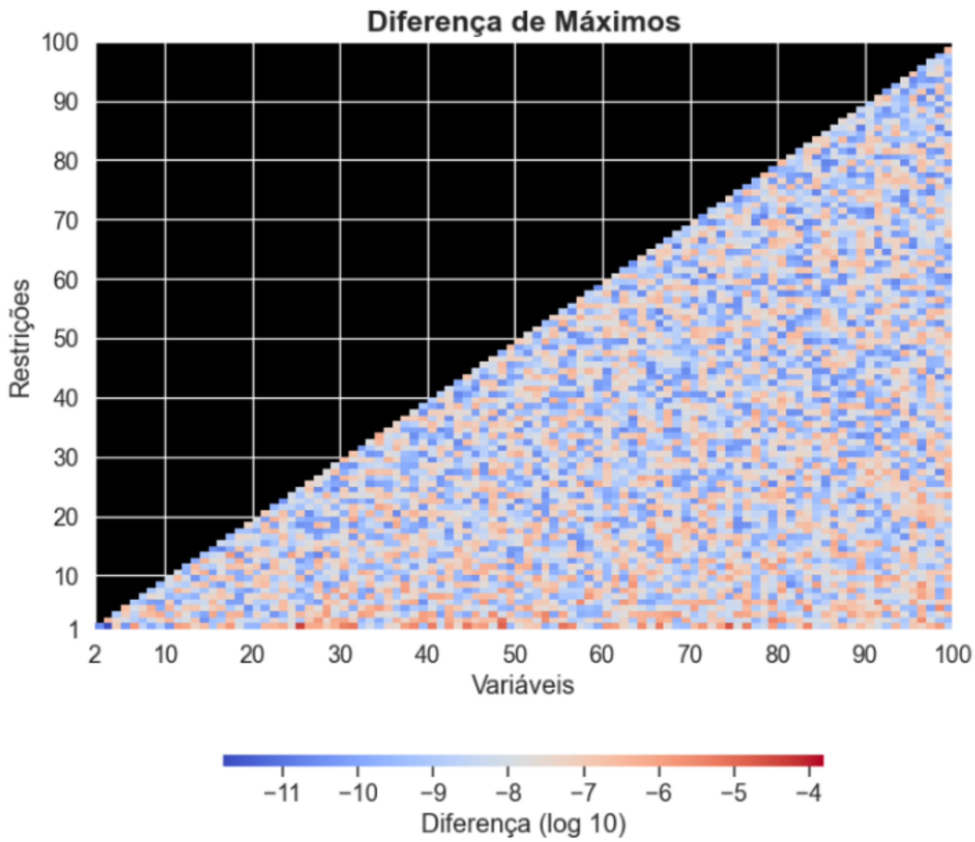
- Dimensões de até **30** variáveis e **30** restrições apresentaram custo inferior a **10** iterações.
- Dimensões de **30** até **60** apresentaram custo entre **10** e **15** iterações.
- Dimensões superiores a **60** apresentaram custo computacional de **15** até **20** iterações.

Comparação de Custo Computacional

- Dimensões de até **30** variáveis e restrições, o Simplex apresentou custo de **5** até **8** vezes maior.
- Dimensões de **30** até **60** variáveis e restrições, o Simplex apresentou custo de **8** até **13** vezes maior.
- Dimensões de **60** até **100** variáveis e restrições, o Simplex apresentou custo de **20** até **25** vezes maior.
- Pontos Interiores apresentou maior mistura de custos, com destaque para correlação positiva da iteração com aumento do número de variáveis. Simplex apresentou regiões mais bem definidas, com poucas observações discrepantes e correlação semelhante com as duas variáveis.

Mapa de Calor Diferença de Máximos

- No eixo **x** foi representado o número de variáveis do problema. No eixo **y** foi representado o número de restrições do problema. O **gradiente** das cores vermelho até azul foi utilizado para representar a diferença entre os máximos de cada método.
- O resultados das diferenças foram transformados para escala log_{10} devido à distribuição assimétrica.



- Dimensões com **1** até **10** restrições apresentaram mais diferenças na ordem de $[10^{-6}, 10^{-4}]$.
- Dimensões superiores a **10** restrições apresentaram dispersão aleatória de diferenças.
- A amplitude de diferenças apresentou variação $[10^{-11}, 10^{-4}]$.
- A magnitude entre a maior e a menor diferença foi da ordem de 10^{-7}.

Comparação de Máximos

- O algoritmo Simplex apresentou máximos superiores na ordem de até 10^{-4} em relação a performance do algoritmo de Pontos Iteriores.
- A maioria, ou 90% das diferenças encontradas, foram iguais ou inferiores 10^{-6}. Tal fato indica que apenas em problemas que demandam extrema precisão científica o máximo encontrado pelo algoritmo de Pontos Interiores poderia ser considerado um erro na precisão da solução.

6. Referências

- Dantzig, George Bernard, “The Simplex Method” Santa Monica, CA: RAND Corporation, 1956.
- A. Vannelli, “Teaching large-scale optimization by an interior point approach” in IEEE Transactions on Education, vol. 36, no. 1, pp. 204-209, Feb. 1993, doi: 10.1109/13.204847.
- plotly.graph.objects.Figure
- optimize.linprog-simplex
- optimize.linprog-interior-point
- seaborn.heatmap
- simplex.method.lpp