Projeto Final de Otimização: Implementação do Algoritmo Simplex

Davi dos Santos Mattos - 119133049
Pedro da Rocha Pacheco Moreira - 121130895
Esteves Emmanuel Melo Ferreira - 117209640

1. Introdução

Para o Projeto Final de Otimização, foi escolhida a OPÇÃO 2, que consiste na implementação do algoritmo simplex. A implementação foi feita em Python, utilizando o método duas fases para solucionar o Problema de Programação Linear (PPL). Os PPLs utilizando a estrutura da equação original não permite isso diretamente (como em desigualdades do tipo "≥");dos para testar o algoritmo foram retirados de livros que fazem parte da bibliografia do curso de ICP365 - Otimização 2025-1

2. Problemas

2.1)Problema exemplo de duas fases

```
Minimize z = −3x1 − 5x2
Sujeito a: x1 ≤ 4
           x2 ≤ 6
           3x1 + 2x2 ≥ 18
    x1, x2 ≥ 0
```

MARINS, Fernando Augusto Silva. Introdução à Pesquisa Operacional. São Paulo: Cultura Acadêmica / Universidade Estadual Paulista, 2011. p76

Esse é um exemplo para testar o fluxograma do método de duas fases do livro que pode retratar um modelo de custos, já que é de minimização. Temos limitações de capacidade e elas devem satisfazer uma certa demanda mínima, representada pela restrição da linha($3x_1 + 2x_2 \geq 18$).

2.2)Problema de Planejamento de Sessões de Radioterapia

A radioterapia é um método utilizado para tratar alguns casos de câncer. Na sessão são utilizadas máquinas de tratamento externas por fluxo de raio para enviar radiação ionizante em direção ao tumor danificando-o, porém tecidos saudáveis também são danificados no processo.

Em um caso particular, devem ser utilizados dois fluxos com dose no ponto de entrada de x1 e x2 kilorads, para os fluxos 1 e 2, respectivamente. A quantidade de kilorads em regiões saudáveis, deve ser minimizada e é dada pela expressão `0.4x1+0.5x2`. Para garantir a eficiência e a segurança do tratamento, algumas restrições precisam ser seguidas: tecidos críticos não devem receber mais que `2.7 krads`, com absorção dada por `0.3x1+0.1x2`, a região do tumor, deve receber em média `6 krads` e tem absorção igual a `0.5x1+0.5x2` e por último, o núcleo do tumor deve receber no mínimo `6 krads`, com absorção de `0.6x1+0.4x2.`

Com isso, o problema é modelado da seguinte forma:

```
Minimize z = 0.4x1 + 0.5x2
Sujeito a :  0.3x1 + 0.1x2 <= 2.7
             0.5 x1 + 0.5x2 = 6
             0.6 + 0.4 >= 6
xi >= 0, para i=1,2
```

HILLIER, Frederick S.; LIEBERMAN, Gerald J. Introdução à pesquisa operacional. 8. ed. São Paulo: McGraw-Hill Interamericana do Brasil, 2006. p. 43.

2.3)Problema de Planejamento Regional

Três comunas agrícolas comunitárias de uma região têm seu planejamento de plantio feito pelo Centro Técnico de Coordenação. Cada comuna possui uma quantidade limitada de terra agricultável e água para irrigação:

Comuna	Terra utilizável (em acres)	Água disponível (em acres pés)
1	400	600
2	600	800
3	300	375

Três diferentes plantações estão sendo avaliadas para o cultivo. Além de possuírem cotas máximas para o plantio, elas se diferem na quantidade de água necessária para consumo e no retorno financeiro:

Plantação	Cota máxima (em acres)	Consumo de água (Em acres pés/pés)	Retorno líquido (U$$/acre)
Beterraba	600	600	1000
Algodão	500	800	750

Sorgo	325	375	250

Além disso, as comunas precisam seguir a mesma proporção de terra utilizada, mas não necessariamente com a mesma plantação. O objetivo final é maximizar o lucro total da região. O problema é modelado da seguinte forma:

```
maximize z = 1000x1 + 1000x2 + 1000x3 + 750x4 +
             750x5 + 750x6 + 250x7 + 250x8 +250x9

Sujeito a: x1 + x4 + x7 <= 400
           x2 + x5 + x8 <= 600
           x3 + x6 + x9 <= 300

           3x1 + 2x4 + x7 <= 600
           3x2 + 2x5 + x8 <= 800
           3x3 + 2x6 + x9 <= 375

           x1 + x2 + x3 <= 600
           x4 + x5 + x6 <= 500
           x7 + x8 + x9 <= 375

           3x1 + 3x4 + 3x7 - 2x2 - 2x5 - 2x8 = 0
           x2 + x5 + x8 - 2x3 -2x6 - 2x9 = 0
           4x3 + 4x6 + 4x9 - 3x1 - 3x4 - 3x7 = 0

xi >= 0, para i = 1,2,...,9
```

```
x1, x2, x3: Acres de beterraba nas comunas 1,2 e 3, respectivamente
x4, x5, x6: Acres de algodão nas comunas 1,2 e 3, respectivamente
x7, x8, x9: Acres de sorgo nas comunas 1,2 e 3, respectivamente
```

HILLIER, Frederick S.; LIEBERMAN, Gerald J. Introdução à pesquisa operacional. 8. ed. São Paulo: McGraw-Hill Interamericana do Brasil, 2006. p. 45.

3. Experiências Computacionais

3.1) Implementação

Para a implementação do Método de Duas Fases, a linguagem de programação escolhida foi Python 3, sendo desenvolvida em um Jupyter Notebook no ambiente virtual do Google Collaboratory. As bibliotecas utilizadas para implementação foram, Regular Expression (**`re`**), Pandas (**`pandas`**), NumPy (**`numpy`**) e Time (**`time`**), esta última para cronometrar o tempo de execução do código.

O código recebe as seguintes entradas, se a função é de máximo (s - Sim, n - Não), a função objetivo (ex: “`x1+3x3`”), as n restrições (Ex: “`x1+x2+7x3=4`”, “`3x1`

`+ x3 <= 15`", "`0.6x1+0.4x2>=6`" e "`0`" para indicar que não mais restrições) e por fim se as variáveis são todas positivas (≥ 0), "s - Sim, n - Não".

`parse_funcao()`: Utilizando a biblioteca **`re`**, extrai os coeficientes da função objetivo Z e os armazena em uma lista.
`parse_restricao()`: Utilizando a biblioteca **`re`**, extrai os coeficientes das restrições e o lado direto das restrições, os armazena em três vetores (listas), uma matriz para os coeficientes, um vetor para os operadores (`"<=","">=","="`), e um vetor para o lado direito.
`montar_tableau_()`: Utilizando as variáveis que foram retornadas das duas funções anteriores, e **`numpy`**, monta uma matriz "tableau inicial" de acordo com as variáveis artificiais e de folga, caso haja variáveis artificiais é adicionado ao tableau a função objetiva artificial W.

<table>
<tr><td>Coeficientes da Restrição</td><td>Variaveis de Folga</td><td>Variáveis Artificiais</td><td rowspan="3">b</td></tr>
<tr><td colspan="3">Coeficientes da Função Objetivo</td></tr>
<tr><td colspan="3">Coeficientes da Função W (Caso tenha Var. Artificial)</td></tr>
</table>

`fase1()`: Aplica o a fase um do método de duas fases no tableau inicial, ou seja, coloca as variáveis artificiais na base, chama a função `simplex()` para executar o algoritmo simplex na matriz, quando o `simplex()` retornar o tableau ótimo (matriz), é retirado da matriz a função objetivo W e as colunas referente às variáveis artificiais, em seguida chama novamente `simplex()`, dando início a fase dois do método. Caso não haja variáveis artificiais na matriz ele pulará direto para fase dois chamando a função `simplex()` para o tableau inicial. Caso ao final da fase um o valor (b) da função artificial seja maior que zero, o programa irá imprimir na tela que o PPL é inviável, e não retorna nada.

`simplex()`: Executa recursivamente ao algoritmo simplex na matriz **`tableau_inicial`**, para otimizar a função objetiva, ao final de cada iteração, os valores dentro da matriz são arredondados a fim de evitar o erro de ponto flutuante. Ao chegar em uma solução ótima é imprimido na tela os valores das variáveis da função objetivo e o valor de Z^*. Caso ao escolher uma variável para entrar na base, não há nenhuma linha com razão positiva finita (ou seja, todas as razões são infinitas), o programa irá imprimir na tela que o Problema é ilimitado e não retorna nada.

3.2) Execução dos Problemas

3.2.1) Problema exemplo

Inserindo dados:

```
Problema é de Maximo? (s/n): n
Função Objetiva: -3x1 - 5x2
Restrição 1 (ou 0 para parar): x1 <= 4
Restrição 2 (ou 0 para parar): x2 <= 6
Restrição 3 (ou 0 para parar): 3x1 + 2x2 >= 18
Restrição 4 (ou 0 para parar): 0
Variáveis positivas? (s/n): s
```

Tableau Inicial:

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 4 \\ 0 & 1 & 0 & 1 & 0 & 0 & 6 \\ 3 & 2 & 0 & 0 & -1 & 1 & 18 \\ -3 & -5 & 0 & 0 & 0 & 0 & 0 \\ -3 & -2 & 0 & 0 & 1 & 0 & -18 \end{bmatrix}$$

Número de folgas: 3
Número de artificiais: 1
Tempo de execução do simplex na Fase 1: 0.0012002s

Tableau ótimo:

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 4 \\ 0 & 0 & 3 & 2 & 1 & 6 \\ 0 & 1 & 0 & 1 & 0 & 6 \\ 0 & 0 & 3 & 5 & 0 & 42 \end{bmatrix}$$

Tempos de execução:
Tempo do parse: 8.63075e-05 segundos
Tempo do método de duas fases: 0.00357556 segundos
Tempo total: 0.00445223 segundos

Resultados:
z=(4.0, 6.0) Z* = -42.0

3.2.2) Problema de Planejamento de Sessões de Radioterapia

Inserindo os dados:

```
Problema é de Maximo? (s/n): n
Função Objetiva: 0.4x1 + 0.5x2
Restrição 1 (ou 0 para parar): 0.3x1 + 0.1x2 <= 2.7
Restrição 2 (ou 0 para parar): 0.5x1 + 0.5x2 = 6
Restrição 3 (ou 0 para parar): 0.6x1 +0.4x2 >= 6
Restrição 4 (ou 0 para parar): 0
Variáveis positivas? (s/n): s
```

Tableau Inicial:

$$\begin{bmatrix} 0.3 & 0.1 & 1.0 & 0.0 & 0.0 & 0.0 & 2.7 \\ 0.5 & 0.5 & 0.0 & 0.0 & 1.0 & 0.0 & 6.0 \\ 0.6 & 0.4 & 0.0 & -1.0 & 0.0 & 1.0 & 6.0 \\ 0.4 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -1.1 & -0.9 & 0.0 & 1.0 & 0.0 & 0.0 & -12.0 \end{bmatrix}$$

Número de folgas: 2
Número de artificiais: 2
Tempo de execução do simplex na Fase 1: 0.00161767 segundos

Tableau ótimo:

$$\begin{bmatrix} 1.0 & 0.0 & 5.000001 & 0.0 & 7.499999 \\ 0.0 & 0.0 & 0.999999 & 1.0 & 0.300001 \\ 0.0 & 1.0 & -5.000007 & 0.0 & 4.500006 \\ 0.0 & 0.0 & 0.500004 & 0.0 & -5.250003 \end{bmatrix}$$

Tempos de execução:

Tempo do parse: 0.000171185 segundos
Tempo do método de duas fases: 0.00336123 segundos
Tempo total: 0.00458503 segundos
Número de iterações: 4

Resultados:

z=(7.5, 4.5)
Z* = 5.25

3.2.3)Problema de Planejamento Regional

Inserindo os dados:

```
Problema é de Maximo? (s/n): s
Função Objetiva: 1000x1 + 1000x2 + 1000x3 + 750x4 + 750x5 + 750x6 + 250x7 + 250x8 +250x9
Restrição 1 (ou 0 para parar): x1 + x4 + x7 <= 400
Restrição 2 (ou 0 para parar): x2 + x5 + x8 <= 600
Restrição 3 (ou 0 para parar): x3 + x6 + x9 <= 300
Restrição 4 (ou 0 para parar): 3x1 + 2x4 + x7 <= 600
Restrição 5 (ou 0 para parar): 3x2 + 2x5 + x8 <= 800
Restrição 6 (ou 0 para parar): 3x3 + 2x6 + x9 <= 375
Restrição 7 (ou 0 para parar): x1 + x2 + x3 <= 600
Restrição 8 (ou 0 para parar): x4 + x5 + x6 <= 500
Restrição 9 (ou 0 para parar): x7 + x8 + x9 <= 375
Restrição 10 (ou 0 para parar): 3x1 + 3x4 + 3x7 - 2x2 - 2x5 - 2x8 = 0
Restrição 11 (ou 0 para parar): x2 + x5 + x8 - 2x3 -2x6 - 2x9 = 0
Restrição 12 (ou 0 para parar): 4x3 + 4x6 + 4x9 - 3x1 - 3x4 - 3x7 = 0
Restrição 13 (ou 0 para parar): 0
Variáveis positivas? (s/n): s
```

Neste problema, o tableau inicial ficou extremamente grande com 14 linhas (12 restrições + objetivo + objetivo fase 1) e 22 colunas (9 variáveis de decisão + 9 variáveis de folga + 3 variáveis artificiais + sbv), logo a sua representação ficaria muito extensa e confusa, por isso foi optado por mostrar apenas suas informações.

Tempos de execução:

Tempo do parse: 0.000179529 segundos
Tempo de execução do simplex na Fase 1: 0.00469875s
Tempo do método de duas fases: 0.0191104 segundos
Tempo total: 0.0208688 segundos
Número de iterações: 8

Resultados:

z=(133.333, 100.0, 25.0, 100.0, 250.0, 150.0, 0.0, 0.0, 0.0)
Z* = 633333.309

4. Conclusão

A implementação do algoritmo Simplex (no caso, de duas fases) se mostrou bastante eficaz na resolução de problemas tanto de maximização quanto de minimização, como podemos observar pelos tempos de execução extremamente baixos. Isso demonstra, além de versatilidade, uma boa performance e eficiência do método.

Os problemas utilizados são clássicos da literatura, sendo o segundo e o terceiro retirados do livro base da disciplina — uma referência consolidada na área — e o primeiro, um exemplo didático amplamente utilizado de um livro dos anos 80.

Uma parte importante do desenvolvimento do algoritmo foi compreender como ele lida com a introdução de variáveis de folga, excesso e artificiais. Corrigir os problemas que surgiram durante esse processo contribuiu significativamente para fixar o conteúdo estudado, proporcionando uma compreensão mais profunda dos conceitos envolvidos.

Vale destacar também alguns desafios enfrentados ao longo do trabalho. Um deles foi a dificuldade de captar corretamente os inputs das equações e inequações, especialmente ao copiá-las automaticamente — o que ainda pode gerar erros. Por esse motivo, optamos por inseri-las manualmente para garantir maior precisão. Enfrentamos ainda problemas com sinais dos números durante os cálculos do tableau, que foram posteriormente solucionados, assim como uma falha na integração entre a primeira e a segunda fase do método, que causava erros quando a Fase 1 não precisava ser executada. Outro ponto importante que vale ser destacado é que mesmo perguntando se as variáveis são positivas ou não, no fim das contas o algoritmo trata sempre como se fosse positivo, mesmo se elas forem livres.

Outro ponto que vale ressaltar foi a utilização do Google Colab que ajudou na colaboração, depuração e mudanças no código corrigindo os erros e adaptando para melhor compreensão e boas práticas. Em suma, o projeto do trabalho concluiu com êxito o objetivo de reforçar os conteúdos aplicados durante o semestre, mostrando um modelo real de aplicação e como funciona na prática traduzir problemas matemáticos e solucioná-los.

5. Referências Bibliográficas

SENNE, Edson Luiz França. O Método Simplex. Disponível em: https://www.feg.unesp.br/Home/PaginasPessoais/profedsonluizfrancasenne/3-o-metodo-simplex.pdf. Acesso em: 21 jun. 2025.

HILLIER, Frederick S.; LIEBERMAN, Gerald J. Introdução à Pesquisa Operacional. 8. ed. São Paulo: AMGH, 2013.

Slides de Aula - Profª Maria Helena Cautiero Horta Jardim

Lista de Exercícios 6 - Profª Maria Helena Cautiero Horta Jardim

6. Apêndice

APÊNDICE A - Link de redirecionamento para o google collaboratory, contendo todo o código fonte

Método de Duas Fase + Simplex

APÊNDICE B - Imagem do programa executando o Problema 2.1

```
28s
Restrição 2 (ou 0 para parar): x2<=6
Restrição 3 (ou 0 para parar): 3x1+2x2>=18"""
metodo_duas_fases()

Problema é de Maximo? (s/n): n
Função Objetiva: -3x1-5x2
Restrição 1 (ou 0 para parar): x1<=4
Restrição 2 (ou 0 para parar): x2<=6
Restrição 3 (ou 0 para parar): 3x1+2x2>=18
Restrição 4 (ou 0 para parar): 0
Variáveis positivas? (s/n): s

Tableau Simplex Inicial:
[[ 1.  0.  1.  0.  0.  0.  4.]
 [ 0.  1.  0.  1.  0.  0.  6.]
 [ 3.  2.  0.  0. -1.  1. 18.]
 [-3. -5.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  1.  0.]]
Número de restrições: 3
Número de folgas: 3
Número de artificiais: 1
```

APÊNDICE C - Imagem dos resultado obtido para o Problema 2.1

```
Tableau Ótimo:
[[ 1.        0.        1.        0.        0.        4.       ]
 [ 0.        0.        3.000003  2.000003  1.        6.000006]
 [ 0.        1.        0.        1.        0.        6.       ]
 [ 0.        0.        3.000003  5.000003  0.       42.000006]]
Variáveis na Base: [1, 5, 2]

z=(4.0, 6.0)
Z* = -42.0

Tempo de execução do simplex na Fase 2: 0.00106311s

Tempo total de execução do simplex: 0.0025351s
==========Tempo de execução==========
Tempo do parse: 0.000132322 segundos
Tempo do método de duas fases: 0.00345469 segundos
Tempo total: 0.00422311 segundos
```

APÊNDICE D - Imagem do programa executando o Problema 2.2

```
metodo_duas_fases()

Problema é de Maximo? (s/n): n
Função Objetiva: 0.4x1+0.5x2
Restrição 1 (ou 0 para parar): 0.3x1+0.1x2<=2.7
Restrição 2 (ou 0 para parar): 0.5x1+0.5x2=6
Restrição 3 (ou 0 para parar): 0.6x1+0.4x2>=6
Restrição 4 (ou 0 para parar): 0
Variáveis positivas? (s/n): s

Tableau Simplex Inicial:
[[ 0.3  0.1  1.   0.   0.   0.   2.7]
 [ 0.5  0.5  0.   0.   1.   0.   6. ]
 [ 0.6  0.4  0.  -1.   0.   1.   6. ]
 [ 0.4  0.5  0.   0.   0.   0.   0. ]
 [ 0.   0.   0.   0.   1.   1.   0. ]]
Número de restrições: 3
Número de folgas: 2
Número de artificiais: 2
```

APÊNDICE E - Imagem dos resultado obtido para o Problema 2.2

```
Tableau Ótimo:
[[ 1.        0.        5.000001  0.        7.499999]
 [ 0.        0.        0.999999  1.        0.300001]
 [ 0.        1.       -5.000007  0.        4.500006]
 [ 0.        0.        0.500004  0.       -5.250003]]
Variáveis na Base: [1, 4, 2]

z=(7.5, 4.5)
Z* = 5.25

Tempo de execução do simplex na Fase 2: 0.000857115s

Tempo total de execução do simplex: 0.00278735s
==========Tempo de execução==========
Tempo do parse: 6.81877e-05 segundos
Tempo do método de duas fases: 0.00331569 segundos
Tempo total: 0.00389171 segundos
```

APÊNDICE F - Imagem do programa executando o Problema 2.3

```
Problema é de Maximo? (s/n): s
Função Objetiva: 1000x1 + 1000x2 + 1000x3 + 750x4 + 750x5 + 750x6 + 250x7 + 250x8 +250x9
Restrição 1 (ou 0 para parar): x1 + x4 + x7 <= 400
Restrição 2 (ou 0 para parar): x2 + x5 + x8 <= 600
Restrição 3 (ou 0 para parar): x3 + x6 + x9 <= 300
Restrição 4 (ou 0 para parar): 3x1 + 2x4 + x7 <= 600
Restrição 5 (ou 0 para parar): 3x2 + 2x5 + x8 <= 800
Restrição 6 (ou 0 para parar): 3x3 + 2x6 + x9 <= 375
Restrição 7 (ou 0 para parar): x1 + x2 + x3 <= 600
Restrição 8 (ou 0 para parar): x4 + x5 + x6 <= 500
Restrição 9 (ou 0 para parar): x7 + x8 + x9 <= 375
Restrição 10 (ou 0 para parar): 3x1 + 3x4 + 3x7 - 2x2 - 2x5 - 2x8 = 0
Restrição 11 (ou 0 para parar): x2 + x5 + x8 - 2x3 -2x6 - 2x9 = 0
Restrição 12 (ou 0 para parar): 4x3 + 4x6 + 4x9 - 3x1 - 3x4 - 3x7 = 0
Restrição 13 (ou 0 para parar): 0
Variáveis positivas? (s/n): s

Tableau Simplex Inicial:
[[    1.     0.     0.     1.     0.     0.     1.     0.     0.     1.
      0.     0.     0.     0.     0.     0.     0.     0.     0.     0.
      0.   400.]
 [    0.     1.     0.     0.     1.     0.     0.     1.     0.     0.
      1.     0.     0.     0.     0.     0.     0.     0.     0.     0.
      0.   600.]
 [    0.     0.     1.     0.     0.     1.     0.     0.     1.     0.
      0.     1.     0.     0.     0.     0.     0.     0.     0.     0.
      0.   300.]
```

APÊNDICE G - Imagem dos resultado obtido para o Problema 2.3

```
z=(133.333, 100.0, 25.0, 100.0, 250.0, 150.0, 0.0, 0.0, 0.0)
Z* = 633333.309

Tempo de execução do simplex na Fase 2: 0.0103734s

Tempo total de execução do simplex: 0.0157003s
=========Tempo de execução=========
Tempo do parse: 0.000167131 segundos
Tempo do método de duas fases: 0.0183709 segundos
Tempo total: 0.0211565 segundos
```