SME0211 - Otimização Linear
Segundo semestre de 2024

Trabalho final

Katlyn Ribeiro Almeida – 14586070
Ian de Holanda Cavalcanti Bezerra – 13835412
Julia Graziosi Ortiz – 11797810
Cody Stefano Barham Setti – 4856322
Matheus Araujo Pinheiro – 14676810

November 21, 2024

1 Escolha de ferramentas

Para o desenvolvimento deste projeto, optamos por utilizar a linguagem de programação Python, com ênfase especial nos notebooks Jupyter. Esta escolha foi motivada por diversas razões:

- **Facilidade na implementação de algoritmos iterativos:** Os notebooks Jupyter oferecem um ambiente interativo que é particularmente adequado para a implementação e teste de algoritmos que requerem múltiplas iterações.

- **Ambiente de execução flexível:** No ambiente do notebook, podemos inicializar variáveis e realizar operações sobre elas sem a necessidade de reinicializá-las a cada execução. Isso proporciona uma grande flexibilidade no desenvolvimento e depuração do código.

- **Visualização integrada:** Os notebooks Jupyter permitem a integração de código, resultados e visualizações, facilitando a análise e apresentação dos resultados obtidos.

- **Simplicidade e eficiência:** Python oferece uma sintaxe clara e intuitiva, facilitando a implementação de algoritmos complexos. Além disso, suas bibliotecas, como NumPy e SciPy, fornecem funções otimizadas para a solução de sistemas lineares, permitindo uma implementação eficiente e concisa do método Simplex.

2 Otimização/Programação Linear

A otimização linear é uma técnica matemática que busca encontrar o valor máximo ou mínimo de uma função linear, sujeita a um conjunto de restrições lineares. Este problema é representado por uma função objetivo linear e um conjunto de desigualdades lineares que limitam as soluções possíveis.

Esses problemas são amplamente utilizados em várias áreas, como economia, logística, produção e finanças, para maximizar lucros, minimizar custos ou otimizar a utilização de recursos.

Para facilitar e unificar as formas de solução desses problemas, buscamos representá-los na forma padrão, resolvendo um sistema de minimização sujeito a restrições de igualdades. Essa forma padrão é frequentemente obtida através da adição de variáveis de folga nas desigualdades originais.

A forma padrão de um problema de programação linear pode ser expressa da seguinte maneira:

$$
\begin{aligned}
\text{Minimizar } & c^T x \\
\text{Sujeito a } & Ax = b \\
& x \geq 0
\end{aligned}
$$

Onde:

- x é o vetor de variáveis de decisão
- c é o vetor de coeficientes da função objetivo
- A é a matriz de coeficientes das restrições
- b é o vetor de termos independentes das restrições

Esta representação padronizada permite unificar as aplicaçoes de métodos de solução, como o algoritmo Simplex, a uma ampla variedade de problemas de otimização linear, independentemente de sua formulação original.

3 O Algoritmo Simplex

O algoritmo Simplex é um método iterativo para resolver problemas de programação linear. Desenvolvido por George Dantzig em 1947, ele se tornou um dos algoritmos mais importantes e amplamente utilizados na otimização linear.

O método funciona percorrendo os vértices do poliedro formado pelas restrições do problema, movendo-se de um vértice a outro de forma a melhorar progressivamente o valor da função objetivo até encontrar a solução ótima. O algoritmo é dividido em duas fases principais:

3.1 Fase I - Encontrando uma Solução Básica Viável

A primeira fase do Simplex tem como objetivo encontrar um ponto inicial viável para começar a otimização. Isso é necessário porque nem sempre é trivial encontrar uma solução que satisfaça todas as restrições do problema. Nesta fase:

1. Cria-se um problema auxiliar introduzindo variáveis artificiais
2. Minimiza-se a soma das variáveis artificiais
3. Se o valor mínimo for zero, uma solução básica viável foi encontrada
4. Se o valor mínimo for positivo, o problema original não tem solução viável

3.2 Fase II - Otimização

Uma vez encontrada uma solução básica viável, a Fase II busca a solução ótima do problema original. Em cada iteração desta fase, o algoritmo:

1. Verifica se a solução atual é ótima através do critério de otimalidade
2. Se não for ótima, identifica uma variável para entrar na base (critério de entrada)
3. Determina qual variável deve sair da base (critério de saída)
4. Atualiza a solução movendo-se para um novo vértice através de operações de pivoteamento

O processo continua até que uma das seguintes condições seja alcançada:

- Uma solução ótima é encontrada
- Detecta-se que o problema é ilimitado

Uma característica fundamental do Simplex é que ele garante encontrar a solução ótima em um número finito de passos, desde que ela exista, embora no pior caso o número de iterações possa crescer exponencialmente com o tamanho do problema.

3.3 O Tableau do Simplex

O Tableau do Simplex é uma representação matricial que organiza todas as informações necessárias para executar o algoritmo de forma sistemática. Esta estrutura tabular inclui:

- Os coeficientes das restrições
- Os coeficientes da função objetivo
- As variáveis básicas e não-básicas

- Os valores das variáveis na solução atual
- Os termos independentes

A forma geral do Tableau é:

Base	x_1	x_2	$\cdots$	x_n	RHS
z	c_1	c_2	$\cdots$	c_n	z_0
x_{B1}	a_{11}	a_{12}	$\cdots$	a_{1n}	b_1
x_{B2}	a_{21}	a_{22}	$\cdots$	a_{2n}	b_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
x_{Bm}	a_{m1}	a_{m2}	$\cdots$	a_{mn}	b_m

Onde RHS (Right Hand Side) representa os termos independentes, e a última linha contém os coeficientes da função objetivo e o valor atual da função objetivo (z_0). O Tableau é atualizado a cada iteração do algoritmo através de operações de pivoteamento, facilitando o acompanhamento do progresso da otimização e a identificação da solução ótima.

3.4 Implementação do Algoritmo

Para implementar o algoritmo Simplex, desenvolvemos um código em Python que segue a estrutura descrita anteriormente. A implementação foi organizada em funções modulares para facilitar a manutenção e compreensão do código:

- **Inicialização:** O código começa configurando o problema na forma padrão, construindo o Tableau inicial com as variáveis de folga necessárias.
- **Fase I:** Implementamos o método das duas fases, onde primeiro buscamos uma solução básica viável inicial através da adição de variáveis artificiais.
- **Fase II:** Uma vez encontrada uma solução viável, o algoritmo procede com a otimização, realizando iterações até encontrar a solução ótima ou detectar ilimitabilidade.
- **Funções auxiliares:** Desenvolvemos funções para:
  - Identificar a variável de entrada (maior coeficiente negativo)
  - Determinar a variável de saída (razão mínima)
  - Realizar operações de pivoteamento
  - Verificar critérios de parada

A implementação faz uso extensivo das bibliotecas NumPy para operações matriciais eficientes e Pandas para manipulação e visualização do Tableau, permitindo uma execução rápida mesmo para problemas de dimensões consideráveis.

3.5 Código

```python
def get_solution_simplex(c, A, b):
    nvar = len(A[0])
    d = (False, nvar)
    c1, A1, b1, d = fase1(c, A, b)

    # Caso seja necessario usar variaveis artificiais(Faze 1)
    if d[0]:
        print("Simplex FASE 1: ")
        A2, b2 = simplex(c1,A1,b1,d)
        if A2 is None:
            print("Problema infactivel")
            return None
        else:
            print("Resultado da fase 1: ",b2)


        # Convertendo tudo para mesmo formato que o simplex(Faze 2)
        A2 = [[float(x) for x in row] for row in A2]
        b2 = [float(x) for x in b2]

        fA, fb = simplex(c, A2, b2,(False, nvar), True)
        print("Resultado da fase 2: ",fb)
    else:  # N o foi necessario usar a fase 1(Resolve direto Faze 2)
        print("Nao Foi nessesario usar a fase 1!!")
        fA, fb = simplex(c, A1, b1, d)
        print("Resultado da fase 2: ", fb)
```

A função `get_solution_simplex` é responsável por coordenar a execução do método Simplex de duas fases. Ela recebe como parâmetros:

- `c`: vetor de coeficientes da função objetivo
- `A`: matriz de coeficientes das restrições
- `b`: vetor de termos independentes

A função primeiro verifica se é necessário executar a Fase I do Simplex (quando não temos uma solução básica viável inicial). Se for necessário:

1. Executa a Fase I para encontrar uma solução básica viável inicial
2. Verifica se o problema é factível
3. Se factível, usa a solução da Fase I como ponto de partida para a Fase II

Caso não seja necessária a Fase I, a função executa diretamente a Fase II do Simplex para encontrar a solução ótima.

A função retorna `None` se o problema for infactível, ou a solução ótima caso contrário.